# Pairwise Markov Chain: A Task Scheduling Strategy for Privacy-Preserving SIFT on Edge

Hengrun Zhang and Kai Zeng
Department of Computer Science, Department of Electrical and Computer Engineering
George Mason University, Fairfax, VA USA
{hzhang18, kzeng2}@gmu.edu

***Abstract*—In this paper, we propose a task scheduling strategy, which can achieve image feature extraction on edge while ensuring privacy. Our task scheduling strategy applies to a fairly popular privacy-preserving Scale-Invariant Feature Transform (SIFT) scheme, where images to be processed are firstly randomly split into two portions for encryption and transmitted to two different edge nodes for feature extraction. Then, in the edge, our task scheduling strategy will re-assign these two portions to proper edge nodes for processing. During the whole process, two portions of the same image should not be assigned to the same edge node in order to preserve privacy. We show that this privacy constraint can be enforced through constructing a pairwise Markov chain, and carefully designing system states and transition probabilities. We further formulate the whole task scheduling problem as a stochastic latency minimization problem and solve it by converting it into a linear programming problem. Simulation results show that our proposed task scheduling strategy can achieve lower latency than baseline strategies while satisfying the privacy constraint.**

## I. Introduction

The emergence of multimedia data colors people's daily life. However, many tasks related to processing those multimedia data are usually computationally expensive, and people have to turn to cloud computing for help. Recently, the emergence of edge computing further improves the computation performance of cloud computing, since edge computing just requires data to be uploaded to close-by nodes in the edge instead of far-away clouds. This will significantly decrease network latency, especially considering the fact that multimedia data usually have large sizes. However, due to computation resource limitation, edge nodes will not have servers with the same capacity as those in remote clouds. In this case, a well-designed task scheduling strategy is needed to optimize the performance on edge. In recent years, several resource allocation and task scheduling works have been proposed for edge computing with different system configurations [1]–[6]. Nevertheless, none of them consider privacy issues during task scheduling.

Actually, the privacy-preserving requirement has been considered for not a short time in many multimedia applications. In computer vision, image matching is based on image features extracted through algorithms usually with very high computational requirement, such as Scale-Invariant Feature Transform (SIFT) [7]. People have to rely on cloud computing when the number of images to be processed is large. However, some images, such as profiles and medical images, have sensitive contents that are not supposed to go public. In this case, when images are uploaded to clouds for SIFT feature extraction, image owners do not want to reveal the image content to the cloud server. In recent years, many related works have been done for privacy-preserving SIFT [8]–[10]. These works usually realize SIFT feature extraction in the encrypted domain through homomorphic addition, multiplication and comparison. Later, it is shown in [11] that if just one cloud is used for securing SIFT, the security and privacy can still be compromised due to unique statistical characteristics of images, and using more than one servers for securing SIFT could be a feasible alternative. In [11], [12], images are randomly split into two portions and transmitted to two clouds for SIFT processing. Such a split actually acts as a means of encryption, which has a much stronger guarantee for privacy than the traditional homomorphic encryption. Besides, the computational requirement is also reduced significantly.

All the above privacy-preserving SIFT algorithms are only based on cloud computing, which always suffer from large network latency, since large size images have to be transmitted to remote servers. With the emergence of the new computing paradigm of edge computing, we now can offload the feature extraction part to the edge. Fig. 1 shows the difference between cloud-based and our proposed edge-based image matching framework. Different from the cloud-based image matching framework, which requires to upload images to the remote cloud, our edge-based counterpart just needs to upload images to the nearby edge for image feature extraction. After that, the extracted features can be uploaded to the cloud for image matching. This will greatly reduce transmission latency and bandwidth consumption in the core network. For example, let's consider an image of $3,000 \times 3,000$, which is the size of images considered in [12]. Since each pixel is usually represented by a value ranging from 0 to 255, we need 8 bits to describe each pixel. Thus, the total data size should be $8 \times 3,000 \times 3,000 = 8.85$ MB if we consider just one image channel. The largest number of SIFT keypoints in those images is 35,299. Each keypoint is described with a 128-dimension vector. Each element in the vector has a value ranging from 0 to 15, which needs 4 bits for representation. Therefore, the total data size will be reduced to $4 \times 128 \times 35299 = 2.26$ MB after SIFT feature extraction, which is almost one-fourth of the original size. In other words, we only have one-fourth of the original transmission latency and bandwidth consumption in the core network if we transmit extracted keypoints instead

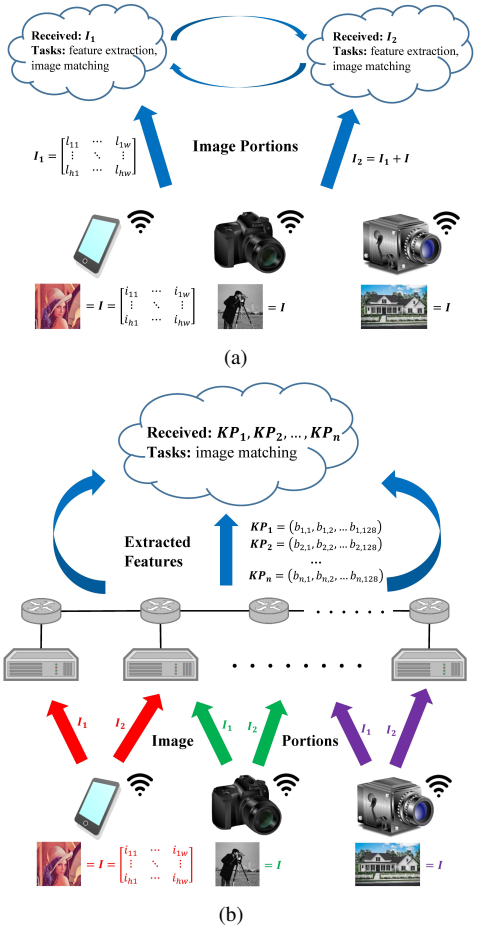



Fig. 1. Comparison between cloud-based and edge-based privacy-preserving image matching framework. (a) Cloud-based privacy-preserving image matching. (b) Edge-based privacy-preserving image matching.

of the original images.

A typical edge network usually consists of multiple connected edge nodes. The feature extraction tasks can be distributed to proper edge nodes for processing in order to optimize the performance, such as delay or load balancing. On the other hand, we need to ensure that two portions of the same image are not scheduled to the same edge node for processing. Otherwise, the edge node could reconstruct the image, which violates privacy. To the best of our knowledge, there are no existing works considering such a privacy constraint in task scheduling on edge. In this paper, we will consider task scheduling for a system like this: to achieve a privacy-preserving scheme like the privacy-preserving SIFT discussed above, an image is first divided into two portions, which are sent to two different edge nodes. After that, portions can be either directly processed at the corresponding edge node or switched to another edge node for processing to minimize queuing delay. For privacy consideration, we need to make sure that no two portions of the same image are ever assigned or switched to the same edge node.

