A Self-Adaptive Scheduling Algorithm of On-Demand Broadcasts

Weiwei Sun, Weibin Shi, Bole Shi, Wenyun Ji and Yijun Yu
Department of Compute Science, Fudan University
Shanghai 200433, P.R. China
86-21-65646451
{wwsun, ly008136}@online.sh.cn, {blshi, wyji }@fudan.edu.cn, yijun@elis.rug.ac.be

ABSTRACT
In mobile wireless systems data on air can be accessed by a large number of mobile users. Many of these applications such as wireless internets and traffic information systems are pull-based, that is, they respond to on-demand user requests. In this paper, we study the scheduling problems of on-demand broadcast environments. Traditionally, the response time of the requests has been used as a performance measure. In this paper we consider the performance as the average cost of request composed of three kinds of costs—access time cost, tuning time cost, and cost of handling failure request. Our main contribution is a self-adaptive scheduling algorithm named LDFC , which computes the delay cost of data item as the priority for broadcast. It performs well compared with some previous algorithms in this context.

Categories and Subject Descriptors
C.2.1[**Wireless Communication**]: Computer Systems Organization – Computer – Communication Networks – Network Architecture and Design.

General Terms
Algorithms, Measurement, Performance, Experimentation.

Keywords
Mobile Computing, Data Broadcast.

1. INTRODUCTION
In a client/server architecture with fixed networks, clients would send a request when it wants to retrieve data from the server. Then the server will respond to the request and send data to clients. Compared with fixed networks, wireless networks have low bandwidth and low communication quality [10]. To support numerous mobile users to access data in server concurrently, a new method of data-transmission is put forward, that is, the server broadcasts data on air and clients could acquire data that way, so-called *data broadcasting.*

Data broadcast technology has many applications in the fields of public information dissemination, such as stock market quote or traffic and landmark information. One important issue in broadcast technology is to determine an optimal broadcast sequence according to the access probability distribution of mobile users, i.e. the data broadcast scheduling. To evaluate the effectiveness of one broadcast scheduling strategy, we need to consider two basic aspects:

(1) Access Time (shortened as AT): It indicates the time elapsed between the query submission and receipt of the response. AT determines the response time of query made by mobile users. We need to concentrate on the arrangement of frequency and location of data items in one broadcast period, so as to make the average AT least, according to various access probabilities of data items. The study on this issue includes [1, 2, 3, 5,7, 9, 13, 15], etc.

(2) Tuning Time (shortened as TT): It indicates the total time that mobile users spend actively listening on the channel in a complete access period. TT determines the power consumption of mobile users because they could slip into doze (stand by) mode when they are not actively listening on the channel. As most of mobile users depend on limited battery supply, the reduction of TT would also be an important issue in data broadcast technology. A widespread method is to insert index segments into broadcast period in order to reduce TT. The study on this issue includes [11, 12, 14], etc.

In on-demand broadcasts, we cannot obtain the access profiles of mobile users, that is to say, their access pattern would have some unpredictable changes. Thus we need a kind of new scheduling algorithm, to determine the contents and organization of data broadcast on the basis of circumstance of recent access and scheduling.

The study of on-demand broadcast scheduling problem includes [4, 8], etc. In this broadcast environment, mobile users communicate with server via wireless channels. These channels include an uplink channel and a downlink channel. Mobile users use this uplink channel to send data access request, and the contents of broadcast will arrive at mobile users through downlink channel. First, mobile users make the access request; second, the server considers all pending request to decide the contents of next broadcast. One core issue is to determine the priorities of data items to broadcast, that is, which data items should be broadcasted in next period. [5] put forward a FCFS (First-Come-First-Served) scheduling algorithm, which sequences data items according to their requested time. Because of its time sequencing principle, any access request would get responded after waiting a finite period. There doesn't exist any case of endless waiting. But it has the deficiency of low average performance, because it considers only the requested time, and doesn't take into account the difference of access frequency of various data items. MRF (Most-Request-First) scheduling algorithm will broadcast those data items with most requests priorly. As there are most-frequently data items in every broadcast, every broadcast will have the highest response ratio (*number of requests responded/number of total requests*), and we could get much lower AT. But it has its own shortcoming: If some data items have few requests, they will always line up behind several most-frequently-requested items, so that the request on these data items could always be unsatisfied and get into endless waiting. [6] suggested LWF (Long- Wait-First) algorithm, which chooses the data item that has the largest wait time (the sum of the total time that all pending requests for that item have been waiting) to broadcast. It considers both the number of requests and the wait time, so as to reduce the occurrence of endless waiting. [4] put forward LTSF (Longest-Total-Stretch-First) algorithm, which considers the factor of variable-size data items. [8] proposed a set of self-adaptive broadcast protocols—CBS/VBS protocols (including server broadcast protocol and client receipt protocol), and raised the idea of dynamic adjusting in priority computing formula.

