Renting Edge Computing
Resources for Service Hosting

Aadesh Madnaik, Sharayu Moharir, and Nikhil Karamchandani

Indian Institute of Technology Bombay, Mumbai, India
aadesh.madnaik@gmail.com
{sharayum,nikhilk}@ee.iitb.ac.in

Abstract. We consider the setting where a service is hosted on a third-party edge server deployed close to the users and a cloud server at a greater distance from the users. Due to the proximity of the edge servers to the users, requests can be served at the edge with low latency. However, as the computation resources at the edge are limited, some requests must be routed to the cloud for service and incur high latency. The system’s overall performance depends on the rent cost incurred to use the edge server, the latency experienced by the users, and the cost incurred to change the amount of edge computation resources rented over time. The algorithmic challenge is to determine the amount of edge computation power to rent over time. We propose a deterministic online policy and characterize its performance for adversarial and stochastic i.i.d. request arrival processes. We also characterize a fundamental bound on the performance of any deterministic online policy. Further, we compare the performance of our policy with suitably modified versions of existing policies to conclude that our policy is robust to temporal changes in the intensity of request arrivals.

Keywords: Service hosting · edge computing · competitive ratio.

1 Introduction

Software as a Service (SaaS) instances like search engines, online shopping platforms, navigation services, and Video-on-Demand services have recently gained popularity. Low latency in responding to user requests/queries is essential for most of these services. This necessitates the use of edge resources in a paradigm known as edge-computing [24], i.e., storage and computation power close to the resource-constrained users, to serve user queries. Due to limited computation resources at the edge, such services are often also deployed on cloud servers which can serve requests that cannot be served at the edge, albeit with more latency given the distance between the cloud servers and the users. Ultimately, introducing edge-computing platforms facilitates low network latency coupled with higher computational capabilities. For instance, consider a scenario where a child goes missing in an urban setting [25]. While cameras are widely used for security, it is challenging to leverage the information as a whole because of
privacy and data traffic issues. In the edge computing paradigm, a workaround would be to push a request to search for the child to a certain subset of devices, thereby making the process faster and more efficient than cloud computing. Several other avenues of edge computing exist in the forms of cloud offloading, AR/VR-based infotainment, autonomous robotics, Industry 4.0 and the Internet of Things (IoT). Several industry leaders offer services for edge resources, e.g., Amazon Web Services [3], Oracle Cloud Infrastructure [14] and IBM with 5G technology [13].

This work considers a system with cloud servers and third-party-owned edge servers. Edge resources, i.e., storage and computation power, can be rented via short-term contracts to host services. Storage resources are needed to store the code, databases, and libraries for the service and computation resources are required to compute responses to user queries. As edge servers are limited in computational capabilities, there is a cap on the number of concurrent requests that can be served at the edge [26]. The amount of edge computational resources rented for the service governs the number of user requests that can be served simultaneously at the edge. We focus on a service that is hosted both on the cloud and edge servers, and the amount of edge computational resources rented can be changed over time based on various factors, including the user request traffic and the cost of renting edge computation resources. Service providers provision for elasticity in the quantity of edge resources rented, and the clients can exploit this based on the number of request arrivals [18]. The total cost incurred by the system is modelled as the sum of the rent cost incurred to use edge resources, the cost incurred due to high latency in serving requests that have to be routed to the cloud, and the switching cost incurred every time the amount of edge computation resource rented is changed [30]. The algorithmic challenge in this work is to determine the amount of edge computation resources to rent over time in the setting where the request arrival sequence is revealed causally with the goal of minimizing the overall cost incurred.

1.1 Our Contributions

We propose a deterministic online policy called Better-Late-than-Never (BLTN) inspired by the RetroRenting policy proposed in [21] and analyze its performance for adversarial and, i.i.d. stochastic request arrival patterns. In addition to this, we also characterize fundamental limits on the performance of any deterministic online policy for adversarial arrivals in terms of competitive ratio against the optimal-offline policy. Further, we compare the performance of BLTN with a suitably modified version of the widely studied Follow the Perturbed Leader (FTPL) policy [5][19] via simulations. Our results show that while the performance of BLTN and FTPL is comparable for i.i.d. stochastic arrivals, for arrival processes with time-varying intensity, e.g., a Gilbert-Elliot-like model, BLTN significant outperforms FTPL. The key reason for this is that BLTN puts extra emphasis on recent arrival patterns of making decisions, while FTPL uses the entire request arrival history to make decisions. For all settings under consideration, the simulations demonstrate that BLTN differs little in performance from
the optimal online policy despite not having information about the incoming request arrival process.

1.2 Related Work

There has been a sharp increase in mobile application latency and bandwidth requirements, particularly when coupled with time-critical domains such as autonomous robotics and the Internet of Things (IoT). These changes have ushered in the advent of the edge computing paradigm away from the conventional remote servers, as discussed in the surveys [1,16,23]. The surveys alongside several academic works elaborate on and model the dynamics of such systems. We briefly discuss some relevant literary works.

Representations of the problem considered in [6,26] model the decision making of which services to cache and which tasks to offload as a mixed-integer non-linear optimization problem. In these cases, the problem is NP-hard. Similarly, [8] models the problem as a graph coloring problem and solves it using parallel Gibbs sampling. While these works try to solve a one-shot cost-minimization problem, in this work we consider the dynamic nature of decision-making based on the input request sequence.

Another model considered in [28] for service hosting focused on the joint optimization of service hosting decision and pricing is a two-stage interactive game between a base-station that provides pricing for edge servers and user equipment which decides whether to offload the task. In another game-theoretic setup, [15,29] delve into the economic aspects of edge caching involving interactions amongst different stakeholders. Some heuristic algorithms have been employed in the works [2,10]. Their approach for the problem is through resource constraint in the latency from the view-point of the edge-cloud infrastructure and not the application provider.

Stochastic models of the system have been considered in [9,17,27]. While [27] assumes that the underlying requests follow a Poisson process, [9,17] do not make any prior assumptions regarding the same. [9,17], through Contextual Combinatorial Multiarmed Bandits aim to use a learning-based approach to make decisions. [27] formulates the service migration problem as a Markov decision process (MDP) to design optimal service migration policies. These models do not provide any worst-case guarantees for the algorithm, simply average guarantees. In our work, we aim to provide both performance guarantees which are crucial to sensitive applications with large variations in the arrival patterns.

Closest to our work, [20,22] consider the setting where a service is always hosted at the cloud and consider the algorithmic task of determining when to host the service at the edge server as a function of the arrival process and various system parameters. The key difference between our work and [20,22] is that, in [20,22], once the service is hosted at the edge, the amount of edge computation resources available for use by the service is either fixed or effectively unlimited. Our model allows us to choose the level of computation resources to rent which is a feature available in popular third-party storage/computation resource providers like AWS and Microsoft Azure. Another critical difference
between our model and the [20–22] is the fact that we consider the setting where a non-zero switch cost is incurred every time we change the level of computation resource rented. Contrary to this, in [20–22], switch cost is unidirectional, i.e., a switch cost is incurred only when a service is not hosted at the edge in a time-slot and has to be fetched from the cloud servers to host on the edge server. Due to this, the algorithms proposed in [20–22] and their performance analyses do not directly extend to our setting. Other works on the service hosting problem include [31]. At a high level, our work is related to the rich body of work on caching [4,5,7,19].

2 Setting

We study a system consisting of a cloud server and a third-party-owned edge server. We focus on the problem of efficiently using edge resources from the perspective of a specific service provider. This service is hosted both at the edge and on the cloud server. Each user query/request is routed either to the edge or the cloud and the answer to the query is computed at that server and communicated back to the user, thus necessitating computation power both at the edge and at the cloud servers. We consider a time-slotted setting where the amount of edge computation power rented by the service provider can be changed over time via short-term contracts.

Request arrival process: We consider adversarial and stochastic arrivals. Under the adversarial setting, we make no structural assumptions on the number of requests arriving over time. For our analytical results for stochastic arrivals, we consider the setting where arrivals are i.i.d. over time.

Assumption 1 (i.i.d. stochastic arrivals) Let $X_t$ be the number of requests arriving in time-slot $t$. Then, for all $t$, $\mathbb{P}(X_t = x) = p_x$ for $x = 0, 1, 2, \cdots$.

In the Gilbert-Elliot-like Model, we make the following assumption:

Assumption 2 (Gilbert-Elliot (GE) Model) Using [11] as a basis, we consider an arrival process governed by a two-state Markov chain, $A_H$ and $A_L$. Transitions from state $A_H \rightarrow A_L$ and from state $A_L \rightarrow A_H$ occur with probabilities $p_{HL}$ and $p_{LH}$. The state transition diagram has been described in [1]. We refer to the two states as the high state and the low state. Under the GE model, if the Markov chain is in the high state, the requests arrive as Poisson($\lambda_H$), and they are Poisson($\lambda_L$) otherwise.

Sequence of events in each time-slot: We first have request arrivals. These requests are served by the edge server subject to constraints due to limited computation power at the edge. The remaining requests, if any, are forwarded to the cloud server for service. The system then makes a decision on how much edge computation power to rent for the next time-slot.

The algorithmic challenge is to determine how much edge computation power to rent over time. Let $P$ be a candidate policy that determines the amount of computation power rented by the service provider over time.
2.1 Cost Model and Constraints

We build on the assumptions in [20–22]. Under policy \( \mathcal{P} \), the service provider incurs three types of costs.

- **Rent cost** \((C_{\mathcal{P}}^R, t)\): The service provider can choose one of two possible levels of edge computation power to rent in each time-slot, referred to as high \((H)\) and low \((L)\). The rent cost incurred per time-slot for levels \(H\) and \(L\) are denoted by \(c_H\) and \(c_L\) respectively.

- **Service cost** \((C_{\mathcal{P}}^S, t)\): This is the cost incurred due to the latency in service user requests. Given the proximity of the edge servers and the users, no service cost is incurred for requests served at the edge. A cost of one unit is levied on each request forwarded to the cloud server. The highest number of requests that can be served at the edge at edge computation power levels \(H\) and \(L\) are denoted by \(\kappa_H\) and \(\kappa_L\) respectively.

- **Switch cost** \((C_{\mathcal{P}}^W, t)\): Switching from edge computation power level \(H\) to \(L\) and \(L\) to \(H\) results in a switch cost of \(W_{HL}\) and \(W_{LH}\) units respectively.

