Stability and Learning in Strategic Queuing Systems

Éva Tardos

Cornell University

Joint work with Jason Gaitonde (Cornell)

Example of a repeated game: traffic routing

- Traffic subject to congestion delays
- cars and packets follow shortest path
- Congestion game =cost (delay) depends only on congestion on edges

Learning in Repeated Games

- Agents play fixed game (bidding in auctions, routing with delay)
- Selfishly aim to minimize own sum of costs over time
- Model agents as using no-regret learning algorithms
  - Simple and efficient algorithms achieve no-regret: Hannan consistent [Hannan'57], multiplicative weights [Freund-Schapire '97], follow-the-perturbed-leader [Kalai-Vempala '03], etc.
  - Simple behavioral assumption: if single action would have been good to play throughout, notice it!
  - Less restrictive assumptions than being stable at a one-shot Nash
  - Some evidence that players satisfy this…

Social Welfare: Price of Anarchy and Learning

- Price of Anarchy [Koutsoupias-Papadimitriou '99]: "how does social cost of Nash outcome compare to social optimum?"
  - 4/3 in affine routing delays [Roughgarden-T '03],
  - ½ in valid utility games [Vetta '02], many others…
- Bicriteria Results: "cost of equilibrium of nonatomic flow is at most optimal social cost with twice the amount of flow" [Roughgarden-T '03]

Quantitative bounds (i.e. price of anarchy) on quality of Nash outcomes often extend directly to learning outcomes [Blum, et al '08; Roughgarden '09; Lykouris, et al '16]

Social Welfare of Learning Outcomes

Critical Assumption: new copy of the same game is repeated (no carryover effect between rounds other than through learning)

Is this reasonable?

Large population games: traffic routing

Morning rush-hour traffic

No carryover effect (except through the learning of the agents)

Second-by-second packet traffic

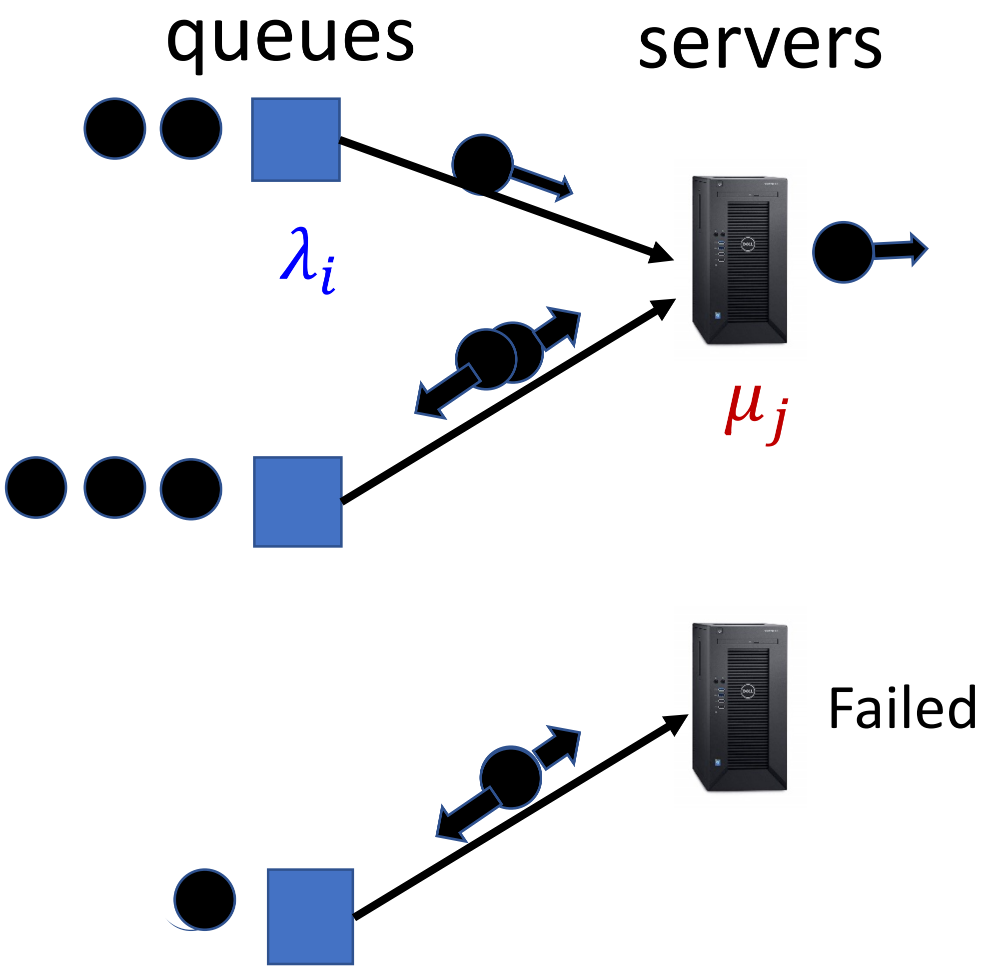
Packets take time to clear, dropped packets need to be resent in the next round

