Computer Networks 000 (2016) 1–15

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An adaptive disorder-avoidance cooperative downloading method

Xiuxiu Wen[a], Guangsheng Feng[a,b,*], Huiqiang Wang[a], Hongwu Lv[a], Junyu Lin[a]

[a] College of Computer Science and Technology, Harbin Engineering University, Harbin, 150001, China
[b] Key Laboratory of Complex Systems Modeling and Simulation of Ministry of Education, Hangzhou Dianzi University, Hangzhou, 310018, China

ARTICLE INFO

Article history:
Received 10 July 2015
Revised 23 March 2016
Accepted 5 April 2016
Available online xxx

Keywords:
Cooperative downloading
Data disorder
Queue theory
Heterogeneous wireless networks

ABSTRACT

Applications of heterogeneous wireless networks can help to achieve ubiquitous services. Cooperative downloading is a download technique used on cellular networks and wireless self-organized networks. It helps wireless users who are nearing the limits of their data plan to download data from the Internet. However, the existing studies on cooperative downloading techniques omit the problem of data disorder. Data disorder can decrease the quality of services experienced by users and increase memory usage. In this paper, we model cooperative downloading using queue theory and propose a calculation method for solving data disorder and decreasing download time. Based on the calculation method, an adaptive disorder-avoidance cooperative downloading method is proposed. The method consists of two parts: an adaptive task dissemination algorithm and a dynamic task delay prediction mechanism. The algorithm is implemented based on the calculation method that takes into account the dynamic features of wireless networks. We also propose a prediction model based on neural network learning and moving average, then use the model in the prediction mechanism to enhance the performance of the proposed method in scenarios with dynamic download rates. We used Network Simulation version 2 for the simulation, and simulation results show that the proposed method can solve the data disorder problem and be adapted to mobile scenarios. Furthermore, it can decrease the download time.

1. Introduction

With the development of wireless communication techniques, heterogeneity has become a prominent characteristic of modern computing environments [1,2]. Cooperative downloading is a downloading technique that operates over both long-range communication networks (cellular networks) and short-range communication networks (self-organized networks) [3]. Under cooperative downloading, a user (primary user) can ask other users (cooperative users) to download data using their data plan allowances to help him/her. The primary user decomposes the task of downloading a file into small tasks (e.g., downloading a segment of the target file). Cooperative users download segments using their data plan allowances and forward the segments to the primary user via short-range communication links. The benefits of the cooperation are two-sided. On one hand, the primary user can still access the Internet even if he/she has already run out of data plan; on the other hand, cooperative users can obtain rewards from cooperation and can manage their data plan more efficiently [4]. However, cooperative users will probably have selected different Internet service providers (ISPs), such as China Mobile or China Unicom, and bought different services, such as 2G, 3G, and 4G. All these differences can cause various task delay and the problem of data disorder. For instance, the head of the target file may be downloaded after the tail, which delays the time when content can be used and impacts the primary user's service experience, especially if the file can be used while downloading (such as streaming media) or there are priorities among segments. The disorder problem can be described as a case when the completion order of a task is different from its index. We call this difference the *C-I difference*. Let C_i be the completion order of Task i, then the *C-I difference* of Task i is $|C_i - i|$, and the average *C-I difference* is $1/M \sum_{i=1}^{M} |C_i - i|$, where M is the number of tasks. Our aim is to design a disorder-avoidance method that can decrease the average *C-I difference* as much as possible without any extra communication overhead or increase in download time.

We propose a cooperative downloading method for solving the data disorder problem in this paper. The proposed method consists of two parts: an adaptive task dissemination algorithm and a dynamic task delay prediction mechanism. The main contributions of our work are as follows:

- We model the downloading process using queue theory and propose a calculation method for minimizing the average *C-I*

* Corresponding author.
E-mail addresses: wenxiuxiu@hrbeu.edu.cn (X. Wen), fengguangsheng6600@163.com (G. Feng).

http://dx.doi.org/10.1016/j.comnet.2016.04.004
1389-1286/© 2016 Elsevier B.V. All rights reserved.

difference in theory. Furthermore, considering the dynamic features of wireless networks, we derive a recursive version of the calculation method and propose the adaptive task dissemination algorithm. In each step of the recursion, the result is calculated according to the parameters in real time. The simulation results show that the maximum usage delay (MUD) caused by disorder can be decreased by 85% under the mobile scenarios by applying our method.
- A task delay prediction mechanism is proposed for enhancing the adaptive capacity of the proposed method under dynamic download rate scenarios. We use the moving average model to implement the prediction model and improve it based on the basic idea of neural network learning. Hence, the prediction results are more accurate. The dataset of [5] was used to test the performance of our model, and the test results show that the average prediction error is less than 3%.
- By taking into consideration the time of task completion for users, we only assign tasks to users with sufficient download rates to avoid the "long tail" of the overall download process. The simulation results show that our method can make full use of available download links and cut off the long tail caused by cooperative users with low download rates.

The rest of this paper is organized as follows. Section 2 briefly browses related work on the cooperative downloading techniques. The system model of the cooperative downloading that we focus on is introduced in Section 3. We present the calculation method in Section 4. The details of the adaptive cooperative downloading method are presented in Section 5. Section 6 presents the results of the simulation. We conclude our work and state some future work directions in Section 7.

2. Related work

There have been many studies on cooperative downloading techniques. Most focus on popular content distribution (PCD), in which all users are interested in the same file. The traditional approaches are broadcasting and probabilistic replication [6]. Combining probabilistic replication with the basic idea of BitTorrent, a successful protocol for downloading popular content in wired networks, [7] proposed SPAWN, which is a swarming protocol consisting of a gossip mechanism and a piece-selection strategy. Furthermore, to get rid of piece selection, network coding has been introduced into cooperative content distribution systems [8–10]. The problem of finding the optimal number of transmissions for RLNC(Random Linear Network Coding)-based cooperative content distribution systems was then categorized as the "coded cooperative data exchange problem" by [11], and it has been studied extensively by [12–14]. There are also some studies discussing how to encourage cooperation under this scenario [15–17], such as the instance pay-off method in [18], fair cooperative content-sharing service proposed by [3], and fair cost allocation method proposed in [19]. The above work was all aimed at PCD. In contrast, we focus on general content downloading scenarios, in which a user may download any file from the Internet, which is not necessarily the same as the others.

Additionally, some studies have investigated content downloading in an environment where devices carried by vehicles move too fast to connect well with Roadside Units (RSUs). This occurs because the commonly used communication techniques for smart transportation systems such as Dedicated Short Range Communication [20] are short range. Trullols-Cruces et al. [21] and Liang et al. [22] assumed that a network consisting of vehicles is delay-tolerate, and they used the help of passing vehicles to forward the traffic of vehicles to a destination RSU. Zhou et al. [23] modelled the driving habits of vehicles on the highway and proposed a linear cluster formation scheme (ChainCluster) for selecting appropriate vehicles as helpers. Furthermore, Sardari et al. [24] considered the limited cooperation-buffer of the helpers and used rateless coding at the RSU side to achieve reliable dissemination from an RSU to the vehicles. A survey on this area can be found in [25]. In contrast to the above work, we focus on cooperation consisting of long-range communication (device-to-cellular access point) and short-range communication (device-to-device).

Cooperation that consists of long-range communication and short-range communication can lead to significant performance gains because short-range wireless technologies provide higher data rates given of their geographical proximity [26]. For general content downloading in this scenario, Do et al. [27] considered the mobility of users and proposed PatchPeer, which selects the user (Closest Peer) with the shortest Euclidean distance from the primary user as the cooperative user, so that the local network with short-range communication can be more reliable. Shijie et al. [28] proposed a Cooperative Content Fetching-based strategy (CCF). CCF consists of a stability estimation model (SEM) and a communication quality forecast model (CQFM). Using SEM and CQFM, CCF selects the stable user with the highest bandwidth as the cooperative user. The above approaches can make the local network more stable, but simply using the user with the best condition as the cooperative user cannot make full use of download links. Tuo et al. [4] proposed a data plan sharing system to help smart phone users manage their data plan more efficiently. Using a dynamic task dissemination strategy, the download links are all fully used in the system, and download time is decreased. However, the disorder problem is not considered.

Disorder can delay the time when content can be used and decrease the memory utilization, especially for files that can be used while downloading. Memory is a limited resource, especially for mobile devices. If the disorder problem is solved, the segments of the target file can be used as soon as they are downloaded, and then they can be removed if the operating systems require it. The advantages are not only good utilization of memory, but also good service experience. Therefore, it is very important to solve the disorder problem. There have been some studies addressing the data disorder problem [29–31]. However, all of them aim at solving the package disorder of multipath routing or traffic splitting (a survey can be found in [32]). Multipath routing or traffic splitting is very different from cooperative downloading. Under multipath routing, a client makes multiple connections directly with a server [33,34]. There is a continuous flow on each connection. Cooperative downloading does not have these direct connections, or a special "server side" like multipath routing, which makes its package disorder solutions hard to apply. Therefore, it is very important to develop new methods for solving the disorder problem of cooperative downloading systems.

3. System model of cooperative downloading

In the cooperative downloading system, users can be classified into two types:

- Primary User (PU): a user who needs to download a file from the Internet, but cannot finish it by him/herself because he/she does not have a sufficient data plan.
- Cooperative User (CU): a user who helps a primary user to download parts of the target file from the Internet and gets rewards from the primary user.

The system model of cooperative downloading is presented in Fig. 1. The primary user broadcasts requests to his/her direct neighbours, and the neighbours accepting the request become cooperative users. We assume that the mobile devices are carried by

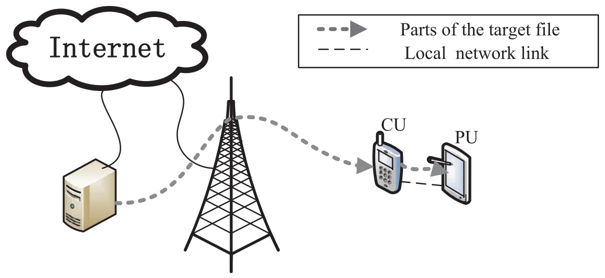


Fig. 1. System model of cooperative downloading.

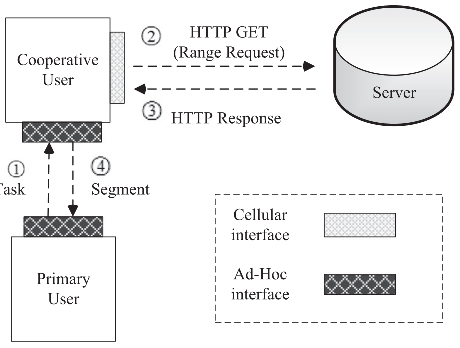


Fig. 2. Process by which a cooperative user completes a task.

pedestrians and that there is a large number users available in scenarios such as in a conference centre, shopping mall, or on campus. The primary user then constructs a local network with his/her cooperative users to guarantee the exchange of data among them. Next, the primary user decomposes the task of downloading the file into small tasks (i.e., downloading a segment of the target file), and assigns the small tasks to the cooperative users.

Each cooperative user completes its task via HTTP, as shown in Fig. 2. When a cooperative user receives a task from the primary user, it sends an HTTP GET request with the “range” field configured to the server, and then obtains the download content (a segment) from the HTTP response. After the segment is downloaded, the cooperative user sends the segment to the primary user via a TCP socket, and then the task is finished. When the primary user obtains all the segments of the target file, the cooperative downloading process ends and the cooperative users are given rewards according to their consumption.

