Learning in Strategic Queuing Systems with Small Buffers

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider learning outcomes in games with carryover effects between rounds: when outcomes in the present round affect the game in the future. An important example of such systems is routers in networking, as they use simple learning algorithms to find the best way to deliver packets to their desired destination. This simple, myopic, and distributed decision process makes large queuing systems easy to operate, but at the same time, the system needs more capacity than would be required if all traffic were centrally coordinated. Gaitonde and Tardos (EC 2020 and JACM 2023) initiated the study of such systems, modeling them as an infinitely repeated game in which routers compete for servers and the system maintains a state (the number of packets held at each queue) that results from outcomes of previous rounds. However, their model assumes that servers have no buffers at all, so routers have to resend all packets that were not served successfully, which makes their system model unrealistic. They show that, even with hugely increased server capacity relative to what is needed in the centrally coordinated case, ensuring that the system is stable requires the use of timestamps and priority for older packets.

We consider a system with two important changes, which make the model more realistic and allow for much higher traffic rates: first, we add a very small buffer to each server, allowing the server to hold on to a single packet to be served later (if it fails to serve it immediately), and second, we do not require timestamps or priority to older packets. Using theoretical analysis and simulations, we show that when queues are learning, a small constant-factor increase in server capacity, compared to what would be needed if centrally coordinating, suffices to keep the system stable, even if servers select randomly among packets arriving simultaneously.

1 Introduction

In this paper, we model routers in networking as agents using simple learning algorithms to find the best way to get their packets served. This simple, myopic, and distributed multi-agent decision system is an important application of multi-agent learning. Managing a networking system in such a distributed way makes large queuing systems simple to operate, but at the same time, the system needs more capacity than would be required if all traffic were centrally coordinated. In a recent paper, Gaitonde and Tardos [2020, 2023] initiated the study of such systems, modeling them as an infinitely repeated game in which routers compete for servers and the system maintains a state (number of packets held by each queue) that results from outcomes of previous rounds. Routers get to send a packet at each step at which they received a packet or have one in their queue[1] to one of the servers, and each server attempts to process only one of the packets arriving to it. However, the model of Gaitonde and Tardos [2020, 2023] (see also Sentenac *et al.* [2021]; Freund *et al.* [2023]) assumes that

[1] Players are routers with states (their queue length); the terms *router* and *queue* are used interchangeably.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

servers have no buffers at all, so queues have to resend all packets that were not served successfully. They show that, in their system, even with hugely increased server capacity relative to what is needed in the centrally-coordinated case, the use of credible timestamps and priority for older packets by all servers is required to ensure that the system is stable, i.e., that queue lengths do not diverge with time (see Section 2 for a formal definition of stability).

In networking systems, servers accepting packets typically have a very small buffer and can hold on to a few packets to be served later. In this paper, we consider the analog of the model of Gaitonde and Tardos [2020, 2023] in such systems with a tiny buffer at each server. We show that even with a buffer capable of holding only a single packet and non-coordinated learning agents, the service capacity of the system greatly increases. However, the tasks of the learning algorithms become more complex when servers have buffers. To see this, consider the case of a single queue with many possible servers. Without buffers, the queue faces a classical multi-arm bandit problem, aiming to learn which of the servers is the best one to send their packets to. By contrast, we observe that with a buffer of even just one packet at each server, a sufficiently large number of low-capacity servers will always ensure stability, as the learning agent can now take advantage of these low-capacity servers as well, sending packets to them rarely enough that they are likely to clear in time before the next packet is sent to them. This makes the goal of a learning algorithm even with just one queue and many servers more complex: it should learn to distribute its packets in a way that takes advantage of the capacity of all the servers. Interestingly, our results show that even simple bandit learning algorithms manage to learn to use low-capacity servers effectively in the presence of buffers.

Our main result, formally stated in Theorem 1, is to show that when multiple agents compete for service while using learning algorithms to identify good servers, a small constant-factor increase in server capacity—relative to what would be required under centralized coordination—suffices to maintain the system stable, even if servers randomly select among simultaneously arriving packets. Specifically, suppose that we have n queues with packet arrival rates of $\lambda_1, \ldots, \lambda_n$, and m servers of service rates $\mu_1, \ldots, \mu_m$. Obviously, we must have $\sum_i \lambda_i < \sum_j \mu_j$, else the system cannot be stable even if fully coordinated. We show that this condition is not strong enough to ensure stability even with full coordination due to the small buffers.[2] However, our result shows that if $\lambda_i < \frac{1}{2}$ for all i, all queues use any type of learning that achieves the no-regret condition with high probability, and $3\sum_i \lambda_i < \sum_j \mu_j$, then this guarantees that the system remains stable. While we do not know whether the factor of 3 in the required capacity constraint is tight, we provide a lower bound, showing that to guarantee that a no-regret outcome of learning by the queues remains stable, it is required to have at least $2\sum_i \lambda_i < \sum_j \mu_j$ (see Remark 1 and Theorem 3 for more details).