This work: what can we say about quality of competitive, learning outcomes in repeated games with carryover?

We study this question in a natural queuing setting.

Model of Learning in a Queuing System

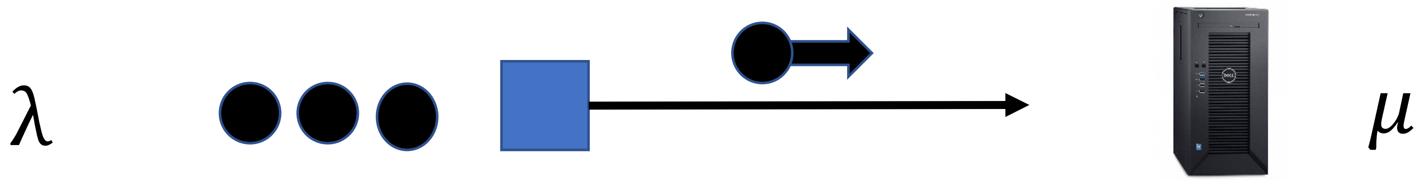
- Queue i gets new packets with a Bernoulli process with rate λ_i
- Server j succeeds at serving a packet with probability μ_j
- Each time step: each queue can send **one** packet to one of the servers to try to get serviced
- **Server can process at most one packet and unserved packets get returned to queue**
- Queues use no-regret learning to selfishly get the best service



Our Main Question

How large should the server capacity be to ensure competitive, no-regret queues remain bounded in expectation over time?

- Example: one queue, one server (no learning, no competition)



- $\lambda < \mu$: expected queue size bounded (biased r.w. on the half-line)
- $\lambda = \mu$: expected queue size grows like $\Theta(\sqrt{t})$ (unbiased r.w.)
- $\lambda > \mu$: expected queue size grows linearly in t ➔ sharp threshold

One queue many servers

- The one queue faces a Bayesian multi-arm bandit learning problem to find the best server

