Coding for Distributed Fog Computing in Internet of Mobile Things

Jing Yue, Member, IEEE and Ming Xiao, Senior Member, IEEE

Abstract—Internet of Mobile Things (IoMTs) refers to the interconnection of mobile devices, for example, mobile phones, vehicles, robots, etc. For mobile data, strong extra processing resources are normally required due to the limited physical resources of the mobile devices in IoMTs. Due to latency or bandwidth limitations, it may be infeasible to transfer a large amounts of mobile data to remote server for processing. Thus, distributed computing is one of the potential solutions to overcome these limitations. We consider the device mobility in IoMTs. Two situations of the movement position of the mobile devices, i.e., unpredictable and predictable, are considered. In addition, three possible relative positions between the two server sets which respectively correspond to the positions of a mobile device for computation tasks offloading and for output results receiving, i.e., within the same server set, with two different server sets and with two adjacent server sets, are studied. Coded schemes with high flexibility and low complexity are proposed based on Fountain codes to reduce the total processing time and latency of the distributed fog computing process in IoMTs for the above different situations. The latency related performance, i.e., the computation, the communication and the transmission loads, is analyzed. We also compare of the Fountain code-based and the unencoded schemes and numerical results demonstrate that shorter total processing time and lower latency can be achieved by the Fountain code-based schemes.

Index Terms—Internet of mobile things, distributed fog computing, MapReduce, fountain codes, processing time and latency

1 INTRODUCTION

The number of connected mobile devices has increased with the development of communication and computing technologies at an unprecedented rate [1]. The connected mobile devices form a cyber-physical infrastructure, namely, the Internet of Mobile Things (IoMTs). Different from the Internet of Things (IoTs), device mobility is one of the most critical properties in IoMTs [2]. Mobility leads to the challenges in mobile data collection and analysis, resource and energy management, security and privacy protection, etc.

The potential application scenarios of IoMTs include healthcare, smart transportation, industrial automation, etc. In healthcare, mobile devices are exploited to collect the patient health-related parameters, e.g., brain activity, heartbeat, breath, position, etc. According to the analysis results of the collected data, IoMT may send a mild electric shock or make an emergency call, etc. For smart transportation, the driving status of the vehicular and the traffic information are collected by the embedded or wireless connected sensors. Decisions are made in real time at the control center according to the analysis results to manage and control the traffic. For industrial automation, the operating status of the industrial devices are sensed. The collected data is processed at servers.

Control information is sent back from servers to the actuators to adjust their operations.

In order to provide real-time services to the above time-critical applications, a great amount of data need to be processed timely. However in many scenarios, data processing cannot be performed by the mobile devices themselves because of their limited physical resources. Fog computing [3] is a potential solution to support the resource-constrained mobile devices [1]. In fog computing, data processing is distributed and close to the devices, and thus the high reliability and low latency [4] are achieved in time-critical applications.

In fog computing, the mobile devices offload their computing tasks into the fog network and exploit its powerful resources. Many works have been reported on task offloading and server migration [5], [6], [7], [8] in IoMTs. For data offloading, offloading decisions, e.g., whether, where and when to offload tasks, are made at the mobile devices according to the situation of their own resources and the computing resources of the servers available for them. The server migration problem is modeled as Markov decision process (MDP) in many researches. Migration decisions, e.g., when and where to migrate, are made at the servers to provide seamless services to the mobile devices. There are also some works focus on the challenges of distributed fog computing process, e.g., the problems that arise from straggling servers and multistage computations [9], [10], [11].

Recently, coding schemes were proposed for fog computing, i.e., minimum bandwidth codes [11], [12], minimum latency codes [13], [14], and the unified coding framework [10], [15]. These coded schemes were designed for solving the problem of straggling servers. Two extensions of the coded distributed computing framework were designed in [9] aiming to solve the problem of multistage computations in distributed
We consider the mobility of devices in IoMTs. Three near-by servers. A server is $M$.

Finally, we give comparison results to show that by applying the Fountain code-based schemes, both the communication and the transmission loads can be reduced with relatively low complexity. The trade-off between the computation, the communication and the transmission loads is studied. Compared with the uncoded scheme, shorter processing time and lower latency for the distributed fog computing process can be achieved by the Fountain code-based schemes. Compared with the MDS code-based scheme, the Fountain code-based scheme has much lower coding complexity and is more flexible to varying scenarios.

Different from the previous coded distributed computing schemes, we consider more practical scenarios where the computation tasks, e.g., the data in datasets, are generated continuously with time. That is, the dataset is not available at the time of accepting the offloading request from the mobile device. For the nonlinear problems with higher complexity, the MDS-based coding schemes become impractical because of high computation and communication costs associated with encoding and decoding procedures. Thus, coded schemes with high flexibility and low complexity are needed to reduce the overall processing time of the distributed fog computing process in time-critical application scenarios.

It is provided in [25, 26] that the Fountain code-based scheme for distributed computing is one of the optimal choices for the applications with high reliability and low latency requirements. In [25], a Fountain code-based scheme was proposed to confront the problem of straggling servers. Fountain codes are capable of recovering the original information from any subset of output symbols with size slightly larger than the original information. In addition, Fountain codes are of low coding complexity. A coded scheme based on batched sparse (BATS) codes, which are enhanced rateless sparse random linear network codes, was proposed in [29] for the balance of communication and computation loads of the distributed fog computing process. Coded mobile edge computing architectures were proposed in [30] for offloading computation requests and reducing communication load from servers to mobile devices. However, to our best knowledge, coded distributed fog computing for IoMTs considering communications both between mobile devices and servers and among servers has not been extensively studied yet.

In this paper, we consider Fountain codes [31] for the code design of distributed fog computing in IoMTs. Different from [25], the mobility of the mobile devices and the time-varying channels in IoMTs are considered. In particular, we focus on the study of the computation task assign, the intermediate result exchange, the reduce task assign, and the output result delivery, for distributed fog computing in IoMTs. Our main contributions are listed as follows:

- We consider the mobility of devices in IoMTs. Three possible situations of the relative positions between the two server sets, which respectively correspond to the computation tasks offloading and the output results receiving positions of a mobile device, are studied. The situations of unpredictable and predictable movement position and the networks with error-free and erasure channels are considered. We propose high flexibility and low complexity coded schemes based on Fountain codes for the above three possible relative positions to reduce the total processing time and latency of the distributed fog computing process in IoMTs under different situations and networks.
- We calculate the total processing time of the distributed fog computing process. The performance related to the processing time and latency, i.e., the computation load at each fog node, the communication load for exchanging intermediate results, and the transmission load for delivering the output results among fog nodes, is analyzed for the uncoded and the proposed Fountain code-based schemes, respectively.
- Finally, we give comparison results to show that by applying the Fountain code-based schemes, both the communication and the transmission loads can be reduced with relatively low complexity. The trade-off between the computation, the communication and the transmission loads is studied. Compared with the uncoded scheme, shorter processing time and lower latency for the distributed fog computing process can be achieved by the Fountain code-based schemes. Compared with the MDS code-based scheme, the Fountain code-based scheme has much lower coding complexity and is more flexible to varying scenarios.

Different from the previous coded distributed computing schemes, we consider more practical scenarios where the computation tasks, e.g., the data in datasets, are generated continuously with time. That is, the dataset is not available for any predetermined operation before the distributed computing starts, e.g., computation task assign.

The remainder of the paper is organized as follows. The device mobility and the MapReduce model for distributed fog computing in IoMTs are introduced in Section 2. Then, Fountain code-based schemes for distributed fog computing in IoMTs are proposed in Section 3. Also in Section 3, the total processing time is calculated. After that, the latency related performance of the uncoded and the Fountain code-based schemes for the situations of unpredictable and predictable movement position is analyzed in Sections 4 and 5, respectively. Comparison and numerical results are given in Section 6. Finally, the conclusions are drawn in Section 7.

## 2 System Model

We consider the mobility of devices in IoMTs. Assume that the geographical distributions of both mobile devices and servers (also known as fog nodes) follow an independent homogeneous poisson point process (HPPP) [36, 37]. A mobile device may observe $F_N$ nearby servers. A server is only accessible to the mobile device if a wireless connection can be established. That is, the mobile device can offload its data to the accessible servers for processing and the offloading decision can be made. The servers, which make the decision of accepting the offloading request from the mobile device, are studied in our future work.

