

Perfect is the Enemy of Good: Lloyd-Max Quantization for Rate Allocation in Congestion Control Plane

Shiva Ketabi, Yashar Ganjali
Department of Computer Science, University of Toronto

Abstract—Decoupling congestion control plane from datapath can expedite the development of new congestion control solutions. It also creates opportunities for explicit rate allocation schemes. Dealing with large numbers of flows remains a major challenge. Max-min fairness – the gold standard for flow rate allocation – has a running complexity proportional to the number of flows, which might be prohibitive in large-scale networks.

To accelerate explicit rate allocation, we present solutions using rate quantization, *i.e.* mapping the continuous range of flow rates to a small number of bins. We use Lloyd-Max, a quantization method that generates bins according to the distribution of flow rates, to dynamically adjust the quantization bins over time. Our experimental evaluation shows that the distortion caused by this quantization scheme is small, and can be negligible compared to intrinsic errors in measuring and enforcing rates in current solutions.

We also show that rate quantization can significantly speed up max-min fair rate allocation, reducing the run-time by 70 – 95%. Besides, Lloyd-Max quantization using recent history of flow rates performs close to the case when we have access to the exact current (or future) rates. This is an interesting observation as it obviates the need for complex techniques that try to predict future rates.

I. INTRODUCTION

Following the idea of control/datapath decoupling in Software-Defined Networking (SDN), a paradigm shift is happening in congestion control architectures. Traditionally, congestion control schemes are end-to-end solutions, intertwined with the datapath (*e.g.* TCP's Linux kernel module). Some recent proposals (*e.g.* [1]–[3]) suggest decoupling the congestion control algorithm from datapath elements, creating a separate Congestion Control Plane (CCP).

Congestion Control Agents (CCAs) running on end-hosts periodically share congestion control signals (*e.g.* round-trip times, losses, ECN, etc.) with the congestion control plane. At the heart of congestion control plane is the Congestion Control Command Center (C4), the equivalent of a network controller in SDN control plane. C4 runs congestion control algorithms and sends Congestion Updates (CUEs) back to CCAs, which in turn control data transmission based on the policies specified by C4 (Figure 1).

C4 can run on any server in general. In the context of SDN, co-locating C4 with the network controller (or as a network application running on the network controller) gives it access to control plane information including flow statistics and routing information. This makes C4 extremely powerful: it

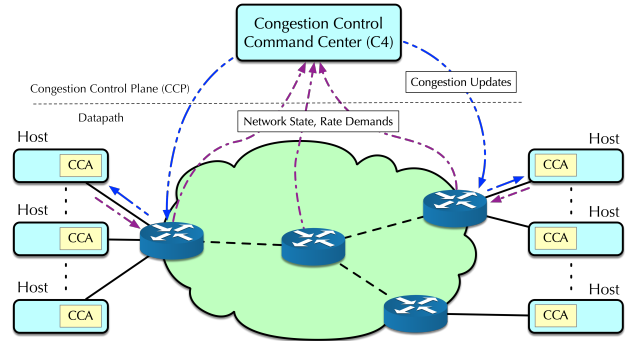


Fig. 1. Separation of congestion control plane and datapath. Congestion control command center can be used for coordinating transmissions.

can identify congestion points in the network, take the global bandwidth demands into account, and therefore, allocate rates to flows in a way that no end-to-end congestion control mechanism can.

New Possibilities. Decoupling CCP from datapath can expedite the development of new congestion control schemes (especially in data centres) in the same way decoupling control and datapath in SDN allows faster development of network control applications. Also, once a new congestion control algorithm is developed, it can be readily used in any network that supports the concept of CCP decoupling without the need to modify all end-host implementations.

More importantly, separation of CCP from datapath makes *aggregate control of multiple flows* possible [4], [5]. Assuming C4 is (logically) centralized, congestion control algorithms can take into account the state of other flows, essentially solving one of the biggest challenges in traditional congestion control: coordination among flows.

In most end-to-end congestion control protocols today, individual flows continuously probe the network, adjusting their injection rates based on the (often implicit) feedback received from the network. Usually, the objective is to achieve the maximum rate without violating fairness, also referred to as *max-min fairness* [6]. This is a very challenging problem given the limited nature of network feedback (usually few bits measured based on loss or delay), intrinsic randomness in packet drops and queuing delays, lack of direct coordination among flows, and the delay between network events and sources being notified about those events.

A global coordination mechanism, implemented as part of

C4, paves the path for optimal direct rate allocation. It eliminates the need for end-hosts to probe/guess the appropriate packet injection rates by estimating available bandwidth, and the behavior of flows in the network (Figure 1).

Moreover, decoupling CCP from datapath would make it easier to push the congestion control stack from software to hardware. A congestion control agent can be implemented as a simple state machine residing in the network interface card of an end-host. This state machine can adjust transmission rates based on the signals received from C4 without the need to have complex congestion control algorithms run on the local machine. This can significantly reduce the load on end-hosts, especially in servers with a significant number of flows.

Challenges. Using a logically centralized congestion control plane for coordination among different flows comes with its own challenges. First, given the sheer number of flows, any explicit coordination and rate allocation mechanism needs to be extremely fast. Max-min fairness, the golden standard for rate allocation, for example, has a running complexity of $O(N)$ with N flows in the network. This can be prohibitively large in any backbone or data center network carrying hundreds of millions of flows at each point of time.

Second, frequently collecting rate demands from, and sending updates to congestion control agents can be very challenging. Since we have pushed the control loop of congestion control out of the datapath, extremely fast response times are essential.

