IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications

Efficient Scheduling Algorithms for On-demand Wireless Data Broadcast

Zaixin Lu*, Weili Wu†‡, Wei Wayne Li§, and Miao Pan¶
*Dept. of Mathematics and Computer Science, Marywood University, Scranton, PA 18509, USA
Email: zlu@marywood.edu
†Taiyuan University of Technology, Taiyuan, China
‡Dept. of Computer Science, University of Texas at Dallas, Richardson, Texas 75080, USA
Email: weiliwu@utdallas.edu
§Dept. of Computer Science, Texas Southern University, Houston, Texas 77004, USA
Email: liw@tsu.edu
¶Dept. of Electrical & Computer Engineering, University of Houston, Houston, Texas 77204, USA
Email: mpan2@uh.edu

***Abstract*—On-demand wireless data broadcast is an efficient way to disseminate data to a large number of mobile users. In many applications, such as stock quotes and flight schedules, users may have to download multiple data items per request. However the multi-item request scheduling has not yet been thoroughly investigated for on-demand wireless data broadcasts. In this paper, we step-up on investigating this problem from viewpoint of theory and simulation. We develop a two-stage scheduling scheme to arrange the requested data items with the objective of minimizing the average access latency. The first stage is to select the data items to be broadcast in the next time period and the second stage is to schedule the broadcasting order for the data items selected in the first stage. We develop algorithms for the two stages respectively and analyze them both theoretically and practically. We also compare the proposed algorithms with other well known scheduling methods through simulation. The theoretical findings and simulation results reveal that significantly better access latency can be obtained by using our scheduling scheme rather than its competitors.**

I. Introduction

A. Background and Motivation

With the rapid growth of mobile data services, there is an increasing need for scalable wireless data dissemination techniques which can concurrently deliver data to a large number of users. Wireless data broadcast is such a method to enhance the system scalability that allows multiple users to access data simultaneously. It is very suitable for data services in asymmetric communication environments, such as news reports, stock quotes, flight schedules, traffic reports, etc. [1], [2], [3]. An important performance concern in wireless data broadcast systems is access latency, which is defined as the time duration from the moment that a request is submitted to the moment that all the requested data items are downloaded. Motivated by reducing the average access latency, there has been much research done on scheduling for data broadcasts (see e.g. [4], [5], [6], [7], [8], [9], [10] for recent works).

In traditional push-based wireless data broadcasts, data items are disseminated cyclically according to a pre-defined schedule. Therefore when the access pattern of data items changes dynamically, hot data items may be broadcast less frequently than cold data items which will result in a poor average access latency. In view of this, on-demand wireless data broadcast that disseminates data items timely according to the current requests was proposed to overcome the aforementioned drawback. In the literature, most previous studies on data scheduling for on-demand wireless data broadcast only investigate single-item requests (i.e. users only download one data item per request [11], [12], [13]). However, in many applications, such as stock quotes and flight schedules, users want to download multiple data items per request. The existing scheduling algorithms for single-item requests may be unable to perform efficiently for scheduling multi-item requests. Therefore, in this study, we investigate new scheduling methods for multi-item requests in on-demand wireless data broadcast environments with the objective of minimizing the average access latency.

B. Our Contribution

1) In on-demand wireless data broadcasts, the server has no knowledge about the requests in advance. An optimal schedule only for the requests currently received may result in poor performance as new requests come during the time. Therefore, each time it is better to compute a schedule for a part of pending requests that can be broadcast in a short time rather than to compute an entire schedule for all the pending requests. In view of this, we first investigate the Maximum Throughput Request Selection (MTRS) problem which aims at finding a subset $\mathcal{Q}$ of pending requests to be broadcast in the next time period such that the throughput, defined as $|\mathcal{Q}|/T(\mathcal{Q})$, is maximized where $|\mathcal{Q}|$ denotes the number of requests in $\mathcal{Q}$ and $T(\mathcal{Q})$ denotes the number of time slots needed to broadcast all the requests in $\mathcal{Q}$. We developed a linear programming based algorithm that can find an optimal solution to MTRS efficiently in polynomial time. It is worthy to note that the pending requests other than $\mathcal{Q}$ will also be broadcast, but just not in the next $T(\mathcal{Q})$ time slots.

2) The optimal request set $\mathcal{Q}$ can be applied directly when it contains a small set of data items. However, it is possible

that the optimal solution of an MTRS instance contains a large number of data items that cannot be broadcast in a short time. In such cases, we need to further prune the request set $\mathcal{Q}$ such that $T(\mathcal{Q})$ is no greater than a pre-defined threshold δ and meanwhile ensure a maximal throughput. We find that the MTRS problem with constraint $T(\mathcal{Q}) \leq \delta$ is strongly $\mathcal{NP}$-hard. Therefore it is computationally expensive to find an optimal solution, and we develop two greedy based heuristic algorithms to prune $\mathcal{Q}$ when it is not small enough. Their performances are evaluated through simulation.

3) To further reduce the average access latency, we investigate the Minimum Latency Request Ordering (MLRO) problem, i.e. determine the broadcasting time for each data item in $\mathcal{Q}$ such that the average access latency for all the requests in $\mathcal{Q}$ is minimized. We developed a dynamic programming based algorithm that can find an optimal solution to MLRO in $\mathcal{O}(2^{\delta})$ time, assuming $T(\mathcal{Q}) \leq \delta$. Therefore, it efficiently solves the MLRO problem in our scheduling scheme. In addition, we show that the MLRO problem with an arbitrary number of requests is strongly $\mathcal{NP}$-hard.

4) Based on the request selection, pruning, and ordering algorithms, we develop an efficient scheduling scheme for on-demand data broadcasts. We compare the proposed scheme with classic scheduling methods for wireless data broadcasts through simulation. The data items in our simulation are generated with access probabilities following the Zipf distribution which is a widely used model to simulate human behaviors. The simulation results show that significantly better result can be obtained by using our scheme rather than its competitors.

C. Paper Organization

The rest of this paper is organized as follows: Section II gives a brief review of related works. Section III presents the system model used in this work and the problem statements of MTRS and MLRO respectively. Section IV presents our scheduling scheme. To be precise, Section IV-A presents an optimal algorithm to MTRS; Section IV-B presents two pruning heuristic algorithms; Section IV-C presents an optimal algorithm to MLRO; and Section IV-D presents the overall design of our scheduling scheme. Finally, Section V provides the simulation results and Section VI concludes this paper.

II. Related Work

In the literature, various optimization problems in wireless data broadcasts have been studied to minimize the access latencies. Acharya et al. first proposed the scheduling problem for wireless data broadcasts. They introduced a skewed data scheduling technique, namely "Broadcast Disk", in [14] to reduce the average access latency. In [15], Kenyon et al. found that the minimum latency data scheduling problem with data items of non-uniform sizes is $\mathcal{NP}$-hard even if there is only one broadcasting channel. Besides single channel, Prabhakara et al. presented a multi-channel model with air cache design for wireless data broadcasts in [16], and Yee et al. developed an optimal algorithm to partition data items with uniform size into multiple channels in [17]. The works mentioned above were under the premise that every user requests only one data item at a time. However, in many applications, users want to download multiple data items per request. Therefore Huang et al. proposed two generic algorithms for scheduling multi-item requests to minimize the total access latency in [18] and [19] respectively.