We assume that there is a central authority and present a method for calculating rewards based on the consumption of users. Let U_i denote the cooperative user with identifier i, and let $C_{\text{coop}}(i)$ be his/her consumption for downloading a unit of data for the primary user. Because cooperative users use their own data plan (bought from ISPs) to download data, $C_{\text{coop}}(i)$ should include energy and data plan consumption, as below:

$$C_{\text{coop}}(i) = \omega C_{\text{dp}}(i) + (1-\omega)E_{\text{coop}}(i) \quad (1)$$

Here, $C_{\text{dp}}(i)$ is data plan consumption and $E_{\text{coop}}(i)$ is energy consumption, which can be calculated as shown in (2) [19]. In addition, ω is a weighting factor between 0 and 1.

$$E_{\text{coop}}(i) = E_{\text{cell}}(i) + E_{\text{tx}}(i) + E_{\text{ct}}(i) \quad (2)$$

Here, $E_{\text{cell}}(i)$ is the energy required to download the data via a base station (Steps 2 and 3 of Fig. 2). In addition, $E_{\text{tx}}(i)$ denotes the energy consumed by transmitting the data in the local network (Step 4 of Fig. 2) and $E_{\text{ct}}(i)$ is the energy consumed by local network management [4]. The total consumption of U_i is $f(i)C_{\text{coop}}(i)$, where $f(i)$ is the amount of data U_i downloaded.

The rewards are paid in virtual money. Let N be the number of cooperative users. We assume that there is a centre in charge of calculating virtual money. The virtual money that U_i can obtain is:

$$R_{\text{ward}}(i) = \frac{C_{\text{coop}}(i)f(i)}{\sum_{i=1}^{N} C_{\text{coop}}(i)f(i)} P_{\text{total}} \quad (3)$$

Here, P_{total} is the total amount of virtual money that the primary user can give, which is calculated by the centre and subtracted from the account of U_i after downloading. To encourage cooperation, P_{total} is calculated based on reputation as follows:

$$P_{\text{total}} = (1+\varpi)^{r_{\text{ep}}} \sum_{i=1}^{n} f(i) \quad (4)$$

Here, r_{ep} is the reputation of the primary user and ϖ is a punishment factor [4]. If the reputation of the primary user is too low, he/she can only give little virtual money to his/her cooperative users, and then he/she has little chance of making others download data for him/her. In this case, the primary user has to download data for others to improve his/her reputation, and hence cooperation can be encouraged.

4. Basic concept

The disorder problem can greatly decrease the service experience of users and increase memory usage. In this section, the cooperative downloading process is modelled using queue theory in terms of task delay, which is then used to analyse the data disorder problem. Based on this analysis, we present our calculation method for solving the data disorder problem and the challenges of applying the method in practice.

4.1. Task delay analysis

Each cooperative user downloads a segment from the Internet after he/she receives a task from the primary user and then he/she sends the segment to the primary user. The amount of time consumed by this process is task delay. Therefore, task delay consists of two parts: the delay of downloading a segment from the server and the delay of sending the segment to the primary user.

For the first part, the download rates may be different for different users because they probably have selected different ISPs and bought different levels of network services. Let R_i be the download rate of U_i. The delay of the first part is $S/\mathrm{E}[R_i]$, where S is the size of a segment.

After the cooperative user downloads the segment, he/she sends the segment to the primary user using the local network links. The wireless transmissions of the cellular networks do not interfere with the wireless transmissions in the local networks because they work at different frequencies. However, cooperative users can interfere with each other, as shown in Fig. 3, and they can suffer interference cause by noise from outside the cooperative downloading system. Because users are close to each other, they share a same channel. We assume that they use TCP-based transmission in the local network. TCP has the ability to adapt its transmission rate, and it can make full use of the network bandwidth. If the interference from outside is fixed, we let r be the maximum data rate when there is only one cooperative user sending data to the primary user. If there are k users sending data to the primary user simultaneously and r_i is the data transmission rate of U_i, then $\sum_{i=1}^{k} r_i = r$, because users should share the wireless channel together and the capacity of the wireless transmission is fixed. In fact, r is a random variable of the interference from the outside. For instance, if the users use 802.11g to transmit a segment, then $\mathrm{E}[r] = 22$ Mbit/s [35]. Furthermore, if k is too big, then r_i would

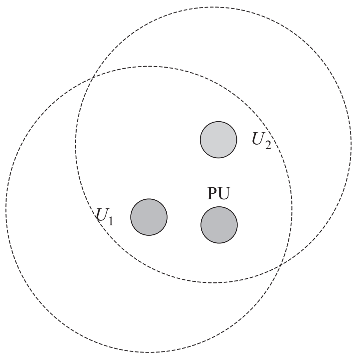


Fig. 3. Channel collisions of the local network

be very small for some i, and it would take a longer time to transmit a segment. Because of mobility, a longer transmission is easier to interrupt. Therefore, we assume that the maximum number of connections that the primary user can accept at the same time is χ, and the extra cooperative users will wait in a queue. Therefore, this process can be modelled using queue theory.

The moment a cooperative user finishes downloading a segment corresponds to the "arrive" portion of the queue. The process in which a cooperative user sends a segment to the primary user corresponds to the "service". Let $N(t)$ be the number of cooperative users in the queue at t, including those waiting and those in service. The number of servers is χ. Each of the servers has variable average service rate $\upsilon_i = \mathrm{E}[r_i]/S$. Let $\upsilon = \sum_{i=1}^{\chi} \upsilon_i$ (the total service rate), then $\upsilon = \mathrm{E}[\sum_{i=1}^{\chi} r_i/S] = \mathrm{E}[r]/S$.

Each cooperative user is an arriving flow. The expected arrival rate of U_i is $\mathrm{E}[R_i]/S$. The number of cooperative users outside the queue is $N - N(t)$, which compose the arrival process as a whole. Each user has the same probability of being outside the queue, so the expected arrival rate of the arrival process is $(N - N(t))\lambda$, where $\lambda = 1/N\sum_{i=1}^{N} \mathrm{E}[R_i]/S$.

The arrival process and the service time may have general distributions. However, to simplify the model, we assume that cooperative users arrive according to the Poisson process and the service time has an exponential distribution. Let $p_{ij}(\Delta t)$ be transition probability. Then, $\mathrm{P}\{N(t+\Delta t) = j | N(t) = i\}$, and $N(t)$ can be described as a continuous time Markov chain with transition probability on the state space {0, 1, 2, $\cdots$, N}.

$$p_{i,i+1}(\Delta t) = \lambda(N-i)\Delta t + o(\Delta t), 0 \le i < N$$

$$p_{i,i-1}(\Delta t) = \Delta t\left(\sum_{i=1}^{\chi} \upsilon_i\right) + o(\Delta t) = \upsilon\Delta t + o(\Delta t), 1 \le i \le N$$

$$p_{i,j}(\Delta t) = o(\Delta t), |i-j| \ge 2$$

Let p_j be the equilibrium probability $\lim_{t\to\infty} P\{N(t) = j\}$. It is then easy to prove that

$$p_j = \binom{N}{j} j!\, p_0 \tag{5}$$

where $\rho = \lambda/\upsilon$ and $p_0 = [\sum_{j=0}^{N} \frac{N!}{(N-j)!}\rho^j]^{-1}$.

Under an equilibrium distribution, the probability that there are already j users in the queue when a user arrives is also p_j [36]. Therefore, the waiting time is $\sum_{j=1}^{\chi-1} p_j \frac{1}{(\upsilon/j)} + \sum_{j=\chi}^{N-1} p_j \frac{j}{\chi(\upsilon/\chi)} = \sum_{j=0}^{N-1} jp_j/\upsilon$. We denote the delay of sending a segment to the primary user as ε. According to the Little formula, ε can be estimated as:

$$\hat{\varepsilon} = 1/\upsilon + \sum_{j=0}^{N-1} jp_j/\upsilon \tag{6}$$

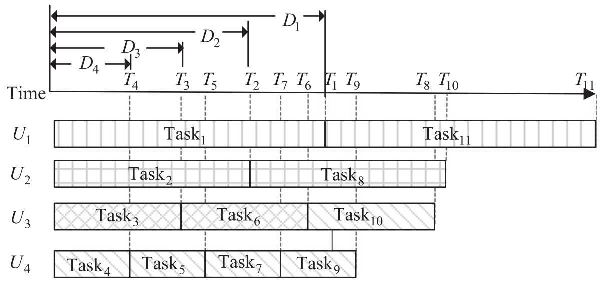


Fig. 4. Data disorder of the dynamic strategy.

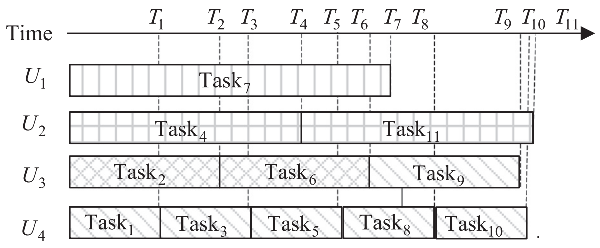


Fig. 5. Task dissemination method without data disorder.

Hence, the average task delay of U_i is

$$D_i = S/\mathrm{E}[R_i] + \hat{\varepsilon} \tag{7}$$

4.2. Disorder problem

Task delay may be different for cooperative users, as shown in (7). In this case, the disorder problem can be very serious without a careful task dissemination strategy. Take, for instance, the case when a primary user assigns tasks one-by-one as the dynamic strategy in [4]. Fig. 4 shows the basic idea.

The target file is divided into 11 segments in sequence, and the small tasks of downloading the segments compose a task list: $L = \{\mathrm{Task}_1, \mathrm{Task}_2, \cdots, \mathrm{Task}_{11}\}$. Once a cooperative user finishes a task, the primary user selects a new task from the front of the task list and assigns it to the cooperative user. There are four cooperative users: U_1, U_2, U_3, and U_4. In addition, T_k is the time when Task$_k$ is finished.

First, the primary user assigns a task to each cooperative user. Here, U_1, U_2, U_3 , and U_4 are given Task$_1$, Task$_2$, Task$_3$, and Task$_4$, respectively. User U_4 finishes his/her task first, and a new task (Task$_5$) is a assigned to him/her. This process continues until all the tasks of L are assigned. Tasks are finished completely out of order, as shown in Fig. 4. For instance, Task$_1$ is the seventh task to be completed. The *C-I difference* of Task$_1$ is 6, and the *C-I difference* of Task$_5$ is 3.

However, if the tasks can be assigned as shown in Fig. 5, then C_i equals i, and $1/M\sum_{i=1}^{M} |C_i - i|$ is minimized to 0. The download time can be decreased at the same time.

4.3. Solution for the disorder problem

According to the analysis in Section 4.2, we should make C_i equal i. To achieve this goal, tasks should be assigned according to the task delay of users. Each cooperative user completes his/her tasks in order, so the expected finishing time of the kth task of U_i is kD_i. We define $\lfloor kD_i/D_j \rfloor$ to be the expected number of tasks that can be finished before kD_i by U_j, and the expected number of tasks finished before kD_i is about $g(k, i)$, as shown below.

$$g(k,i) = \sum_j \left\lfloor kD_i/D_j \right\rfloor \tag{8}$$

Assigning $\text{Task}_{g(k,\,i)}$ to U_i as his/her kth task can then make $C_{g(k,\,i)}$ equal $g(k, i)$, and if all tasks are assigned in this way, *C-I difference* can be 0 for all tasks. Nevertheless, some related problems still need to be considered, such as whether there is any task assigned to more than one cooperative user or if there are any tasks that are not assigned to any cooperative user. We describe these problems in a proposition and prove it as follows.