Related Work. The classical focus of work on scheduling in queuing systems is aimed at finding schedules that achieve optimal throughput (see, e.g., the textbook of Shortle *et al.* [2018]). For work evaluating efficiency loss due to selfishness in different classical queuing systems, see the book of Hassin [2020] and survey of Hassin and Haviv [2003]. Closer to our motivation, there is a growing literature that aims to understand how systems perform when the queues use simple learning algorithm to find good service. Krishnasamy *et al.* [2016] considers a queue using a no-regret learning algorithm to find what may be the best servers, but does not consider multiple queues competing for service. Their primary goal is to study the expected size of the queue of packets as a function of time. They also extend the result to the case of multiple queues scheduled by a single coordinated scheduling algorithm, assuming there is a perfect matching between queues and optimal servers that can serve them. Sentenac *et al.* and Freund *et al.* [2022] extended this work to decentralized learning dynamics in bipartite queuing systems that attain near-optimal performance without the matching assumption. For a survey on the role of learning and information in queuing systems see Walton and Xu [2021].

In our work, we diverge from the literature on queuing and scheduling and instead focus on a game-theoretic model with learning agents: queues using learning algorithms to best distribute their packets to get good service, while also repeatedly interacting with other learners and competing for servers. We assume that each queue separately learns to selfishly make sure its own packets are served at the highest possible rate, offering a strategic model of scheduling packets in a queuing system. Closest to our model from this literature is the work Gaitonde and Tardos [2020, 2023], who consider the

[2]This condition allows to create a fractional assignment such that $\sum_j x_{ij} > \lambda_i$ and $\sum_i x_{ij} < \mu_j$. Using the matching decomposition of this assignment in a coordinated schedule will guarantee a bounded expected number of packets both at the queues and servers.

same bipartite model of queues and servers as we do, but without buffers. They show the exact condition to make such a system stable with central coordination of packets and prove that no-regret learning by the queues guarantees stability of the system if it has double the capacity needed for central coordination, assuming packets carry a timestamp and servers choose the oldest packet to serve. Fu *et al.* [2022] extended this work to a general network. Baudin *et al.* [2023] propose an alternative, episodic queuing system where agents have incentives to hold jobs in an episode before sending to a central server, but suffer penalties should their jobs not be completed before the end of the episode. They show that both equilibrium and no-regret outcomes ensure stability, so long as these costs are sufficiently large.

In the works discussed so far, the servers receiving the packets have no buffers: all unserved packets are held in a queue and get resent to be served later. In queuing systems aimed at modeling networking, the receivers (servers) each do have a very small buffer, and can hold on to a few packets to be served later. It turns out that even having a buffer for a single packet significantly changes the effective service capacity of the network with learning agents, as already explained above.

In the case without such buffers, it was feasible to exactly characterize the capacity needed to be able to make the system stable with central coordination by an elegant use of linear programming duality (see Fu *et al.* [2022]). In the presence of limited buffer capacity at the servers, the rate at which a server accepts a packet depends on the state of the buffers. This is in contrast to a system without buffers, where the probability that a server accepts a packet is equal to the service rate of the server, and is stable over time. There is some recent work considering queuing systems with such evolving service probabilities, see for example Grosof *et al.* [2024b,a]. The goal of that work is to characterize the exact condition that allows the system to remain stable with central coordination of the schedule. Our work is the first to consider stability in such a system assuming each queue independently aims to optimize its own service rate running a no-regret learning algorithm.

There is a large literature on bounding the price of anarchy for various games, beginning with Koutsoupias and Papadimitriou [1999]. These results extend to analyzing outcomes in repeated games assuming players are no-regret learners Balcan *et al.* [2013]; Roughgarden [2015]; Syrgkanis and Tardos [2013]. However, these works make the strong assumption that games in different rounds are independent. By contrast, different rounds in queuing games are not independent: the number of packets in the system depends on the success in previous periods. Work on games with carryover effects, or context, are considered in multi-agent reinforcement learning as well as in Markov games, see for example Littman [1994]; Busoniu *et al.* [2008]; Zhang *et al.* [2021]. However, this line of work does not focus on the overall quality of learning outcomes. Another line of work on repeated games with such carryover effects between rounds, focusing on such overall learning outcomes, is the repeated ad-auction game with limited budgets, where the remaining budgets make the games no longer independent. See for example Balseiro and Gur [2019]; Gaitonde *et al.* [2023]; Fikioris and Tardos [2023]; Fikioris *et al.* [2024].

2 Model

We consider a more realistic version of the model used by Gaitonde and Tardos [2020, 2023]. There is a system of n queues and m servers. We divide time into discrete time steps $t = 0, 1, \ldots$. In every step t, the following occurs. See also the illustration in Figure 1.

1. Packets arrive at each queue i according to a fixed probability λ_i. Formally, we model this by defining $B_t^i \sim \text{Bern}(\lambda_i)$ as an independent random variable that indicates whether a packet arrives at queue i at time t. After arrival, each queue that contains at least one packet selects a server to which it attempts to send a single packet.

2. Each server j processes incoming packets as follows: (i) If the server's buffer is full, all received packets are rejected and returned to their respective queues. (ii) Otherwise, the server accepts one of the received packets, chosen uniformly at random, and places it in its buffer, rejecting the remaining packets. (iii) The server then attempts to process the packet in its buffer. With probability μ_j, the processing succeeds, and the packet is removed from the buffer. If the processing fails, the packet remains in the buffer.