In terms of on-demand scheduling, the earliest study for data broadcast can be traced back to 1980s. In [20], three heuristic algorithms: First Come First Served (FKFS), Most Requests First (MRF) and Longest Wait First (LWF) were proposed and evaluated in terms of average access latency. In [11], [12], the performances of the above algorithms were evaluated, considering both access latency and deadline. Based on FCFS, MRF and LWF, Aksoy and Franklin [21] developed an algorithm which considers both waiting time and data access frequency to reduce the average access latency. In [22], a Lazy Data Request (LDR) strategy was proposed so that indexing can be used to reduce the average access latency. Besides single-item requests, Chen et al. [23] developed an algorithm that combines the benefits of data item scheduling and request scheduling to allocate data items for multi-item requests in on-demand wireless data broadcasts. In [24], they further investigated preemptive on-demand scheduling for multi-item requests which considers the factors of urgency, item productivity, and request serving status to schedule data items. Besides single channel, Liu et al. and Lv et al. studied scheduling algorithms for multi-channel on-demand data broadcasts in [25] and [26] respectively.

In addition to data scheduling at the server side, there are also many works done on scheduling the data retrieval process at the client side. When retrieving multiple data items from parallel channels, clients probably need switchings among the channels, which will cause possible conflicts [27], [28], [29]. Juran et al. and Hurson et al., developed heuristic algorithms in [27], [28] to reduce the access latency and the number of channel switchings for parallel data retrieving, in which they assumed the data items were partitioned over multiple channels without replications. This assumption simplifies the algorithm design, but restricts their works for practice because in many wireless data broadcasting applications, one data item may appear multiple times in one broadcasting cycle [14], [18], [10], [17]. In [8] and [2], Peng et al. and Lu et al. studied the multi-item data retrieval scheduling problem for wireless data broadcasts, in which a data item can be repeatedly broadcast in one cycle. Compared with the model proposed in [27], [28], this is a more generalized assumption which fits most real wireless data broadcast applications.

III. System Model and Problem Statement

In this section we present problem statements and related notations. Suppose an on-demand wireless data broadcasting system has a single broadcasting channel and a database with a set $\mathcal{D} = \{d_1, \cdots, d_i, \cdots, d_N\}$ of data items. Without loss of generality, we have the following assumptions:

- The broadcasting channel is partitioned into discrete time slots of equal size H and each time slot is the smallest unit to broadcast a data item. Therefore, let $S(d_i)$ denote the size of data item d_i $(1 \le i \le N)$, the number of time slots needed to broadcast d_i is $\lceil S(d_i)/\Big(B \cdot H\Big)\rceil$ where B is the channel bandwidth.

- Let time t denote the time slot with sequence number "t", a user can start to download data items at time $t+1$ when he (or she) submitted a request at time t.
- The server does not know in advance the coming requests, i.e. at any particular time t, the server only knows the pending requests that are submitted before time t.
- Let $D(t_i, t_j)$ denote the set of data items being broadcast during time $[t_i, t_j]$. A request Q submitted at time t_i is complete at time t_j if and only if $D(Q) \subseteq D(t_i+1, t_j)$ and $D(Q) \nsubseteq D(t_i+1, t_j-1)$ where $D(Q)$ denotes the set of data items in request Q.
- The access latency of a request is defined as the time duration from the moment that the request is submitted to the moment that it is complete.

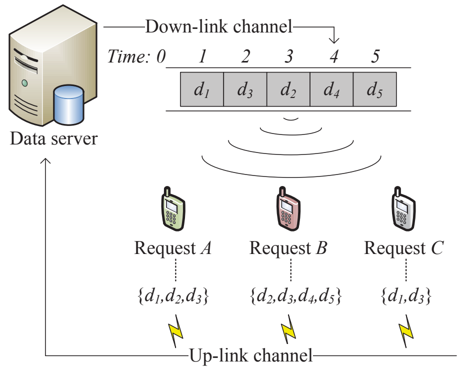

Fig. 1. An example of on-demand wireless data broadcast

Given a broadcasting schedule $\mathcal{S}$, we denote by $L(\mathcal{S}, Q, t)$ the access latency of a request Q submitted at time t. By the above assumptions, we have $L(\mathcal{S}, Q, t) = \min\{l : l \in \mathcal{Z}, D(Q) \subseteq D(t+1, t+l)\}$. Fig. 1 shows an example with three requests: A, B and C where $D(A) = \{d_1, d_2, d_3\}$, $D(B) = \{d_2, d_3, d_4, d_5\}$ and $D(C) = \{d_1, d_3\}$. Assume, for simplicity, that each data item d_i $(1 \le i \le 5)$ can be broadcast in one time slot and all the requests are submitted at time 0. According to the broadcasting schedule shown in Fig. 1, the access latencies for requests A, B and C are $L(\mathcal{S}, A, 0) = 3$, $L(\mathcal{S}, B, 0) = 5$, and $L(\mathcal{S}, C, 0) = 2$ respectively.

Let $\mathcal{P} = \{Q_1, \cdots, Q_i, \cdots, Q_M\}$ denote the set of pending requests received, the MTRS problem is to select a subset $\mathcal{Q}$ of $\mathcal{P}$ to maximize the throughput for the next time period. In this study, we define the throughput of $\mathcal{Q}$ as the ratio of $|\mathcal{Q}|$ over $T(\mathcal{Q})$ where $|\mathcal{Q}|$ denotes the number of requests in $\mathcal{Q}$ and $T(\mathcal{Q})$ denotes the number of time slots needed to broadcast all the requests in $\mathcal{Q}$. As the example shown in Fig. 1, the requests A, B, and C are completely broadcast in 3, 5, and 5 time slots respectively. Therefore the throughput of request sets $\{A\}$, $\{A, B\}$, and $\{A, B, C\}$ are $1/3$, $2/5$, and $3/5$ respectively. Let $D(\mathcal{Q})$ denote the set of data items involved in request set $\mathcal{Q}$ (i.e., $D(\mathcal{Q}) = \bigcup_{Q \in \mathcal{Q}} D(Q)$) and $\tau(d_i) = \lceil S(d_i)/B \cdot H \rceil$ denote the number of time slots needed to broadcast data item d_i, then $T(\mathcal{Q}) = \sum_{d_i \in D(\mathcal{Q})} \tau(d_i)$. The formal definition of MTRS is as follows:

(MTRS) : Given a set $\mathcal{P} = \{Q_1, \cdots, Q_i, \cdots, Q_M\}$ of pending requests with associated data item sets $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_M)$, the Maximum Throughput Request Selection **(MTRS)** problem is to find a subset $\mathcal{Q}$ of $\mathcal{P}$ such that the throughput, $|\mathcal{Q}|/T(\mathcal{Q})$, is maximized.

When $\mathcal{Q}$ contains two or more requests, different broadcasting orders will result in different average access latencies. Therefore after selecting a subset $\mathcal{Q} \subseteq \mathcal{P}$ of requests in MTRS, we also need to find a proper broadcasting order for the requests in $\mathcal{Q}$ to further reduce the average access latency. For this purpose, we define the MLRO problem as follows:

(MLRO) : Given a set $\mathcal{Q} = \{Q_1, \cdots, Q_i, \cdots, Q_{|\mathcal{Q}|}\}$ of selected requests with associated data item sets $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_{|\mathcal{Q}|})$ and a starting time t, the Minimum Latency Request Order **(MLRO)** problem is to find a broadcasting schedule $\mathcal{S}$ for all the data items in $D(\mathcal{Q})$ such that $\left(\left(\sum_{Q \in \mathcal{Q}} L(\mathcal{S}, Q, t)\right)/|\mathcal{Q}|\right)$ is minimized.

IV. Scheduling Scheme for On-demand Wireless Data Broadcast

In this section, we present a two-stage scheme for scheduling multi-item requests for on-demand wireless data broadcasts. The algorithms applied in each stage and their theoretical analysis are presented before presenting the overall system.

