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

Tao Zhao[1] Korok Ray[2] I-Hong Hou[1]

[1]Department of ECE
Texas A&M University

[2]Mays School of Business
Texas A&M University

WiOpt'17

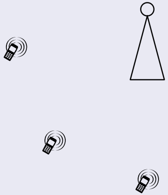
Motivation

Optimal Rate Control

- A wireless network with multiple clients
- Individual utility: function of request arrival rate
- Problem: Find optimal rates that maximize total utility

Game theory is needed.

- Clients: selfish and strategic
- Individual utility: private



Motivation

Existing Work

- Auction: e.g. VCG auction
- Direct payment between client and server

Issues of Monetary Mechanisms

- Monetary exchange requires addtional infrastructure.
- Pricing every packet? Impractical.

Motivation

Existing Work

- Auction: e.g. VCG auction
- Direct payment between client and server

Issues of Monetary Mechanisms

- Monetary exchange requires addtional infrastructure.
- Pricing every packet? Impractical.

Non-monetary mechanism!

How Non-Monetary?

Observation

- Each client suffers disutility based on experienced delay.
- Server can control delay by scheduling.

How Non-Monetary?

Observation

- Each client suffers disutility based on experienced delay.
- Server can control delay by scheduling.

Our Approach

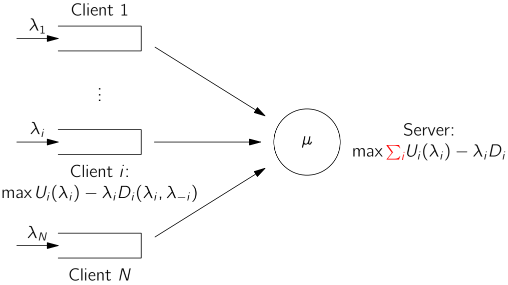
Use delay as the currency!

Main Contribution

A non-monetary mechanism by efficient delay allocation

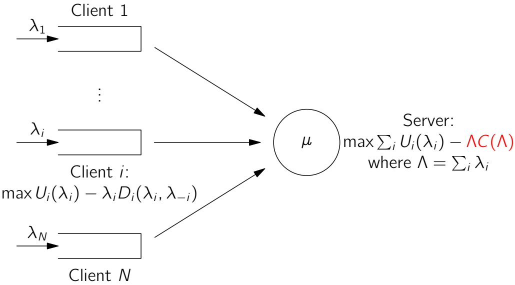
System Model

- One server: Average request service rate μ
- Client $i = 1, 2, \ldots, N$:
  - Average request arrival rate λ_i: adjustable
  - Utility $U_i(\lambda_i)$: increasing, twice differentiable, concave
  - Average request delay $D_i(\lambda_i, \lambda_{-i})$

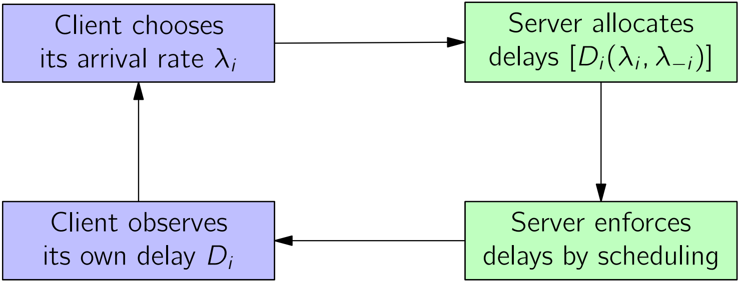


System Model

- Total average delay
  - Function of total average request arrival rate, $\Lambda := \sum_i \lambda_i$
  - Increasing and convex
  - Fitted by a $(N-2)$-order polynomial $C(\Lambda)$
- Assume feasible $\boldsymbol{\lambda} := [\lambda_i]$ satisfies $\Lambda < (1-\epsilon)\mu, \lambda_i > \lambda_\delta > 0$



Game Between Clients and Server



Nash Equilibrium and Efficiency

Definition

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

Definition

A rule of allocating delays, $[D_i(\cdot)]$, is said to be efficient if the vector that maximizes the total net utility, $\boldsymbol{\lambda}^* := [\lambda_i^*]$, is the only Nash Equilibrium.

Remark

Server's problem is to find and enforce the rule that allocates delays, $[D_i(\cdot)]$, to induce optimal choices of $[\lambda_i]$.