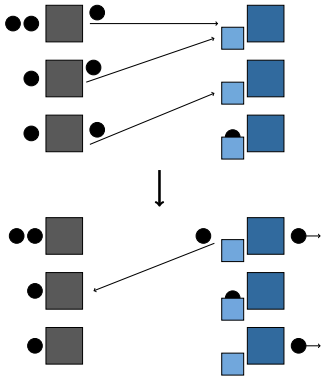


Figure 1: A system with three queues and three servers in a single time step. Discs represent packets; the left squares are queues, the right squares are servers, and the small squares are buffers. In this time step, two packets are sent to the top server. One of these packets is accepted into the buffer and then processed, while the other is returned to its queue. For the other two servers, one admits a packet into its buffer but fails to serve it, and the other processes a packet that was already in its buffer from a previous round.

3. Any packets not accepted into a server's buffer are returned to their respective queues. Each queue receives bandit feedback based on the outcome of sending its packet: a reward of 0 if the packet was not accepted by the server or 1 if the packet was successfully placed in the server's buffer.[3]

Definition 1. *We say that a system with a schedule for sending packets is* stable *if the expected number of packets waiting to be sent or served in the system is bounded by a constant at all times. We allow the constant to depend on the number of queues or servers and the parameters of the system, but it may not grow with time T.*

It is clear that if $\sum_i \lambda_i > \sum_j \mu_j$ then even a fully coordinated system cannot be stable, and the total number of packets waiting at the queues must grow linearly in time. In the case of a single queue with a single server, $\lambda < \mu$ is the necessary and sufficient condition for keeping the packets remaining in the system bounded by a constant (which depends on the gap in the inequality) at all times. The analogous condition is clearly necessary but not sufficient in the case of multiple servers.

Lemma 1. *Consider an example with 1 queue and 2 servers, each with $\mu = \frac{1}{2}$. We claim that $\lambda < \frac{23}{24} < 1$ is needed for the system to be stable even with full coordination.*

Proof. To see why this is true, consider 3 consecutive time steps. To analyze the maximum processing rate we look at the scenario where the queue sends a packet in each of these steps. We claim that with probability $1/8$ it holds that in one of the three times the queue must fail to have a packet accepted by a server. Assume that the packet sent in the first of the three periods is accepted into a buffer but not processed for two steps. This event has probability at least $\frac{1}{4}$. With coordination, a packet in the next period will be sent to the other server. Assume that this packet is not processed when it is sent, which has probability at least $\frac{1}{2}$. Now in the third period, both buffers are full, so no packet can be accepted.

This proves that during three consecutive steps, the maximum expected service rate is at most $\frac{\frac{1}{8}\cdot 2+\frac{7}{8}\cdot 3}{3} = \frac{23}{24}$. Thus, if the arrival rate λ is higher, we have a buildup of the queue linear in T. □

3 Analysis

Our main result is the following theorem. This result holds under either bandit or full feedback, and applies to arbitrary learning algorithms, as long as they achieve sublinear regret.

Theorem 1. *Assuming $\sum_i \lambda_i < \frac{1}{3}\sum_j \mu_j$, $\lambda_i < \frac{1}{2}$ for all i, and all queues use a form of learning guaranteeing no-regret with high probability to identify servers they can use, the system remains stable with the expected number of packets in the system bounded by a (time-independent) constant at all times.*

The key tool in our analysis is the following theorem of Pemantle and Rosenthal [1999]. Informally, the theorem states that a stochastic sequence that has negative drift when it grows large and is sufficiently regular (has bounded moments) is bounded by a constant at all times.

Theorem 2. *Let $X_1, X_2, \ldots$ be a sequence of nonnegative random variables with the property that*

1. *There exist constants $\alpha, \beta > 0$ such that $\mathbb{E}[X_{t+1} - X_t | \mathcal{F}_t \& X_t \geq \beta] < -\alpha$, where $\mathcal{F}_t$ denotes the history until period t.*

[3] Feedback is immediate after a packet is placed in the buffer, not necessarily when it is successfully processed.

2. There exist $p > 2$ and a constant $\theta > 0$ such that for any history, $\mathbb{E}[|X_{t+1} - X_t|^p | \mathcal{F}_t] \leq \theta$.

Then, for any $0 < r < p - 1$, there exists an absolute constant $M = M(\alpha, \beta, \theta, p, r)$ not depending on t such that $\mathbb{E}[X_t^r] \leq M$ for all t.

To use this theorem we consider a long enough time period of length T and use a potential function that depends on the number of packets in each of the queues at the start of that time period. Assume that queue i has N_i packets at the start of the period. We will use the potential function $\Phi = \sum_i (N_i - (\lambda_i^2 + 3\delta)T)^+$, where $x^+ = \max(x, 0)$, and will aim to show that between the beginning and end of a T long period the change in this potential satisfies the conditions of Theorem 1.

For T large enough, we expect that the packet arrivals and the servers' service rates are all close enough to their expectations, and each queue has small enough regret in their learning algorithm. We will call a period of length T *good*$_\delta$ for a parameter $\delta > 0$ (to be chosen later) if the following three conditions are all satisfied. Later, we will also account for what happens when *good*$_\delta$ does not occur.

Condition 1. *The number of packets arriving to each queue i in each period of length $\hat{T} \leq T$ is at most $\lambda_i \hat{T} + \delta T$.*

Condition 2. *Each server j that has a packet in its buffer (possibly one that just arrived) for at least half of the steps during this T-length period succeeds in serving at least $(\frac{1}{2}\mu_j - \delta)T$ packets.*

Condition 3. *The learning algorithm of each queue i accumulates regret at most δT during the T-length period.*

To show the expected decrease in the potential function, we consider whether a period of length T satisfied the condition of being *good*$_\delta$. We will show below (in Lemmas 4 and 5) that the probability that *good*$_\delta$ fails to hold is bounded by $(m + 2n)\eta$. The maximum possible increase in the potential function during a period of T steps is at most nT (if packets arrive during each step at each of the queues, and none reach the servers), so bad events contribute at most $(m + 2n)\eta nT$ to the expected change in potential over the T steps.