A. *Optimal Algorithm to MTRS*

We first present a polynomial time optimal algorithm for the MTRS problem. Let $x_i = 1$ denote request Q_i is selected and $x_i = 0$ otherwise, and let $y_j = 1$ denote data item d_j is selected and $y_j = 0$ otherwise, then we can formulate MTRS into the following integer programming:

$$\begin{aligned} \max \quad & \frac{\sum_{Q_i} x_i}{\sum_{d_j}\Big(\tau(d_j)\cdot y_j\Big)} \\ \text{s.t.} \quad & x_i \le y_j, \forall d_j \in D(Q_i) \\ & x, y \in \{0,1\}. \end{aligned} \tag{1}$$

We next show that problem (1) can be solved by using a classic linear programming algorithm with a simple rounding method. The linear programming is as follows:

$$\begin{aligned} \max \quad & \sum_{Q_i} x_i \\ \text{s.t.} \quad & x_i \le y_j, \forall d_j \in D(Q_i) \\ & \sum_{d_j}\Big(\tau(d_j)\cdot y_j\Big) \le 1 \\ & 0 \le x, y \le 1. \end{aligned} \tag{2}$$

It is well known that linear programming can be solved in polynomial time. Let (x^*, y^*) be an optimal solution of problem (2), the basic idea is to find a threshold η $(0 \le \eta \le 1)$ such that we can convert (x^*, y^*) into an integral solution for problem (1) by setting $x_i^* = 1$ for each $x_i^* \ge \eta$ and setting $x_j^* = 0$ for each $x_j^* < \eta$. The detailed steps are given in Algorithm 1.

Algorithm 1 Optimal Algorithm to MTRS

```
1: The input: $\mathcal{P} = \{Q_1, \cdots, Q_i, \cdots, Q_M\}$ with associated data item sets: $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_M)$.
2: let $\mathcal{Q} \leftarrow \emptyset$ ($\mathcal{Q}$ holds the selected requests);
3: find an optimal solution $(x^*, y^*)$ for problem (2) by using a classical linear programming algorithm;
4: let $\eta \leftarrow \max x_i^*$;
5: for each $x_i^*$ $(1 \le i \le M)$, let $x_i^* \leftarrow 1$ if $x_i^* \ge \eta$, and let $x_i^* \leftarrow 0$ if $x_i^* < \eta$;
6: for each $x_i^* = 1$ $(1 \le i \le M)$, put $Q_i$ into $\mathcal{Q}$;
7: for each $y_j^*$ $(1 \le j \le N)$, let $y_j^* \leftarrow 1$ if $d_j \in D(\mathcal{Q})$, and let $y_j^* \leftarrow 0$ if $d_j \notin D(\mathcal{Q})$;
8: The output: $\mathcal{Q}$.
```

To show Algorithm 1 is an optimal algorithm to MTRS, we first prove the following lemma.

Lemma 1. *For an arbitrary MTRS instance, the optimal objective value of problem (2) is equal to or greater than the objective value of any feasible solution for problem (1).*

Proof. We prove Lemma 1 by construction. Given an MTRS instance with a set $\mathcal{P}$ of current pending requests. Assume (x^ℓ, y^ℓ) is an optimal solution for problem (1) with objective value *opt*, we can construct a feasible solution (x^*, y^*) for problem (2) with the same objective value.

Let $\kappa = \sum_{d_j}\Big(\tau(d_j) \cdot y_j^\ell\Big)$, we set $y_i^* = \frac{1}{\kappa}$ for each $y_i^\ell = 1$ and set $y_i^* = 0$ for each $y_i^\ell = 0$, and we set $x_i^* = \frac{1}{\kappa}$ for each $x_i^\ell = 1$ and set $x_i^* = 0$ for each $x_i^\ell = 0$. Since $x_i^\ell \le y_j^\ell$ for any i, j, we have

$$x_i^* \le y_j^*, \forall d_j \in D(Q_i).$$

In addition, since $\sum_{d_j}\Big(\tau(d_j) \cdot y_j^\ell\Big) = \kappa$, we have

$$\sum_{d_j}\Big(\tau(d_j) \cdot y_j^*\Big) = \sum_{d_j}\Big(\tfrac{1}{\kappa} \cdot \tau(d_j) \cdot y_j^\ell\Big) = 1.$$

Therefore (x^*, y^*) is a feasible solution of problem (2). On the other hand, the objective value of (x^*, y^*) for problem (2) is defined as

$$\sum_{Q_i} x_i^* = \sum_{Q_i}\Big(\tfrac{1}{\kappa} \cdot x_i^\ell\Big) = \frac{\sum_{Q_i} x_i^\ell}{\sum_{d_j}\Big(\tau(d_j) \cdot y_j^\ell\Big)} = opt.$$

In sum, we have that (x^*, y^*) is a feasible solution of problem (2) and its objective value is no less than the optimal objective value of problem (1). The proof of Lemma 1 is complete. □

Theorem 1. *Let $\mathcal{P}$ be the input of an arbitrary MTRS instance and (x^*, y^*) be an optimal solution for problem (2). For any $\eta \in y^*$, if we set $x_i^* = 1$ for each $x_i^* \ge \eta$ and $y_i^* = 1$ for each $y_i^* \ge \eta$, and set the remainder $x_i^* = 0$ and $y_j^* = 0$, then (x^*, y^*) is an optimal solution of problem (2).*

Proof. Let $\eta_1, \cdots, \eta_j, \cdots \eta_J$ be the distinct values for all $y_j^* \in y^*$. First we claim that for any x_i^*, there exists a value $\eta \in \{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$ such that $x_i^* = \eta$. Assume, for the sake of contradiction, that there exists x_i^* whose value is different from any value in $\{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$. Since $x_i^* \le y_j^*$ for any $d_j \in D(Q_i)$, there exists $\eta \in \{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$ such that $x_i^* < \eta$. Let η be the smallest value in $\{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$ that is greater than x_i^*, then setting $x_i^* = \eta$ will result in a feasible solution to problem (2) with a larger objective value. This contradicts the assumption that (x^*, y^*) is an optimal solution to problem (2). Therefore our claim holds.

Without loss of generality, assume $\eta_1, \cdots, \eta_i, \cdots \eta_J$ are in the increasing order (i.e., $\eta_1 < \cdots < \eta_j < \cdots < \eta_J$). Define $N(\eta) = \sum_{y_j^* \ge \eta} \tau(d_j)$ and let $\eta_0 = 0$, we have

$$\sum_{j=1,\cdots,J}\Big((\eta_j - \eta_{j-1}) \cdot N(\eta_j)\Big) = \sum_{d_j}\Big(\tau(d_j) \cdot y_j^*\Big) \le 1,$$

in which the first equation is because $\tau(d_j) \cdot y_j^* = \tau(d_j) \cdot \Big(\sum_{j'=1,\cdots,J'}(\eta_{j'} - \eta_{j'-1})\Big)$ for any $1 \le j \le J$ where $\eta_{j'} = y_j^*$.