But all these papers mentioned above didn't take into account the handling of a request waited for quite a long time. They only referred to some measures to reduce the probability of occurrence. Being unable to deal with those requests that didn't get responded for quite a long time, i.e., the permission of endless waiting, will lead to serious problems. For example, server would not receive the access request because of transmission errors, in this case mobile user (the request sender) will wait for an impossible response; responding to an access request sent a long time ago would also lead to ineffectiveness of the response, because the mobile user who sent this request could probably have left the broadcast covered area, or it would not listen to the channel for the reason of saving power.

Therefore, we should set up a Response Time Limit (*RTL*) for every access request. Mobile user sends one request and starts to listen to the contents of broadcast. If it doesn't get responded within the *RTL*, this request would be identified as a failure and mobile user would not continue to listen to. Similarly, after the broadcast server received the request sent by mobile user, if it couldn't add corresponding data item to broadcast contents within the *RTL*, it would delete the request from the request sequence.

Besides, in the determination of which item should be added to broadcast, the priority computing formula seems unable to explain strongly why those data items with low priority should be delayed. And the significance of those cost computing models is vague.

In this paper, we put forward a self-adaptive scheduling algorithm of on-demand broadcast—LDCF (Largest-Delay- Cost-First). It computes the delay cost for every data item and uses it as the priority to schedule the data items, taking into account three kinds of costs—AT, TT and request failure. The parameters of delay cost computing formula will be adjusted automatically according to recent scheduling circumstances.

The rest of the paper is organized as follows. Section 2 shows the on-demand broadcast model and defines the problem of broadcast scheduling. We also make some basic assumptions here. Section 3 shows the delay cost computing formula of data items, which indicates the increased cost if every data item wouldn't be broadcasted in next period, including access time cost, tuning time cost and request failure cost. On basis of this formula, we put forward LDCF scheduling algorithm. We describe the simulation experiments and discuss their results in Section 4. We make some conclusions in Section 5.

2. PROBLEM DEFINITION AND PRELIMINARIES

A typical on-demand broadcast system could be shown as figure 1[4].

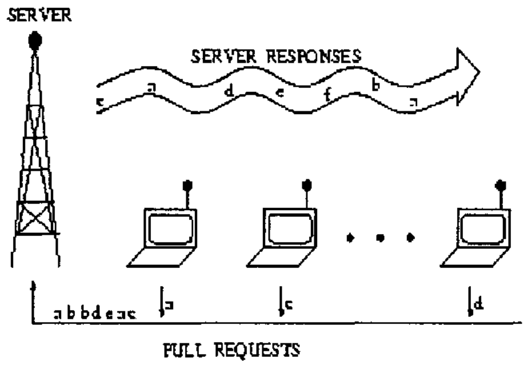


Figure 1. A typical on-demand broadcast system.

The relationship between radio transmitter (base station) and mobile users could be seen as server and clients. Mobile users are clients, and radio transmitter is the server. To the convenience of our study, we make some restrictions on the broadcast environment. Our basic assumptions are as follows.

Mobile users communicate with server via wireless information channels. These channels include an uplink channel and a downlink channel. Mobile users use this uplink channel to send data access request, and the contents of broadcast will arrive at mobile users through downlink channel. After the broadcast server receives an access request, it will respond to this request within a pre-determined response time limit, and add the requested data item in broadcasting contents; otherwise this request would be regarded as a failure. (The disposal of failed request could be in two ways: either the server would do nothing, waiting mobile user to send request again if mobile user still want to access the data item; or it could create a separate wireless link to send date item to mobile user.)