The number of requests that can be served by the edge server in a time-slot is limited to \(\kappa_H\) for state \(S_H\) and \(\kappa_L\) for state \(S_L\) in \(\mathbb{Z}^+\), where \(\mathbb{Z}^+\) is the set of all positive integers. Let \(r_t \in \{H, L\}\) denote the edge computation power rented during time-slot \(t\) and \(X_t\) denote the number of request arrivals in time-slot \(t\). It follows that

\[
C_{\mathcal{P}, t} = C_{\mathcal{P}, R, t} + C_{\mathcal{P}, S, t} + C_{\mathcal{P}, W, t},
\]

where,

\[
C_{\mathcal{P}, R, t} = \begin{cases} 
  c_H & \text{if } r_t = H \\
  c_L & \text{if } r_t = L 
\end{cases}
\]

\[
C_{\mathcal{P}, S, t} = \begin{cases} 
  X_t - \min\{X_t, \kappa_H\} & \text{if } r_t = H \\
  X_t - \min\{X_t, \kappa_L\} & \text{if } r_t = L 
\end{cases}
\]

\[
C_{\mathcal{P}, W, t} = \begin{cases} 
  W_{HL} & \text{if } r_{t-1} = H \text{ and } r_t = L \\
  W_{LH} & \text{if } r_{t-1} = L \text{ and } r_t = H \\
  0 & \text{otherwise.} 
\end{cases}
\]

**Remark 1.** We limit our discussion to the case where \(\kappa_H - \kappa_L > c_H - c_L\). If \(\kappa_H - \kappa_L \leq c_H - c_L\), the optimal policy is to always use computation level \(L\).
2.2 Performance metrics

We use the following metrics for adversarial and stochastic request arrivals.

For adversarial arrivals, we compare the performance of a policy $\mathcal{P}$ with the performance of the optimal offline policy (OPT-OFF) which knows the entire arrival sequence a priori. The performance of policy $\mathcal{P}$ is characterized by its competitive ratio $\rho$ defined as

$$\rho = \sup_{a \in \mathcal{A}} \frac{C^\mathcal{P}(a)}{C^{\text{OPT-OFF}}(a)},$$

(2)

where $\mathcal{A}$ is the set of all possible finite request arrival sequences, and $C^\mathcal{P}(a)$ and $C^{\text{OPT-OFF}}(a)$ are the total costs of service for the request arrival sequence $a$ under the policy $\mathcal{P}$ and the optimal offline policy respectively.

For i.i.d. stochastic arrivals, we compare the performance of a policy $\mathcal{P}$ with the optimal online policy (OPT-ON) which might know the statistics of the arrival process, but does not know the sample path. The performance metric $\sigma_T^\mathcal{P}$ is defined as the ratio of the expected cost incurred by policy $\mathcal{P}$ in $T$ time-slots to that of the optimal online policy in the same time interval. Formally,

$$\sigma_T^\mathcal{P} = \frac{\mathbb{E}\left[\sum_{t=1}^{T} C^\mathcal{P}_t\right]}{\mathbb{E}\left[\sum_{t=1}^{T} C^{\text{OPT-ON}}_t\right]},$$

(3)

where $C^\mathcal{P}_t$ is as defined in (1).

Goal: The goal is to design online policies with provable performance guarantees for both adversarial and stochastic arrivals.

3 Policies

In our analysis, we focus the discussion towards online policies. At each time-slot, a singular decision must be made determining whether to switch states.

3.1 Better Late than Never (BLTN)

The BLTN policy is inspired by the RetroRenting policy proposed in [21]. BLTN is a deterministic policy that uses recent arrival patterns to evaluate decisions by checking if it made the correct choice in hindsight. Let $t_{\text{switch}} < t$ be the most recent time when the state was changed from $H \rightarrow L$ or $L \rightarrow H$ under BLTN. The policy searches for a time-slot $\tau$ such that $t_{\text{switch}} < \tau < t$, and the total cost incurred is lower if the state is switched in time-slot $\tau - 1$ and switched back in time-slot $t$ than the cost incurred if the state is not changed during time-slots $\tau - 1$ to $t$. If there exists such a time $\tau$, BLTN switches the state in time-slot $t$. 
Consider a scenario where the state in time slot $t$ is $S_H$. Let $t_{\text{switch}} < t$ be the time when the server had last changed state to $S_L$ under BLTN. Let $H_i$ and $L_i$ denote cost incurred in time slot $i$ where $H_i$ and $L_i$ are evaluated for $r_i = r_{i-1} = H$ and $L$ respectively. Analytically, the decision to switch to state $S_L$ is made if the algorithm can find a time $\tau$ such that

$$ W_{LH} + W_{HL} + \sum_{t_{\text{switch}} \leq i < \tau} H_i + \sum_{\tau \leq j \leq t} L_j < \sum_{t_{\text{switch}} \leq i \leq t} H_i \tag{4} $$

which simplifies to $W_{LH} + W_{HL} < \sum_{i=\tau}^t (H_i - L_i)$.

A similar analytical condition can be made for the decision to switch from $S_L$ to state $S_H$. The decision is made if the algorithm can find a time-slot $\tau$ such that

$$ W_{LH} + W_{HL} < \sum_{i=\tau}^t (L_i - H_i). \tag{5} $$

A naive implementation of the algorithm has been constructed in the Appendix.

While a naive implementation of the BLTN policy can have $O(T)$ space and time complexity, using techniques proposed in [21], the time and computational complexity can be reduced to $O(1)$ as shown through Algorithm 1.

### 3.2 Follow The Perturbed Leader (FTPL)

FTPL [5,19] is a randomized policy. In time-slot $t$, it compares suitably perturbed versions of the cost incurred from time 1 to $t$ under two static decisions, i.e., state $L$ from time 1 to $t$ and state $H$ from time 1 to $t$. The state of the system is then set to the one which has the lower perturbed cost.

The variation of the perturbation $\mathcal{N}(0, \sqrt{t})$ increases as $\sqrt{t}$, while the total difference in cost scaled linearly with time. This implies that the FTPL policy, over time, chooses to remain static in the state with the least cost.

### 3.3 An illustration

We consider a sample sequence of arrivals. Let $r_t$ be the state sequence with time index $t$. We consider the case where $W_{LH} = W_{HL} = 275, c_H = 600, c_L = 400, \kappa_H = 700, \kappa_L = 300$ and a request sequence

| Time-slot index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|-----------------|---|---|---|---|---|---|---|---|---|----|----|
| Number of requests: | 900 | 900 | 900 | 900 | 900 | 900 | 200 | 200 | 200 | 200 | 200 |

Initially, we consider the edge server to be in state $S_L$. We observe that the optimal state to serve 900 incoming requests is $S_H$. While hosting under the BLTN policy, the first switch to state $S_H$ occurs in the time-slot 3, thus $r_4 = H$.
Algorithm 1: Better Late than Never (BLTN)

1. Input: Sum of switch costs $W$ units, maximum number of our service requests served by edge server ($\kappa_H$ and $\kappa_L$), rent cost: $c_H$ and $c_L$, number of requests: $x_t$, $x_t^H = \min\{x_t, \kappa_H\}$, $x_t^L = \min\{x_t, \kappa_L\}$, $t > 0$
2. Output: Service hosting strategy $r_{t+1} \in \{H, L\}$, $t > 0$
3. Initialize: Service hosting variable $r_1 = 0$, $\Delta(0) = 0$
4. for each time-slot $t$ do
5. \hspace{1cm} $\Delta(t-1) = \Delta(t)$
6. \hspace{1cm} if $r_t = H$ then
7. \hspace{2cm} $\Delta(t) = \max \{0, \Delta(t-1) + x_t^L - x_t^H + c_H - c_L\}$
8. \hspace{2cm} if $\Delta(t) > W$ then
9. \hspace{3cm} $t_{\text{switch}} = t$
10. \hspace{3cm} $\Delta(t) = 0$
11. \hspace{3cm} return $r_{t+1} = L$
12. \hspace{2cm} else
13. \hspace{3cm} return $r_{t+1} = H$
14. \hspace{1cm} end
15. \hspace{1cm} else if $r_t = L$ then
16. \hspace{2cm} $\Delta(t) = \max \{0, \Delta(t-1) + x_t^H - x_t^L + c_L - c_H\}$
17. \hspace{2cm} if $\Delta(t) > W$ then
18. \hspace{3cm} $t_{\text{switch}} = t$
19. \hspace{3cm} $\Delta(t) = 0$
20. \hspace{3cm} return $r_{t+1} = H$
21. \hspace{2cm} else
22. \hspace{3cm} return $r_{t+1} = L$
23. \hspace{1cm} end
24. end

Algorithm 2: Follow the Perturbed Leader (FTPL)

1. Input: Switch costs $W_{HL}$ and $W_{LH}$ units, maximum number of our service requests served by edge server ($\kappa_H$ and $\kappa_L$), rent cost: $c_H$ and $c_L$, number of requests: $x_t$, $x_t^H = \min\{x_t, \kappa_H\}$, $x_t^L = \min\{x_t, \kappa_L\}$, $t > 0$, Gaussian distribution with mean $\mu$ and variance $\sigma$: $N(\mu, \sigma)$
2. Output: Service hosting strategy $r_{t+1} \in \{H, L\}$, $t > 0$
3. Initialize: Service hosting variable $r_1 = 0$, $\Delta(0) = 0$
4. for each time-slot $t$ do
5. \hspace{1cm} $\Delta(t-1) = \Delta(t)$
6. \hspace{1cm} $\Delta(t) = \Delta(t-1) + x_t^L - x_t^H + c_H - c_L$
7. \hspace{1cm} if $\Delta(t) + \gamma N(0, \sqrt{t}) > 0$ then
8. \hspace{2cm} return $r_{t+1} = L$
9. \hspace{1cm} else
10. \hspace{2cm} return $r_{t+1} = H$
11. end
12. end
Renting Edge Computing Resources for Service Hosting

and \( r_{1,2,3} = L \). The cost incurred in the case where the state is \( S_L \) till \( t = 3 \) is \( \sum_{i=1}^{3} (x_i - \kappa_L)^+ + (3 - 1 + 1) \times c_L = 3000 \), while the cost incurred in state \( S_H \) is \( \sum_{i=1}^{3} (x_i - \kappa_H)^+ + (3 - 1 + 1) \times c_H = 2400 \). The difference in the cost equates \( 600 > W = 550 \), and that is the first time the condition [5] is satisfied. \( t_{\text{switch}} \) is updated to 3, and \( r_4 = H \).

From time-slot 4 onward, BLTN hosts in state \( S_H \) up to time-slot 8. We set \( t = 8, \tau = 6 \), we have \( \sum_{i=6}^{8} ((x_i - \kappa_H)^+ - (x_i - \kappa_L)^+) + (8 - 6 + 1) \times (c_H - c_L) = 600 \geq W = 550 \). This is the first time-slot since 3 that the condition is satisfied, thus \( r_8 = H, r_9, \ldots = L \) till the next time BLTN decides to switch.

4 Main Results and Discussion

Our first theorem characterizes the performance of BLTN for adversarial arrivals by giving a worst-case guarantee on the performance of the BLTN policy against the optimal offline policy. We also characterize a lower bound performance of any deterministic online policy.