1. In this work, we only consider linear Map functions and focus on the study of coded distributed computing for providing seamless service for mobile devices. For the nonlinear problems with higher complexity in many practical applications, there are mainly two methods to handle nonlinear problems in coded distributed computing. One is to calculate a number of independent functions or encoded dataset in parallel and aggregate the computation results to obtain the result of the corresponding nonlinear problem [32, 33, 34], and another one is to apply machine learning to train the encoding and decoding models for coded distributed computing design [35]. A further study of distributed computing for nonlinear problems is beyond the scope of our paper and will be studied in our future work.
device, are called available servers for the mobile device. The processes of decision making for offloading at the mobile device and for accepting at the servers are beyond the scope of this paper. Here, we only take the effect of these decision processes into consideration through parameter $\lambda_f$, i.e., the distribution density of available servers (fog nodes).

Without loss of generality, we focus on the study of an arbitrary mobile device in IoMTs. Let $R$ denote the bidirectional communication distance between the mobile device and servers (fog nodes). The wireless connection area for a mobile device and its moving route are shown in Fig. 1. We use $F$ to represent the number of available servers for the mobile device to offload its data for processing. Based on the HPPP assumption for the geographical distribution of servers, the probability that the device can access $F$ servers in the current decision period can be calculated by the following probability mass function (PMF) of HPPP [5], [37]

$$P_V(F) = \frac{\lambda_f^F F^F}{F!} e^{-\lambda_f F}, \quad F = \{0, 1, 2, \ldots, F_N\},$$

where $F \in \{0, 1, 2, \ldots, F_N\}$. The error becomes arbitrarily small when $F_N$ is large.

Among the existing schemes for distributed fog computing, MapReduce [38] is the commonly used framework [39]. As shown in Fig. 2, a distributed fog computing process contains three stages, namely, Map, Data Shuffling and Reduce stages. In Map stage, distributed fog nodes process the computation tasks and generate intermediate results according to their designed Map functions. Next in Data Shuffling stage, the calculated intermediate results are exchanged among fog nodes. Then, in Reduce stage, the final output results are calculated distributively at fog nodes using their designed Reduce functions [9]. The output results are finally sent back to the devices.

We assume the networks with error-free and erasure channels. Here, error-free networks mean that the links between the mobile devices and the fog nodes, and among the fog nodes, are lossless. Erasure networks mean that data may be lost during transmissions. We use $\epsilon_{DF}$, $\epsilon_{EF}$ and $\epsilon_{FD}$ to represent the erasure probabilities of the links from the mobile devices to the fog nodes, among the fog nodes, and from the fog nodes to the mobile devices, respectively. Then we have $\epsilon_{DF} = \epsilon_{EF} = \epsilon_{FD} = 0$ for error-free networks and $\epsilon_{DF}, \epsilon_{EF}, \epsilon_{FD} > 0$ for erasure networks. The connections among nodes in the networks are changing with time. We note that for erasure networks, the underlining physical layer channel can be e.g., Rayleigh fading channels and error detection is used. That is, the network topologies are time-varying. Therefore, any predetermined operation, such as coding and task assign, etc., is impractical.

We define the computation load as the total number of fog nodes used for processing one common computation task. Communication load is defined as the ratio between the actual number of intermediate results exchanged and the total number of intermediate results needed at all the fog nodes. Transmission load is defined as the ratio between the actual number of delivered output results among fog nodes and the total number of output results required at all the fog nodes. Uncoded scheme means that no coding operation is performed neither on the computation tasks at mobile devices in Map stage nor on the output results at each fog node in Output Delivery stage.

We list the main parameters in an abbreviation table, i.e., Table 1, for reading convenience.

### 3 Fountain Code-Based Schemes for Fog Computing in IoMTs

All the available servers (fog nodes) in the wireless connection area of a mobile device form a set, called server set. We use $\Omega_s$ and $\Omega_r$ to represent the two server sets, which respectively correspond to the positions of a mobile device for computation task offloading and for output result receiving. In this section, three possible relative positions of the two server sets will be studied. We consider two different situations for the movement position of the mobile device, i.e., unpredictable and predictable. Coded schemes based on Fountain codes will be designed for the above different situations to reduce both the communication and the transmission loads of the distributed fog computing process in IoMTs. Finally, the total processing time will be calculated.

#### 3.1 Relative Positions

For the two server sets $\Omega_s$ and $\Omega_r$, their three possible relative positions are shown in Fig. 3. The three relative
positions are referred to within the same server set \((\Omega_o = \Omega_r)\), with two different server sets \((\Omega_o \cap \Omega_r = \phi)\), and with two adjacent server sets \((\Omega_o \neq \Omega_r \text{ and } \Omega_o \cap \Omega_r = \phi)\), respectively. Here \(\phi\) represents empty set. The situation \(\Omega_o = \Omega_r\) is formed due to circle moving of the mobile device or very low moving speed, etc. In the beginning, the mobile device offloads its computation tasks to servers in set \(\Omega_o\). After time \(t\), the mobile device starts to receive output results from servers (fog nodes) in set \(\Omega_r\). The servers in \(\Omega_o\) and \(\Omega_r\) are all the same, i.e., \(\Omega_o = \Omega_r\) as shown in Fig. 3a. The situation \(\Omega_o \cap \Omega_r = \phi\) is formed due to a relatively high moving speed of the mobile device. The servers in set \(\Omega_o\) are completely different from set \(\Omega_r\) after time \(t\), as shown in the Fig. 3b. For the situation \(\Omega_o \neq \Omega_r\) and \(\Omega_o \cap \Omega_r = \phi\), the mobile device moves to a adjacent server set after a duration of time \(t\). In this situation, some of the servers (fog nodes) locate in both the two adjacent server sets \(\Omega_o\) and \(\Omega_r\), simultaneously, as shown in Fig. 3c.

### 3.2 Fountain Code-Based Schemes for Fog Computing

Due to the mobility of devices and the changes of relative positions of server sets in IoMTs, an additional stage for the coded scheme design for distributed fog computing, i.e., Output Delivery stage, should be taken into consideration. Therefore, the novel coded schemes we will design for IoMTs are composed of four stages, i.e., Map, Data Shuffling, Reduce and Output Delivery stages. The goal of the coded scheme design is to achieve shorter total processing time and lower latency of the distributed fog computing process in IoMTs through reducing the communication and the transmission loads. We consider the two situations for the movement position of a mobile device, i.e., unpredictable and predictable. In the following, Fountain code-based schemes are proposed for the above two different situations, respectively. Without loss of generally, we take the process in an arbitrary fog computing round as an example to illustrate the coded schemes. A fog computing round includes the processes of computation task assign at each mobile device in Map stage, computation task process at each fog node, intermediate result exchange among fog nodes in Data Shuffling stage, reduce task assign and process at each fog node in Reduce stage, output result delivery among fog nodes in Output Delivery stage, and output result send back from fog nodes to mobile devices. As shown in Fig. 4, is a block diagram to illustrate the entire computation process.

| Parameters | Meaning | Value |
|------------|---------|-------|
| \( F_N \)  | The number of nearby servers (fog nodes) of a mobile device | \( F_N \geq 0 \) |
| \( F \)   | The number of available servers for a mobile device | \( 0 \leq F \leq F_N \) |
| \( R \)   | The bidirectional communication distance between a mobile device and servers | \( R \geq 0 \) |
| \( \lambda_f \) | The distribution density of available servers | |
| \( \Omega_o \) | The server set corresponds to the position of a mobile device for computation task offloading | |
| \( \Omega_r \) | The server set corresponds to the position of a mobile device for output result receiving | |
| \( K \) | The number of computation tasks at a mobile device | \( K \geq 0 \) |
| \( N \) | The number of coded computation tasks at a mobile device | \( N \geq 0 \) |
| \( T_I \) | The length of each coded computation task (The length of each intermediate result) | \( T_I \geq 0 \) bits |
| \( T_o \) | The length of each output | \( T_o \geq 0 \) bits |
| \( \eta_1 \) | The expending coefficient of the used Fountain codes in Map stage | \( 1 + \eta_1 = \frac{N}{K} \eta_1 \geq 0 \) |
| \( \Psi(x) \) | The degree distribution of the used Fountain codes in Map stage | \( \Psi(x) = \sum_{i=0}^{K} \Psi_i x^{\psi_i} \) |
| \( Q \) | The number of intermediate results for each coded task | \( Q \geq 0 \) |
| \( F_I \) | The number of fog nodes responsible for reducing outputs | \( 1 \leq F_I \leq F \) |
| \( S_I \) | The number of times that each output is reduced at \( F_I \) fog nodes | \( S_I \geq 0 \) |
| \( \gamma_r \) | The average number of reduce tasks assigned to each of \( F_I \) fog nodes | |
| \( F_o \) | The number of fog nodes responsible for transmitting the output results to a mobile device | |
| \( S_o \) | The number of output results for each of the \( F_o \) fog nodes to transmit | |
| \( \eta_2 \) | The expending coefficient of the used Fountain codes in Output Delivery stage | \( \eta_2 \geq 0 \) |
| \( \mu(x) \) | The degree distribution of the used Fountain codes in Output Delivery stage | \( \mu(x) = \sum_{i=0}^{Q} \mu_i x^{\mu_i} \) |
| \( F_{o,p} \) | The number of fog nodes in both of the two adjacent server sets \( \Omega_o \) and \( \Omega_r \), simultaneously | \( F_{o,p} = \beta F_o, 0 \leq \beta \leq 1 \) |
| \( \alpha F_{o,p} \) | The number of fog nodes already have partial output results before Output Delivery stage | \( 0 \leq \alpha \leq 1 \) |
| \( d \) | The distance of the offloading and the receiving positions of the mobile device | |
| \( F_I = a_I F \) | Each of the \( F_I \) fog nodes will be assigned reduce tasks | \( 0 \leq a_I \leq 1 \) |
| \( S_I = b_I F_I \) | Each of the \( F_I \) fog nodes will be assigned \( b_I Q \) reduce tasks | \( b_I > 0 \) |
Assume that fog nodes are assigned times. Let different fog nodes. Thus, on average each of the groups. 1 fog nodes ¼ tasks are combined to form one coded computation. The same process continues on until outputs. Let reduce tasks are assigned randomly to different fog nodes. We use ð Þ to represent the expending coefficient of the used Fountain codes, where 1 ð Þ 2 and ð Þ 0. The encoding process for the computation task assign is described in the following steps.

