A Non-Monetary Mechanism for Optimal Rate Control Through Efficient Delay Allocation

Tao Zhao
Department of ECE
Texas A&M University
College Station, TX 77843, USA
Email: alick@tamu.edu

Korok Ray
Mays School of Business
Texas A&M University
College Station, TX 77843, USA
Email: korok@tamu.edu

I-Hong Hou
Department of ECE
Texas A&M University
College Station, TX 77843, USA
Email: ihou@tamu.edu

***Abstract*—This paper proposes a practical non-monetary mechanism that induces the efficient solution to the optimal rate control problem, where each client optimizes its request arrival rate to maximize its own net utility individually, and at the Nash Equilibrium the total net utility of the system is also maximized. Existing mechanisms typically rely on monetary exchange which requires additional infrastructure that is not always available. Instead, the proposed protocol is based on efficient delay allocation, where the server controls the delay experienced by each client through an intelligent scheduling policy. Specifically, we present an efficient delay allocation rule for the server to determine the target delay of each client. Then we propose a simple scheduling policy to achieve such delay allocation. Furthermore, we design a distributed rate control protocol for the system to converge to the Nash Equilibrium. The optimality of our mechanism is validated via extensive simulations on two representative systems against a baseline mechanism with FIFO scheduling and centralized rate control.**

I. Introduction

The mobile Internet market has been enjoying an unprecedented growth since the recent few years. It is predicted that the trend will continue, and the global mobile data traffic will increase nearly eightfold between 2015 and 2020 [1]. With the growing market, it is of great interest to understand the economics of the network. In this paper, we are interested in finding a practical mechanism to induce the efficient solution to the optimal rate control problem in a network system of multiple selfish and strategic clients. We consider systems where a server processes requests from multiple clients, and each client can dynamically adjust its own request arrival rate. Each client obtains some utility based on its request arrival rate and its own utility function, but also suffers from some disutility based on its experienced delay. Each client optimizes its request arrival rate to maximize its own net utility individually, and at the Nash Equilibrium the total net utility is also maximized. Our system model can be applied to a wide range of networks. For example, the clients might be smartphones, wearable devices, tablets and so on, and the server can be a cellular base station (e.g. LTE eNodeB) or a WiFi hotspot which provides Internet services to the clients. Each request corresponds to an LTE subframe or an IP packet.

The optimal rate control problem, which entails maximizing the total net utility in the system, is typically convex, and it is thus easy to solve when one has complete information of all the individual utility functions. In practice, however, the utility functions are often private information of clients, and a strategic client that aims to maximize its own net utility may not reveal its true utility function. Further, request rates are directly controlled by clients, instead of the server. Most existing work employs some auction or pricing schemes to ensure strategic clients to reveal their true functions and follow the assigned rates from the server [2], [3]. However, these schemes involve additional monetary exchange between clients and the server, which requires additional infrastructure that is not always available.

In this paper, we propose a novel non-monetary mechanism for optimal rate control to address this issue. Note that each client suffers from some disutility based on its experienced delay, and the server can indirectly control the delay experienced by each request through its employed scheduling policy. Therefore, the server can potentially steer request rates of strategic clients toward the optimal point through its scheduling policy. Effectively, the server uses "delay" as a kind of "currency."

In economic terms, there are negative externalities from a client increasing its request rate, since this increases the overall average delay of all clients. This is an analogy to a public goods problem [4], in which one client's consumption choice affects the utility and payoffs of the other clients. As such, the server's objective is to design an allocation scheme such that each client internalizes these negative externalities, thereby leading to efficient consumption of resources.

In designing the non-monetary mechanism, we make the following contributions:

1) We propose an efficient delay allocation rule through which the server can guide the clients to the Nash Equilibrium where the total net utility of the system is also maximized.
2) We then come up with a simple yet effective scheduling policy that can be proved to enforce the efficient delay allocation rule in the heavy traffic regime, that is when the total request rate approaches the service rate.
3) We present a scalable and lightweight distributed rate control protocol between the server and the clients which drives the system to converge to the Nash Equilibrium.

Altogether, they form our non-monetary mechanism for opti-

mal rate control through efficient delay allocation.

The rest of the paper is organized as follows. Section II reviews the literature related to our work. Section III introduces our system model and problem formulation. Section IV, V, and VI present the efficient delay allocation rule, the efficient delay scheduling policy, and the distributed rate control protocol respectively. Simulation study is described in Section VII, and we conclude our paper in Section VIII.

II. Related Work

There has been a number of literature that studies networks from the respect of economics. Altman et al. gave a comprehensive survey on networking games [5]. Specifically for rate control, Kelly et al. analyzed the stability and fairness of pricing based rate control algorithms [3]. Alpcan and Başar gave a utility-based congestion control scheme for cost minimization and showed its stability for a general network topology [6]. Hou and Kumar presented a truthful and utility-optimal auction for wireless networks with per-packet deadline constraints [2]. Gupta et al. studied network utility maximization where flows are aggregated into flow classes [7]. Ramaswamy et al. considered the case when a client can choose from a number of congestion control protocols [8].

Besides, our work shares a similar spirit as delay-based TCP variants, such as TCP Vegas [9], TCP Westwood+ [10], [11], and FAST TCP [12], in the sense that delay is used as the signal for the clients to adjust their request rates.

The intellectual foundation of our research comes from economics. The early literature began with problems of creating incentives to reduce free riding in teams, such as in Groves [13]. This research uses much of the similar logic as our method on the behavior of other agents in a strategic game. Baldenius et al. [14], Moulin and Shenker [15], and Rajan [16] studied the problem of cost allocation, namely, how to allocate a common cost to separate corporate departments.

III. System Model and Problem Formulation

Consider a system with N clients and a server. Each client i generates requests by some predefined random process, such as Poisson random process, but it can dynamically adjust its average request rate, denoted by λ_i. We use $\boldsymbol{\lambda} := [\lambda_i]$ to denote the vector containing the average request rates of all clients, and λ_{-i} to denote the vector of average request rates of all clients other than i.

On the other hand, the server employs some scheduling policy to determine which request to process. Unserved requests are queued in the system. The processing time of each request is a random variable with mean $\frac{1}{\mu}$. If the server's scheduling policy is work-conserving, which never idles as long as there is at least one request available for processing, then the average delay of all requests is a function of the total average request arrival rate, $\Lambda := \sum_i \lambda_i$, regardless of the employed scheduling policy. The average delay function $\bar{C}(\Lambda)$ is strictly increasing and strictly convex. We assume that the average delay $\bar{C}(\Lambda)$ can be well fitted by a $(N-2)$-order polynomial function $C(\Lambda)$ via, for example, Chebyshev least squares approximation.