**Theorem 1.** Let \( \Delta \kappa = \kappa_H - \kappa_L, \Delta c = c_H - c_L, W = W_{LH} + W_{HL} \). If \( \Delta \kappa > \Delta c \) then,

(a) \( \rho^{BLTN} \leq \left( 1 + \frac{2W + \Delta \kappa}{W(1 + \frac{\Delta c}{c_H - c_L} + \frac{\Delta c}{c_L})} \right) \),

(b) \( \rho^{P} \geq \min \left\{ \frac{\Delta \kappa + c_L}{c_H}, \frac{\Delta \kappa + c_H}{c_L} + \frac{W}{c_H + c_L + W} \right\} \), for any deterministic policy \( P \).

Theorem 1 provides a worst-case guarantee on the performance of the BLTN policy against the optimal offline policy. Unlike the BLTN policy, the optimal offline policy has complete information of the entire arrival sequence beforehand. We note that the competitive ratio of BLTN improves as the sum of switch costs \( W_{LH} + W_{HL} \) increases. Also, the competitive ratio of BLTN increases linearly with the difference of the caps on requests served at the edge, \( \Delta \kappa \). Supplementing it, we have Theorem 1(b) which shows that the competitive ratio of any deterministic online policy increases linearly with \( \Delta \kappa \). While Theorem 1(a) provides a worst-case guarantee for the BLTN policy, it must be noted that through subsequent simulations, the performance of BLTN is substantially closer to the optimal offline policy.

Next we summarize the performance of the BLTN policy for i.i.d. stochastic arrivals (Assumption 1, Section 2).

This lemma gives a bound on the expected difference between the costs incurred in a time-slot by the BLTN policy and the optimal online policy. We use the functions \( f(\cdot), g(\cdot) \) which are defined in the Appendix. The functions \( f(\cdot), g(\cdot) \) are formulated using Hoeffding’s inequality to bound the probability of certain events. We use the functions \( f(\cdot), g(\cdot) \) for the sake of compactness.

**Lemma 1.** Let \( \Delta P = E[ C^{P}_{L} - C^{OPT-ON}_{L} ] \), \( \mu_H = E[X_{t,H}], \mu_L = E[X_{t,L}], \Delta \mu = \mu_H - \mu_L \), \( \Delta \kappa = \kappa_H - \kappa_L \), and \( \Delta c = c_H - c_L \).
Then, under Assumption 1,

\[ f(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c) = (W + \Delta \mu - \Delta c) \times \left( 2 \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \exp\left(-2 \frac{(\Delta \mu - \Delta c)^2 W}{\lambda(\Delta \kappa)^2} \right) \right. \]

\[ + \exp\left(-2 \frac{(\lambda - 1)^2 W(\Delta \mu - \Delta c)}{\lambda(\Delta \kappa)^2} \right), \quad \text{and} \]

\[ g(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c) = (\Delta c - \Delta \mu + W) \times \left( \exp\left(-2 \frac{(\lambda - 1)^2 (\Delta c - \Delta \mu) W}{\lambda(\Delta \kappa)^2} \right) \right. \]

\[ + 2 \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \exp\left(-2 \frac{(\Delta c - \Delta \mu)^2 W}{\lambda(\Delta \kappa)^2} \right) \right]. \]

Then, under Assumption 7.

- Case \( \Delta \mu > \Delta c \):

\[ \Delta^\text{BLTN}_t(\lambda) \leq \begin{cases} 
W + \Delta \mu - \Delta c, & t \leq \left\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \right\rceil, \\
\Delta^\text{BLTN}_t(\lambda) \leq \begin{cases} 
W + \Delta \mu - \Delta c, & t \leq \left\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \right\rceil, \\
g(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c), & t > \left\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \right\rceil, 
\end{cases}
\end{cases} \]

- Case \( \Delta \mu < \Delta c \):

\[ \Delta^\text{BLTN}_t(\lambda) \leq \begin{cases} 
W + \Delta \mu - \Delta c, & t \leq \left\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \right\rceil, \\
g(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c), & t > \left\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \right\rceil.
\end{cases} \]

We can conclude that for large enough \( t \), the difference between the cost incurred by BLTN and the optimal online policy in time-slot \( t \), decays exponentially with \( W \) and \( |\Delta \mu - \Delta c| \).

Theorem gives an upper bound on the ratio of the expected cost incurred under BLTN and the optimal online policy in a setting where request arrivals are stochastic. In the statement of this theorem, we used the functions \( f \) and \( g \) which are defined in Lemma 1.

**Theorem 2.** Let \( \nu = \mathbb{E}[X], \mu_H = \mathbb{E}[X_H], \) and \( \mu_L = \mathbb{E}[X_L] \). Let the rent cost per time-slot be \( c_H \) or \( c_L \) depending on the states \( S_H \) or \( S_L \) respectively. Define \( \Delta \mu = \mu_H - \mu_L, \Delta \kappa = \kappa_H - \kappa_L, \Delta c = c_H - c_L. \) Recall the definition of \( \sigma^\text{BLTN}_T \) given in [3].

- Case \( \Delta \mu > \Delta c \): For the function \( f \) defined in Lemma 1

\[ \sigma^\text{BLTN}_T(\lambda) \leq \min_{\lambda > 1} \left( 1 + \frac{\lambda W}{\Delta \mu - \Delta c} \right) \left( W + \Delta \mu - \Delta c \right) + \left( T - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \right) f(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c) \]

- Case \( \Delta \mu < \Delta c \): For the function \( g \) defined in Lemma 4

\[ \sigma^\text{BLTN}_T(\lambda) \leq \min_{\lambda > 1} \left( 1 + \frac{\lambda W}{\Delta \mu - \Delta c} \right) \left( W + \Delta \mu - \Delta c \right) + \left( T - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \right) g(\Delta \kappa, \lambda, W, \Delta \mu, \Delta c). \]

We observe that the bounds in Lemma worsen with an increase in \( \Delta \kappa \). It must be noted that this is a bound obtained using Hoeffding’s inequality which does not assume any specific i.i.d. process (Chernoff bound presents a stronger
inequality here). For generic cases, the performance of BLTN does not worsen with an increase in $\Delta \kappa$ as shown via simulations in the next section.

For large values of $T$, the bound on the ratio of total expected costs reduces exponentially with the sum of switch costs $W$. The performance guarantees obtained for BLTN in this section show that BLTN performs well in both the general and the i.i.d. stochastic settings without making any assumptions on the request arrival process.

5 Simulations

Since our analytical results only provide bounds on the BLTN policy, we now compare the performance of BLTN, FTPL, and the optimal online policy which uses the knowledge of the statistics of the arrival process to make decisions via simulations. Recall that both BLTN and FTPL do not know the statistics of the arrival process.

Unless stated otherwise, the parameter values in the simulations are as follows: $c_L = 300; \Delta c = 300; k_L = 400; \Delta \kappa = 400; W_{HL} = W_{LH} = 300$. For the GE model, the transition probability from either state of the two-state Markov chain is 0.01. In Algorithm 2, we set $\gamma = 500$, through empirical observations of overall performance. All the simulations have been averaged over the same set of 50 random seeds over 10,000 time-slots. The captions highlight the arrival sequence model - Assumption 1 (i.i.d. Poisson) or Assumption 2 (GE).

In Figure 2, we see that for i.i.d. Poisson arrivals, the performance of BLTN matches that of FTPL for most of the $\lambda$ values considered except around $\lambda = 600$. At this value of $\lambda$, the expected cost incurred at levels $L$ and $H$ is very close and as a result, the switch cost incurred by FTPL is high, thus leading to poor performance. We note that for the GE model, BLTN outperforms FTPL for a large range of $\lambda_H$. The superior performance of BLTN is a consequence of the

Fig. 2: Performance of various policies as a function of the request arrival rate. For the GE model, we fix $\lambda_L = 300$ and vary $\lambda_H$. 

In Figure 2, we see that for i.i.d. Poisson arrivals, the performance of BLTN matches that of FTPL for most of the $\lambda$ values considered except around $\lambda = 600$. At this value of $\lambda$, the expected cost incurred at levels $L$ and $H$ is very close and as a result, the switch cost incurred by FTPL is high, thus leading to poor performance. We note that for the GE model, BLTN outperforms FTPL for a large range of $\lambda_H$. The superior performance of BLTN is a consequence of the
fact that unlike FTPL, BLTN puts added emphasis on recent arrival patterns when making decisions. The same trend follows in Figure 3.

Fig. 3: Performance of various policies as a function of \( \Delta k \)

(a) i.i.d. Poisson, \( \lambda = 700 \)
(b) GE, \( \lambda_H = 800, \lambda_L = 300 \)

Fig. 4: Performance of various policies as a function of difference in rent costs \( \Delta c \)

(a) Poisson(\( \lambda = 700 \))
(b) GE(\( \lambda_H = 800, \lambda_L = 300 \))

Fig. 5: Performance of various policies as a function of switch costs, \( W = W_{HL} = W_{LH} \)
Through Theorems 1 and 2, it is suggested that the bounds worsen as $\Delta \kappa$ increases. However, under Assumption 1, there is significant difference in performance only when $\Delta \kappa = \Delta c$, and under Assumption 2, there is significant difference whenever $\Delta \kappa > \Delta c$.

In arrival sequences characterized by Assumption 2, usually BLTN performs better owing to its ability to draw conclusions from history. FTPL fails to account for switching costs and so, is sub-optimal.