Finally, in case of enforcing rates in hardware accelerated agents in end-hosts, we might have significant limitations on available memory. Even assuming a simple state machine acting as a congestion control agent (changing state based on network and CCP signals) we need to ensure the amount of state is kept to a minimum.

Rate Quantization. In this paper, we propose rate quantization as a simple way to address these challenges. The idea is based on the observation that max-min fair rate allocation has a run-time complexity proportional to the number of *unique* flow demands. In the worst case scenario, each flow has a unique demand and therefore, the run-time complexity is proportional to the number of flows in the network, which can be prohibitively large.

However, using quantization we can simply reduce the number of unique rate demands. Max-min fair allocation schemes can be modified to deal with these groups of flows rather than individual flows, thus significantly reducing the run-time of max-min fair rate allocation.

Rate quantization has been used for accelerating network algorithms in the past [7], [8]. The downside of rate quantization is the error it introduces. Clearly, the choice of the quantization method can have a significant impact. We show that using appropriate quantization schemes, a reasonable balance between accuracy and the run-time of max-min fair rate allocation can be found. We also note that all congestion control schemes today, have an intrinsic error in estimating flow rates, and network available bandwidth. Therefore, a 100% accurate solution would be impractical in the first place,

making a solution with reasonable error levels acceptable in practice.

In addition to reducing the run-time of max-min fair rate allocation, rate quantization can significantly reduce the overhead of data collection. Congestion control agents need to send updates to the CCP when their demand rates change. Rate quantization reduces this overhead by *i)* reducing the number of bits required as we are dealing with limited number of bins; and *ii)* reducing the frequency of such updates as the congestion control agents need to send updates when the change in rate demand is big enough to push it to a new quantization bin.

Contributions. In this paper, we take a closer look at rate allocation in data centre networks where congestion control plane is decoupled from datapath. We show that rate quantization is a simple way to reduce the run-time complexity of max-min fair rate allocation. Using a simple, yet powerful quantization scheme called Lloyd-Max Quantization [9], [10], we show that we can reduce the run-time of max-min fair rate allocation by 70 – 95%, with acceptable error rates due to quantization. In particular, using traces from real networks, we show that the 99th percentile of the distortion of the max-min rate resulting from quantization is smaller than the intrinsic errors in many network measurements, and thus the accuracy of current congestion control schemes.

We also show that rate quantization can obviate the need for complex future rate predictions techniques. More specifically, the Lloyd-Max quantization using recent history of flow rates performs close to the case when we have access to the exact current (or even future) rates.

The rest of the paper is organized as follows. We start by formally defining the problem in Section II, along with several observations that lead to our solution. Section III presents our rate quantization schemes, followed by evaluations in Section IV.

II. PROBLEM STATEMENT

In this section, we formalize the problem and show how rate quantization can impact the run-time of max-min fair rate allocation.

A. Max-min Fairness Run-time

Let us consider N flows $(f_1, f_2, \dots, f_N)$ sharing a bottleneck link with capacity C . Let d_i represent the demand for flow i ($1 \leq i \leq N$). Let us assume the congestion control plane collects these flow statistics from flow sources. A rate allocation $S = (s_1, s_2, \dots, s_N)$, where s_i is the rate assigned to flow f_i ($1 \leq i \leq N$), is called *max-min fair* if:

- The allocation is feasible, *i.e.*, $\sum_{i=1}^N s_i \leq C$, for the bottleneck link capacity C .
- The share of each flow is less than or equal to its demand, *i.e.*, $s_i \leq d_i$.
- Any increase in the share of any flow results in a decrease in the rate assigned to another flow with a smaller share.

It is straightforward to show that max-min fair allocation is unique. The following algorithm (or some variations of it) is

how we typically find a max-min fair allocation. We assume the demands are sorted in a non-decreasing manner¹, *i.e.* $d_i \leq d_{i+1}$ for all i , $1 \leq i \leq N$.

- 1) Set the unallocated capacity C_u equal to C , and $i = 1$.
- 2) If i is equal to $N + 1$, terminate.
- 3) If $d_i < C_u/(N - i + 1)$ set $s_i = d_i$, otherwise, set $s_i = C_u/(N - i + 1)$. In other words, if a flow is asking for less than its fair share, allocate all of its demand; otherwise, allocate the fair share.
- 4) Set $C_u = C_u - s_i$, and $i = i + 1$, and go back to Step 2.

As soon as a flow demand is more than its fair share, the algorithm can terminate by allocating the fair share to all the remaining flows. This is due to the assumption that demands are sorted in a non-decreasing manner. The following example shows a worst-case scenario for which the algorithm terminates after exactly N iterations.

Example. Let us assume $d_i = i$ for all i , $1 \leq i \leq N$, and $C = N(N + 1)/2$. As we go through the flows, each flow i is requesting less than its fair share (of the remaining capacity), and the algorithm will terminate only after N iterations.

This run-time complexity can be prohibitive in many online applications of max-min fairness. The problem becomes even more significant if we notice that any increase in d_i can lead to a complete update of the allocated rates to flows f_i to f_N . Therefore, any incremental algorithm which deals with demand changes over time, still has a complexity of $O(N)$ at each point of time.

Grouping Flows. Consider a scenario that the flow demands can be divided into K groups, $g_1, g_2, \dots, g_K$, where all flows in group g_i have the same demand d_i . It is easy to see that a simple modified version of the max-min fairness algorithm that deals with groups rather than individual flows has a complexity of $O(K)$. If it happens that we naturally have $K \ll N$ in a network, this can be a significant reduction in run-time. Even if this is not the case for a certain set of demands, we still have the option to quantize demands into a small number of groups. The question is whether the error resulting from this quantization is acceptable. If so, we have a reasonable trade-off for reducing the run-time complexity of max-min fair rate allocation.

