Adaptive Data Delivery in Wireless Communication Environments

Chi-Wai Lin Dik-Lun Lee
Computer Science Department
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{lcw, dlee}@cs.ust.hk

Abstract

The combination of broadcast and on-demand data delivery services is an economic way to build a highly scalable wireless information system with limited bandwidth. The use of data broadcasting should be adaptive so that the system response time can always be minimised. A traditional approach requires the development of a system response time equation in order to find the optimal solution. However, obtaining such an equation is not always possible. We observe that by maintaining a certain level of on-demand request arrival rate, a close approximation to the optimal solution can be obtained. Using this approach, a real-time adaptive data delivery algorithm is developed. Our algorithm does not require the access information of the data items to be known exactly, which is needed normally for this kind of optimization problems. A simple and low overhead bit vector mechanism is able to capture the relative popularities of the data items. With this information, our algorithm can give a performance comparable to the ideal case in which the access information for each data item is known exactly.

1. Introduction

In a mobile computing environment, scarce wireless bandwidth, asymmetric communication, limited battery power and mobility of the mobile clients are the major challenges to the design of a mobile system [7]. The main focus of this paper is on the effective use of limited bandwidth in a wireless communication environment. One of the major research problems in this area is to design a wireless information system to provide mobile clients with wireless access to a central database. A similar access pattern is assumed to be shared among clients. With limited wireless bandwith, the system should be able to provide wireless data access to as many clients as possible. At the same time, the system response time should always be minimized under different system workloads.

A promising approach to building such a system is to use both on-demand and broadcast data delivery services that complement each other [2, 5, 6, 3, 8, 9, 11]. To minimize the response time of the system, usage between on-demand and broadcast data delivery services must be balanced. This can be done by providing a certain number of frequently accessed data items on the broadcast channel according to the workload, while the rest are serviced by the on-demand channels. Traditionally, this requires the development of an explicit system response time equation [8, 11, 9]. Based on this, together with the exact [8, 9] or estimated [11] request arrival rates of the data items, the optimal number of broadcast data items can be approximated [8, 11, 9]. However, developing such an equation is not always possible. Besides, only the best achieved system response time under different system workloads has been previously considered [8, 11, 9]. None of them have considered the real-time adaptiveness of the system in reaction to the changes in the system workload.

In this paper, we consider the situation when the development of an explicit system response time equation is impossible, and, the system workload is assumed to be changing dynamically in real-time fashion. Without knowing the exact request arrival rates but only the relative popularities of the data items, we want to approximate the optimal system response time under different system workloads with a real-time adaptive data delivery algorithm.

The rest of this paper is organised as follows. Section 2 shows our wireless data delivery model. Section 3 gives a detail description of our adaptive data delivery algorithm. Section 4 describes our simulation model and discusses about the experimental results. Finally, Section 5 concludes the paper and suggests possible future research directions.

2. Wireless Data Delivery Model

2.1. System Architecture

The architecture of the system adopted in our model is the same as the one in [6]. A single broadcast channel and a set of on-demand point-to-point channels are used in a single cell environment. A server (Mobile Support Station) provides mobile clients with wireless access to a central database through the broadcast and on-demand channels. All data items in the database are of the same size. One data item can be stored in a single frame on the broadcast channel. The data items are read-only and will not be updated by the clients or server. Unlike the model in [6], there is no queueing buffer for the on-demand channels.

A flat broadcast is used for the broadcast data [1]. The use of a broadcast channel is to avoid overloading the on-demand channels. As the workload of the on-demand channels changes, the number of data items on the broadcast channel k will be increased or decreased. The value k is embedded in every frame sent on the broadcast channel. It is used by the clients to determine how many frames on the broadcast channel they should listen to before issuing an on-demand request to the server.

2.2. Collecting Data Access Information

To make necessary adaptive adjustments of the broadcast data items, data access information has to be collected continuously. Access information of the data requests made through the on-demand channels can be obtained by the server without extra cost. However, the server does not know which data items of the broadcast channel have been accessed by the clients. In view of this, the MFA[1] vector mechanism proposed in [5] is adopted.

This mechanism requires each client to maintain a MFA vector and a broadcast version number. A MFA vector is basically a bit vector in which each bit represents a data item on the broadcast channel. Whenever a request is answered by a broadcast data item, the bit corresponding to the accessed data item is set. The broadcast version number is used by the server to ensure the validity of the relationship between the bit positions and the data items. It is incremented whenever the set of broadcast data items is changed.