Non-Monetary Mechanism for Optimal Rate Control

1. Efficient Delay Allocation Rule
2. Scheduling Policy to Enforce Allocated Delays
3. Distributed Rate Control Protocol

Non-Monetary Mechanism for Optimal Rate Control

1. Efficient Delay Allocation Rule
2. Scheduling Policy to Enforce Allocated Delays
3. Distributed Rate Control Protocol

Property of Efficient Delay Allocation Rule

Server

$\boldsymbol{\lambda}^*$ is the solution to

$$\max \sum_i U_i(\lambda_i) - \Lambda C(\Lambda).$$

Hence,

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

Client

$\boldsymbol{\lambda}^*$ is the solution to

$$\max U_i(\lambda_i) - \lambda_i D_i(\lambda_i, \lambda_{-i}^*).$$

Hence,

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

Observation

Want $\Lambda C(\Lambda) - \lambda_i D_i(\lambda_i, \lambda_{-i}) =: R_i(\lambda_{-i})$, the external disutility, independent of λ_i

Delay Allocation Rule

Delay Allocation Rule

- $\lambda_i D_i(\lambda_i, \lambda_{-i}) = \Lambda C(\Lambda) - R_i(\lambda_{-i})$
- $R_i(\lambda_{-i}) = \sum_{j=1}^{N-1} \beta_i^j$
- $\beta_i^j = c_j \sum_{\boldsymbol{p} \in P_i^j} \frac{N-1}{N-G(\boldsymbol{p})} \frac{j!}{p_1! \cdots p_N!} \lambda_1^{p_1} \cdots \lambda_N^{p_N}$
- c_j: j-th order coefficient of polynomial $\Lambda C(\Lambda)$
- $P_i^j := \{\boldsymbol{p} = [p_n] \mid p_n \in \mathbb{Z}^*, \sum_{i=1}^{N} p_n = j, p_i = 0\}$
- $G(\boldsymbol{p})$ be the number of nonzero coordinates of $\boldsymbol{p}$

Theorem

Our rule of delay allocation $[D_i(\cdot)]$ is efficient.

An Example of Delay Allocation Rule

Example ($N = 3$)

β_i^j	$j = 1$	$j = 2$
$i = 1$	$c_1(\lambda_2 + \lambda_3)$	$c_2(\lambda_3^2 + 4\lambda_2\lambda_3 + \lambda_2^2)$
$i = 2$	$c_1(\lambda_1 + \lambda_3)$	$c_2(\lambda_3^2 + 4\lambda_1\lambda_3 + \lambda_1^2)$
$i = 3$	$c_1(\lambda_2 + \lambda_1)$	$c_2(\lambda_1^2 + 4\lambda_2\lambda_1 + \lambda_2^2)$

- External disutility R_i (row sum) is independent of λ_i
- Allocated disutility $\lambda_i D_i = \Lambda C(\Lambda) - R_i$
- Total disutility $\sum_i \lambda_i D_i = 3\Lambda C(\Lambda) - \sum_i R_i = \Lambda C(\Lambda)$

Non-Monetary Mechanism for Optimal Rate Control

1. Efficient Delay Allocation Rule
2. Scheduling Policy to Enforce Allocated Delays
3. Distributed Rate Control Protocol

Scheduling Policy

Problem

How to enforce target delay $D_i(\lambda_i, \lambda_{-i})$ for client i?

MRQ Scheduling Policy

Let $Q_i(t)$ be the queue length of client i at time t, and $g_i := \lambda_i D_i$. At time t, the MRQ policy schedules the client with the maximum relative queue length, defined as $Q_i(t)/g_i$.

Intuition

Eventually all relative queue lengths are equal on average in steady state, or equivalently, average queue length (delay) = target queue length (delay).

State Space Collapse

Theorem (State Space Collapse)

The efficient delay allocation rule is enforced by the MRQ scheduling policy in the heavy traffic regime.

Remark

- Heavy traffic: $\Lambda \to \mu$
- Show the deviation of the limiting queue length vector from the target queue length vector approaches 0
- Lyapunov drift based technique

Non-Monetary Mechanism for Optimal Rate Control

1. Efficient Delay Allocation Rule

2. Scheduling Policy to Enforce Allocated Delays

3. Distributed Rate Control Protocol

How Distributed?

We already know

- Our delay allocation rule is efficient.
- Our MRQ scheduling policy enforces the delay allocation rule.

Problem

How are the clients supposed to update their request rates distributedly to converge to the Nash Equilibrium?

Idea

- Projected gradient method: Centralized
- How to make it distributed?

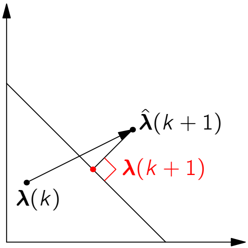
Centralized $\rightarrow$ Distributed

- Centralized update:

$$\hat{\boldsymbol{\lambda}}(k+1) = \boldsymbol{\lambda}(k) + \frac{\kappa(k)}{\eta(k)} \nabla \left[\sum U_i(\lambda_i) - \Lambda C(\Lambda) \right],$$

$$\boldsymbol{\lambda}(k+1) = P(\hat{\boldsymbol{\lambda}}(k+1))$$

- $\kappa(k)$: step size at the k-th iteration
- $\eta(k)$: Euclidean norm of the gradient
- P: projection to the feasible region s.t. $\lambda_i > \lambda_\delta$ and $\Lambda < (1-\epsilon)\mu$



