A New Hybrid Scheduling Framework for Asymmetric Wireless Environments with Request Repetition

Navrati Saxena
Dept. of Informatics & Telecom.
University of Trento
Trento, Italy
Email:navrati@dit.unitn.it

Cristina M. Pinotti
Dept. of Maths & Informatics
University of Perugia
Perugia, Italy
Email: pinotti@unipg.it

Kalyan Basu and Sajal K. Das
CReWMaN Laboratory
Compute Science & Engineering Department
University of Texas at Arlington
Arlington, TX, USA
Email: {basu, das}@cse.uta.edu

***Abstract*— The ever-increasing popularity of web services, growing demand for wireless multimedia and introduction of new, feature-enhanced, hand-held devices has already given birth to a new set of data-centric applications. Providing such applications with enhanced data processing capability calls for an efficient scheduling and transmission technique. The goal of most scheduling strategy lies in reducing the average waiting time. However, in most practical systems the variation of waiting time often results in client's impatience, thus provoking the clients to send repeated requests for the particular data item(s). In this paper we have developed a new hybrid scheduling framework for heterogeneous, asymmetric environments, by exploring the advantages of broadcasting very popular (push) data and dissemination of less popular (pull) data. The data access probabilities and the cut-off point used to segregate push and pull sets are dynamically computed. Packet Fair Scheduling (PFS) and stretch-optimal scheduling principle is deployed to obtain the push and pull schedule respectively. The framework explicitly takes care of the repeated requests originating from the impatient clients and minimizes the overall expected access time by obtaining an optimal cut-off point. Extensive performance analysis and simulation experiments are performed to show the efficiency of the system in reducing the overall expected access time (delay).**

I. Introduction

The significant upsurge in the web-applications over the past few years, has already resulted in the arrival of new information-centric applications, which needs huge data storage and processing. While today's Internet still provides best-effort data service, significant researches are focussed on development of real-time web applications. The major objective of most real-time applications lies in reduction of delay or client's waiting time. A close look into the data dissemination-based wireless systems reveals that there exists significant asymmetry in the system arising from difference in uplink and downlink bandwidth, message-size and number of clients and server. These data dissemination applications can be broadly categorized into *push-based* and *pull-based* systems. In push-based systems, the clients continuously monitor a broadcast process from the server and obtain the data items they require, *without making any requests*, e.g., stock quotes on a web browser. In contrast, in a *pull-based* system, the clients initiate the data transfer by sending requests *on demand*, which the server schedules to satisfy, e.g., stock quotes from a financial web-site.

Although, there exists some works on hybrid scheduling schemes in the literature [1], [9], [6], however, but none of those schemes considers the following factors simultaneously:

- *Heterogenous System:* The above hybrid scheduling schemes assume that data items have the same length (size). However, in most practical systems, the data items are of varying sizes. In this situation, the most fair approach is to serve the data items of lower length more quickly than those of higher length. Thus, the decision to choose a particular data item for service should be dependent on its length.
- *Dynamic computation of data access probabilities:* The dynamics of today's Internet and web services need an online hybrid scheduling scheme, which can adapt to the changes of the system. However, existing hybrid scheduling techniques [1], [9], [6] make a priori assumption of data access probabilities.
- *Client Impatience and Anomalies:* In practical systems, the clients often looses their patience, while waiting for a particular data item. This often compels the client to send multiple requests for the required data item. This *repeat-attempt* behavior of the clients need to be carefully studied and analyzed to get a suitable and correct picture of the system.

The above discussions motivate us to design an efficient, online/dynamic, hybrid scheduling strategy, which will take clients' impatience into consideration, while minimizing the overall waiting time in heterogenous, asynchronous communication environments. In particular, this paper proposes a dynamic, hybrid scheduling that effectively combines broadcasting of more popular data (push data) and dissemination upon-request for less popular data (pull data), for asymmetric heterogeneous environments. In this approach, the server continuously broadcasts one push item and disseminates one pull item. The clients send their requests to the server, which queues them up for the pull items. At any instance of time, the item to be broadcast is selected by applying a flat scheduling, while the item to be pulled is chosen from the pull-queue by applying *stretch-optimal (max-request min-service-time first)* scheduling algorithm. In order to develop a better and practical framework, our approach also takes into account the repeated requests by *impatient* clients due to to prolonged waiting. The performance of the proposed hybrid scheduler is analyzed to derive the expected waiting time. The cut-off point between push and pull items is chosen so that the overall waiting time of the system is minimized.