Define $M(\eta)$ as the number of $x_i^* \in x^*$ that are no less than η, and let $J_{x_i^*}$ denote the sequence number for $\eta_{J_{x_i^*}} = x_i^*$ and ζ the objective value of (x^*, y^*) for problem (2), since for each x_i^*, there exists $\eta \in \{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$ such that $\eta = x_i^*$, we have

$$\sum_{j=1\ldots J}\Big((\eta_j - \eta_{j-1}) \cdot M(\eta_j)\Big)$$
$$= \sum_{Q_i}\Big(\sum_{j=1\ldots J_{x_i^*}}\Big(\eta_j - \eta_{j-1}\Big)\Big) = \sum_{Q_i} x_i^* = \zeta.$$

To prove Theorem 1, it is sufficient to show that for any $\eta \in \{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$, $\frac{M(\eta)}{N(\eta)} = \zeta$. By Lemma 1, the objective value of any feasible solution of problem (1) is no greater than ζ. Therefore $\frac{M(\eta)}{N(\eta)} \le \zeta$ for any η. Assume that there exists η_j such that $\frac{M(\eta_j)}{N(\eta_j)} < \zeta$. Then we have $M(\eta_j) < \zeta \cdot N(\eta_j)$ which indicates that

$$\sum_{j=1}^{J}\Big((\eta_j - \eta_{j-1}) \cdot M(\eta_j)\Big) < \sum_{i=1}^{k}\Big((t_i - t_{i-1}) \cdot N(t_i) \cdot \zeta\Big)$$
$$= \zeta \cdot \Big(\sum_{i=1\ldots k}\Big((t_i - t_{i-1}) \cdot N(t_i)\Big)\Big) \le \zeta,$$

which contradicts with $\sum_{j=1\ldots J}\Big((\eta_j - \eta_{j-1}) \cdot M(\eta_j)\Big) = \zeta$. Therefore $\frac{M(\eta)}{N(\eta)} = \zeta$ for any $\eta \in \{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$. Let $y_j^* = 1$ for each $y_j^* \ge \eta$ and $x_i^* = 1$ for each $x_i^* \ge \eta$, and set the remaining x_i^* and y_j^* to zero, we have $\frac{\sum_{Q_i} x_i}{\sum_{d_j} T(d_j) \cdot y_j} = \frac{M(\eta)}{N(\eta)} = \zeta$. The proof of Theorem 1 is complete. □

It is clear that Algorithm 1 runs in polynomial time. By Theorem 1 we can select any value from $\{\eta_1, \cdots, \eta_j, \cdots \eta_J\}$ as η in Algorithm 1 to get a maximum throughput request set $\mathcal{Q}$. Since we want to find a set of requests that can be broadcast in a short time, we select the one at which the smallest $T(\mathcal{Q})$ can be obtained.

B. Pruning the result of MTRS

Although Algorithm 1 efficiently solves MTRS, it is possible that the solutions of some MTRS instances are so large that cannot be broadcast in a short time. To overcome this shortcoming, the best way is to achieve a request set $\mathcal{Q}$ satisfying

- $T(\mathcal{Q}) \leq \delta$ where δ is a pre-determined time threshold
- for any request set $\mathcal{Q}'$ with $T(\mathcal{Q}') \leq \delta$, $|\mathcal{Q}'| \leq |\mathcal{Q}|$, i.e., we want to obtain a request set $\mathcal{Q}$ which has the maximum throughput among all the request sets that can be broadcast in δ time slots.

However, this additional constraint makes the problem computationally intractable.

Theorem 2. *The MTRS problem, under time constraint $T(\mathcal{Q}) \leq \delta$ is $\mathcal{NP}$-hard for general δ.*

Proof. To prove Theorem 2, we do a reduction from the Clique problem. Its input is a general graph $G(V, E)$ without self-loop and an integer K, and it is $\mathcal{NP}$-hard to determine whether there is a complete graph $G'(V'E')$ of K nodes in $G(V, E)$.

Given a graph $G(V, E)$ and an integer K for a K Clique problem, we construct a set $\mathcal{P}$ of $|E|$ requests as follows. First, for each node $u \in V$, create a data item d_u and let $T(d_u) = 1$. Second, for each edge $(u, v) \in E$, create a request $Q_{u,v}$ consisting of data items d_u and d_v. Last, let $\delta = K$. It is clear that the reduction can be done in polynomial time. In addition, there is a one-to-one correspondence between the nodes and data items, and the edges and requests respectively. Therefore, the K Clique problem is a "yes" instance if and only if there exists a set $\mathcal{Q}$ of $(K-1) \cdot K/2$ requests with K data items. Since every data item can be broadcast in one time slot, $\mathcal{Q}$ can be broadcast in K time slots and thus the throughput is $\frac{|\mathcal{Q}|}{T(\mathcal{Q})} = \Big(K-1\Big) \cdot K/2K = \frac{K-1}{2}$. Clearly, the throughput is no more than $\Big(K'-1\Big)/2$ for any request set $\mathcal{Q}'$ with K' data items. Therefore, $\Big(K-1\Big)/2$ is the maximum throughput for any request $\mathcal{Q}$ that can be broadcast within K time slots, which implies that there is a K Clique in G if and only if there is a request set $\mathcal{Q} \subseteq \mathcal{P}$ whose throughput is $\Big(K-1\Big)/2$. The proof of Theorem 2 is complete. □

By Theorem 2, unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial time optimal algorithm to find a maximum throughput request set that can be broadcast in a designate time. In this work, instead of finding the time constrained maximum throughput request set directly, we develop two greedy based heuristic algorithms: Maximum Gain and Least Loss, and apply them to prune the maximum throughput request set outputted by Algorithm 1 iteratively until it can be broadcast within a designate time. Given a request set $\mathcal{Q}$ and a time constraint δ, Maximum Gain Heuristic repeatedly select requests in $\mathcal{Q}$ until the selected requests cannot be broadcast in δ time slots, while Least Loss Heuristic iteratively remove requests from $\mathcal{Q}$ until the remainder requests can be broadcast in δ time slots. The pseudo-codes are given in Algorithms 2 and 3.

Typically greedy based heuristics choose the local best option at each stage according to some scenarios and terminates when a specific condition has been met. Since the objective is to keep as many requests as possible subject to a time constraint, it stands to reason that we select requests without increasing too much broadcasting time. In addition, we should delete data items that belong to a small number of requests but require a relatively long time to be broadcast. According to these observations, we develop Maximum Gain Heuristic (Algorithm 2) and Least Loss Heuristic (Algorithm 3).

Algorithm 2 Pruning Algorithm: Maximum Gain Heuristic

```
1: The input: $\mathcal{Q} = \{Q_1, \cdots, Q_i, \cdots, Q_{|\mathcal{Q}|}\}$ with associated
   data item sets: $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_{|\mathcal{Q}|})$ and $\delta$.
2: let $\mathcal{H} \leftarrow \emptyset$ ($\mathcal{H}$ holds the selected requests);
3: loop
4:     find a request $Q \in (\mathcal{Q} \setminus \mathcal{H})$ such that $\frac{1}{T(\mathcal{H} \cup \{Q\}) - T(\mathcal{H})}$
       is maximized.
5:     if $T(\mathcal{H} \cup \{Q\}) \leq \delta$ then
6:         let $\mathcal{H} \leftarrow \mathcal{H} \cup \{Q\}$;
7:     else
8:         break;
9:     end if
10: end loop
11: Let $\mathcal{Q} \leftarrow \mathcal{H}$;
12: The output: $\mathcal{Q}$;
```

Algorithm 3 Pruning Algorithm: Least Loss Heuristic

```
1: The input: $\mathcal{Q} = \{Q_1, \cdots, Q_i, \cdots, Q_{|\mathcal{Q}|}\}$ with associated
   data item sets: $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_{|\mathcal{Q}|})$ and $\delta$.
2: let $\mathcal{W} \leftarrow D(\mathcal{Q})$ ($\mathcal{W}$ holds the remainder data items);
3: while $T(\mathcal{Q}) > \delta$ do
4:     find a data item $d \in \mathcal{W}$ such that $\frac{|\{Q|Q \in \mathcal{Q}, d \in D(Q)\}|}{\tau(d)}$ is
       minimized.
5:     delete $d$ from $\mathcal{W}$ and delete $Q$ from $\mathcal{Q}$ if $d \in D(Q)$;
6: end while
7: The output: $\mathcal{Q}$;
```

In Algorithm 2, each time a request Q is selected based on formula $\frac{1}{T(\mathcal{H} \cup \{Q\}) - T(\mathcal{H})}$, where $T(\mathcal{H} \cup \{Q\})$ is the broadcasting time when Q is added and $T(\mathcal{H})$ is the broadcasting time before Q is added. Therefore, each time it selects a request that maximizes $\frac{1}{T(\mathcal{H} \cup \{Q\}) - T(\mathcal{H})}$, and thus maximizes the marginal gain $\frac{|\mathcal{H} \cup \{Q\}|}{T(\mathcal{H} \cup \{Q\})} - \frac{|\mathcal{H}|}{T(\mathcal{H})}$. Since at most $|\mathcal{Q}|$ requests can be selected and each selection requires $|Q|$ comparisons, Algorithm 2 runs in $\mathcal{O}(|\mathcal{Q}|^2)$ time.