Based on the above system and privacy constraint, we propose a stochastic task scheduling strategy. A Markov chain will be constructed to facilitate the stochastic task scheduling on the edge nodes. Once edge nodes receive image portions, they will decide whether to process those portions themselves or assign portions to other nodes based on the constructed Markov chain. Considering the privacy constraint, we cannot consider each node separately. Instead, each two nodes will be considered in pairs, and this is why we call our Markov chain "pairwise". The chain will be constructed carefully with possible pair transitions. We can construct our pairwise Markov chain in advance based on the property that the stationary state of a Markov chain is not affected by its initial state. The optimized queuing and processing latency on edge nodes can then be computed and recorded in advance, which saves computational delay when deciding the scheduling. Simulation results validate that our proposed strategy can achieve an efficient task scheduling while ensure the privacy. The main contributions of this paper are as follows:

- We propose an edge computing based deployment for a fairly popular application of privacy-preserving SIFT. We formulate the task scheduling problem with the consideration of privacy constraint that no two portions of the same image should be scheduled to the same edge node for processing. To the best of our knowledge, we are the first one to consider such a privacy constraint in task scheduling problems on edge.
- We propose a pairwise Markov chain to enforce the privacy constraint. The system states and transition probabilities are carefully designed. Achievability of the stationary state of the chain is proved.
- We integrate the proposed pairwise Markov chain with the optimization model, construct a stochastic optimization problem for queuing plus processing latency minimization with the privacy constraint enforcement, and show that the problem can be solved through transforming it to a linear programming problem.

The rest of this paper is organized as follows. In Section II, we give a full description of our privacy constrained task scheduling system. In Section III, our constructed pairwise Markov chain for privacy constraint enforcement is introduced. Our optimization strategy through linear programming is presented in Section IV. Experimental results for validation and evaluation are given in Section V. We discuss possible improvements and extensions in Section VI. Section VII discusses related works and conclusions will be drawn in Section VIII.

## II. Model Construction and Problem Formulation

In this paper, we consider the privacy-preserving SIFT proposed in [12] as an application scenario. According to [7], SIFT is realized through keypoint localization, orientation assignment, and keypoint descriptor. The key part is generation of the difference-of-Gaussian scale space, which requires extensive convolution computation. For privacy-preserving SIFT in [12], the original image $\boldsymbol{I}$ is split into two portions. For the first portion, a number ranging from 0 to 255 is randomly selected for each pixel. This will generate $\boldsymbol{I_1}$. The second portion is generated through adding the original image with the first portion. In other words, we have $\boldsymbol{I_2} = \boldsymbol{I} + \boldsymbol{I_1}$. After that,

the two portions will respectively generate two difference-of-Gaussian scale spaces, and those three steps for SIFT feature extraction can be done through homomorphic addition and comparison. With the above privacy-preserving scheme, users need to split the image into two portions for encryption, and transmit these two portions to two different edge nodes. In the edge, we need to ensure that no two portions of the same image are assigned or switched to the same edge node.

### A. System Model

Let's consider an edge network, denoted as $G = (V, E)$, where $V$ and $E$ represent the set of nodes and links in the network. Each edge node is assumed to have both communication and computing capabilities. All the links among edge nodes are considered full-duplex. In this paper, we assume that communication among edge nodes are through high-speed wired links. For example, in a campus wireless network, the edge nodes could be access points (APs), which are connected by high speed Ethernet with transmission rate of 10 Gbps. Therefore, we assume that the transmission delay among the edge nodes are negligible compared with queuing delay in the edge node.

For an edge network with $K$ edge nodes $V = \{v_1, v_2, \cdots, v_K\}$, when an image portion is sent to an edge node $v_i$ by the user (either through a wireless or wired link), the node will firstly decide whether it will process the portion itself or switch the portion to another node $v_j \neq v_i$. We assume that if a portion gets switched to another node once, it will not be switched again. In other words, when the portion arrives at the head-of-line of node $v_j$, it will be processed by $v_j$. Thus, the whole process can be seen as a *2-hop* behavior. In this paper, we use $P_i$ to denote the proportion of image portions that are processed by itself for node $v_i$ when the edge network gets stable.

Each edge node $v_i$ has a *2-hop* process from its own perspective. An example of the *2-hop* process with four edge nodes ($K = 4$) is illustrated in Fig. 2(a). The *2-hop* process will generate $K$ queues. The first queue records image portions to be processed by node $v_i$ itself, which is denoted as $\mathcal{Q}_{i,i}$. All the other queues $\mathcal{Q}_{i,j}$ record image portions to be switched to node $v_j$. With a slight abuse of notation, we use $Q_{i,i}(t)$ and $Q_{i,j}(t)$ to denote their queue lengths in each time slot $t$, respectively. Besides, we use $D_{i,i}(t)$ and $D_{i,j}(t)$ to denote their head-of-line (HOL) delay [13], respectively. According to Little's Law [14], if the entire edge network can become stable, the queue length and HOL delay should have the following relationship in the stationary state.

$$d_{i,k} = \frac{1}{\alpha_{i,k}} \lim_{n_0 \to \infty} \sum_{n=0}^{n_0} n \cdot Pr\{q_{i,k} = n\} \tag{1}$$

Here, node $v_k$ can be both $v_i$ and $v_j$. $\alpha_{i,k}$ represents the long-term average arrival rate for queue $\mathcal{Q}_{i,k}$. In this paper, we use the average arrival rate within a long enough time interval for approximation. Note that a unit arrival process is assumed in this paper, which makes all the arrival rates between 0 and 1. $d_{i,k}$ and $q_{i,k}$ respectively represent the HOL delay and queue length for queue $\mathcal{Q}_{i,k}$ as the entire system becomes stable. $n$ and $n_0$ are the possible and maximum queue length individually. $Pr\{q_{i,k}\} = n$ reflects the probability where the stable queue length of $\mathcal{Q}_{i,k}$ is equal to $n$.

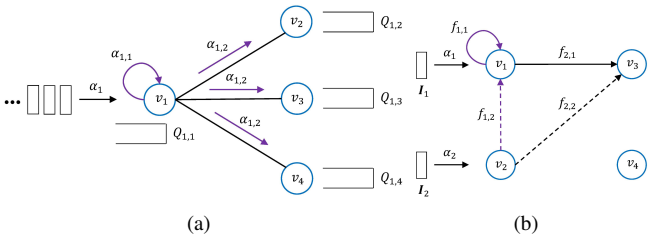



Fig. 2. Example of our *2-hop* process and transmissions not permitted under privacy constraint. (a) *2-hop* process. (b) Transmissions not permitted.