Whenever an on-demand request is sent to the server, the MFA vector and the broadcast version number will also be piggybacked to the server at the same time. The server keeps a request received record for each data item. Whenever an explicit on-demand request for a data item is received or a corresponding bit is set for a data item in a valid MFA vector[2] received, the corresponding request received record of that particular data item is incremented by one.

[1] MFA stands for Most Frequently Accessed.

3. Real-Time Adaptive Data Delivery

3.1. Optimal System Response Time

To approximate the optimal system response time, we have made the following observations. Since on-demand channels can provide instant access to data items, they should be fully utilized. However, on-demand channels must not be overloaded. Otherwise, the system response time will be unbounded. Thus, the idea of utilizing the on-demand channels as much as possible while not overloading them requires a certain level of on-demand request arrival rate to be maintained.

3.2. Difficulty in Maintaining On-demand Request Arrival Rate

Assuming that there are n data items in the server database and λ_d is the set containing the request arrival rate for each of the data items such that λ_{d_i} is the request arrival rate for data item i. The workload of the system is characterized by the aggregate request arrival rate of the data items in the server database. It is given in the following equation [9]:

$$\lambda = \sum_{i=1}^{n} \lambda_{d_i} \tag{1}$$

Suppose, an on-demand request arrival rate λ'_o is to be maintained. When $\lambda > \lambda'_o$, the k most popular data items have to be identified and put onto the broadcast channel, while the rest of the data items are serviced by the on-demand channels so that:

$$k = Min\{j | \sum_{i=1}^{j} \gamma_i \geq \lambda - \lambda'_o\} \tag{2}$$

where $\gamma_i \in \lambda_d$ and $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_n$. The idea is to control the amount of data on the broadcast channel k so that the on-demand request arrival rate is less than or equal to λ'_o.

When $\lambda \leq \lambda'_o$, the system workload is so low that all requests can be served by the on-demand channels. In

[2] A valid MFA vector is defined as a MFA vector with the same broadcast version number as the current broadcast version number used by the server.

this case, a frame containing one of the most frequently accessed data items will still be broadcast. The algorithm for the selection of broadcast data is shown in Figure 1.

Select-Broadcast-Data(λ_d, λ'_o)

$\lambda \leftarrow$ j $\leftarrow$ k $\leftarrow 0$
List-Clear(List-Of-Broadcast-Data)

for $i \leftarrow 1$ **to** n
do $\lambda \leftarrow \lambda + \lambda_{d_i}$

$\gamma[] \leftarrow$ Sort-In-Decending-Order(λ_d)

if $\lambda \leq \lambda'_o$
then List-Insert(List-Of-Broadcast-Data, γ[1].data-id)
else **while** j $< \lambda - \lambda'_o$
  do k $\leftarrow$ k + 1
    j $\leftarrow$ j + γ[k].value
    List-Insert(List-Of-Broadcast-Data, γ[k].data-id)

return List-Of-Broadcast-Data

Figure 1. Algorithm for Broadcast Data Selection

This technique requires the request arrival rates, and hence, access frequencies of data items, to be known exactly by the server. However, the exact access information cannot be obtained by using the MFA vector mechanism. Therefore, the algorithm cannot be used directly to maintain the on-demand request arrival rate in our model.

3.3. Techniques to Maintain On-demand Request Arrival Rate

3.3.1. Basic Idea

Figure 2 shows the relationship between the on-demand channel response time and the on-demand request arrival rate. According to the on-demand request arrival rate, four on-demand channel states are defined: *TOO LOW*, *SUB-OPTIMAL*, *TOO HIGH* and *UNSTABLE*. The *Sub-Optimal Point* is the on-demand