Proposition 1. *If $D_i a \neq D_j b$ for any $i \neq j \leq N$, where a and b are positive integers less than M, then there is a set partition $L_1, L_2, \cdots, L_N$ of L such that*

$$L_i = \{\text{Task}_{g(n,i)} | n = 1, 2, 3, \cdots, \text{AND } g(n, i) \leq M\} \tag{9}$$

Proof of Proposition 1. If $L_1, L_2, \cdots L_N$ is a set partition of L, then $\bigcup_{1\leq i\leq N} L_i = L$, and $L_i \bigcap L_j = \varnothing$ should hold for any $i \neq j \leq N$. It is obvious that $\bigcup_{1\leq i\leq N} L_i \subset L$, and if we can prove $\bigcup_{1\leq i\leq N} L_i \supset L$, then $\bigcup_{1\leq i\leq N} L_i = L$ can be proved. To prove $\bigcup_{1\leq i\leq N} L_i \supset L$, we only need to demonstrate that any element belonging to L belongs to $\bigcup_{1\leq i\leq N} L_i$, that is, $\text{Task}_k \in \bigcup_{1\leq i\leq N} L_i$ for all $k = 1, 2, \cdots, M$. We can prove $\bigcup_{1\leq i\leq N} L_i \supset L$ using mathematical induction.

Each cooperative user completes his/her tasks in order, so the completion time of $\text{Task}_{g(k,\,i)}$ is kD_i for any $i < N$, $k \in Z^*$. Let D_c be $\min_{1\leq i\leq N} \{D_i\}$. It is obvious that $g(1, c) = 1$ and $\text{Task}_1 \in L_c \subset \bigcup_{1\leq i\leq N} L_i$.

If we can prove that $\text{Task}_{k+1} \in \bigcup_{1\leq i\leq N} L_i$ based on the assumption of $\text{Task}_k \in \bigcup_{1\leq i\leq N} L_i$, then, by induction, $\text{Task}_k \in \bigcup_{1\leq i\leq N} L_i$ for all $k \leq M$ can be proved. The aim is to find a set from $L_1, L_2, \cdots L_N$ that contains Task_{k+1}. We first obtain the expression of T_k, then construct a special time based on T_k that corresponds to the finishing time of a task in a set denoted by L_y. We then prove that the task is Task_{k+1} and the special time is T_{k+1}. The details are presented as follows:

Because $\text{Task}_k \in \bigcup_{1\leq i\leq N} L_i$, there must be an L_x containing Task_k, so we have $k = g(k_x, x) = \sum_{1\leq q\leq N} \lfloor (k_x D_x)/D_q \rfloor$, where k_x is an integer according to (8). It is obvious that $T_k = k_x D_x$.

Let $k_y D_y$ be $\min_{1\leq i\leq N} \{k_i D_i | k_i D_i > k_x D_x, k_i = 1, 2, \ldots .M\}$, the special time (as it is obvious that if $k_y D_y < k_x D_x$, then $g(k_y, y) \leq g(k_x, x) = k$, so we construct the special time after $k_x D_x$).

Let k_q be $\lfloor (k_y D_y)/D_q \rfloor$, so $(k_y D_y)/D_q \geq k_q$, where "=" holds if and only if $q = y$. Because $k_y D_y > k_x D_x$ and $\lfloor (k_y D_y)/D_q \rfloor \geq \lfloor (k_x D_x)/D_q \rfloor$, then there must be an integer, $z \geq 0$, holding $\lfloor (k_x D_x)/D_q \rfloor = k_q - z$. So we have $(k_x D_x)/D_q < k_q - z + 1$ and $(k_x D_x) < k_q D_q - (z-1)D_q$.

If $q \neq y$, and $z \geq 1$, we have $k_y D_y > k_q D_q > k_x D_x$, which contradicts $k_y \times D_y = \min_{1\leq i\leq N} \{k_i D_i > k_x D_x\}$, thus $z = 0$ and $\lfloor (k_y D_y)/D_q \rfloor = \lfloor (k_x D_x)/D_q \rfloor$.

If $q = y$, we have $k_y = k_q$ and $(k_x D_x) < k_y D_y - (z-1)D_y$. Then, z must be 1 and $\lfloor (k_y D_y)/D_q \rfloor$ must equal $\lfloor (k_x D_x)/D_q \rfloor + 1$. Therefore, we have $g(k_y, y) - k = \sum_{1\leq q\leq N} \left(\lfloor (k_y D_y)/D_q \rfloor - \lfloor (k_x D_x)/D_q \rfloor \right) = 1$, $g(k_y, y) = k + 1$ and $\text{Task}_{\,k+1} \in L_y \subset \bigcup_{1\leq i\leq N} L_i$.

Because both the basis and inductive steps have been performed by mathematical induction, $\text{Task}_k \in \bigcup_{1\leq i\leq N} L_i$ holds for all $k \leq M$. Therefore, $\bigcup_{1\leq i\leq N} L_i = L$.

Because $k_y D_y = \min_{1\leq i\leq N} \{k_i D_i | k_i D_i > k_x D_x,\ k_i = 1, 2, \ldots .M\}$ and $k_y D_y \neq k_m D_m$ for any $m \neq y$, $\text{Task}_{\,k+1}$ is only contained in L_y. Therefore, $L_i \bigcap L_j = \varnothing$ holds for any $i \neq j \leq N$. From there, we may conclude that $L_1, L_2, L_3, \cdots L_N$ is a set partition of L, and $T_k = T_{g(k_x,x)} = k_x D_x < k_y D_y = T_{g(k_y,y)} = T_{k+1}$. □

Based on Proposition 1, if $L_1, L_2, L_3, \cdots L_N$ is a set partition of L, then the tasks assigned to different users will not overlap with each other, and all tasks are assigned. It is obvious that C_k equals k if $T_k \leq T_{k+1}$ for any $1 \leq k < M$.

Proposition 1 presents our basic concept for solving the disorder problem in theory. However, heterogeneous wireless networks can be very dynamic. The complexity of the environment makes it challenging to implement our basic concept in practice. In cellular networks, a cellular tower (base station) serves a massive number of users, and users outside the cooperative downloading system can contend for the service of the base station anytime, resulting in dynamic download rates and unstable task delay. Moreover, the coverage of the local network is small (because of the limitation of smart phones). Therefore, users may leave the cooperative downloading system at any time. We propose an adaptive disorder-avoidance cooperative downloading method that takes into consideration of all these challenges in the next section.

5. Adaptive disorder-avoidance cooperative downloading method

We propose an adaptive cooperative downloading method for solving the disorder problem in this section. The proposed method consists of two parts: (1) an adaptive task dissemination algorithm and (2) a task delay prediction mechanism. The algorithm is the core of the proposed method, and the task delay prediction mechanism assists the method to enhance the performance of the algorithm in dynamic download rate scenarios.

After the cooperative downloading starts, the primary user assigns a task to each cooperative user, and the index of the task is calculated based on the adaptive task dissemination algorithm. Once the task is finished, the primary user assigns a new task to the cooperative user based on the same algorithm. This process continues until all the tasks are assigned.

There are three states for each task: UNASSIGNED, ASSIGNED, and FINISHED. The initial state is UNASSIGNED. When the task is assigned to a cooperative user, its state changes to ASSIGNED. The cooperative user may fail to transmit the segment back to the primary user because of the mobility. Therefore, if U_i cannot finish his/her task in $D_j + \delta$, then the state of the task returns to UNASSIGNED again.

5.1. Adaptive task dissemination algorithm

When the primary user needs to assign a task to a cooperative user, the adaptive task dissemination algorithm is applied. Proposition 1 presents the basic idea for solving the data disorder in theory. To increase the adaptability of the proposed method and decrease calculation errors in real scenarios with dynamic task delay and unstable cooperative users, we derive a recursive version of Proposition 1. In each step, the result is calculated based on the parameters in real time and predicted task delay, which is obtained from the task delay prediction mechanism. Eqs. (8) and (9) can be rewritten as

$$g'(T) = \sum_j \lfloor T/D_j \rfloor \tag{10}$$

$$L_i = \{\text{Task}_{g'(T)} | T = D_i, 2D_i, \cdots, \text{AND } g'(T) \leq M\} \tag{11}$$

From the perspective of physical significance, $g'(T)$ is the number of tasks finished before T. Assuming that U_i is the cooperative user requesting a new task, the current time is t, and $T = t + D_i$ is the expected time that U_i finishes his/her next task, $k = g'(T)$ can be calculated as

$$\begin{aligned} g'(T) &= \sum_j \lfloor (T)/D_j \rfloor \\ &= \sum_j \left(\left\lfloor \frac{T - \lfloor t/D_j \rfloor D_j}{D_j} \right\rfloor + \lfloor t/D_j \rfloor \right) \\ &= \sum_j \left\lfloor \frac{T - \lfloor t/D_j \rfloor D_j}{D_j} \right\rfloor + \sum_j \lfloor t/D_j \rfloor \\ &= \sum_j \left\lfloor \frac{T - \lfloor t/D_j \rfloor \times D_j}{D_j} \right\rfloor + g'(t) \end{aligned}$$

Here, $\lfloor t/D_j \rfloor$ is the number of tasks that have been finished by U_j before t, and $P_j = \lfloor t/D_j \rfloor D_j$ is the time when U_j finished these tasks. Hence, $g'(T)$ can be obtained as follows:

$$g'(T) = \sum_j \lfloor (T - P_j)/D_j \rfloor + g'(t) \tag{12}$$

Eq. (12) is a recursive version of (10), where $g'(t)$ is the number of tasks that have been finished by all cooperative users until t. Because the current time is t, the accurate values of P_j and $g'(t)$ are already known. Time T and the prediction result of D_j can be obtained from the task delay prediction mechanism in Section 5.2. In addition, D_j and the number of users are assumed to be constants over the period from P_j to T. By reducing the amount of time over which they are assumed to be constants, the calculation of $g'(T)$ can be more accurate.

The algorithm for task dissemination is shown in Algorithm 1.

Algorithm 1 Adaptive task dissemination

```
Require: i , identification of the cooperative user
Ensure: k, the index of the task to be assigned
 1: /*Get the expected completion time of next task.*/
 2: $T \leftarrow GetCurrentTime() + D_i$
 3: /*Get the number of tasks that have been finished.*/
 4: $k \leftarrow 1$
 5: while $Task(k).state$ ==FINISHED do
 6:     $k \leftarrow k + 1$
 7: end while
 8: /*Get the target task according to Eq. (10) */
 9: for $j = 1 \rightarrow N$ do
10:     $Q[j] = \lfloor (T - P_j)/D_j \rfloor$
11:     if $Q(j) == (T - P_j)/D_j - \sigma$ then
12:         $Q(j) = Q(j) - 1$
13:     end if
14: end for
15: $k = k + sum(Q)$
16: /* Solve collisions among cooperative users */
17: while $k > M$ AND $Task(k).state$ ==ASSIGNED do
18:     $k \leftarrow k + 1$
19: end while
20: if $k > M$ then
21:     return $-1$
22: else
23:     return $k$
24: end if
```

Line 2 determines T, and Lines 4–7 obtain $g'(t)$. Lines 9–16 determine $g'(T)$ according to (12). It is possible that different cooperative users may obtain the same expect completion time, and therefore will be assigned the same task, so Lines 17–19 use the "first-come, first-assigned" strategy to solve the collision. If the download rate of the cooperative user is too low, the k obtained after Line 19 may be larger than M. In this case, -1 is returned, and the primary user cancels the cooperative relationship with U_i. This strategy can prevent users with very low download rates from slowing down the overall download process.