6 Conclusion

We consider the problem of renting edge computing resources for serving customer requests at the edge. We propose an online policy called Better-Late-Than-Never (BLTN) and provide performance guarantees for adversarial and stochastic request arrivals. Further, we compare the performance of BLTN with the widely studied FTPL policy. We conclude that BLTN outperforms FTPL for most settings considered, especially when the statistics of the arrival process are time-varying. The main reason for this is that BLTN makes decisions based on recent request arrival patterns while FTPL uses the entire request arrival history to make decisions.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: A survey. IEEE Internet of Things Journal 5(1), 450–465 (2018). https://doi.org/10.1109/JIOT.2017.2750180
2. Ascigil, Ö., Tasiopoulos, A., Phan, T.K., Sorous, V., Psaras, I., Pavlou, G.: Resource provisioning and allocation in function-as-a-service edge-clouds. IEEE Transactions on Services Computing (2021)
3. AWS: (2022), https://aws.amazon.com
4. Belady, L.A.: A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal 5(2), 78–101 (1966)
5. Bhattacharjee, R., Banerjee, S., Sinha, A.: Fundamental limits on the regret of online network-caching. Proceedings of the ACM on Measurement and Analysis of Computing Systems 4(2), 1–31 (2020)

6. Bi, S., Huang, L., Zhang, Y.J.A.: Joint optimization of service caching placement and computation offloading in mobile edge computing system. arXiv preprint arXiv:1906.00711 (2019)

7. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: 2010 Proceedings IEEE INFOCOM. pp. 1–9. IEEE (2010)

8. Chen, L., Xu, J.: Collaborative service caching for edge computing in dense small cell networks. arXiv preprint arXiv:1709.08662 (2017)

9. Chen, L., Xu, J.: Budget-constrained edge service provisioning with demand estimation via bandit learning. arXiv preprint arXiv:1903.09080 (2019)

10. Choi, H., Yu, H., Lee, E.: Latency-classification-based deadline-aware task offloading algorithm in mobile edge computing environments. Applied Sciences 9(21), 4696 (2019)

11. Gilbert, E.N.: Capacity of a burst-noise channel. The Bell System Technical Journal 39(5), 1253–1265 (1960). https://doi.org/10.1002/j.1538-7305.1960.tb03959.x

12. Hoeffding, W.: Probability inequalities for sums of bounded random variables. In: The Collected Works of Wassily Hoeffding, pp. 409–426. Springer (1994)

13. IBM: https://www.ibm.com/cloud/edge-computing

14. Infrastructure, O.C.: https://www.oracle.com/a/ocom/docs/cloud/edge-services-100.pdf

15. Jiang, C., Gao, L., Wang, T., Luo, J., Hou, F.: On economic viability of mobile edge caching. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). pp. 1–6. IEEE (2020)

16. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: A survey. IEEE Communications Surveys & Tutorials 23(4), 2131–2165 (2021). https://doi.org/10.1109/COMST.2021.3106401

17. Miao, Y., Hao, Y., Chen, M., Gharavi, H., Hwang, K.: Intelligent task caching in edge cloud via bandit learning. IEEE Transactions on Network Science and Engineering 8(1), 625–637 (2020)

18. Mouradian, C., Naboulis, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE Communications Surveys & Tutorials 20(1), 416–464 (2017)

19. Mukhopadhyay, S., Sinha, A.: Online caching with optimal switching regret. In: 2021 IEEE International Symposium on Information Theory (ISIT). pp. 1546–1551. IEEE (2021)

20. Narayana, V.C.L., Agarwala, M., Karamchandani, N., Moharir, S.: Online partial service hosting at the edge. In: 2021 International Conference on Computer Communications and Networks (ICCCN). pp. 1–9. IEEE (2021)

21. Narayana, V.C.L., Moharir, S., Karamchandani, N.: On renting edge resources for service hosting. ACM Transactions on Modeling and Performance Evaluation of Computing Systems 6(2), 1–30 (2021)

22. Prakash, R.S., Karamchandani, N., Kavitha, V., Moharir, S.: Partial service caching at the edge. In: 2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT). pp. 1–8. IEEE (2020)

23. Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., Rana, O.: Fog computing for the internet of things: A survey. ACM Trans. Internet Technol. 19(2), 18:1–18:41 (Apr 2019). https://doi.org/10.1145/3301443 http://doi.acm.org/10.1145/3301443
Renting Edge Computing Resources for Service Hosting

24. Satyanarayanan, M.: The emergence of edge computing. Computer 50(1), 30–39 (2017)
25. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal 3(5), 637–646 (2016). https://doi.org/10.1109/JIOT.2016.2579198
26. Tran, T.X., Chan, K., Pompili, D.: Costa: Cost-aware service caching and task offloading assignment in mobile-edge computing. In: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). pp. 1–9. IEEE (2019)
27. Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., Leung, K.K.: Dynamic service migration in mobile edge computing based on markov decision process. IEEE/ACM Transactions on Networking 27(3), 1272–1288 (2019). https://doi.org/10.1109/TNET.2019.2916577
28. Yan, J., Bi, S., Duan, L., Zhang, Y.J.A.: Pricing-driven service caching and task offloading in mobile edge computing. IEEE Transactions on Wireless Communications (2021)
29. Zeng, F., Chen, Y., Yao, L., Wu, J.: A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing. Peer-to-Peer Networking and Applications pp. 1–15 (2020)
30. Zhang, M., Zheng, Z., Shroff, N.B.: An online algorithm for power-proportional data centers with switching cost. In: 2018 IEEE Conference on Decision and Control (CDC). pp. 6025–6032 (2018). https://doi.org/10.1109/CDC.2018.8619443
31. Zhao, T., Hou, I.H., Wang, S., Chan, K.: Red/led: An asymptotically optimal and scalable online algorithm for service caching at the edge. IEEE Journal on Selected Areas in Communications 36(8), 1857–1870 (2018)
Appendix

7 BLTN Naive Algorithm

Algorithm 3: Better Late Than Never (BLTN)

1 Input: Sum of switch costs \( W \) units, maximum number of our service requests served by edge server in states \( S_H \) and \( S_L \) as \( \kappa_H \) and \( \kappa_L \), rent costs \( c_H \) and \( c_L \), request arrival sequence: \( \{x_l\}_{l=0}^t, t > 0 \)

2 Output: Service switching strategy \( r_{t+1}, t > 0 \)

3 Initialize: Service switching variable \( r_1 = L \)

4 for each time-slot \( t \) do

5 \( r_{t+1} = r_t \)

6 if \( r_t = H \) then

7 for \( t_{\text{switch}} < \tau < t \) do

8 if

9 \[ \sum_{l=\tau}^{t}(x_l-\kappa_H)^+ + (t-\tau+1) \times c_H \geq \sum_{l=\tau}^{t}(x_l-\kappa_L)^+ + (t-\tau+1) \times c_L + W, \]

10 then

11 \( r_{t+1} = L, t_{\text{switch}} = t \)

12 break

13 end

14 end

15 if \( r_t = L \) then

16 for \( t_{\text{switch}} < \tau < t \) do

17 if

18 \[ \sum_{l=\tau}^{t}(x_l-\kappa_L)^+ + (t-\tau+1) \times c_L \geq \sum_{l=\tau}^{t}(x_l-\kappa_H)^+ + (t-\tau+1) \times c_H + W, \]

19 then

20 \( r_{t+1} = H, t_{\text{switch}} = t \)

21 break

22 end

23 end

end

8 Proof Outlines

In this section we outline the proofs of the results discussed in Section [4]. The proof details follow.
8.1 Proof Outline for Theorem 1 (a)

The time axis is partitioned into ‘frames’. Frame $i$ for $i \in \mathbb{Z}^+$ begins when OPT-OFF switches the state of the edge-server for the $2i^{th}$ time. The time interval before the first frame is denoted as Frame 0.

Fig. 7: Figure representing switching by OPT-OFF and BLTN in the $i^{th}$ frame.

In figure 7, the downward arrows represent switches to state $S_H$, upward arrows indicate switches to state $S_L$. We designate red and black arrows to the BLTN and OPT-OFF policies respectively. The state of the system under policies OPT-OFF and BLTN have been indicated using the two bars below the time axis. The solid red and solid black portions represent the intervals during which BLTN and OPT-OFF host the service in state $S_H$ respectively.

We use the properties of BLTN and OPT-OFF to show that each frame has the following structure (Figure 7):

- BLTN switches state back and forth exactly once.
- BLTN starts in state $S_L$.
- The switch to $S_H$ by BLTN in Frame $i$ is before OPT-OFF switches to state $S_L$ in Frame $i$.
- The switch to $S_L$ by BLTN in Frame $i$ is after OPT-OFF switches to $S_L$ in Frame $i$.

We note that both BLTN and OPT-OFF switch exactly once back and forth in a frame and therefore, the total switch cost under BLTN and OPT-OFF is identical for both policies.

From Lemma 5, we have that after switching to state $S_H$, OPT-OFF hosts the service for at least $\frac{W}{\Delta c}$ time-slots. Similarly, after switching to $S_L$, OPT-OFF hosts the service for at least $\frac{W}{\Delta c}$ (Lemma 6) time-slots.

Across subframes $i.b$ and $i.d$, the rent and service costs are the same for both the policies. However, the differences between the cumulative service and rent costs incurred by BLTN and OPT-OFF in subframes $i.a$ and $i.c$ are capped by $W + \Delta \kappa - \Delta c$ and $W + \Delta c$ respectively. Also, the total switch cost under both policies in a frame is the same. Using these results, we have that the total cost incurred by BLTN and OPT-OFF in a frame differs by at most $2W + \Delta \kappa$. 
In case of the last frame, if OPT-OFF switches the service, the analysis is
the same as that of the previous frame. Otherwise, we bound the ratio of the
cost incurred by BLTN and cost incurred by OPT-OFF in the frame.

Combining the results for individual frames obtained earlier, the final result
follows.

8.2 Proof Outline for Theorem 1 (b)

All deterministic online policies can be partitioned into two subsets. A policy
that hosts the service in $S_H$ during the first time-slot is in the first subset. All
other polices are in the second subset.

In either of the subset, for each policy we construct an arrival sequence
and calculate the ratio of the cost of the deterministic online policy and an
alternative policy. Following from the definition, this quantity is a lower-bound
on the competitive ratio of the deterministic policy.

8.3 Proof Outline for Theorem 2

Through Lemma 15, we identify a lower bound on the cost per time-slot incurred
by any online policy.

First, we consider the case where $\Delta \mu > \Delta c$. Through Hoeffding’s inequality
[12], we upper bound the probability of the state being $S_L$ during time-slot $t$
under BLTN. Conditioned on being in state $S_H$ and not switching to $S_L$ in
time-slot $t$, the expected cumulative cost incurred by BLTN is at the most $c_H + \mathbb{E}[X_t - \min\{X_t, \kappa_H\}]$ and is upper bounded by $W + c_L + \mathbb{E}[X_t - \min\{X_t, \kappa_L\}]$
otherwise. The result follows.

Next, we consider the case where $\Delta \mu < \Delta c$. We upper bound the probability
of the service being hosted in state $S_H$ in time-slot $t$ under BLTN through
Hoeffding’s inequality [12]. Similarly, the result then follows by the fact that the
expected total cost incurred by BLTN is no more than $c_L + \mathbb{E}[X_t - \min\{X_t, \kappa_L\}]$
and at the most $W + c_H + \mathbb{E}[X_t - \min\{X_t, \kappa_H\}]$ otherwise, conditioned on being
in state $S_L$ and not switched to $S_H$ in time-slot $t$.

9 Proofs

9.1 Proof of Theorem 1 (a)

The notation used in this subsection is given in Table 1.

We use the following lemmas to prove Theorem 1 (a).

The following two lemmas give lower bounds of the difference in the number
of requests served at the edge-server in the time interval between switching from
$L \rightarrow H \rightarrow L$, and $H \rightarrow L \rightarrow H$ by OPT-OFF.