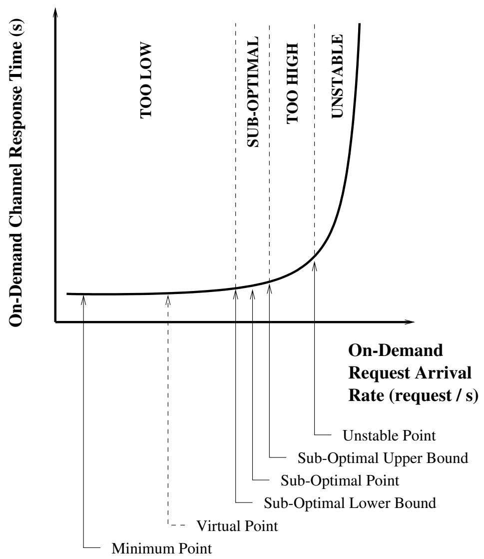


Figure 2. On-Demand Channel Response Time VS On-Demand Request Arrival Rate

request arrival rate that we want to maintain. It is assumed that either one of the *previous state* or *current state* of the on-demand channels or both are within the range between the *Sub-Optimal Upper Bound* and *Sub-Optimal Lower Bound*; the on-demand request arrival rate at the *Sub-Optimal Point* is regarded as successfully maintained.

The basic idea of our real-time adaptive data delivery algorithm is that, whenever the on-demand request arrival rate cannot be maintained at the *Sub-Optimal Point*, the amount of data on the broadcast channel will be increased or decreased according to the on-demand channel states. To decide which data item is to be put on or removed from the broadcast channel next, relative popularities of data items are used.

3.3.2. Controlling the Number of Broadcast Data Items

To control the increase or decrease of the number of broadcast data items, the broadcast data selection algorithm shown in Figure 1 is used indirectly by maintaining a virtual request arrival rate called *Virtual Point*. The name virtual is used because the *Virtual Point* is used to assist the system to maintain the on-demand request arrival rate at the *Sub-Optimal Point*, rather than at the *Virtual Point* itself.

Suppose the current on-demand request arrival rate is too high, then the *Virtual Point* will be decreased until the on-demand request arrival rate can be maintained at the *Sub-Optimal Point*. In fact, decreasing the *Virtual Point* has the effect of increasing the number of broadcast data items. Once enough data items are broadcast, the desired on-demand request arrival rate can be maintained. Similarly, when the current on-demand request arrival rate is too low, then the *Virtual Point* will be increased until the on-demand request arrival rate can be maintained at the *Sub-Optimal Point*.

3.3.3. Adjusting the Virtual Point to Control the Number of Broadcast Data Items

Two additional control parameters which are not shown in Figure 2 are used to control the adjustment speed of the *Virtual Point*:

- *Normal Step:* a request arrival rate stepping used for the adjustment of the *Virtual Point* when the on-demand channels are not overloaded.

- *Large Step:* a larger request arrival rate stepping used for the adjustment of the *Virtual Point* when the on-demand channels are overloaded.

Virtual Point can be adjusted between the *Sub-Optimal Point* and the *Minimum Point*. The decision for the adjustment of the *Virtual Point* is made according to the *previous state* and *current state* of the on-demand channels:

- *Case I:* If both the *previous state* and *current state* of the on-demand channels are UNSTABLE, the *Virtual Point* would be decreased with a *Large Step*. It is because when the on-demand channels are overloaded, system response time can be increased exponentially. Therefore, the decrease of the Virtual Point must be able to stabilize the system as soon as possible.

- *Case II:* If only the *current state* of the on-demand channels is *UNSTABLE*, or both the *previous state* and *current states* are *TOO HIGH*, the *Virtual Point* would be decreased by one or more *Normal Steps*. The amount of decrease would be accumulative[3].

- *Case III:* If both the *previous state* and *current state* of the on-demand channels are *TOO LOW*, the *Virtual Point* would be increased by one or more *Normal Step*. The amount of increase would also be accumulative.

- *Case IV:* If either the *previous state* is *TOO LOW* and the *current state* is *TOO HIGH*, or the *previous state* is *TOO HIGH* and the *current state* is *TOO LOW*, it means that the previous increase or decrease made to the *Virtual Point* has been too much. Instead of decreasing or increasing the *Virtual Point* incrementally from the very beginning, a mid-point would be taken from the previous and current *Virtual Points*. This approach helps to speed up the time required for the on-demand channels to reach the *SUB-OPTIMAL* state.

Figure 3 shows the pseudo-code for the adjustment of the *Virtual Point*.