Specific contributions and novelty of our work are summarized below.

1) We design a new dynamic hybrid algorithm for heterogeneous (variable length) data items. The varying length of data items introduces variation in service time. In such systems *stretch* is the most fair measure to select the request for a particular data item. The existing stretch-optimal scheduling, proposed in [24] works for pure push systems only. According to our knowledge we are the first to consider stretch as selection criteria in hybrid scheduling.
2) The *cut-off point* that separates the push and the pull sets is computed dynamically to better analyze the system behavior and more practical scenarios. This has the power to achieve an optimal performance.
3) In most practical systems, the clients often get impatient while waiting for the designated data item. The clients' impatience resulting from their prolonged waiting for any item, or a new requests for the same data item by another client often makes them to transmit repeated requests. The server keeps these repeated requests in the *repeat-attempt (retrial)* queue, thereby distinguishing such requests from the new requests arriving in the pull queue. The service of an item from the pull queue needs to consider the service of the instances of same items from the repeat-attempt queue also. Using a multi-dimensional Markov model the average performance of the overall heterogenous, hybrid scheduling system is derived.
4) Simulation experiments are conducted starting with a pure pull system, without prior knowledge of the data access probabilities. Later, the probabilities are re-computed dynamically to determine the cut-off-point between the push and the pull sets. Experimental results closely match the analytical results.

The rest of the paper is organized as follows: The related work in data broadcasting and scheduling is reviewed in Section II. Section III introduces the new heterogeneous, hybrid scheduling framework, which explicitly considers the repeated trials of the clients present in the system, in addition to normal client-requests. An appropriate performance analysis of the repeat-attempt hybrid system is developed to estimate the average behavior of the system in Section IV. We perform simulation results in Section V to demonstrate the efficiency of our proposed repeat-attempt hybrid scheduling strategy. Finally, Section VI concludes the paper.

II. Related Work

As mentioned earlier, scheduling algorithms can be broadly classified into push and pull based techniques. In order to achieve optimal expected access time, the broadcast scheduling has been related to the *packet fair queuing* problem [9]. However, the optimal solution for data broadcast problems with non-uniform transmission time and multiple data items [12] are shown to be NP-hard, thus leading to the development of various approximation techniques. The existence of a simple optimal schedule composed of repetitive patterns of files with dependencies has been proved in [4]. A three-player *client-provider-server* model [5], which logically separates the servers from service providers, is proposed for asymmetric communication environments. In order to capture the time-varying dynamism of mobile client's data accesses, a broadcasting strategy needs to be adaptive. The solution proposed in [18] performs dynamic adjustment of *channel allocation trees* to adapt with the changes of clients' data access frequencies. In another adaptive data dissemination scheme [10] for asymmetric wireless environment, the dynamics of broadcast information is subsumed into groups and an online slot-exchange policies between different groups is proposed to alleviate the adverse effects of heavy dynamic traffic. An algorithmic framework to provide near-optimal jitter approximation tradeoff for periodic scheduling is introduced in [7]. Asymptotically optimal algorithms for obtaining data access-costs, which are not proportional to their own waiting time, are provided in [6]. However, we also need to consider the clients' behavior that often affects the performance of the dynamic system as follows. While waiting for a data item, a client might get impatient and leave the system. The scheme in [11] is the only work that considered this effect for push-based broadcasting. Using an exponential distribution for inter-arrival time, this work focussed on minimizing the average delay, while maximizing the service ratio for broadcasting.

On the other hand, many preemptive and non-preemptive, pull-based algorithms like First Come First Served (FCFS), Most Requests First (MRF) and $R \times W$ algorithm [3] exist in the literature. While MRF provides optimal waiting time for popular set of items, it suffers from unfairness. Although FCFS is fair, it suffers from sub-optimality and *convoy effect*. A combination of these two as suggested in [3] often provides an acceptable solution. While the performance of the on-demand pull systems are often estimated by *response time*, recently the average of access time cost, tuning time cost and cost of handling failure [22] is proposed as a more appropriate performance metric. The pull-based real-time data dissimilation system, discussed in [8], proposes Aggregated Critical Requests (ACR) scheduling algorithm to meet the pre-determined timing constraints. A suitable broadcast scheduling scheme is developed in [21] to compare the performance of push and pull based systems.