Suppose each client obtains some *utility* based on its request rate λ_i and suffers from *disutility* for every unit delay experienced by each of its request. Specifically, the utility of client i is $U_i(\lambda_i)$, where $U_i(\cdot)$ is an increasing, twice differentiable, and concave function. Let $D_i(\lambda_i, \lambda_{-i})$ be the average delay that client i experiences for all its requests. The disutility of client i is $\lambda_i D_i(\lambda_i, \lambda_{-i})$. The *net utility* of client i is therefore $U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \lambda_{-i})$.

The server aims to maximize the total net utility in the system, which can be written as $\sum_i U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \lambda_{-i})$. Since the average delay of *all* requests is the weighted average $\frac{\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i})}{\Lambda} \approx C(\Lambda)$, we say that the server aims to maximize $\sum_i U_i(\lambda_i) - \Lambda C(\Lambda)$. Note that the average delay of all requests is always infinite when the system is overloaded with $\Lambda \geq \mu$. To simplify discussions, we assume that $\boldsymbol{\lambda}$ has the properties that $\Lambda := \sum_i \lambda_i < (1-\epsilon)\mu$, where $\epsilon > 0$ is a predetermined value known to the server. We further assume that $\lambda_i > \lambda_\delta$ for all i, for some predetermined $\lambda_\delta > 0$ known to the server. These assumptions are not restrictive since we can choose ϵ and λ_δ arbitrarily close to 0. The server's optimization problem is thus formally:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} U_i(\lambda_i) - \Lambda C(\Lambda), \\ \text{subject to} \quad & \Lambda < (1-\epsilon)\mu, \\ & \lambda_i > \lambda_\delta, \forall i. \end{aligned} \tag{1}$$

Since $U_i(\cdot)$ is concave, $C(\cdot)$ is convex, and the feasible region $\mathcal{S}_{\boldsymbol{\lambda}} := \{\boldsymbol{\lambda} \,|\, \Lambda < (1-\epsilon)\mu, \lambda_i > \lambda_\delta\}$ is a convex set, the problem of maximizing the total net utility can be easily solved when one has complete information of all these functions. In practice, however, the function $U_i(\cdot)$ is the private information of client i, and a strategic client may not reveal its true $U_i(\cdot)$. Now consider a game where, given $\boldsymbol{\lambda}$, the server determines the average delay experienced by each client i, $D_i(\lambda_i, \lambda_{-i})$, with the constraint that $\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i}) \geq \Lambda C(\Lambda)$. On the other hand, given λ_{-i} and $U_i(\cdot)$, each client i aims to maximize its own net utility by solving

$$\tilde{\lambda}_i = \operatorname*{argmax}_{\lambda_i} U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \lambda_{-i}). \tag{2}$$

Note that we allow $\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i})$ to be strictly larger than $\Lambda C(\Lambda)$, which can be achieved by employing a policy that is not work-conserving and may arbitrarily delay, or drop, packets.

We say that the system reaches a Nash Equilibrium if no client in the system can improve its own net utility unilaterally.

Definition 1. *A vector $\tilde{\boldsymbol{\lambda}} := [\tilde{\lambda}_i]$ is said to be a Nash Equilibrium if $\tilde{\lambda}_i = \operatorname{argmax}_{\lambda_i} U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \tilde{\lambda}_{-i}), \forall i$.*

Let $\boldsymbol{\lambda}^* := [\lambda_i^*] \in \mathcal{S}_{\boldsymbol{\lambda}}$ be the vector that maximizes the total net utility, then the server's problem is to find the rule that allocates delays, $[D_i(\cdot)]$, to induce optimal choices of $[\lambda_i]$.

Definition 2. *A rule of allocating delays, $[D_i(\cdot)]$, is said to be* efficient *if $\boldsymbol{\lambda}^*$ is the only Nash Equilibrium.*

IV. Efficient Delay Allocation

In this section, we propose an efficient delay allocation rule. We first study some basic properties of the optimal vector $\boldsymbol{\lambda}^* = [\lambda_i^*]$ that maximizes total net utility $\sum_i U_i(\lambda_i) - \Lambda C(\Lambda)$. We have

$$\frac{\partial}{\partial \lambda_i}\left[\sum_i U_i(\lambda_i^*) - \Lambda^* C(\Lambda^*)\right] = 0. \tag{3}$$

Hence,

$$U_i'(\lambda_i^*) = \frac{\partial}{\partial \lambda_i}\Lambda^* C(\Lambda^*). \tag{4}$$

On the other hand, if $\boldsymbol{\lambda}^*$ is also the Nash Equilibrium under some delay allocation rule $[D_i(\cdot)]$, then λ_i^* maximizes $U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \lambda_{-i}^*)$, and we have

$$\frac{\partial}{\partial \lambda_i}[U_i(\lambda_i^*) - \lambda_i^* D_i(\lambda_i^*, \lambda_{-i}^*)] = 0. \tag{5}$$

Hence,

$$U_i'(\lambda_i^*) = \frac{\partial}{\partial \lambda_i}\lambda_i^* D_i(\lambda_i^*, \lambda_{-i}^*). \tag{6}$$

Combining the above equations yields

$$\frac{\partial}{\partial \lambda_i}[\Lambda^* C(\Lambda^*) - \lambda_i^* D_i(\lambda_i^*, \lambda_{-i}^*)] = 0. \tag{7}$$

Eq. (7) suggests that an efficient rule of delay allocation should ensure that $\Lambda C(\Lambda) - \lambda_i D_i(\lambda_i, \lambda_{-i})$ is only determined by λ_{-i}, and is not influenced by λ_i. This implication has indeed been formally stated and proved in [4]:

Proposition 1. *$[D_i(\cdot)]$ is efficient if and only if there exists functions $R_i : \mathbb{R}^{N-1} \to \mathbb{R}$ such that for all i,*

$$\lambda_i D_i(\lambda_i, \lambda_{-i}) = \Lambda C(\Lambda) - R_i(\lambda_{-i}), \tag{8}$$

and

$$\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i}) = \Lambda C(\Lambda). \tag{9}$$

Recall that $C(\Lambda)$ is a $(N-2)$-order polynomial. Therefore, $\Lambda C(\Lambda)$ is a $(N-1)$-order polynomial, and can be expressed as $\Lambda C(\Lambda) = c_1\Lambda + c_2\Lambda^2 + \cdots + c_{N-1}\Lambda^{N-1}$.

We now define some helpful terminology. First define the sets

$$P^j := \left\{\boldsymbol{p} = [p_i] \;\middle|\; p_i \text{ is a nonnegative integer}, \sum_{i=1}^N p_i = j\right\}, \tag{10}$$

$$P_i^j := \{\boldsymbol{p} \in P^j \,\big|\, p_i = 0\}, \tag{11}$$

for $j = 1, \ldots, N-1$ and $i = 1, \ldots, N$. Next, for $\boldsymbol{p} \in P^j$, let $G(\boldsymbol{p})$ be the number of nonzero coordinates of $\boldsymbol{p}$: $G(\boldsymbol{p}) := \big|\{l \;\big|\; p_l \neq 0\}\big|$. Note that $G(\boldsymbol{p})$ is at most j, for all $\boldsymbol{p} \in P^j$. Finally, define $\binom{j}{\boldsymbol{p}} := \frac{j!}{p_1!\cdots p_N!}$.