To show the decrease in the potential in a *good*$_\delta$ period of length T, the following two lemmas separately consider servers that serve packets most of the time and those that do not.

Lemma 2. *If all servers have a packet in their buffer (possibly one that just arrived) for more than half of the iterations, and we are in the* good$_\delta$ *case, the total number of packets at the queues decreases by at least $(\frac{1}{2}\sum_j \mu_j - \sum_i \lambda_i - (n + m)\delta)T$.*

Proof. By condition 2, the total number of packets served is at least $\sum_j (\frac{1}{2}\mu_j - \delta)T$. By condition 1, the total number of arriving packets is at most $\sum_i (\lambda_i + \delta)T$. Combining these two bounds establishes the lemma. □

Lemma 3. *Suppose there is a server whose buffer is empty more than half of the T iterations, and we are in the* good$_\delta$ *case. Consider a queue i and assume that $\delta < \frac{1}{2}(\frac{1}{2} - \lambda_i)$, and $\delta < \lambda_i$. Then, the number of packets at this queue will decrease, unless the number N_i' left at the end the period is at most $N_i' \leq (\lambda_i^2 + 3\delta)T$. If at the start there are at least T packets in the queue, then the number decreases by at least: $\left(\frac{1}{2} - (\lambda_i + 2\delta)\right)T$.*

Proof. Consider a queue i. Suppose that the queue starts with N_i packets and ends the period with N_i'. If the server was sending a packet in every step, then by condition 3 it cleared at least $(\frac{1}{2} - \delta)T$ packets, and by condition 1 at most $(\lambda_i + \delta)T$ arrived, decreasing the total by at least $(\frac{1}{2} - (\lambda_i + 2\delta))T$. This proves the second part of the claim, as T packets (or more) at the start will guarantee sending a packet in every step.

Now suppose that the queue has packets in the last $\hat{T}$ steps but was empty in the previous step. Let S_i denote the number of packets it successfully cleared in this $\hat{T}$-step period. Assume that some $(T - \hat{T})\lambda_i + X_i$ packets arrived during the first $(T - \hat{T})$ steps with $X_i \leq \delta T$ by Condition 1. Now by the no-regret Condition 3 we get that

$$N_i + \lambda_i(T - \hat{T}) + X_i + S_i \geq \frac{1}{2}(N_i + \lambda_i(T - \hat{T}) + X_i + \hat{T}) - \delta T,$$

which implies (using that $X_i \leq \delta T$) that $N_i \geq \hat{T} - \lambda_i(T - \hat{T}) - 2S_i - 3\delta T$.

By condition 1, the final number of packets in the queue at the end of the period is at most $N_i' \leq \lambda_i \hat{T} + \delta T - S_i$. Putting these two inequalities together we get that

$$N_i - N_i' \geq \hat{T} - \lambda_i(T - \hat{T}) - 2S_i - 3\delta T - (\lambda_i \hat{T} + \delta T - S_i) \geq \hat{T} - \lambda_i T - 4\delta T - S_i.$$

This shows that the number of packets at queue i decreases, unless $\hat{T} \leq (\lambda_i + 4\delta)T + S_i$. However, in this case we get that the number of packets at the end of the period is at most $N_i' \leq \lambda_i \hat{T} + \delta T - S_i \leq (\lambda_i^2 + 4\lambda_i\delta) + \delta)T - (1 - \lambda_i)S_i \leq ((\lambda_i^2 + 3\delta)T$, as $\lambda_i < 1/2$, proving the claim. □

To prove Theorem 1 we need to bound the probability that *good*$_\delta$ fails to holds. We use Chernoff and union bounds to bound from below the probability that conditions 1 and 2 hold when choosing a sufficiently large T.

Lemma 4. *For any constants $\delta > 0$ and $\eta > 0$, by choosing the period length T high enough, we can guarantee that the probability that conditions 1 or 2 fail to hold for a single queue or single server is bounded by η, and hence with probability $(1 - \eta(n + m))$ the conditions hold.*

To guarantee that Condition 3 holds with high probability, we will assume that queues use a learning algorithm with a high probability regret guarantee, such as EXP3.P.1, of Auer *et al.* [2002a], which offers a regret guarantee of $\sqrt{Tm \ln(mT/\eta)}$ with probability $1 - \eta$.

Lemma 5. *For any constants $\delta > 0$ and $\eta > 0$, by choosing the period length T high enough we can guarantee that if all queues use EXP3.P.1 as their learning algorithm, the probability that Condition 3 holds for a single queue is at least $1 - \eta$. Hence, the condition holds with probability $(1 - n\eta)$.*

Now we are ready to start proving the main result.

Proof. (Theorem 1) Recall the potential function $\Phi = \sum_i (N_i - (\lambda_i^2 + 3\delta)T)^+$. Clearly, Φ has bounded moments, as the maximum possible increase in the potential function over a period of length T is when a packet arrives at every single queue during each step and no packets are accepted at any of the servers. That is a total increase of nT. Similarly, the maximum possible decrease in the potential function over a period of length T is when no packets arrive, and each queue succeeds in sending a packet to a server at each step. That is a total decrease of nT.