Multiple Bottlenecks Scenario. Let us relax the single bottleneck assumption and assume there are B bottlenecks in the network. First, we note that a trivial generalization of the single-bottleneck solution does not work here. Any change to the demand/rate of a given flow f impacts all bottlenecks that f passes through, and the rate allocation of all flows sharing these bottlenecks. These flows in turn can impact other flows in the network, creating a cascading effect.

In the literature, there are several proposals to solve this problem using techniques ranging from linear programming [11]–[16] to distributed k-water filling [17], [18]. Interestingly, the same idea of grouping flows can be applied for the multiple-bottleneck scenario, *i.e.*, if we can group flows

¹We ignore the cost of sorting here. In an incremental solution, where demands are updated one by one, we need to sort only once and the amortized cost of maintaining the order would be negligible.

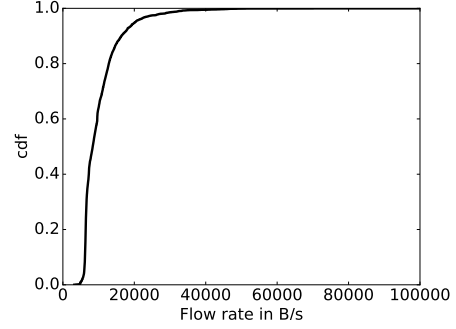


Fig. 2. CDF of 99th percentile of flow rates over time.

into K groups (with equal demand/rate) the run-time of the algorithm will be proportional to $O(K)$ instead of $O(N)$.

B. Observations

Before introducing our rate quantization solution, let us start with three observations.

Observation 1: Measuring flow rates and available bandwidth in a network is intrinsically inaccurate. Any rate allocation scheme relies on some type of direct or indirect measurement of flow demands and available bandwidth that are inevitably inaccurate. For instance, TCP packet transmission rates depend on estimated round-trip times and packet drop rates which are known to be susceptible to measurement errors.

Even if (hypothetically) the application layer is expected to provide the value of demands to the transmission layer, the value would not be completely precise as most applications do not have the ability to predict future demands precisely. Therefore, if a slight modification of the flow demands leads to considerable improvements in run-time, such compromises might be acceptable as long as the resulted skew is bounded.

Observation 2: Flow rates are bounded. Maximum flow rates in a network are bounded by link capacities. Moreover, depending on the number of flows sharing a given link, the maximum rate of a given flow can be much smaller in practice. In fact, having more flows (which is the scenario we are interested here, as it increases the run-time) sharing a link means that the expected rate of each flow will be smaller.

To illustrate this observation, we used real network traces (from a well-known study of data center traffic characteristics by Benson et al. [19]), and measured flow rates over time. We divided the time to 10 seconds time intervals, and for each time interval calculated the value of the 99th percentile of flow rates. Figure 2 shows the cumulative distribution function (CDF) of these 99th percentile values. As the figure suggests, almost all the flow rates are less than 100 KB/s, which is much smaller than typical link capacities in data centers (which is in the order of tens of Gb/s).

Observation 3: Flow rates have higher density around certain regions. Typically, flow rates in a network are not uniformly distributed. In fact, depending on the network topology, link capacities, and the type of applications generating the traffic, the flow distribution might be highly dense around

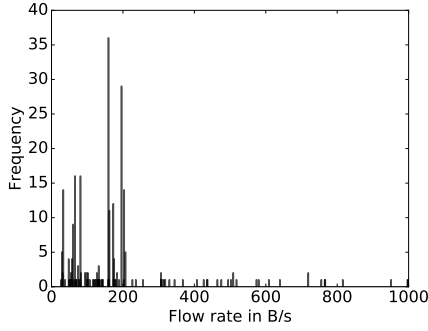


Fig. 3. Histogram of flow rates in a sample interval.

some bounded number of values and sparse in many other areas of the distribution plot.

To illustrate this point, we randomly selected a fixed interval of 10 seconds from the dataset in [19]. Figure 3 shows the empirical distribution of flow rates during this interval. As we can see, the majority of bins in this graph have no or very small number of flows.

Lessons. Observation 1 suggests that we cannot estimate the exact value of many network variables including the flow rates. In other words, any distortion in the order of a few KB/s is negligible compared to the intrinsic errors of the problem. Observations 2 and 3 suggest that flows are not uniformly distributed. We have highly dense regions as well as a large area of sparse rates.

We note that flow rates in dense regions do not necessarily have the *exact* same value. However, if we quantize the rate values, we can significantly reduce the number of *unique* rates by slightly distorting rate values. As motivated in Section II-A, this reduction in the number of unique rates, can reduce the run-time of the max-min fairness algorithm.

III. RATE QUANTIZATION

Based on the three observations described in Section II-B, our goal is to convert a set of flow rates to groups of equal rates. More specifically, we map the continuous range of possible flow rates to a finite number of bins. Each bin is associated with a representative and two boundaries. The distance of each flow rate from its bin representative is the *quantization distortion* for that flow. Let us denote the total number of bins with K , and use a_i, b_i , and m_i to represent the left and right boundaries, and the representative value of the i -th bin respectively.

The quantization algorithm takes as input a given demand rate r and maps it to its corresponding bin representative: $Q(r) = m_i$ if $r \in [a_i, b_i]$. As previously discussed, quantization can reduce the run-time of rate allocation algorithms to the number of bins, $O(K)$, instead of the number of flows, $O(N)$.