3.4. Computation of Request Arrival Rates

Request arrival rate λ_{d_i} for data item i is computed with an exponentially weighted moving average technique [11]:

$$\lambda_{d_i,j} = (1-\alpha)\lambda_{d_i,j-1} + \alpha\frac{r_{i,j}}{t_j} \tag{3}$$

where $\lambda_{d_i,j}$ is the request arrival rate of data item i for the j-th evaluation period, $0 < \alpha \leq 1$, $r_{i,j}$ is the number of requests received corresponding to data item i during the j-th evaluation period and t_j is the length of the j-th evaluation period.

On-demand request arrival rate is computed with the following equation:

$$\lambda_{o,j} = \frac{o_j}{t_j} \tag{4}$$

where $\lambda_{o,j}$ is the on-demand request arrival rate for the j-th evaluation period, o_j is the total requests received through the on-demand channels over the j-th evaluation period and t_j is the length of the j-th evaluation period.

3.5. Evaluation Time for the System

Evaluation period e_{cycle} in terms of number of broadcast cycles is obtained with the following equation:

$$e_{cycle} = Min(c_{max}, \lfloor\frac{t_{max}}{t_{cycle}}\rfloor) \tag{5}$$

where c_{max} is the *Maximum Evaluation Cycle*, t_{max} is the *Maximum Evaluation Time*, t_{cycle} is the time required for a complete broadcast of the current set of broadcast data items and we have assumed $t_{max} \geq t_{cycle}$. Basically this equation means that evaluation period equals to an integer multiple of broadcast cycles not more than the *Maximum Evaluation Cycle* and the total length in time does not exceed the *Maximum Evaluation Time*.

[3] When the same condition is sustained consecutively, more and more *Normal Steps* would be used. For instance, when the condition holds for the first time, one *Normal Step* would be used. When the condition holds consecutively for the second time, two *Normal Steps* would be used and so on.

```
Adjust-Virtual-Point(Previous-State,
                     Current-State)

temp = Virtual-Point

if Previous-State = UNSTABLE and
   Current-State = UNSTABLE
then Increase ← Decrease ← 0
     Virtual-Point ←
       Virtual-Point − Large-Step

else if ((Previous-State = UNSTABLE or
          Previous-State = TOO-HIGH) and
         Current-State = TOO-HIGH) or
        Current-State = UNSTABLE
then Increase ← 0
     Decrease ← Decrease + Normal-Step
     Virtual-Point ←
       Virtual-Point − Decrease

else if Previous-State = TOO-LOW and
        Current-State = TOO-LOW
then Decrease ← 0
     Increase ← Increase + Normal-Step
     Virtual-Point ←
     Virtual-Point + Increase

else   Increase ← Decrease ← 0

       if (Previous-State = TOO-HIGH and
           Current-State = TOO-LOW) or
          (Previous-State = TOO-LOW and
           Current-State = TOO-HIGH)
       then Virtual-Point ←
              (Previous-Virtual-Point +
               Virtual-Point) / 2

Previous-Virtual-Point ← temp

if Virtual-Point > Sub-Optimal-Point
then Virtual-Point ← Sub-Optimal-Point

if Virtual-Point < Minimum-Point
then Virtual-Point ← Minimum-Point

return Virtual-Point
```

Figure 3. Algorithm for the Virtual Point Adjustment

3.6. Real-Time Adaptive Data Delivery Algorithm

With all the techniques described above, a real-time adaptive data delivery algorithm is developed. Once the system starts, a dummy data item is first broadcast for a short period of time, so that the system can compute the request arrival rates based on the number of requests received through the on-demand channels. As long as the system is up, the following steps are repeated. The request arrival rate for each data item and the on-demand request arrival rate are computed with Equation 3 and 4 respectively. Then the *current state* of the on-demand channels is determined according to the on-demand request arrival rate. After that the *Virtual Point* is adjusted according to the *previous state* and *current state* of the on-demand channels using the *Virtual Point* adjustment algorithm shown in Figure 3. Next, the set of broadcast data items is determined with the broadcast data selection algorithm shown in Figure 1, according to the *Virtual Point* and the request arrival rates of the data items. The selected set of broadcast data items will then be broadcast for a specific number of cycles. The number of cycles is determined by Equation 5.

The running time of our algorithm is $O(n \log n)$. It is attributed to the need for sorting the request arrival rates of the data items in the broadcast data selection algorithm shown in Figure 1. The pseudo-code for our real-time adaptive data delivery algorithm is shown in Figure 4.