**Lemma 2.** If $r^*(n-1) = L$, $r^*(t) = H$ for $n \leq t \leq m$ and $r^*(m+1) = L$, then
$$\sum_{l=n}^{m} (x_{l,H} - x_{l,L}) \geq W + \sum_{l=n}^{m} (c_H - c_L).$$
| Symbol | Description |
|--------|-------------|
| $t$    | Time index  |
| $W_{HL}$ | Switch cost from H to L |
| $W_{LH}$ | Switch cost from L to H |
| $W$    | Sum of switching costs ($W_{HL} + W_{LH}$) |
| $c_t$  | Rent cost per time-slot $t$ |
| $c_L$  | $c_t$ in state $S_L$ |
| $c_H$  | $c_t$ in state $S_H$ |
| $x_t$  | Request arrivals in time-slot $t$ |
| $\mathcal{Z}_{t,L}$ | $\min\{x_t, \kappa_L\}$ |
| $\mathcal{Z}_{t,H}$ | $\min\{x_t, \kappa_H\}$ |
| $\delta_{t,L}$ | $x_t - \mathcal{Z}_{t,L}$ |
| $\delta_{t,H}$ | $x_t - \mathcal{Z}_{t,H}$ |
| $r^*(t)$ | Indicator variable; $H$ if the state is $S_H$ by OPT-OFF during time-slot $t$ and $L$ otherwise |
| $r^{\text{BLTN}}(t)$ | Indicator variable; $H$ if the state is $S_H$ by BLTN during time-slot $t$ and $L$ otherwise |
| $\eta$  | Notation for a policy |
| $C^\eta(n,m)$ | Total cost incurred by the policy $\eta$ in the interval $[n,m]$ |
| $C^{\text{OPT-OFF}}(n,m)$ | Total cost incurred by the offline optimal policy in the interval $[n,m]$ |
| Frame $i$ | The interval between the $i^{th}$ and the $(i+1)^{th}$ switch to $S_H$ by the offline optimal policy |
| $C^{\text{OPT-OFF}}(i)$ | Total cost incurred by the offline optimal policy in Frame $i$ |
| $C^{\text{BLTN}}(i)$ | Total cost incurred by BLTN in Frame $i$ |

Table 1: Notation
Proof. The cost incurred by OPT-OFF in \( n \leq t \leq m + 1 \) is \( W + \sum_{l=n}^{m} c_H + c_L + \sum_{l=n}^{m} \delta_{l,H} + \delta_{m+1,L} \). We prove Lemma 2 by contradiction. Let us assume that \( \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) < W + \sum_{l=n}^{m} (c_H - c_L) \). We construct another policy \( \eta \) which behaves same as OPT-OFF except that \( r_{\eta}(t) = L \) for \( n \leq t \leq m + 1 \). The total cost incurred by \( \eta \) in \( n \leq t \leq m + 1 \) is \( m^{+1} \sum_{l=n}^{m+1} (x_l - x_{l,H}) + \sum_{l=n}^{m} c_L \). It follows that 

\[
C_{\eta}(n,m+1) - C_{\text{OPT-OFF}}(n,m+1) = \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) - W_{LH} - W_{HL} - \sum_{l=n}^{m} (c_H - c_L),
\]

which is negative by our assumption. This contradicts the definition of the OPT-OFF policy, thus proving the result.

Lemma 3. If \( r^*(n-1) = H \), \( r^*(t) = L \) for \( n \leq t \leq m \) and \( r^*(m+1) = H \), then, \( \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) + W \leq \sum_{l=n}^{m} (c_H - c_L) \).

Proof. The cost incurred by OPT-OFF in \( n \leq t \leq m + 1 \) is \( W + \sum_{l=n}^{m} c_H + c_L + \sum_{l=n}^{m} \delta_{l,L} + \delta_{m+1,H} \). We prove Lemma 3 by contradiction. Let us assume that \( \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) + W > \sum_{l=n}^{m} (c_H - c_L) \). We construct another policy \( \eta \) which behaves same as OPT-OFF except that \( r_{\eta}(t) = H \) for \( n \leq t \leq m \). The total cost incurred by \( \eta \) in \( n \leq t \leq m + 1 \) is \( m^{+1} \sum_{l=n}^{m+1} (x_l - x_{l,H}) + \sum_{l=n}^{m} c_H \). It follows that 

\[
C_{\eta}(n,m+1) - C_{\text{OPT-OFF}}(n,m+1) = \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) - W + \sum_{l=n}^{m} (c_H - c_L),
\]

which is negative by our assumption. This contradicts the definition of the OPT-OFF policy, thus proving the result.

The next lemma shows that if the difference in the number of requests that can be served by the edge server in the two states in a time-interval exceeds a certain value (which is a function of the length of that time-interval) and the state is \( S_L \) at the beginning of this time-interval, then OPT-OFF switches to state \( S_H \) the service at least once in the time-interval.

Lemma 4. If \( r^*(n-1) = L \), and \( \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) \geq W + \sum_{l=n}^{m} (c_H - c_L) \), then OPT-OFF switches the state to \( S_H \) at least once in the interval from time-slots \( n \) to \( m \).
Proof. We prove Lemma 4 by contradiction. We construct another policy $\eta$ which behaves same as OPT-OFF except that $r_\eta(t) = H$ for $n < t \leq m$. The total cost incurred by $\eta$ in $n \leq t \leq m$ is $C^\eta(n,m) = W + \sum_{l=n}^{m} c_H + \sum_{l=n}^{m} \delta_{l,H}$. It follows that $C^\eta(n,m) - C^{\text{OPT-OFF}}(n,m) = W + \sum_{l=n}^{m} (c_H - c_L) - \sum_{l=n}^{m} \left( x_{l,H} - x_{l,L} \right)$, which is negative. Hence there exists at least one policy $\eta$ which performs better than OPT-OFF. This contradicts the definition of the OPT-OFF policy, thus proving the result.

The next lemma provides a lower bound on the duration for which OPT-OFF hosts the service once it is fetched.

**Lemma 5.** Once OPT-OFF switches to state $H$, the state is constant for at least $W \left( \kappa_H - \kappa_L \right) - (c_H - c_L)$ slots.

**Proof.** Suppose OPT-OFF switches to $S_H$ at the end of the $(n - 1)^{th}$ time-slot and switches to $S_L$ at the end of time-slot $m > n$. From Lemma 2 $\sum_{l=n}^{m} \left( x_{l,H} - x_{l,L} \right) \geq 0$ and $\sum_{l=n}^{m} (c_H - c_L) \geq (m - n + 1) \times (c_H - c_L)$, i.e, $(m - n + 1) \geq W \left( \kappa_H - \kappa_L \right) - (c_H - c_L)$. This proves the result.

**Lemma 6.** Once OPT-OFF switches to state $L$, the state is constant for at least $W \left( \frac{c_H - c_L}{c_H - c_L} \right)$ slots.

**Proof.** Suppose OPT-OFF switches to $S_L$ at the end of the $(n - 1)^{th}$ time-slot and switches to $S_L$ at the end of time-slot $m > n$. From Lemma 3 $\sum_{l=n}^{m} \left( x_{l,H} - x_{l,L} \right) \geq W \left( \frac{c_H - c_L}{c_H - c_L} \right)$. This proves the result.
Lemma 7. Let $r^{BLTN}(n-1) = H$, $r^{BLTN}(t) = L$ for $n \leq t \leq m$ and $r^{BLTN}(m+1) = H$. Then for any $n \leq n' < m$, $\sum_{l=n'}^{m-1} (\xi_{l,H} - \tilde{\xi}_{l,L}) < \sum_{l=n'}^{m-1} (c_H - c_L) + W + (\kappa_H - \kappa_L)$.

Proof. Given $r^{BLTN}(m) = L$ and $r^{BLTN}(m+1) = H$, then for any $n \leq n' < m$, $\sum_{l=n'}^{m-1} (\tilde{\delta}_{l,L} - \tilde{\delta}_{l,H}) < \sum_{l=n'}^{m-1} (c_H - c_L) + W$. By definition, $\sum_{l=n'}^{m-1} (\xi_{l,H} - \tilde{\xi}_{l,L}) = \sum_{l=n'}^{m-1} (\tilde{\delta}_{l,L} - \tilde{\delta}_{l,H}) + \sum_{l=n'}^{m-1} (c_H - c_L) < \sum_{l=n'}^{m-1} \sum_{l=n'}^{m-1} (c_H - c_L) + W + \sum_{l=n'}^{m-1} (\kappa_H - \kappa_L)$, thus proving the result.

Lemma 8. Let $r^{BLTN}(n-1) = L$, $r^{BLTN}(t) = H$ for $n \leq t \leq m$ and $r^{BLTN}(m+1) = L$. Then for any $n \leq n' < m$, $\sum_{l=n'}^{m-1} (\tilde{\xi}_{l,H} - \tilde{\xi}_{l,L}) < \sum_{l=n'}^{m-1} (c_H - c_L) + W$.

Proof. Given $r^{BLTN}(m) = H$ and $r^{BLTN}(m+1) = L$, then for any $n \leq n' < m$, $\sum_{l=n'}^{m-1} (\delta_{l,H} - \delta_{l,L}) + \sum_{l=n'}^{m-1} (c_H - c_L) < W$. By definition, $\sum_{l=n'}^{m-1} (\tilde{\xi}_{l,H} - \tilde{\xi}_{l,L}) = \sum_{l=n'}^{m-1} (\delta_{l,H} - \delta_{l,L}) + \sum_{l=n'}^{m-1} (c_H - c_L) - W$, thus proving the result.

Consider the event where both BLTN and OPT-OFF have hosted in state $S_H$ in a particular time-slot. The next lemma states that given this, OPT-OFF switches to $S_L$ before BLTN.

Lemma 9. If $r^{BLTN}(n) = H$, $r^*(t) = H$ for $n \leq t \leq m$, and $r^*(m+1) = L$. Then, $r^{BLTN}(t) = H$ for $n+1 \leq t \leq m+1$.

Proof. We prove this by contradiction. Let $\exists \tilde{m} < m$ such that $r^{BLTN}(\tilde{m}+1) = L$.

Then, from Algorithm 3, there exists an integer $\tau > 0$ such that $\sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} (\tilde{\xi}_{l,H} - \tilde{\xi}_{l,L}) < \sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} (c_H - c_L) - W$. The cost incurred by OPT-OFF in the interval $\tilde{m} - \tau + 1$ to $\tilde{m}$ is $\sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} c_H + \sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} \delta_{l,H}$.