Broadcast server doesn't know the probability distribution of the access of various data items by mobile users. Therefore, the server could determine suitable broadcast scheduling only after it received those requests. (That we say the server doesn't know the access pattern of mobile users, isn't to mean that the access by mobile user hasn't any regular patterns; actually, the access by various mobile user do have some patterns.)

The least unit of broadcast is data item, and all data items are of identical size.

Data broadcast uses a kind of constant-period method, that is to say, no matter what changes taken place in the content of broadcast, the size of every broadcast is fixed.

Mobile users access one data item in each request, and any two accesses are independent.

The following are some definitions and notions that we may use in our further discussion.

Unit Time: suppose the broadcast time of one data item is 1.

Data: The number of data items in one broadcast period.

Index: The index length in one broadcast period.

BP: Broadcast Period. *BP=Data+Index.*

The structure of data broadcast: see figure 2. The former is index segment *Index,* and the latter is data segment *Data.* The size of *Index* and *Data* is fixed.

Index	*Data*

Figure 2. The structure of data broadcast.

D_i: data item, i=1..*M*. *M* indicates the total number of all data items.

Q<D, T_{req}>: indicates one access request, *D* is the data item that *Q* requests to be broadcasted; T_{req} is the time when that request is sent.

RTL: Response Time Limit. It indicates the longest time elapsed between the time when mobile user sends an access request of data item and the time when the server responds to that request. If the server could not add the requested data item to broadcast contents within this time limit, we should say that this request is failed. The server could create a separate wireless link and send data item to mobile user. If one mobile user sends a request at T_0, and at T_1 ($T_1 <= T_0+RTL$) it finds in broadcast index that the requested data item would appear at T_2 ($T_2 > T_0+RTL$). In this case, we consider that the request gets valid response.

$Cost_{AT}$: Access time cost for mobile user to obtain data item (on the basis of unit time). If one mobile user waits for 100 unit time to obtain his requested data item, then total access time cost would be 100 $Cost_{AT}$.

$Cost_{TT}$: Tuning time cost for mobile user to search the location of one data item in the index segment. If one mobile user wait for 10 broadcast period to obtain one data item, and he searches the index for 10 times, then total tuning time cost would be $10\,Cost_{TT}$.

In a pure push-based data dissemination scheduling, we have two main performance metrics: access time and tuning time. While in on-demand broadcast scheduling, we should consider not only AT and TT, but the cost of handling failed requests as well, because we have introduced the notion of request failure.

$Cost_{failure}$: Cost of handling a failed request. If the server could not respond to the access request within a determined response time *RTL*, we should use $Cost_{failure}$ as the cost of creating a separate wireless link between server and mobile user to obtain data item.

3. LDCF SELF-ADAPTIVE BROADCAST SCHEDULING ALGORITHM

3.1 Delay Cost Computing Model

The key of LDCF (Largest-Delay-Cost- First) scheduling algorithm is its Delay-Cost computing model. We can compute the cost of every request delayed one broadcast period, according to such parameters as the length of broadcast period, tuning time cost for mobile user to search for needed data item in the index section of broadcast, failure probability of access request, and the cost of handling failed request, etc. This cost is composed of three aspects: access time, tuning time, and request failure.

Some description of several notions would be given as below. Then we could illustrate the formulas of Delay Cost in our discussion.

PF_D^T : The popularity factor of data item *D* at time *T*, which indicates there are PF_D^T number of mobile users requesting to access data item *D*. The initial value of PF_D^T is zero; every time when a new request for data item *D* arrives, PF_D^T will increase by 1; when one request isn't satisfied within one *RTL*, PF_D^T will decrease by 1; if the data item *D* appears in the broadcast line, PF_D^T will be set as zero again.

SF_Q^T : The safety factor, expressed by remaining broadcast periods, which indicates there are SF_Q^T number of opportunities (excluding the next one) to satisfy request Q by broadcast at time *T*. The formula of SF_Q^T is

$$SF_Q^T = \left\lfloor \frac{T_{req} + RTL - T}{BP} \right\rfloor,$$

T_{req} stands for the sending time of request *Q*. One thing that we need mention is, when we discuss the safety factor, *T* stands for the time next broadcast begins. If the safety factor is zero, it means that if server doesn't broadcast the data item *Q* needs in the next period, then request *Q* would fail. We name it as safety factor, because we want to use it to express the "distance" of request *Q* to the failure. Obviously, in order to minimize failed requests, server should respond to those requests with lower SF_Q^T.