We assume time is divided into intervals of size T_1 seconds. At the beginning of each time interval, specific bins are constructed based on demand rates measured during the previous time interval. We note that the frequency of running max-min fairness algorithm is (significantly) higher than our

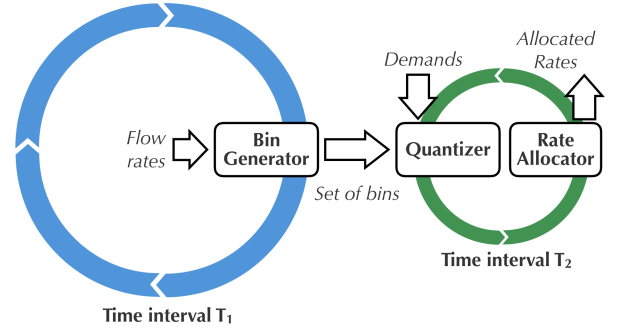


Fig. 4. High-level overview of our proposed solution.

quantization scheme, *i.e.* during each interval T_1 we run max-min fairness several times. All these runs use the bins selected at the beginning of the time interval T_1 based on rates from previous time interval. Therefore, although constructing the quantization bins requires $O(N)$ run-time, it is a one-time operation during the whole interval T_1 . Our goal is to reduce the cost of rate allocation which happens at a much higher frequency here.

Figure 4 shows a high-level overview of our proposed solution for accelerating rate allocation algorithms. The Bin Generator block receives the distribution of flows every T_1 seconds and uses the Lloyd-Max quantization scheme (described in the following section) to generate the quantization bins and representatives. There is an overhead for collecting flow distributions from previous intervals, but that is intrinsic to any centralized max-min fairness solution.²

The Quantizer block uses the output of the Bin Generator block to trim the demand rates and map the input rates to a limited set of quantized values every T_2 seconds. Here, we assume that $T_2 < T_1$. The quantized values are then fed to the Rate Allocator block, which runs max-min fairness or any other rate allocation algorithm.

A. Lloyd-Max Rate Quantization

For rate quantization, we use the Lloyd-Max quantization scheme. As Observation 2 suggests, flow rates are not uniformly distributed, and Lloyd-Max [9], [10] is a quantization technique that assigns bins based on the probability distribution of flow rates. It starts with an arbitrary set of initial representative points. Then, it iteratively updates the bin boundaries and representatives to minimize the mean squared error of quantization of flow rates. For updating the boundaries, it finds the midpoint of two representatives, *i.e.*, $b_i = \frac{m_i + m_{i+1}}{2}$, and for updating the representatives, it finds the mean of the flow rates within each bin given the probability distribution of the rates. These steps are repeated until it converges to an optimal solution where the mean squared error is below a certain accepted error level.

²In this paper, we focus on comparison of the traditional centralized max-min fairness algorithms and our suggested one. In another line of work, we have designed a hierarchical congestion control model to reduce these types of overheads by solving the max-min fairness problem in a hierarchical design. As a result, the overhead of message passing between routers and C4 is kept to a minimum.

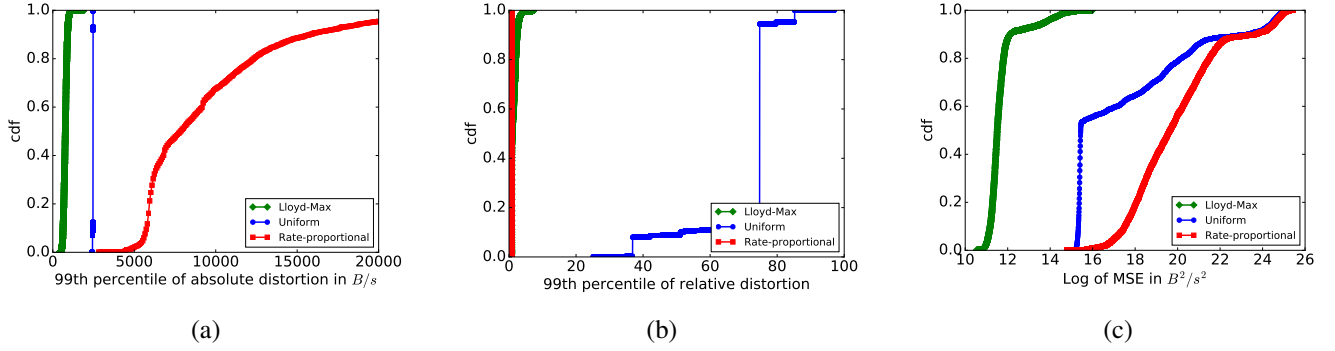


Fig. 5. Comparison of the error of the three methods (a) CDF of 99th percentile of absolute distortion, (b) CDF of 99th percentile of relative distortion, (c) CDF of log of mean squared error.

To illustrate the effects of the underlying quantization scheme, we compare the Lloyd-Max scheme with two other quantization methods as baselines: uniform and rate-proportional. We describe the details of these two methods next.

Baseline Method 1: Uniform Quantization. In this method, we uniformly divide the range of flow rates into bins. All bins are of the same size and we represent this size by l . We find the 99th percentile of flow rates and denote it by r_{max} . Thus, the number of bins is $\frac{r_{max}}{l}$. We allocate flow rates to bins such that a flow with rate r is placed in bin $\lfloor \frac{r}{l} \rfloor + 1$. The boundaries of the i -th bin are $a_i = (i-1) \times l$ and $b_i = i \times l$. The midpoint of each bin is selected as the representative of each bin, *i.e.*, $m_i = \frac{a_i + b_i}{2}$. As a result, the maximum quantization error is $\frac{l}{2}$ for flows whose rates are less than r_{max} . Large flows whose rates are greater than r_{max} , are less than 1 percent of flows which are mapped to r_{max} .