In Algorithm 3, each time a data item d is deleted according to formula $|\{Q \mid Q \in \mathcal{Q}, d \in D(Q)\}|/\tau(d)$, where $\tau(d)$ is the number of time slots needed to broadcast d and $\{Q \mid Q \in \mathcal{Q}, d \in D(Q)\}$ is the set of requests containing d. Therefore, each time it deletes a data item that minimizes $|\{Q \mid Q \in \mathcal{Q}, d \in D(Q)\}|/\tau(d)$, i.e., it minimizes the ratio between the number of requests containing d and the number of time slots needed to broadcast d. The time complexity follows from the pseudo-code. Let n denote the number of data items in $\mathcal{Q}$, there are at most n data item deletions and each deletion requires at most n comparisons. In addition, there are at most $|\mathcal{Q}|$ request deletions in total. Therefore, Algorithm 2 runs in $\mathcal{O}(n^2 + |\mathcal{Q}|)$ time.

C. Optimal Algorithm to MLRO

In this subsection we present an optimal algorithm to schedule the broadcasting time of requests such that the average access latency is minimized. The naive solution can be obtained by checking all possible arrangements of the n data items which requires $n!$ time. However, the solution space can be reduced from $\mathcal{O}(n!)$ to $\mathcal{O}(\min(2^n, 2^{|\mathcal{Q}|}))$ for a request set $\mathcal{Q}$ by using Dynamic Programming (DP). Before the formal algorithm, we first define the $(d, \mathcal{W})$-optimal schedule for a set $\mathcal{W}$ of data items containing data item d and the $(Q, \mathcal{Q})$-optimal schedule for a set $\mathcal{Q}$ of requests containing request Q respectively.

Definition 1. *Given a set $\mathcal{W}$ of data items and a set $\mathcal{Q}$ of requests, each of which is strictly contained by $\mathcal{W}$ (i.e. $\forall Q \in \mathcal{Q}, D(Q) \subseteq \mathcal{W}$), a schedule $\mathcal{S}_d^{\mathcal{W}}$ is $(d, \mathcal{W})$-optimal if*

- *d is the last broadcast data item according to $\mathcal{S}_d^{\mathcal{W}}$.*
- *for any schedule $\mathcal{S}$ that broadcasts d by the end, $\mathcal{S}_d^{\mathcal{W}}$ holds that $\sum_{Q\in\mathcal{Q}} L(\mathcal{S}_d^{\mathcal{W}}, Q, t_Q) \leq \sum_{Q\in\mathcal{Q}} L(\mathcal{S}, Q, t_Q)$, where t_Q denotes the time slot when Q is submitted and $L(\mathcal{S}_d^{\mathcal{W}}, Q, t_Q)$ denote the access latency of request Q according to schedule $\mathcal{S}_d^{\mathcal{W}}$.*

Definition 2. *Given a set $\mathcal{Q}$ of requests, a schedule $\mathcal{S}_Q^{\mathcal{Q}}$ is $(Q, \mathcal{Q})$-optimal if*

- *Q is the last broadcast request according to $\mathcal{S}_Q^{\mathcal{Q}}$.*
- *for any schedule $\mathcal{S}$ that broadcasts Q by the end, $\mathcal{S}_Q^{\mathcal{Q}}$ holds that $\sum_{Q\in\mathcal{Q}} L(\mathcal{S}_Q^{\mathcal{Q}}, Q, t_Q) \leq \sum_{Q\in\mathcal{Q}} L(\mathcal{S}, Q, t_Q)$.*

Lemma 2. *For any set $\mathcal{W}$ of data items and $d \in \mathcal{W}$, there exists an algorithm that computes a $(d, \mathcal{W})$-optimal schedule in $\mathcal{O}(2^n(n|\mathcal{Q}|)^{\mathcal{O}(1)})$ time where n is the number of requests and $|\mathcal{Q}|$ is the number of requests in $\mathcal{Q}$.*

Proof. We can compute $\mathcal{S}_d^{\mathcal{W}}$ recursively. First for any data item set $\mathcal{W}' \subseteq \mathcal{W}$ and $d \in \mathcal{W}'$, we denote by $\mathcal{S}_d^{\mathcal{W}'}$ the $(d, \mathcal{W}')$-optimal schedule for $\mathcal{W}'$, $\mathcal{S}^{\mathcal{W}'}$ the optimal schedule for $\mathcal{W}'$, and $\mathrm{AAL}(\mathcal{S})$ the Average Access Latency for any schedule $\mathcal{S}$, then $\mathcal{S}_d^{\mathcal{W}}$ can be computed through the following recursions:

$$\mathcal{S}^{\mathcal{W}'} = \mathrm{argmin}_{(\mathcal{S}_d^{\mathcal{W}'}, d\in\mathcal{W}')}\mathrm{AAL}(\mathcal{S}_d^{\mathcal{W}'}) \quad \forall d \in \mathcal{W}' \tag{3}$$

$$\mathcal{S}_d^{\mathcal{W}'\cup\{d\}} = \mathcal{S}^{\mathcal{W}'} + d \quad \forall d \notin \mathcal{W}' \tag{4}$$

where "$\mathcal{S}^{\mathcal{W}'} + d$" means attaching data item d at the end of schedule $\mathcal{S}^{\mathcal{W}'}$.

It is clear that the optimal schedule $\mathcal{S}^{\{d\}}$ for all single-item data sets $\{d\}$ can be enumerated in linear time. Therefore, the $(d, \mathcal{W}')$-optimal schedule $\mathcal{S}_d^{\mathcal{W}'}$ for any data set $\mathcal{W}'$ that contains two data items can be computed by recursion (3) in linear time. After that, the optimal schedule $\mathcal{S}^{\mathcal{W}'}$ can be obtained by recursion (4) directly. Therefore we can obtain $\mathcal{S}^{\mathcal{W}'}$ and $\mathcal{S}_d^{o\mathcal{W}'}$ for any $\mathcal{W}' \subseteq \mathcal{W}, d \in \mathcal{W}'$ by a bottom up approach. Since there are n data items, the number of data sets $\mathcal{W}'$ is 2^n. For each $\mathcal{W}'$, there are $|\mathcal{W}'|$ distinct $(d, \mathcal{W}')$-optimal schedules where $|\mathcal{W}'| \leq n$. Therefore, $\mathcal{S}_d^{\mathcal{W}}$ can be obtained in $\mathcal{O}(2^n(n|\mathcal{Q}|)^{\mathcal{O}(1)})$ time. □

Lemma 3. *For any set $\mathcal{Q}$ of requests and $Q \in \mathcal{Q}$, there exists an algorithm that computes a $(Q, \mathcal{Q})$-optimal schedule in $\mathcal{O}(2^{|\mathcal{Q}|}(|\mathcal{Q}|^{\mathcal{O}(1)}))$ time where $|\mathcal{Q}|$ is the number of requests in $\mathcal{Q}$.*

Proof. The proof of Lemma 3 is similar as that of Lemma 2 except the recursions as shown below.