$Request\_No^{SF}$: it indicates the total number of pending requests whose safety factor equals to SF.

$Remained\_Request\_No^{SF}$: it indicates the total number of requests that could not be satisfied in next broadcast period, whose safety factor equals to *SF*.

$Broadcasted\_Request\_No^{SF}$: it indicates the total number of requests that could be satisfied in next broadcast period, whose safety factor equals to *SF*. Obviously,

$$Request\_No^{SF} = Remained\_Request\_No^{SF} + Broadcasted\_Request\_No^{SF}.$$

$Remain\_Rate^{SF}$: it indicates the ratio of requests that could not be satisfied in next broadcast period, whose safety factor equals to *SF*.

$$Remain\_Rate^{SF} = Remained\_Request\_No^{SF} / Request\_No^{SF}.$$

$Fail\_Rate^{SF}$: it indicates the failure probability of the requests whose safety factor equals to *SF*, if they could not be satisfied in next broadcast period.

Apparently, if one request $Q<T,D>$ with SF=0 could not be satisfied in next period, then Q will fail, i.e., $Fail\_Rate^0 = 1$.

When SF>0, for request $Q<T,D>$, if D could not be satisfied in next broadcast period, the SF of Q will decrease by 1; the probability of those requests (safety factor=SF-1) that could not be satisfied immediately is $Fail\_Rate^{SF}$, the probability of those requests that could not be satisfied immediately and finally get failed is $Fail\_Rate^{SF-1}$, thus

$$Fail\_Rate^{SF} = Remain\_Rate^{SF-1} \cdot Fail\_Rate^{SF-1}.$$

$Delay\_Cost_Q$: The increased cost if request Q be delayed and could not be satisfied in the next broadcast period.

Lemma 1: The cost of request Q if it would be delayed

$$Delay\_Cost_Q = BP \cdot Cost_{AT} + Cost_{TT} + Fail\_Rate^{SF_Q^T} \cdot Cost_{failure}.$$

$Delay\_Cost_Q$ is composed of three parts: the first part indicates the access time cost increased because of delay, the second part indicates the tuning time cost increased because of delay, and the third part indicates the estimated cost of request failure because of delay.

P_D ($Delay\_Cost_D$): The increased cost if data item D be delayed and not appear in the next broadcast period, i.e. the priority of data item D. *Data* numbers of data items with highest priority would be broadcasted in the next period.

Theorem 1: The cost of data item D if it would be delayed

$$P_D = \sum_{Q<D,T_{req}>} Delay\_Cost_Q = PF_D^T \cdot (BP \cdot Cost_{AT}) + PF_D^T \cdot Cost_{TT} + \sum_{Q<D,T_{req}>} Fail\_Rate^{SF^T_{Q<D,T_{req}>}} \cdot Cost_{failure}.$$

Both P_D ($Delay\_Cost_D$) and $Delay\_Cost_Q$ include three parts: Access Time Cost, Tuning Time Cost, and Request Failure Cost.

BP, $Cost_{AT}$, $Cost_{TT}$, $Cost_{failure}$ are pre-determined constants, while PF_D^T , $Fail\_Rate^{SF^T_{Q<D,T_{req}>}}$ will change along with recent circumstances of access and broadcast. When the failure rate rises, those requests with low *SF* will be satisfied first; when the failure rate goes down, those data items with high *PF* will be broadcasted priorly.

If $Cost_{failure}$=0, LDCF will degenerate to MRF, and the priority of data item D, P_D , is in direct proportion to the number of pending requests for data item D, PF_D .