- At first, the mobile device selects a number , with probability according to the degree distribution .
- Then, the mobile device selects computation tasks uniformly at random from the ones. The selected are combined to form one coded computation task. The same process continues on until coded computation tasks are formed at the mobile device.
- Finally, the mobile device sends the computation tasks to their respective fog nodes for processing. For the mobile device, there are available servers (fog nodes). The number is known at the mobile device. The computation tasks will be assigned to the servers for processing. The tasks assign in Map stage is as follows,
  - If , the coded computation tasks are assigned randomly to different fog nodes. Then, we set . The number of coded tasks at each of the fog node is .
  - If , let and . Here, represents floor function. At first, coded computation tasks are selected and divided into groups. In each group there are coded tasks. Then, the rest coded tasks are assigned to different groups among the ones. After that, the groups of coded tasks are assigned randomly to different fog nodes. Thus, on average each fog node is assigned coded tasks.

The computation task assign includes two steps, i.e., the encoding process for task assign and coded computation task assign. The two steps can be summarized as Algorithms 1 and 2, respectively.

We consider that all the computation tasks are with equal importance and use a classic encoding method of Fountain codes. That is, selecting a fix number of data uniformly at random from the whole dataset and combining the selected data to form a coded data.

### Algorithm 1. The Encoding Process for Task Assign

for do
  Select a number , with probability according to
  Select computation tasks uniformly at random from the ones
  Combine the selected computation tasks to form one coded computation task
end for

### Algorithm 2. Coded Computation Task Assign

if then
  Randomly assign the coded tasks to fog nodes
  Set
end if
if then
  Select coded computation tasks
  Divide the coded tasks into groups
  Assign the rest coded tasks to different groups
  Randomly assign the groups of coded tasks to fog nodes
end if

Reduce Task Assign in Reduce Stage. For each coded task, intermediate results are generated after computing. Each of the intermediate results together with the corresponding ones resulted from other tasks will be used to calculate one output. That is, intermediate results correspond to different tasks are used to calculate one output. Thus, the total number of outputs is . Let represent the number of fog nodes responsible for reducing outputs. Then, we have . Assume that each output is reduced at times. Let and . At first, each of the fog nodes are assigned different reduce tasks. Then, the rest coded tasks are assigned randomly to different fog nodes. Thus, on average each of the fog nodes is assigned different reduce tasks.

Intermediate Result Exchange in Data Shuffling Stage. For the computation of each output, intermediate results correspond to different tasks are required. Since on average, each of fog nodes is responsible for reducing outputs, and intermediate results are already known at each of the fog nodes. Thus, there are two situations for the operations at fog nodes in Data Shuffling stage.

- If a fog node is only responsible for computing intermediate results, this node will not require any intermediate result from other fog nodes in Data Shuffling stage. But this node has to transmit its intermediate results to fog nodes.
- If a fog node is responsible for both computing and reducing, this fog node not only has to transmit its intermediate results to the rest fog nodes, but also requires intermediate results from the other fog nodes to complete its reduce tasks.

Output Result Transmission in Output Delivery Stage. After the previous three stages, outputs are generated at the fog nodes. Let be the number of fog nodes responsible for transmitting the output results to the mobile device. Let be the number of output results for each of the fog nodes to
transmit. Then, we have \( Q \leq S_o F_o \leq (1 + \eta_2)Q \), where \( \eta_2 (\eta_2 \geq 0) \) is the expanding coefficient of the Fountain codes used in Output Delivery stage.

- For the situation \( \Omega_o = \Omega_r \): We set \( S_o = \gamma_o \) and \( F_o = F_I \). That is, the fog nodes, which are responsible for reducing the outputs, are also responsible for transmitting its generated output results to the mobile device.

- For the situation \( \Omega_o \cap \Omega_r = \emptyset \): At each of \( F_I \) fog nodes, \( \gamma_o \) output results are encoded to form \((1 + \eta_2)\eta\) coded output results, where \( \eta = \left( \begin{array}{c} h \left( \begin{array}{c} q \end{array} \right) \end{array} \right) \). This represents ceiling function. Then, each of the \( F_I \) fog nodes delivers its \((1 + \eta_2)\eta\) coded output results to the \( F_o \) fog nodes. Without loss of generality, we take an arbitrary one of the \( F_I \) fog nodes as an example to illustrate the encoding process.
  - At first, the fog node selects a number \( \rho_i \) with probability \( \mu_{\rho_i} \) according to the degree distribution \( \mu(x) = \sum_{i=1}^{c} \mu_{\rho_i} x^{\rho_i} \).
  - After that, the fog node selects \( \rho_i \) output results uniformly at random from the \( \gamma_o \) ones. Then, the selected \( \rho_i \) outputs are combined to form one coded output results. The same process continues on until \((1 + \eta_2)\eta\) coded output results are formed.
  - Finally, the fog node transmits its \((1 + \eta_2)\eta\) coded output results to the \( F_o \) fog nodes.

The above encoding process at each of the \( F_I \) fog nodes can be summarized as Algorithm 3.

**Algorithm 3. An Example of the Encoding Process in Output Delivery Stage**

\[
i_{flag} \leftarrow 1
\]

while \( i_{flag} \leq (1 + \eta_2)\eta \) do

Select a number \( \rho_i \) with probability \( \mu_{\rho_i} \) according to \( \mu(x) \)

Select \( \rho_i \) output results uniformly at random from the \( \gamma_o \) ones

Combine the \( \rho_i \) selected output results to form the \( i_{flag} \)th coded output result

\[
i_{flag} \leftarrow i_{flag} + 1
\]

end while

Transmit the \((1 + \eta_2)\eta\) coded output results to the \( F_o \) fog nodes

- For the situation \( \Omega_o \neq \Omega_r \) and \( \Omega_o \cap \Omega_r = \emptyset \): Assume that \( F_o = \beta F_o \) (0 ≤ \( \beta \) ≤ 1) fog nodes are located in both of the two adjacent server sets simultaneously. Among the \( F_o \) fog nodes, \( \alpha F_o \) (0 ≤ \( \alpha \) ≤ 1) ones are assigned reduce tasks in Reduce stage. Thus, the \( \alpha F_o \) fog nodes already have partial output results before Output Delivery stage starts. For each of the \( \alpha F_o \) fog nodes, \( \max\{0, S_o - \gamma_o\} \) output results are required. For the rest \( F_o = \beta F_o \) fog nodes, \( S_o \) output results are required. Each of the \( F_I \) fog nodes in server set \( \Omega_r \) has to transmit its \((1 + \eta_2)\eta\) coded output results to the \( F_o \) fog nodes in set \( \Omega_r \).

3.2.2 Predictable Movement Position

Predictable movement position means that the moving route of a mobile device is predictable and the position of the mobile device after time \( t \) is known. The four stages of the Fountain code-based scheme for the situation with predictable movement position are described as follows.