By the multinomial expansion theorem, it holds that

$$(\lambda_1 + \cdots + \lambda_N)^j = \sum_{\boldsymbol{p}\in P^j}\binom{j}{\boldsymbol{p}}\lambda_1^{p_1}\cdots\lambda_N^{p_N}. \tag{12}$$

We now introduce our delay allocation rule. Let

$$\beta_i^j = c_j \sum_{\boldsymbol{p}\in P_i^j}\frac{N-1}{N-G(\boldsymbol{p})}\binom{j}{\boldsymbol{p}}\lambda_1^{p_1}\cdots\lambda_N^{p_N}, \tag{13}$$

for $j = 1, \ldots, N-1$. We then choose $R_i(\lambda_{-i})$ as

$$R_i(\lambda_{-i}) = \sum_{j=1}^{N-1}\beta_i^j, \tag{14}$$

and

$$\lambda_i D_i(\lambda_i, \lambda_{-i}) = \Lambda C(\Lambda) - R_i(\lambda_{-i}). \tag{15}$$

Theorem 1. *The rule of delay allocation $[D_i(\cdot)]$ as defined by Eq. (14) and (15) is efficient.*

Proof: Since $p_i = 0$ for all $\boldsymbol{p} \in P_i^j$, it is obvious that $R_i(\lambda_{-i}) = \sum_{j=1}^{N-1}\beta_i^j$ is not influenced by λ_i.

Next, we check the condition $\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i}) = \Lambda C(\Lambda)$. By Eq. (13), for every $\boldsymbol{p} \in P^j$, the term $\frac{N-1}{N-G(\boldsymbol{p})}\binom{j}{\boldsymbol{p}}\lambda_1^{p_1}\cdots\lambda_N^{p_N}$ appears in β_i^j if and only if $p_i = 0$, and there are $(N - G(\boldsymbol{p}))$ different i with $p_i = 0$. Therefore, the term $\frac{N-1}{N-G(\boldsymbol{p})}\binom{j}{\boldsymbol{p}}\lambda_1^{p_1}\cdots\lambda_N^{p_N}$ appears in $[\beta_i^j]$ a total number of $(N - G(\boldsymbol{p}))$ times. We then have

$$\begin{aligned}\sum_{i=1}^N R_i(\lambda_{-i}) &= \sum_{i=1}^N\sum_{j=1}^{N-1}\beta_i^j \\ &= \sum_{j=1}^{N-1} c_j \sum_{\boldsymbol{p}\in P^j}(N-1)\binom{j}{\boldsymbol{p}}\lambda_1^{p_1}\cdots\lambda_N^{p_N} \\ &= (N-1)\Lambda C(\Lambda),\end{aligned} \tag{16}$$

and

$$\sum_i \lambda_i D_i(\lambda_i, \lambda_{-i}) = N\Lambda C(\Lambda) - \sum_{i=1}^N R_i(\lambda_{-i}) = \Lambda C(\Lambda). \tag{17}$$

Therefore, by Proposition 1, the rule of delay allocation $[D_i(\cdot)]$ as defined by Eq. (14) and (15) is efficient. ∎

V. Efficient Delay Scheduling

In this section, we propose an online scheduling policy that ensures that the actual delay experienced by each client is the same as its allocated delay, as described in Eq. (14) and (15). Before introducing the policy, we note that the allocated delays of some clients might be negative following the efficient delay allocation rule $[D_i(\cdot)]$. We call those clients with negative allocated delays "VIP" clients, because their allocated delays are among the smallest. Since it is impossible to drive the delays of VIP clients to negative in practice, we resort to ensure their delays are as close to zero as possible. If there exists one or more VIP clients with unserved requests, the server will schedule the VIP client with the largest number of unserved requests. Otherwise, we will schedule one of the non-VIP clients.

From now on, in this section, we focus on the non-VIP clients, and assume that $g_i := \lambda_i D_i > 0$ for all i. According to Little's law, $g_i = \lambda_i D_i$ can be interpreted as the target average

queue length (i.e. number of requests in the system) of client i, which is known to the server. Based on this observation, we propose the following maximum-relative-queue-length (MRQ) policy:

Definition 3 (MRQ)**.** *Let $Q_i(t)$ be the queue length of client i at time t. At time t, the MRQ policy schedules the client with the largest relative queue length, defined as $Q_i(t)/g_i$, breaking tie by scheduling the client with the lowest ID.*

The intuition behind MRQ is that by always scheduling the client with the largest relative queue length, eventually all relative queue lengths are equal on average in steady state, or equivalently, the average queue length of each client is roughly the same as its target queue length.

Below we will show that the MRQ policy indeed achieves the desirable efficient delay allocation in the heavy traffic regime.[1] In particular, we show that the deviation of the actual average delay from the target delay is bounded for each client i, regardless of the difference between the total request rate Λ and the service rate μ. When Λ approaches μ, the actual average delay goes to infinity, and therefore the deviation becomes negligible compared to the actual average delay. Our technical approach is similar to the state space collapse results in the queueing theory literature [17].

Let $\boldsymbol{g} := [g_i]$ be the vector of target queue lengths for all clients. Let $\hat{\boldsymbol{g}} := \boldsymbol{g}/\sum_i g_i$ be the normalized vector of $\boldsymbol{g}$ such that $\hat{g}_i > 0$ is the fraction of target queue length for client i and $\sum_i \hat{g}_i = 1$. Define the weighted inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ by:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i=1}^{N} \frac{x_i y_i}{\hat{g}_i},$$

and the norm of a vector $\boldsymbol{x}$ by:

$$\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}.$$

Note that $\|\hat{\boldsymbol{g}}\| = 1$ and thus $\hat{\boldsymbol{g}}$ is the unit vector in the direction of $\boldsymbol{g}$.

Let $\boldsymbol{Q}(t), \boldsymbol{A}(t)$, and $\boldsymbol{S}(t)$ be the vector of queue lengths, arrivals, and services respectively for all clients at time t. To simplify discussions, we assume that time is slotted and the duration of a time slot is τ. Moreover, in each time slot, each client can generate at most one request, and the server can serve at most one request. This assumption is not restrictive as we can set τ to be arbitrarily small. Next we define the generalized projection of $\boldsymbol{Q}(t)$ onto $\boldsymbol{g}$, denoted by $\boldsymbol{Q}_{\parallel}(t)$, as follows:

$$\boldsymbol{Q}_{\parallel}(t) := \langle \boldsymbol{Q}(t), \hat{\boldsymbol{g}} \rangle \hat{\boldsymbol{g}} = \sum_{i=1}^{N} Q_i(t) \hat{\boldsymbol{g}}.$$

Since the total queue length is $\sum_i Q_i(t)$, the queue length of each client i is exactly the i-th element of $\boldsymbol{Q}_{\parallel}(t)$ if we allocate queue lengths proportionally to $\boldsymbol{g}$. Therefore, $\boldsymbol{Q}_{\parallel}(t)$ can be thought of as the vector of target queue lengths of all clients under perfect state space collapse.

