Formulating Cost-Effective Data Distribution Strategies Online for Edge Cache Systems

Xiaoyu Xia, *Member, IEEE*, Feifei Chen, *Senior Member, IEEE*, Qiang He, *Senior Member, IEEE*, John Grundy, *Senior Member, IEEE*, Mohamed Abdelrazek, Jun Shen, *Senior Member, IEEE*, Athman Bouguettaya, *Fellow, IEEE*, and Hai Jin, *Fellow, IEEE*

Abstract—Edge Computing (EC) enables a new kind of caching system in close geographic proximity to end-users by allowing app vendors to cache popular data on edge servers deployed at base stations. This edge cache system can better support latency-sensitive applications. However, transmitting data from the centralized cloud to the edge servers without proper transmission strategies may cost app vendors dearly. Cost-effective data distribution strategies are of particular importance for applications, whose data to be cached at the edge often changes dynamically. In this paper, we study this *Online Edge Data Distribution* (OEDD) problem, aiming to minimize app vendors' total transmission cost, while ensuring low transmission latency in the long term. We first model this problem and prove its $\mathcal{NP}$-hardness. We then combine Lyapunov optimization and game theory to propose a novel Latency-Aware Online (LAO) approach for solving this OEDD problem over time in a distributed manner with provable performance guarantees. The evaluation of LAO based on a real-world dataset demonstrates that it can help app vendors formulate cost-effective edge data distribution strategies in an online manner.

Index Terms—edge cache system, data distribution, optimization, online algorithm.

1 Introduction

The number of mobile devices has increased exponentially, approaching 500 billion by 2030 as predicted by CISCO [1]. These devices generate an enormous load on networks. Considerable network resources are required to transmit such massive data. Conventional caching systems facilitated by cloud computing cannot fulfill the rapidly-increasing need for low latency raised by real-time applications [2], such as virtual reality, real-time gaming, Industry 4.0, etc. Edge Computing (EC) has emerged as the extension of cloud computing to tackle this critical limitation. EC distributes resources to edge servers deployed at base stations in geographic proximity to nearby end-users [3], [4]. App vendors like Facebook Horizon can hire storage and computing capacities on edge servers for caching popular data and hosting their applications to serve users within those edge servers' coverage areas with low end-to-end data retrieval latency [5].

- *X. Xia is with School of Computer Science, The University of Adelaide, Australia. E-mail: xiaoyu.xia@adelaide.edu.au.*
- *F. Chen and M. Abdelrazek are with School of Information Technology, Deakin University, Australia. E-mail: {feifei.chen, mohamed.abdelrazek}@deakin.edu.au.*
- *Q. He is with Department of Computing Technologies, Swinburne University of Technology, Australia. E-mail: qhe@swin.edu.au.*
- *J. Grundy is with Faculty of Information Technology, Monash University, Australia. E-mail: john.grundy@monash.edu.*
- *J. Shen is with School of Computing and Information Technology, University of Wollongong, Australia. E-mail: jshen@uow.edu.au.*
- *A. Bouguettaya is with the School of Computer Science, University of Sydney, Australia. Email: athman.bouguettaya@sydney.edu.au.*
- *H. Jin is with School of Computer Science and Technology, Huazhong University of Science and Technology, China. Email: hjin@hust.edu.cn.*
- *Qiang He and Feifei Chen are the corresponding authors.*

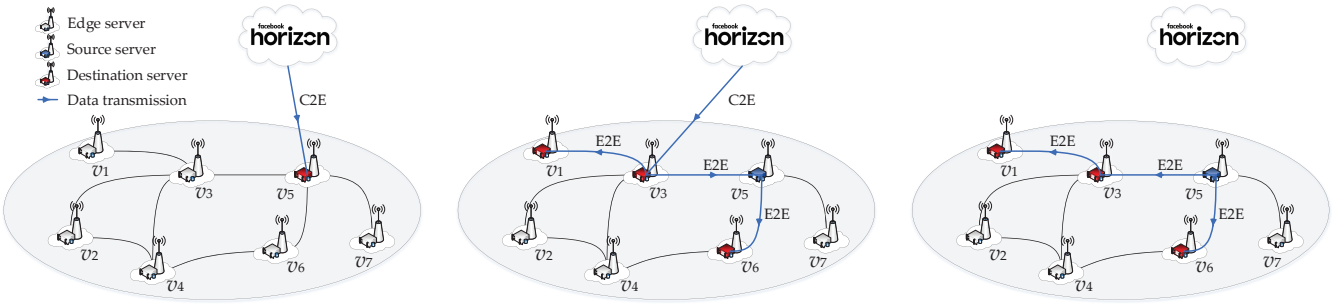
Data caching has been intensively investigated to reduce data retrieval latency [6], [7]. A fundamental obstacle to further advances in caching is the often unpredictable high latency between end-users and the cache [8]. EC addresses this issue by expanding the boundary of data caching to reach geographical proximity to end-users [9], [10], [11]. As edge servers become the entry points for an increasing number of devices, a lot of traffic go through edge servers. Caching data on edge servers considerably reduces users' data retrieval latency as there is no need to retrieve data from remote cloud servers. Additionally, this could have the effect of reducing the volume of data transmitted from the cloud to users and decreasing the corresponding *transmission costs* [12].

Edge servers deployed in an area constitute an *edge cache system*. Existing research on edge cache systems has focused on caching data across edge servers for different optimization objectives, e.g., maximizing hit ratio [6], [11], minimizing caching cost [13] or minimizing retrieval latency [14]. Excessive costs incurred by transmitting data to edge servers from the cloud has been completely ignored. For example, 1 GB data transferred out charges up to US$0.11 under Amazon Web Services price model[1]. It is an important consideration for app vendors when leveraging edge cache systems. Moreover, taking too long to transmit data onto edge servers impacts users' data retrieval latency and causes user abandonment [15].

Example. Consider the example of Facebook Horizon (FH), a VR platform. FH application can benefit significantly from caching data like popular[2] VR videos (and the

1. https://aws.amazon.com/s3/pricing/

2. Data popularity prediction has been investigated extensively in the past two decades [16]. Thus, the data to be distributed by LAO is assumed to be popular in this study.



(a) EDD for v_5 in time slot t_0 (b) EDD for v_1, v_3 and v_6 in time slot t_1 (c) OEDD for v_1, v_3 and v_6 in time slot t_1

Fig. 1: Example EDD and OEDD processes. Note: This example presents the distribution of one VR video over time. The processes for distributing different VR videos are not correlated. Thus, their strategies are individually formulated.

corresponding HTML, CSS, JS and image files) on edge servers [17]. FH users in the edge cache system can retrieve popular VR videos from edge servers with low data retrieval latency. It also considerably reduces the data transmission cost incurred by the data traffic between FH's cloud server and FH users. In Fig. 1(a), an edge cache system involves a set of edge servers $\{v_1, ..., v_7\}$ covering different geographic areas. In the edge cache system, the edge servers constitute the edge server network [17]. It overcomes single-point failures in edge-cloud architectures , where edge servers can communicate with others through a centralized macro base station. In time slot t_0, a VR video d is requested by many of the FH users within edge server v_5's coverage areas. As illustrated by Fig. 1(a), d is transmitted from the remote FH cloud server to be cached in v_5 via a C2E (cloud to edge server) transmission. Future FH users covered by v_5 can retrieve d with low latency.

In [17], we study an *edge data distribution* (EDD) problem, aiming to transmit data cost-effectively to *destination edge servers*, i.e., edge servers to cache that data, from the cloud. However, the approach proposed in [17] is designed to handle data transmissions in an offline manner without considering temporal data dynamics in real-world: various users' demands on data over time [18]. Every time edge servers request a piece of data in the system, it has to be transmitted to the destination edge servers from the cloud through intermediate ones. Take Fig. 1(b) for example, where edge servers v_1, v_3 and v_6 need to cache VR video d after v_5 in time slot t_1. As demonstrated, the approach proposed in [17] has to transmit d from the cloud to v_3 first via a C2E transmission. Then, v_3 will transmits d to v_1, and v_6 through v_5 via E2E (edge server to edge server) transmissions. As d goes viral over time, d is requested by many other edge servers in the system, e.g., v_2, v_4 and v_7. The inevitable and expensive C2E transmissions can easily incur excessive transmission costs. In the meantime, the E2E transmissions incur *extra* data retrieval latency for users served by v_1 and v_6 compared to those served by v_3 (and those served by v_5 in Fig. 1(a)). This undermines EC's pursuit of low data retrieval latency for users.

Data transmissions in a real-world edge cache system must be handled in an online manner to overcome the above critical limitations. The key is to source data for destination edge servers from other edge servers rather than the cloud if it is more cost-effective to do so. As illustrated by Fig. 1(c), d can be transmitted from v_5 to v_1, v_3 and v_6. Compared with the strategy presented in Fig. 1(b), this strategy is more cost-effective, because it greatly reduces the total transmission cost and transmission latency by avoiding slow and expensive C2E transmissions when possible. However, sourcing data from an edge server far away from the destination servers may take excessive time and incurs high *transmission latency* which are ignored in [17]. Thus, to strike a proper balance, *cost-effective online edge data distribution (OEDD) must minimize the total transmission cost while stabilizing low average transmission latency in the long term*.

This paper proposes LAO, a **L**atency-**A**ware **O**nline approach, for app vendors to formulate cost-effective OEDD strategies. As the first attempt at the OEDD problem, its key contributions include:

- We formulate the OEDD problem and prove its $\mathcal{NP}$-hardness.
- We propose an innovative online approach named LAO for solving the OEDD problem based on the Lyapunov optimization. Unlike conventional Lyapunov approaches that model target systems as queuing systems and pursue to minimize the queue(s), LAO aims to stabilize data transmission latency over time.
- To address the issue of performance bottleneck raised by centralized Lyapunov optimization, LAO tackles the OEDD problem in a decentralized manner by combining game theory and Lyapunov optimization innovatively.
- We theoretically analyze the performance of LAO and evaluate LAO with extensive experiments conducted on a real-world dataset.