3.3 Total Processing Time and Latency

The total processing time includes the time for processing the computation tasks, exchanging intermediate results, and delivering the output results. For the coded scheme, the time for encoding should also be taken into consideration. Assume that the time for processing one computation task at each fog node is \( t_{c_o} \) for exchanging one intermediate result is \( t_{e_o} \), and for delivering one output result is \( t_{d_o} \). We use \( t_{a_o} \)
and $t_{cd}$ to represent the encoding time for computation tasks assign and for outputs delivery, respectively. Then, the total time for encoding is $t_{co} = t_{ca} + t_{cd}$.

Assume that all the fog nodes process their assigned tasks or perform encoding operations simultaneously. For the uncoded and the Fountain code-based schemes, the total processing time $T_{uncoded}$ and $T_{coded}$ can be written as follows:

$$ T_{uncoded} = \frac{rK}{F}t_e + KQL_{uncoded}(r)t_e + S_o F_o t_{uncoded}(r)t_d, $$

$$ T_{coded} = \left(1 + \eta_1\right)\frac{K}{F}t_e + KQL_{coded}(\eta_1)t_e + S_o F_o t_{coded}(\eta_2)t_d + t_{co}, $$

where $r$ is the computation load of the uncoded scheme and $\eta_1$ and $\eta_2$ represent the expanding coefficients of the Fountain codes used for computation tasks assign in Map stage and for reduce tasks assign in Reduce stage, respectively. In (3), $1 + \eta_1$ is used to represent the computation load of the Fountain code-based schemes.

The time for encoding is related with the coding parameters $\eta_1$, $\eta_2$, $\Psi(x)$ and $\mu(x)$. Generally, only simple addition and multiplication operations are needed during the encoding process. Through proper degree distribution design, the total time for encoding, i.e., $t_{co}$, can be reduced to a relatively small value. Thus, the differences in total processing time between the uncoded and the coded schemes is mainly determined by the parameters $r$, $1 + \eta_1$, $L_{uncoded}(r)$, $L_{coded}(\eta_1)$, $D_{uncoded}(r)$ and $D_{coded}(\eta_2)$, i.e., the computation, the communication and the transmission loads. Shorter total processing time means lower latency for the distributed fog computing process. Thus, lower latency can be achieved through reducing the communication and the transmission loads. For energy consumption of the system, the analysis process is similar to that for the total processing time. And thus, we expect that lower energy consumption can also be achieved through reducing the communication and the transmission loads. In the following, the latency related performances, i.e., the communication and the transmission loads, will be analyzed.

4 PERFORMANCE ANALYSIS WITH UNPREDICTABLE MOVEMENT POSITION

In this section, we consider three possible relative positions of server sets $\Omega_o$ and $\Omega_r$. The communication and the transmission loads of the fog computing process by using the uncoded and the Fountain code-based schemes with unpredictable movement position under error-free and erasure networks will be calculated, respectively.

4.1 Error-Free Networks

4.1.1 Within the Same Server Set

Theorem 1. For the uncoded scheme in error-free networks, the communication load for exchanging the intermediate results can be calculated by

$$ L_{uncoded}(r) = \frac{S_I}{F_I} \left(1 - \frac{r}{F}\right). $$

where $r$ is the computation load for the uncoded scheme and $r \in \{1, 2, \ldots, F\}$.

Proof 1. By using the uncoded scheme, to obtain one output, $K$ intermediate results correspond to $K$ different computation tasks are required. The total number of intermediate results needed is $KQT_I$. Since on average $\frac{F}{r}$ computation tasks have been assigned to each fog node for processing in Map stage, at each of the fog node, $\frac{F}{Q}Q$ intermediate results is already known. For the situation $\Omega_o = \Omega_r$ and unpredictable movement position, each of the fog node is assigned $\gamma_o$ reduce tasks. Therefore, each fog nodes requires $(K - \frac{F}{Q})\gamma_o$ intermediate results from the other fog nodes. The length of each intermediate result is $T_I$. Then, the size of data for transmission becomes $(K - \frac{F}{Q})\gamma_o T_I$. Therefore, we have the communication load $L_{uncoded}(r) = \frac{(K - \frac{F}{Q})\gamma_o T_I}{F_I} = \frac{S_I}{F_I} (1 - \frac{r}{F})$. $\Box$

Theorem 2. For the Fountain code-based scheme in error-free networks, the communication load for exchanging the intermediate results can be calculated by

$$ L_{coded}(\eta_1) = \frac{S_I}{F_I (1 + \eta_1)} \left(1 - \frac{1 + \eta_1}{F}\right), $$

where $1 + \eta_1$ is the expanding coefficient and $1 + \eta_1 = \frac{F}{k}$.

Proof 2. The same as that for the uncoded scheme, to obtain one output by using the Fountain code-based scheme, $K$ intermediate results generated from $K$ different computation tasks is required. At each of the fog node, $\gamma_o Q$ intermediate results is already known. Each fog nodes requires $(K - \tau_o)\gamma_o$ intermediate results from the other fog nodes. Since the computation tasks are encoded in Map stage, and the coding expanding coefficient is $1 + \eta_1$. Therefore, the communication load for the Fountain code-based scheme can be calculated by $L_{coded}(\eta_1) = \frac{(K - \tau_o)\gamma_o T_I}{F_I} = \frac{S_I}{F_I (1 + \eta_1)} \left(1 - \frac{1 + \eta_1}{F}\right)$. $\Box$

For the situation $\Omega_o = \Omega_r$, the transmission load for delivering the output results is 0 for both the uncoded and the coded schemes.

4.1.2 Two Different Server Sets

The communication loads for the uncoded and the Fountain code-based schemes in the situation $\Omega_o \cap \Omega_r = \phi$ are the same as that for the situation $\Omega_o = \Omega_r$. In the following, we will calculate the transmission load for delivering the output results among fog nodes when $\Omega_o \cap \Omega_r = \phi$.

Theorem 3. For the uncoded scheme in error-free networks, the transmission load for delivering the output results among the fog nodes can be calculated by

$$ D_{uncoded}(r) = \min \left\{1, \frac{S_I Q}{S_o F_o} \right\}, $$

Proof 3. At each of the $F_o$ fog nodes, $S_o$ output results are required. Assume that the size of each output is $T_o$ bits. There are $S_I Q$ outputs in total at the $F_I$ fog nodes. The size of data for transmission should be the minimum value between $S_o F_o T_o$ and $S_I Q T_o$. Therefore, we have the transmission load
Theorem 4. For the Fountain code-based schemes in error-free networks, the transmission load for delivering the output results can be calculated as

\[
D_{\text{coded}}(\eta_2) = \min \left\{ 1, \frac{(1 + \eta_2)Q}{S_o F_o} \right\},
\]

where \( Q \leq S_o F_o \leq (1 + \eta_2)Q \). \( 1 + \eta_2 \) is the expanding coefficient of the Fountain codes used in Output Delivery stage.

**Proof 4.** By using the Fountain code-based schemes, \((1 + \eta_2)Q\) outputs at the \( F_i \) fog nodes should be delivered to the \( F_o \) fog nodes. Therefore, the transmission load can be calculated by

\[
D_{\text{coded}}(\eta_2) = \min \left\{ 1, \frac{(1 + \eta_2)Q}{S_o F_o} \right\} = \min \left\{ 1, \frac{(1 + \eta_2)Q}{S_o F_o} \right\}.
\]

\[ \square \]

### 4.1.3 Two Adjacent Server Sets

The communication loads for the uncoded and the Fountain code-based schemes in the situation \( \Omega_o \neq \Omega_r \) and \( \Omega_o \cap \Omega_r \neq \phi \) are the same as that for the situation \( \Omega_o = \Omega_r \). As shown in Fig. 5, \( F_{a,p} = \beta F_o \) fog nodes are located in the two adjacent server sets simultaneously. Among the \( F_{a,p} \) fog nodes, \( \alpha F_{a,p} \) ones already have partial output results before Output Delivery stage starts.

**Theorem 5.** For the uncoded scheme in error-free networks, the transmission load for delivering the output results can be calculated by

\[
D_{\text{uncoded}}(r) = \frac{S_o F_o T_o}{S_o F_o} \max \left\{ 0, \min \left\{ 1, \frac{S_o F_o T_o}{S_o F_o} - \frac{\alpha \beta y_o}{S_o} \right\} \right\}.
\]

where \( 0 \leq \alpha \leq 1 \) and \( 0 \leq \beta \leq 1 \).