5.2. Dynamic task delay prediction mechanism

This mechanism assists the task dissemination algorithm. In the task dissemination algorithm, it is necessary to predict the time when a cooperative user will finish his/her next task. Task delay consists of two parts, as analysed in Section 4.1. In practice, download rates are slow and unstable in cellular networks. Therefore, the main causes of dynamic task delay are the dynamic download rates. We predict download rates according to the history download information using a neural network learning based moving average model, and then we use the predicted result to obtain the task delay. Accordingly, the mechanism consists of three basic functions: ***Initialization, Training***, and ***Prediction***. We first present the basic functions, and then introduce an additional function, ***Inheritance***, to reduce the learning time. The complexity of the mechanism is then analysed.

The download rate prediction model is:

$$\hat{R}_{i,n_i} = b_i + \sum_{j=1}^{q} w_{i,j} R_{i,n_i-j} \tag{13}$$

Here, n_i is the number of tasks finished by U_i. R_{i,n_i} is the average download rate when U_i downloads his/her n_ith segment. $\hat{R}_{i,n_i}$ is the prediction value of R_{i,n_i}, and q is the order of the prediction model. When U_i joins, the primary user initializes a prediction model for U_i using the ***Initialization*** function. ***Training*** and ***Prediction*** are called after U_i finishes a task.

In ***Training***, $w_{i,j}$ and b_i are adjusted automatically according to the error of the last prediction. The revised parameters are then used to predict the task delay in ***Prediction***.

(1). ***Initialization***: The initial values of the parameters are $b_i = 0, w_{i,1} = 1, w_{i,j} = 0$, for $1 < j \le q$. Next, n_i is set to 1, and $\hat{R}_{i,1}$ is set according to the official network performance reports released by the ISPs. We note that at the beginning of the cooperative downloading, n_i may be less than q, which makes (13) hard to apply, so $R_{i,-q} = \cdots = R_{i,0} = 0$ is introduced. Because the initial values of the parameters are set as $w_{i,1} = 1$ and $w_{i,j} = 0$ for $1 < j \le q$, then $R_{i,-q}, \cdots, R_{i,0}$ will not interfere with the result of the model.

(2). ***Training***: The error of the last prediction is $\hat{R}_{i,n_i} - R_{i,n_i}$, $E = \frac{1}{2}(\hat{R}_{i,n_i} - R_{i,n_i})^2$ is the mean square error, and η is the learning rate of the prediction model. The parameters are adjusted as below:

$$\begin{aligned} w_{i,j} &= w_{i,j} - \Delta_{w_{i,j}} \\ b_i &= b_i - \Delta_{b_i} \\ \Delta_{w_{i,j}} &= \eta \frac{\partial E}{\partial w_{i,j}} = \eta(\hat{R}_{i,n_i} - R_{i,n_i}) R_{i,n_i-j} \\ \Delta_{b_i} &= \eta \frac{\partial E}{\partial b_i} = \eta(\hat{R}_{i,n_i} - R_{i,n_i}) \end{aligned} \tag{14}$$

(3). ***Prediction***: $\hat{R}_{i,n_i+1}$ can be calculated as $b_i + \sum_{j=1}^{q} w_{i,j}\ R_{i,n_i+1-j}$. Combined with (7), the task delay of the $n_i + 1$ th task of U_i becomes $D_i = S/\hat{R}_{i,n_i+1} + \hat{\varepsilon}$, and n_i is increased by 1.

By applying the ***Training***, we can decrease prediction errors step-by-step. However, because of mobility, the encounter time between users is limited. Models of new users may not have enough time to converge, and some converged models are abandoned because their corresponding users have left. To solve this problem,

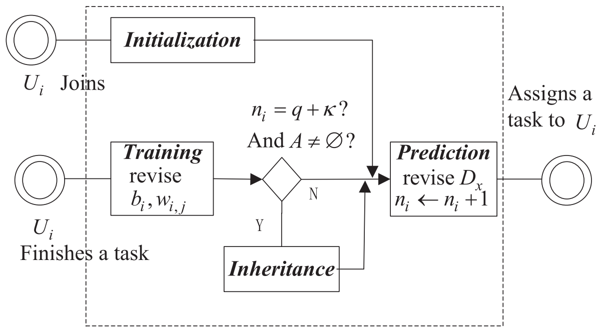


Fig. 6. Task delay prediction mechanism

we propose the ***Inheritance*** function. As explained at the beginning of this paper, different ISPs and levels of services are the main reasons for different download rates, and users of the same group probably have similar trends. Therefore, we classify users into groups according to their ISPs and levels of services. When a new user joins, such as U_i, let A be the set of the cooperative users in the same group with U_i. We use ***Inheritance*** to selects a user from A, whose download rate trend coincides best with U_i, and let U_i inherit the model of the selected user.

(4). ***Inheritance***: We define the consistency of the download rate trends of U_i and $U_x \in A$ as the cumulative estimation error:

$$E(n_i, x) = \sum_{k=q}^{n_i} \left| R_{i,k} - \hat{R}_{i,k}(x) \right|$$
$$\hat{R}_{i,k}(x) = b_x + \sum_{j=1}^{q} w_{x,j} R_{i,k-j} \tag{15}$$

Here, $\hat{R}_{i,k}(x)$ is the prediction result if we let U_i inherit the model of U_x and $\hat{R}_{i,n_i}(i) = \hat{R}_{i,n_i}$. In addition, $E(n_i, x)$ is the cumulative prediction error. Let U_y be the user with the minimum cumulative estimation error, $y = \min\{E(n_i, x) | U_x \in A,\ n_i < n_x\}$, and $E(n_i, y) < E(n_i, i)$. We then set $w_{i,j}$ to $w_{y,j}$ for $1 \le j \le q$, and set b_i to b_y.

It is obvious that not all models in A are suitable for inheritance, only those who have been trained more times than U_i. Hence, we add $n_i < n_x$ as a condition. Furthermore, when a cooperative user leaves, the information of its model will be deleted if it is not the last cooperative user in his/her group. Therefore, $A = \varnothing$ if and only if U_i is the first cooperative user in his/her group. We keep a model in A because it probably still works well even though it has not been trained for a while, so that the mechanism can work better in the mobile environment. Meanwhile, there is a possibility that no models in A are suitable, so we add the condition $E(n_i, y) < E(n_i, i)$.

The final working process of the task delay prediction mechanism is shown in Fig. 6. ***Inheritance*** is called when $A \neq \varnothing$ and n_i equals $q + \kappa$. ***Inheritance*** is not performed right after ***Initialization*** because we need to gather some history data to understand the download rate trends.

The complexity of the mechanism involves three parts. ***Initialization*** is called when a cooperative user joins and the number of calculations is constant, so the complexity of this part is O(N). After a cooperative user finishes a task, ***Training*** and ***Prediction*** are always called. Therefore, the complexity of this part is O(M). The number of calculations is O(N) every time ***Inheritance*** is called, and the number of calls is less than N because it is called once at most for each user. Hence, the total complexity of the mechanism is O($N^2 + M$).

5.3. Theoretical analysis of performance

Prediction errors for the task delay cannot be avoided, and this can impact the performance of our method. In this section, we compare the *C-I difference* before and after applying our proposed method while considering the prediction errors.

The current time is t, and we assume that the primary user assigns a task to U_i without loss of generality. According to (12), Task $g'(T)$ is assigned. It is clear that $\lfloor (T - P_j)/D_j \rfloor = (T - P_j)/D_j - \theta'_j$, where θ'_j is a random variable with a distribution of $U(0, 1)$. Furthermore, because $0 < t - P_j < D_j$, then $(T - P_j)/D_j = D_i/D_j + \theta''_j$, where $0 < \theta''_j < 1$. We assume that θ''_j is also a random variable following $U(0, 1)$. Let $\theta_j = \theta''_j - \theta'_j$, then we can obtain $\lfloor (T - P_j)/D_j \rfloor = D_i/D_j + \theta_j$. Because $\mathrm{E}[\theta_j] = 0$, we omit it to simplify the analysis and use D_i/D_j to approximate $\lfloor (T - P_j)/D_j \rfloor$. Users are classified into groups, similarly to the previous section, and we let N_{group} be the number of groups. We analyse *C-I difference* under a simplified scenario, where users of the same group have the same download rate. We denote the predicted download rate of Group k as $\hat{G}_k$ (corresponding to $\hat{R}_{i,n_i}$ of the previous section), and let $\hat{N}_k$ be the number of cooperative users in Group k at t. According to (12), $g'(T)$ is calculated as:

$$\begin{aligned} g'(T) &= \sum_{j=1}^{N} \lfloor (T - P_j)/D_j \rfloor + g'(t) \\ &\approx \sum_{j=1}^{N} (D_i/D_j) + g'(t) \\ &= g'(t) + \sum_{k=1}^{N_{\text{group}}} \hat{N}_k \left(\frac{S/\hat{G}_i + \hat{\varepsilon}}{S/\hat{G}_k + \hat{\varepsilon}} \right) \end{aligned}$$

Let φ denotes the completion order of Task $g'(T)$. This order is decided by the real parameters. Let G_k be the real download rate of Group k in the period from t to T, and let N_k be the real number of cooperative users in Group k from t to T. We then can obtain

$$\varphi = g'(t) + \sum_{k=1}^{N_{\text{group}}} N_k \left(\frac{S/G_i + \varepsilon}{S/G_k + \varepsilon} \right) \tag{16}$$

Here, $\hat{G}_i$, $\hat{N}_k$ and $\hat{\varepsilon}$ are the expected values of G_i, N_k and ε, respectively. Therefore, we assume that $\varepsilon \sim N(\hat{\varepsilon}, {\delta_0}^2)$, $G_k \sim N(\hat{G}_k, {\delta_k}^2)$, and $N_k \sim N(\hat{N}_k, {\delta'_k}^2)$ for $k = 1, 2, \cdots, N_{\text{group}}$. Because $g'(t)$ is a constant, we only need to focus on the second part of (16). Let φ_{u} and φ_{d} be the upper and lower bounds, respectively, of the second part. According to the 3σ rule of normal distribution, we can determine φ_{d} and φ_{u} as follows.

$$\begin{aligned} \varphi_{\text{d}} &= \sum_{k=1}^{N_{\text{group}}} (\hat{N}_k - 3\delta'_k) \left(\frac{\frac{S}{(\hat{G}_i + 3\delta_i)} + \hat{\varepsilon} - 3\delta_0}{\frac{S}{(\hat{G}_k - 3\delta_k)} + \hat{\varepsilon} + 3\delta_0} \right) \\ \varphi_{\text{u}} &= \sum_{k=1}^{N_{\text{group}}} (\hat{N}_k + 3\delta'_k) \left(\frac{\frac{S}{(\hat{G}_i - 3\delta_i)} + \hat{\varepsilon} + 3\delta_0}{\frac{S}{(\hat{G}_k + 3\delta_k)} + \hat{\varepsilon} - 3\delta_0} \right) \end{aligned} \tag{17}$$

If the disorder-avoidance algorithm is not applied, tasks will be assigned one-by-one, as shown in Fig. 4. The index of the task assigned to U_i will be $g'(t)$, and the *C-I difference* is $\varphi - g'(t)$. If our algorithm is applied, the task assigned to U_i will be Task $g'(T)$, and the *C-I difference* will be $\left| g'(T) - \varphi \right|$. Hence, we can obtain the *C-I difference* ratio as $\left| g'(T) - \varphi \right| / (\varphi - g'(t))$. It is obvious that $g'(t) + \varphi_{\text{d}} < g'(T) < g'(t) + \varphi_{\text{u}}$, and the upper bounds of the ratio can be obtained as:

$$\frac{|g'(T)-\varphi|}{\varphi-g'(t)} < \{(\varphi_u-\varphi_d)/\varphi_d, g'(T) \\ \geq \varphi(\varphi_u-\varphi_d)/\varphi_u, g'(T) < \varphi \tag{18}$$