As has been discussed above, our scheduling policy can be seen as a *2-hop* process. We can assume that the long-term average arrival rate of node $v_i$ is $\alpha_i$. This $\alpha_i$ can actually also be seen as the arrival rate in the first hop. Since once an image portion arrives at node $v_i$, the portion will be processed by node $v_i$ itself with a proportion $P_i$, the arrival rate for $\mathcal{Q}_{i,i}$ can then be denoted as $\alpha_{i,i} = P_i\alpha_i$. On the other hand, the proportion where an arrived image portion is scheduled to another node is $1 - P_i$ obviously. In this paper, we assume that the scheduled node is randomly selected from the other $(K-1)$ nodes, which makes the long-term average arrival rate of $\mathcal{Q}_{i,j}$ equal to $\alpha_{i,j} = \frac{1}{K-1}(1 - P_i)\alpha_i$. From this equation, we can see that all $\alpha_{i,j}$ are actually the same. For notation ease, we can respectively use $\alpha_{i,1}$ and $\alpha_{i,2}$ to represent $\alpha_{i,i}$ and all the $\alpha_{i,j}$.

In this paper, we assume that all edge nodes have the same computation capacity. In this case, service time is just related to tasks. For SIFT feature extraction, most of the computations happen in generation of the difference-of-Gaussian scale space. Therefore, if the number of scales is selected the same and all image portions have the same size, we can assume that privacy-preserving SIFT feature extraction for those image portions has the same computational overhead. Based on the above assumption, we can consider service time for those image portions the same, which is denoted as $N$, in the number of time slots. With the same arrival and service rate, we can claim that when the whole edge network becomes stable, all those queue lengths $Q_{i,j}(t)$ should converge to a single length $q_{i,2}$. Considering Little's Law discussed above, we also have the converged HOL delay $d_{i,2}$. Besides, we also use $q_{i,1}$ and $d_{i,1}$ to represent the converged queue length and HOL delay for $\mathcal{Q}_{i,i}$.

### B. Optimization Problem Formulation

In this paper, we aim at minimizing the total latency under the system model introduced in the last section. According to the definition, the total latency is actually composed of queuing time and service time. However, as has been discussed in the last section, service time is fixed with the given servers and tasks. In this case, our optimization problem can actually be simplified to minimizing queuing time, which can be represented by HOL delay. In our defined system model, we

have two kinds of HOL delay, $d_{i,1}$ and $d_{i,2}$. Obviously, we want them both to be minimized. In this case, we define the summation of these two kinds of HOL delay as our objective to be minimized, which is denoted as $T_i$ shown below:

$$T_i \triangleq d_{i,1} + d_{i,2} \tag{2}$$

From the perspective of each edge node $v_i$, there is a similar *2-hop* process, which corresponds to a similar optimization problem. In this paper, we only target one of these optimization problems, which can be defined as below:

$$\begin{aligned} \min \quad & T_i \\ \text{s.t.} \quad & \phi(\boldsymbol{I_1}, \mathcal{Q}_{i,k}) + \phi(\boldsymbol{I_2}, \mathcal{Q}_{i,k}) \neq 2 \quad i,k = 1,2,\cdots,K \end{aligned} \tag{3}$$

The indicator function $\phi(\boldsymbol{I}, \mathcal{Q}_{i,k})$ is defined as follows:

$$\phi(\boldsymbol{I}, \mathcal{Q}_{i,k}) \triangleq \begin{cases} 1 & \boldsymbol{I} \in \mathcal{Q}_{i,k} \\ 0 & \boldsymbol{I} \notin \mathcal{Q}_{i,k} \end{cases} \tag{4}$$

The constraint in (4) actually corresponds to the privacy constraint in this paper, which requires that no edge nodes should get both two portions of the same image. Fig. 2(b) shows an example with four edge nodes. In this example, two image portions of the same image individually arrive at node $v_1$ and $v_2$. We illustrate two pairs of transmissions that should not be permitted, $(f_{1,1}, f_{1,2})$ and $(f_{2,1}, f_{2,2})$. The privacy constraint is enforced through our pairwise Markov chain to be described in detail in Section III. We will tune the transition probabilities in our Markov chain to optimize the delay performance.

## III. Pairwise Markov Chain for Privacy Constraint Enforcement

In this paper, we consider an attack model like this: an attacker wants to recover the content of some images. In order for this, the attacker randomly selects and eavesdrops a node in our edge network. Therefore, our task scheduling policy should avoid transmitting two portions of the same image to the eavesdropped node. Since the eavesdropped node can be any node in the edge network, our policy should be further extended to the case that two portions of the same image should not be transmitted to any node in the edge network. When it comes to a traditional task scheduling problem, the whole process can actually be described by a Markov chain. Here, we carefully modify the Markov chain by considering each two edge nodes in pairs to reflect the privacy constraint.

### *A. Optimization Problem Reconsideration and Pairwise Markov Chain Construction*

Our constructed pairwise Markov chain is based on a pairwise architecture. We start from a simple example of an edge network with three edge nodes, shown in Fig. 3.

From this figure, we can see that each two edge nodes are considered in pairs to generate a new node. Based on our pairwise architecture, if two different edge nodes $v_1$ and $v_2$ receive two portions of the same image, generally speaking, they will have two choices: processing the two portions

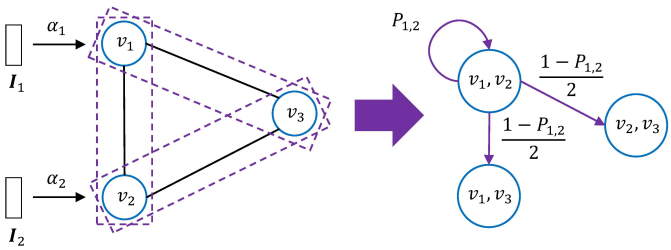



Fig. 3. Example of an edge network and transformed pairwise architecture.

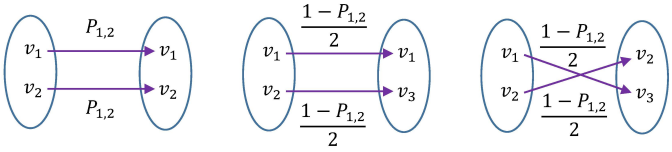



Fig. 4. Detailed assignments of image portions.

themselves, or transmitting the two portions to another pair. If they choose to process the two portions themselves, obviously, there will be no violation of the privacy constraint. If they choose to schedule the portions to another pair, there will be another two choices: transmitting the two portions to $v_1$ and $v_3$, or $v_2$ and $v_3$. When image portions are transmitted to $v_1$ and $v_3$, it can actually be seen as the case that the first image portion is still processed by $v_1$ itself, while the second image portion is transmitted from $v_2$ to $v_3$. A similar case applies to image portions being transmitted to $v_2$ and $v_3$. The detailed assignments are shown in Fig. 4. It is easy to imagine when an edge network has $K(K > 3)$ nodes, a node pair $(v_{i_1}, v_{i_2})$ can have three kinds of choices (instead of just two in the above example) when two received image portions are determined to be transmitted to another pair. Besides being transmitted to node pair $(v_{i_1}, v_{j_2})$ and $(v_{j_1}, v_{i_2})$, the image portions can also be transmitted to node pair $(v_{j_1}, v_{j_2})$. Here, $i_1, i_2, j_1, j_2 = 1, 2, \cdots, K$, and $i_1, i_2, j_1$ and $j_2$ are all different.