While push-based scheduling is suitable for asymmetric environments, it often wastes resources by repeatedly broadcasting less popular items. This is solved by on-demand pull-based scheduling. Unfortunately, pull-based scheduling suffers from resource scarcity in uplink channels. Naturally, a hybrid approach that combines the flavors of both push-based and pull-based algorithms in one system, appears to provide a better solution. perhaps the first hybrid technique for scheduling and data transmission in asymmetric environment was proposed in [2]. In this work, the server pushes all the data items according to some push-based scheduling, but simultaneously the clients are provided with a limited back channel capacity to make requests for the items. An $O(n)$ dynamic channel allocation strategy for broadcast push and on-demand pull systems are investigated in [13]. The adaptive real-time data transmission strategy in [14] combines broadcast push and on-demand pull strategies to achieve a near-optimal system response time even with inexact data access information. Instead of transmitting the request for data items, in the lazy data request approach [15], the clients first monitor the channel condition for a specific time. If the required data item is already being broadcasted, then the client does not send any request to the server; otherwise the client transmits the request and the data items are delivered on demand. The user-retrial phenomenon and its effects on wireless hybrid-broadcast services is discussed in [23]. Our previous work on hybrid scheduling [20] partitions a set of heterogenous data items into two sets according to their access probabilities. Subsequently, it minimizes the overall waiting time by broadcasting (push) the higher probable set of items and disseminating (pull) the lower probable set.

III. New Hybrid Scheduling Strategy

Figure 1 highlights the overview of a *repeat-attempt system*. In the conventional communication, any request which finds the terminal busy is put on the waiting queue. In a repeat-attempt model however, a request which finds the server busy checks whether the item is in the waiting queue. If not, the item is kept in the waiting queue. If the item is already in the waiting queue, it is stored in the repeat-attempt queue. This forms the basis of our newly-proposed repeat-attempt hybrid scheduling system. The database at the server consists of a total number of D distinct, heterogeneous items, out of which K items are pushed and the remaining $(D-K)$ items are pulled. The access probability P_i of an item i, i.e., the popularity of the items amongst the clients, is governed by the Zipf's distribution and depends on the access skew-coefficient (θ). From time to time the value of θ is changed dynamically for our hybrid system, thus varying P_i of all items and hence varying the size of the push and the pull sets dynamically.

The server maintains the database of all variable-length items. Periodically the server pushes the data items using a broadcast schedule. We have used the Packet Fair Scheduling (PFS) principle [9], which schedules the data items in an order such that two consecutive instances of the same data items are always equally spaced. When a client needs an item i, it sends to the server its request for item i and waits until it listens for i on the channel. If the request is for a push item, the server simply ignores the request as the item will be pushed according to the PFS algorithm. However, if the request is for a pull item, then the server first checks whether it is a new item-request from a client or it is a request for the same data item by another client. If it is a request for a new item, it inserts the request into the pull queue with the arrival time and updates its stretch value. On the other-hand, if the request is not a new one, i.e., some other client has already requested the item, the server considers it as a *repeat attempt* from an impatient client, inserts the item into the repeat-attempt (retrial) queue and updates its stretch-value. After every push, if the pull queue is not empty,

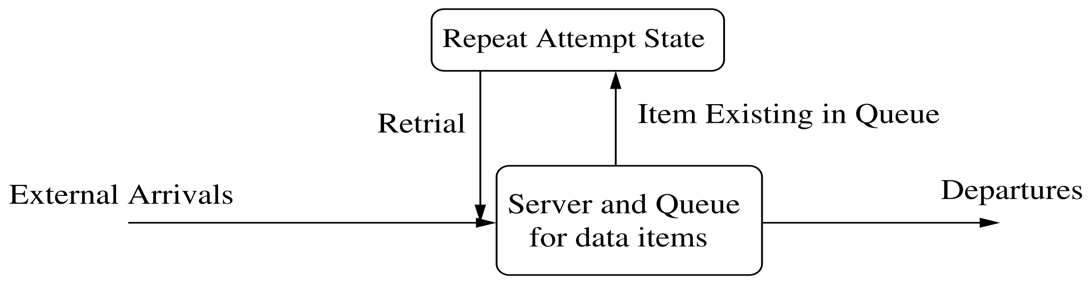

Fig. 1. Overview of Repeat Attempt System

the server chooses one item based on optimal stretch value, i.e, the item having *max-request min-service-time* value from the pull-queue. It now pulls that item and clears the pending requests for that item in the pull-queue. Subsequently, the server now checks the repeat-attempt queue and clear the requests associated with the instances of the same item. Figure 2 provides the pseudo-code of the repeat-attempt, heterogeneous hybrid scheduling algorithm executing at the server-side, where the procedure *Access and pull()* is depicted in Figure 3.

```
Procedure Hybrid Scheduling with
          Retrials;
while (true) do
begin
    Broadcast all pages of an item,
    selected according to the PFS;
    Access and pull();
    if (pull-queue is not empty) then
       extract an item, from pull
       queue, that optimizes stretch;
       if (tie)
         extract the item with the
         smallest index;
       clear the number of pending
       requests for this item in the
       pull queue;
       clear the pending requests for
       the instance of the same item
       in the repeat-attempt queue;
       pull the particular item;
    Access and pull();
end;
```