$$\mathcal{S}^{\mathcal{Q}'} = \mathrm{argmin}_{(\mathcal{S}_Q^{\mathcal{Q}'}, Q\in\mathcal{Q}')}\mathrm{AAL}(\mathcal{S}_Q^{\mathcal{Q}'}) \quad \forall Q \in \mathcal{Q}' \tag{5}$$

$$\mathcal{S}_Q^{\mathcal{Q}'\cup\{Q\}} = \mathcal{S}^{\mathcal{Q}'} + Q \quad \forall Q \notin \mathcal{Q}' \tag{6}$$

where $\mathcal{S}_d^{\mathcal{W}'}$ denotes the $(Q, \mathcal{Q}')$-optimal schedule for $\mathcal{Q}'$, $\mathcal{S}^{\mathcal{Q}'}$ denotes the optimal schedule for $\mathcal{Q}'$, and "$\mathcal{S}^{\mathcal{Q}'} + Q$" means attaching the data items in $\Big(D(Q) \setminus D(\mathcal{Q}')\Big)$ at the end of $\mathcal{S}^{\mathcal{Q}'}$. □

Theorem 3. *For any MLRO problem with a set $\mathcal{Q}$ of requests, there exists a $\mathcal{O}(\min(2^n, 2^{|\mathcal{Q}|})(n|\mathcal{Q}|^{\mathcal{O}(1)}))$ time optimal solution where n the number of data items and $|\mathcal{Q}|$ is the number of requests.*

Proof. According to Lemmas 2 and 3, we can obtain an optimal schedule for request set $\mathcal{Q}$ in $\mathcal{O}(2^n(n|\mathcal{Q}|)^{\mathcal{O}(1)})$ and $\mathcal{O}(2^{|\mathcal{Q}|}(|\mathcal{Q}|^{\mathcal{O}(1)}))$ time respectively. Therefore, MLRO can be solved in $\mathcal{O}(\min(2^n, 2^{|\mathcal{Q}|})(n|\mathcal{Q}|^{\mathcal{O}(1)}))$ time. □

Algorithm 4 Optimal Algorithm to MLRO

```
1: The input: $\mathcal{Q} = \{Q_1, \cdots, Q_i, \cdots, Q_{|\mathcal{Q}|}\}$ with associated
   data item sets: $D(Q_1), \cdots, D(Q_i), \cdots, D(Q_{|\mathcal{Q}|})$.
2: let $\mathcal{W} \leftarrow D(\mathcal{Q})$ and $n \leftarrow |D(\mathcal{Q})|$;
3: if $n \leq |\mathcal{Q}|$ then
4:     find an optimal schedule $\mathcal{S}^{\mathcal{W}}$ for data set $\mathcal{W}$ according
       to Equations (3) and (4), and set $\mathcal{S} \leftarrow \mathcal{S}^{\mathcal{W}}$;
5: else
6:     find an optimal schedule $\mathcal{S}^{\mathcal{Q}}$ for request set $\mathcal{Q}$ according
       to Equations (5) and (6), and set $\mathcal{S} \leftarrow \mathcal{S}^{\mathcal{Q}}$;
7: end if
8: The output: $\mathcal{S}$;
```

By Theorem 3, we can solve MLRO efficiently when either n or $\mathcal{Q}$ is not large. Since after pruning, $T(\mathcal{Q})$ is no greater than δ and one time slot is the smallest unit to broadcast a data item, we have that $\min(n, |\mathcal{Q}|)$ is no greater than δ, which is a pre-determined small constant. Therefore, in our system scenario, we can obtain an optimal solution to MLRO efficiently by Algorithm 4, although it requires exponential time in terms of n and $|\mathcal{Q}|$.

Next, we deny the existence of any polynomial time optimal algorithm to MLRO, assuming $\mathcal{P} \neq \mathcal{NP}$.

Theorem 4. *The MLRO problem is $\mathcal{NP}$-hard.*

Proof. To prove Theorem 2, we do a reduction from the Precedence Constrained Scheduling problem. Its input is a set of tasks and their processing time. In addition, there is a set of precedence constraints among the tasks. A constraint $A \rightarrow B$ means task A must be processed before task B.

A Precedence Constrained Scheduling problem can also be presented by an acyclic directed graph where nodes denote the tasks and edges denote the precedence constraints. It is well known that to minimize the sum of waiting time for the Precedence Constrained Scheduling problem is strongly $\mathcal{NP}$-hard.

Given a Precedence Constrained Scheduling problem with a set $\{A_1, \cdots, A_i, \cdots, A_N\}$ of tasks, we construct a special MLRO problem as follows. First, create a data item d^{A_i} for every task A_i and let $T(d^{A_i}) = t(A_i)$ where $t(A_i)$ denote the processing time of A_i. Second, create a request Q^{A_i} with data set $\{d^{A_i}\}$ for every task A_i without any precedence constraint. Third, create the rest of requests recursively. Let $\mathcal{P}$ denote the current set of requests, each time find a task $A_j \notin \mathcal{P}$ such that $Q^{A_i} \in \mathcal{P}$ for any precedence constraint $A_i \to A_j$ in the Precedence Constrained Scheduling problem. Create a request Q^{A_j} and add data item d^{A_j} into $D(Q^{A_j})$. In addition, add all data items of $D(Q^{A_i})$ into $D(Q^{A_j})$ if there exists a precedence constraint $A_i \to A_j$. The construction is complete when there exists no such A_j. Clearly the reduction can be done in polynomial time. Since for any precedence constraint $A_i \to A_j$, $D(Q^{A_i})$ is a subset of $D(Q^{A_j})$, a schedule $\mathcal{A}^* = A_1^* \to \cdots \to A_i^* \to A_N^*$ is a feasible solution to the Precedence Constrained Scheduling problem if and only if $\mathcal{S}^* = Q^{A_1^*} \to \cdots \to Q^{A_i^*} \to Q^{A_N^*}$ is a feasible solution to the MLRO problem. Moreover, let T^* denote the sum of waiting time for all the tasks according to $\mathcal{A}^*$, then the average access latency for all the requests according to $\mathcal{S}^*$ for the MLRO problem is T^*/N. Therefore, minimizing the sum of waiting time for the Precedence Constrained Scheduling problem is equivalent to minimizing the average access latency for the MLRO problem. The proof is complete. □

D. Overall Scheme

Based on the algorithms proposed in Section V-A, we develop a scheduling scheme for on-demand wireless data broadcasts. Besides request selection, pruning, and ordering, the scheduling scheme also has an information system to store the states of pending requests. The information system works as follows:

- When a new request Q is submitted, add it into the list $\mathcal{L}$ of pending requests.
- When a data item d is completely broadcast, for each pending request $Q \in \mathcal{L}$ containing d, delete d from Q.
- When a request $Q \in \mathcal{L}$ is empty, delete Q from $\mathcal{L}$ and mark it as a completed request.

Let $\mathcal{L}$ and $\mathcal{C}$ denote the lists of pending and completed requests and $\mathcal{V}$ denote the queue of data items to be broadcast respectively, the overall scheduling scheme is presented in Algorithm 5. There are two parts. In the first part (Steps 1–8), it updates the lists of pending and completed requests. In the second part (Steps 9–19), it applies the request selection, pruning, and ordering algorithms to obtain a sequence of data items and put them into the broadcasting queue.

Algorithm 5 Scheduling Scheme

```
    At each time slot $t$ do:
 1: if a new request $Q$ is submitted at time $t-1$ then
 2:     put $Q$ into $\mathcal{L}$;
 3: end if
 4: if a data item $d_i$ is broadcast completely at time $t$ then
 5:     for each request $Q \in \mathcal{L}$ containing $d_i$ do
 6:         delete $d_i$ from $Q$, and delete $Q$ if it is empty from
            $\mathcal{L}$ and put it into $\mathcal{C}$;
 7:     end for
 8: end if
 9: if $\mathcal{V}$ is empty then
10:     apply Algorithm 1 to get a maximum throughput re-
        quest set $\mathcal{Q}$;
11:     if $T(\mathcal{Q}) > \delta$ then
12:         apply Algorithms 2 or 3 to prune request set $\mathcal{Q}$;
13:     end if
14:     apply Algorithm 4 to order the data items in $\mathcal{Q}$ and put
        the data items into $\mathcal{V}$;
15: end if
16: if a data item $d_i$ is broadcast completely at time $t$ then
17:     let $d_j \leftarrow \mathcal{V}.dequeue()$;
18:     start to broadcast $d_j$ at time $t+1$;
19: end if
```

V. Simulation Result

In this section, we present the simulation results. All the experiments are conducted on a PC with an Intel i5 2.5 Ghz processer and 8 GB memory. The algorithms are described in Section V-A; the broadcast environment is presented in Section V-B; and the experimental results are discussed in Section V-C.