The rest of this paper is organized as follows. We present system models, formulate the OEDD problem and prove its $\mathcal{NP}$-completeness in Section 2. Section 3 presents LAO in detail and analyzes its performance theoretically. Section 4 evaluates LAO experimentally. We review the related work and conclude this paper in Section 5 and Section 6.

2 System Model

We introduce the edge cache system architecture first in this section. Then we formally define the transmission cost. After that, we formulate the OEDD problem and prove its $\mathcal{NP}$-hardness. The notations are summarized in Appendix A.

2.1 System Architecture

The edge server network in an edge cache system can be modeled as a graph $G(V, E)$, where V denotes the set of edge servers and E denotes the set of links. The link between edge servers and cloud servers is also modeled as an edge. With one or multiple cloud servers, modeled as node C, we use $G(V, C, E)$ to represent the entire edge cache system.

A piece of data d can be transmitted to a destination edge server from either the cloud or another edge server in the system. Denote V_S^t as the set of *source servers* in time slot t, including the cloud C and the edge servers that have d in cache, e.g., v_5 in Fig. 1(b).

Let V_D^t denote destination edge servers in time slot t. Let vector $\omega^t =< \omega_0^t, ..., \omega_n^t >$, where $\omega_0^t = \omega_C^t$ and $\omega_v^t \in \{0, 1\}$ $(0 \leq v \leq n)$, indicate whether node v is a *transmission node* in time slot t, i.e., whether v is involved in the transmission of d as the source node, the intermediate node or the destination node. Similarly, let vector $\gamma^t =< \gamma_{0,0}^t, \gamma_{0,1}^t, ..., \gamma_{n,n}^t >$ $(\gamma_{u,v}^t \in \{0, 1\})$ denote whether transmitting d via edge $e_{u,v}$ in t. Notice that $\gamma_{u,v} \equiv 0$ if $u = v$ or $e_{u,v} \notin E$. For each node $v \in V \setminus V_S^t$, d is transmitted through v if d is transmitted via an edge connecting another edge server and v:

$$\sum_{u \in V} \gamma_{u,v}^t \geq \omega_v^t, \forall v \in V \setminus V_S^t \tag{1}$$

If d is transmitted via $e_{u,v}$ in time slot t, it will be transmitted through u and v:

$$\gamma_{u,v}^t = \omega_v^t \cdot \omega_v^t, \forall u, v \in V \tag{2}$$

2.2 Transmission Cost

The prices for C2E and E2E transmissions are region-specific and vary in edge cache systems built by different edge infrastructure providers. In this paper, we measure transmission costs in a generic manner, similar to [17]. In this way, actual C2E and E2E transmission costs can be calculated with specific pricing models, e.g., Amazon or Google's. Let $cost_{u,v}$ represent the minimum transmission cost from u to v and $cost_{C,v}$ as that from the cloud C to v. Denoted by $cost(\gamma^t)$, the transmission cost incurred by the EDD strategy γ^t in time slot t is:

$$cost(\gamma^t) = \sum_{v \in V} \gamma_{C,v}^t \cdot cost_{C,v} + \sum_{u \in V} \sum_{v \in V} \gamma_{u,v}^t \cdot cost_{u,v} \tag{3}$$

Please note that cache spaces on edge servers are expensive because of limited sizes [19]. App vendors need to compete for the storage spaces in the edge cache system. A common practice is to reserve cache spaces in advance rather than on-demand [12]. Thus, the expenses of storing data in the edge cache system are fixed and not included in (3).

2.3 Transmission Latency

Let $\bar{l}$ denote the time-averaged maximum transmission latency expected by the app vendor, and $l_v^{\gamma^t}$ denote the time taken for transmitting the data to edge server v according to EDD strategy γ^t. As discussed in Section 1, cost-effective OEDD must stabilize low transmission latency in the long term. This is achieved by stabilizing the average time taken to transmit data to each destination server below $\bar{l}$ over T time slots:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\sum_{v \in V_D^t} l_v^{\gamma^t}}{|V_D^t|} \leq \bar{l} \tag{4}$$

In particular, data d does not need to be transmitted to any of the source servers because d is already in their cache:

$$l_v^{\gamma^t} = 0, \forall v \in V_S^t \tag{5}$$

If d goes through edge $e_{u,v}$ in G according to γ^t, the times taken to transmit d to u and v fulfil:

$$l_u^{\gamma^t} - l_v^{\gamma^t} = l_{u,v}^t, \forall u, v \in V, \text{if } \gamma_{u,v}^t = 1 \tag{6}$$

2.4 Problem Formulation and Hardness

As discussed in Section 1, the app vendor wants to minimize the total transmission cost over time. Thus, in a time slot t, d can go through any nodes in G only once. Accordingly, (1) can be refined as follows:

$$\sum_{u \in V} \gamma_{u,v}^t = \omega_v^t, \forall v \in V \setminus V_S^t \tag{7}$$

For each destination edge server $v \in V_D^t$, it must be included in γ^t, i.e., the EDD strategy in t:

$$\omega_v^t = 1, \forall v \in V_D^t \tag{8}$$

In the dynamic edge cache system, edge servers may not always be available, due to cyber attacks, software exceptions, hardware faults, etc. These edge servers will block the data transmission unavailable and must be considered in the OEDD problem. Let V_F^t denote unavailable edge servers in t. Since data d cannot go through an unavailable node, we can obtain:

$$\sum_{u \in V} \gamma_{u,v}^t = \sum_{u \in V} \gamma_{v,u}^t = 0, \forall v \in V_F^t \tag{9}$$

In the edge cache system, users' data requests arrive and leave randomly. Formulating edge data distribution strategies in time slots individually may not ensure low total transmission cost and time-averaged latency over time. Thus, these must be considered and managed in the long term. To achieve the minimum transmission cost over time, the OEDD problem is formulated as:

$$\begin{aligned} \mathcal{P}_1: \quad & \min \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} cost(\gamma^t) \\ s.t. \quad & \gamma_{u,v}^t, \omega_v^t \in \{0, 1\}, \forall u, v \in V \\ & (2), (4), (5), (6), (7), (8), (9) \end{aligned}$$

Remark. Please note that the storage cost is not considered in the OEDD problem. This is because the app vendor has

already determined which edge servers need the data, i.e., the destination edge servers where data storage cost incurs.

Now, we introduce the t-OEDD problem, i.e., the sub-problem of $\mathcal{P}_1$ in an individual time slot t without (4).

Theorem 1. The t-OEDD problem is $\mathcal{NP}$-hard.

The proof of this theorem is provided in Appendix B.

Based on Theorem 1, the OEDD problem is $\mathcal{NP}$-hard, because the t-OEDD problem is a special case of the OEDD problem where $T = 1$.

3 Online Algorithm Design And Analysis

Lyapunov optimization theory has been widely applied in many domains by modeling dynamic systems as queuing systems to stabilize queues like task queues and request queues. In this section, we leverage Lyapunov optimization theory innovatively to formulate an approximation problem for stabilizing transmission latency over time. In addition, game theory, as a powerful tool, is useful for solving edge computing problems in a decentralized manner [20]. Thus, we combine Lyapunov optimization and game theory here to design a **L**atency-**A**ware **O**nline (LAO) algorithm to solve the approximation problem from the OEDD problem effectively and efficiently.

3.1 Latency-Aware Model Translation

In $\mathcal{P}_1$, constraint (4) aims to stabilize the time-averaged maximum transmission latency. However, it requires complete information about destination edge servers in all time slots, which cannot be obtained in advance in real-world edge cache systems. To tackle this challenge, we introduce the *accumulated latency* in Definition 1.

Definition 1. Denoted by σ_{t+1}, the accumulated latency is the overdue delay accumulated in an edge cache system over t time slots, calculated by:

$$\sigma_{t+1} = \lfloor \sigma_t + \sum_{v \in V_D^t} (l_v^{\gamma^t} - \bar{l}) \rfloor_+ \tag{10}$$

where $\sigma_0 = 0$, i.e., the initial value of the accumulated latency. The accumulated latency increases when the average transmission latency exceeds $\bar{l}$, i.e., the expected time-averaged maximum transmission latency. According to Definition 1, the long-term transmission latency constraint (4) can be converted to:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\sigma_t] \leq 0 \tag{11}$$

Based on (10), the Lyapunov function $\mathcal{L}(\sigma(t)) \triangleq \frac{1}{2}\sigma^2(t)$ is defined to measure the accumulated latency $\sigma(t)$. Next, we define $\Delta(\sigma(t))$ below to introduce constraint (11) as part of the optimization objective to strike the balance between time-averaged transmission latency and total transmission cost in the long term:

$$\begin{aligned} \Delta(\sigma_t) &= \mathbb{E}[\mathcal{L}(\sigma_{t+1}) - \mathcal{L}(\sigma_t)|\sigma_t] = \frac{1}{2}\mathbb{E}[\sigma_{t+1}^2 - \sigma_t^2|\sigma_t] \\ &= \mathbb{E}[\sigma_t \sum_{v \in V_D^t} (l_v^{\gamma^t} - \bar{l})|\sigma_t] + \frac{1}{2}\mathbb{E}[(\sum_{v \in V_D^t} (l_v^{\gamma^t} - \bar{l}))^2|\sigma_t] \end{aligned} \tag{12}$$

Next, we combine $\Delta(\sigma_t)$ with the optimization objective of $\mathcal{P}_1$:

$$\zeta \cdot \mathbb{E}[cost(\gamma^t)|\sigma_t] + \Delta(\sigma_t) \tag{13}$$

where ζ is the app-specific parameter representing the urgency to maintain the time-averaged transmission latency by lowering current transmission latency when (11) was violated in the previous time slot. In general, a lower ζ will accelerate the stabilization at a potentially higher transmission cost. Its impact on LAO will be experimentally evaluated in Section 4.