With the above pairwise architecture, two image portions of the same image have no chance to be transmitted to the same node. On the other hand, we need to reconsider the system model and optimization problem described in Section II-A. Previously, we consider the whole system from the perspective of each edge node. With the introduction of privacy constrain, we should consider the system in node pairs. Fortunately, the system still follows Little's Law, but with a little modification for the arrival rate of each edge node. In Fig. 3, we give the proportion of image portions processed by node pair $(v_1, v_2)$ itself, denoted as $P_{1,2}$. Based on previous discussion, we have shown that even if two image portions of the same image will be transmitted to another node pair, the situation still exists that one of these two image portions will still be processed by the original node. In other words, the proportion of a node $P_{i_1}$ is not necessarily equal to the proportion of a node pair $P_{i_1,i_2}$. In the example shown in Fig. 3, we actually have $P_1 = P_2 = P_{1,2} + \frac{1-P_{1,2}}{2} = \frac{1+P_{1,2}}{2}$. When the case is extended

to an edge network with $K$ edge nodes, we should have the following relationship:

$$\begin{aligned} P_{i_1} = P_{i_2} &= P_{i_1,i_2} + \frac{\binom{1}{K-1}-1}{\binom{2}{K}-1}(1-P_{i_1,i_2}) \\ &= \frac{2+(K-1)P_{i_1,i_2}}{K+1} \end{aligned} \tag{5}$$

The arrival rates need to be modified accordingly when described by $P_{i_1,i_2}$.

From (5), we can see that $P_{i_1}$ and $P_{i_2}$ are the same. Actually, if we consider the whole edge network from the perspective of node pair $(v_{i_1}, v_{i_2})$, the first-hop arrival rate of these two nodes $\alpha_{i_1}$ and $\alpha_{i_2}$ are also the same, denoted as $\lambda_{i_1,i_2}$ for notation ease. This is because a node pair corresponds to two image portions of the same image. Obviously, these two image portions always arrive at the same time. In this situation, the minimal HOL delay summation, described in (2), of these two nodes should be the same. In other words, we just need to minimize the HOL delay summation from the perspective of one of these two nodes, which makes the optimization objective in (3) unchanged. Besides, with our generated pairwise architecture, the privacy requirement can be met, which corresponds to the constraint in (3).

Similar to traditional scheduling policies, our pairwise scheduling process can also be described by a Markov chain, which is called as pairwise Markov chain in this paper. Each system state of our constructed pairwise Markov chain is denoted as $\boldsymbol{Z}(t) = (\boldsymbol{W}(t), \boldsymbol{Q}(t))$, where $\boldsymbol{W}(t)$ and $\boldsymbol{Q}(t)$ represent the working status and queue length vector, respectively. Each element $w_k(t)$, with $k = 1, 2, \cdots, K$, in $\boldsymbol{W}(t)$ reflects how many time slots are still needed for the $k$-th server to complete its current task. Recall that in Section II-A, we assume that the total number of time slots for an edge node to complete a task is $N$. Therefore, each $w_k(t)$ has $N+1$ possible values, with $w_k(t) = 0, 1, 2, \cdots, N$, where $w_k(t) = 0$ means that the server is idle in this time slot. Each element $q_k(t)$ in $\boldsymbol{Q}(t)$ represents the queue length with a structure similar to that of $w_k(t)$. For the edge network with $K$ nodes, the state space $S \subseteq \{0, 1, \cdots, N\}^K \times \{0, 1, \cdots M\}^K$ represents all possible states of our Markov chain that can be reached from the initial state, which is defined as $\boldsymbol{Z}_0 \triangleq (\boldsymbol{0}_{K\times 1}, \boldsymbol{0}_{K\times 1})$. $M$ depends on the storage limit of each server.

### *B. Pairwise Markov Chain Achievability*

According to [15], an irreducible Markov chain has a positive stationary distribution if and only if all of its states are positive recurrent. In other words, we can prove that our Markov chain has a stationary distribution by showing that it is irreducible and positive recurrent.

**Proposition 1.** *The pairwise Markov chain proposed in this paper is irreducible.*

*Proof.* For irreducibility, we should show that any two states can be reached from each other in our Markov chain. Since by the definition of our Markov chain, the initial state can reach any other states, we just need to show that any states can reach the initial state. In our Markov chain, we assume that the initial state has a possible transition to itself. This transition is necessary and reasonable, since we should consider the case that no tasks are in the edge in a certain time slot. Such an assumption will just make our Markov chain aperiodic. This means that for a given set of transition probabilities, our Markov chain will just have one stationary distribution, which does not influence our proof.

With the above assumption, we can give a proof for irreducibility as follows. For any state $\boldsymbol{Z}(t)$, we can find the server with the longest queue length $q_{max}$. Recall that each task can be completed in $N$ time slots. Then, we can assume an event that in $N(q_{max}+1)$ time slots, there are no tasks arriving at the edge. From Section III-A, we know that the arrival rate of node pair $(v_{i_1}, v_{i_2})$ is $\lambda_{i_1,i_2}$. Then, the probability of this event is $(1-\lambda_{i_1,i_2})^{N(q_{max}+1)} > 0$, which means that this event can happen. If this event happens, the edge will complete all tasks that are both in service and queued in $N(q_{max}+1)$ time slots. In other words, the current state will transit to the initial state $\boldsymbol{Z}_0$ in at most $N(q_{max}+1)$ time slots. Note that some states $\boldsymbol{Z}(t)$ may transit to the initial state in less than $N(q_{max}+1)$ time slots with our task switch operations. However, this does not matter since for the initial state, we have already assumed a possible transition to itself. Since any state in our Markov chain can have such an event, any state can reach back to the initial state, which proves the irreducibility. ■

Now, we have shown that our Markov chain is irreducible. We still have to show that our Markov chain is positive recurrent. Here, we utilize Foster-Lyapunov theorem [16] to prove it.

**Proposition 2.** *The pairwise Markov chain proposed in this paper is positive recurrent.*

*Proof.* Consider a Lyapunov function defined as follows:

$$V(\boldsymbol{Z}(t)) \triangleq \|\boldsymbol{W}(t)\| + \|\boldsymbol{Q}(t)\| = \sum_{k=1}^{K} w_k(t) + q_k(t) \tag{6}$$

Here, $\|\cdot\|$ is the $L_1$-norm. Then, according to Foster-Lyapunov theorem, it suffices to show that for any given state $\boldsymbol{Z}_c$, our Markov chain has:

$$E[V(\boldsymbol{Z}(t+1)) - V(\boldsymbol{Z}(t))|\boldsymbol{Z}(t) = \boldsymbol{Z}_c] \leq -\delta,\ \boldsymbol{Z}_c \in F \tag{7a}$$

$$E[V(\boldsymbol{Z}(t+1)) - V(\boldsymbol{Z}(t))|\boldsymbol{Z}(t) = \boldsymbol{Z}_c] < C,\ \boldsymbol{Z}_c \notin F \tag{7b}$$

Here, $E[\cdot]$ calculates the expected value. $\delta$ and $C$ are two strict positives. $F$ denotes some finite set. Next, we will show how to find $\delta$, $C$ and $F$.