A. Algorithms

In the experimental results, we denote by SMGH our Scheduling scheme with Maximum Gain Heuristic and SLLH our Scheduling scheme with Least Loss Heuristic. In both SMGH and SLLH, we apply CPLEX to solve the linear programming and the time constraint parameter δ is set to 30 time slots. In addition to SMGH and SLLH, we also implement two request level scheduling algorithms: First Come First Serve (FCFS) scheduling and Request Starvation and Bandwidth Utilization (RSBU) scheduling [25]. Briefly FCFS broadcasts the requests in the order of their arrival time and breaks the ties arbitrary, and RSBU first selects a request among all pending requests according to the waiting time, number of incomplete data items, and request frequencies of data items, and then broadcasts the hottest data item in that request each time. For comparison purposes, we also implement two data item level scheduling algorithms: Most Requests First (MRF) scheduling and R×W scheduling [20], [21]. In MRF, the data item with the largest number of pending requests will be broadcast first, while in R×W, the data item with the largest product of R and W will be broadcast first, where R denotes the number of pending requests for a data item and W denotes the longest waiting time.

B. Simulation Environment

We simulate a base station with a broadcasting channel of bandwidth 1 Megabyte per Second. The database to be broadcast has N data items. The sizes of data items vary between 10K bytes to 30K bytes and the size of one time slot is a Centisecond. Therefore a data item occupies 1 to 3 time slots. The requests are generated randomly with access probabilities following the Zipf distribution [31], which is a typical model for web data with non-uniform access patterns [32]. Let θ denote the Zipf skew factor, the probability that data item d_i is requested is $p_i = \frac{(\frac{1}{i})^\theta}{\sum_{i=1}^{N}(\frac{1}{i})^\theta}$ where $0 \leq \theta \leq 1$. In the experiments, the default values of N and θ are 1000 and 0.8 respectively, and the users submit requests with 3 to 5 data items at one time (i.e. $|Q|$ is a random number between 3 and 5).

C. Experimental Results

We perform three experiments to evaluate the practical efficiency of the algorithms. The performance metric used in the experiments is AAL (Average Access Latency). For each experiment, we simulate 1000 requests to get the AAL.

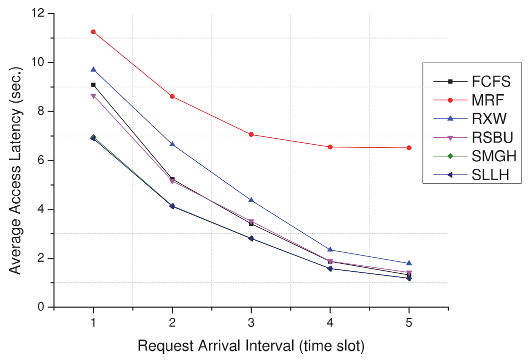

Fig. 2. AAL for different request arrival interval (I)

The first experiment is to investigate the performance of algorithms for different request arrival rates. The experimental result is exhibited in Fig. 2. Let I denote the average request arrival interval (i.e. there are about $1/I$ requests submitted per time slot), the AALs of all algorithms decrease sub-linearly as I increases. SMGH and SLLH perform similarly, and they significantly outperform their competitors, especially when I is small. For instance, when $I = 1$, the AAL for either SMGH or SLLH is less than that of RSBU by about 17%, which ranks the third in terms of minimizing AAL in this experiment. Among all the algorithms, MRF performs the worst. This is reasonable since it only broadcasts hot data items which will result in a extremely long access latency for requests containing some cold data items. From Fig. 2 we also can get that all algorithms except MRF perform similarly when I is greater than 3, probably because of the low workload (a higher value of I means a lower request arrival rate).

In the second experiment, the performance evaluation of algorithms for various θ is conducted. As the experiential result shown in Fig. 3, the AALs of all algorithms decrease slightly

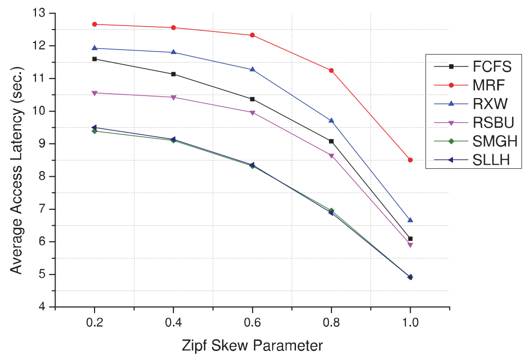

Fig. 3. AAL for different Zipf skew factor (θ)

as θ increases from 0.2 to 0.6, and they decrease greatly as θ increases from 0.6 to 1.0. In addition, all request level scheduling algorithms (FCFS, RSBU, SMGH, and SLLH) are much better than data item level scheduling algorithms (MRF and R$\times$W) for minimizing AAL, which fully shows that data item level scheduling is unable to perform efficiency for multi-item requests. MRF still performs the worst among all the algorithms. RSBU outperforms FCFS when θ is small and they perform similarly when θ is close to 1.0. When applying SMGH (or SLLH) instead of RSBU to do the request scheduling, the AAL further decreases. The AAL of SMGH (or SLLH) is about 10% to 20% less than that of RSBU.

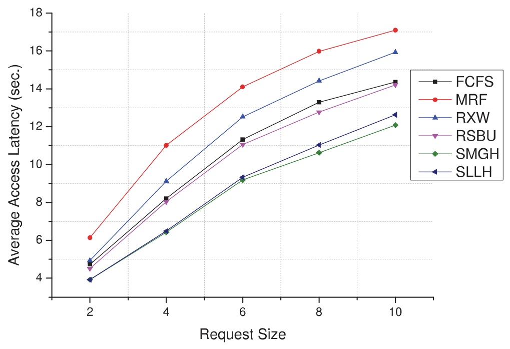

Fig. 4. AAL for different request size ($|Q|$)

In the third experiment, we investigate the impact of request size on the AAL. The experiential result is exhibited in Fig. 4. The AALs of all algorithms increase gradually as the request size $|Q|$ increases. This agrees with the intuition that the access latency of a request Q is linearly proportional to its size $|Q|$. SMGH and SLLH still perform better than other algorithms. They outperforms RSBU, FCFS, MRF R$\times$W greatly for any $|Q|$. Moreover, different from the first two experiments, the performance of SMGH is better than that of SLLH. Its AAL exactly matches that of SLLH when $|Q| \leq 6$ and is slightly less than that of SLLH when $|Q| \geq 6$. Therefore, when requesting more than 5 data items at one time, SMGH is a better option for minimizing the AAL.

Finally, we would also like to evaluate the average running time of the proposed algorithms so that we can validate the scalability of SMGH and SLLH. For this purpose, we simulate a wireless data broadcast environment with 5000 requests. We set the request arrival interval to 2 time slots and other parameters at their default values. The average running time of SMGH and SLLH are 13.1 milliseconds, 12.9 milliseconds respectively. To be specific, the request selection and ordering algorithms (Algorithms 1 and 4) require 4.8 milliseconds and 8.1 milliseconds on average, and the running time of the pruning heuristics (Algorithms 3 and 4) is negligible. Therefore, both SMGH and SLLH scale well in practice.