The deviation $\boldsymbol{Q}_{\perp}(t)$ of actual queue lengths $\boldsymbol{Q}(t)$ from the target queue lengths $\boldsymbol{Q}_{\parallel}(t)$ is defined as:

$$\boldsymbol{Q}_{\perp}(t) := \boldsymbol{Q}(t) - \boldsymbol{Q}_{\parallel}(t).$$

Now we introduce a helpful lemma to prove the state space collapse property. Our proof is based on the Lyapunov drift techniques. First, define the following Lyapunov functions:

$$V_{\perp}(t) := \|\boldsymbol{Q}_{\perp}(t)\|, W(t) := \|\boldsymbol{Q}(t)\|^2, W_{\parallel}(t) := \|\boldsymbol{Q}_{\parallel}(t)\|^2.$$

The respective drifts are defined as follows:

$$\begin{aligned} \Delta V_{\perp}(t) &:= V_{\perp}(t+\tau) - V_{\perp}(t) \\ \Delta W(t) &:= W(t+\tau) - W(t) \\ \Delta W_{\parallel}(t) &:= W_{\parallel}(t+\tau) - W_{\parallel}(t) \end{aligned}$$

The following lemma, adapted from Lemma 7 in [17], shows that the drift $\Delta V_{\perp}(t)$ can be bounded by $\Delta W(t)$ and $\Delta W_{\parallel}(t)$, and absolutely bounded.

Lemma 1. *We have*

$$\Delta V_{\perp}(t) \le \frac{1}{2\|\boldsymbol{Q}_{\perp}(t)\|} (\Delta W(t) - \Delta W_{\parallel}(t)), \tag{18}$$

and

$$|\Delta V_{\perp}(t)| \le 2\sqrt{\frac{N}{\hat{g}_{\min}}}, \tag{19}$$

where $\hat{g}_{\min} := \min_i \hat{g}_i$.

The proof is similar to that of Lemma 7 in [17] and is omitted here.

Since we are considering a single server system, it is easy to see our MRQ policy stabilizes the queues of all clients as long as $\Lambda < \mu$. Therefore, $\boldsymbol{Q}(t)$ converges to a limiting random vector $\bar{\boldsymbol{Q}}$ in steady state.

Consider the following limiting queueing process: Fix a vector $\hat{\boldsymbol{g}}$ of unit length with $\hat{g}_i > 0$, we consider all systems whose allocated delays satisfy $\boldsymbol{g}/\sum_i g_i = \hat{\boldsymbol{g}}$. Each system is indexed by $\varepsilon := \mu - \Lambda^{(\varepsilon)}$, where $\Lambda^{(\varepsilon)}$ is the total request arrival rate of the system. We use $\bar{\boldsymbol{Q}}^{(\varepsilon)}$ to denote the random vector of queue lengths in steady state for the system, and use $\bar{\boldsymbol{Q}}_{\perp}^{(\varepsilon)}$ to denote the deviation in steady state. The effectiveness of MRQ is formally stated in the following theorem:

Theorem 2. *The efficient delay allocation rule is enforced by the MRQ scheduling policy in the heavy traffic regime. That is, there exists a sequence of finite integers $\{N_r\}$ such that $\mathbb{E}\left[\left\|\bar{\boldsymbol{Q}}_{\perp}^{(\varepsilon)}\right\|^r\right] \le N_r$ for all $r = 1, 2, \ldots$ and for all $\varepsilon > 0$.*

Proof: Below the superscript $^{(\varepsilon)}$ is omitted for brevity. By [17, Lemma 1], we only need to show the Lyapunov drift $\Delta V_{\perp}(t)$ is *1)* negative when $\|\boldsymbol{Q}_{\perp}(t)\|$ is sufficiently large, and *2)* absolutely bounded. Lemma 1 has shown that *2)* is satisfied. Moreover, *1)* can be reduced to bound $\Delta W(t)$ and $\Delta W_{\parallel}(t)$.

[1] On the other hand, if the traffic is light and queues are not built up, it is not quite necessary to employ an advanced scheduling policy. The clients will also increase their request rates to maximize their net utilities.

Consider $\mathbb{E}\left[\Delta W(t) \mid \boldsymbol{Q}\right] := \mathbb{E}\left[\Delta W(t) \mid \boldsymbol{Q}(t) = \boldsymbol{Q}\right]$.

$$\begin{aligned}
&\mathbb{E}\left[\Delta W(t) \mid \boldsymbol{Q}\right] \\
&= \mathbb{E}\left[\left\|\boldsymbol{Q}(t+\tau)\right\|^2 - \left\|\boldsymbol{Q}(t)\right\|^2 \,\middle|\, \boldsymbol{Q}\right] \\
&= \mathbb{E}\left[\left\|\left(\boldsymbol{Q}(t) + \boldsymbol{A}(t) - \boldsymbol{S}(t)\right)^+\right\|^2 - \left\|\boldsymbol{Q}(t)\right\|^2 \,\middle|\, \boldsymbol{Q}\right] \qquad (20)\\
&\le \mathbb{E}\left[\left\|\boldsymbol{Q}(t) + \boldsymbol{A}(t) - \boldsymbol{S}(t)\right\|^2 - \left\|\boldsymbol{Q}(t)\right\|^2 \,\middle|\, \boldsymbol{Q}\right] \\
&\le 2\mathbb{E}\left[\langle \boldsymbol{Q}(t), \boldsymbol{A}(t) - \boldsymbol{S}(t)\rangle \mid \boldsymbol{Q}\right] + K_1,
\end{aligned}$$

where $(\cdot)^+ := \max\{0, \cdot\}$ and K_1 is a bounded constant. Below we will omit (t) in the derivation for brevity.

Given a request rate vector $\boldsymbol{\lambda}$, define a hypothetical service rate vector $\boldsymbol{\mu} := \boldsymbol{\lambda} + \varepsilon\hat{\boldsymbol{g}}$, where $\varepsilon > 0$. Note that $\mu_\Sigma := \sum_i \mu_i = \Lambda + \varepsilon = \mu$. Recall μ is the service rate the server can provide.