Furthermore, let ς be $\max\limits_{1\leq i\leq N_{\text{group}}} \{3\delta_i/\hat{G}_i\}$, and let ζ be $\max\limits_{1\leq k\leq N_{\text{group}}} \{3\delta_k{}'/\hat{N}_k\}$, the maximum estimation error. We further assume that $3\delta_0/\hat{\varepsilon} \leq \varsigma$, so φ_d and φ_u can be rewritten as:

$$\begin{aligned} \varphi_d &> \varphi_m(1-\zeta)(1-\varsigma)/(1+\varsigma) \\ \varphi_u &< \varphi_m(1+\zeta)(1+\varsigma)/(1-\varsigma) \end{aligned} \tag{19}$$

where $\varphi_m = \sum_{k=1}^{N_{\text{group}}} \hat{N}_k\left(\frac{S/\hat{G}_i+\hat{\varepsilon}(1-\zeta^2)}{S/\hat{G}_k+\hat{\varepsilon}(1-\zeta^2)}\right)$. Substituting (19) into (18), we can determine the upper bounds for the difference ratio.

$$\frac{|g'(T)-\varphi|}{\varphi-g'(t)} < \left\{\frac{(1+\varsigma)^2(1+\zeta)}{(1-\varsigma)^2(1-\zeta)} - 1, g'(T) \\ \geq \varphi 1 - \frac{(1-\varsigma)^2(1-\zeta)}{(1+\varsigma)^2(1+\zeta)}, g'(T) < \varphi \right. \tag{20}$$

It is obvious that the upper bound when $g'(T) < \varphi$ is lower than the bound when $g'(T) \geq \varphi$, which implies that the maximum *C-I difference* probably is higher if $g'(T) \geq \varphi$. Hence, if we decrease $g'(T)$ (making it approximately $\varphi_d + g'(t)$), then $g'(T) < \varphi$ almost holds, and the smaller upper bound can be taken as the upper bound of our algorithm. In fact, we consider this in the implementation of our algorithm, and we decrease $g'(T)$ by introducing the σ in Line 11 of Algorithm 1.

6. Simulation

The goal of our simulation is to test the performance of the prediction model proposed in Section 5.2 and our proposed disorder-avoidance method. The first part of the simulation is conducted using a dataset collected from the real world, and the second part is performed using NS-2 (Network Simulation version 2).

6.1. Prediction model test

We use a dataset[1] to test the performance of the prediction mechanism. Because the core of the mechanism is the download rate prediction model, we focus on the performance of the model in this section. There are 10 sets of data available in the dataset, and we use four of them, called “nyupoly-dash-1,2,3,4” (User 1, 2, 3, and 4). The field “empiricalRate_bps” of the dataset records the download rate of each segment.

Fig. 7 shows the download rates of users. They are in the same group, and it is obvious that they have similar trends (which support our idea that users of one group can inherit a model from each other). Accordingly, we first present the estimation errors when a user (User 2) is the first user in the group ($A = \varnothing$), and then we present the estimation errors when a user (User 3) joins later than the other users ($A = \{\text{User 1}, \text{User 2}, \text{User 4}\}$). Next, we present the effect of the order (q) and the learning rate (η) on the performance. Figs. 8–10 show the results. The estimation (prediction) errors are normalized as $\left|\frac{\hat{R}_{i,n_i}-R_{i,n_i}}{R_{i,n_i}}\right| \times 100\%$, and average estimation errors are obtained as $\frac{1}{M}\sum_{n_i=1}^{M}\left|\frac{\hat{R}_{i,n_i}-R_{i,n_i}}{R_{i,n_i}}\right| \times 100\%$.

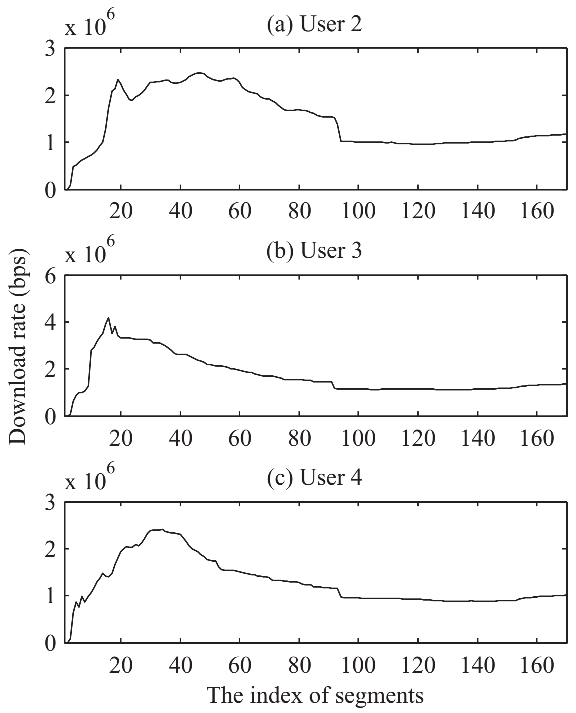


Fig. 7. Download rates of users in the dataset.

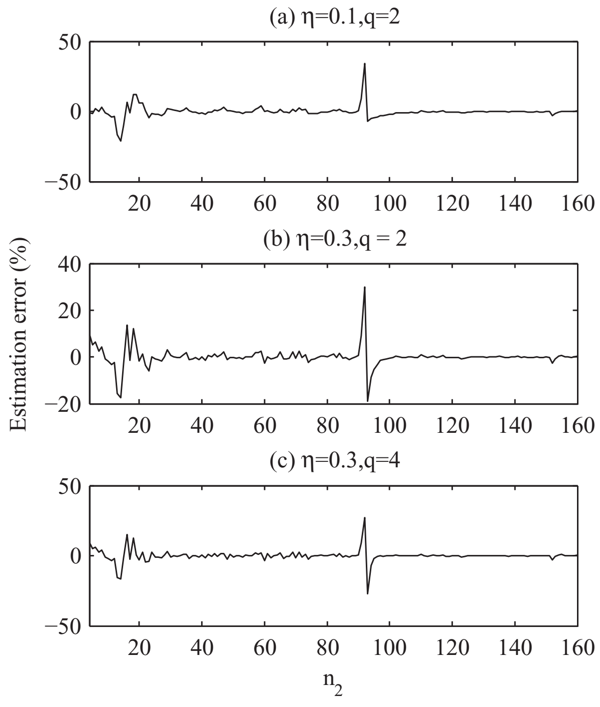


Fig. 8. Estimation errors for User 2 with $A = \varnothing$.

Fig. 8 shows the estimation errors for User 2. After the download rate prediction model of User 2 is initialized, it has to be trained on itself because $A = \varnothing$. Its estimation errors first fluctuate for a while, and then tend to 0. This is because ***Training*** is

[1] The data were collected over the GENI WiMAX networks installed on the campuses of the Polytechnic Institute of NYU and University of Massachusetts Amherst. Each installation includes a commercial WiMAX base station operating in a licensed frequency band, as well as other components required to route traffic from WiMAX clients to the Internet or other networks. More information can be found in [5].

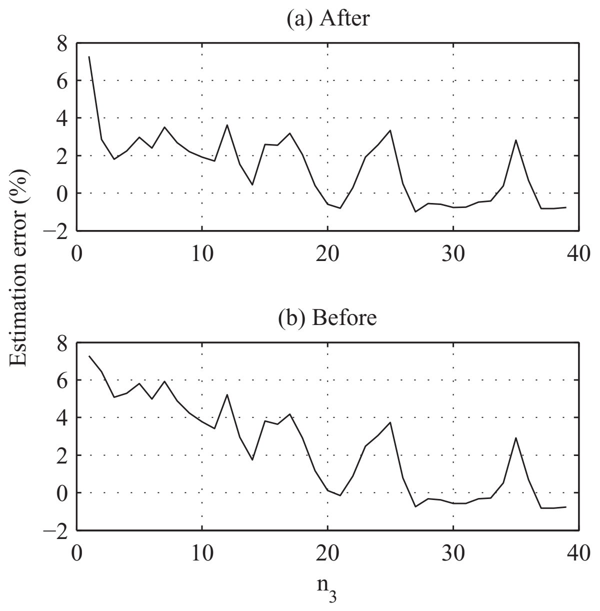


Fig. 9. Estimation errors for User 3, $\eta = 0.05, q = 2, \kappa = 2, A = \{\text{User 1}, \text{User 2}, \text{User 4}\}$.

called on every segment, and the model tends to be more accurate as n_2 increases. There is an outlier at $n_2 = 90$. This is because the trend of the download rate changes suddenly after $n_2 = 90$. The download rates of users can be found in Fig. 7, which first increase and then decrease. After that, they become stable. The actual parameter values of the model in the stages are different, so at the beginning of each stage, outliers of the estimation error appear. After the outliers, the model adjusts itself using the neural network learning, and the estimation errors tend to 0 again.

Fig. 9 shows the estimation errors for User 3 after and before the application of ***Inheritance***. In Fig. 9(b), the ***Inheritance*** method is not used, and the estimation errors vary in a similar way to those in Fig. 8. In Fig. 9(a), the ***Inheritance*** procedure is called when $n_3 = q + \kappa$. It can be seen that the estimation errors decrease faster than in Fig. 9(b) because we let User 3 inherit the model of a user in A, who has a similar download rate trend. Most importantly, the selected model has been trained for a longer time than the model of User 3. Additionally, ***Inheritance*** is called when $n_3 = q + \kappa$, so the estimation errors in Figs. 9(a) and (b) are the same when $n_3 < q + \kappa$.

Fig. 10 shows the effect of q and η on the average estimation errors of the prediction model. The results are averaged for Users 1, 2, 3, and 4. The average estimation errors first decrease with η, as shown in Fig. 10. Here, η is the learning rate, which can influence the sensitivity of the prediction model. The higher the learning rate is, the more sensitive the model becomes. This is the reason why the average estimation errors decrease with η when it is relatively small. However, if the learning rate is too high, the model becomes too sensitive or even unstable. Fig. 10 shows that once a certain learning rate has been reached, the average estimation errors begin to increase substantially. Therefore, η should be restricted within a reasonable range, and the appropriate interval for η is 0 to 0.4. The order of the prediction model q can also influence the performance of the prediction model. The model obtains the minimum average estimation error when $q = 5, \eta = 0.33$, as shown in Fig. 10, which is about 2.6%. When $\eta < 0.4$, increasing q can slightly increase performance. However, when $\eta > 0.4$, increasing q risks huge estimation errors, as shown in Fig. 10. Therefore, q should also be carefully chosen. Furthermore, to find the most appropriate q, we average the estimation errors on the learning rate for each q based on the results. The average estimation error is 3.23% for $q = 1$, and 3.03% for $q = 2$. Therefore, the most appropriate value for q is 2.