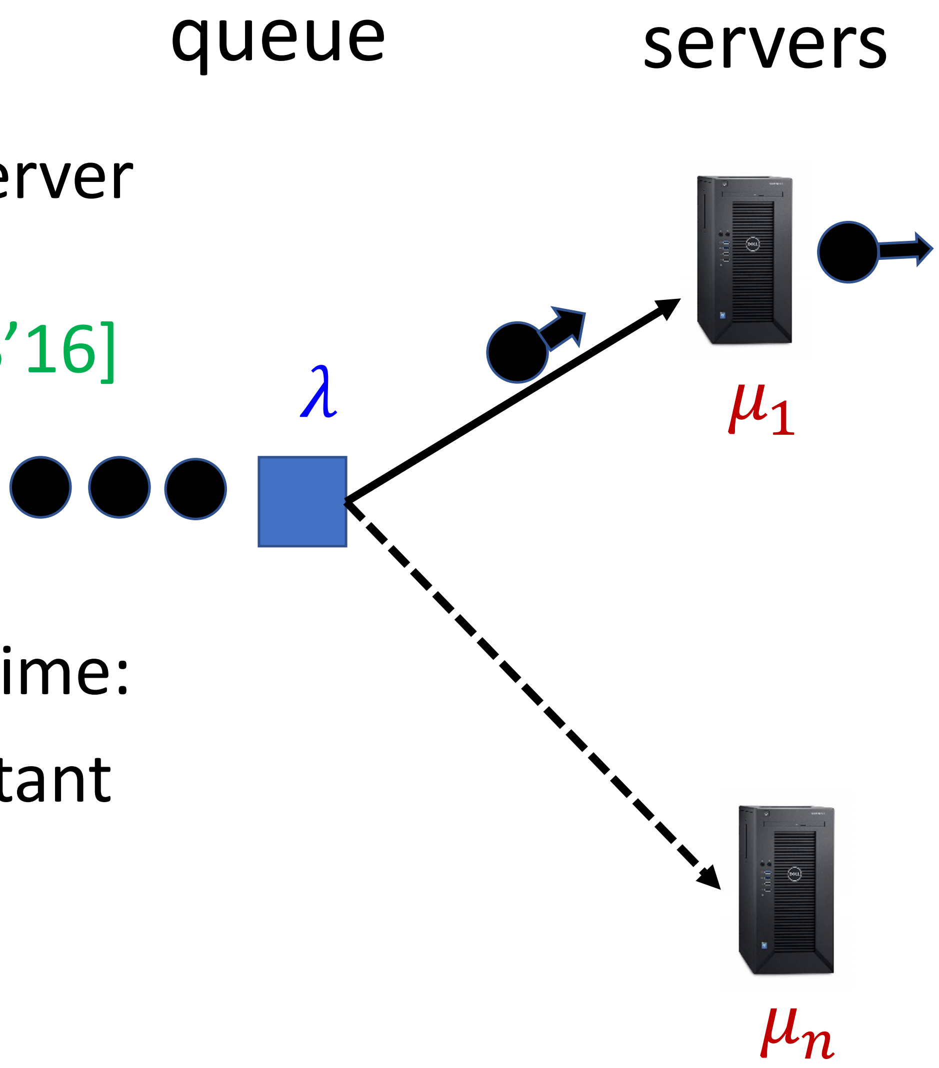
[Krishnasamy, Sen, Johari, & Shakkottai NIPS'16]

- Queue is searching for the best server:

needs $\lambda < \mu_i$

- Study the evolution of queue length over time:

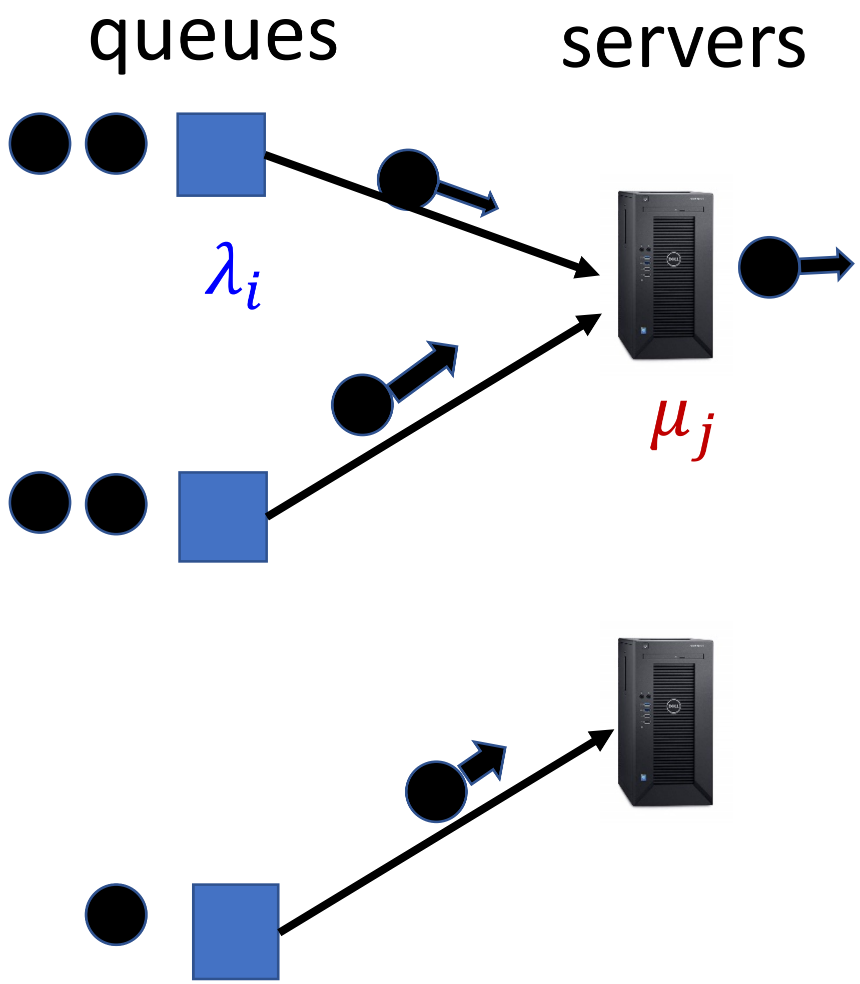
goes up to $O(\log t)$ and then back to a constant once the best server is identified