Next, we bound the term $\mathbb{E}\left[\langle \boldsymbol{Q}, \boldsymbol{A} - \boldsymbol{S}\rangle \mid \boldsymbol{Q}\right]$ in Eq. (20). Without loss of generality, suppose at time t, client 1 has the largest relative queue length, that is $Q_1(t)/g_1 \ge Q_i(t)/g_i$ for all i. Note that by the definition of the MRQ scheduling policy,

$$\langle \boldsymbol{Q}, \mathbb{E}\left[\boldsymbol{S} \mid \boldsymbol{Q}\right]\rangle = \frac{Q_1}{\hat{g}_1}\mu \ge \frac{Q_i}{\hat{g}_i}\mu.$$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\langle \boldsymbol{Q}, \boldsymbol{A} - \boldsymbol{S}\rangle \mid \boldsymbol{Q}\right] &= \langle \boldsymbol{Q}, \boldsymbol{\lambda} - \boldsymbol{\mu}\rangle + \langle \boldsymbol{Q}, \boldsymbol{\mu} - \mathbb{E}\left[\boldsymbol{S} \mid \boldsymbol{Q}\right]\rangle \\
&= -\varepsilon\|\boldsymbol{Q}_\parallel\| - \sum_{i=1}^N \mu_i \left|\frac{Q_i}{\hat{g}_i} - \frac{Q_1}{\hat{g}_1}\right| \\
&\le -\varepsilon\|\boldsymbol{Q}_\parallel\| - \mu_{\min}\sum_{i=1}^N \left|\frac{Q_i}{\hat{g}_i} - \frac{Q_1}{\hat{g}_1}\right|,
\end{aligned} \qquad (21)$$

where $\mu_{\min} := \min_i \mu_i$.

Since $0 < \hat{g}_i < 1$ for all i, we know $\hat{g}_i^2 < \hat{g}_i$, and thus

$$\sum_{i=1}^N \left|\frac{Q_i}{\hat{g}_i} - \frac{Q_1}{\hat{g}_1}\right| \ge \sqrt{\sum_{i=1}^N \left(\frac{Q_i}{\hat{g}_i} - \frac{Q_1}{\hat{g}_1}\right)^2} \ge \left\|\boldsymbol{Q} - \frac{Q_1}{\hat{g}_1}\hat{\boldsymbol{g}}\right\|,$$

Hence,

$$\begin{aligned}
\mathbb{E}\left[\langle \boldsymbol{Q}, \boldsymbol{A} - \boldsymbol{S}\rangle \mid \boldsymbol{Q}\right] &\le -\varepsilon\|\boldsymbol{Q}_\parallel\| - \mu_{\min}\left\|\boldsymbol{Q} - \frac{Q_1}{\hat{g}_1}\hat{\boldsymbol{g}}\right\| \\
&\le -\varepsilon\|\boldsymbol{Q}_\parallel\| - \mu_{\min}\|\boldsymbol{Q}_\perp\| \\
&\le -\varepsilon\|\boldsymbol{Q}_\parallel\| - \delta\|\boldsymbol{Q}_\perp\|,
\end{aligned} \qquad (22)$$

for any δ such that $0 < \delta < \min_i \lambda_i$.

Substituting Eq. (22) to Eq. (20), we get

$$\mathbb{E}\left[\Delta W(t) \mid \boldsymbol{Q}\right] \le -2\varepsilon\|\boldsymbol{Q}_\parallel\| - 2\delta\|\boldsymbol{Q}_\perp\| + K_1. \qquad (23)$$

Next, we obtain a lower bound of $\Delta W_\parallel(t)$. Consider $\mathbb{E}\left[\Delta W_\parallel(t) \,\middle|\, \boldsymbol{Q}\right] := \mathbb{E}\left[\Delta W_\parallel(t) \,\middle|\, \boldsymbol{Q}(t) = \boldsymbol{Q}\right]$. Let $\boldsymbol{\Psi}(t)$ be the unused service at time t such that $\boldsymbol{Q}(t+1) = \boldsymbol{Q}(t) + \boldsymbol{A}(t) - \boldsymbol{S}(t) + \boldsymbol{\Psi}(t)$. Note that $0 \le \psi_i \le 1$ for all i.

$$\begin{aligned}
\mathbb{E}\left[\Delta W_\parallel(t) \,\middle|\, \boldsymbol{Q}\right] =& \mathbb{E}\left[\langle \hat{\boldsymbol{g}}, \boldsymbol{Q} + \boldsymbol{A} - \boldsymbol{S} + \boldsymbol{\Psi}\rangle^2 - \langle \hat{\boldsymbol{g}}, \boldsymbol{Q}\rangle^2 \,\middle|\, \boldsymbol{Q}\right] \\
=& \mathbb{E}\Big[2\langle \hat{\boldsymbol{g}}, \boldsymbol{Q}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{A} - \boldsymbol{S}\rangle + \langle \hat{\boldsymbol{g}}, \boldsymbol{A} - \boldsymbol{S}\rangle^2 \\
&+ 2\langle \hat{\boldsymbol{g}}, \boldsymbol{Q} + \boldsymbol{A} - \boldsymbol{S}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{\Psi}\rangle + \langle \hat{\boldsymbol{g}}, \boldsymbol{\Psi}\rangle^2 \,\Big|\, \boldsymbol{Q}\Big] \\
\ge& 2\langle \hat{\boldsymbol{g}}, \boldsymbol{Q}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{\lambda} - \mathbb{E}\left[\boldsymbol{S} \mid \boldsymbol{Q}\right]\rangle \\
&- 2\mathbb{E}\left[\langle \hat{\boldsymbol{g}}, \boldsymbol{S}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{\Psi}\rangle \mid \boldsymbol{Q}\right] \\
\ge& 2\langle \hat{\boldsymbol{g}}, \boldsymbol{Q}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{\lambda} - \mathbb{E}\left[\boldsymbol{S} \mid \boldsymbol{Q}\right]\rangle - K_2,
\end{aligned} \qquad (24)$$

where $K_2 := 2N^2$ considering $S_i \le 1$ and $\psi_i \le 1$ for all i. The first term can be further reduced as follows:

$$2\langle \hat{\boldsymbol{g}}, \boldsymbol{Q}\rangle\langle \hat{\boldsymbol{g}}, \boldsymbol{\lambda} - \mathbb{E}\left[\boldsymbol{S} \mid \boldsymbol{Q}\right]\rangle = 2\|\boldsymbol{Q}_\parallel\|(\Lambda - \mu) = -2\varepsilon\|\boldsymbol{Q}_\parallel\|.$$

Therefore,

$$\mathbb{E}\left[\Delta W_\parallel(t) \,\middle|\, \boldsymbol{Q}\right] \ge -2\varepsilon\|\boldsymbol{Q}_\parallel\| - K_2. \qquad (25)$$

By taking expectation of Eq. (18), and substituting Eq. (23) and (25) into it, we have

$$\mathbb{E}\left[\Delta V_\perp(t) \mid \boldsymbol{Q}\right] \le -\delta + \frac{K_1 + K_2}{2\|\boldsymbol{Q}_\perp\|},$$

which establishes the negative drift of $\mathbb{E}\left[\Delta V_\perp(t) \mid \boldsymbol{Q}\right]$. Along with the absolute boundness provided by Lemma 1, we can conclude that the conditions for Lemma 1 of [17] are satisfied, and thus there exists a sequence of finite integers $\{N_r\}$ such that $\mathbb{E}\left[\left\|\bar{\boldsymbol{Q}}_\perp^{(\varepsilon)}\right\|^r\right] \le N_r$ for all $r = 1, 2, \ldots$. ∎

Remark: Since the constants in these bounds are all independent of ε, the deviation of the limiting queue length vector $\bar{\boldsymbol{Q}}^{(\varepsilon)}$ from the target queue length vector $\boldsymbol{g}$ becomes negligible as $\varepsilon \to 0$. Therefore, we observe the state space collapse behavior of relative queue lengths, and the efficient delay allocation rule is enforced by our MRQ scheduling policy in the heavy traffic regime.