3.2 LDCF Scheduling Algorithm

We describe LDCF scheduling algorithm as follows:

Algorithm 1: LDCF

```
Input: request sequence;
Output: a broadcast scheduling;
Proceeding:
main()
{
  failed_rate[]:=[1,0,0,...,0]
  time=0;
  while true do
  {
    for i:=1 to BP
    {time=time+1;
     receive the new requests {req<Di, time>}, and add them to
        RequestSequence;
    }
    LDCF(time);
  }
}

procedure LDCF(time)
{
  for each data item Di
    DataItem[Di].priority:=0;
  for each pending request req<Di, req_time>
      in RequestSequence
  {
    SF:= $\left\lfloor \frac{req\_time + RTL - time}{BP} \right\rfloor$;
    DataItem[Di].priority:=
        DataItem[Di].priority+BP* $Cost_{AT}$ +
        $Cost_{TT}$ +fail_rate[SF]* $Cost_{failure}$;
    }
    select Data number of data items with largest priority from DataItem[];
    add these data items into broadcast period sorted by the value of PF (in descending order), and make the index;
    compute fail_rate[] once again;
    delete those requests that have been responded or failed in RequestSequence;
}
```

In above description of process, we mainly focus on the illustration of LDCF algorithm, therefore some implementation details have been omitted. For example, when one request *req<Di, req_time>* would not get responded and fail because of time out, we didn't make a concrete analysis on creating direct wireless link between server and mobile user to send requested data items. Also, we will not fully explain how to select data items with largest priority, how to add broadcast contents and make index, etc.

4. EXPERIMENTS AND COMPARISONS

We intend to use simulation method to compare LDCF scheduling algorithm with MRF, FCFS and LWF algorithms, so as to evaluate the performance of LDCF scheduling algorithm.

At each time, the server will receive access requests from mobile users, compute the priority of every data item on the basis of all pending requests, then select *Data* number of data items with largest priority and add them to broadcast contents.

4.1 Experimental Data

The numerical distribution of new arrived requests during one time period: suppose the probability that every mobile user will send request during one time period is *p*, the number of mobile user is *MU_no*, then the probability that number of new requests equals to *new_request_no* is:

$$p^{new\_request\_no} \cdot C_{MU\_np}^{new\_request\_no}$$
$$\cdot (1-p)^{MU\_no-new\_request\_no} \cdot C_{MU\_np}^{MU\_no-new\_request\_no}.$$

The numerical distribution of data items required by new requests during one time period: we use function Zipf(*k*) to describe the skewed distribution of data access. In generating the distribution of data access with Zipf(*k*), we suppose the skewness *k* at any time could change randomly in one interval. Besides, we would randomly select 10% data items, multiply their distribution results by a random number between 0 to 10.

We use the randomizer provided by http://www.randomizer.org to generate bench- mark random numbers for our experiments.

4.2 Experiment Results and Analysis

(1) Parameter settings

The following are some common parameters:

Parameter	Meaning
M	The number of data items that Server could be used to broadcast. Suppose it is 1000 in the following experiments.
Data	The number of data items in one broadcast period
Index	The length of index section in one broadcast period. Suppose it is 6 in the following experiments.
Request number per unit time	The number of requests that server would receive at each time
k	Parameter of function Zipf, indicating the skewness of data access distribution
RTL	Response time limit
$Cost_{AT}$	Cost of AT per unit time. Suppose it is 1 in the following experiments.
$Cost_{TT}$	Cost of TT per seeking index. Suppose it is 20 in the following experiments.
$Cost_{failure}$	Cost of handling a failed request. Suppose it is 2000 in the following experiments.

(2) Experiment 1: Performance when fail rate of request is low

First, we will discuss the performance comparison of LDCF algorithm with other three algorithms in the situation of low workloads. The setting of parameters is shown as follows. We will consider the effect of various *RTL* on Average Cost of request. The result is shown in figure 3.

Parameter	Value
Data	100
Request number per unit time	10.10
k	0
RTL	1500~3500

As *RTL* increases, Average Cost of request will decrease. When *RTL*>=2500, there aren't any failed requests and all requests are satisfied in MRF scheduling, that is, all data items whose SF=0 are belonged to those data items with largest *PF*. In LDCF scheduling, all data items with SF=0 are belonged to those data items with largest $Delay\_Cost_D$, too. Therefore, at this time LDCF and MRF are identical, both of them are optimal scheduling algorithm, which have least average AT and TT. The performance of LWF is a little worse than LDCF and MRF, while FCFS is the worst.

In short, LDCF and MRF occupy the first place, LWF comes second, and FCFS is the worst.