Baseline Method 2: Rate-Proportional Quantization. As Figure 3 suggests, a large number of flows have small rates. Thus, it is reasonable to allocate more bins to small flow rates and fewer bins to sparse areas of the flow rate range. Moreover, when considering the relative errors, the effect of a fixed error is more significant for small rates compared to larger ones. Therefore, instead of uniformly distributing the bins over the given range, we might want to increase the bin sizes as the rates they represent grow.

To be more specific, we want to guarantee that the maximum error of flow rates is less than ϵ as a fraction of their observed values. The quantization method introduced in [7] has these properties. In their method, the bin representatives are powers of a constant factor ϵ *i.e.* $m_i = (1 + \epsilon)^i$.

B. Traffic Predictability

To run the Lloyd-Max quantization algorithm, we need the probability distribution of flow rates in each interval as input. Predicting future rates in a network is extremely challenging though. In fact, studies on data center traffic have shown that it is almost impossible to predict the traffic matrix for periods more than 150 seconds [20], [21].

Even for shorter periods of time (*e.g.* 1.5 to 5 seconds), only 35% of the flows are predictable [21]–[23]. Here, we call a flow predictable if its future rate is within a 20% margin of its

current rate. Obviously, the situation can be worse, if we aim at higher accuracy levels. Any solution that uses the history of flows to extrapolate future rates, needs to either accurately predict future rates, or might fail.

Interestingly, predicting future bins is a fundamentally different problem than predicting the rate of an individual flow. Intuitively, the behaviour of individual flows can be very noisy due to stochastic nature of the application generating the flow, and the randomness in the network. However, the set of bins identified for rate allocation, might not change as quickly. In Section IV, we show that quantization bins can be predicted with high accuracy even with a simple prediction strategy of using the immediate past flow rates as the input distribution.

A reasonable question here is how the presented solution deals with short-lived flows lasting less than T_1 . These flows by definition last less than the duration of our bin generation interval, which means the flow is terminated by the time we can assign it to a bin. As argued above, predictability in this problem is not defined per flow, but for the flow rate distribution. Even though individual short-lived flows might finish quickly, we have a constant presence of such short-lived flows, which means our Bin Generator will naturally create bins for such flows during every time interval.

For extremely short flows that last less than T_2 , we need to make special considerations as they finish before even a single Rate Allocator cycle. This is a major problem for any rate allocation solution (including ours). In our solution, we start with a default rate for short-lived flows, and update the rate if the flow continues for more than T_2 seconds. We also allocate a fixed fraction of the available bandwidth for this class of flows. We leave finding better solutions for short-lived flow rate allocations (for example, estimating rates based on similarities of past of present flows) as a future work.

IV. EVALUATION

In this section, we evaluate our proposed solution for acceleration of the max-min fairness algorithm. We compare the Lloyd-Max quantization method with two baseline solutions namely the uniform and rate-proportional quantization schemes.

We use real network traces to measure the reduction in the number of unique flows and the run-time of max-min fairness,

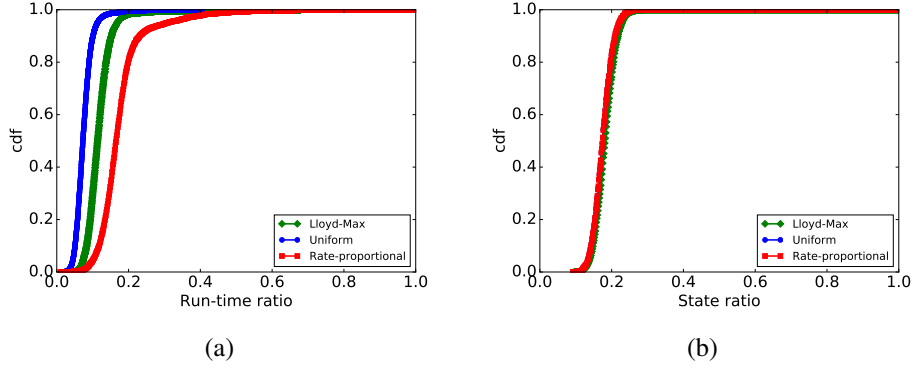


Fig. 6. Comparison of the run-time and the number of bins for Lloyd-Max, uniform, and rate-proportional quantization: (a) CDF of run-time ratio, (b) CDF of state ratio.

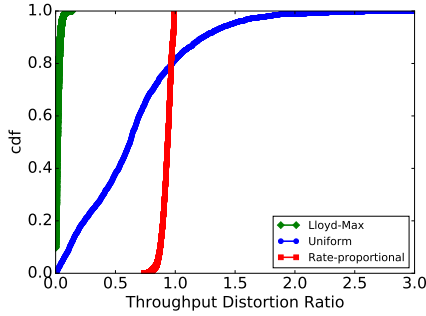


Fig. 7. CDF of changes in throughput for the three quantization methods as a fraction of the optimal case with no quantization.

as well as absolute, relative, and mean squared errors resulting from each method. We also take a closer look at predictability of the proposed solution, and how well it performs using recent history (as opposed to current and future rates).

We note that all these experiments focus on a single bottleneck scenario. Our goal is to understand how rate quantization and reduction of unique flow rates impact the run-time of rate allocation. Similar results can be expected for the multiple bottleneck scenario.