Consider an alternative policy $\eta$ for which $r_\eta(t) = 0$ for $\tilde{m} - \tau + 1 \leq t \leq \tilde{m}$, $r_\eta(\tilde{m}+1) = 1$, and $r_\eta(t) = r^*(t)$ otherwise. It follows that $C^\eta - C^{OPT-OFF} = \sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} (\tilde{\xi}_{l,H} - \tilde{\xi}_{l,L}) + W - \sum_{l=\tilde{m}-\tau+1}^{\tilde{m}} (c_H - c_L)$ which is negative by our assumption. This contradicts the definition of the OPT-OFF policy, thus proving the result.
Consider the case where both BLTN and OPT-OFF have hosted in state \( S_H \) in a particular time-slot. From the previous lemma, we know that, OPT-OFF switches states to \( S_L \) before BLTN. The next lemma gives a lower bound on the difference in the number of requests that can be served by the edge server in the interval which starts when OPT-OFF switches states to \( S_L \) from the edge server and ends when BLTN switches states to \( S_L \) from the edge server.

**Lemma 10.** Let \( r^*(n-1) = H \), \( r^*(n) = L \), \( r^{BLTN}(t) = H \) for \( n-1 \leq t \leq m \) and \( r^{BLTN}(m+1) = L \). Then for any \( n \leq n' < m \),

\[
\sum_{l=n'}^{m-1} (x_{l,H} - x_{l,L}) \geq \sum_{l=n'}^{m-1} (c_H - c_L) - W_{HL}.
\]

**Proof.** Given \( r^{BLTN}(m) = H \) and \( r^{BLTN}(m+1) = L \), then for any \( n \leq n' < m \),

\[
\sum_{l=n'}^{m-1} (x_{l,H} - x_{l,L}) > \sum_{l=n'}^{m-1} \Delta c - W. \text{By definition, } \sum_{l=n'}^{m-1} (x_{l,H} - x_{l,L}) = \left( \sum_{l=n'}^{m-1} (x_{l,H} - x_{l,L}) \right) + \left( x_{n,L} - x_{n,H} \right) > \sum_{l=n'}^{m-1} \Delta c - W + 0, \text{thus proving the result.}
\]

Our next result states that BLTN does not switch states to \( S_H \) in the interval between a switch to \( S_L \) and the subsequent switch to \( S_H \) by OPT-OFF.

**Lemma 11.** If \( r^*(n-1) = H \), \( r^*(n) = L \) for \( n \leq t \leq m \), and \( r^*(m+1) = H \), then BLTN does not switch to state \( H \) in time-slots \( n, n+1, \ldots, m-1 \).

**Proof.** We prove this by contradiction. Let BLTN switch states to \( S_H \) in time-slot \( t \) where \( n \leq t \leq m-1 \). Then from Algorithm 3 there exists an integer \( r > 0 \) such that \( t - \tau \geq n \) and

\[
\sum_{l=t-\tau+1}^{t} (x_{l,H} - x_{l,L}) \geq \sum_{l=t-\tau+1}^{t} (c_H - c_L) + W_{LH} + W_{HL}.
\]

If this condition is true, by Lemma 4, OPT-OFF would have fetched the service at least once in the interval \( t - \tau + 1 \) and \( t \) for all \( n \leq t \leq m - 1 \). Hence BLTN does not switch states to \( S_H \) between \( n \) and \( m-1 \).

The next lemma states that in the interval between a switch from \( L \to H \) and subsequent switch from \( H \to L \) by OPT-OFF, BLTN hosts the service for at least one time-slot.

**Lemma 12.** If \( r^*(n-1) = L \), \( r^*(n) = H \) for \( n \leq t \leq m \) and \( r^*(m+1) = L \), then, for some \( n < t \leq m \), \( r^{BLTN}(t) = H \).

**Proof.** We prove this by contradiction. Let \( r^{BLTN}(t) = L \) for all \( n \leq t \leq m \). Then by the definition of the BLTN policy,

\[
\sum_{l=t-\tau+1}^{t} (x_{l,H} - x_{l,L}) < \sum_{l=t-\tau+1}^{m} (c_H - c_L) + W
\]

for any \( \tau > 0 \) and \( \tau \leq t - n + 1 \). If we choose \( t = m \) then

\[
\sum_{l=n}^{m} (x_{l,H} - x_{l,L}) <
\]
\[ \sum_{l=n}^{m} (c_H - c_L) + W_{HL} + W_{LH}, \] which is false from Lemma 2. This contradicts our assumption.

If both BLTN and OPT-OFF are in state \( S_H \) in a particular time-slot, from Lemma 9 we know that OPT-OFF switches states to \( S_L \) before BLTN. The next lemma states that BLTN switches states to \( S_L \) before the next time OPT-OFF switches to state \( S_H \).

**Lemma 13.** If \( r^*(n-1) = H \), \( r^*(t) = L \) for \( n \leq t \leq m \), \( r^*(m+1) = H \), and \( r_{BLTN}^{*}(n-1) = H \), then, BLTN switches states to \( S_L \) by the end of time-slot \( m \) and \( r_{BLTN}^{*}(m+1) = L \).

**Proof.** We prove this by contradiction. Assume that BLTN does not switch states to \( S_L \) in any time slot \( t \) for all \( n \leq t \leq m \). Then from the definition of the BLTN policy, \[ \sum_{l=\tau-1+1}^{t} (x_{l,H} - x_{l,L}) + W = \sum_{l=\tau-1+1}^{t} (c_H - c_L) \] for all \( \tau \) such that \( 0 < \tau \leq t - n + 1 \). As a result, at \( t = m \), \[ \sum_{l=n}^{m} (x_{l,H} - x_{l,L}) + W_{HL} + W_{LH} \geq \sum_{l=n}^{m} (c_H - c_L). \]

Given this, it follows that OPT-OFF will not switch states to \( S_L \) at the end of time-slot \( n - 1 \). This contradicts our assumption. By Lemma 11, BLTN does not switch states to \( S_H \) in the interval between switches from \( S_H \) to \( S_L \) and back by OPT-OFF. Therefore, \( r_{BLTN}^{*}(m+1) = 0 \).

To compare the costs incurred by BLTN and OPT-OFF we divide time into frames \([1, t_2 - 1], [t_1, t_2 - 1], [t_2, t_3 - 1], \ldots\), where \( t_i - 1 \) is the time-slot in which OPT-OFF switches to state \( S_H \) for the \( i^{th} \) time for \( i \in \{1, 2, \ldots\} \). Our next result characterizes the sequence of events that occur in any such frame.

**Lemma 14.** Consider the interval \([t_i, t_{i+1} - 1]\) such that OPT-OFF switches the state to \( S_H \) at the end of time-slot \( t_i - 1 \) and switches the state to \( S_L \) again the end of time-slot \( t_{i+1} - 1 \). By definition, there exists \( \tau \in [t_i, t_{i+1} - 2] \) such that OPT-OFF switches the state to \( S_L \) in time-slot \( \tau \). BLTN switches to \( S_H \) from \( S_L \) and back exactly once each in \([t_1, t_2 - 1]\). The switch to \( S_H \) by BLTN is in time-slot \( t_{BLTN}^{*} \) such that \( t_1 \leq t_{BLTN}^{*} \leq \tau \) and the switch to \( S_L \) by BLTN is in time-slot \( t_{BLTN}^{*} \) such that \( \tau < t_{BLTN}^{*} < t_2 \) (Figure 8).

**Proof.** Without loss of generality, we prove the result for \( i = 1 \). Since \( r^*(t_1 - 1) = L \), \( r^*(t) = H \) for \( t_1 \leq t \leq \tau \) and \( r^*(\tau) = L \) then by Lemma 12, \( r_{BLTN}(t_{BLTN}^{*}) = H \) for some \( t_1 < t_{BLTN}^{*} \leq \tau \). In addition, by Lemma 13, \( r_{BLTN}(t_1) = L \). Therefore, BLTN the state to \( S_H \) at least once in the interval \([t_1, t_2 - 1]\). By Lemma 9 if \( t_{BLTN}^{*} < \tau \), since both BLTN and OPT-OFF stay in state \( S_H \) during time-slot \( t_{BLTN}^{*} + 1 \), OPT-OFF switches to \( S_L \) before BLTN, therefore, once fetched, BLTN does not switch to \( S_L \) before time-slot \( \tau + 1 \), i.e., \( r_{BLTN}(t) = H \) for \( t_{BLTN}^{*} + 1 \leq t \leq \tau + 1 \). Since \( r^*(\tau) = H \), \( r^*(t) = L \) for \( \tau + 1 \leq t \leq t_2 - 1 \) and
Proof (of Theorem 1). As mentioned above, to compare the costs incurred by BLTN and OPT-OFF we divide times into frames $[1, t_1 - 1], [t_1, t_2 - 1], [t_2, t_3 - 1], \ldots$, where $t_i - 1$ is the time-slot in which OPT-OFF downloads the service for the $i$th time for $i \in \{1, 2, \ldots, k\}$. For convenience, we account for the switch costs incurred by OPT-OFF in time-slot $t_i$ in the cost incurred by OPT-OFF in Frame $i$. Given this, the cost under BLTN and OPT-OFF is the same for $[1, t_1 - 1]$ (Frame 0) since both policies host the service in the same state.

Note that if the total number of switches made by OPT-OFF is less than $2k < \infty$, there are exactly $k + 1$ frames (including Frame 0). The $(k + 1)$th frame either has no switch by OPT-OFF to $S_L$ or OPT-OFF switches to $S_L$ and then never switches back.

We now focus on Frame $i$, such that $0 < i < k$, where $2k$ is the total number of switches made by OPT-OFF.

Note: $\Delta \kappa = \kappa_H - \kappa_L, \Delta c = c_H - c_L$

Without loss of generality, we focus on Frame 1. Recall the definitions of $\tau$, $i_{HL}^{BLTN}$, and $i_{LH}^{BLTN}$ from Lemma 14, also seen in Figure 8. By Lemma 14, we have that BLTN switches from $S_H$ to $S_L$ and back exactly once each in $[t_1, t_2 - 1]$ such that the switch to $S_H$ by BLTN is in time-slot $i_{HL}^{BLTN}$ such that $t_1 \leq i_{HL}^{BLTN} \leq \tau$ and the eviction by BLTN is in time-slot $i_{LH}^{BLTN}$ such that $\tau < i_{LH}^{BLTN} < t_2$.

Both OPT-OFF and BLTN makes one pair of switches in the frame. Hence the difference in the switch costs is zero. We now focus on the service and rent cost incurred by the two policies.
By Lemma 7, the difference in total cost $= C^{BLTN} - C^{OPT-OFF} = \sum_{l=t_1}^{t_{BLTN}} (\delta_{l,L} - \delta_{l,H}) + \sum_{l=t_1}^{t_{BLTN}} (c_L - c_H) < W + \Delta \kappa - \Delta c$.

The service and rent cost incurred by OPT-OFF and BLTN in $[t_{BLTN} + 1, \tau]$ are equal.