**Proof 5.** There are \( S_o F_o T_o \) outputs at the \( F_i \) fog nodes. By using the uncoded scheme, \( S_o \) output results are required at each of the \( F_i \) fog nodes. For the situation \( \Omega_o \neq \Omega_r \) and \( \Omega_o \cap \Omega_r \neq \phi \), \( \alpha F_{a,p} \) outputs are already known at the \( \alpha F_{a,p} \) fog nodes among the \( F_o \) ones. Therefore, the transmission load can be calculated by

\[
D_{\text{uncoded}}(r) = \frac{1}{S_o F_o T_o} \max \left\{ 0, \min \left\{ S_o F_o T_o, S_o F_o T_o \right\} \right\}.
\]

**Theorem 6.** For the Fountain code-based schemes in error-free networks, the transmission load for delivering the output results can be written as

\[
D_{\text{coded}}(\eta_2) = \max \left\{ 0, \min \left\{ 1, \frac{S_o F_o T_o}{S_o F_o} - \frac{\alpha \beta y_o}{S_o} \right\} \right\}.
\]

where \( A(\alpha, \beta, F_o, F_i, S_i) \) represents the number of output results already known at the \( \alpha F_{a,p} \) fog nodes, and \( \frac{\alpha \beta F_o y_o}{T_o} \leq A(\alpha, \beta, F_o, F_i, S_i) \leq \min \{ Q, \alpha \beta F_o y_o \} \).

**Proof 6.** At each of the \( F_o \) fog nodes, \( S_o \) output results are required. There are \((1 + \eta_2)Q\) outputs at the \( F_i \) fog nodes. For the situation with two adjacent server sets, \( A(\alpha, \beta, F_o, F_i, S_i) \) outputs are already known at the \( \alpha F_{a,p} \) fog nodes among the \( F_o \) ones. Therefore, the transmission load can be calculated by

\[
D_{\text{coded}}(\eta_2) = \frac{1}{S_o F_o T_o} \max \left\{ 0, \min \left\{ S_o F_o - \alpha \beta F_o y_o T_o, (1 + \eta_2) \right\} \right\} \cdot (Q - A(\alpha, \beta, F_o, F_i, S_i) T_o).
\]

**Proof 7.** Each fog node requires \((K - \frac{r}{F}) y_o \) intermediate results from the other fog nodes to complete its \( y_o \) tasks. Since transmission between the mobile device and the fog nodes and among the fog nodes are over erasure channels, the channel parameters \( e_{DF} \) and \( e_{EF} \) should be taken into consideration. The probability for one computation task to be successfully delivered at the fog node is 

\[
1 - e_{DF} T_o = \frac{1}{1 - e_{DF} T_o}.
\]

Therefore, the transmission load can be calculated by

\[
D_{\text{uncoded}}(r) = \frac{S_i}{F_i(1 - e_{DF})(1 - e_{EF})} \left( 1 - \frac{r}{F} \right).
\]

where \( r \in \{1, 2, \ldots, F\} \). \( e_{DF} \) and \( e_{EF} \) are the erasure probabilities of the links from the mobile devices to the fog nodes, and among the fog nodes, respectively.

**Theorem 7.** For the uncoded scheme in erasure networks, the communication load for exchanging the intermediate results can be calculated by

\[
L_{\text{uncoded}}(r) = \frac{S_i}{F_i(1 - e_{DF})(1 - e_{EF})} \left( 1 - \frac{r}{F} \right).
\]

where \( r \in \{1, 2, \ldots, F\} \). \( e_{DF} \) and \( e_{EF} \) are the erasure probabilities of the links from the mobile devices to the fog nodes, and among the fog nodes, respectively.
Proof 8. The same as for the uncoded scheme, the channel parameter \( \frac{1-e_{DP}}{1-e_{FP}} \) should be taken into account for calculating the communication load. Since the intermediate results are obtained from processing the coded computation tasks, the expanding coefficient of the Fountain codes used for encoding the computation tasks, i.e., \( 1 + \eta_1 \), should also be taken into consideration. Therefore, the communication load can be written as

\[
L_{\text{coded}}(\eta_1) = \frac{(K - \tau_0)\gamma_o T_I + \frac{1-e_{DP}}{1-e_{FP}}}{KQT_I}.
\]

For the situation \( \Omega_o = \Omega_r \), the transmission load for delivering the output results is 0 for both the uncoded and the coded schemes.

4.2.2 Two Different Server Sets
The communication loads for the uncoded and the Fountain code-based schemes in the situation \( \Omega_o \cap \Omega_r = \phi \) are the same as that for the situation \( \Omega_o = \Omega_r \).

Theorem 9. For the uncoded scheme in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

\[
D_{\text{uncoded}}(r) = \min \left\{ \frac{1}{1 - e_{FP}} S_o F_o, S_f Q \right\}. \tag{15}
\]

Proof 9. The channel parameter among the fog nodes, i.e., \( e_{FP} \), should be taken into account when calculating the transmission load. Therefore, the transmission load by using the uncoded scheme can be written as

\[
D_{\text{uncoded}}(r) = \min \left\{ \frac{S_f T_o}{S_o F_o}, \frac{S_f Q}{S_o F_o} \right\} = \min \left\{ \frac{1}{1 - e_{FP}} S_o F_o, S_f Q \right\}.
\]

Theorem 10. For the Fountain code-based schemes in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

\[
D_{\text{coded}}(\eta_2) = \min \left\{ \frac{1}{1 - e_{FP}} (1 + \eta_2) Q \right\}. \tag{16}
\]

where \( Q \leq S_o F_o \leq (1 + \eta_2)Q \) and \( 1 + \eta_2 \) is the expanding coefficient.

Proof 10. We omit the proof, which is relatively straightforward.

4.2.3 Two Adjacent Server Sets
The communication loads for the uncoded and the Fountain code-based schemes in the situation \( \Omega_o \cap \Omega_r = \phi \) are the same as that for the situation \( \Omega_o = \Omega_r \).

Theorem 11. For the uncoded scheme in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

\[
D_{\text{uncoded}}(r) = \frac{1}{S_o F_o T_o} \max \left\{ 0, \min \left\{ \frac{S_o F_o - \alpha F_o y_o}{1 - e_{FP}}, \frac{T_o}{1 - e_{FP}} \right\} \right\}, \tag{17}
\]

where \( F_o \in \{1, 2, \ldots, F\} \).

Proof 11. We omit the proof, which is relatively straightforward.

Theorem 12. For the Fountain code-based schemes in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

\[
D_{\text{coded}}(\eta_2) = \max \left\{ 0, \min \left\{ \frac{1 - \alpha \beta y_o}{S_o F_o}, \frac{(1 + \eta_2) Q}{S_o F_o} \right\} \right\}. \tag{18}
\]

where \( A(\alpha, \beta, F_o, F_I, S_I) \in \left[ \left( \frac{\alpha F_o y_o}{F_I} \right), \min \{Q, [\alpha F_o y_o] \} \right] \).

Proof 12. We omit the proof, which is relatively straightforward.

5 PERFORMANCE ANALYSIS WITH PREDICTABLE MOVEMENT POSITION
In this section, we consider that the moving route of a mobile device is predictable. Communication loads of the fog computing process by using the uncoded and the Fountain code-based schemes with predictable movement position in error-free and erasure networks will be calculated, respectively.

5.1 Predictable Movement Position
If the movement position of the mobile device is predictable, the reduce tasks and the output tasks can be assigned according to the movement position of the device. For the situation \( \Omega_o \neq \Omega_r \) and \( \Omega_o \cap \Omega_r = \phi \), the effect of the predictable movement position on the performance is more significant than the situations \( \Omega_o = \Omega_r \) and \( \Omega_o \cap \Omega_r = \phi \). Thus, we focus on the study of predictable movement position in the situation \( \Omega_o \neq \Omega_r \) and \( \Omega_o \cap \Omega_r = \phi \). As shown in Fig. 6,

![Diagram](image-url)

Fig. 6. The common area of two adjacent server sets.
there is a common area of the two adjacent server sets, i.e., $J$. We use $d$ to represent the distance of the offloading and the receiving positions of the mobile device. The angle $\theta$ can be calculated by $\theta = \arccos \frac{d}{2R}$.

The number of available fog nodes for a mobile device in the common area of sets $\mathbf{\Omega}_o$ and $\mathbf{\Omega}_r$ is $F_P$. The corresponding PMF of HPPP is

$$P_r(F_P) = \frac{\lambda_f A(J)^{F_P}}{F_P!} e^{-\lambda_f A(J)},$$

(19)

where $\lambda_f$ is the distribution density of fog nodes. $A(J)$ represents the area of $J$, i.e., the common area of the two adjacent server sets, and it can be calculated by

$$A(J) = 2R^2 \arccos \frac{d}{2R} - d \sqrt{R^2 - \left(\frac{d}{2}\right)^2}.$$

(20)

### 5.2 Transmission Loads

The communication loads of the situation $\mathbf{\Omega}_o \neq \mathbf{\Omega}_r$ and $\mathbf{\Omega}_o \cap \mathbf{\Omega}_r \neq \phi$ when the movement position is predictable are the same as that for the unpredictable situation. In the following, we will calculate the transmission loads for delivering output results among fog nodes by using the uncoded and the Fountain code-based schemes in error-free and erasure networks, respectively.