4. Experiments and Results

4.1. Simulation Model

Our simulation program is implemented with C and Csim [10]. The simulation model used in the experiments is shown in Figure 5.

4.1.1. Server Model

The server has a database with size, *Database Size*. Each data item has a size of *Data Item Size*. The server is modeled by two sub-processes, namely *broadcast manager* and *pull manager*.

There is a single broadcast channel with bandwidth, *Broadcast Bandwidth*. The *broadcast manager* is responsible for broadcasting data adaptively on the broadcast channel according to the system workload. The adaptive broadcast algorithm shown in Figure 4 is used.

There are *On-Demand Channel* number of on-demand channels, each having a bandwidth of *On-*

Adaptive-Broadcast()

Evaluation-Start-Time $\leftarrow$ Clock

Broadcast(Dummy-Items)

while SYSTEM-UP
do Evaluation-Time $\leftarrow$
Clock - Evaluation-Start-Time

for $i \leftarrow 1$ **to** n
do $\lambda_{d_i} \leftarrow (1-\alpha)\lambda_{d_i} + \alpha \dfrac{\text{Request-Received-For-Data-Item}_i}{\text{Evaluation-Time}}$

$\lambda_o \leftarrow \dfrac{\text{On-Demand-Requests-Received}}{\text{Evaluation-Time}}$

Previous-State $\leftarrow$ Current-State

if $\lambda_o \geq$ Unstable-Point
then Current-State $\leftarrow$ UNSTABLE
else if $\lambda_o >$ Sub-Optimal-Upper-Bound
then Current-State $\leftarrow$ TOO-HIGH
else if $\lambda_o <$ Sub-Optimal-Lower-Bound
then Current-State $\leftarrow$ TOO-LOW
else Current-State $\leftarrow$ SUB-OPTIMAL

Virtual-Point $\leftarrow$
Adjust-Virtual-Point(Previous-State,
Current-State)

Set-Of-Broadcast-Data $\leftarrow$
Select-Broadcast-Data(λ_d, Virtual-Point)

cycle $\leftarrow$ Min(Maximum-Evaluation-Cycle,
$\lfloor \dfrac{\text{Maximum-Evaluation-Period}}{\text{Time-For-A-Broadcast-Cycle}} \rfloor$)

Evaluation-Start-Time $\leftarrow$ Clock

for $i \leftarrow 1$ **to** cycle
do Broadcast(Set-Of-Broadcast-Data)

Figure 4. Algorithm for Adaptive Broadcast

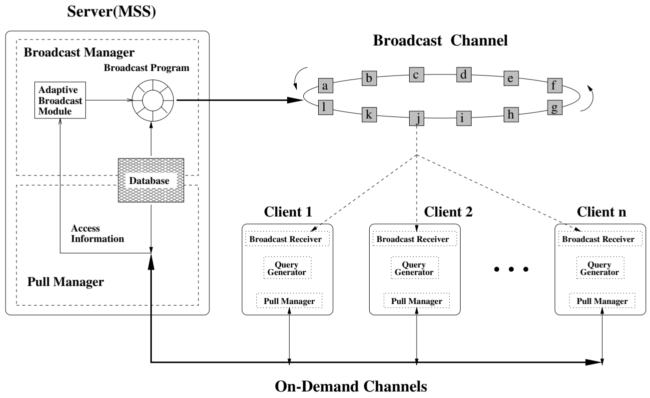


Figure 5. Simulation Model

Demand Bandwidth. The *pull manager* is responsible for granting free on-demand channels to clients.

4.1.2. Client Model

To simulate the contention of the on-demand channels among clients, each client is modeled by an independent process. It is assumed that each client can have a single request at a time, so a client will not issue a new request unless the previous request is completed. Each request addresses a single data item only.

For each request, a client waits for a random amount of time drawn from a negative exponential distribution with the mean of *Think Time* before generating another data request with the *query generator*. The *query generator* randomly selects a data item from a Gaussian distribution [4], such that 68% of the generated requests fall within a particular set of data items with size, *Hot Spot Size*. After that, the request is directed to the *broadcast receiver*.