By Lemma 8, the difference of costs in $[\tau + 1, t_{BLTN}]$ is $C^{BLTN} - C^{OPT-OFF} = \sum_{l=\tau+1}^{t_{BLTN}} (\delta_{l,H} - \delta_{l,L}) + \sum_{l=\tau+1}^{t_{BLTN}} (c_H - c_L) < \sum_{l=\tau+1}^{t_{BLTN}} (c_L - c_H) + W + \sum_{l=\tau+1}^{t_{BLTN}} (c_H - c_L) = W + \Delta c$.

The difference in total cost $= C^{BLTN} - C^{OPT-OFF} = \sum_{l=t_{BLTN}}^{t_{BLTN}-1} (\delta_{l,L} - \delta_{l,H}) + \sum_{l=t_{BLTN}}^{t_{BLTN}-1} (c_L - c_H) = W + \Delta \kappa - \Delta c$.

The service and rent cost incurred by OPT-OFF and BLTN in $[t_{BLTN} + 1, t_2 - 1]$ are equal.

Let $C^{BLTN}(i)$, $C^{OPT-OFF}(i)$ denote the costs incurred in the $i$th frame by BLTN and OPT-OFF respectively. We therefore have that,

$$C^{BLTN}(i) - C^{OPT-OFF}(i) \leq 2W + \kappa_H - \kappa_L$$  \hspace{1cm} (6)

By Lemma 5, once OPT-OFF switches to state H, it stays put for at least $\tau_H = \frac{W}{\Delta \kappa - \Delta c}$ slots. And by Lemma 6, once OPT-OFF switches to state L, it stays put for at least $\tau_L = \frac{W}{\Delta \kappa}$ slots. Therefore,

$$C^{OPT-OFF}(i) \geq W_{LH} + c_H \tau_H + W_{HL} + c_L \tau_L$$

From (6) and (7),

$$C^{BLTN}(i) \leq \left( 1 + \frac{2W + \Delta \kappa}{W (1 + \frac{c_H}{\Delta \kappa} - \frac{c_L}{\Delta c})} \right) C^{OPT-OFF}(i).$$  \hspace{1cm} (7)

Frames 1 to $k - 1$ have now been characterized completely.

For Frame $k$, which is the last frame, there are two possible cases, one where OPT-OFF switches states to $S_L$ in Frame $k$, in which case the analysis for Frame $k$ is identical to that of Frame 1, and the other when OPT-OFF does not switch states to $S_L$ in Frame $k$. We now focus on the latter.

Given that OPT-OFF switches to state $S_H$ the service in time-slot $t_k - 1$, there exists $m > t_k$ such that $\sum_{l=t_k}^{m} (\varnothing_{l,H} - \varnothing_{l,L}) \geq W + \sum_{l=t_k}^{m} (c_H - c_L)$. By Step 8 in Algorithm 3, BLTN switches to state $S_H$ at the end of time-slot $m$. Let $\tau_k = m - t_k$. By Lemma 8, the difference in the number of requests that can be served by the edge server during these $\tau_k$ time-slots is at most $\sum_{l=\tau_k}^{m-1} (c_H - c_L) + W$. Since BLTN is in state $S_L$ during these $\tau_k$ time-slots, the rent cost
Renting Edge Computing Resources for Service Hosting

incurred by BLTN is \( c_L \) per slot and the service cost incurred by BLTN is at most \( W + \sum_{l=t_k}^{m-1} c_L + \kappa_L + \sum_{l=t_k}^{m} \delta_{l,H} \). OPT-OFF rents the edge server during these \( \tau_k \) time-slots at cost \( \sum_{l=t_k}^{m} c_H \) and the service cost incurred by OPT-OFF is \( \sum_{l=t_k}^{m-1} \delta_{l,H} \).

There is no difference between the cost of BLTN and OPT-OFF after the first \( \tau_k \) slots in Frame \( k \). It follows that

\[
C_{BLTN}(k) - C_{OPT-OFF}(k) \leq 2W + \Delta \kappa. \tag{8}
\]

From (7) and (8),

\[
C_{BLTN}(k) \leq \left( 1 + \frac{2W + \Delta \kappa}{W \left( 1 + \frac{c_H}{\Delta \kappa - \Delta c} + \frac{c_L}{\Delta \kappa} \right)} \right) C_{OPT-OFF}(k). \tag{9}
\]

Stitching together the results obtained for all frames, the result follows.

9.2 Proof of Theorem 1(b)

**Proof.** Let \( \mathcal{P} \) be a given deterministic online policy and \( C^P(a) \) be the cost incurred by this policy for the request sequence \( a \).

We first consider the case where \( \mathcal{P} \) starts in state \( S_H \).

We define \( t^{(1)} \geq 1 \) as the first time the policy \( \mathcal{P} \) switches to state \( S_H \) when there are \( \kappa_H \) arrivals in each of the first \( t^{(1)} \) time-slots. As \( \mathcal{P} \) is a deterministic policy, the value of \( t^{(1)} \) can be computed a-priori.

We define \( t^{(2)} \geq 1 \) as the first time the policy \( \mathcal{P} \) switches to state \( S_L \) when there are \( \kappa_L \) arrivals in each of the first \( t^{(2)} \) time-slots. Similarly, as \( \mathcal{P} \) is a deterministic policy, the value of \( t^{(2)} \) can be computed a-priori.

Consider the arrival process \( a \) with \( \kappa_H \) request arrivals each in the first \( t^{(1)} \) time-slots and \( \kappa_L \) request arrivals each in the next \( t^{(2)} \) time-slots.

Note: \( \Delta \kappa = \kappa_H - \kappa_L, \Delta c = c_H - c_L, W = W \)

It follows that \( C^P(a) = t^{(1)}(\Delta \kappa + c_L) + t^{(2)}c_H + W \).

Consider an alternative policy ALT which is in state \( S_H \) from time-slots 1 to \( t^{(1)} \) and in state \( S_L \) from time-slots \( t^{(1)} + 1 \) to \( t^{(1)} + t^{(2)} \). It follows that \( C^{ALT}(a) = c_H t^{(1)} + c_L t^{(2)} + W \). By definition, \( \rho^P \geq \frac{(\Delta \kappa + c_L) t^{(1)} + c_H t^{(2)} + W}{c_H t^{(1)} + c_L t^{(2)} + W} \).

Therefore,

\[
\rho^P \geq \min \left\{ \frac{\Delta \kappa + c_L}{c_H}, \frac{c_H}{c_L}, \frac{\Delta \kappa + c_L + c_H + W}{c_H + c_L + W} \right\}
\]
9.3 Proof of Theorem 2

We use the following lemmas to prove Theorem 2.

**Lemma 15.** Let $X_t$ be the number of requests arriving in time-slot $t$, $\nu = \mathbb{E}[X_t]$, $X_{t,H} = \min\{X_t, \kappa_H\}$ and $\mu_L = \mathbb{E}[X_{t,L}]$, $X_{t,L} = \min\{X_t, \kappa_L\}$ and $\mu_L = \mathbb{E}[X_{t,L}]$. Let the rent cost per time-slot be $c_H$ or $c_L$ depending on the states $S_H$ or $S_L$ respectively. Under Assumption 1, let $\mathbb{E}[C_i^{OPT-ON}]$ be the cost per time-slot incurred by the OPT-ON policy. Then, $\mathbb{E}[C_i^{OPT-ON}] \geq \min\{c_H + \nu - \mu_H, c_L + \nu - \mu_L\}$.

**Proof.** If the service is hosted in state $S_H$ on the edge in time-slot $t$, the expected cost incurred is at least $\mathbb{E}[X_t - \min\{X(t), \kappa_H\} + c_H] = c_H + \nu - \mu_H$. Else, if the service is hosted in state $S_L$ on the edge in time-slot $t$, the expected cost incurred is at least $\mathbb{E}[X_t - \min\{X(t), \kappa_L\} + c_L] = c_L + \nu - \mu_L$.

**Lemma 16.** Let $X_t$ be the number of requests arriving in time-slot $t$, $X_{t,H} = \min\{X_t, \kappa_H\}$, $\mu_H = \mathbb{E}[X_{t,H}]$, $X_{t,L} = \min\{X_t, \kappa_L\}$ and $\mu_L = \mathbb{E}[X_{t,L}]$. Let the rent cost per time-slot be $c_H$ or $c_L$ depending on the states $S_H$ or $S_L$ respectively. Define $\Delta X_t = X_{t,H} - X_{t,L}$, $\Delta \nu = \mu_H - \mu_L$, $\Delta \kappa = \kappa_H - \kappa_L$, $\Delta c = c_H - c_L$, $Y_t = \Delta X_t - \Delta c$, and $Y = \sum_{t=1}^{\tau} Y_t$ then $Y$ satisfies, for $(\Delta c - \Delta \nu) + W > 0$, $\mathbb{P}(Y \leq \tau c - W) \leq \exp\left(-\frac{2((\Delta c - \Delta \nu) + W)^2}{(\Delta \kappa)^2}\right)$, and for $(\Delta \nu - \Delta c) + W > 0$, $\mathbb{P}(Y \leq \tau c - W) \leq \exp\left(-\frac{2((\Delta \nu - \Delta c) + W)^2}{(\Delta \kappa)^2}\right)$.

**Proof.** Using i.i.d. condition of $\{X_t\}_{t \geq 1}$, it follows that for $s > 0$, $\mathbb{E}[\exp(s Y)] \leq \prod_{t=1}^{\tau} \mathbb{E}[\exp(s Y_t)]$. Moreover, $Y_t \in [-\Delta c, \Delta \kappa - \Delta c]$. Then the result follows by Hoeffding’s inequality.