#### 5.2.1 Error-Free Networks

**Theorem 13.** For the uncoded scheme in error-free networks, the transmission load for delivering output results among the fog nodes can be written as

$$D_{\text{uncoded}}(r, F_P) = \min \left\{ \max \left\{ 0, 1 - \frac{F_P}{S_o F_o} \right\}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\},$$

(21)

where $F_P$ fog nodes are located at both the two adjacent server sets with probability $P_r(F_P)$, and $0 \leq F_P \leq F$.

**Proof 13.** For the situation $\mathbf{\Omega}_o \neq \mathbf{\Omega}_r$ and $\mathbf{\Omega}_o \cap \mathbf{\Omega}_r \neq \phi$, since $F_P$ fog nodes of the $F_o$ ones already known the outputs, the number of required outputs becomes $(F_o - F_P)S_o$. There are $S_l Q$ output results at the $F_l$ fog nodes, $\gamma_o F_P$ ones are known at the $F_o$ fog nodes. Therefore, we have the transmission load by using the uncoded scheme $D_{\text{uncoded}}(r, F_P) = \frac{1}{S_o F_o T_o} \min \{ \max \{ 0, 1 - \frac{F_P}{F_o} \}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \} = \min \left\{ \max \left\{ 0, 1 - \frac{F_P}{F_o} \right\}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\},$.

The expectation of the transmission load $D_{\text{uncoded}}$ can be calculated by

$$D_{\text{uncoded}} = \int_0^F \left( \frac{\lambda_f A(J)^{F_P}}{F_P!} e^{-\lambda_f A(J)} \right) \cdot \frac{1}{S_o F_o T_o} \min \left\{ \max \left\{ 0, 1 - \frac{F_P}{F_o} \right\}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\} dF_P.$$

(22)

### 5.2.2 Erasure Networks

**Theorem 14.** For the Fountain code-based schemes in error-free networks, the transmission load for delivering output results among the fog nodes can be written as

$$D_{\text{coded}}(\eta_2, F_P) = \min \left\{ \max \left\{ 0, 1 - \frac{\eta_2 F_P}{S_o F_o} \right\}, 1 - \frac{B(F_P, S_l, F_l)}{Q} \cdot \frac{(1 + \eta_2) Q}{S_o F_o} \right\},$$

(23)

where $B(F_P, S_l, F_l)$ represents the number of outputs already known at the $F_P$ fog nodes and $B(F_P, S_l, F_l) \in \left[ \lfloor \frac{Q F_P}{F_o} \rfloor, \min \{ Q, \lfloor \eta_o F_P \rfloor \} \right]$.

**Proof 14.** For the situation $\mathbf{\Omega}_o \neq \mathbf{\Omega}_r$ and $\mathbf{\Omega}_o \cap \mathbf{\Omega}_r \neq \phi$, since $F_P$ fog nodes of the $F_o$ ones already known partial of the outputs, the number of required outputs is $(F_o - F_P)S_o$. There are $(1 + \eta_2)Q$ coded outputs at the $F_l$ fog nodes, and among which $B(F_P, S_l, F_l)$ ones are known. Therefore, the transmission load by using the Fountain code-based schemes can be calculated by

$$D_{\text{coded}}(\eta_2, F_P) = \frac{1}{S_o F_o T_o} \cdot \min \{ \max \{ 0, (S_o F_o - \gamma_o F_P) T_o \}, (1 + \eta_2) \cdot (Q - B(F_P, S_l, F_l)) T_o \}.$$

(24)

The expectation of the transmission load $D_{\text{coded}}(\eta_2)$ can be calculated by

$$D_{\text{coded}}(\eta_2) = \int_0^F \left( \frac{\lambda_f A(J)^{F_P}}{F_P!} e^{-\lambda_f A(J)} \right) \cdot D_{\text{coded}}(\eta_2, F_P) dF_P.$$

(25)

### 5.2.3 Error-Free Networks

**Theorem 15.** For the uncoded scheme in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

$$D_{\text{uncoded}}(r, F_P) = \min \left\{ \max \left\{ 0, 1 - \frac{F_P}{F_o} \right\}, \frac{1}{1 - e_{FF}} \cdot \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\},$$

(26)

where $e_{FF}$ should be multiplied to the number of required outputs. Therefore, the transmission load by using the uncoded scheme can be calculated by

$$D_{\text{uncoded}}(r, F_P) = \frac{1}{S_o F_o T_o} \min \left\{ \frac{S_o T_o}{1 - e_{FF}}, \max \{ 0, (F_o - F_P) \}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\}.$$

(27)

**Proof 15.** For the erasure networks, the channel factor $\frac{1}{1 - e_{FF}}$ should be multiplied to the number of required outputs. Therefore, the transmission load by using the uncoded scheme can be calculated by

$$D_{\text{uncoded}}(r, F_P) = \frac{1}{S_o F_o T_o} \min \left\{ \frac{S_o T_o}{1 - e_{FF}}, \max \{ 0, (F_o - F_P) \}, \frac{S_l Q}{S_o F_o} \frac{\gamma_o F_P}{S_o F_o} \right\}.$$
Theorem 16. For the Fountain code-based schemes in erasure networks, the transmission load for delivering output results among the fog nodes can be written as

$$D_{\text{coded}}(\eta_2, F_p) = \min \left\{ \frac{\max\{0, 1 - \eta_2 F_p/e_{FF}\}}{1 - e_{FF}}, \frac{1}{S_o F_o} \cdot (1 + \eta_2)(Q - B(F_p, S_l, F_i)) \right\},$$

(28)

where $B(F_p, S_l, F_i) = \left[ \frac{Q F_p}{F_p}, \min\{Q, \gamma_o F_p\} \right]$.

Proof 16. We omit the proof, which is relatively straightforward. □

The expectation of the transmission load for the uncoded and the Fountain code-based schemes can be calculated by

$$D_{\text{uncoded}}(r) = \int_0^F \frac{\Gamma_{\lambda_1}(\lambda_1)}{F_p} e^{-\lambda F_1} D_{\text{uncoded}}(r, F_p) dF_p$$

and

$$D_{\text{coded}}(\eta_2) = \int_0^F \frac{\Gamma_{\lambda_1}(\lambda_1)}{F_p} e^{-\lambda F_1} D_{\text{coded}}(\eta_2, F_p) dF_p,$$

respectively.

6 NUMERICAL RESULTS AND DISCUSSION

In what follows, we shall provide a set of numerical results that cast further insights on the proposed Fountain code-based schemes for distributed fog computing. We will compare the communication and the transmission loads achieved by the uncoded and the Fountain code-based schemes. The comparison of the processing time between the uncoded and Fountain code-based schemes will also be given. Finally, the trade-off between the computation, the communication and the transmission loads will be discussed. The comparison of the MDS coded-based and the Fountain code-based schemes will also be discussed in this section.

6.1 Numerical Results

Fig. 7 shows the communication loads achieved by the uncoded and the Fountain code-based schemes with $F = 10$, $F_1 = a_1 F$ ($0 \leq a_1 \leq 1$), $S_l = b_l F_l$ ($b_l > 0$), and computation load ($r$ for the uncoded scheme and $1 + \eta_1$ for the coded schemes) varies from 1 to $F$. $F_1 = a_1 F$ means that $a_1 F$ fog nodes will be assigned reduce tasks. $S_l = b_l F_l$ means that each of the $F_l$ fog nodes will be assigned $b_l Q$ reduce tasks. For both the uncoded and the coded schemes, the communication load decreases with the increasing of the computation load. Compared with the uncoded scheme, the communication load can be reduced by using the Fountain code-based schemes. If more reduce tasks are assigned to fewer fog nodes, e.g., $S_l = 0.5 F_l$, $F_l = 0.5 F$, the communication load increases. If less reduce tasks are assigned to the fog nodes, e.g., $S_l = 0.5 F_l$, $F_l = F$, the communication load decreases.