According to (12), the calculation of $\Delta(\sigma_t)$ in time slot t requires $\mathcal{L}(\sigma_{t+1})$, which is not available in time slot t. To address this issue, we now try to find the upper bound on (13) that can be calculated based on information available in time slot t. Because $l_v^{\gamma^t}$ is not lower than 0, $\sum_{v \in V_D^t}(l_v^{\gamma^t} - \bar{l})^2$ is not higher than $\sum_{v \in V_D^t} \bar{l}^2$. Let us define a constant $\Theta = \frac{1}{2}\sum_{v \in V_D^t} \bar{l}^2$. Based on (12), the upper bound on $\Delta(\sigma_t)$ fulfils:

$$\Delta(\sigma_t) \leq \sigma_t \cdot \mathbb{E}[\sum_{v \in V_D^t} (l_v^{\gamma^t} - \bar{l})|\sigma_t] + \Theta \tag{14}$$

Based on (14), the upper bound on (13) can be found:

$$\begin{aligned} &\zeta \cdot \mathbb{E}[cost(\gamma^t)|\sigma_t] + \Delta(\sigma_t) \leq \\ &\zeta \cdot \mathbb{E}[cost(\gamma^t)|\sigma_t] - \sigma_t \cdot \mathbb{E}[\sum_{v \in V_D^t} (\bar{l} - l_v^{\gamma^t})|\sigma_t] + \Theta \end{aligned} \tag{15}$$

It can be calculated based on the information available in current time slot. Now, $\mathcal{P}_1$ is approximated by finding the solution to $\mathcal{P}_2$ below that minimizes the upper bound on (13) according to (15) in each time slot over T:

$$\mathcal{P}_2: \quad \min \quad \zeta \cdot cost(\gamma^t) - \sigma_t \cdot \sum_{v \in V_D^t} (\bar{l} - l_v^{\gamma^t}) + \Theta \tag{16}$$

$$s.t.: (2), (5), (6), (8), (7)$$

3.2 Latency-Aware Online Algorithm Design

Finding the optimal solution to $\mathcal{P}_2$ in each time slot is computationally intensive in large-scale OEDD scenarios. To find a solution quickly, we formulate an EDD game in each time slot to solve $\mathcal{P}_2$ effectively and efficiently. In the EDD game, each participating node is simulated as a player to submit its EDD decision consisting of a set of edges, i.e., a path, based on a specific benefit function for achieving objective (16) of $\mathcal{P}_2$ in each time slot. Nodes will exchange information about their own data transmission latency incurred by in implementing the EDD strategy in each time slot and update their benefit functions based on the accumulated latency. Then, they make decisions based on the updated benefit functions. Finally, they formulate the final solution by reaching a Nash equilibrium in a decentralized manner.

Let γ_v^t denote v's EDD decision in t and γ_{-v}^t denote EDD decisions included in the EDD strategy except v's in t . Given γ_{-v}^t, EDD decision γ_v^t is made for node v to maximize its benefit in terms of data distribution cost and latency, calculated by benefit function (17):

$$\eta(\gamma_v^t, \gamma_{-v}^t) = \max\{\sigma_t, 1\} \cdot (\bar{l} - l_v^{\gamma^t}) - \zeta \cdot \Delta cost(\gamma_v^t, \gamma_{-v}^t) \tag{17}$$

where $\Delta cost(\gamma_v^t, \gamma_{-v}^t)$ denotes the cost incurred by transmitting data to v based on γ_v^t and γ_{-v}^t. Please note that when $\sigma_t = 0$, pursuing the minimum cost without considering latency is likely to incur high latency in time slot t and cause more costs in subsequent time slots to converge σ_t. To avoid such situations, we use $\max\{\sigma_t, 1\}$ in this benefit function to include latency-awareness in the EDD game. The advantage of this setting is validated and analyzed experimentally in Section 4.2.1.

According to (17), $\mathcal{P}_2$ can be formulated as the EDD game $Game_{LAO} = \{V, C, E, \{\mathcal{Y}_v^t\}_{v \in V_D^t}, \{\eta(\gamma_v^t, \gamma_{-v}^t)\}_{\gamma_v^t \in \mathcal{Y}_v^t}\}$, where $\mathcal{Y}_v^t$ is v's finite set of possible EDD decisions. In the EDD game, admitting one Nash equilibrium is significant due to Property 1 [21]:

Property 1. The EDD decision γ_v^{*t} is v's best choice in $\mathcal{Y}_v^t$ based on γ_{-v}^t, if the strategy γ^{*t} is a Nash equilibrium.

Based on Property 1, a Nash equilibrium in the EDD game can be employed as a self-enforcing solution. Here, we introduce the potential game [22]:

Definition 2. A potential function $\phi(\gamma^t) = \phi(\gamma_v^t, \gamma_{-v}^t)$ exists in a potential game, fulfilling:

$$\eta(\gamma_v^t, \gamma_{-v}^t) < \eta(\gamma_v'^t, \gamma_{-v}^t) \Rightarrow \quad \phi(\gamma_v^t, \gamma_{-v}^t) < \phi(\gamma_v'^t, \gamma_{-v}^t) \tag{18}$$

for any $v \in V_D^t$, $\gamma_v^t, \gamma_v'^t \in \mathcal{Y}_v^t$ and $\gamma_{-v}^t \in \prod_{u \neq v} \mathcal{Y}_u^t$.

The Nash equilibrium in an EDD game can be interpreted in different ways. An EDD strategy γ^{*t} is a Nash equilibrium if $\phi(\gamma_v^{*t}, \gamma_{-v}^{*t}) = \max_{\gamma_v^t \in \mathcal{Y}_v^t} \phi(\gamma_v^t, \gamma_{-v}^{*t}), \forall v \in V_D^t$. Now, we show that the EDD game is a potential game:

Theorem 2. With the potential function $\phi(\gamma_v^t, \gamma_{-v}^t)$ below, the EDD game is a potential game.

$$\begin{aligned}\phi(\gamma_v^t, \gamma_{-v}^t) = &- \sum_{v \in V_D^t} (\max\{\sigma_t, 1\} l_v^{\gamma^t} + \zeta \cdot \Delta cost(\gamma_v^t, \gamma_{-v}^t)) \\ &\cdot \sum_{u \in V_D^t \bigcap \neg\{u \in V_v^t | l_u^{\gamma'^t} > l_u^{\gamma^t}\} \bigcap \neg\{v\}} (\max\{\sigma_t, 1\} l_u^{\gamma^t} \\ &+ \zeta \cdot \Delta cost(\gamma_u^t, \gamma_{-u}^t))\end{aligned} \tag{19}$$

where V_v^t is the set of destination nodes connecting to v in time slot t.

The proof of this theorem is provided in Appendix C.

After a finite number of iterations, the Nash equilibrium of a potential game can be found [22]. Here, we propose the **L**atency-**A**ware **O**nline (LAO) approach in Algorithm 1. Its pseudo code is shown in Algorithm 1. LAO begins with initializing EDD strategies and the accumulated latency (Lines 3). In each time slot, it observes the status of all nodes to find the source nodes V_S^t, destination nodes V_D^t and unavailable nodes V_F^t (Line 5). The algorithm creates a new set of destination nodes $V_D'^t$ by removing nodes with the cached data from V_D^t (Line 6). Then, it creates a set of source candidates $V_{Can}^t = C \bigcup V_S^t$, for distributing data from those candidates to destination nodes V_D^t, and set the benefit of each node in $V_D'^t$ to $-\infty$ (Lines 7-8).

Next, for each node $v \in V_D'^t$, it calculates the benefit produced by its current EDD decision γ_v^t (Line 11). After that, we use the weighted Dijkstra algorithm to find the EDD decision with the maximum value of $\eta(\gamma_v'^t, \gamma_{-v}^t)$ (Line 12). When the EDD decision $\eta(\gamma_v'^t, \gamma_{-v}^t)$ produces a higher benefit than v's current EDD decision $\eta(\gamma_v^t, \gamma_{-v}^t)$, it will be submitted to contend for update (Line 14). For the winner, all destination nodes added by γ_v^t are removed from V_{Can}^t and those visited by $\gamma_v'^t$ are included in V_{Can}^t (Lines 16-17). In each iteration, one submitted EDD decision is selected randomly for implementation. The calculation in Lines 4-22 is performed in parallel for individual users. It iterates until no update request. At the end of each time slot, the accumulated latency σ_t is updated, according to (10) (Line 25). Finally, the EDD strategy γ is returned as the OEDD solution (Line 27).

Algorithm 1 **L**atency-**A**ware **O**nline (LAO) Approach

1: **Input:** $G(V, C, E), \zeta, \bar{l}$, T
2: **Output:** data distribution strategy $\gamma = \{\gamma^0, ..., \gamma^{T-1}\}$
3: $\sigma_0 = 0$, $t = 0$, $\gamma \leftarrow \varnothing$
4: **repeat**
5: observe the source nodes V_S^t, destination nodes V_D^t and unavailable nodes V_F^t
6: create a new set $V_D'^t = \{v | v \in V_D^t \bigcap \neg V_S^t\}$ and set $l_v^{\gamma^t} = 0, \forall v \in V_D^t \bigcap V_S^t$
7: create $V_{Can}^t \leftarrow C \bigcup V_S^t$ /*V_{Can}^t: the set of candidates*/
8: set $\eta(\gamma_v^t, \gamma_{-v}^t) = -\infty$ for each node $v \in V_D'^t$
9: **repeat**
10: **for** each node $v \in V_D'^t$ **do**
11: calculate v's benefit $\eta(\gamma_v^t, \gamma_{-v}^t)$
12: obtain the set of EDD decision $\gamma_v'^t$ that includes the set of edges from v to V_{Can}^t with the maximum value of $\eta(\gamma_v'^t, \gamma_{-v}^t)$ by weighted dijkstra algorithm under (9)
13: **if** $\eta(\gamma_v'^t, \gamma_{-v}^t) > \eta(\gamma_v^t, \gamma_{-v}^t)$ **then**
14: submit $\gamma_v'^t$ to the contend and wait
15: **if** v is selected to update $\gamma_v'^t$ **then**
16: remove all destination nodes added into V_{Can}^t by γ_v^t and set their benefits to 0
17: add all destination nodes visited by $\gamma_v'^t$ into V_{Can}^t and update their benefits according to $\gamma_v'^t$
18: $\gamma_v^t \leftarrow \gamma_v'^t$
19: **end if**
20: **end if**
21: **end for**
22: **until** no EDD decisions are submitted
23: $\gamma = \gamma \bigcup \gamma^t$
24: $t = t + 1$
25: calculate $l_v^{\gamma^t}$ for each edge server $v \in V_D^t$, and update accumulated latency σ_t based on (10)
26: **until** $t = T$
27: **return** γ