VI. Distributed Rate Control Protocol

Theorem 1 has shown that our proposed delay allocation rule in Section IV is efficient. That is, suppose there is a unique vector $\boldsymbol{\lambda}^* = [\lambda_i^*]$ that maximizes total net utility $\sum_i U_i(\lambda_i) - \Lambda C(\Lambda)$ in Eq. (1), then $\boldsymbol{\lambda}^*$ is also the unique vector of Nash Equilibrium under our delay allocation rule. In this section, we propose a distributed rate control protocol for clients to dynamically adjust their rates so as to converge to the Nash Equilibrium.

Our protocol is based on the projected gradient method [18], a simple yet effective method to solve convex optimization problems. The projected gradient method consists of two steps: initialization and iterative update. In the initialization step, the method arbitrarily chooses a vector $\boldsymbol{\lambda}(1) \in \mathcal{S}_{\boldsymbol{\lambda}}$. Recall that $\mathcal{S}_{\boldsymbol{\lambda}}$ is the feasible region for $\boldsymbol{\lambda}$. In each subsequent iteration k, the projected gradient method updates $\boldsymbol{\lambda}$ by:

$$\begin{aligned}
\hat{\boldsymbol{\lambda}}(k+1) &= \boldsymbol{\lambda}(k) + \frac{\kappa(k)}{\eta(k)}\nabla\left[\sum_{i=1}^N U_i(\lambda_i) - \Lambda C(\Lambda)\right], \\
\boldsymbol{\lambda}(k+1) &= P(\hat{\boldsymbol{\lambda}}(k+1)),
\end{aligned}$$

where $\kappa(k) > 0$ is the step size at the k-th iteration, $\eta(k)$ is the Euclidean norm of the gradient at the k-th iteration, and P is the projection to the convex set $\mathcal{S}_{\boldsymbol{\lambda}}$. Note that the index k of iteration should not be confused with the time slot for scheduling. We assume a *time scale separation*, where rate update happens in a more coarse time scale than scheduling, so that there is sufficient time for the scheduling policy to steer the clients and enforce the efficient delay allocation rule. [18] has shown that the projected gradient method converges to the unique optimal solution, and therefore also converges to the Nash Equilibrium.

Theorem 3. *If $\kappa(k)$ satisfies $\sum_{k=0}^{\infty}\kappa(k) = \infty$ and $\sum_{k=0}^{\infty}\kappa^2(k) < \infty$, then the projected gradient method either stops at some iteration k, or the infinite sequence $\{\boldsymbol{\lambda}(k)\}$ generated by the method converges to the optimal point.*

Note that stopping at some iteration k means the method reaches the optimality in finite steps. However, the projected gradient method is a centralized algorithm. In particular, calculating the projection $\boldsymbol{\lambda}(k+1) = P(\hat{\boldsymbol{\lambda}}(k+1))$ requires the knowledge of all elements in $\hat{\boldsymbol{\lambda}}(k+1)$. Below, we propose a distributed rate control protocol that is inspired by the projected gradient method.

Since

$$\frac{\partial}{\partial\lambda_i}[\Lambda C(\Lambda)] = \frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}\frac{\partial\Lambda}{\partial\lambda_i} = \frac{\mathrm{d}}{\mathrm{d}\Lambda}[\Lambda C(\Lambda)],$$

$\hat{\boldsymbol{\lambda}}(k+1)$ can be acquired by each client updating its own request rate:

$$\hat{\lambda}_i(k+1) = \lambda_i(k) + \frac{\kappa(k)}{\eta(k)}\left[U_i'(\lambda_i(k)) - \frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}\right].$$

Note that, to facilitate the update, the server only needs to broadcast the value of $\kappa(k), \eta(k)$ and $\frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}$ in each iteration to all clients.

To ensure that $\boldsymbol{\lambda}(k+1)$ satisfies both constraints $\Lambda(k+1) \leq (1-\epsilon)\mu$ and $\lambda_i(k+1) \geq \lambda_\delta$, each client i further chooses

$$\lambda_i(k+1) = \min\{\max\{\hat{\lambda}_i(k+1), \lambda_\delta\}, \lambda_i(k)\frac{(1-\epsilon)\mu}{\Lambda(k)}\}.$$

This step ensures that $\lambda_\delta \leq \lambda_i(k+1) \leq \lambda_i(k)\frac{(1-\epsilon)\mu}{\Lambda(k)}$, and therefore $\Lambda(k+1) \leq \Lambda(k)\frac{(1-\epsilon)\mu}{\Lambda(k)} = (1-\epsilon)\mu$. We also note that, to facilitate this step, the server only needs to broadcast the value of $\Lambda(k)$ in each iteration.

The complete distributed protocol is summarized in Protocol 1. Compared with the centralized method, our distributed protocol is more scalable and lightweight, since it utilizes the broadcast nature of wireless channel and requires less resource of the server and the channel.

Protocol 1: Distributed rate control protocol

Server: on convergence of relative queue lengths:

1. $k \leftarrow k+1$
2. Broadcast $\Lambda(k), \kappa(k), \eta(k)$, and $\frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}$

Client i: on reception of server broadcast message:

1. Update: $\hat{\lambda}_i \leftarrow \lambda_i + \frac{\kappa(k)}{\eta(k)}\left[U_i'(\lambda_i) - \frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}\right]$
2. Projection: $\lambda_i \leftarrow \min\{\max\{\hat{\lambda}_i, \lambda_\delta\}, \lambda_i\frac{(1-\epsilon)\mu}{\Lambda}\}$

We conjecture that our distributed protocol also converges to the Nash Equilibrium. This conjecture will be verified by simulations in the next section.