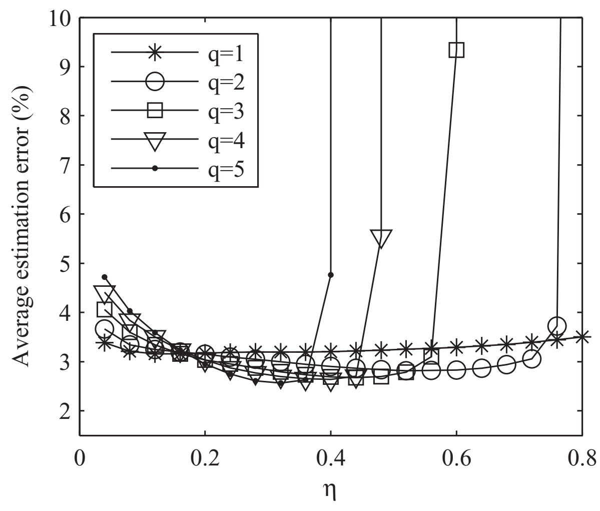


Fig. 10. Average estimation error.

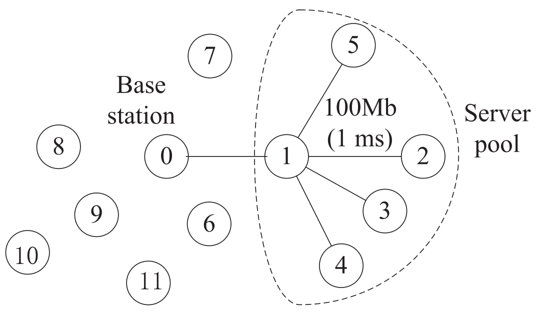


Fig. 11. Network layout of the simulation.

6.2. Comparison

We compare our adaptive disorder-avoidance cooperative downloading method (ADCD) with the CCF in [28], PatchPeer [27], the dynamic dissemination method (DD) [4], and the package fragmentation method (PF) in [34]. We note that PF was proposed for multipath, not cooperative downloading. We adapt the basic idea of PF to cooperative downloading, and use it as comparison. PF assigns tasks to users according to their task delay in a static way. It can be seen as a simplified version of our proposed method, where the disorder problem is not considered, and the dynamic task delay prediction mechanism is not used.

6.2.1. Simulation setting

The network for the simulation is heterogeneous. Its layout is presented in Fig. 11.

The base station connects wireless nodes with the server gate, which connects servers (Nodes 2–5). When a wireless node starts to download data, it selects a server at random, and downloads data via the base station using HTTP. We imitate the differences of users by controlling the data sending rates at the servers. Because the cellular network and local network work at different frequencies, we add multiple channels to NS-2. The cellular network works on Channel 1, and the local network works on Channel 2. There is no interference between the channels. The period for cooperative user detection is set at 5 s for each method, and the period for bandwidth probing is set to 10 s for PF and CCF [28]. Other parameters are listed in Table 1.

Table 1
Parameter settings for the wireless network.

Parameters	Value
Number of tasks, *M*	50
Size of each segment	200 KB
Size of simulation scenarios	$100 \times 100m^2$
Propagation model	TwoRayGround
Mac interface	Mac 802_11
Ad hoc routing protocol	AODV
Transmission rate of the base station	100 Mb
Transmission rate of wireless nodes	50 Mb
Transmission range of Channel 1	100 m
Transmission range of Channel 2	25 m

6.2.2. Evaluation metrics and simulation scenarios

Our evaluation uses two metrics, MUD and download time, defined as follows:

- MUD: Let t_k be the time that all tasks prior to Task$_k$ are downloaded, including Task$_k$. Then $t_i = \max\{T_j | 1 \le j \le i\}$, and the amount of time that Task$_k$ has been waiting is $t_i - T_i$, therefore, the average MUD is

$$MUD = 1/M \sum_{i=1}^{M} |t_i - T_i| \tag{21}$$

- Download time: the average amount of time needed to download the target file.

We observe the effect of the download rate difference, dynamic download rate, and mobility on the performance of the methods. The simulation scenarios are:

(1) Download rate difference: The number of cooperative users was fixed at 20. The download rates for the cooperative users were uniformly distributed in a range with a fixed upper limit of 400 kbps and a lower limit that varied from 280 kbps to 370 kbps less than the upper limit.

(2) Dynamic download rates: Download rates increased with time during the cooperative downloading. The interval of the changes was fixed at 10 s, and the increment was varied from 5 kbps to 50 kbps. In each test, there were 10 cooperative users. Users were divided in two groups. The first group was the slow download group (5 users), for which the initial download rates were uniformly distributed in the range of 20–40 kbps. The second group was the fast download group (5 users), for which the initial download rates were uniformly distributed in the range of 250–650 kbps.

The above scenarios were immobile, and users were uniformly distributed within the transmission range of the primary user.

(3) Mobility: The mobile scenarios were generated by the cmu-scen-gen module of NS-2. The pause time was varied from 0 to 50 s in increments of 5 s. The number of users was set at 60. The average moving speed was set as 1.3 m/s, which is the average moving speed of pedestrians [37]. Users were divided in two groups. The first group was the fast download group (30 users) with download rates fixed at 1400 kbps, which is the theoretical download rate of 3G China Unicom. The second group was the slow download group (30 users) with download rates fixed at 300 kbps, which is the theoretical download rate of 3G China Telecom.

Because CCF and PatchPeer are proposed for mobile scenarios, we only compare them with the other methods in (3). Furthermore, because they only select one user as the cooperative user, so they do not suffer from the disorder problem, and hence we only compare them with other methods in terms of download time and extra communication overhead.

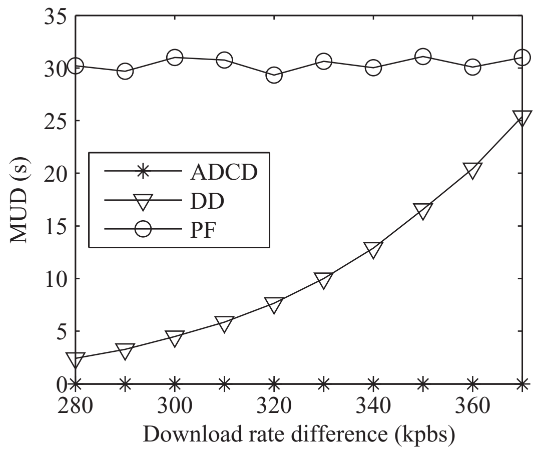


Fig. 12. Effect of download rate difference on MUD.

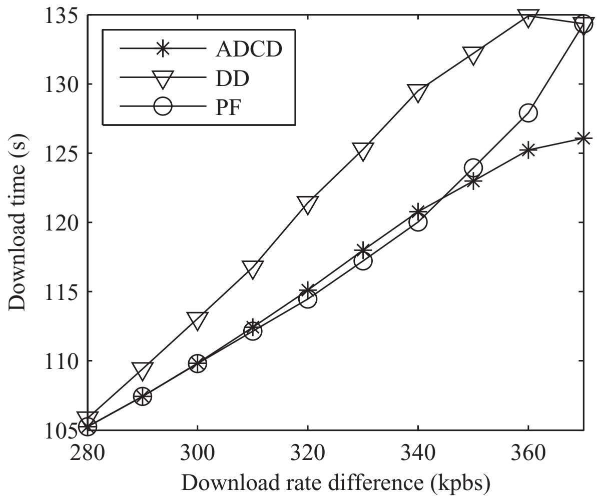


Fig. 13. Effect of download rate difference on download time.

6.2.3. Simulation results

(1) Download rate difference

The MUD of DD increases with the download rate difference, as shown in Fig. 12. Our method can decrease the MUD to 0. When the download rates are uniformly distributed, users are given different download rates and different task delay. DD can cause disorder in this case, as explained in Section 4.2. As the difference in the users increases, the task delay difference increases, which causes the MUD for DD to increase. We consider the differences in the users, and assign tasks according to their task delay, which makes our method adapt well to this difference.

Fig. 13 shows that the download time increases as the download rate difference increases for each method. This is because the average download rate decreases along with the difference according to the setting. DD consumes more time than both our method and PF. The reason is that there are some cooperative users with very low download rates in the system. Tasks assigned to these users may take much longer to be completed than other tasks, so these users can slow down the download process. Our method and PF can find these users based on their expected task completion time and cancel the cooperative relationship with them. This is why our method is better than DD in term of download time.

(2) Dynamic download rates

Figs. 14 and 15 show the effect of dynamic download rate on the performance of each method. The MUD of DD is stable at 18 s

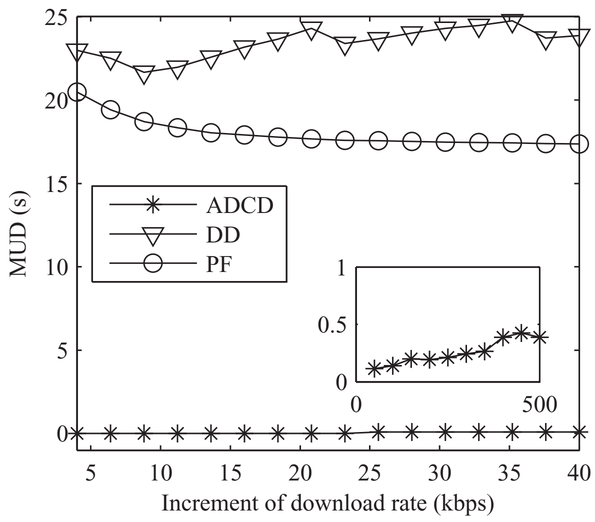


Fig. 14. Effect of download rate increments on MUDs.

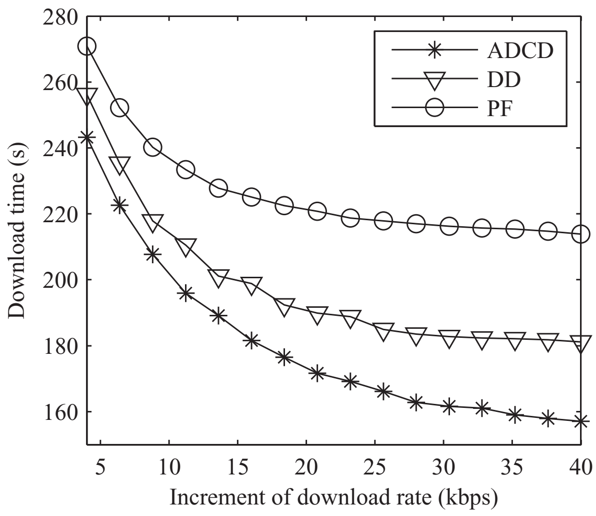


Fig. 15. Effect of download rate increments on download time.

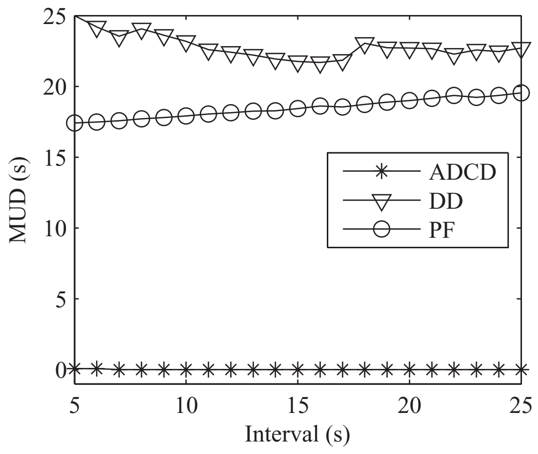


Fig. 16. Effect of download rate changing intervals on MUDs with the increments of download rates fixed at 20 kbps.