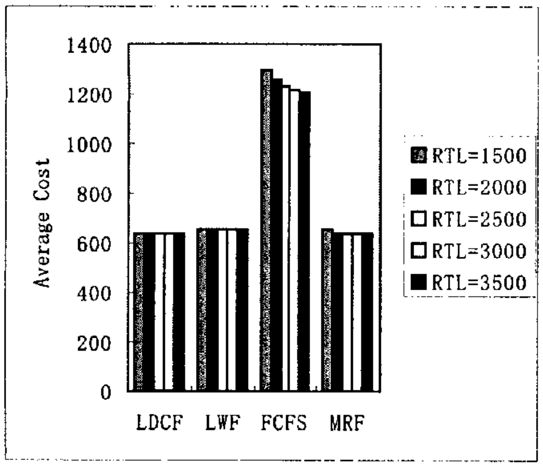


Figure 3. Performance for various algorithms when fail rate of request is low

(3) Experiment 2: Performance when fail rate of request is high

In this experiment, we will discuss the performance comparison between LDCF and other three algorithms when fail rate of request is high. The setting of parameters is shown as follows. The result is shown in figure 4.

Parameter	Value
Data	120
Request number per unit time	247.15
k	-1.5~1.5
RTL	1500

If the skewness *k* of data access randomly changes between [-1.5, 1.5], lots of requests will be failed when there are many requests at each unit time.

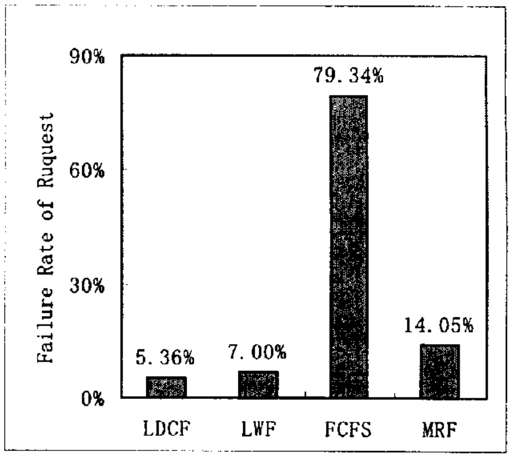
The fail rate of LDCF scheduling is the lowest, and its average cost lowest too. The performance of LWF scheduling is a little worse than that of LDCF.

FCFS scheduling has much higher fail rate and larger average cost. It shows that the average performance of FCFS scheduling is unsatisfactory, because it only considers time factor, not the number of requests.

The performance of MRF scheduling still lags behind of LDCF and LWF. It also shows that it is insufficient to consider only the number of requests, not the time factor.

We won't compare FCFS and MRF with our algorithm in further discussion.

In short, LDCF could efficiently reduce the number of failed requests, and it has least average cost. Consideration of only one factor (request number or time, as in the case of MRF and FCFS) will lead to lots of request failure.



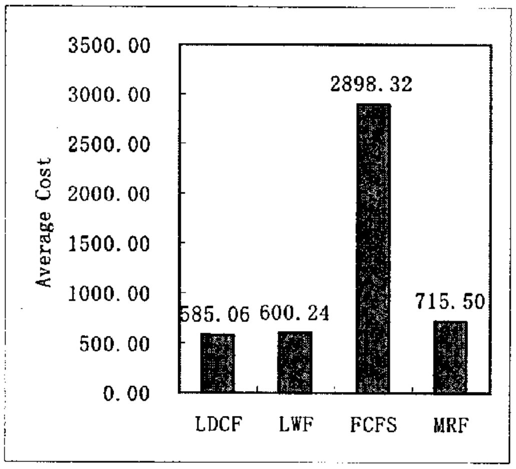


Figure 4. Performance for various algorithms when fail rate of request is high

(4) Experiment 3: Effect of *Data*

In this experiment, we will discuss the effect of length of data segment in one broadcast period. The setting of parameters is shown as follows. The result is shown in figure 5.

Parameter	Value
Data	60~200
Request number per unit time	98.86
k	0
RTL	1500

In this experiment, our conclusion is: the number of data items contained in one broadcast period should be moderate. Too small a value will drastically increase the average cost, but once its value increase above one certain point, average cost will rise instead. Still, the performance of LDCF is better than that of LWF.