A. Evaluation Settings

We assume a dumbbell topology with 25 source nodes and 25 destination nodes, with the core link as the single bottleneck of the network. We use publicly available network traces from a study of data center traffic by Benson et al. [19]. More specifically, we extract flow level information from the packet traces of UNI1 and UNI2 data centers and use the five tuple (source IP, destination IP, source port, destination port, protocol) to identify flows and map source and destination IP addresses to source and destination nodes in the dumbbell topology.

We divide time into fixed intervals, and for each interval we calculate the rate associated with each active flow. The number of active flows changes in different intervals (since we are using real traffic traces); the average number of flows in different intervals is 290.94. We implement a C4 application that receives the flow demands and allocates max-min rates to flows accordingly. We simulate the operation of C4 to evaluate

its operation when receiving quantized flow rates compared to the original rates.

There is a trade-off between the run-time of the rate allocation scheme, and the distortion in rates resulting from quantization. To understand this trade-off in Lloyd-Max quantization and the other two baseline quantization schemes, we consider the following metrics.

- **Run-time.** We measure the run-time of max-min rate-allocation with and without rate quantization, and study their ratio. Obviously, any value below 1 shows an improvement in run-time and lower values represent higher improvements.
- **State Ratio.** We quantify state reduction by measuring the ratio of the number of bins to the number of flows in each interval.
- **Absolute Distortion.** We find the difference of the max-min fair rate allocated to each flow when the input demand rates are quantized, and the max-min fair rate allocated to the same flow given the original demand rates (in B/s). We call this measure as absolute distortion in the following results. In each interval, we measure absolute distortion caused by quantization for each individual flow and calculate the 99th percentile of this measure.
- **Relative Distortion.** We also measure the absolute distortion as a fraction of the original flow rate value.
- **Mean squared error (MSE).** We measure the mean squared error in each interval as an indication of how close the quantized values and the original rates are.
- **Throughput Distortion Ratio.** The optimal throughput is achieved when the exact demand rates are given to the max-min fairness algorithm. We find the difference of the optimal throughput and the achieved throughput when the demand rates are quantized relative to the magnitude of the optimal throughput. For this metric, values that are closer to zero indicate that the throughput values are closer to optimal.

At the beginning of each time interval, we run the quantization algorithm based on rates observed during the previous time interval. The resulting bins are used by the max-min fairness algorithm during the entire interval. As noted before, the max-min fairness algorithm runs at a much higher frequency. We measure absolute and relative errors, and the run-

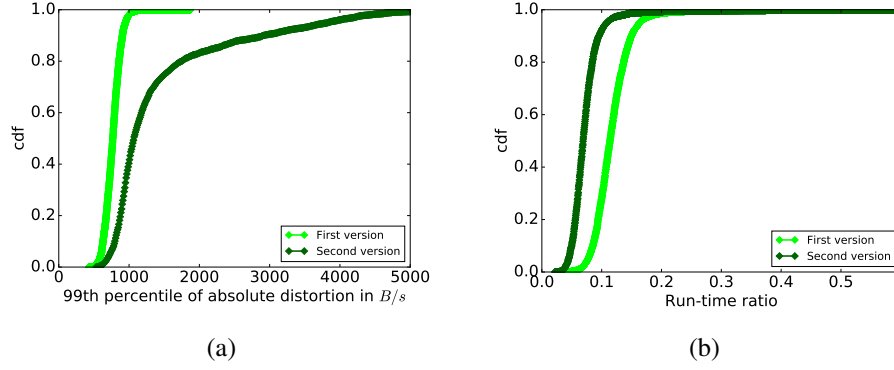


Fig. 8. Comparison of the two versions: one with bin-ignoring option disabled and the other enabled (a) CDF of 99th percentile of absolute distortion, (b) CDF of run-time ratio.

time of the max-min fairness algorithm each time the max-min fairness is executed. We run the quantization methods every $T_1 = 10$ seconds and max-min fairness every $T_2 = 2$ seconds. We set the number of bins for quantization equal to 50.

We start by reviewing the observations made in Section II-B. We measure the 99th percentile of flow rates in different intervals and call this $r_{\max}$. Flows with rates higher than $r_{\max}$ show extremely high variations from interval to interval. Therefore, including these flows in bins resulting from the Bin Generator (Figure 4) can lead to extremely large errors.

To avoid these errors, we generate quantization bins for flows with rates below $r_{\max}$ and map these flows to the their corresponding bins. For flows with rates higher than $r_{\max}$ we create one bin per flow, mapping each flow demand to its exact rate. This decision significantly reduces the quantization error by adding a small number of quantization bins. The 99th percentile of flows below $r_{\max}$ can be quantized with a small number of bins and with small error levels, and the remaining 1% get their individual bins with no errors.

B. Evaluation Results

Impact of Quantization Method on Rate Distortion. Our first set of experiments are designed to evaluate the Lloyd-Max quantization and the two baselines (uniform and rate-proportional quantization) in terms of their distortion.

Figure 5(a) shows the CDF of 99th percentile of the absolute distortion (error) over time. The distortion is bounded for the Lloyd-Max method, and flows experience a distortion less than 1 KB/s. Lloyd-Max has the least distortion among the three methods studied in all intervals. Intuitively, this is because Lloyd-Max adapts the bins according to the distribution of the flow demands as opposed to uniform and rate-proportional schemes which have fixed bins regardless of rate distributions. By placing bin representatives in high density areas, Lloyd-Max can keep the absolute distortion level low compared to the other two methods. As shown in this Figure, rate-proportional quantization shows the worst performance in terms of absolute error.

Focusing on relative errors, we can see that rate-proportional quantization shows the best performance (Figure 5(b)) followed very closely by the Lloyd-Max scheme. This is expected as the rate-proportional method focuses on relative errors.