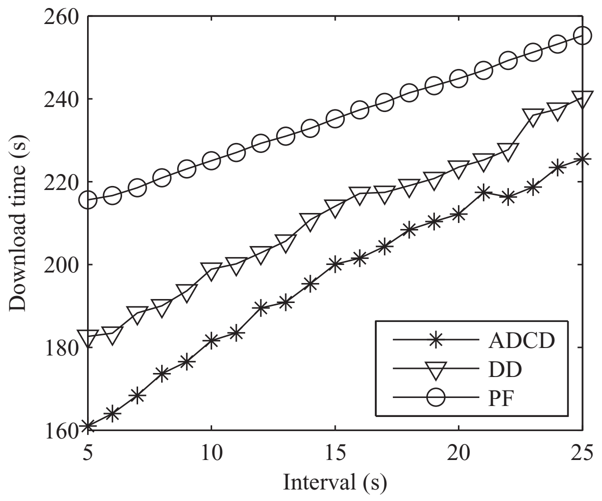


Fig. 17. Effect of download rate changing interval on download time with the increments of download rates fixed at 20 kbps.

and PF is stable at about 22 s. The MUD of our method is almost 0, as shown in Fig. 14. When the increment is small, the download rates change slowly, and the proposed dynamic task delay prediction mechanism can predict the download rates well. However, if the download rates change too fast, prediction errors occur. The errors can then cause calculation deviation of the dynamic task dissemination algorithm. This is the reason why the MUD of our method increases, as shown in the subfigure of Fig. 14.

Fig. 15 shows the effect of dynamic download rate on download time. DD uses 10% more time than our method. PF assigns all tasks to users at one time. This is a pre-assignment strategy, so it cannot adapt to dynamic download rate scenarios. DD uses a dynamic strategy, and this is the reason why it can adapt to the scenarios better than PF. However, DD cannot detect the users that slow down the download process, so its download time is longer than our method.

Furthermore, we vary the changing interval from 5 s to 25 s. Figs. 16 and 17 show the results. The results are similar with the previous test, except for the trend of the download time. Fig. 15 decreases, while Fig. 17 increases. Because the download rates increase with time, if they increase faster, then it will take less time to finish downloading. It is obvious that there are two ways to make the download rates increase faster: by increasing the increment, as shown in Fig. 15, or by decreasing the interval, as shown in Fig. 17. Therefore, the trends of Figs. 15 and 17 are different.

(3) Mobility

Figs. 18–21 show the effect of mobility on MUD. The maximum MUD of our method is 1 s, which is 85% less than DD, as shown in Fig. 18. The mobility of pedestrians is slow, so we assume that the number of cooperative users is stable over short periods. We then calculate the tasks to be assigned in a recursive way, as described in Section 5.1. The results shown in Fig. 18 illustrate the rationality of the assumption and the adaptive capacity of our method in mobile scenarios.

Figs. 19 and 20 show the effect of network density on MUD, where the number of nodes are 20, 40, and 60, respectively. The MUD of DD is presented in Fig. 19. Comparing Fig. 19 with Fig. 20, it can be seen that the MUD of DD is higher, and the reason is similar to that of scenario (2), “Dynamic Download Rates.” The MUD decreases slightly with the pause time for our method. As the pause time increases, the stability of the network increases, which improves the applicability of the assumption that the number of cooperative users is stable over short periods, and hence the MUD can be decreased. The mobility affects the MUD of our method indirectly because of the number of unstable cooperative users

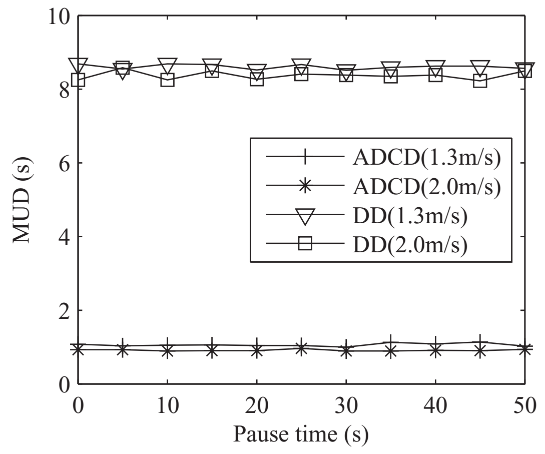


Fig. 18. Effect of pause time on MUD.

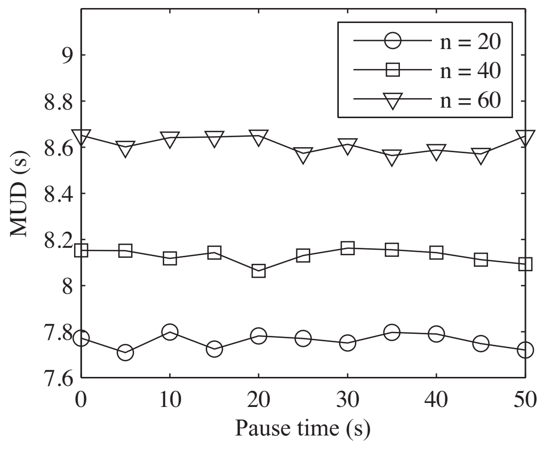


Fig. 19. Effect of pause time and network density on DD.

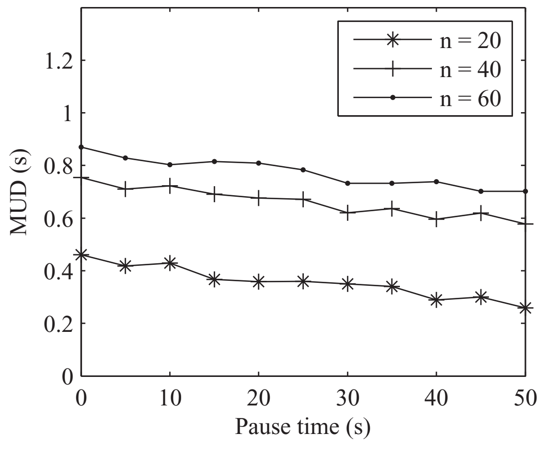


Fig. 20. Effect of pause time and network density on ADCD.

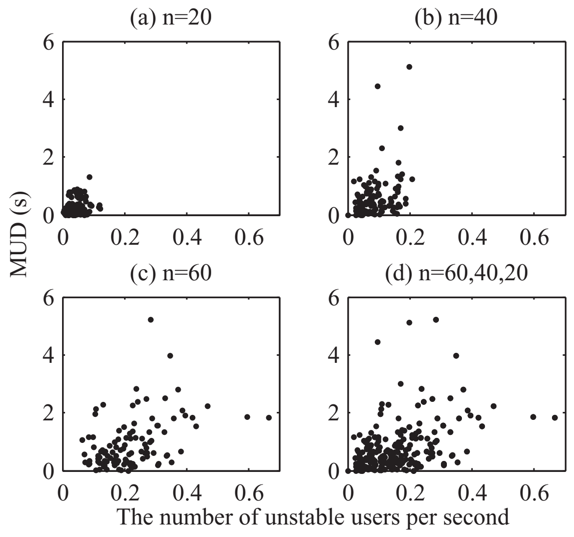


Fig. 21. Relation between MUD and unstable users using ADCD.

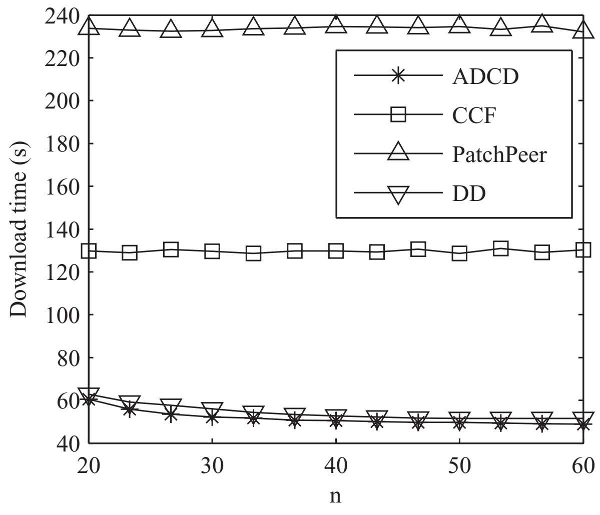


Fig. 22. Effect of network density on download time with the average moving speed fixed at 2.0 m/s and a pause time of 5 s.

(unstable cooperative users are the users that join or quit cooperative downloading halfway). Fig. 21 shows the MUD versus the number of unstable users. As the network density increases, the number of unstable users increases, which also explains why the MUD increases along with network density in Fig. 20.

Fig. 22 shows the effect of network density on download time. The most time consuming method is PatchPeer. CCF consumes less time than PatchPeer. Our method and DD take a similar amount of time, which is much less than that of PatchPeer and CCF. PatchPeer selects the user with the shortest Euclidean distance from the primary user as the cooperative user to increase the link quality between cooperative and primary users, so the transmission rate on the link is higher than in other methods. However, the download rate of the selected cooperative user is probably very low. Based on this observation, CCF selects the cooperative user with the consideration of both stability and bandwidth, and this is the reason why CCF consumes less time than PatchPeer. However, they only use the user with the best condition (stability or bandwidth) as the cooperative user, and the available download links of the other users are wasted. Our method and DD use multiple cooperative users, and can make full use of the download links. This is also the reason why the download time decreases with the number of users for the two methods.

Fig. 23 shows the effect of network density on the extra communication overhead of the four methods. We define extra communication overhead to be the number of packages used for supporting the system (packages for carrying segments are not included). It can be seen that CCF has the highest communication overhead, which increases with network density. DD and our method have the least communication overhead, which is almost constant. Communication overhead is proportional to download time and the number of one-hop neighbours. Because the download time decreases with network density for DD and our method,

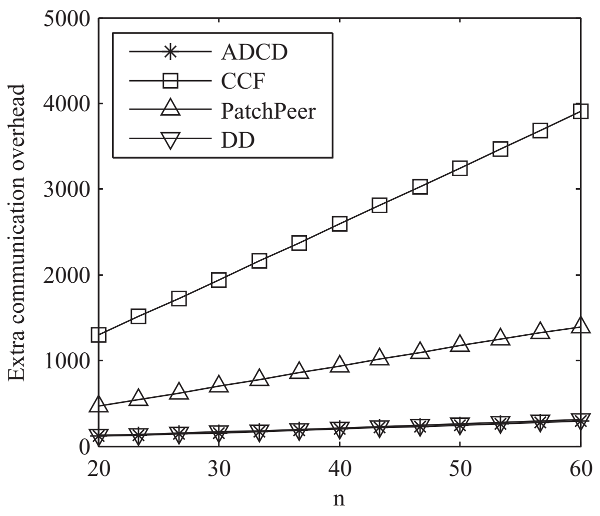


Fig. 23. Effect of network density on extra communication overhead. The average moving speed is set to 2.0 m/s and the pause time is set to 5 s.

the communication overhead does not increase with network density for these two methods. The download time is almost constant for CCF and PatchPeer, as can be seen in Fig. 22, so the communication overhead increases with network density. Furthermore, to select the user with the best stability and bandwidth, CCF needs users to send their locations and velocities periodically to the primary user. Besides, each user needs to probe their bandwidth every few seconds. PatchPeer does not have the bandwidth probing process, so it has less communication overhead than CCF, even though its download time is longer than that of CCF. DD and our method also do not have a bandwidth probing process, but the download time of these methods is less PatchPeer, so DD and our method has the least communication overhead. Above all, PatchPeer and CCF do not have disorder problem, but their download time and communication overhead are much higher than our method. DD and our method have similar communication overhead, but DD has the disorder problem as it can be seen from Fig. 19.