In our Markov chain, we define $F$ to include all states where every node in the server is busy with some task. Formally, $F = \{\boldsymbol{Z}_F = (\boldsymbol{W}_F, \boldsymbol{Q}_F) | \boldsymbol{Z}_F \in \boldsymbol{Z}, w_{F,k} \neq 0 \text{ for } \forall w_{F,k} \in$

$\boldsymbol{W}_F, k = 1, 2, \cdots, K\}$. In this case, for any $\boldsymbol{Z}_c \in F$, we have:

$$\begin{aligned}
&E[V(\boldsymbol{Z}(t+1)) - V(\boldsymbol{Z}(t))|\boldsymbol{Z}(t) = \boldsymbol{Z}_c] \\
=&E[\|\boldsymbol{W}(t+1)\| + \|\boldsymbol{Q}_(t+1)\| - \|\boldsymbol{W}(t)\| - \|\boldsymbol{Q}(t)\|| \\
&\boldsymbol{Z}(t) = \boldsymbol{Z}_c] \\
=&\sum_{k=1}^{K}[w_k(t+1) - w_k(t)] + \sum_{k=1}^{K}[q_k(t+1) - q_k(t)] \\
\leq& -K + 2 < 0
\end{aligned} \tag{8}$$

Since every edge node is busy with some task, after one time slot passes, each element in the working status vector can only decrease by 1. With totally $K$ edge nodes, $\|\boldsymbol{W}(t+1)\|$ will decrease by $K$ compared with $\|\boldsymbol{W}(t)\|$. Besides, with the unit arrival process assumption, there are at most two new tasks (one image split into two portions) arriving at node pair $(v_{i_1}, v_{i_2})$ in the current time slot. Then, no matter whether these two tasks are processed by the current nodes or transmitted to other nodes, $\|\boldsymbol{Q}(t+1)\|$ will not be further changed. Therefore, $\|\boldsymbol{Q}(t+1)\|$ can at most increase by 2 compared with $\|\boldsymbol{Q}(t)\|$. Then, the expected value is at most equal to (8). Furthermore, if we want to make the scheduling problem meaningful, $K$ should be larger than 2. Therefore, (8) is less than 0. In this case, for any $\boldsymbol{Z}_c \in F$, the expected value is strictly less than 0. In other words, we can find a strictly positive $\delta$, satisfying formula (7a).

On the other hand, for any $\boldsymbol{Z}_c \notin F$, some edge nodes may be idle, which may make the total decrease of $\|\boldsymbol{W}(t)\|$ less than $K$. Besides, similar to the case of $\boldsymbol{Z}_c \in F$, $\|\boldsymbol{Q}(t+1)\|$ can at most increase by 2 compared with $\|\boldsymbol{Q}(t)\|$. Then, we can see that the largest expected value happens when all edge nodes are idle. In other words, we have (9), which is not greater than 2. In this case, we can pick any $C > 2$. Then, we can have that for any $\boldsymbol{Z}_c \notin F$, formula (7b) is satisfied.

$$\begin{aligned}
&E[V(\boldsymbol{Z}(t+1)) - V(\boldsymbol{Z}(t))|\boldsymbol{Z}(t) = \boldsymbol{Z}_c] \\
=&\sum_{k=1}^{K}[w_k(t+1) - w_k(t)] + \sum_{k=1}^{K}[q_k(t+1) - q_k(t)] \\
\leq& 0 + 2 = 2
\end{aligned} \tag{9}$$

Based on the above analysis, we have proved that our Markov chain is irreducible and positive recurrent. In this case, we can claim that the chain has a stationary distribution. ■

### C. Privacy Constrained Stochastic Task Scheduling Modeling

Next, let's take a look at how to relate the privacy constraint to our constructed pairwise Markov chain. Previously, we have listed four situations that do not violate the privacy constraint. These four situations can be ensured through carefully designing transition probabilities in our constructed pairwise Markov chain. For a network with totally $K$ nodes, we will have $(K-1)K/2$ pairs, which is denoted as $(v_i, v_j)$ in this section. From the perspective of a given node pair $(v_{i_1}, v_{i_2})$ and for each state $\boldsymbol{Z}(t)$, we will assign a probability for an assignment to node pair $(v_i, v_j)$, denoted as $P^{i,j}_{\boldsymbol{Z}(t)}$. Note that as has been discussed before, $i = i_1$ or $j = i_2$ actually correspond to the cases that image portions will be processed by the current nodes. We can construct relationships between these probabilities and transition probabilities to ensure the privacy constraint. Here, two cases are discussed in detail for relationship construction between those two probabilities. Similar discussions can be done from the perspective of any given node pair.

**Case 1:** In this case, we will discuss all the states in our constructed Markov chain whose elements in the working status vector are all not zero. This means that all edge nodes are in service. In other words, when our Markov chain arrives at the state $\boldsymbol{Z}(t) = ((w_k(t) \neq 0)_{K\times 1}, (q_k(t))_{K\times 1})$, it can only transit to the state $\boldsymbol{Z}(t+1) = ((w_k(t)-1)_{K\times 1}, (q_k(t) + \Delta q_k)_{K\times 1})$. Here, $\Delta q_k$ can be 0 or 1. Value 0 means that there is no new task assigned to node $v_k$ in the current time slot, while value 1 means that there is an assigned task to $v_k$. In this case, we should have:

$$Pr\{\boldsymbol{Z}(t+1)|\boldsymbol{Z}(t)\} = \begin{cases} \lambda_{i_1,i_2} P^{i,j}_{\boldsymbol{Z}(t)} & \Delta q_i = \Delta q_j = 1 \\ 1 - \lambda_{i_1,i_2} & \Delta q_k = 0, \forall k \end{cases} \tag{10}$$

It is easy to imagine that all $\Delta q_k = 0$ represents there are no newly arrived tasks.

**Case 2:** In this case, we will discuss all the states in our Markov chain who have elements with value 0 in the working status vector. This means that some edge nodes are idle, and can process their next queued tasks. Then, we should have $\boldsymbol{Z}(t+1) = ((w_k(t)+\Delta w_k)_{K\times 1}, (q_k(t)+\Delta q_k)_{K\times 1})$. $\Delta w_k$ can be $-1, 0, N$, and $\Delta q_k$ can be $-1, 0, 1$. Each combination of $\Delta w_k$ and $\Delta q_k$ corresponds to a subsequent system state. The value of $\Delta w_k$ and $\Delta q_k$ are determined by the arrival process, working status of each edge node and task assignment strategy. When $w_k(t) \neq 0$, the corresponding edge node is in service, and Case 2 will get simplified to Case 1. If the edge node is idle ($w_k(t) = 0$), $\Delta w_k$ can be 0 or $N$. It is 0 when the edge node has processed all the queued tasks and has no newly assigned task. In this situation, $\Delta q_k$ can only be 0. Otherwise, $\Delta w_k$ will be $N$. As for $\Delta q_k$, its value should be -1 if there is no newly arrived task, and 0 with a newly arrived task.