The *broadcast receiver* first reads a frame from the broadcast channel. The frame contains *1)* a data item, *2)* a number k, which specifies the number of data on the broadcast channel, *3)* the bit position in the MFA vector corresponding to this particular data item, and *4)* the broadcast version number. Whenever a new broadcast version number is detected, the MFA vector will be reset. If the data item received is not the desired one, the *broadcast receiver* continues to examine up to $k-1$ succeeding frames on the broadcast channel. If the desired data item is encountered, the corresponding bit in the MFA vector is set. Otherwise, the request is directed to the *pull manager*.

The *pull manager* first checks if an on-demand channel is available. If not, it waits for a random period of time which is drawn from a uniform $[0, \textit{Wait Time})$ distribution. This is repeated until a point-to-point connection is set up between the client and the server.

After the connection is established, the *pull manager* sends a pull request with size *Pull Request Size* to the server. At the same time, the broadcast version number with size, *Version Size* and the MFA vector are also sent to the server. The server then replies with the requested data item to the *pull manager* through the same channel.

4.2. Parameter Settings

The system parameters and control parameters used in the simulations are shown in Table 1 and Table 2 respectively.

Parameter	Value
Broadcast Bandwidth	80000 bps
On-Demand Bandwidth	8000 bps
Broadcast Channel	1
On-Demand Channel	10
Database Size	1000 data items
Hot Spot Size	50 data items
Data Item Size	8000 bits
Pull Request Size	128 bits
Version Size	16 bits
Think Time	10 s
Wait Time	2 s
α	0.5

Table 1. System Parameter Settings

Parameter	Value
Sub-Optimal Point	8 requests/s
Sub-Optimal Upper Bound	8.3 requests/s
Sub-Optimal Lower Bound	7.7 requests/s
Unstable Point	9.4 requests/s
Minimum Point	0.5 request/s
Normal Step	0.2 request/s
Large Step	2.5 requests/s
Maximum Evaluation Cycle	20 cycles
Maximum Evaluation Time	120 s

Table 2. Control Parameter Settings

4.3. Experiment 1: Relationship between Optimal System Response Time and On-Demand Request Arrival Rate

This experiment shows the relationship between the optimal system response time and on-demand request arrival rate. Since an explicit system response time equation cannot be obtained for our model, the optimal system response time is found by simulation only. Client access pattern is assumed to be unchanged. To find the optimal system response time, a fixed number of most frequently accessed data items are broadcast under a specific client population. The number of data items is tried exhaustively from 1 to n which is the number of data items in the database. The one which gives the minimum system response time is regarded as the optimal solution[4].

As shown in Figure 6(a), for the IDEAL[5] case, maintaining the on-demand request arrival rate at 8 requests/s can obtain a nearly optimal system response time under different client populations. Even for the MFA[6] case, as shown in Figure 6(b), maintaining the on-demand request arrival rate at 8 requests/s can also obtain a performance close to the optimal solution.

It should be noted that when the on-demand request arrival rate is maintained at 6 requests/s, the performance loss is not too large. It means that, the choice of the *Sub-Optimal Point* is not too critical to the system performance. As long as the setting of the *Sub-Optimal Point* is reasonably far away from the *Unstable Point*, a reasonably good system performance can already be obtained.

From Figure 6(b), it is found that the gap between the 8 requests/s line and the optimal line gets larger when the system workload is higher. It is because when the system workload is higher, more data items will be broadcast, so the fraction of MFA vectors received by the server is smaller[7]. Hence, the accuracy of the captured relative popularities of the data items is lower, so the system performance gets worse.

4.4. Experiment 2: Real-Time Adaptive Performance of the System

In this experiment, system performance under real-time change of client population is studied. The client access pattern is assumed to be unchanged and the system response time is sampled every 60 seconds in the experiment.

Figure 7 shows the real-time system performance under the change of client populations. Each line in the sub-figure shown in Figure 7 represents a different

[4] The optimal lines shown in both Figure 6(a) and 6(b) are identical.

[5] When the data access information is perfectly known.

[6] When MFA vector mechanism is used to collect the data access information.

[7] If the requests of a client can all be answered by the broadcast data, the MFA vector would not be sent to the server.