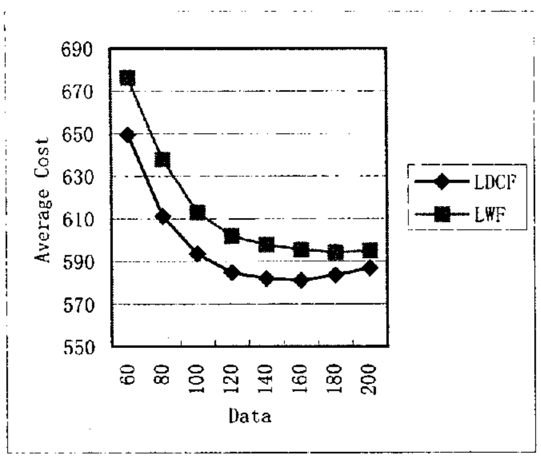


Figure 5. Effect of *Data*

(5) Experiment 4: Effect of *RTL*

In this experiment, we will discuss the effect of Response Time Limit. The setting of parameters is shown as follows. The result is shown in figure 6.

Parameter	Value
Data	100
Request number per unit time	98.86
k	-1.5~1.5
RTL	800~2000

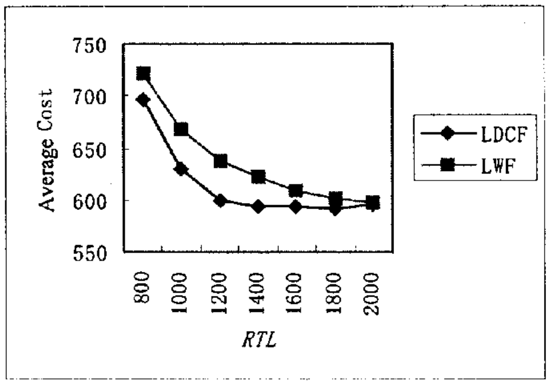


Figure 6. Effect of *RTL*

In this experiment, we conclude that *RTL* the larger, average cost the lower. When *RTL*>1200, the performance gap between two algorithms is very little. Again, LDCF shows some advantages over LWF.

(6) Experiment 5: Effect of skewness of data access distribution

In this experiment, we will discuss the effect of skewness *k* of data access distribution.

First, we will consider the cases when skewness *k* is a certain value. The setting of parameters is shown as follows. The result is shown in figure 7.

Parameter	Value
Data	100
Request number per unit time	98.86
k	0~1.5
RTL	1000

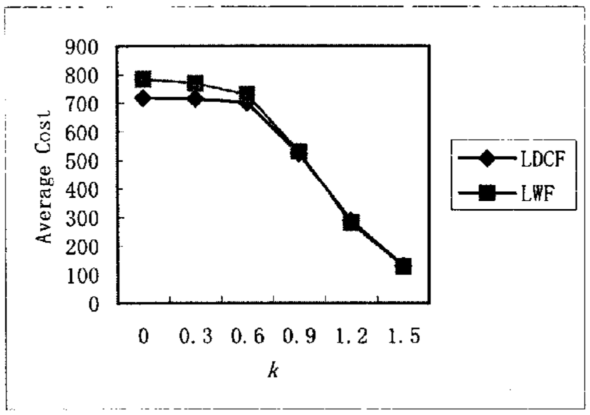


Figure 7. Effect of skewness *k* with a certain value

Second, we consider the cases when skewness *k* is a random number in a certain interval centered on zero. (The skewness *k* might be different at any time)

The setting of parameters is shown as follows. The result is shown in figure 8.

Parameter	Value
Data	100
Request number per unit time	98.86
k	[0, 0], [-0,3, 0,3] ... [-1.5, 1.5]
RTL	1000

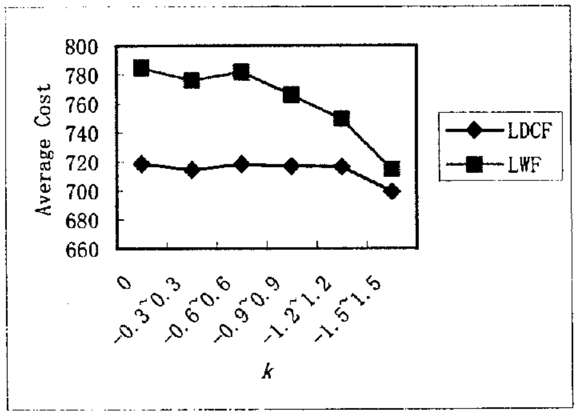


Figure 8. Effect of skewness *k* with a certain interval

In this experiment, our conclusion is: when skewness *k* holds one certain value, the average cost of LDCF will decrease as *k* increases. Its overall performance is superior to that of LWF.