Fig. 8 shows the communication loads achieved by the uncoded and the Fountain code-based schemes with different parameters $F_1$ and $S_l$ when $F = 10$ and $r = 2$ (or $1 + \eta_1 = 2$). As shown in Fig. 8a, the communication load decreases with the increasing of $a_1$. It means that the communication load can be reduced by assigning the reduce tasks to more fog nodes. As shown in Fig. 8b, the communication load increases with the increasing of $S_l$. To reduce the communication load, the parameter $S_l$ should be set as small as possible.

The transmission loads achieved by the uncoded and the Fountain code-based schemes for the situations of with two different server sets and with two adjacent server sets are given in Figs. 9a and 9b, respectively. The parameters are set as $F = 10$, $a = 0.6$, $b = 0.5$, $1 + \eta_2$ varies from 1.0 to 3.0, $S_l$ changes from 1 to 3, $F_l = F_o = F_l$, and $S_o = 10$. The transmission load by using the coded scheme increases with the increasing of $1 + \eta_2$. When the channel condition becomes worse, e.g., $e_{FF}$ changes from 0.2 to 0.4, the transmission load increases. If $\alpha > 0$ and $\beta > 0$, the transmission load for the situation $\Omega_o \neq \Omega_r$ and $\Omega_o \cap \Omega_r \neq \phi$ is lower than that for $\Omega_o \cap \Omega_r = \phi$.

The transmission loads achieved by the uncoded and the Fountain code-based schemes for the situations of with two different server sets and with two adjacent server sets are given in Fig. 10. The parameters are set as $F = 10$, $1 + \eta_2$ varies from 1.0 to 2.0, $S_l = 3$, $F_l = F_o = F_l$, and $S_o = 10$. In Fig. 10a, the communication load decreases with the increasing of the computation load. Compared with the uncoded scheme, the communication load can be reduced by using the Fountain code-based schemes. If more reduce tasks are assigned to fewer fog nodes, e.g., $S_l = 0.5 F_l$, $F_l = 0.5 F$, the communication load increases. If less reduce tasks are assigned to the fog nodes, e.g., $S_l = 0.5 F_l$, $F_l = F$, the communication load decreases.

Fig. 9. The transmission loads of the uncoded and the Fountain code-based schemes for the situations of (a) $\Omega_o \cap \Omega_r = \phi$, and (b) $\Omega_o \neq \Omega_r$ and $\Omega_o \cap \Omega_r \neq \phi$. 
\( \alpha \) changes from 0.2 to 0.6, and \( \beta = 0.5 \). Larger value of \( \alpha \) means more fog nodes locate in both the two adjacent server sets are assigned reduce tasks. In Fig. 10b, \( \beta \) changes from 0.3 to 0.7, and \( \alpha = 0.6 \). As shown in Figs. 10a and 10b, the transmission load decreases with the increasing of \( \alpha \) and \( \beta \).

As shown in Fig. 11 are the transmission loads for the situation \( \Omega_o \neq \Omega_e \) and \( \Omega_o \cap \Omega_e \neq \phi \) with predictable movement position. The parameters are set as \( F = 10, 1 + \eta_2 \) varies from 1.0 to 10.0, \( S_3 = 3, F_1 = F_3 = F, S_o = 10, R = 20 \text{ m}, \lambda_f = 0.01, \) and the distance between the center of the two adjacent server set \( d = 20 \text{ m} \). The transmission load by using the Fountain code-based schemes is always lower than that of the uncoded scheme when the movement position of the mobile device is predictable. When the channel condition becomes worse, e.g., \( e_{FP} \) changes from 0.2 to 0.4, the transmission load increases.

Fig. 12 shows the transmission loads for the situation \( \Omega_o \neq \Omega_e \) and \( \Omega_o \cap \Omega_e \neq \phi \) with predictable movement position in erasure networks. The parameters are set as \( F = 10, 1 + \eta_2 \) varies from 1.0 to 10.0, \( S_3 = 3, F_1 = F_3 = F, S_o = 10, R = 20 \text{ m}, \lambda_f = 0.01, \) and the distance between the center of the two adjacent server set \( d \) varies from 10 to 30 m. As \( d \) increases, the common area between the two adjacent server sets decreases and the number of fog nodes in the common area decrease as well. Thus, the transmission load increases with the increasing of the distance \( d \).

If set the computation load \( r = 1 + \eta_2 \), the communication and the transmission loads by using the Fountain code-based scheme is always lower than the uncoded scheme. Therefore, the processing time using the coded scheme is always shorter than using the uncoded scheme, i.e., \( T_{\text{coded}} < T_{\text{uncoded}} \). That is, the latency for distributed fog computing by using the Fountain code-based schemes is lower than the uncoded scheme. As shown in Fig. 13 is an example of the processing time reduction by using the Fountain code-based schemes for the situation with two adjacent server sets.

An example of the ratio of processing time reduction by using the coded scheme with different \( r = 1 + \eta_1 \) and \( 1 + \eta_2 \) is given in Fig. 13a. The parameters are set as \( r = 1 + \eta_1 = 1.2 \) and \( 2.0, 1 + \eta_2 = 1.2 \) and \( 2.0, R = 20 \text{ m}, \lambda_f = 0.01, d = 20 \text{ m}, \) and \( e_{DF} = e_{FF} = 0.1 \). When \( r = 1 + \eta_1 \) increases from 1.2 to 2.0, the ratio of reduced processing time for exchanging the intermediate results (corresponds to the communication load) by using the Fountain code-based schemes increases. When \( 1 + \eta_2 \) changes from 1.2 to 2.0, the ratio of reduced processing time for delivering the output results (corresponds to the transmission load) by using the Fountain code-based schemes decreases. Therefore, by properly code design and carefully choosing the coding parameters \( \eta_1 \) and \( \eta_2 \), the processing time by the Fountain code-based schemes would be much shorter than the uncoded scheme.

The ratio of reduced processing time by the Fountain code-based schemes with different distance \( d \) is given in Fig. 13b. The parameters are set as \( r = 1 + \eta_1 = 1.2, 1 + \eta_2 = 1.2, R = 20 \text{ m}, \lambda_f = 0.01, d \) changes from 10 to 35 m, and \( e_{FF} = 0.1 \). As the distance \( d \) increases from 10 to 35 m, the ratio of reduced processing time by using the Fountain code-based schemes decreases. For example, when \( d = 10 \text{ m}, \) around 95 percent of the time for delivering the output results can be reduced by the coded scheme in the best case. When \( d = 30 \text{ m}, \) around 56 percent of the time can be reduced by the Fountain code-based schemes.

### 6.2 Discussion
As it is analyzed in Section 3.3, the total processing time is mainly determined by the computation, the communication, and the transmission loads. Shorter total processing time
means lower latency for the distributed fog computing process. To achieve shorter processing time and lower latency, the computation, the communication, and the transmission loads should be lower. According to the theorems given in Sections 4 and 5 and the numerical results in Section 6.1, to reduce the computation load at each fog node in Map stage, the parameters $S_1$ and $S_2$ should be chosen as small as possible, and the parameter $F_1$ should be as large as possible. To reduce the communication load for exchanging the intermediate results among fog nodes, the parameters $F_1$ and $F_2$ should be as large as possible, while $S_1$ should be chosen as small as possible.

To reduce the transmission load for delivering the output results among fog nodes, the parameters $h_1$ and $h_2$ should be chosen as large as possible, while $h_1$ should be chosen as small as possible. The transmission load decreases with the increasing of the computation load, i.e., increasing $h_1$. The transmission load decreases with the decreasing of $h_2$. On the other hand, if reliability is also considered, the parameters $h_1$ and $h_2$ should be large enough to ensure the transmission reliability in erasure networks. Thus, in order to balance the computation and the communication loads, and reduce the communication and the transmission loads, the parameters $h_1$ and $h_2$ should be chosen with properly values.

The main factors, which determine the value of $h_1$ and $h_2$, include the degree distributions $\Psi(x)$ and $\mu(x)$, the processing capability of fog nodes, the quality of channels, etc. The criteria for degree distribution design is: (1) Ensure the minimum possible number of encoded packets required to recover all original information successfully, and (2) Keep the average degree of all encoded data as low as possible. $h_1$ and $h_2$ should be with relative large value if the channel quality is worse, and with relative small value if the channel quality is good. For choosing $h_1$, the processing capability of fog nodes should also be taken into account. For example, when the processing capability of fog nodes is strong, $h_1$ should be with a relative large value. When the processing capability of fog nodes is weak, $h_1$ can be smaller.

When $1 + h_2 = 1.0$, the transmission load in Figs. 9, 10, 11, and 12 corresponds to the performance achieved by the MDS code-based schemes. In theoretic terms, MDS codes are able to achieve optimal performance. However, high complexity makes the MDS code-based schemes impractical in large-scale and time-varying networks. For example, device mobility causes the network parameters, such as the quality of communication links, the number of connected devices, the continuously generated computation tasks, etc., time-varying. Compared with the MDS code-based schemes, the Fountain code-based schemes have much lower complexity and is more suitable to be used in large-scale and time-varying networks.