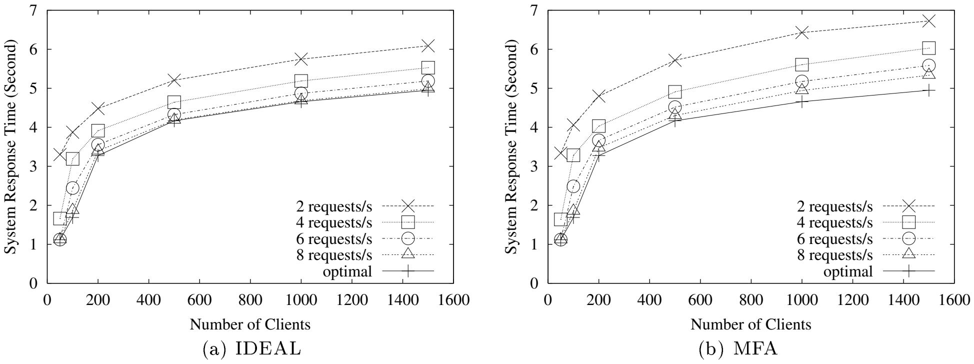


Figure 6. Different On-demand Request Arrival Rates

number of clients in the server during the intermediate period of the simulation time.

From 0 to 1000 seconds, there are only 50 clients in the server. Then the number of clients increases linearly during the period 1000 to 2000 seconds from 50 clients to 100, 200, 500, 1000 and 2000 clients respectively for each line shown in each sub-figure in Figure 7. After that, the number of clients remains unchanged until 5000 seconds. Finally, the number of clients decreases linearly back to 50 clients during the period 5000 to 6000 seconds. In fact, each line models a different rate change of client population.

It is found that the length of evaluation period, and hence, the choice of the *Maximum Evaluation Cycle* is very critical to the real-time system performance. It affects the adaptability and stability of the system.

As shown in Figure 7(a), when a small *Maximum Evaluation Cycle* is used, the transition of the system to the change of client population from 1000 to 2000 seconds is the smoothest. It is because the system is evaluated more frequently, so an appropiate adjustment can always be made before the on-demand channels approach the overloaded state. Nevertheless, the system response time from 2000 to 5000 seconds is significantly higher and more fluctuating than the other cases. It is because the evaluation period used in Figure 7(a) is too short, so the relative popularities of the data items captured are not accurate enough. Therefore, some less popular data items are broadcast while some more popular data items are not broadcast, resulting in a poor performance.

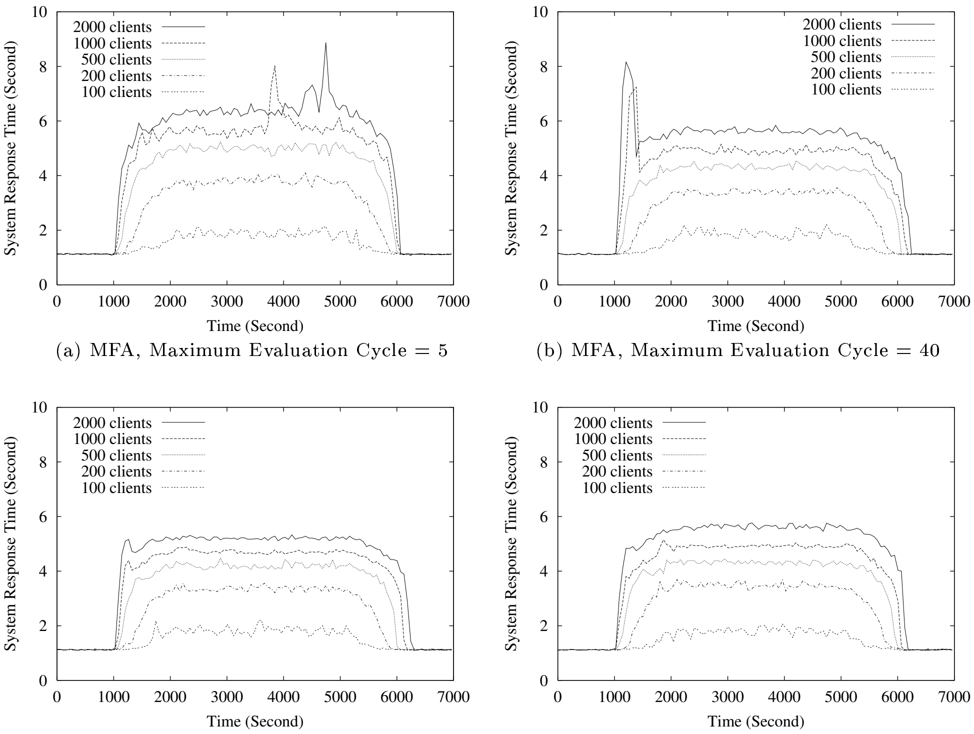
As shown in Figure 7(b), if the *Maximum Evaluation Cycle* chosen is too large, the evaluation period used during the system transition from 50 to 1000 clients and 50 to 2000 clients becomes too long. Therefore, the on-demand channels can get into the overloaded state easily before any adjustment can be made by the system. Nevertheless, once the instability of the system, and hence, overloading of the on-demand channels, is detected, our adaptive data delivery algorithm can stabilize the system quickly. Since the evaluation period used is long enough for a fairly accurate capturing of the relative popularities of the data items, the system performance from 2000 to 5000 seconds in Figure 7(b) is a lot better than that in Figure 7(a).