Many queues, many servers and learning

Today learning in game:

- non-cooperative, selfish play and
- carry-over effect

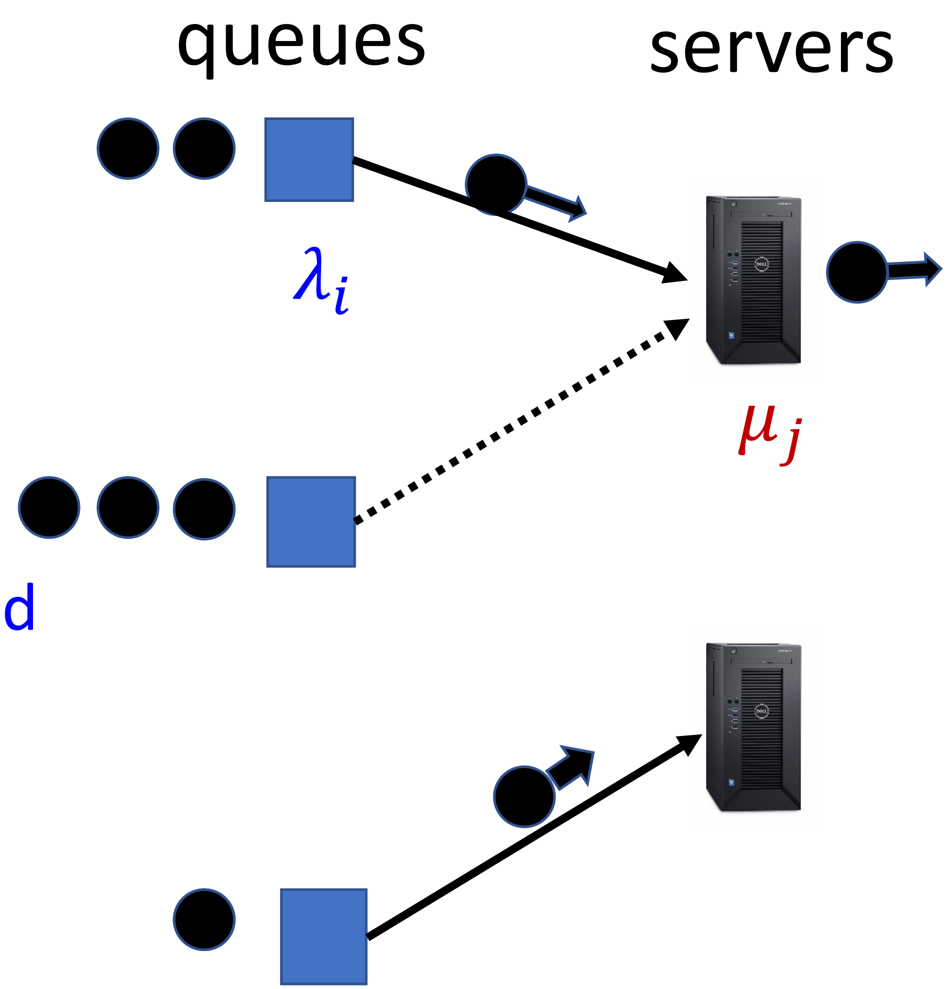


Baseline Measure: Coordinated Queues

Assume queues and servers are sorted:

$$1 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

$$1 \geq \mu_1 \geq \mu_2 \geq \cdots \geq \mu_m > 0$$



Claim: necessary/sufficient condition for centralized stability: for all *k*,

$$\sum_{i=1}^{k} \lambda_i < \sum_{i=1}^{k} \mu_i$$

(Recall: can only send one packet each round)

How Do Servers Choose Between Packets?

- Option 1: uniformly random
- Option 2: oldest first

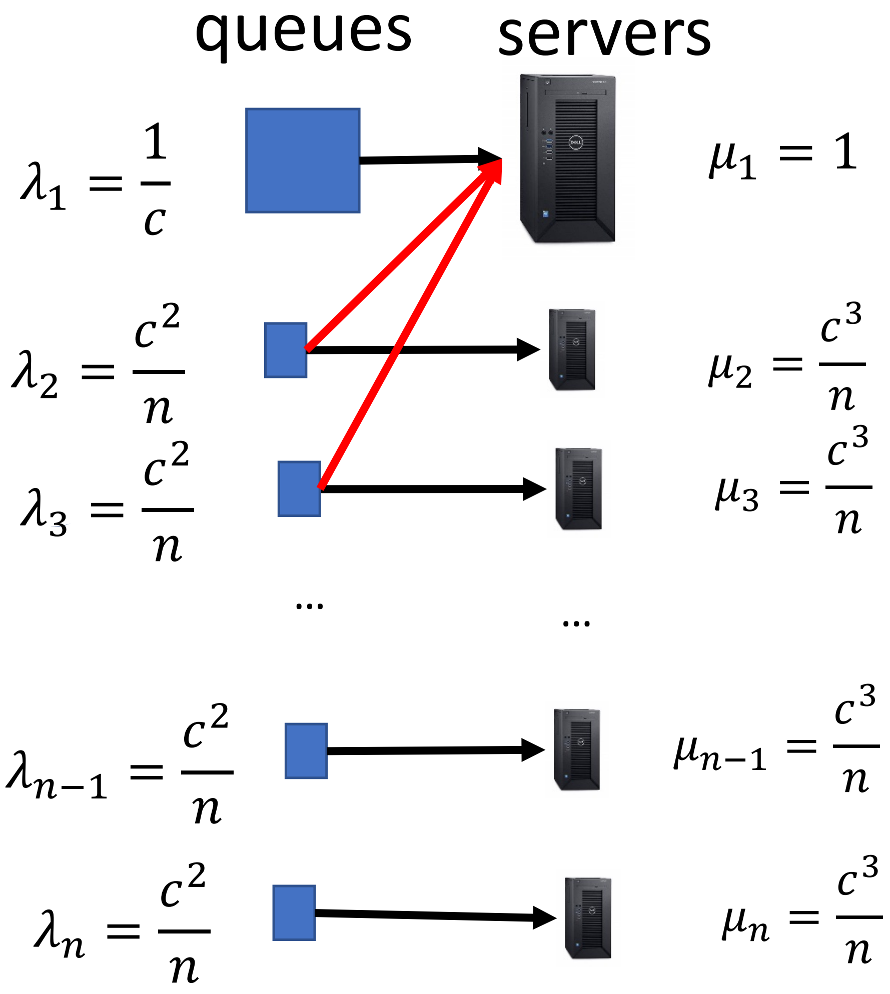
Main Results [Gaitonde-T ’20]

- Uniformly random: selfishness need not help coordinate queues, unless prohibitively larger service rates
- Oldest first: selfish learning helps coordinate so long as service rate is at least twice the arrival rate, i.e. for all k,

$$\sum_{i=1}^{k} \lambda_i < \frac{1}{2} \sum_{i=1}^{k} \mu_i$$

Why Uniform Selection Fails

- One big queue/server and many small queues w/ matching servers ➔ slack $c > 1$
- Simple coordinated strategy: send to own server!
- But: small queues can saturate large server ➔ big queue cannot clear!