VI. Conclusion and Further Research

In this paper, we investigate efficient scheduling for multi-item requests in on-demand wireless data broadcast environments. To reduce the average access latency, we formally define two optimization problems, namely Maximum Throughput Request Selection and Minimum Latency Request Ordering. We analyze the two problems theoretically and finally develop efficient algorithms for them. Based on the proposed algorithms, we develop an efficient scheduling scheme for on-demand wireless data broadcasts. The theoretical analysis and simulation results demonstrate that significantly shorter average access latency can be obtained by using our scheduling scheme rather than its competitors. In the future, we would like to extend our work to the MIMO environments, in which multiple processes can be used to download data items from parallel broadcasting channels concurrently.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China under grant 61472272 and National Science Foundation of USA under grants 1137732 and 1241626.

References

[1] R. Gandhi, Y. Kim, S. Lee, J. Ryu, P.J. Wan: Approximation Algorithms for Data Broadcast in Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(7): pp. 1237-1248, 2012.

[2] Z. Lu, Y. Shi, W. Wu, B. Fu: Data Retrieval Scheduling for Multi-Item Requests in Multi-Channel Wireless Broadcast Environments. *IEEE Transactions on Mobile Computing*, 13(04): pp. 752–765, 2014.

[3] B. Zheng, W.C. Lee, P. Liu, D.L. Lee, X. Ding: Tuning On-Air Signatures for Balancing Performance and Confidentiality. *IEEE Transactions on Knowledge and Data Engineering*, 21(12): pp. 1783–1797, 2009.

[4] W. Hu, C. Xia, B. Du, M. Wu: An On-demanded Data Broadcasting Scheduling Considering the Data Item Size. *Wireless Networks*, 21(1): pp. 35–56, 2015.

[5] Z. Lu, Y. Shi, W. Wu, B. Fu: Efficient Data Retrieval Scheduling for Multi-channel Wireless Data Broadcast. *the 2012 IEEE International Conference on Computer Communications*, pp. 891–899, 2012.

[6] M. Narvekar, S. Lopes, S. Mantha: Comparative Study of Indexing Techniques for Data Dissemination in Wireless Environment. *Procedia Computer Science*, 45: pp. 671–680, 2015.

[7] C. Liu, T. Su, J. Wang, Y. Chen: Data Broadcasting for Dependent Information Using Multiple Channels in Wireless Broadcast Environments. *Journal of Parallel and Distributed Computing*, 74(9): pp. 2795–2807, 2015.

[8] C. Peng, J. Zhou, B. Zhu, H. Zhu: Complexity Analysis and Algorithms for the Program Download Problem. *Journal of Combinatorial Optimization*, 29(1): pp. 216–227, 2015.

[9] Z. Lu, W. Wu, B. Fu: Optimal Data Retrieval Scheduling in the Multi-channel Wireless Broadcast Environments. *IEEE Transactions on Computers*, 62(12): pp. 2427–2439, 2013.

[10] P. He, H. Shen, H. Tian: On-demand Data Broadcast with Deadlines for Avoiding Conflicts in Wireless Networks. *Journal of Systems and Software*, 103: pp.118–127, 2015.

[11] W. Mao: Competitive Analysis of On-line Algorithms for On-Demand Data Broadcast Scheduling. *the 2000 International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 292–296, 2000.

[12] X. Wu, V. Lee: Wireless Real-Time On-Demand Data Broadcast Scheduling with Dual Deadlines. *Journal of Parallel and Distributed Computing*, 65(6): pp. 714–728, 2005.

[13] J. Xu, X. Tang, W. Lee: Time-critical On-demand Data Broadcast: Algorithms, Analysis, and Performance Evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 17(1): pp. 3–14, 2006.

[14] S. Acharya, R. Alonso, M. Franklin, S. Zdonik: Broadcast Disks: Data Management for Asymmetric Communication Environments. *the 1995 ACM SIGMOD/PODS Conference*, pp. 199–210, 1995.

[15] C. Kenyon, N. Schabanel: The Data Broadcast Problem with Non-Uniform Transmission Time. *the 1999 ACM-SIAM Symposium on Discrete Algorithms*, pp. 547–556, 1999.

[16] K. Prabhakara, K.A. Hua, J. Oh: Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment. *the 2000 IEEE International Conference on Data Engineering*, pp. 167–176, 2000.

[17] W.G. Yee, S.B. Navathe, E. Omiecinski, C. Jermaine: Efficient Data Allocation over Multiple Channels at Broadcast Servers. *IEEE Transactions on Computers*, 51(10): pp. 1231–1236, 2002.

[18] J.L. Huang, M.S. Chen, W.C. Peng: Broadcasting Dependent Data for Ordered Queries without Replication in a Multi-Channel Mobile Environment *the 2003 IEEE International Conference on Data Engineering*, pp. 692–694, 2003.

[19] J.L. Huang and M.S. Chen. Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment. *IEEE Transactions on Knowledge and Data Engineering*, 16(9): pp. 1143–1156, 2004.

[20] H. D. Dykeman, M. Ammar, J. W. Wong: Scheduling Algorithms for Videotex Systems under Broadcast Delivery. *the 1986 IEEE International Conference on Communications*, pp. 1847–1851, 1986.

[21] D. Aksoy, M. Franklin: Scheduling for Large-Scale On-Demand Data Broadcasting. *the 1998 IEEE International Conference on Computer Communications*, pp. 651–659, 1998.

[22] W. Ni, Q. Fang, S. Vrbsky: A Lazy Data Request Approach for On-Demand Data Broadcasting *the 2003 IEEE International Conference on Distributed Computing Systems*, 2003.

[23] J. Chen, G. Huang, V.C.S. Lee: Scheduling Algorithm for Multi-Item Requests with Time Constraints in Mobile Computing Environments. *International Conference on Parallel and Distributed Systems*, pp. 1–7, 2007.

[24] J. Chen, V. Lee, K. Liu: On the Performance of Real-time Multi-item Request Scheduling in Data Broadcast Environments. *Journal of Systems and Software*, 83(8): pp. 1337–1345, 2010.

[25] K. Liu, V.C.S. Lee: On-demand Broadcast for Multi-item Requests in a Multiple Channel Mobile Environment. *Information Sciences*, 180(22): pp. 4336–4352, 2010.

[26] J. Lv, V.C.S. Lee, ,M. Li, E. Chen: Profit-Based Scheduling and Channel Allocation for Multi-Item Requests in Real-Time On-Demand Data Broadcast Systems. *Data & Knowledge Engineering*, 73: pp. 23–42, 2012.

[27] J. Juran, A.R. Hurson, N. Vijaykrishnan, S. Kim: Data Organization and Retrieval on Parallel Air Channels: Performance and Energy Issues. Wireless Networks, 10(2): pp. 183–195, 2004.

[28] A.R. Hurson, A.M. Munoz-Avila, N. Orchowski, B. Shirazi, Y. Jiao: Power Aware Data Retrieval Protocols for Indexed Broadcast Parallel Channels. *Pervasive and Mobile Computing*, 2(1): pp. 85–107, 2006.

[29] X. Gao, Z. Lu, W. Wu, B. Fu: Algebraic Algorithm for Scheduling Data Retrieval in Multi-Channel Wireless Data Broadcast Environments *the 2011 International conference on Combinatorial Otimization and Applications*, 2011.

[30] U. Feige, G. Kortsarz, D. Peleg: The Dense K Subgraph Problem. *Algorithmica*, 29: pp. 410–421, 2001.

[31] C.D. Manning, H. Schutze. *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[32] S. Jung, B. Lee, S. Pramanik: A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environment. *IEEE Transactions on Knowledge and Data Engineering*, 17(3): pp. 311–325, 2005.