With a proper setting of the evaluation period, as shown in Figure 7(d), both the stability and reactiveness of the system can be maintained. As shown in Figure 7(c) and 7(d), performance of the MFA case can be very close to the IDEAL case. Since the relative popularities captured by the MFA vector are not 100% accurate, so the performance of the MFA case is slightly worse than the IDEAL case.

5. Conclusion

In this paper, a real-time adaptive data delivery algorithm is presented. By maintaining a certain level of on-demand request arrival rate, our algorithm can approximate the optimal system response time under different system workloads. Most importantly, it does not require the development of a system response time equation nor require to know the exact request arrival rate for each data item. These features are highly advantageous because the development of a system response time equation is not always possible and the requirement for the system to know the exact request arrival rate for each broadcast data item is impractical.

Our algorithm can also be applied to other hybrid data delivery models with different system architectures. For instance, models which use a shared uplink channel to receive requests from the clients can maintain the uplink channel utilization with our algorithm to obtain a reasonably close to optimal system response



(a) MFA, Maximum Evaluation Cycle = 5

(b) MFA, Maximum Evaluation Cycle = 40

(c) IDEAL, Maximum Evaluation Cycle = 20

(d) MFA, Maximum Evaluation Cycle = 20

Figure 7. Real-Time Change of Client Population

time.

Since our adaptive data delivery algorithm is developed on an ad hoc basis, for future work, we would like to develop an algorithm with the use of control theory. For instance, we would like to see how fuzzy control theory [12] can be applied to maintain the on-demand request arrival rate.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 199–210, San Jose, California, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing push and pull for data broadcast. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 183–194, Tuscon, Arizona, May 1997.

[3] A. Datta, A. Celik, J. Kim, D. VanderMeer, and V. Kumar. Adaptive broadcast protocols to support efficient and energy conserving retrieval from databases in mobile computing environments. In *Proceedings of IEEE International Conference on Data Engineering*, Birmingham, England, April 1997.

[4] S. Goldberg. *Probability: An Introduction.* Dover Publications, Inc., New York, 1986.

[5] Q. L. Hu, D. L. Lee, and W.-C. Lee. Dynamic data delivery in wireless communication environments. In *Workshop on Mobile Data Access*, pages 213–224, Singapore, November 1998.

[6] Q. L. Hu, D. L. Lee, and W.-C. Lee. Performance evaluation of a wireless hierarchical data dissemination system. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking*, Seattle, USA, August 1999.

[7] T. Imielinski and B. R. Badrinath. Mobile wireless computing : Challenges in data management. *Communication of ACM*, 37(10), 1994.

[8] T. Imielinski and S. Viswanathan. Adaptive wireless information systems. In *Proceedings of SIGDBS Conference*, Tokyo, Japan, October 1994.

[9] W.-C. Lee, Q. L. Hu, and D. L. Lee. A study of channel allocation methods for data dissemination in mobile computing environments. *Mobile Networks and Applications (MONET)*, To appear.

[10] H. Schwetman. *Csim user's guide (version 17).* MCC Corporation, 1992.

[11] K. Stathatos, N. Roussopoulos, and J. S. Baras. Adaptive data broadcast in hybrid networks. In *Proceedings of the 23rd VLDB Conference*, pages 326–335, Athens, Greece, August 1997.

[12] L. X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis.* Prentice-hall: Engleweed Cliffs, NJ, 1994.