After the above discussion about possible combinations of $\Delta w_k$ and $\Delta q_k$, it is the time to consider those transition probabilities. Before that, we further summarize those possible combinations into five categories. Each combination is denoted as $c_i$, with $i = 1, 2, \cdots, 5$. In detail, $c_1 = (\Delta q_k = 1, \Delta w_k = -1)$, $c_2 = (\Delta q_k = 0, \Delta w_k = N)$, $c_3 = (\Delta q_k = 0, \Delta w_k = 0)$, $c_4 = (\Delta q_k = 0, \Delta w_k = -1)$, and $c_5 = (\Delta q_k = -1, \Delta w_k = N)$. $c_1$ and $c_2$ correspond to the situations that there is a newly assigned task, and we use a set $C_Y$ to include them. $c_3$, $c_4$ and $c_5$ correspond to the situations that there is no newly assigned task, which is included by the set $C_N$. Then, the transition probability can be described as follows:

$$Pr\{\boldsymbol{Z}(t+1)|\boldsymbol{Z}(t)\} = \begin{cases} \lambda_{i_1,i_2} P^{i,j}_{\boldsymbol{Z}(t)} & (\Delta q_i, \Delta w_i) \in C_Y \\ & (\Delta q_j, \Delta w_j) \in C_Y \\ 1 - \lambda_{i_1,i_2} & (\Delta q_k, \Delta w_k) \in C_N, \forall k \end{cases} \tag{11}$$

Based on the above discussion, we can construct relationships between $P^{i,j}_{\boldsymbol{Z}(t)}$ and transition probabilities. The parameters to be tuned are actually $P^{i,j}_{\boldsymbol{Z}(t)}$.

### D. Optimization Problem Modeling

Next, we will talk about how to complete constructing our optimization model in detail. Recall that in Section II, we just gave a general idea of the proportion $P_{i_1}$ and $P_{i_2}$ ($P_i$ in Section II), the privacy constraint, and parameters to be tuned for our optimization problem. In Section III-A, we have shown that $P_{i_1} = P_{i_2}$, and we just need to consider the HOL delay summation from the perspective of one of those two nodes $v_{i_1}$ and $v_{i_2}$. In this section, $v_{i_1}$ is chosen in our objective function. Besides, in the last section, we constructed our pairwise Markov chain for privacy constraint enforcement, and defined parameters to be tuned as $P^{i,j}_{\boldsymbol{Z}(t)}$. In this case, our optimization problem can be rewritten as (12). For simplicity, $Pr\{\boldsymbol{Z}(t+1)|\boldsymbol{Z}(t)\}$ is represented by $Pr_{z_1,z}$. The stationary distribution is described as $Dist_z$.

$$\begin{aligned} \min_{P^{i,j}_{\boldsymbol{Z}(t)}} \quad & T_{i_1} = d_{i_1,1} + d_{i_1,2} \\ \text{s.t.} \quad & \sum_{z\in S} Pr_{z_1,z} \cdot Dist_z = Dist_{z_1}, \forall z_1 \in S \\ & \sum_{z\in S} Dist_z = 1 \\ & P^{i,j}_{\boldsymbol{Z}(t)} \geq 0 \end{aligned} \tag{12}$$

Here, the first two constraints represent stationary state equations of our Markov chain. The relationship between $P^{i,j}_{\boldsymbol{Z}(t)}$ and $Pr_{z_1,z}$ is described in equation (10) and equation (11). With our Markov chain, $d_{i_1,k}$ can be calculated through the extension of equation (1) as follows:

$$\begin{aligned} d_{i_1,k} &= \frac{1}{\alpha_{i_1,k}} \sum_{n=0}^{M} n \cdot Pr\{q_{i_1,k} = n\} \\ &= \frac{1}{\alpha_{i_1,k}} \sum_{n=0}^{M} n \sum_{\substack{z\in S \\ q_{i_1,k}=n}} Dist_z \end{aligned} \tag{13}$$

Here, $M$ corresponds to the storage limit of each server. Recall that when $k = i_1$, $\alpha_{i_1,i_1} = \alpha_{i_1,1}$, $q_{i_1,i_1} = q_{i_1,1}$ and $d_{i_1,i_1} = d_{i_1,1}$, while for all $k \neq i_1$, $\alpha_{i_1,k} = \alpha_{i_1,2}$, $q_{i_1,k} = q_{i_1,2}$ and $d_{i_1,k} = d_{i_1,2}$.

If we can solve the optimization problem described in (12), the proportion $P_{i_1}$ and $P_{i_2}$ can be calculated with those $P^{i,j}_{\boldsymbol{Z}(t)}$. In detail, we can firstly calculate the node pair proportion $P_{i_1,i_2}$ based on (14). Then, $P_{i_1}$ and $P_{i_2}$ can be derived based on the relationship described in (5).

$$P_{i_1,i_2} = \frac{\sum_{z\in S} Dist_z \cdot \sum_{z_1\in S_1} P^{i,j}_z}{\sum_{z\in S} Dist_z \cdot \sum_{z_1\in S_2} P^{i,j}_z} \tag{14}$$

Here, $\boldsymbol{Z}(t)$ and $\boldsymbol{Z}(t+1)$ are respectively simplified to $z$ and $z_1$. Given any system state $z \in S$, $S_1$ indicates a portion of subsequent system states of $z$, These subsequent system states correspond to the case that $i = i_1$ and $j = i_2$. In other words, the newly arrived tasks are decided to be processed by the current node pair. $S_2$ corresponds to the set of all possible subsequent system states of $z$.

## IV. Optimization Problem Solving

As has been discussed in the last section, we need to firstly solve the optimization problem described in (12) in order to derive the proportion $P_{i_1}$ and $P_{i_2}$. The optimization model described in (12) can theoretically be solved. However, considering its nonlinearity, it is not computationally practical. In this section, we will give a full description of how to transform our optimization problem to a linear equivalent form, and solve it through linear programming.