Next, we want to prove that when $\Phi > Tn$, the potential is expected to decrease. Recall that when *good*$_\delta$ fails to hold the potential can increase by as much as nT. By Lemmas 4 and 5, this can contribute at most $\eta(m + 2n)nT$ to the expectation.

To analyze the case when *good*$_\delta$ holds, we consider two cases separately. If all servers have packets to try to serve (in their buffer, or just arriving) at least half of the time, then by Lemma 2 the total number of packets decreases by at least $(\frac{1}{2}\sum_j \mu_j - \sum_i \lambda_i - (n + m)\delta)T$. Some of this decrease may not affect the potential function, since some of the cleared packets can come from queues that contribute zero to the potential. However, this can only affect $\sum_i (\lambda_i^2 + 3\delta)T$ packets in queue i, and so the potential is decreasing by at least

$$\Big(\frac{1}{2}\sum_j \mu_j - \sum_i (\lambda_i + \lambda_i^2) - (4n + m)\delta\Big)T. \tag{1}$$

By the assumption that $\lambda_i < \frac{1}{2}$ and $\sum_j \mu_j > 3\sum_i \lambda_i$, this is an $\Omega(T)$ decrease, assuming we choose a small enough δ.

Now consider the case where a server is open (i.e., its buffer is empty, and no new packet arrived) half the time. If $\Phi > nT$ then some queue i will have at least T packets. In this case, by Lemma 3 this queue will have its packet count decrease by at least $(\frac{1}{2} - \lambda_i - 2\delta)T$, contributing this decrease to the potential function. Lemma 3 also shows that all other queues will have their contribution to the potential decrease, unless it is zero at the end of the T-length period. This shows a total decrease in the potential of at least $(\frac{1}{2} - \lambda_i - 2\delta)T = \Omega(T)$, again assuming δ is small enough.

Putting together the expected increase when *good*$_\delta$ fails and expected decrease in the two cases when *good*$_\delta$ holds (and using that $\lambda_i \leq \frac{1}{2}$), we get that the potential function is expected to decrease after a T-length period by at least

$$T \min\big(\frac{1}{2} - \lambda_i - 2\delta,\ \frac{1}{2}\sum_j \mu_j - \frac{3}{2}\sum_i \lambda_i - (4n + m)\delta\big) - T\eta(m + 2n)n.$$

To guarantee the required expected decrease of the potential, we choose δ small enough so both terms in the $\min$ are at least a constant (so in the case $good_\delta$ happens the potential decreases by $\Omega(T)$) and then choose η small enough to make the total positive. □

Remark 1. *Note that the proof actually shows a slightly stronger statement dependent on the λ_i values. Using Equation (1), it is enough to require $\sum_i(\lambda_i + \lambda_i^2) < \frac{1}{2}\sum_j \mu_j$ in place of $\sum_i \lambda_i < \frac{1}{3}\sum \mu_j$. This improves the bound when the arrival rates are small, approaching a factor of 2 in this case.*

While we do not know if the factor of 3 in the main theorem is a tight worst-case bound even when arrival rates λ_i are close to $\frac{1}{2}$ (in which case $\sum_i(\lambda_i + \lambda_i^2) = \sum_i \lambda_i(1+\lambda_i) = \frac{3}{2}\sum_i \lambda_i$ yields this factor), the following example shows that we need at least a factor of 2 to ensure stability.

Theorem 3. *[Lower Bound] Consider a queue with $1/2$ arrival rate, facing k servers, each with only capacity $1/n$. Assume that the server chooses a uniform random server at each step. This sending plan will clearly satisfy the no-regret Condition 3. We claim that we need at least n servers to make the solution stable proving a lower bound of 2 needed for the needed increase in server capacity.*

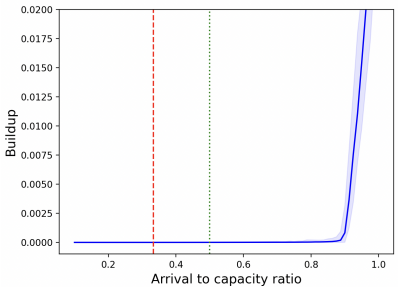
Proof. So with k servers, the chance that when the queue sends to a server, the previous time it did so was i steps ago is $1/k \cdot (1-1/k)^{i-1}$. If the previous time the queue did sent there, the chance that the server is still busy is $(1-1/n)^i$, so the chance that the new packet gets accepted is

$$\sum_{i=1}^{\infty} 1/k \cdot (1-1/k)^{i-1}(1-(1-1/n)^i) = \frac{1}{k}\sum_{i=1}(1-\frac{1}{k})^{i-1} - \frac{n-1}{nk}\sum_{i=1}^{\infty}\left((1-\frac{1}{k})(1-\frac{1}{n})\right)^{i-1}$$

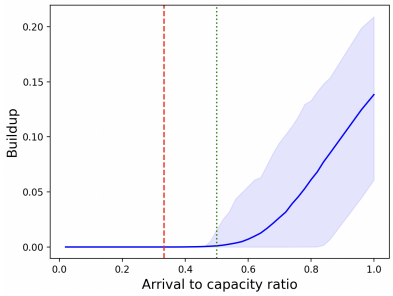
$$= 1 - \frac{n-1}{nk}\frac{1}{1-(1-1/k)(1-1/n)} = 1 - \frac{n-1}{n+k-1}.$$

To make this stable we need this value to be above $1/2$, so we need $k > n-1$. This results in a total capacity of $\frac{k}{n} > \frac{n-1}{n}$, essentially double the arrival rate of the queue. □

4 Experiments



(a) Buildup for a symmetric system with 3 servers with capacity μ and 3 queues with arrival rates λ as a function of λ/μ.