Here, uniform distribution shows the worst performance, with 90% of flows having relative errors of more than 30.

Figure 5(c) shows the CDF of the mean squared error (log scale) for all three schemes. The absolute winner here is the Lloyd-Max scheme, with extremely low errors all across the board. The uniform and rate-proportional schemes show similar behavior here, with much higher errors compared to Lloyd-Max.

Impact of Quantization Method on Run-time and State.

Figure 6(a) shows the CDF over time of the improvement in max-min fairness run-time for the three proposed schemes. This ratio is between 0.02 to 0.3 for the uniform quantization, and between 0.05 to 0.3 for the Lloyd-Max methods, which is a relative reduction of 70-95% in run-time of the max-min rate allocation. The ratio is between 0.1 to 0.5 for the relative method which is higher compared to the two other methods.

Figure 6(b) shows the state reduction caused by quantization. It shows that the number of bins is on average around $0.2 \times$ the number of flows. Note that the curves representing each of the three methods overlap in this graph since they all use the same number of bins for each interval in our experiments.

Impact of Quantization Method on Throughput. To study the impact of quantization method on throughput, we calculate the change in throughput as a fraction of the optimal throughput (*i.e.* when no quantization is used) shown as a CDF over time in Figure 7. This figure suggests the throughput for Lloyd-Max is extremely close to the optimal, while the uniform and rate proportional methods lead to significant reductions in throughput.

Impact of Ignored Bins. We run two versions of each quantization method. In the first version, we keep all the generated bins while in the second version, we dynamically remove any bin that does not contain any flows from the previous interval. This modification can enhance the run-time, but comes at the cost of increased error if demand changes push a given flow to a non-existing bin. In that case, the flow will be assigned to the closest representative with potentially larger errors.

Figures 8(a) and 8(b) show a comparison of the absolute distortion and run-time ratio for Lloyd-Max when bin ignoring option is enabled and disabled. As suggested by these figures,

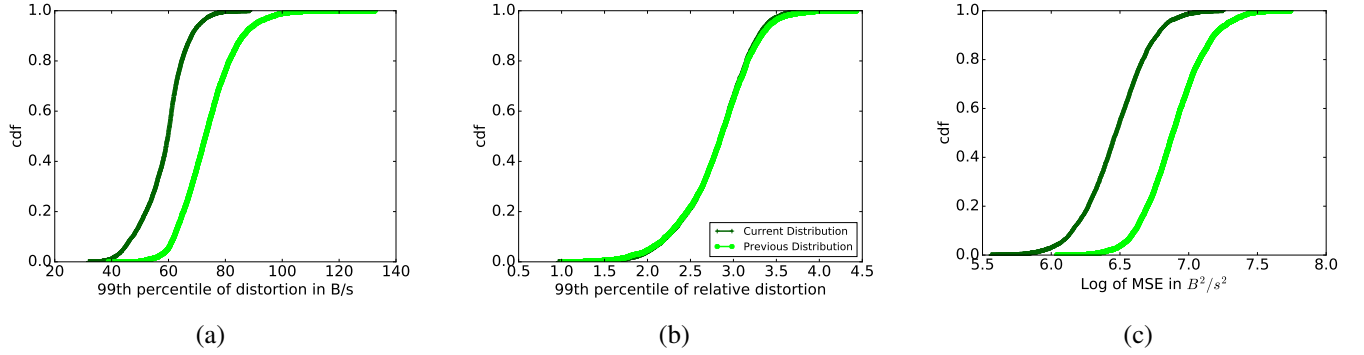


Fig. 9. Comparison of the two versions of Lloyd-Max: one that uses the distribution of the previous interval and one that uses the distribution of the current/future interval as an input: (a) CDF of 99th percentile of distortion, (b) CDF of 99th percentile of relative distortion, (c) CDF of log of MSE error.

bin-ignoring speeds up the run-time by a factor of about 1.5 for different intervals with the cost of increasing the distortion. The relative distortion and MSE metrics change with a similar pattern to the absolute distortion and the state ratio changes similar to the run-time ratio. Also, the uniform and rate-proportional methods show similar patterns for the run-time and state ratio. However, the error metrics are not bounded for the uniform method when bin-ignoring is enabled. Therefore, an acceleration in the run-time using bin-ignoring is not suggested for the uniform method.

Effects of Learning and Prediction Methods. Existing rate allocation schemes that benefit from rate quantization [7], [8] require estimates of rate demands as input. In practice, there is a natural delay between rate demand measurements, rate allocation, and enforcement of allocated rates. Rapid changes in rate demands can lead to significant errors. To address this problem, we either need to predict future demands, or develop techniques that are resilient to natural changes in rates.

Lloyd-Max rate quantization is a very effective tool when it comes to predicting future rate demands. In general, predicting future rate of a given flow based on its past behavior is an extremely challenging problem [19], [21]. When dealing with quantized rates, however, we are interested in how the distribution of quantized rates changes, rather than predicting the exact rate of a given flow. In other words, if the output of rate quantization is a set of bins $B = \{b_1, b_2, \dots, b_k\}$, we are interested in changes in B rather than the changes in the rate of individual flows. This is a fundamentally different problem than individual flow rate prediction.

To see how Lloyd-Max quantization obviates the need for predicting future rates, we compare two different versions of Lloyd-Max. The first version is what we have used so far in this paper: we feed the flow distribution of the previous time interval as input to the Lloyd-Max scheme, and receive a set of bins as output. In the second version of Lloyd-Max, we feed current rates (as opposed to rates from previous time interval) of flows to the Lloyd-Max scheme. Given the natural delays in network, this is equivalent to having an *oracle* that can perfectly predict future rates of each flow, which is impossible in practice. We note that such an oracle provides an upper-bound for the performance of any solution that tries to predict exact current or future rates, and we use it to understand how

well Lloyd-Max performs.