Firstly, we let $X^{i,j}_z = Dist_z \cdot P^{i,j}_z$. Note that $P^{i,j}_{\boldsymbol{Z}(t)}$ is simplified to $P^{i,j}_z$ here. Since $\sum_{i,j=1,i\neq j}^{K} P^{i,j}_z = 1$, we should have $\sum_{i,j=1,i\neq j}^{K} X^{i,j}_z = Dist_z$ correspondingly. After that, the original optimization model is considered together with equation (5), (11), (13) and (14), and can be transformed to the following form:

$$\begin{aligned} \min_{X^{i_k,j}_z, P_{i_1,i_2}} \quad & T_{i_1} = d_{i_1,1} + d_{i_1,2} \\ \text{s.t.} \quad & C_1(X^{i,j}_z, P_{i_1,i_2}) = 0 \\ & C_2(X^{i,j}_{z_1}) = 0, \forall z_1 \in S \\ & \sum_{z\in S} \sum_{i,j=1,i\neq j}^{K} X^{i,j}_z = 1 \\ & X^{i,j}_z \geq 0 \end{aligned} \tag{15}$$

Where $d_{i_1,1}$ and $d_{i_1,1}$ can be described by $X^{i_k,j}_z, P_{i_1,i_2}$ with the following two formulas:

$$\begin{aligned} d_{i_1,1} =& \frac{K+1}{\alpha_{i_1}[2+(K-1)P_{i_1,i_2}]} \sum_{n=0}^{M} n \sum_{\substack{z\in S \\ q_{i_1,1}=n}} \sum_{i,j=1,i\neq j}^{K} X^{i,j}_z \\ d_{i_1,2} =& \frac{K+1}{\alpha_{i_1}(1-P_{i_1,i_2})} \sum_{n=0}^{M} n \sum_{\substack{z\in S \\ q_{i_1,2}=n}} \sum_{i,j=1,i\neq j}^{K} X^{i,j}_z \end{aligned} \tag{16}$$

The first and second constraint respectively come from equation (14) and the first stationary state equation of our Markov chain. In detail, we have:

$$\begin{aligned} C_1(X^{i,j}_z, P_{i_1,i_2}) =& \sum_{z\in S} \sum_{z_1\in S_1} X^{i,j}_z - P_{i_1,i_2} \sum_{z\in S} \sum_{z_1\in S_2} X^{i,j}_z \\ C_2(X^{i,j}_{z_1}) =& \sum_{z\in S} \sum_{z_1\in S_3} \alpha_{i_1} X^{i,j}_z \\ &+ \sum_{z\in S} \sum_{z_1\in S_4} (1-\alpha_{i_1}) \sum_{i,j=1,i\neq j}^{K} X^{i,j}_z - \sum_{i,j=1,i\neq j}^{K} X^{i,j}_{z_1} \end{aligned} \tag{17}$$

Here, $S_3$ and $S_4$ are respectively the set of subsequent states with and without newly arrived tasks. With (15), (16) and (17), the original optimization problem is transformed to an equivalent linear programming problem. We can use a one-dimensional search algorithm proposed in [4] to solve it. The

optimal set of $X_z^{i,j}$, denoted as $X_p$, will be firstly obtained for each given $P_{i_1,i_2} \in [0,1]$. Then, we will conduct a horizontal comparison for all combinations of the optimal set $X_p$ and given $P_{i_1,i_2}$ to find the optimal $P^*$ and the corresponding ultimate optimal set $X^*$. Finally, all transition probabilities can be calculated through $P_z^{i,j} = X_z^{i,j} \cdot (\sum_{i,j=1,i\neq j}^{K} X_z^{i,j})^{-1}$ and equation (11).

## V. Simulation and Evaluation

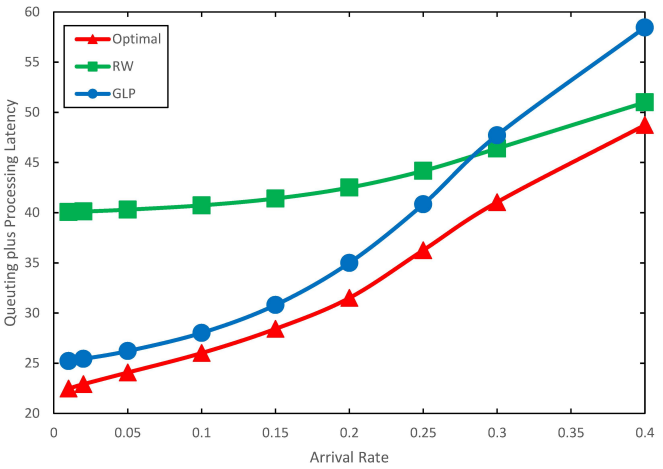
In this section, we will give a full analysis for the efficiency of our proposed stochastic task scheduling strategy. Our scheme is compared with two baselines:

- Random Walk (RW): Each idle pair receiving two portions of the same image will randomly select another pair, which does not include themselves, and transmit those two portions to that pair for processing. This case actually corresponds to the proportion for local processing $P_{i_1} = P_{i_2} = 0$.
- Greedy Local Processing (GLP): Each idle pair receiving two portions of the same image will choose to process those two portions themselves anytime. This case actually corresponds to the proportion for local processing $P_{i_1} = P_{i_2} = 1$.

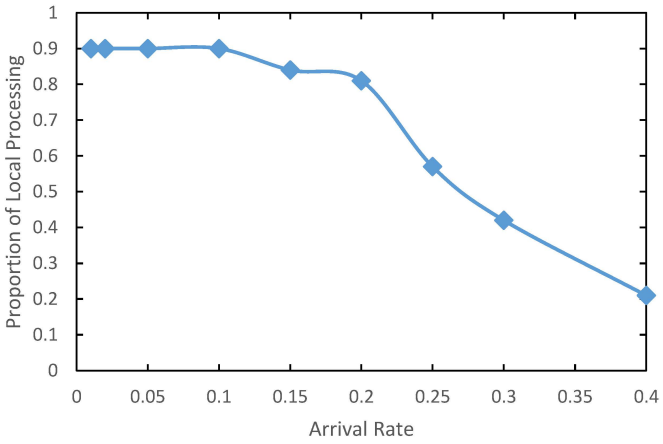
In the simulation, we consider an edge with six nodes. The largest possible number of queued tasks $M$ is set to be 5. As has been mentioned in Section II-A, the processing latency $N$ for all tasks in this paper can be considered the same. Here, we assume that $N = 20$ time slots. Besides, we respectively add a bias of 100 time slots to reflect possible node and link corruptions. Such biases are randomly applied to the whole network. Note that all the simulation results here are from the perspective of one single node $v_{i_1}$.

Fig. 5(a) shows the comparison among our task scheduling strategy, RW and GLP. From this figure, we can see that as the arrival rate grows, all of the three task scheduling strategies have longer latency, which is consistent to our intuition. Overall, our task scheduling strategy has the most optimal result. When the arrival rate is low, GLP almost has the same latency as our task scheduling strategy. This is because at this time, edge nodes usually do not have queued tasks. In this case, tasks are better to be served locally as soon as they reach the edge nodes. Transmission to other nodes can meet additional link corruptions besides node corruptions. When the arrival rate rises, GLP begins to achieve performance similar to our task scheduling strategy. This indicates that passing tasks to other nodes is proper for a higher arrival rate. Finally, Fig. 5(b) shows the optimal proportion of local processing $P_{i_1}$ for different arrival rates. From this figure, we can notice that the proportion $P_{i_1}$ is not exactly 1, 0.9 actually. The reason is that node corruptions will cause queued tasks. When tasks are queued in one node, it is better to distribute them to other nodes to fully utilize computation resources of the whole network.