Algorithm 1 can find the Nash equilibrium of an EDD game within finite iterations. Let l_{min} and $cost_{min}$ denote the minimum latency and minimum cost of any edge in G, and I denote the upper bound of the iteration number. In the LAO game, the iteration number is no more than $\frac{|V|^2 \Delta\phi}{\Delta\eta}$, where $\Delta\phi$ is the difference between the maximum and minimum values of the potential function, calculated with $\max\{\phi(\gamma_v^t, \gamma_{-v}^t) | v \in V, t \in T\} - \min\{\phi(\gamma_v^t, \gamma_{-v}^t) | v \in V, t \in T\}$, and $\Delta\eta$ is the mini-

mum increase in the value of the potential function, calculated with $\min\{\min\{l^t_{u,v}|e_{u,v} \in E \bigcup \{e_{C,v}|v \in V\}, l^t_{u,v} \neq l_{min}, t \in T\} - l_{min}, \zeta \cdot (\min\{cost_{u,v}|e_{u,v} \in E \bigcup \{e_{C,v}|v \in V\}, cost_{u,v} \neq cost_{min}\} - cost_{min})\}$.

Since the weighted Dijkstra algorithm takes $O(|V| \cdot log|V| \cdot |E| \cdot log|V|)$ computation operations, the computation complexity of the iteration process is also $O(|V| \cdot log|V| \cdot |E|)$, according to Algorithm 1. Thus, the computation complexity of LAO in each time slot is $O(I \cdot |V| \cdot log|V| \cdot |E|)$.

Considering the space complexity, LAO creates several arrays to store the benefit, the EDD decision cost and the EDD decision latency for each node. Since the EDD decision contains at most $|E|$ edges between the edge servers and $|V|$ edges between the cloud and edge servers, the space complexity of LAO is $O(|V| \cdot \max\{|E|, |V|\})$.

Remark. In the EC environment, app vendors hire resources on edge servers for hosting their data [20], [23]. They are not responsible for detecting or fixing node failures at the edge, which is the edge infrastructure provider's responsibility, e.g., Amazon and Google. App vendors formulate their edge data distribution strategies based on the current network status provided by the edge infrastructure provider. When a node fails, the edge infrastructure provider will notify the corresponding app vendors so that they can adapt their edge data distribution strategies.

3.3 Theoretical Performance Analysis

In this section, we theoretically analyze LAO's effectiveness in minimizing the total transmission cost and stabilizing the time-averaged transmission latency with Theorem 3 and Theorem 4, respectively.

Theorem 3. LAO's total transmission cost over T time slots is bounded by $O(\frac{1}{\zeta})$ compared with the optimal solution.

The proof of this theorem is provided in Appendix D.

Theorem 4. The time-averaged accumulated latency achieved by LAO is bounded by $O(\zeta)$.

The proof of this theorem is provided in Appendix E.

Based on the analysis above, when $\zeta \to \infty$, the solution found by LAO is the near-optimal one to $\mathcal{P}_1$. However, with a higher value of ζ, the accumulated latency will increase and more time slots are required to stabilize this time-averaged transmission latency (11). The performance analysis presented in this section is also validated and analyzed experimentally in Section 4.

4 Experimental Evaluation

We experimentally evaluated the performance of LAO in different OEDD scenarios by varying different parameters. We conducted all the experiments on a Windows-10 machine and a real-world dataset.

4.1 Experimental Settings

4.1.1 Benchmark approaches

Three representative approaches are implemented to be compared against LAO:

- *OEDD-IP*: This approach finds the optimal solution to $\mathcal{P}_2$ in Section 3.1 with IBM's CPLEX Optimizer.
- *Latency-Oriented data distribution (LO)*: This approach always finds the optimal solution with the minimum data distribution latency in each time slot. It is originated from the optimal solution presented in [24] with IBM's CPLEX Optimizer.
- *Enhanced-EDD-IP (EEI)*: This approach is the enhanced version of EDD-IP in [17]. Besides the cloud, it may also source data for destination edge servers from other edge servers in the system. If the long-term transmission latency constraint (4) is fulfilled, EEI aims to minimize the data distribution cost while fulfilling constraints (2), (5), (6), (8) and (7), similar to OEDD-IP. Otherwise, EEI finds the solution that minimizes the transmission latency to fulfill constraint (4), similar to LO. EEI also finds the solution with IBM's CPLEX Optimizer.

4.1.2 Performance metrics

Three metrics are implemented in the experiments to evaluate the performance of LAO:

- Time-averaged transmission latency, measured in milliseconds and calculated according to the left side of (4), the lower the better. We also observe whether the approaches fulfill the long-term transmission latency constraint (4).
- Total transmission cost, calculated with (3) over T, the lower the better.
- Maximum overhead over T, consisting of the computation time taken to find a solution and the communication time incurred. It is measured in milliseconds, the lower the better.

4.1.3 Dataset

The experiments are conducted on a real-world EUA dataset, which is widely used to conduct experiments in the field of edge computing [3], [17], [25]. The EUA dataset includes the locations of 1,464 real-world base stations and 174,305 end-users within metropolitan Melbourne, Australia. Specifically, the geographic locations of 125 base stations (edge servers) in the Melbourne CBD area, Australia are used to simulate EC environments. Similar to many studies in the EC environment [3], [17], we measure transmission cost in a generic manner. Specifically, E2E transmission cost is 1 and C2E transmission cost is $ratio$. To simulate real-world OEDD scenarios, we deploy virtual machines in the ARDC Nectar Research Cloud[3] as edge servers and an Amazon EC2 instance as the cloud. Links are built between adjacent edge servers based on their locations until the target density ρ is reached. In this experiment environment, the E2E transmission latency ranges from 10 to 30 milliseconds and the C2E transmission latency from 160 to 200 milliseconds. In each experiment, the time slot number is 100 ($T = 100$). In each experiment, $|V|$ edge servers are randomly selected to simulate an edge cache system. In each time slot t, data d is randomly decached from each individual edge server according to probability p_{dec}. The edge servers that still cache d will be the source edge servers in t. In the meantime, a number of the other

3. https://ardc.edu.au/services/nectar-research-cloud/

TABLE 1: Parameter Settings

	$\|V\|$	ρ	$ratio$	$\bar{l}$	p_{dec}	p_{req}	ζ
Set #1	20	1.0	10	80	0.5	0.20	1
Set #2	10,15,20,25,30	1.0	10	80	0.5	0.20	1
Set #3	20	1.0,1.1,1.2,1.3,1.4,1.5	10	80	0.5	0.20	1
Set #4	20	1.0	6,8,10,12,14	80	0.5	0.20	1
Set #5	20	1.0	10	60,70,80,90,100	0.5	0.20	1
Set #6	20	1.0	10	80	0.7,0.6,0.5,0.4,0.3	0.20	1
Set #7	20	1.0	10	80	0.5	0.10,0.15,0.20,0.25,0.30	1
Set #8	20	1.0	10	80	0.5	0.10	10,1,0.1

edge servers randomly request d according to probability p_{req} and become destination edge servers in t. As discussed in Section 2.4, edge servers may not always be available. In the experiments, we assume that the unavailable rate of an edge server is 0.05. Then, the four approaches are performed to formulate corresponding EDD strategies in time slot t.

4.1.4 Experiment Parameters

To comprehensively analyze the performance of LAO, we conduct eight sets of experiments, i.e., Set #1 to Set #8, to simulate various real-world edge cache systems and OEDD scenarios. In our experiments, we vary the values of the seven parameters below to observe their impacts on LAO in different OEDD scenarios. Except in Set #1, we vary the value of one setting parameter while fixing the other six parameters in each set of the experiments, as summarized in Table 1. When the value of one setting parameter varies, we repeat the experiments for 30 times and obtain the averaged results.

- Number of edge servers ($|V|$) - Set #2. This parameter decides the size of the edge cache system.
- Edge density (ρ) - Set #3. This parameter represents the density of graph $G(V, E)$ that represents the edge server network in the system. It is calculated by $\rho = |E|/|V|$.
- Ratio of C2E transmission cost over E2E transmission cost ($ratio$) - Set #4. This parameter simulates different pricing models from edge infrastructure providers for data transmissions.
- Time-averaged maximum transmission latency ($\bar{l}$) - Set #5. Used in Eq. (10), this parameter is the app vendor's expected time-averaged maximum transmission latency. A low $\bar{l}$ indicates app vendor's high priority for pursuing low latency.
- Data decache probability (p_{dec}) - Set #6. This parameter determines the probability for an edge server to remove d from its cache. It simulates the scarcity of app vendor's cache spaces in the system and impacts the number of source edge servers in each time slot.
- Data request probability (p_{req}) - Set #7. This parameter controls the probability that an edge server becomes a destination edge server in each time slot. A high p_{req} represents high popularity of d.
- Trade-off parameter (ζ) - Set #8. As discussed in Section 3.3, this parameter is used in (16) to represent the urgency to stabilize the time-averaged transmission latency when (11) was violated.