When skewness k is a random number in a certain interval centered on zero, the average cost of LDCF shows little influence of interval size.

5. CONCLUSION

In this paper, we have studied the problem of scheduling on-demand broadcasts. Compared with pure push-based data dissemination scheduling, we need uplink channel to send data access request in an on-demand broadcast-based environment. The server would not know the access profiles of mobile users, and it should take into account the situation when request fails because of time out.

Previous works in this context mainly discuss how to reduce the average AT of mobile users. In practical applications, the handling of a request waited for quite a long time must be considered and so we introduce the notion *request failure*. When discussing the performance of a scheduling algorithm of on-demand broadcasts, we take into account not only AT, but also TT and request failure. We put forward a self-adaptive scheduling algorithm —LDCF, It computes the delay cost for every data item and uses it as the priority to schedule the data items, the parameters of delay cost computing formula will be adjusted automatically according to recent scheduling circumstances.

Our work raises the open algorithmic problem of determining a schedule that minimizes the average cost of request considering all kinds of cost—AT, TT and failure. We compare LDCF with LWF, FCFS and MRF via several experiments, which indicate the average cost of LDCF scheduling was the least.

6. REFERENCES

[1] Acharya, S., Alonso, R., Franklin, M. and Zdonik, S. Broadcast Disks: Data management for asymmetric commu- nications environments. In Michael J. Carey and Donovan A. Schneider, editors, Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, 1995.

[2] Acharya, S., Franklin, M. and Zdonik, S. Disseminating updates on broadcast disks. In: Proc 22nd VLDB Conf, 1996.

[3] Acharya, S., Franklin, M. and Zdonik, S. Dissemination-based data delivery using Broadcast Disks. IEEE Personal Communications, 1995,2(6): 50-60.

[4] Acharya, S. and Muthukrishnan, S. Scheduling on-demand broadcasts: new metrics and algorithms. Proceeding of ACM/IEEE MobiCom, Dallas, TX, 1998.

[5] Aksoy, D. and Franklin, M. Scheduling for large scale on-demand data broadcast. In Proc.. of IEEE INFOCOM, San Francisco, CA, 1998.

[6] Bender, M., Chakrabarti, S. and Muthukrishnan, S. Flow and stretch metrics for scheduling continuous job streams. In Proceedings of the Ninth Annual ACM - SIAM Symposium on Discrete Algorithms, San Francisco, California, 1998.

[7] Chiueh, T. Scheduling for broadcast-based file systems. In: Proc MOBIDATA Workshop, 1994.

[8] Datta, A., Vandermeer, D. E., Celik, A. and Kumar, B.V. Broadcast protocols to support efficient retrieval from database by mobile users. ACM TODS, 1999, 24(1) 1-79.

[9] Gondhalekar, V. A. Scheduling periodic wireless data broadcast. MS Thesis, the University of Texas at Austin, 1995.

[10] Imielinski, T. and Badrinath, B. R. Mobile wireless computing: solutions and challenges in data management. Communications of the ACM, 1994, 37(10).

[11] Imielinski, T., Viswanathan, S. and Badrinath, B. R. Energy efficient indexing on air. In: Proc ACM SIGMOD 1994.

[12] Shivakumar, N. and Venka, S. Efficient indexing for broadcast-based wireless systems. ACM Mobile Networks and Nomadic Applications, 1996, 1(4): 433-446.

[13] Su, C. and Tassiulas, L. Broadcast scheduling for information distribution. In Proceedings of IEEE INFOCOM, Los Alamitos, CA USA, April 1997.

[14] Viswanathan, S. and Imielinski, T. Pyramid broadcasting for Video on Demand service. Technical Report DCS TR-311, Rutgers University, 1994.

[15] Wong, J. W. Broadcast delivery. Proceedings of the IEEE, 1988, 76(12): 1566-1577.