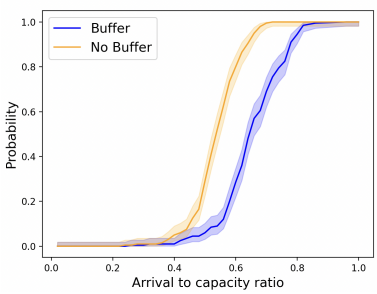


(b) Buildup for an ensemble of 200 randomized systems with 5 queues and 6 servers as a function of $\sum_i \lambda_i / \sum_j \mu_j$.

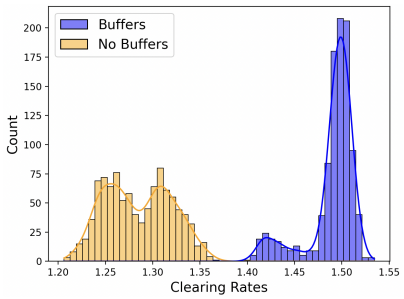
Figure 2: Empirical buildup of total queue sizes normalized by $n \cdot T$ as a function of the ratio $\sum_i \lambda_i / \sum_j \mu_j$ across different scenarios. The dashed vertical lines mark the ratios of $1/3$ and $1/2$ from our analysis.

In the preceding section, we derived theoretical stability conditions and worst-case bounds on the buildup of queue lengths. Specifically, our proof showed that in a system with singleton buffers, whenever a queue grows large, given that a condition of the total service capacity is satisfied, a no-regret property of the agents selecting servers guarantees that it will shrink at a linear rate, keeping the total lengths of queues bounded. In this section, we complement these theoretical guarantees with computational experiments to observe the typical behavior of these systems beyond the worst case, and explore the empirical impact of buffers on the game dynamics.

We conduct simulations using the EXP3 algorithm Auer *et al.* [2002b] (see also Slivkins [2019]), implemented in Python. For the simulation parameters, we denote γ for exploration rate and T for the time horizon of the simulation.



(a) Probability that a randomized system with time horizon T has a queue with buildup greater than $\sqrt{T}$ as a function of the ratio $\sum_i \lambda_i / \sum_j \mu_j$. Shaded regions show 95% confidence interval for the probability estimation.



(b) Empirical clearing rates (the number of packets cleared by a system normalized by time horizon T) across 1,000 samples of a system with fixed parameters. The left histogram is without buffers and the right one is with buffers.

Figure 3: A comparison of identical systems with and without buffers.

4.1 Empirical Queue Buildup

We start by studying the relationship between the system's arrival rate and its service capacity. The service capacity must strictly exceed the arrival rate to prevent queue buildup. This observation is seen also in typical experimental scenarios.

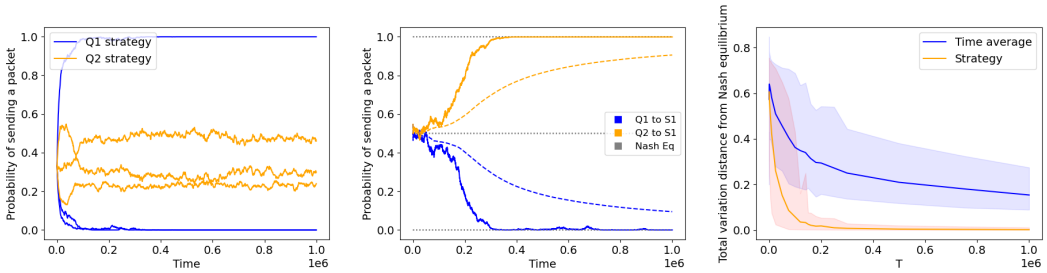
Figure 2 illustrates the total empirical buildup, which is the sum of all packets left in queues after T iterations, normalized by $n \cdot T$ (vertical axis) as a function of the arrival-to-capacity ratio $\sum_i \lambda_i / \sum_j \mu_j$ (horizontal axis). Figure 2a focuses on a symmetric system with three servers such that $\mu_1 = 0.8$ and $\mu_2 = \mu_3 = 0.2$ and three queues with equal arrival rates λ_i, which range between 0.02 and 0.4. The simulations run for $T = 50{,}000$ steps with $\gamma = 1/\sqrt{T}$. The solid line represents the average buildup over 200 independent simulations for each value of λ_i, and the shaded region indicates the range between minimum and maximum buildup values observed in the simulations. A clear transition emerges: when $\sum_i \lambda_i / \sum_j \mu_j > 0.9$, the buildup becomes proportional to T. Crucially, this transition occurs at a point above 0.5 and below 1 (where buildup is inevitable).