4.2 Experimental Evaluation

4.2.1 Performance Comparison

The results of Set #1 is shown in Fig. 2. Fig. 2(a) depicts that LAO quickly stabilizes the transmission latency and keeps it below $\bar{l}$, fulfilling the long-term transmission latency constraint (4). We can also see that LAO does this at the lowest of the total transmission cost among all the four approaches. OEDD-IP, LO and EEI can also fulfil (4) over time. Fig. 2(b) demonstrates that LAO significantly outperforms EEI and LO in minimizing transmission cost, by an average of 37.79% and 50.81% over T, respectively, across 100 time slots. Interestingly, the advantage of LAO is 1.89% over OEDD-IP. This is because that the time-averaged latency incurred by LAO is higher than that incurred by OEDD-IP. Another reason is that the benefit function (17) employed by LAO always considers the latency even when $\sigma_t = 0$. This may incur costs in time slot t but can reduce the cost of converging σ_t in subsequent time slots, as discussed in Section 3.2. LAO's superior performance demonstrated in Fig. 2 validates the importance of tackling the EDD problem in an online manner.

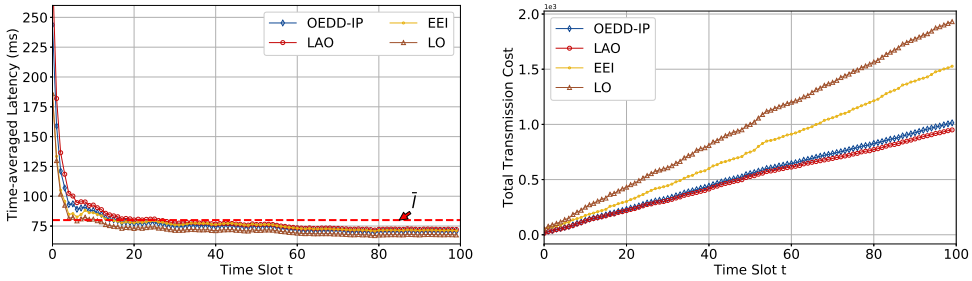
The results of the time-averaged latency achieved by all the four approaches in Sets #2 - #7 are similar to Fig. 2(a). Thus, we summarize the average results in Table 2. It shows that LAO is second only to LO in all the experiments. Next, we focus on the total transmission costs incurred by the approaches in Sections 4.2.2 - 4.2.7.

TABLE 2: Time-averaged Latency

	Set #1	**Set #2**	**Set #3**	**Set #4**	**Set #5**	**Set #6**	**Set #7**
OEDD-IP	70.48	75.48	69.06	74.89	76.30	79.51	78.61
LAO	72.10	76.89	70.61	75.55	77.45	79.96	79.33
EEI	71.11	76.90	68.13	83.74	79.00	80.09	84.87
LO	67.56	73.18	63.16	74.41	73.76	77.56	78.05

4.2.2 Impact of number of edge servers ($|V|$)

Fig. 3 compares the transmission costs incurred by the four approaches when the size of the edge cache system $|V|$ varies from 10 to 30. It is clear that LAO incurs the lowest total transmission costs over time when $|V|$ increases from 15 to 30 and LAO, and the cost incurred by LAO is slightly higher than that incurred by OEDD-IP when $|V|$ is 10. On average, the advantages of LAO are 2.98% over OEDD-IP, 32.17% over EEI and 48.56% over LO. When $|V|$ increases



(a) Latency over T (b) Cost over T

Fig. 2: Set #1

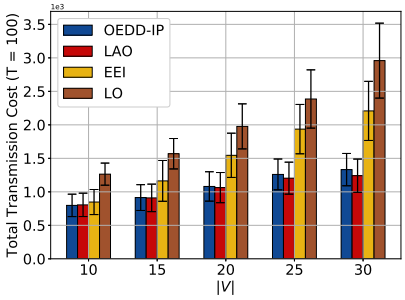


Fig. 3: Cost vs. $|V|$ (Set #2)

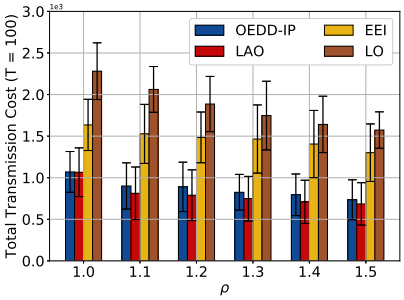


Fig. 4: Cost vs. ρ (Set #3)

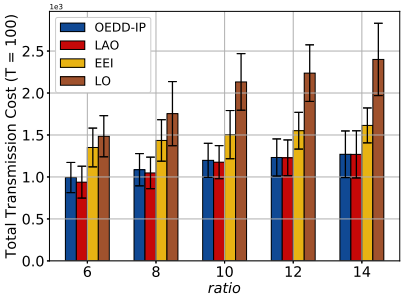


Fig. 5: Cost vs. $ratio$ (Set #4)

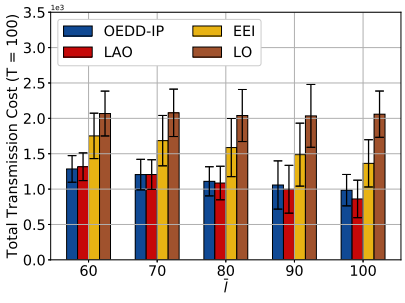


Fig. 6: Cost vs. $\bar{l}$ (Set #5)

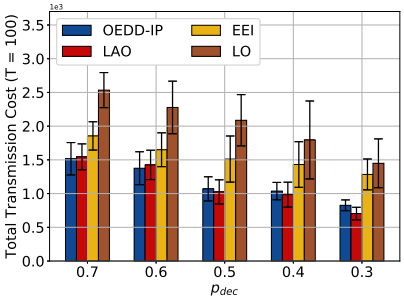


Fig. 7: Cost vs. p_{dec} (Set #6)

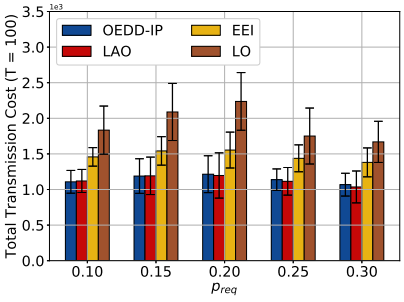


Fig. 8: Cost vs. p_{req} (Set #7)

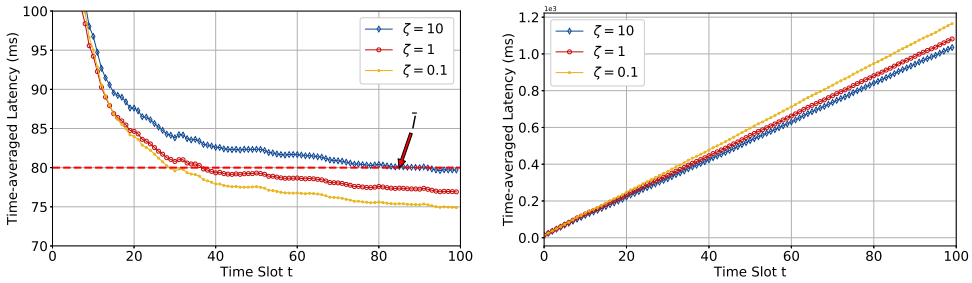
from 10 to 30, more edge servers will request data d on average in each time slot and it takes more time to transmit d to all the destination edge servers in general. LAO can source d for destination edge servers close to them and thus does not incur much more transmission costs when $|V|$ increases. Specifically, when $|V|$ increases from 10 to 30 by 300%, LAO's total transmission cost increases from 806 to 1,214 by only 54.07%, much lower than OEDD-IP's 66.93% increase from 798 to 1,332, EEI's 160.38% increase from 848 to 2,208 and LO's 134.10% from 1,264 to 2,959.

4.2.3 *Impact of edge density (ρ)*

Fig. 4 presents the impacts of ρ on the total transmission costs. Overall, LAO achieves the lowest total transmission cost. On average, the advantages of LAO are 7.75% over OEDD-IP, 45.43% over EEI and 57.00% over LO. When ρ increases from 1.0 to 1.5, the total transmission costs of all the four approaches decrease, by 35.74% from 1,066 to 685 for LAO, by 31.31% from 1,070 to 735 for OEDD-IP, by 20.43% from 1,635 to 1,301 for EEI and by 31.04% from 2,281 to 1,573 for LO. The reason is that, with an increase in ρ, each edge server is connected to more others averagely. In this case, the average distance between edge servers is shortened, and data can be sourced to destination edge servers with lower transmission costs.

4.2.4 *Impact of ratio of C2E transmission cost over E2E transmission cost ($ratio$)*

Fig. 5 compares the total transmission costs when $ratio$ increases from 6 to 14 in Set #4. Overall, LAO outperforms OEDD-IP by 1.92%, while outperforming EEI and LO with large margins, i.e., 23.97% against EEI and 43.37% against LO. The increase in $ratio$ increase the cost of C2E transmissions over E2E transmissions and consequently increases the total transmission costs incurred all the four approaches. Fig. 5 also shows that given different $ratio$ values, LAO can always formulate the most cost-effective solutions by finding the best combinations of C2E and E2E transmissions.



(a) Time-average Transmission Latency vs. ζ over T (b) Total Transmission Cost vs. ζ over T

Fig. 9: Latency and Cost achieved by LAO vs. ζ (Set #8)

4.2.5 Impact of time-averaged maximum transmission latency $\bar{l}$

Fig. 6 illustrates the impact of $\bar{l}$ on the total transmission costs in Set #5. Again LAO is the clear winner, with its 34.57% transmission cost decrease from 1,316 to 861 when $\bar{l}$ increases from 60 milliseconds to 100 milliseconds. OEDD-IP and EEI's costs also decrease, from 1,285 to 985 by 23.35% and from 1,752 t0 1,364 by 22.15%, respectively. The increase in $\bar{l}$ relaxes the constraint for low data retrieval latency. LAO, OEDD-IP and EEI can source data for destination edge servers from edge servers further away to lower their total transmission costs. LO does not consider $\bar{l}$ and thus are not impacted by the value of $\bar{l}$.