Conjecture 1. *If $\kappa(k)$ satisfies $\sum_{k=0}^{\infty}\kappa(k) = \infty$ and $\sum_{k=0}^{\infty}\kappa^2(k) < \infty$, then the distributed rate control protocol either stops at some iteration k, or the infinite sequence $\{\boldsymbol{\lambda}(k)\}$ generated by the protocol converges to the Nash Equilibrium of the system.*

VII. Simulations

In this section, we evaluate the performance of our overall design via simulations. Specifically, we validate the polynomial approximation assumption for average delay functions, the state space collapse behavior of relative queue lengths through the MRQ scheduling policy, and the optimality of our distributed rate control protocol. For comparison, we also consider a baseline mechanism with the classic first-in-first-out (FIFO) policy for scheduling and centralized projected gradient method for rate control. Note that with FIFO scheduling, each client experiences the same average delay, i.e. $D_i(\lambda_i, \lambda_{-i}) = C(\Lambda)$.

In our simulation, we consider two systems each with $N = 10$ clients and one server. Both systems have Poisson arrivals of requests from all clients. The service time distribution of one system is exponential, and the other is deterministic. Hence, the two systems correspond to an M/M/1 queue and an M/D/1 queue respectively. Each system has an average service rate $\mu = 1 \times 10^3\,\mathrm{s}^{-1}$ and an initial total average request rate $\Lambda = 0.99\mu = 0.99 \times 10^3\,\mathrm{s}^{-1}$. We round up all inter-arrival times between two consecutive requests and service times of requests to the nearest microsecond. We make a scheduling decision every microsecond.

A. Polynomial Approximation

First, we evaluated the assumption that the average delay function can be well approximated by a polynomial $C(\Lambda)$. There are two methods to obtain the average delay function: One is via the theoretical formula, and the other is via simulation. Here, we use the first method. For the M/M/1 queue, the theoretical average delay function is:

$$\bar{C}(\Lambda) = \frac{1}{\mu - \Lambda}.$$

For the M/D/1 queue, it is:

$$\bar{C}(\Lambda) = \frac{1}{\mu} + \frac{\Lambda}{2\mu(\mu-\Lambda)}.$$

In our simulation, we fit $\bar{C}(\Lambda)$ with ten samples in our most interested heavy traffic region, where $\Lambda/\mu \in [0.95, 0.995]$, to get the polynomial $C(\Lambda)$. Recall that the total disutility in terms of total average queue length is $\Lambda C(\Lambda)$. The total disutility functions before and after approximation are compared in Fig. 1, labeled as "Theory" and "Approx" respectively. We can

observe that the polynomial approximation fits the theoretical functions very well. In fact, the order of the polynomial $C(\Lambda)$ is as small as six, and the largest relative error of the approximation is only about 2.66%.

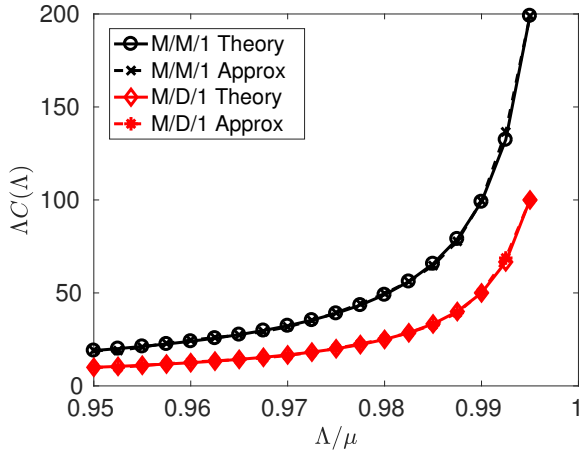


Fig. 1. Polynomial approximation of the total disutility functions.

B. Scheduling Policy

We implemented our MRQ scheduling policy and validated the state space collapse behavior in the simulation. We use a new metric, *the relative difference of queue lengths*, defined as:

$$\left(\max_i \frac{Q_i(t)}{g_i} - \min_i \frac{Q_i(t)}{g_i}\right) \Big/ \sum_i \frac{Q_i(t)}{g_i}$$

to evaluate the state space collapse performance. Theorem 2 has shown that, given the target queue length g_i of each client i, our MRQ policy ensures that the relative difference of queue lengths converges to 0 in the heavy traffic regime.

Fig. 2 shows the evolution of the relative difference of queue lengths for both systems for two sets of initial request rates, "Same rate" and "Diff rates". "Same rate" means all ten clients have the same request rate $\lambda = \Lambda/N = 99\,\mathrm{s}^{-1}$, while in "Diff rates" we have two groups of request rates: $\lambda_i = 99.6\,\mathrm{s}^{-1}$ for $i = 1, 2, \ldots, 5$ and $98.4\,\mathrm{s}^{-1}$ for $i = 6, 7, \ldots, 10$. We initialize the queue length of client i to be i^2 to exhibit the convergence of relative queue lengths more clearly. We can see that the relative difference of queue lengths converges to 0 quickly for each scenario.

C. Nash Equilibrium

Furthermore, we evaluated our distributed rate control protocol in the simulation. We set the utility functions for both systems to be $U_i(\lambda_i) = \alpha w_i \log \lambda_i$, where $\alpha = 1000$ is the common scaling coefficient for all clients, and w_i's are different weights for different clients. We set the weights to be in two groups: $w_i = 1 - 5 \times 10^{-4}$ for $i = 1, 2, \ldots, 5$ and $1 + 5 \times 10^{-4}$ for $i = 6, 7, \ldots, 10$. Therefore, the evolution of request rates of all the clients can be captured by those of Client 1 and Client 10. For the step size, we let $\kappa(k) = 1/k$ for all k.

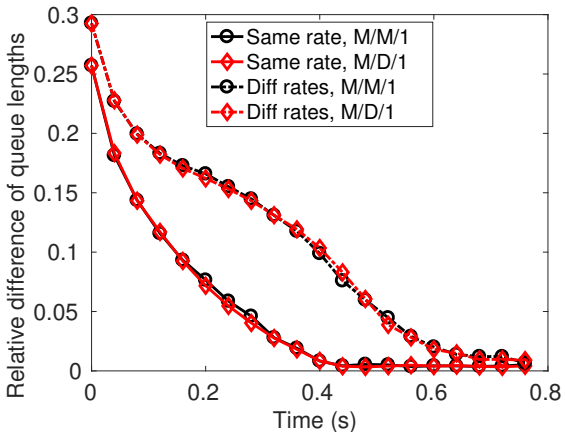


Fig. 2. State space collapse of relative queue lengths.

Fig. 3a and Fig. 3b shows the rate convergence performance for the two systems respectively. We can see that for each system, the request rates converge to two distinct values after tens of iterations. Observe that the distributed rate control protocol ("Dist" in the figure) has almost the same rate updates as the projected centralized gradient method ("Cent" in the figure). It validates our conjecture that the distributed rate control protocol achieves the Nash Equilibrium of the system.