Figure 9 shows the errors resulting from Lloyd-Max with and without an oracle. As shown in 9(a), the absolute distortion of Lloyd-Max is reduced by only a few bytes per second as a result of perfect prediction of current/future rates. The relative distortion (Figure 9(b)) is almost similar for both approaches. The mean squared error (Figure 9(c)) slightly differs between the two versions, with the second version showing a small improvement. In general, these figures suggest the performance of the two version are very close.

The second version of Lloyd-Max (when we have an oracle) is equivalent to a rate prediction solution with 100% accuracy. Therefore, it provides an upper-bound for improvements resulting from any rate predicting solution. Figure 9 suggests we can only expect minor improvement in this case, which means Lloyd-Max using past intervals obviates the need for any complex rate prediction techniques.

V. CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of run-time complexity for explicitly and directly using max-min fairness for rate allocation. We argued that if the number of unique values for flow demands are small, the run-time can be significantly reduced. We proposed quantization of flow demands using Lloyd-Max algorithm and showed that this scheme can lead to 70 – 95% improvement in run-time of max-min fairness allocation with small errors (about 1 KB/s).

Our solution relies on previously observed demands to identify the flow distribution and quantization bins. Our experiments show that any complex rate prediction technique (even with 100% accuracy) can only lead to minor improvements.

As for future work, we note that when congestion control agents are implemented in hardware, dealing with a limited set of rates can significantly reduce the complexity of the hardware, and the memory requirements. The required memory of any hardware based congestion control agent, independent of its congestion control plane, correlates with the size of possible rates (*i.e.* accuracy of congestion control scheme). Therefore, mapping the infinite range of flow rates to finite quantized values can significantly reduce the memory requirements. This is an interesting area that can be explored as a future work.

REFERENCES

- [1] A. Narayan, F. Cangialosi, P. Goyal, S. Narayana, M. Alizadeh, and H. Balakrishnan, "The case for moving congestion control out of the datapath," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 101–107.
- [2] A. Narayan, F. Cangialosi, D. Raghavan, P. Goyal, S. Narayana, R. Mittal, M. Alizadeh, and H. Balakrishnan, "Restructuring endpoint congestion control," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 30–43.
- [3] M. Ghobadi, S. H. Yeganeh, and Y. Ganjali, "Rethinking end-to-end congestion control in software-defined networks," in *Proceedings of the 11th ACM Workshop on Hot Topics in networks*. ACM, 2012, pp. 61–66.
- [4] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '99. New York, NY, USA: ACM, 1999, pp. 175–187, event-place: Cambridge, Massachusetts, USA. [Online]. Available: <http://doi.acm.org/10.1145/316188.316220>
- [5] D. Zhuo, Q. Zhang, V. Liu, A. Krishnamurthy, and T. Anderson, "Rack-level Congestion Control," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: ACM, 2016, pp. 148–154, event-place: Atlanta, GA, USA. [Online]. Available: <http://doi.acm.org/10.1145/3005745.3005772>
- [6] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.
- [7] B. Awerbuch and Y. Shavitt, "Converging to approximated max-min flow fairness in logarithmic time," in *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 1998, pp. 1350–1357.
- [8] A. Charny, K. Ramakishnan, and A. G. Lauck, "Scalability issues for distributed explicit rate allocation in atm networks," in *Proceedings of IEEE INFOCOM'96. Conference on Computer Communications*, vol. 3. IEEE, 1996, pp. 1198–1205.
- [9] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [10] A. Ortega and M. Vetterli, "Adaptive scalar quantization without side information," *IEEE Transactions on Image Processing*, vol. 6, no. 5, pp. 665–676, 1997.
- [11] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 846–854.
- [12] G. Rézvári, J. Bíró, and T. Cinkler, "Fairness in capacitated networks: A polyhedral approach," in *INFOCOM, 2007*, pp. 1604–1612.
- [13] D. Nace, "A linear programming based approach for computing optimal fair splittable routing," in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*. IEEE, 2002, pp. 468–474.
- [14] D. Nace, L. N. Doan, O. Klopfenstein, and A. Bashllari, "Max–min fairness in multi-commodity flows," *Computers & Operations Research*, vol. 35, no. 2, pp. 557–573, 2008.
- [15] M. Pióro, P. Nilsson, E. Kubilinskas, and G. Fodor, "On efficient max-min fair routing algorithms," in *Computers and Communication, 2003.(ISCC 2003). Proceedings. Eighth IEEE International Symposium on*. IEEE, 2003, pp. 365–372.
- [16] M. Allalouf and Y. Shavitt, "Centralized and distributed algorithms for routing and weighted max-min fair bandwidth allocation," *IEEE/ACM Transactions on networking*, vol. 16, no. 5, pp. 1015–1024, 2008.
- [17] L. Jose, S. Ibanez, M. Alizadeh, and N. McKeown, "A distributed algorithm to calculate max-min fair rates without per-flow state," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 2, p. 21, 2019.
- [18] J. Ros and W. K. Tsai, "A theory of convergence order of maxmin rate allocation and an optimal protocol," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 717–726.
- [19] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 267–280.
- [20] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 65–72.
- [21] —, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh Conference on emerging Networking EXperiments and Technologies*, 2011, pp. 1–12.
- [22] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.
- [23] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," 2009.