4.2.6 Impact of data decache probability (p_{dec})

Fig. 7 shows the impact of p_{dec} on the four approaches in Set #6. Overall, LAO outperforms OEDD-IP, EEI and LO, by 2.55%, 26.51% and 43.95% respectively. As p_{dec} increases, it is more likely for LAO, OEDD-IP, EEI and LO to source data from within the system rather than the cloud. Using this advantage, their costs decrease with the increase in p_{dec} significantly, from 1,545 to 703 by 54.50% for LAO, from 1,516 to 826 by 45.51% for OEDD-IP, from 1,855 to 1,284 by 30.78% for EEI and from 2,534 to 1,449 by 42.82% for LO.

4.2.7 Impact of data request probability (p_{req})

Fig. 8 demonstrates the impact of p_{req} on the total transmission costs. Overall, the advantages of LAO are again significant, in terms of the total transmission cost, i.e. 1.01% over OEDD-IP, 23.23% over EEI and 40.92% over LO. With a higher $request_q$, there are more destination edge servers in each time slot. Thus, total transmission costs incurred by all approaches increase when $request_q$ increases from 0.10 to 0.20. However, when $request_q$ increases from 0.20 to 0.30, the total transmission costs incurred by LAO, OEDD-IP, EEI and LO decrease. This is because more edge servers become source edge servers due to a fixed data decache probability at 0.5. In this way, there is a higher chance for destination edge servers to obtain data from within the system and the costs incurred by LAO, OEDD-IP, EEI and LO decrease.

4.2.8 Impact of trade-off parameter (ζ)

Fig. 9 shows the impact of ζ (introduced by (13) into $\mathcal{P}_2$) on LAO. As discussed in Section 3.2, a lower ζ will accelerate the stabilization of the time-averaged transmission latency at a potentially higher transmission cost. It is validated experimentally by the results presented in Fig. 9(b) and Fig. 9(a). As shown in Fig. 9(a), at the very beginning of the experiments, the average latency is high because data d has to be sourced from the cloud. When $\zeta = 10$, LAO takes 92 time slots to decrease the average latency to below $\bar{l}$ before it is stabilized. When $\zeta = 1$ and 0.1, LAO takes only 38 and 30 time slots, respectively, to do the same. Fast stabilization comes the price of high cost, as demonstrated in Fig. 9(b). When $\zeta = 0.1$, LAO incurs the highest cost of 11,647, 7.71% and 12.57% higher than when $\zeta = 1$ and $\zeta = 10$, respectively.

4.2.9 Efficiency

TABLE 3: Maximum overhead over T. The values in the parentheses are the communication overhead incurred.

	OEDD-IP	LAO	EEI	LO
Set #1	12,004 (84)	426 (420)	12,351 (90)	19,174 (94)
Set #2	192,137 (102)	642 (629)	267,228 (101)	297,333 (104)
Set #3	277,065 (80)	475 (465)	674,619 (82)	938,134 (86)
Set #4	8,026 (81)	424 (408)	10,392 (86)	33,052 (83)
Set #5	6,996 (80)	453 (437)	25,726 (84)	38,709 (81)
Set #6	8,123 (95)	550 (546)	35,719 (94)	36,034 (95)
Set #7	30,760 (98)	618 (600)	38,043 (106)	41,190 (101)

Table 3 presents the results of Set #1 - #7. It demonstrates that *LAO takes multiple-order-of-magnitude less computation overhead than others*. The maximum computation overheads taken by OEDD-IP, EEI and LO are 277,065 milliseconds, 674,612 milliseconds and 938,134 milliseconds, respectively, in Set #3, while the maximum computation overhead taken by LAO is 642 milliseconds. This shows that *LAO can solve very large-scale OEDD problems efficiently*.

LAO incurs the most communication overhead, i.e., 629 milliseconds, while OEDD-IP, EEI and LO take 80-106 milliseconds to find a solution. The reason is that LAO is the only decentralized approach, which requires frequent decision exchanges between edge servers to achieve a Nash equilibrium. Compared with the maximum overheads incurred by other three approaches, such minor differences in the communication overhead are negligible.

4.3 Threats to Validity

4.3.1 Threats to Internal Validity

Whether the experiment setting favors LAO over other approaches is the main threat to the internal validity. Thus, various OEDD scenarios were simulated by changing seven parameters to tackle this threat. This way, we could comprehensively and fairly compare different approaches. Moreover, we repeated the experiments for 30 times when a parameter was changed. Another one is the data popularity in the experiments. Data popularity prediction is a research topic orthogonal to this study. Thus, we employ p_{dec} and p_{req} to generically indicate the results of data popularity prediction. Their values may vary, depending on the data popularity prediction approaches and data storage strategies, and impact the performance of the approaches. The last main threat is the EUA dataset used in the experiments. This is because user and edge server distributions are different across various cities. Different user and edge server distributions can thus lead to different OEDD strategies and impact the performance of the approaches. However, the values of p_{dec} and p_{req}, and the distributions of users and edge servers do not impact the performance differences between these approaches in general.

4.3.2 Threats to External Validity

To the external validity, LAO's generalization and application in different OEDD scenarios are the main threats. To reduce such threat, the unit transmission cost was used to measure the C2E transmission cost and E2E transmission cost, similar to [17], and specific cost models can be easily interpreted, e.g. Amazon or Google's price model. Furthermore, the EUA dataset was adopted to execute the experiments. We also vary the size and the complexity of the OEDD scenarios by changing parameters to improve the comprehensiveness of the evaluation and mitigate the threat to external validity.

5 Related Work

The evolution of broad and mobile communications in the past two decades has continuously lowered the expenses of bandwidth and latency has become the main obstacle to improving service performance [8]. In large-scale content delivery infrastructures like content delivery networks (CDNs), end-users still have to retrieve data from the public internet and it is difficult to predict, control or reduce their data retrieval latency [8]. It is the main barrier to further performance improvement of data caching for latency-sensitive applications like gaming and VR.

Edge computing (EC) enables 5G to break that barrier by caching data on edge servers [4]. The caching system facilitated by edge servers reduces the physical distance between end-users and data to hundreds of meters [6]. In very recent years, edge data caching has attracted great quantity of attention [12]. Similar to studies of conventional cloud cache systems like CDN, existing studies of edge cache systems try to achieve various optimization goals, e.g., maximizing caching revenue [12] and minimizing data retrieve latency [14]. The approaches proposed in these studies leverage the short physical distance between data and end-users in edge cache systems. However, the impacts of the long distance between the cloud and edge servers on edge cache systems have been ignored completely, including the potentially high costs and latency incurred by the data transmission from the cloud.

Game theory is a well-acknowledged mathematical tool for solving optimization problems distributively in edge computing [26]. To name a few, the authors of [27] investigated the resource allocation problem for fulfilling service providers' resource requirements in edge computing with consideration of fairness. Based on game theory, they designed a convex programming approach to achieve a Nash equilibrium. Cui et al. [28] studied edge demand response, aiming to maximize user coverage, minimize energy consumption and maximize users' data rate. They solved the problem in a distributed manner by a game-theoretically approach. However, those approaches can only solve problems in quasi-static scenarios, without consideration of system dynamics over time.

Lyapunov optimization has been proven to be a powerful tool for solving online problems by building a queuing system [29]. Kwak et al. [30] investigated a content caching problem and applied Lyapunov optimization to control caching on the cloud side or edge side, based on the length of the data request queue. In [25], the authors also studied an edge data caching problem. An online approach based on Lyapunov optimization was proposed to minimize total cost while guaranteeing users' low latency. Similarly, Asheralieva et al. [31] studied the edge data caching and sharing problem, considering wireless communication. They minimized both caching and sharing costs while stabilizing request queuing system based on Lyapunov optimization. However, those centralized approaches are standard implementations of Lyapunov optimization and suffer from high computational overheads.

Recently, some researchers have attempted the leverage of game theory and Lyapunov optimization to solve optimization problems [32], [33], [34], [35]. To name a few, Liu et al. [35] investigated the task offloading and resource allocation problem in edge computing. They designed a matching game to allocate user devices to edge servers, considering task queue, edge servers' available resources and communication interference. According to the user allocation strategy obtained by the matching game, they employed a Lyapunov-based approach to minimize the power consumption in task offloading, while stabilizing the queue length. The authors of [34] studied the content delivery problem among devices and unmanned aerial vehicles in a cloud-based content delivery network. They first proposed a game-theoretical approach to minimize the transmission cost, and then proposed a Lyapunov-based approach for the network operator to maximize its revenue and stabilize the cache queue. These studies utilize the advantages of game theory and Lyapunov optimization in different phases separately, unlike LAO that integrates game theory into Lyapunov optimization to combine their advantages.

To help app vendors utilize edge cache systems cost-effectively in the long term, this study tackles the online edge data distribution (OEDD) problem with a new approach named LAO. It allows data to be sourced from within the edge cache system. It innovatively combines

Lyapunov optimization and game theory to solve the OEDD problem in a distributed and online manner. The experimental results presented and discussed in Section 4 demonstrate its superior performance in minimizing data transmission costs and stabilizing data transmission latency over time.

6 Conclusion

In this paper, we studied an online edge data distribution (OEDD) problem from the app vendor's perspective in the Edge Computing (EC) environment. We first formulated the OEDD problem and proved its $\mathcal{NP}$-hardness. Then, we proposed a new Latency-Aware OEDD algorithm, named LAO, that formulates cost-effective EDD strategies online over time. It combines game theory and Lyapunov optimization to tackle the OEDD problem effectively and efficiently in a decentralized manner. Its performance is evaluated both theoretically and experimentally. This research builds a solid foundation for enabling edge cache systems in the real world and opens up a number of new research directions along the line. In our future work, we will investigate the distribution of streaming data in OEDD scenarios and consider fault-tolerant and failure detection during the data distribution process.

Acknowledgement

This research is partially funded by Australian Research Council Discovery Projects No. DP200102491 and Laureate Fellowship FL190100035. We gratefully thank Guangming Cui for his technical support.