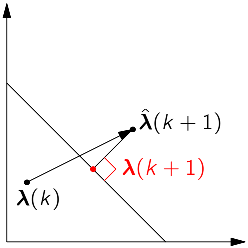
Centralized → Distributed

- Distributed update:

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

$$\boldsymbol{\lambda}(k+1) = P(\hat{\boldsymbol{\lambda}}(k+1))$$

- $\kappa(k)$: step size at the k-th iteration
- $\eta(k)$: Euclidean norm of the gradient
- P: projection to the feasible region s.t. $\lambda_i > \lambda_\delta$ and $\Lambda < (1-\epsilon)\mu$



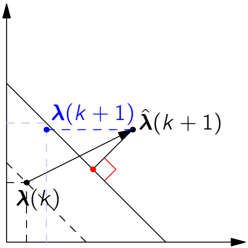
Centralized → Distributed

- Distributed update:

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

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

- $\kappa(k)$: step size at the k-th iteration
- $\eta(k)$: Euclidean norm of the gradient



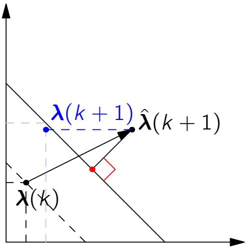
Centralized → Distributed

- Distributed update:

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

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

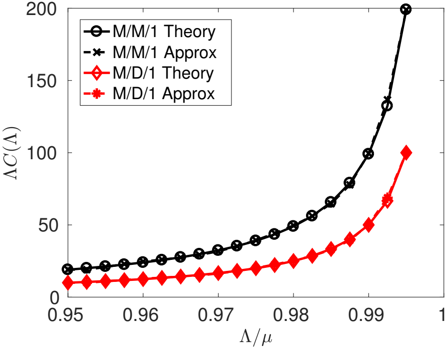
- $\kappa(k)$: step size at the k-th iteration
- $\eta(k)$: Euclidean norm of the gradient
- $\Lambda(k), \kappa(k), \eta(k)$, and $\frac{\mathrm{d}[\Lambda C(\Lambda)]}{\mathrm{d}\Lambda}$ are the same for all clients: Broadcast!



Simulations

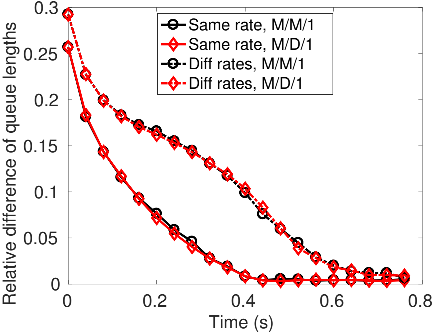
- Validate our non-monetary mechanism
  - Polynomial approximation assumption
  - State space collapse in scheduling
  - Optimality of distributed rate control protocol
- Baseline mechanism
  - FIFO (first-in-first-out) scheduling policy
  - Centralized projected gradient method for rate control
- Two systems: M/M/1 v.s. M/D/1
  - $N = 10$ clients
  - Poisson arrivals: $\Lambda = 0.99 \times 10^3\,\mathrm{s}^{-1}$
  - Exponential/Deterministic service time: $\mu = 1 \times 10^3\,\mathrm{s}^{-1}$

Polynomial Approximation



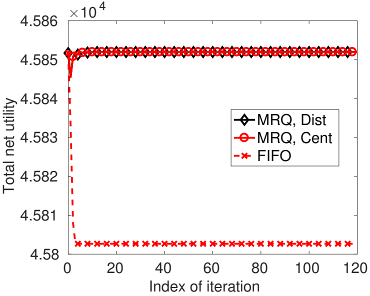
Total disutility $\Lambda C(\Lambda)$ v.s. Normalized total request rate Λ/μ

State Space Collapse

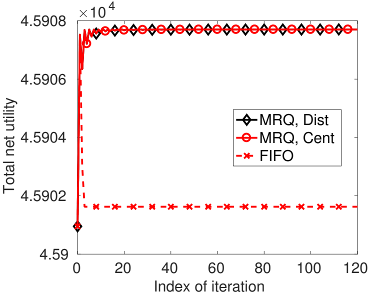


Normalized difference of relative queue lengths v.s. Time

Nash Equilibrium



M/M/1 system



M/D/1 system

Summary

Non-Monetary Mechanism for Optimal Rate Control

- Efficient delay allocation rule
- MRQ scheduling policy
- Distributed rate control protocol

Summary

Non-Monetary Mechanism for Optimal Rate Control

$$\text{Delay} = \text{Currency}$$
$$\text{Time} = \text{Money}$$

Thank you!

Tao Zhao
alick@tamu.edu

TEXAS A&M UNIVERSITY