Fig. 4a and Fig. 4b shows the convergence performance in terms of total net utility for the two systems. The total net utility settles down quickly with our distributed protocol ("MRQ, Dist" in the figures), and the evolution is again almost the same as the centralized method ("MRQ, Cent" in the figures). In these figures we also plot the performance of the baseline mechanism with the FIFO scheduling policy for comparison. We can see that the baseline mechanism also makes the total net utility converge. However, it converges to a suboptimal value, which indicates that the delay allocation rule of the baseline mechanism is not efficient.

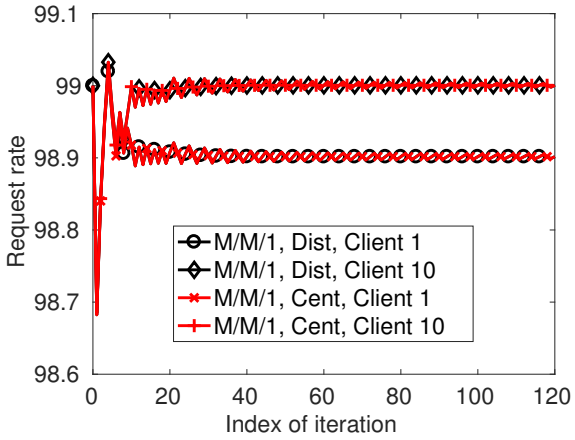
VIII. Conclusions

We have presented our non-monetary mechanism for optimal rate control through efficient delay allocation. First, we give our delay allocation rule and prove its efficiency based on multinomial expansion. Then we propose our MRQ scheduling policy that can enforce the delay allocation rule effectively in the heavy traffic regime. Furthermore, we design a distributed rate control protocol which can lead the system to Nash Equilibrium. Finally, simulation results validate the optimality of our mechanism and depict its convergence efficiency. We consider to further study VIP clients in future work.

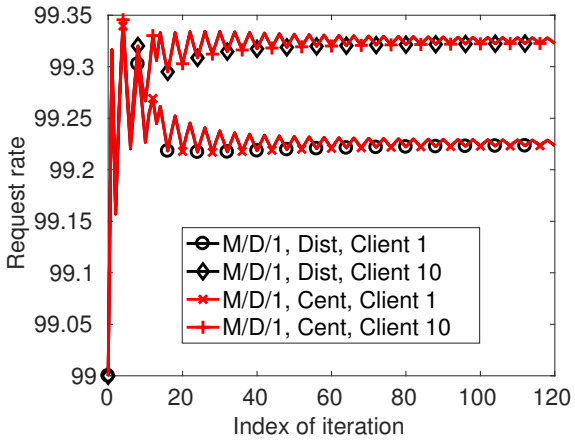
References

[1] Cisco and/or its affiliates, "Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020," Cisco, White Paper, Feb. 2016.

[2] I.-H. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Networking and Computing*. Chicago, Illinois, USA: ACM, 2010, pp. 31–40.

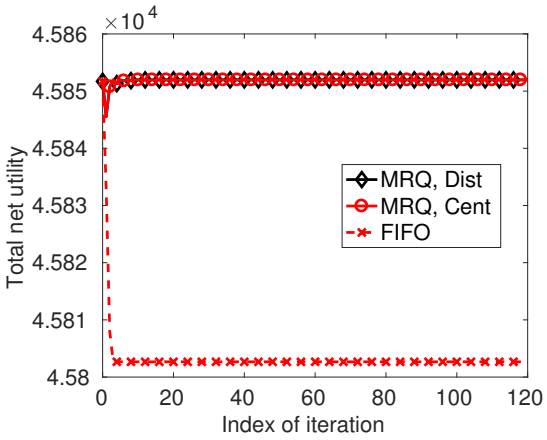


(a) M/M/1 system

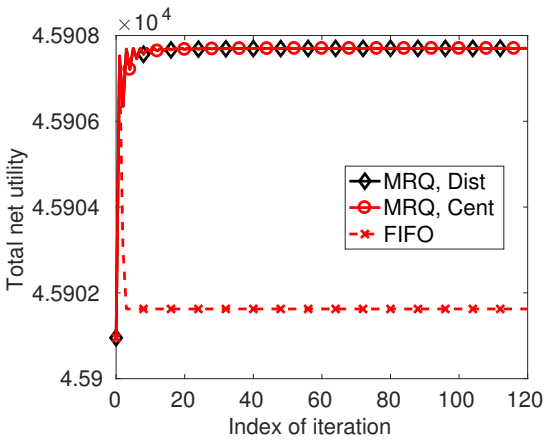


(b) M/D/1 system

Fig. 3. Simulation results to validate rate convergence.



(a) M/M/1 system



(b) M/D/1 system

Fig. 4. Simulation results to validate utility convergence.

[3] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[4] K. Ray and M. Goldmanis, "Efficient cost allocation," *Management Science*, vol. 58, no. 7, pp. 1341–1356, 2012.

[5] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Computers & Operations Research*, vol. 33, no. 2, pp. 286–311, 2006, special issue on Game Theory: Numerical Methods and Applications.

[6] T. Alpcan and T. Başar, "A utility-based congestion control scheme for Internet-style networks with delay," in *IEEE INFOCOM 2003*, vol. 3, Mar. 2003, pp. 2039–2048.

[7] R. Gupta, L. Vandenberghe, and M. Gerla, "Centralized network utility maximization over aggregate flows," in *2016 14th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2016.

[8] V. Ramaswamy, D. Choudhury, and S. Shakkottai, "Which protocol? Mutual interaction of heterogeneous congestion controllers," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 457–469, Apr. 2014.

[9] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[10] L. A. Grieco and S. Mascolo, *TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks*. Springer, Berlin, Heidelberg, 2002, vol. 2334, pp. 130–146.

[11] ——, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 25–38, Apr. 2004.

[12] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006.

[13] T. Groves, "Incentives in teams," *Econometrica*, vol. 41, no. 4, pp. 617–631, 1973.

[14] T. Baldenius, S. Dutta, and S. Reichelstein, "Cost allocation for capital budgeting decisions," *The Accounting Review*, vol. 82, no. 4, p. 837, 2007.

[15] H. Moulin and S. Shenker, "Serial cost sharing," *Econometrica*, vol. 60, no. 5, pp. 1009–1037, 1992.

[16] M. V. Rajan, "Cost allocation in multiagent settings," *The Accounting Review*, vol. 67, no. 3, pp. 527–545, 1992.

[17] A. Eryilmaz and R. Srikant, "Asymptotically tight steady-state queue length bounds implied by drift conditions," *Queueing Systems*, vol. 72, no. 3, pp. 311–359, 2012.

[18] Y. I. Alber, A. N. Iusem, and M. V. Solodov, "On the projected subgradient method for nonsmooth convex optimization in a Hilbert space," *Mathematical Programming*, vol. 81, no. 1, pp. 23–35, 1998.