References

[1] CISCO, *Cisco Edge-to-Enterprise IoT Analytics for Electric Utilities Solution Overview*, February 1, 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/big-data/solution-overview-c22-740248.html

[2] L. Yuan, Q. He, S. Tan, B. Li, J. Yu, F. Chen, H. Jin, and Y. Yang, "Coopedge: A decentralized blockchain-based platform for cooperative edge computing," in *Proceedings of the Web Conference*, 2021, pp. 2245–2257.

[3] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2019.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, "Pyramid: Enabling hierarchical neural networks with edge computing," in *The Web Conference*, 2022, pp. 1860—-1870. [Online]. Available: http://dx.doi.org/10.1145/3485447.3511990

[6] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1087–1100, 2019.

[7] Y. Wang, B. Veeravalli, and C.-K. Tham, "On data staging algorithms for shared data accesses in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 825–838, 2012.

[8] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 159–170.

[9] X. Xia, F. Chen, G. Cui, M. Abdelrazek, J. Grundy, H. Jin, and Q. He, "Budgeted data caching based on k-median in mobile edge computing," in *27th IEEE International Conference on Web Services*. IEEE, 2020, pp. 197–206.

[10] X. Xia, F. Chen, Q. He, G. Cui, P. Lai, M. Abdelrazek, J. Grundy, and H. Jin, "Graph-based optimal data caching in edge computing," in *International Conference on Service-Oriented Computing*. Springer, 2019, pp. 477–493.

[11] Q. Li, Y. Zhang, A. Pandharipande, Y. Xiao, and X. Ge, "Edge caching in wireless infostation networks: Deployment and cache content placement," in *IEEE Conference on Computer Communications Workshops*. IEEE, 2019, pp. 1–6.

[12] X. Xia, F. Chen, Q. He, G. Cui, J. Grundy, M. Abdelrazek, A. Bouguettaya, and H. Jin, "Ol-medc: An online approach for cost-effective data caching in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TMC.2021.3107918

[13] Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, T. Qiu, B. Hu, and R. Kwok, "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1277–1292, 2021.

[14] T. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2018.

[15] E. Schurman and J. Brutlag, "The user and business impact of server delays, additional bytes, and http chunking in web search," in *Velocity Web Performance and Operations Conference*. oreilly, 2009.

[16] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[17] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 31–44, 2021.

[18] Y. Jiang, M. Ma, M. Bennis, F.-C. Zheng, and X. You, "User preference learning-based edge caching for fog radio access network," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1268–1283, 2018.

[19] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus of blockchains in pervasive edge computing environments," *IEEE Transactions on Mobile Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TMC.2021.3053230

[20] X. Xia, F. Chen, Q. He, G. Cui, J. Grundy, M. Abdelrazek, X. Xu, and H. Jin, "Data, user and power allocations for caching in multi-access edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1144–1155, 2021. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2021.3104241

[21] C. A. Holt and A. E. Roth, "The nash equilibrium: A perspective," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 3999–4002, 2004.

[22] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[23] D. Sabella, V. Sukhomlinov, L. Trang, P. Paglierani, R. Rossbach, X. Li, Y. Fang, D. Druta, F. Giust, L. Cominardi, W. Featherstone, B. Pike, and S. Hadad, "Developing software for multi-access edge computing," *ETSI White Paper No. 20*, pp. 1–38, 2019.

[24] F. Tang, H. Zhang, and L. T. Yang, "Multipath cooperative routing with efficient acknowledgement for leo satellite networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 179–192, 2018.

[25] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2020.

[26] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 260–288, 2018.

[27] E. Moro and I. Filippini, "Joint management of compute and radio resources in mobile edge computing: a market equilibrium approach," *IEEE Transactions on Mobile Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TMC.2021.3091764

[28] G. Cui, Q. He, X. Xia, F. Chen, T. Gu, H. Jin, and Y. Yang, "Demand response in noma-based mobile edge computing: A two-phase game-theoretical approach," *IEEE Transactions on Mobile Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TMC.2021.310858

[29] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[30] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030–3045, 2018.

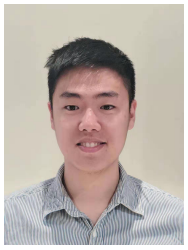
[31] A. Asheralieva and D. Niyato, "Combining contract theory and lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1213–1226, 2020.

[32] Y. Sun, M. Peng, and S. Mao, "A game-theoretic approach to cache and radio resource management in fog radio access networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 145–10 159, 2019.

[33] J. Zhao, X. Sun, Q. Li, and X. Ma, "Edge caching and computation management for real-time internet of vehicles: an online and distributed approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2183–2197, 2020.

[34] A. Asheralieva and D. Niyato, "Game theory and lyapunov optimization for cloud-based content delivery networks with device-to-device and uav-enabled caching," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 094–10 110, 2019.

[35] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.



Xiaoyu Xia received his Master degree from The University of Melbourne, Australia and his PhD degree from Deakin University, Australia. He is a research fellow at The University of Adelaide, Australia. His research interests include edge computing, service computing, software engineering and cloud computing. More details about his research can be found at https://sites.google.com/view/xiaoyuxia.

Feifei Chen received her PhD degree from Swinburne University of Technology, Australia in 2015. She is a senior lecturer at Deakin University. Her research interests include software engineering, cloud computing and green computing.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China, in 2010. He is an Associate Professor at Swinburne. His research interests include edge computing, software engineering, service computing and cloud computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

John C. Grundy received the BSc (Hons), MSc, and PhD degrees in computer science from the University of Auckland, New Zealand. He is currently Australian Laureate Fellow and a professor of software engineering at Monash University, Melbourne, Australia. He is an associate editor of the IEEE Transactions on Software Engineering, the Automated Software Engineering Journal, and IEEE Software. His current interests include domain-specific visual languages, model-driven engineering, large-scale systems engineering, and software engineering education. More details about his research can be found at https://sites.google.com/site/johncgrundy/.

Mohamed Abdelrazek is an Associate Professor of Software Engineering and IoT at Deakin University. Before joining Deakin University in 2015, he worked as a senior research fellow at Swinburne University of Technology and Swinburne-NICTA software innovation lab (SSIL). Before 2010, he was the head of software development department at Microtech. More details about his research can be found at https://sites.google.com/site/mohamedalmorsy/.

Jun Shen is an Associate Professor at the University of Wollongong in Wollongong, NSW Australia. His expertise is on web services and Semantic Web. He has been an editor, PC chair, guest editor, and a PC member for numerous journals and conferences published by the IEEE, ACM, Elsevier, and Springer. From 2007, he was a chair of Education Chapter of IEEE NSW section. He is a senior member of the IEEE.

Athman Bouguettaya is a Professor in the School of Computer Science at University of Sydney, Australia. He received his PhD in Computer Science from the University of Colorado at Boulder (USA) in 1992. He was previously Science Leader in Service Computing at CSIRO ICT Centre, Canberra. Australia. Before that, he was a tenured faculty member and Program director in the Computer Science department at Virginia Polytechnic Institute and State University (USA). He is or has been on the editorial boards of several journals including, the IEEE Transactions on Services Computing, ACM Transactions on Internet Technology and VLDB Journal. He is a Fellow of the IEEE and a Distinguished Scientist of the ACM.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. Jin received his PhD in computer engineering from HUST in 1994. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security.

APPENDIX A
SUMMARY OF NOTATIONS

TABLE 4: Summary of Notations

Notation	Description
C	remote cloud servers
$cost(\gamma^t)$	transmission cost incurred by γ^t
$cost_{u,v}$	minimum cost of transmission through edge $e_{u,v}$
d	data to be cached
E	set of links/edges existing in G
$e_{u,v}$	edge from node u to v
G	graph presenting a particular area
$\bar{l}$	time-averaged maximum transmission latency
$l^t_{u,v}$	time taken to transmit the data via edge $e_{u,v}$ in time slot t
$l_v^{\gamma^t}$	time taken to transmit the data to v according to γ^t
V_D^t	set of destination edge servers in time slot t
V_F^t	set of unavailable servers in time slot t
V_S^t	set of source servers in time slot t
$V^t_{D\leftarrow S}$	set of destination nodes retrieving data from a source node in time slot t
t	time slot t
T	number of time slots
u, v	nodes u, v in graph G
V	set of edge servers
γ	EDD strategy
γ^t	set of binary variables indicating whether data is transmitted through any edge in E in time slot t
$\gamma^t_{u,v}$	binary variable indicating whether data is transmitted through $e_{u,v}$ in time slot t
ω^t	set of binary variables indicating whether data transmission strategy visits node $v \in V$ in time slot t
ω^t_v	binary variable indicating whether data transmission strategy visits node v in time slot t

APPENDIX B
PROOF OF THEOREM 1

Proof To do this proof, we first introduce a known $\mathcal{NP}$-hard problem, the Rooted Minimum Steiner Tree (RMST) problem. Given a set of nodes V, a set of edges E and a graph $G(V, E, R, W, r)$ where R is the set of destination nodes $R \in V$, and W is the weights of edges $\forall e \in E$, the RMST problem is to find a Steiner tree from G that starts from node r with minimum weights. Now we present how to reduce the t-OEDD problem to the RMST problem. We first include a virtual node r' into graph G. Then, we add an edge to connect this virtual node to each of the source nodes in G. In the t-OEDD problem, each edge $e_{u,v}$ has a specific transmission cost $cost_{u,v}$, which can be treated as the weight of the corresponding edge in the RMST problem. In this way, the t-OEDD problem is converted to: given a graph G and a starting point r', find the Steiner tree with the minimum total weight (the minimum transmission cost). A solution that fulfills this reduced t-OEDD problem also fulfills the RMST problem. This indicates that the t-OEDD problem can be reduced to the RMST problem and the t-OEDD problem is thus $\mathcal{NP}$-hard. □

APPENDIX C
PROOF OF THEOREM 2

Proof Assuming two EDD decisions γ_v^t and $\gamma_v'^t$ that fulfill $\eta(\gamma_v^t, \gamma_{-v}^t) < \eta(\gamma_v'^t, \gamma_{-v}^t)$. In this case, this node should change its EDD decision from γ_v^t to $\gamma_v'^t$. Based on (17), we can obtain:

$$\begin{aligned}&\max\{\sigma_t, 1\} l_v^{\gamma^t} + \zeta \cdot \Delta cost(\gamma_v^t, \gamma_{-v}^t) \\ &> \max\{\sigma_t, 1\} l_v^{\gamma'^t} + \zeta \cdot \Delta cost(\gamma_v'^t, \gamma_{-v}^t)\end{aligned} \tag{20}$$

Once node v needs to update its EDD decision, the benefits of nodes V_v^t, might be impacted by v's new latency . Thus, we analyze the following two cases: 1) $l_v^{\gamma'^t} > l_v^{\gamma^t}$; and 2) $l_v^{\gamma'^t} \leq l_v^{\gamma^t}$.