7. Conclusions

The data disorder problem in the cooperative downloading system increases the memory usage and decreases the quality of services. In this paper, we analyse the disorder problem and propose an adaptive disorder-avoidance cooperative downloading method that considers the dynamic features of wireless networks. We decrease the effect of unstable users on performance by applying recursion and propose a dynamic task delay prediction mechanism for enhancing the performance of our method in dynamic download rate scenarios. The simulation results show that our method can decrease the MUD by 85% and reduce the download time under various scenarios, including those with dynamic download rates and those with mobility. Furthermore, we only use direct neighbours as cooperative users in this work. Our future work will focus on solving the disorder problem when there are multiple hops between cooperative users and the primary user.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (60973027,61370212,61402127,61502118), the Fundamental Research Fund for the Central Universities (HEUCF100601, HEUCFZ1213), and the Research Fund for the Doctoral Program of Higher Education of China (20102304120012).

References

[1] A.O. Lim, Y. Kado, Deployment and experimental verification for data communication in heterogeneous wireless networks, in: 2011 7th International Conference on Information Technology in Asia (CITA 11), 2011, pp. 1–7, doi:10.1109/CITA.2011.5999529.
[2] T. Shiao-Li, W. Chen-Wei, L. Yun-Ciou, C. Ray-Guang, A dynamic load-balancing scheme for heterogeneous wireless networks, in: 2014 IEEE Wireless Communications and Networking Conference (WCNC), 2014, pp. 3070–3075, doi:10.1109/WCNC.2014.6952997.
[3] L. Militano, A. Iera, F. Scarcello, A fair cooperative content-sharing service, Comput. Netw. 57 (9) (2013) 1955–1973.
[4] Y. Tuo, Z. Zilong, Z. Da, W. Xinbing, L. Yunxin, L. Songwu, Indapson: an incentive data plan sharing system based on self-organizing network, in: INFOCOM 2014, 2014, pp. 1545–1553, doi:10.1109/INFOCOM.2014.6848090.
[5] F. Fraida, W. Cong, L. Yong, K. Thanasis, Z. Michael, P. Shivendra, Crawdad dataset nyupoly/video (v. 2014-05-09), 2014, 10.15783/C7W30R
[6] D. Gavidia, M. van Steen, A probabilistic replication and storage scheme for large wireless networks of small devices, in: 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008, pp. 469–476, doi:10.1109/MAHSS.2008.4660060.
[7] A. Nandan, S. Das, G. Pau, M. Gerla, M.Y. Sanadidi, Cooperative downloading in vehicular ad-hoc wireless networks, in: Second Annual Conference on Wireless On-demand Network Systems and Services, 2005, pp. 32–41, doi:10.1109/WONS.2005.7.
[8] S. Ahmed, S.S. Kanhere, Vanetcode: network coding to enhance cooperative downloading in vehicular ad-hoc networks, 2006. 10.1145/1143549.1143654
[9] M. Li, Z. Yang, L. Wenjing, Codeon: cooperative popular content distribution for vehicular networks using symbol level network coding, IEEE J. Sel. Areas Commun. 29 (1) (2011) 223–235, doi:10.1109/JSAC.2011.110121.
[10] Y. Qiben, M. Li, Z. Yang, L. Wenjing, Z. Hongqiang, Throughput analysis of cooperative mobile content distribution in vehicular network using symbol level network coding, IEEE J. Sel. Areas Commun. 30 (2) (2012) 484–492, doi:10.1109/JSAC.2012.120229.
[11] S. El Rouayheb, A. Sprintson, P. Sadeghi, On coding for cooperative data exchange, in: 2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo), 2010, pp. 1–5, doi:10.1109/ITWKSPS.2010.5503135.
[12] N. Milosavljevic, S. Pawar, S.E. Rouayheb, M. Gastpar, K. Ramchandran, Optimal deterministic polynomial-time data exchange for omniscience (2011). arXiv preprint arXiv:16108.6046 .
[13] N. Milosavljevic, S. Pawar, M. Gastpar, K. Ramchandran, Efficient algorithms for the data exchange problem under fairness constraints, in: 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2012, pp. 502–508, doi:10.1109/Allerton.2012.6483260.
[14] M. Gonen, M. Langberg, Coded cooperative data exchange problem for general topologies, IEEE Trans. Inf. Theory 61 (10) (2015) 5656–5669, doi:10.1109/TIT.2015.2457443.
[15] M. Felegyhazi, J.P. Hubaux, L. Buttyan, Nash equilibria of packet forwarding strategies in wireless ad hoc networks, IEEE Trans. Mobile Comput. 5 (5) (2006) 463–476, doi:10.1109/TMC.2006.68.
[16] D.R. Brown, F. Fazel, A game theoretic study of energy efficient cooperative wireless networks, J. Commun. Netw. 13 (3) (2011) 266–276, doi:10.1109/JCN.2011.6157436.
[17] J. Yang, I. Brown D. R., Energy efficient relaying games in cooperative wireless transmission systems, in: the Forty-First Asilomar Conference on Signals, Systems and Computers, 2007, pp. 835–839, doi:10.1109/ACSSC.2007.4487334.
[18] L. Militano, F. Fitzek, A. Iera, A. Molinaro, On the Beneficial Effects of Cooperative Wireless Peer-to-Peer Networking, Springer US, pp. 97–109.
[19] A. Iera, L. Militano, L.P. Romeo, F. Scarcello, Fair cost allocation in cellular-bluetooth cooperation scenarios, IEEE Trans. Wireless Commun. 10 (8) (2011) 2566–2576.
[20] M. Xiaomin, Z. Jinsong, W. Tong, Reliability analysis of one-hop safety-critical broadcast services in vanets, IEEE Trans. Veh. Technol. 60 (8) (2011) 3933–3946, doi:10.1109/TVT.2011.2165975.
[21] O. Trullols-Cruces, M. Fiore, J.M. Barcelo-Ordinas, Cooperative download in vehicular environments, IEEE Trans. Mobile Comput. 11 (4) (2012) 663–678, doi:10.1109/TMC.2011.100.
[22] H. Liang, J.Y. Le Boudec, M. Vojnoviae, Optimal channel choice for collaborative ad-hoc dissemination, in: INFOCOM 2010, 2010, pp. 1–9, doi:10.1109/INFCOM.2010.5462163.
[23] H. Zhou, B. Liu, T.H. Luan, F. Hou, L. Gui, Y. Li, Q. Yu, X. Shen, Chaincluster: engineering a cooperative content distribution framework for highway vehicular communications, IEEE Trans. Intell. Trans. Syst. 15 (6) (2014) 2644–2657, doi:10.1109/TITS.2014.2321293.
[24] M. Sardari, F. Hendessi, F. Fekri, Infocast: a new paradigm for collaborative content distribution from roadside units to vehicular networks, in: 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2009, pp. 1–9, doi:10.1109/SAHCN.2009.5168939.
[25] L. Ning, C. Nan, Z. Ning, S. Xuemin, J.W. Mark, Connected vehicles: Solutions and challenges, IEEE Internet Things 1 (4) (2014) 289–299, doi:10.1109/JIOT.2014.2327587.
[26] L. Al-Kanj, Z. Dawy, E. Yaacoub, Energy-aware cooperative content distribution over wireless networks: design alternatives and implementation aspects, IEEE Commun. Surv. Tut. 15 (4) (2013) 1736–1760, doi:10.1109/SURV.2012.121912.00052.

[27] T.T. Do, K.A. Hua, A. Aved, F. Liu, N. Jiang, Scalable video-on-demand streaming in mobile wireless hybrid networks, in: 2009 IEEE International Conference on Communications, 2009, pp. 1–6, doi:10.1109/ICC.2009.5199435.

[28] J. Shijie, X. Changqiao, G. Jianfeng, Z. Hongke, G.M. Muntean, A novel cooperative content fetching-based strategy to increase the quality of video delivery to mobile users in wireless networks, IEEE Trans. Broadcasting 60 (2) (2014) 370–384, doi:10.1109/TBC.2014.2322772.

[29] A. Jungmaier, E.P. Rathgeb, A novel method for sctp load sharing, in: 4th International IFIP-TC6 Networking Conference, 3462, Springer Verlag, 2005, pp. 1453–1456.

[30] J.R. Iyengar, P.D. Amer, R. Stewart, Concurrent multipath transfer using sctp multihoming over independent end-to-end paths, IEEE/ACM Trans. Netw. 14 (5) (2006) 951–964, doi:10.1109/TNET.2006.882843.

[31] M. Yabandeh, S. Zarifzadeh, N. Yazdani, Improving performance of transport protocols in multipath transferring schemes, Comput. Commun. 30 (17) (2007) 3270–3284, doi:10.1016/j.comcom.2007.02.017.

[32] R.K. Singh, N.S. Chaudhari, K. Saxena, Load balancing in ip/mpls networks: a survey, Commun. Netw. 02 (4) (2012) 151–156.

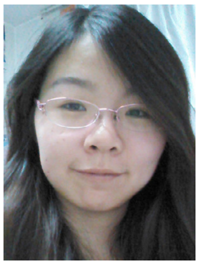
[33] K. Chebrolu, R.R. Rao, Bandwidth aggregation for real-time applications in heterogeneous wireless networks, IEEE Trans. Mobile Comput. 5 (4) (2006) 388–403, doi:10.1109/TMC.2006.1599407.

[34] J.-Y. Wu, X.-Q. Qiao, B. Cheng, J.-L. Chen, Y.-L. Sun, End-to-end qos guarantee for delay-sensitive mobile multimedia conferencing, Jisuanji Xuebao/Chinese J. Comput. 36 (7) (2013) 1399–1412, doi:10.3724/SP.J.1016.2013.01399.

[35] R. Ahola, A. Aktas, J. Wilson, K.R. Rao, F. Jonsson, T. Hakala, J. Hanze, M. Sanden, Y. Guo, T. Knuuttila, et al., A single chip cmos transceiver for 802.11 a/b/g wlans, in: 2004 IEEE International Solid-State Circuits Conference, IEEE, 2004, pp. 92–515.

[36] C.R. B, Introduction to Queuing Theory, 2nd ed, Elsevier, North-Holland, New York, pp. 77–78.

[37] O. Helgason, S.T. Kouyoumdjieva, G. Karlsson, Opportunistic communication and human mobility, IEEE Trans. Mobile Comput. 13 (7) (2014) 1597–1610, doi:10.1109/TMC.2013.160.



Xiuxiu Wen received her B.E. degree in computer science and technology from Harbin Engineering University (HEU), Harbin, China, in 2011. Now, she is studying for her Ph.D. degree at HEU. Her research interests involve system performance analysis and wireless self-organized networks. Recently, she focuses on designing efficient cooperative downloading methods under heterogeneous environments.

Guangsheng Feng received his M.E. degree from HIT in 2005, and received his Ph.D. degree from HEU in 2009. Now, he is engaged in teaching and researching as a lecturer at HEU. His research interests involve cross-layer design, information sensing, and wireless channel access control.

Huiqiang Wang received his received M.E. and Ph.D. degrees from HEU in 1985 and 2005, respectively. From 2001 to 2002, he was at Queen's University, Ontario, Canada, as a senior visiting scholar. Now, he is engaged in teaching and researching as a professor and a doctoral advisor at HEU. Up to now, he holds ten Chinese patents. His research interests involve network security, cognitive networks, and autonomic computing.

Hongwu Lv received his M.S. and Ph.D. degrees from HEU in 2009 and 2011, respectively. Now, he is engaged in teaching and researching as a lecturer at HEU. His research interests involve cognitive computing and dependability analysis.

Junyu Lin received his M.E. and Ph.D. degrees from HEU in 2009 and 2014, respectively. Now, he is engaged in teaching and researching as a lecturer at HEU. His research interests involve ubiquitous services and trustworthiness analysis.