Selfish Queuing with Priorities

- Main Theorem [informal, Gaitonde-T '20]: suppose that:
  - Servers attempt to serve oldest packet received in each round,
  - Queues use no-regret learning algorithms,
  - and for all k,

$$\sum_{i=1}^{k} \lambda_i < \frac{1}{2} \sum_{i=1}^{k} \mu_i$$

  Then, **all queue sizes remain bounded** in expectation uniformly over time. Moreover, factor 1/2 is tight.

Proof Ideas

- Use potential function

$$\Phi \approx \sum_\tau \Phi_\tau$$

with $\Phi_\tau = \#$ packets aged τ or older in the system

- [Pemantle, Rosenthal '04]: random process satisfying
  i. Sufficiently regular
  ii. Negative drift when large

  remains bounded in expectation for all times

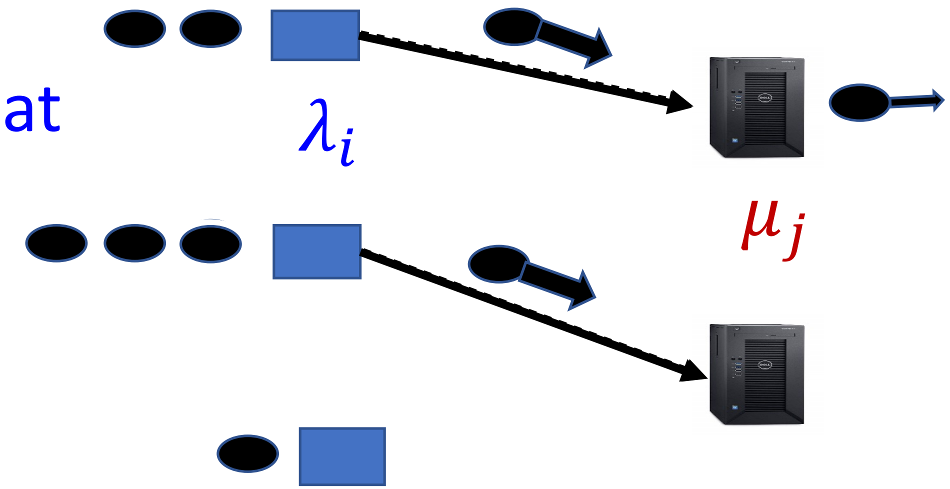
- No-regret + factor 2 slack implies negative drift when queues have large backup

Why Φ and How No-Regret Helps

- Look at queues with packets at least τ-old; they have priority
- Fix long window and look at best/fastest servers
- Either: i) many τ-old queues send there throughout window ➔ decrease in queue size, OR

  ii) they do not ➔ had priority there so no-regret kicks in:

  Any queue with τ-old packets would have regret, unless it managed to get service for at least this much!



Apply at all thresholds τ simultaneously to get no-regret at all scales ➔ implies negative drift

Extra Technical Details

- Need no-regret to hold on specific windows of long enough size with high-probability

  unlikely bad situations will happen, need to be able to recover

- Other technical issues for applying Pemantle/Rosenthal result

  use model with deferred decisions: study *ages* instead of *sizes:*

  age of oldest packet T_i^t in queue i

  $\Phi_\tau = \sum_{i:T_i^t > \tau} \lambda_i (T_i^t - \tau) \approx$ # packets age τ or older in the system

- apply concentration bounds, avoid bad correlations for the analysis,
- "sufficiently regular" = bounded moments

Summary and Future Directions

Thanks!

- Learning in games has many attractive features, but not much known on quality of outcomes in games with carryover effect
- We prove stability results of selfish learners in queuing model with strong dependencies over time via returned packets and priority
- Can these kinds of results be extended to other natural games with carryover (auctions with budgets? More complicated routing schemes/feedback structures?)?
- Upcoming work: Is this the right learning? More patience in evaluating results: $\frac{e}{e-1} \approx 1.58 \ldots$ factor is enough.