Case 1: $l_v^{\gamma'^t} > l_v^{\gamma^t}$.

In this case, the latencies of all nodes in V_v^t increase. Thus, those nodes are not in the set of $V_D^t \bigcap \neg\{u \in V_v^t | l_u^{\gamma'^t} > l_u^{\gamma^t}\} \bigcap \neg\{v\}$. In this way, we can obtain:

$$\begin{aligned}&\sum_{u \in V_D^t \bigcap \neg\{u \in V_v^t | l_u^{\gamma'^t} > l_u^{\gamma^t}\} \bigcap \neg\{v\}} (\max\{\sigma_t, 1\} l_u^{\gamma^t} + \\ &\zeta \cdot \Delta cost(\gamma_u^t, \gamma_{-u}^t)) > \sum_{u \in V_D^t \bigcap \neg\{u \in V_v^t | l_u^{\gamma'^t} > l_u^{\gamma^t}\} \bigcap \neg\{v\}} \\ &(\max\{\sigma_t, 1\} l_u^{\gamma'^t} + \zeta \cdot \Delta cost(\gamma_u'^t, \gamma_{-u}^t))\end{aligned} \tag{21}$$

Based on (19), (20) and (21), we can obtain:

$$\phi(\gamma_v^t, \gamma_{-v}^t) < \phi(\gamma_v'^t, \gamma_{-v}^t) \tag{22}$$

Case 2: $l_v^{\gamma'^t} \leq l_v^{\gamma^t}$.

In this case, nodes in V_v^t would keep their current decisions because their benefits increase, due to the lower latency and same cost. Thus, we also can obtain (21) and prove (22).

According to Definition 2, $\phi(\gamma_v^t, \gamma_{-v}^t)$ is a potential function and this theorem stands. □

APPENDIX D
PROOF OF THEOREM 3

Proof Let $\gamma^* = \{\gamma^{*0}, ..., \gamma^{*T-1}\}$ denote the optimal solution to $\mathcal{P}_1$, which incurs transmission cost $cost^{*t}$ and $\gamma' = \{\gamma'^0, ..., \gamma'^{T-1}\}$ denote the optimal solution to $\mathcal{P}_2$, which incurs transmission cost $cost'^t$. The solution to $\mathcal{P}_2$ from its solution space contains the solution space of $\mathcal{P}_1$. Now, we deduce the upper bound on (13) with respect to $cost^{*t}$:

$$\begin{aligned}&\zeta \cdot \mathbb{E}[cost'^t | \sigma_t] + \Delta(\sigma_t) \\ &\overset{(15)}{\leq} \zeta \cdot \mathbb{E}[cost'^t | \sigma_t] - \mathbb{E}[\sigma_t \cdot \sum_{v \in \mathcal{R}^t} (\bar{l} - l_v^{\gamma^t}) | \sigma_t] + \Theta \\ &\leq \zeta \cdot \mathbb{E}[cost^{*t} | \sigma_t] - \mathbb{E}[\sigma_t \cdot \sum_{v \in \mathcal{R}^t} (\bar{l} - l_v^{\gamma^{*t}}) | \sigma_t] + \Theta \\ &\overset{(\dagger)}{\leq} \zeta \cdot \mathbb{E}[cost^{*t} | \sigma_t] - \sigma_t \cdot \sum_{v \in \mathcal{R}^t} (\bar{l} - l_v^{\gamma^{*t}}) + \Theta \\ &\overset{(\ddagger)}{\leq} \zeta \cdot \mathbb{E}[cost^{*t} | \sigma_t] + \Theta\end{aligned} \tag{23}$$

Note that the optimal solution γ^* is independent of the accumulated latency. Thus, inequality (†) in (23) stands. Inequality (‡) holds because constraint (4) must be fulfilled

by the optimal solution γ^* to problem $\mathcal{P}_1$ while $\sigma_t \geq 0$ based on (10). By summing (23) across $t \in [0, T-1]$, we can obtain:

$$\begin{aligned}
&\lim_{T\to\infty} \frac{\zeta}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost'^t|\sigma_t] \\
&\leq \lim_{T\to\infty} \frac{\zeta}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost^{*t}|\sigma_t] + \Theta - \frac{\mathbb{E}[\mathcal{L}(\sigma_T) - \mathcal{L}(0)]}{T} \\
&\leq \lim_{T\to\infty} \frac{\zeta}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost^{*t}|\sigma_t] + \Theta
\end{aligned} \tag{24}$$

where $\mathcal{L}(\sigma_T) - \mathcal{L}(0)$ is always non-negative according to definition of $\mathcal{L}(\sigma_t)$.

Thus, the margin between the solution γ' found by $\mathcal{P}_2$ and the optimal solution γ^* is bounded by $\frac{\Theta}{\zeta}$:

$$\lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost'^t - cost^{*t}|\sigma_t] \leq \frac{\Theta}{\zeta} \tag{25}$$

Now, we analyze the margin between γ found by LAO and the optimal solution γ^* found by $\mathcal{P}_1$. Let $cost_{mm}$ denote the maximum value of the minimum transmission cost between any two nodes in G, calculated with $\max\{cost_{u,v}|u, v \in V \bigcup C\}$. Please note that the strategy found by LAO in time slot t does not impact the sets of source nodes, destination nodes and unavailable nodes in the following time slot. Thus, the difference between the total cost incurred by LAO and that incurred by the optimal solution γ' found by $\mathcal{P}_2$ is at most $\Delta V \cdot (cost_{mm} - cost_{min})$, where $\Delta V = \lim_{T\to\infty} |V_D^t \neg \bigcap V_S^t|$:

$$\lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost^t - cost'^t|\sigma_t] \leq \Delta V \cdot (cost_{mm} - cost_{min}) \tag{26}$$

Based on (25) and (26), we can obtain the margin between γ found by LAO and γ^* found by $\mathcal{P}_1$:

$$\begin{aligned}
&\lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=0}^{T-1} \mathbb{E}[cost^t - cost^{*t}|\sigma_t] \leq \\
&\Delta V \cdot (cost_{mm} - cost_{min}) + \frac{\Theta}{\zeta}
\end{aligned} \tag{27}$$

As mentioned in Section 3.2, Θ is a constant of $\frac{1}{2}\sum_{v\in\mathcal{R}^t} \bar{l}^2$. Since $cost_{mm}$, $cost_{min}$ and ΔV are also constants in an EDD scenario, the margin between γ and γ^* is $O((cost_{mm} - cost_{min}) \cdot \Delta V + \frac{\Theta}{\zeta}) = O(\frac{1}{\zeta})$. □

APPENDIX E
PROOF OF THEOREM 4

Proof According to (10) and (11), let us assume a positive value $\mathcal{X}$ and the existence of a γ'^t that fulfill:

$$\mathbb{E}[\sum_{v\in V_D^t} (l_v^{\gamma'^t} - \bar{l})|\sigma_t] \leq -\mathcal{X} \tag{28}$$

Let $cost_{min}$ and $cost_{max}$ denote the lowest and highest transmission costs achieved by the all the possible solutions to $\mathcal{P}_1$, respectively. Introducing $cost_{min}$ and $cost_{max}$ into (15) leads to:

$$\begin{aligned}
&\Delta(\sigma_t) + \zeta \cdot cost_{min} \leq \\
&\qquad \Theta + \zeta \cdot cost_{max} + \sigma_t \cdot \mathbb{E}[\sum_{v\in V_D^t} (l_v^{\gamma'^t} - \bar{l})|\sigma_t]
\end{aligned} \tag{29}$$

Define $\Theta' = \Theta + \zeta \cdot (cost_{max} - cost_{min})$. We have the following:

$$\begin{aligned}
\Delta(\sigma_t) &\leq \Theta' + \sigma_t \cdot \mathbb{E}[\sum_{v\in V_D^t} (l_v^{\gamma'^t} - \bar{l})|\sigma_t] \\
&\overset{(28)}{\leq} \Theta' - \mathcal{X} \cdot \sigma_t
\end{aligned} \tag{30}$$

The upper bound on the average accumulated latency across all the time slots can be obtained:

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1} \sigma_t &\overset{(30)}{\leq} \frac{\Theta'}{\mathcal{X}} - \frac{\sum_{t=0}^{T-1} \Delta(\sigma_t)}{T\mathcal{X}} \\
&\overset{(12)}{=} \frac{\Theta'}{\mathcal{X}} - \frac{\mathbb{E}[L(\sigma_T) - \mathbb{E}[L(\sigma_0)]}{T\mathcal{X}} \leq \frac{\Theta'}{\mathcal{X}}
\end{aligned} \tag{31}$$

Considering the fact that $O(\frac{\Theta'}{\mathcal{X}}) = O(\zeta)$, the averaged accumulate latency across all the time slots of LAO is bounded by $O(\zeta)$. With a lower ζ, the time-averaged accumulate latency is also lower, indicating the acceleration of the stabilization. □