Fig. 6 shows the optimal distribution of queue length for a particular edge node in different arrival rates. From this figure, we can see that when the arrival rate is low, the case that there are no queued tasks is extremely likely to happen. With the arrival rate rising, probability of the edge node having no




(a)




(b)

Fig. 5. Efficiency evaluation results. (a) Queuing plus processing latency vs. arrival rate. (b) Optimal proportion for local processing vs. arrival rate

queued tasks decreases significantly. This is consistent to our observation above that when the arrival rate rises, the total latency also increases. Since processing time for all the tasks is assumed to be equal, the extra more latency can only be attributed to longer queuing time, which results from longer queues.

## VI. Discussion

In this section, we want to discuss briefly about potential improvements and extensions of our privacy-preserving scheduling policy. In this paper. we assume that the attacker can only eavesdrop one edge node. In the future, the case with multiple nodes getting hacked is worth considering. Even worse, all edge nodes have the possibility to be owned by the same entity and all get hacked. The case with all nodes getting attacked may not be solved properly just by privacy-preserving scheduling. However, the case with some nodes getting hacked can still be considered. In addition, schedulers may also be attacked. It is also interesting to look into the case that schedulers act maliciously.

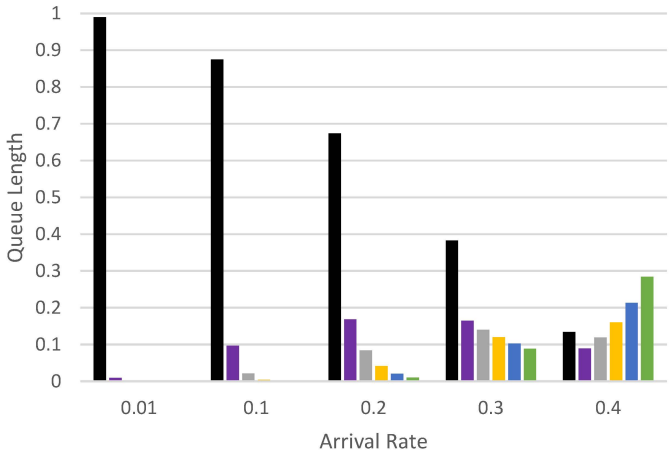



Fig. 6. Distribution of Queue Length in Different Arrival Rates

The extension to the case that the edge network has a multihop architecture is also interesting. According to [17], the linear relationship between queue length and HOL delay does not necessarily hold in multihop networks. The relay issue will cause the arrival process to change. In this case, our *2-hop* process cannot be directly extended to the multihop case. Besides the newly arrived tasks from outside of the network, the queue of one node should also contain tasks transmitted from other nodes in the network. Furthermore, communication latency and throughput constraint should also be considered. Backpressure scheduling policies [13], [18] could be a solution to these issues.

## VII. Related Works

Edge computing helps improve latency performance by processing information on the edge nodes near users. It can also reduce the bandwidth consumption in the core network by preprocessing or aggregating information at edge before sending them to the cloud. Under this new computing paradigm, a well-designed task scheduling strategy is needed to decide which computing task to be assigned to which edge node. Up to now, several resource allocation and task scheduling works [1]–[6] have been done for edge computing systems with different configurations. However, none of them considered privacy issues.

SIFT algorithm [7] has been widely used in image matching owing to its robustness. Since SIFT requires a large number of convolution computations, many cloud computing solutions have been proposed for it. With the concern of preserving image privacy against untrusted third party cloud servers, privacy-preserving SIFT schemes have been proposed. [8]–[10] propose to utilize homomorphic encryption to satisfy privacy requirements. Later, [11] claims that relying on just one server cannot well guarantee privacy, and [11], [12] propose to randomly split images for encryption and transmit these two portions to two different remote servers for privacy-preserving SIFT processing. In this paper, we propose an edge computing based deployment for such a privacy-preserving application. During the whole process, it should be guaranteed that no edge nodes are assigned two portions of the same image. Therefore, a task scheduling strategy with such a privacy constraint needs to be designed.

A Markov-chain-based task scheduling strategy is proposed in this paper. The privacy constraint is guaranteed by paring the corresponding edge nodes. The achievability of our Markov chain is proved by irreducibility and positive recurrence [15]. Irreducibility is ensured by the structure of our Markov chain, and positive recurrence is claimed through Foster-Lyapunov theorem [16].

## VIII. Conclusion

In this paper, we propose a stochastic task scheduling strategy based on a privacy-preserving SIFT algorithm on edge. Images are randomly split into two portions for encryption and transmitted to two edge nodes for processing. During the whole process, all edge nodes should not have the chance to get both two portions of the same image. In order to guarantee such a privacy constraint, we construct a pairwise Markov chain to take care of it. Edge nodes are considered in pairs in all system states of our chain. The achievability can be proved through showing irreducibility and positive recurrence. Queuing plus processing latency can be minimized based on our constructed pairwise Markov chain. Simulation results validate that our task scheduling strategy can achieve minimal latency, and guarantee the privacy constraint at the same time.

## References

[1] Z. Jiang and S. Mao, "Energy delay trade-off in cloud offloading for mutli-core mobile devices," in *Proc. IEEE Global Communications Conference (GLOBECOM'15)*, San Diego, CA USA, Dec. 2015, pp. 2306–2316.

[2] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J-SAC*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[3] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE TSIPN*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[4] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE International Symposium on Information Theory (ISIT'16)*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[5] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE TWC*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[6] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *Proc. IEEE International Conference on Computer Communications (INFOCOM'18)*, Honolulu, HI USA, Apr. 2018.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[8] W. Lu, A. L. Varna, A. Swaminathan, and M. Wu, "Secure image retrieval through feature protection," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'09)*, Taipei, Taiwan, Apr. 2009.

[9] C.-Y. Hsu, C.-S. Lu, and S.-C. Pei, "Image feature extraction in encrypted domain with privacy-preserving SIFT," *IEEE Trans. Image Processing*, vol. 21, no. 11, pp. 4593–4607, Nov. 2012.

[10] S. Wang, M. Nassar, M. Atallah, and Q. Malluhi, "Secure and private outsourcing of shape-based feature extraction," in *Proc. International Conference on Information and Communications Security (ICICS'13)*, 2013, pp. 90–99.

[11] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. ACM International Conference on Multimedia (MM'14)*, Nov. 2014, pp. 497–506.

[12] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Trans. Image Processing*, vol. 25, no. 7, pp. 3411–3425, May 2016.

[13] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in *Proc. IEEE ICCCN'96*, 1996.

[14] S. M. Ross, *Introduction to Probability Models*. Oxford, UK: Academic Press, 2014.

[15] R. Serfozo, *Basics of Applied Stochastic Processes*. Berlin, Germany: Springer, 2009.

[16] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge, UK: Cambridge University Press, 2014.

[17] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *IEEE/ACM TON*, vol. 21, no. 5, pp. 1593–1552, Oct. 2013.

[18] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.