**Proof (of Lemma 16).** We first consider the case when $\Delta \nu > \Delta c$. We define the following events

$E_{t_1,t_2} = \sum_{l=t_1}^{t_2} \Delta X_l \leq \sum_{l=t_1}^{t_2} \Delta c - W, E^r = \bigcup_{t_1=1}^{\tau} E_{t_1\tau}, E_{t-1} = \bigcup_{t=1}^{\tau} E^r,$

$E_t = \bigcup_{\tau = t-\lfloor \frac{\Delta \nu}{\Delta c} \rfloor}^{t_2} E^{-1}, F : \sum_{l=t_1}^{t_2} \Delta X_l \geq \sum_{l=t_1}^{t_2} \Delta c + W.$

By Lemma 16 it follows that $\mathbb{P}(E_{t_1,t_2}) \leq \exp\left(-\frac{2((\Delta \nu - \Delta c)^2(t_2 - t_1 + 1))}{(\Delta \kappa)^2}\right)$, and therefore,

$$\mathbb{P}(E^r) \leq \sum_{t_1=1}^{\tau-\lfloor \frac{\Delta \nu}{\Delta c} \rfloor+1} \exp\left(-\frac{2((\Delta \mu - \Delta c)^2(\tau - t_1 + 1))}{(\Delta \kappa)^2}\right) \leq \exp\left(-\frac{2((\Delta \mu - \Delta c)^2)}{(\Delta \kappa)^2}\right).$$

(10)
Using 10 and the union bound, \( P(E_{t-1}) \) and \( P(E_t) \) are upper bounded by

\[
\left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \frac{\exp \left( -2 \frac{(\Delta \mu - \Delta c)^2 W}{(\Delta \kappa)^2} \right)}{1 - \exp \left( -2 \frac{(\Delta \mu - \Delta c)^2}{(\Delta \kappa)^2} \right)}.
\]

(11)

By Lemma 16,

\[
P(F_c) \leq \exp \left( -2 \frac{(\Delta \mu - \Delta c)^2 [\frac{\lambda W}{\Delta \mu - \Delta c}] - W^2}{\frac{\lambda W}{\Delta \mu - \Delta c} (\Delta \kappa)^2} \right) \leq \exp \left( -2 \frac{(\lambda - 1)^2 W (\Delta \mu - \Delta c)}{\lambda (\Delta \kappa)^2} \right).
\]

(12)

By (11) and (12),

\[
P(E_t^c \cap E_{t-1}^c \cap F) \geq 1 - 2 \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \frac{\exp \left( -2 \frac{(\Delta \mu - \Delta c)^2 W}{(\Delta \kappa)^2} \right)}{1 - \exp \left( -2 \frac{(\Delta \mu - \Delta c)^2}{(\Delta \kappa)^2} \right)} - \exp \left( -2 \frac{(\lambda - 1)^2 W (\Delta \mu - \Delta c)}{\lambda (\Delta \kappa)^2} \right).
\]

(13)

Consider the event \( G = E_t^c \cap E_{t-1}^c \cap F \) and the following three cases.

Case 1: The service is hosted in state \( S_H \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \). Conditioned on \( E_{t-1}^c \), by the properties of the BLTN policy, the service is not switched to \( S_L \) in time-slots \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \) to \( t - 1 \). It follows that in this case, the service is hosted in state \( S_H \) during time-slot \( t \).

Case 2: The service is hosted in state \( S_L \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \) and the state is switched to \( S_H \) in time-slot \( \tilde{\tau} \) such that \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] + 1 \leq \tilde{\tau} \leq t - 2 \). Conditioned on \( E_{t-1}^c \), by the properties of the BLTN policy, the service is not switched to \( S_L \) in time-slots \( \tilde{\tau} + 1 \) to \( t - 1 \). It follows that in this case, the service is hosted in state \( S_H \) during time-slot \( t \).

Case 3: The service is hosted in state \( S_L \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \) and is not switched to \( S_H \) in time-slots \( t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] + 1 \) to \( t - 2 \): In this case, in time-slot \( t - 1, t_{\text{evict}} \leq t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \). Conditioned on \( F \), by the properties of the BLTN policy, condition in Step 8 in Algorithm 3 is satisfied for \( \tau = t - \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \). It follows that in this case, the decision to switch states is made in time-slot \( t - 1 \) and therefore, the service is hosted in state \( S_H \) during time-slot \( t \).

We thus conclude that conditioned on \( G = E_{t-1}^c \cap F \), the service is hosted in state \( S_H \) during time-slot \( t \). In addition, conditioned on \( E_t^c \), the service is not switched in time-slot \( t \). We now compute the expected cost incurred by the BLTN policy. By definition, \( E[C_t^{\text{BLTN}}] = E[C_t^{\text{BLTN}}|G]P(G) + E[C_t^{\text{BLTN}}|G^c] \times P(G^c) \).
Note that, $E[C_{t}^{\text{BLTN}}|G] = c_H + \nu - \mu_H$, $E[C_{t}^{\text{BLTN}}|G^c] \leq W + c_L + \nu - \mu_L$. Therefore,

$$E[C_{t}^{\text{BLTN}}] = c_H + \nu - \mu_H + (W + \Delta \mu - \Delta c)P(G^c)$$

$$\leq c_H + \nu - \mu_H + (W + \Delta \mu - \Delta c) \times \left( 2 \left[ \frac{\lambda W}{\Delta \mu - \Delta c} \right] \exp\left( -2 \frac{(\Delta \mu - \Delta c)^2 W}{\lambda W} \right) \right.$$\

$$+ \exp\left( -2 \frac{(\lambda - 1)^2 W (\Delta \mu - \Delta c)}{\lambda (\Delta \kappa)^2} \right) \bigg).$$ (14)

We optimize over $\lambda > 1$ to get the tightest possible bound. By Lemma 15 and 14, we have the result for BLTN.

Next, we consider the case when $\Delta \mu < \Delta c$. We define the following events $F_{t_1,t_2} : \sum_{l=t_1}^{t_2} \Delta X_l \geq \sum_{l=t_1}^{t_2} \Delta c + W$, $F^\tau = \bigcup_{t_1=1}^{\tau} F_{t_1,\tau}$, $F_{t-1} = \bigcup_{\tau=t-\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \rceil + 1}^{t-1} F^\tau$, $F_1 = \bigcup_{\tau=t-\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \rceil + 1}^{t} F^\tau$, $E: \sum_{l=t-\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \rceil}^{\tau+1} \Delta X_l + W < \sum_{l=t-\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \rceil}^{\tau+1} \Delta c$.

By Lemma 16, it follows that $P(F_{t_1,t_2}) \leq \exp\left( -2 \frac{(\Delta c - \Delta \mu)^2 (t_2 - t_1 + 1)}{(\Delta \kappa)^2} \right)$, and therefore,

$$P(F^\tau) \leq \sum_{t_1=1}^{\tau-\lceil \frac{\lambda W}{\Delta \mu - \Delta c} \rceil + 1} \exp\left( -2 \frac{(\Delta c - \Delta \mu)^2 (t_1 + 1)}{(\Delta \kappa)^2} \right)$$

$$\leq \frac{\exp\left( -2 \frac{(\Delta c - \Delta \mu)^2 W}{\Delta \kappa^2} \right)}{1 - \exp\left( -2 \frac{(\Delta c - \Delta \mu)^2}{\Delta \kappa^2} \right)}.$$ (15)

Using (15) and the union bound, $P(F_t)$ and $P(F_{t-1})$ are upper bounded by

$$\left[ \frac{\lambda W}{\Delta \mu - \Delta \kappa} \right] \exp\left( -2 \frac{(\Delta \mu - \Delta \mu)^2 W}{\Delta \mu - \Delta c} \right) \frac{1}{1 - \exp\left( -2 \frac{(\Delta c - \Delta \mu)^2}{\Delta \kappa^2} \right)}.$$ (16)

By Lemma 16,

$$P(E^c) \leq \exp\left( -2 \frac{((\Delta c - \Delta \mu)[\frac{\lambda W}{\Delta \mu - \Delta c} - W]^2}{\Delta \mu - \Delta \kappa (\Delta \kappa)^2} \right) \leq \exp\left( -2 \frac{(\lambda - 1)^2 (\Delta c - \Delta \mu) W}{\lambda (\Delta \kappa)^2} \right).$$ (17)
By (16) and (17),

\[
\mathbb{P}(F_i^c \cap F_{t-1}^c \cap E) \geq 1 - \exp \left( -2 \frac{(\lambda - 1)^2(\Delta c - \Delta \mu)W}{\lambda (\Delta \kappa)^2} \right) - 2 \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \frac{\exp \left( -2 \frac{(\Delta c - \Delta \mu)^2 W}{(\Delta \kappa)^2} \right)}{1 - \exp \left( -2 \frac{(\Delta c - \Delta \mu)^2}{(\Delta \kappa)^2} \right)}.
\]

(18)

Consider the event \( G = F_i^c \cap F_{t-1}^c \cap E \) and the following three cases.

Case 1: The service is hosted in state \( S_L \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \): Conditioned on \( F^c \), by the properties of the BLTN policy, the service is not switched to \( S_H \) in time-slots \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \) to \( t - 1 \). It follows that in this case, the service is hosted in state \( S_L \) during time-slot \( t \).

Case 2: The service is hosted in state \( S_H \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \) and the state is switched to \( S_L \) in time-slot \( \tilde{t} \) such that \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] + 1 \leq \tilde{t} \leq t - 2 \); Conditioned on \( F^c \), by the properties of the BLTN policy, the state is not switched to \( S_H \) in time-slots \( \tilde{t} + 1 \) to \( t - 1 \). It follows that in this case, the service is hosted in state \( S_L \) during time-slot \( t \).

Case 3: The service is hosted in state \( S_H \) during time-slot \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \) and the state is not switched to \( S_L \) in time-slots \( t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] + 1 \) to \( t - 2 \); In this case, in time-slot \( t - 1 \), \( t_{\text{evict}} \leq t - \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \). Conditioned on \( E \), by the properties of the BLTN policy, condition in Step 16 in Algorithm 3 is satisfied for \( \tau = t - \left[ \frac{\lambda W}{2 \Delta \mu - \Delta c} \right] \).

It follows that in this case, the decision to switch states is made in time-slot \( t - 1 \) and therefore, the service is hosted in state \( S_L \) in time-slot \( t \).

We thus conclude that conditioned on \( F_{t-1}^c \cap E \), the service is hosted in state \( S_L \) during time-slot \( t \). In addition, conditioned on \( F_t^c \), the service is not switched to \( S_H \) in time-slot \( t \). We now compute the expected cost incurred by the BLTN policy. By definition, \( \mathbb{E}[C_t^{\text{BLTN}}] = \mathbb{E}[C_t^{\text{BLTN}} | G] \mathbb{P}(G) + \mathbb{E}[C_t^{\text{BLTN}} | G^c] \mathbb{P}(G^c) \).

Note that, \( \mathbb{E}[C_t^{\text{BLTN}} | G] = \nu - \mu_L + c_L \), \( \mathbb{E}[C_t^{\text{BLTN}} | G^c] \leq c_H + \nu - \mu_H + W \).

Therefore,

\[
\mathbb{E}[C_t^{\text{BLTN}}] = \nu - \mu_L + c_L + (\Delta c - \Delta \mu + W) \mathbb{P}(G^c) \\
\leq \nu - \mu_L + c_L + (\Delta c - \Delta \mu + W) \times \left( \exp \left( -2 \frac{(\lambda - 1)^2(\Delta c - \Delta \mu)W}{\lambda (\Delta \kappa)^2} \right) \right)
\]

\[
+ 2 \left[ \frac{\lambda W}{\Delta c - \Delta \mu} \right] \frac{\exp \left( -2 \frac{(\Delta c - \Delta \mu)^2 W}{(\Delta \kappa)^2} \right)}{1 - \exp \left( -2 \frac{(\Delta c - \Delta \mu)^2}{(\Delta \kappa)^2} \right)}.
\]

(19)

We optimize over \( \lambda > 1 \) to get the tightest possible bound. By Lemma 15 and 19, we have the result for BLTN.