Figure 2b presents the same analysis for an ensemble of randomized systems with 5 queues and 6 servers. To generate such instances, we proceed as follows: generate 200 base parameters by selecting 5 queue arrival rates, λ_i, and 6 server capacity rates, μ_j, uniformly at random in $(0, 1)$. For each value of the ratio $r = \sum_i \lambda_i / \sum_j \mu_j$ (with r in $(0, 1]$), rescale the arrival rates such that the ratio $\sum_i \lambda_i / \sum_j \mu_j = r$ is satisfied. This process is repeated for each of the 200 sets of base parameters. The simulations are run for $T = 50{,}000$ and $\gamma = 1/\sqrt{T}$. In Figure 2b, the solid line represents the average buildup over the 200 simulation for each r value; the shaded region indicates the range between the 2.5 and 97.5 percentiles of buildup values. The results confirm that, even for asymmetric and randomized systems, the transition point remains above 0.5 and below 1. In both Figure 2a and Figure 2b the vertical lines indicate where $r = \frac{1}{3}$ and $r = \frac{1}{2}$. The empirical buildups in these simulations show that while the worst case bound is $\sum_i \lambda_i < \frac{1}{3} \sum_j \mu_j$, systems with learning agents typically do better than the worst case at clearing packets.

4.2 The Impact of Buffers

We now compare systems with vs. without buffers. First we note that even a buffer of just one packet at each server can significantly increase server capacity. To see this, consider a system with a single queue receiving packets at a high rate, say with probability $\lambda = \frac{1}{2}$ at each time step, and multiple servers, each with a service rate, say $\mu = \frac{1}{3}$. Without buffers, the condition for stability of such a system is that there is a single server with a service rate of $\mu > \lambda$, as a queue can only send a packet to one server at a time. However, with buffers, two such servers will make the system stable if the queue alternates sending its packets to them.

Our empirical methodology for illustrating the value of buffers follows the approach described in the previous subsection. Figure 3a illustrates the effect of adding buffers to a system in the random-instance scenario, with $n = 5$ queues and $m = 6$ servers as described earlier. It shows the probability that some queue in the system has at time T a buildup greater than $\sqrt{T}$ as a function of



(a) Dynamics in a system with two queues with arrival rates $\lambda_1 = 1/3$, $\lambda_2 = 1/6$ and three servers with service capacities $\mu_1 = 2/3$, $\mu_2 = 2/9$, $\mu_3 = 1/9$ as a function of time.

(b) Probability of sending to server 1 in a system with 2 queues with $\lambda_i = 1/4$, and 2 servers with $\mu_i = 2/3$, compared to Nash equilibria. Dashed lines show time averages.

(c) Total variation distance from a pure Nash equilibrium as a function of time in multiple simulations in the system from Figure 4b. Shaded regions are 95% confidence intervals.

Figure 4: Dynamics of probability distributions and the empirical play across different scenarios.

the arrival-to-capacity ratio for systems with and without buffers. The solid lines show the observed frequency in 300 independent simulations for each value of $r = \sum_i \lambda_i / \sum_j \mu_j$ in $(0, 1]$; the shaded region shows the 95% confidence interval in estimating these probabilities. We observe that systems without buffers require a higher capacity relative to the arrival rate to prevent buildup.

Figure 3b shows the distributions of packet clearing rates for two identical systems, with and without buffers, over 1,000 simulations with $T = 10{,}000$ and $\gamma = 1/\sqrt{T}$. There are 4 queues with $\lambda_1 = 0.6, \lambda_2 = \lambda_3 = \lambda_4 = 0.3$ and 5 servers with $\mu_1 = 0.8, \mu_2 = \mu_3 = 0.4, \mu_4 = \mu_5 = 0.2$. The clear separation between the distributions demonstrates that buffers increase efficiency. In the system without buffers, queues preferred the servers with higher capacity, which led to more competition and a lower clearing rate since the three high-capacity servers could not process all packets.[4] In the system with buffers, queues do learn to extract good service rates also from the lower capacity servers, which increases the overall clearing rate of the system.

4.3 Dynamics and Convergence

Next, we look at the dynamics of the learning agents in our systems. We observe that in all parameter configurations we have tested, the dynamics converge in last iterate (up to a small noise level due to the finite time horizon and step size in our simulations) to Nash equilibria, as illustrated in Figure 4. This is surprising, as even without the carryover effects that our games have, no-regret dynamics need not converge to Nash equilibria.[5] Figure 4a depicts the dynamics in systems with 2 queues and 3 servers. It can be seen that the probabilities of play of the algorithms approximately converge to a stationary distribution. Note that a stationary distribution of play and the no-regret condition imply that this distribution is a Nash equilibrium. Figure 4b shows a dynamic in a simpler system which has two pure Nash equilibria and one mixed equilibrium. As can be seen in the instance in the figure, which is a typical one, the dynamic in this system converges to a pure equilibrium. Figure 4c depicts the total variation distance from the closest Nash equilibrium as a function of the horizon T in 200 simulations of the same system for each T. The strategies quickly converge, and so the distance of the historical average of play shrinks as well, roughly as $1/\sqrt{T}$.

Conclusion: This work contributes to the growing literature on learning in games with carryover effects between rounds. Routing is an important setting where such effects arise and where routers use simple distributed learning algorithms. Carryover effects appear in many other strategic multi-agent settings, such as auctions with budget constraints, or in investment markets where current wealth affects future opportunities. A key observation from our results—potentially relevant more broadly to mechanism design with carryover effects—is that if the mechanism can smooth the payoffs learners experience over time, the resulting dynamics improve even with simple learners. In our model, buffers play this role by allowing a packet to be sent now and processed later. Exploring this idea in other domains is an interesting direction for future work.

[4] The total service rate of the three top servers equals the total arrival rate, hence is not enough for stability.