7 Conclusion

We considered the mobility of devices in IoMTs. Fountain code-based schemes with high flexibility and low complexity were proposed for the distributed fog computing process in IoMTs. The total processing time for the un-coded and the Fountain code-based schemes was calculated. The latency related performance, i.e., the communication load for exchanging intermediate results and the transmission load for delivering the output results among fog nodes, was analyzed. Comparison and numerical results showed that by applying the proposed Fountain code-based schemes, both the communication and the transmission loads can be reduced. That is, compared with the un-coded scheme, shorter total processing time and lower latency for the distributed fog computing process in IoMTs can be achieved by using the Fountain code-based schemes.

Acknowledgments

The work of Jing Yue was supported by Swedish Strategic Research Foundation (SSF) project “Cyber Security for Next Generation Factory” (SSF grant RIT17-0032) and SRA ICT TNG Post-doc project “Coding for Reliability and Low-latency”. The work of Ming Xiao was supported in part by EU Marie Sklodowska-Curie Actions Project entitled “High-reliability Low-latency Communications with network coding”, and ERA-NET Smart Energy Systems SG+ 2017 Program, “SMART-MLA” with Project number 89029 (and SWEA number 42811-2), and SSF project “High-reliable Low-latency Industrial Wireless Communications”. The authors would like to thank Dr. Urban Bilstrup, Manager of Reliable Wireless Lab at RISE, Sweden, for his valuable suggestions and great assistance to this work.

References

[1] C. Puliafito, E. Mingozzi, and F. Anastasi, “Fog computing for the internet of mobile things: Issues and challenges,” in Proc. IEEE Int. Conf. Smart Comput., 2017, pp. 1–6.
[2] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu, “Internet of mobile things: Mobility-driven challenges, designs and implementations,” in Proc. IEEE 1st Int. Conf. Internet-of-Things Design Implementation, 2016, pp. 25–36.
[3] Cisco and/or its affiliates, “Fog computing and the internet of things: Extend the cloud to where the things are,” White Paper, pp. 1–6, 2015.
[4] J. Cabrera G., D. Lucani R., and P. Fitzek, “On network coded distributed storage: How to repair in a fog of unreliable peers,” in Proc. IEEE Int. Symp. Wireless Commun. Syst., 2016, pp. 188–193.
[5] Y. Zhang, D. Niyato, and P. Wang, “Offloading in mobile cloudlet systems with intermittent connectivity,” IEEE Trans. Mobile Comput., vol. 14, no. 12, pp. 2516–2529, Dec. 2015.
[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. Letaief, “A survey on mobile edge computing: The communication perspective,” IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322–2358, Aug. 2017.
[7] Y. Kim, J. Kwak, and S. Chong, “Dual-side optimization for cost-delay tradeoff in mobile edge computing,” IEEE Trans. Veh. Technol., vol. 67, no. 2, pp. 1765–1781, Feb. 2018.
[8] S. Wang, J. Xu, N. Zhang, and Y. Liu, “A survey on service migration in mobile edge computing,” IEEE Access, vol. 6, pp. 23 511–23 528, Apr. 2018.
[9] S. Li, M. Maddah-Ali, and A. Avestimehr, “Coded distributed computing: Straggling servers and multistage dataflows,” in Proc. 54th Annu. Allerton Conf. Commun. Control Comput., Sep. 27–30, 2016, pp. 164–171.
[10] S. Li, M. Maddah-Ali, and A. Avestimehr, “A unified coding framework for distributed computing with straggling servers,” in Proc. IEEE Globecom Workshops, Dec. 4–8, 2016, pp. 1–6.
[11] S. Li, M. Maddah-Ali, Q. Yu, and A. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” IEEE Trans. Inf. Theory, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[12] S. Li, M. Maddah-Ali, and A. Avestimehr, “Fundamental tradeoff between computation and communication in distributed computing,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 10–15, 2016, pp. 1814–1818.
[13] K. Lee, M. Lamm, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” in Proc. IEEE Int. Symp. Inf. Theory, 2016, pp. 1143–1147.
[14] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” IEEE Trans. Inf. Theory, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[15] S. Li, M. Maddah-Ali, and A. Avestimehr, “Coding for distributed fog computing,” IEEE Commun. Mag., vol. 55, no. 4, pp. 34–40, Apr. 2017.

[16] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, “Coded computation over heterogeneous clusters,” in Proc. IEEE Int. Symp. Inf. Theory, 2017, pp. 2408–2412.

[17] N. Ferdinand and S. Draper, “Hierarchical coded computation,” in Proc. IEEE Int. Symp. Inf. Theory, 2018, pp. 1620–1624.

[18] H. Park, K. Lee, J. Sohn, C. Suh, and J. Moon, “Hierarchical coding for distributed computing,” in Proc. IEEE Int. Symp. Inf. Theory, 2018, pp. 1630–1634.

[19] S. Kiani, N. Ferdinand, and S. Draper, “Exploitation of stragglers in coded computation,” in Proc. IEEE Int. Symp. Inf. Theory, 2018, pp. 1988–1992.

[20] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in synchronous gradient descent,” 2017, arXiv:1612.03301v2.

[21] N. Rav, J. Tamo, R. Tandon, and A. G. Dimakis, “Gradient coding from cyclic MDS codes and expander graphs,” 2017, arXiv:1707.03858.

[22] Q. Yu, M. Maddah-Ali, and S. Avestimehr, “Polynomial codes: An optimal design for high-dimensional coded matrix multiplication,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 4403–4413.

[23] W. Halbawi, N. Azzam, F. Salehi, and H. Hasibi, “Improving distributed gradient descent using Reed-Solomon codes,” in Proc. IEEE Int. Symp. Inf. Theory, 2018, pp. 2027–2031.

[24] A. Vardy, “The intractability of computing the minimum distance of a code,” IEEE Trans. Inf. Theory, vol. 43, no. 6, pp. 1757–1766, Nov. 1997.

[25] A. Severinson, A. Amat, and E. Rosnes, “Block-diagonal and LT codes for distributed computing with straggling servers,” IEEE Trans. Commun., vol. 67, no. 3, pp. 1739–1753, Mar. 2019.

[26] J. Yue and M. Xiao, “Coded decentralized learning with gradient descent for big data analytics,” IEEE Commun. Lett., to be published, doi: 10.1109/LCOMM.2019.2930513.

[27] M. Luby, “LT codes,” in Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci., 2002, pp. 271–280.

[28] A. Shokrollahi, “Raptor codes,” IEEE Trans. Inf. Theory, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[29] J. Yue, M. Xiao, and Z. Pang, “Distributed fog computing based on batched sparse codes for industrial control,” IEEE Trans. Ind. Informat., vol. 14, no. 10, pp. 4683–4691, Oct. 2018.

[30] S. Li, M. Maddah-Ali, and A. Avestimehr, “Architectures for coded mobile edge computing,” in Proc. IEEE Fog World Congr., 2017, pp. 1–6.

[31] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” in Proc. ACM SIGCOMM Comput. Commun. Rev., 1998, pp. 56–67.

[32] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Coded computation for multicore setups,” in Proc. IEEE Int. Symp. Inf. Theory, 2017, pp. 2413–2417.

[33] Q. Yu, M. A. Maddah-Ali, and A. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” 2018, arXiv:1801.07487v2.

[34] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, “Lagrange coded computing: Optimal design for resiliency, security, and privacy,” 2019, arXiv:1806.00939v4.

[35] J. Kosaian, K. Rashmi, and S. Venkatesan, “Learning a code: Machine learning for approximate non-linear coded computation,” 2018, arXiv:1806.01259v1.

[36] F. Baccelli, B. Blaszczyszyn, and P. Muhlethaler, “An Aloha protocol for multihop mobile wireless networks,” IEEE Trans. Inf. Theory, vol. 52, no. 2, pp. 421–436, Feb. 2006.

[37] D. Molchanov, “Distance distributions in random networks,” Ad Hoc Netw., vol. 10, no. 6, pp. 1146–1166, Aug. 2012.

[38] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in Proc. 6th USENIX Symp. Operatng Syst. Design Implementation, 2004, pp. 137–150.

[39] S. Li, M. Maddah-Ali, and A. Avestimehr, “Coded MapReduce,” in Proc. 53rd Annu. Allerton Conf. Commun. Control Comput., 2015, pp. 1–8.