[5] In fact, no-regret dynamics may fail to converge to any single coarse correlated equilibrium even in the average-iterate sense Kolumbus and Nisan [2022a,b].

References

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Circumventing the price of anarchy: Leading dynamics to good behavior. *SIAM J. Comput.*, 42(1):230–264, 2013.

Santiago R. Balseiro and Yonatan Gur. Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Manag. Sci.*, 65(9):3952–3968, 2019.

Lucas Baudin, Marco Scarsini, and Xavier Venel. Strategic behavior and no-regret learning in queueing systems. *CoRR*, abs/2302.03614, 2023.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Giannis Fikioris and Éva Tardos. Liquid welfare guarantees for no-regret learning in sequential budgeted auctions. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 678–698, 2023.

Giannis Fikioris, Robert Kleinberg, Yoav Kolumbus, Raunak Kumar, Yishay Mansour, and Éva Tardos. Learning in budgeted auctions with spacing objectives. *arXiv preprint arXiv:2411.04843*, 2024.

Daniel Freund, Thodoris Lykouris, and Wentao Weng. Efficient decentralized multi-agent learning in asymmetric queuing systems. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 4080–4084. PMLR, 2022.

Daniel Freund, Thodoris Lykouris, and Wentao Weng. Quantifying the cost of learning in queueing systems. In *NeurIPS*, 2023.

Hu Fu, Qun Hu, and Jia'nan Lin. Stability of decentralized queueing networks beyond complete bipartite cases. In Kristoffer Arnsfelt Hansen, Tracy Xiao Liu, and Azarakhsh Malekian, editors, *Web and Internet Economics - 18th International Conference, WINE 2022, Troy, NY, USA, December 12-15, 2022, Proceedings*, volume 13778 of *Lecture Notes in Computer Science*, pages 96–114. Springer, 2022.

Jason Gaitonde and Éva Tardos. Stability and learning in strategic queuing systems. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pages 319–347. ACM, 2020.

Jason Gaitonde and Éva Tardos. The price of anarchy of strategic queuing systems. *J. ACM*, 70(3):20:1–20:63, 2023.

Jason Gaitonde, Yingkai Li, Bar Light, Brendan Lucier, and Aleksandrs Slivkins. Budget pacing in repeated auctions: Regret and efficiency without convergence. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 52:1–52:1. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

Isaac Grosof, Yige Hong, and Mor Harchol-Balter. Analysis of markovian arrivals and service with applications to intermittent overload. *CoRR*, abs/2405.04102, 2024.

Isaac Grosof, Yige Hong, Mor Harchol-Balter, and Alan Scheller-Wolf. The RESET and MARC techniques, with application to multiserver-job analysis. *SIGMETRICS Perform. Evaluation Rev.*, 51(4):6–7, 2024.

Refael Hassin and Moshe Haviv. *To Queue or Not to Queue: Equilibrium Behavior in Queueing Systems*. International Operations Resea. Springer US, 2003.

Refael Hassin. *Rational Queueing*. Chapman and Hall/CRC Series in Operations Research Series. CRC Press, 2020.

Yoav Kolumbus and Noam Nisan. Auctions between regret-minimizing agents. In *Proceedings of the ACM Web Conference 2022*, pages 100–111, 2022.

Yoav Kolumbus and Noam Nisan. How and why to manipulate your own agent: On the incentives of users of learning agents. *Advances in Neural Information Processing Systems*, 35:28080–28094, 2022.

Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer, 1999.

Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, and Sanjay Shakkottai. Regret of queueing bandits. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1669–1677, 2016.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

Robin Pemantle and Jeffrey S Rosenthal. Moment conditions for a sequence with negative drift to be uniformly bounded in L^r. *Stochastic Processes and their Applications*, 82(1):143–155, 1999.

Tim Roughgarden. Intrinsic robustness of the price of anarchy. *J. ACM*, 62(5):32:1–32:42, 2015.

Flore Sentenac, Etienne Boursier, and Vianney Perchet. Decentralized learning in online queuing systems. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS 2021*.

Flore Sentenac, Etienne Boursier, and Vianney Perchet. Decentralized learning in online queuing systems. In *NeurIPS*, 2021.

John F Shortle, James M Thompson, Donald Gross, and Carl M Harris. *Fundamentals of Queueing Theory*. Wiley, 2018.

Aleksandrs Slivkins. Introduction to multi-armed bandits. *Found. Trends Mach. Learn.*, 12(1-2):1–286, 2019.

Vasilis Syrgkanis and Éva Tardos. Composable and efficient mechanisms. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 211–220. ACM, 2013.

Neil Walton and Kuang Xu. *Learning and Information in Stochastic Networks and Queues*, chapter 6, pages 161–198. 2021.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The full formal model and all the proofs are included in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: All the model assumptions are clearly specified. A discussion on the tightness of the analysis is included and further exploration is done in the experiential section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: As mentioned above, the full model, assumptions, and proofs are specified.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Simulation details are specified in the experimental section. The code for the simulations will be available in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code for the simulations will be available in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experiments are simulation based. All simulation parameters as well as the code are provided.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Experiments include multiple repetitions for each parameter configuration. Confidence intervals are given.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: The experiments are run on a simple laptop computer.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms with the ethics code. The setting we analyze and the results do not have any direct ethical implications.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses in what conditions systems generate high social welfare for their users and avoid unbounded accumulation of demand. While well operating networking systems are important for various economic and social reasons, the analysis itself does on have direct societal implications.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Not applicable.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.