Fig. 2. Hybrid Scheduling Algorithm with Repeat-Attempts

```
Procedure Access and Pull();
while (true) do
begin
    take a specific number of accesses
    after broadcasting each page;
    if(the request is for push-item)
     ignore the request;
    else-if(the request is for pull-item)
        if(new request)
         insert the request into the
         pull queue with arrival time;
        else
         mark the request as a
         repeat-attempt;
         insert the request into the
         repeat-attempt queue;
end;
```

Fig. 3. Access and Pull Scheduling

IV. Performance Analysis of the Hybrid Repeat Attempt System

In normal pull-based scheduling strategy, the clients send explicit request to the server and the server queues the requests. The item with maximum requests or maximum stretch (request/square of length) is selected for service. However, in real systems often the clients are *impatient*, i.e., they often send multiple requests for a data item while it is not being serviced. Similarly, if a data item is already requested by a client and is waiting for service, and another client requests the same data item, the item is also considered as repeat-attempt item. In these scenarios, the data items having multiple requests are assumed to be in a new state, termed *repeat attempt state*.

We have assumed Poisson's arrival and exponential

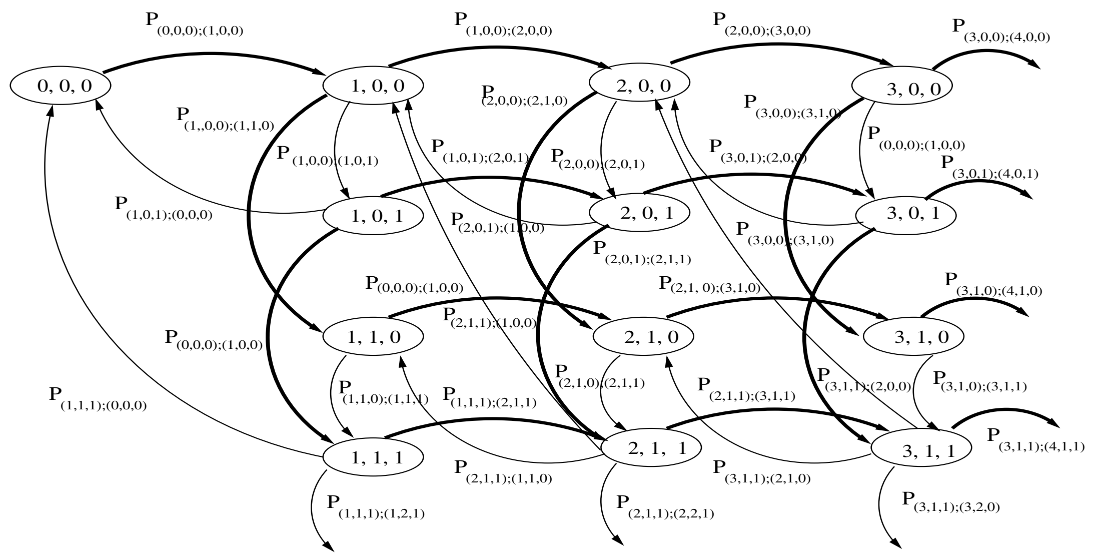

Fig. 4. Repeat Attempt Markov Model of Hybrid Scheduling

service of the items to make the analysis mathematically tractable. Figure 4 shows the schematic diagram of such a multi-dimensional Markov model representing the repeat-attempt hybrid system. Any state of the system is represented by (x, y, z), where x represents number of unique items in the pull queue ($0 \leq x \leq D - K$) and y represents number of repeat-attempt items in the repeat-attempt queue and $z = 0$ (or 1) represents push (or pull) system is currently under operation. The average arrival rate of the pull queue is assumed as λ. On the other hand, the arrival in the repeat-attempt queue is assumed to be directly proportional of the number of items present in the pull queue. Thus, the arrival rate in the repeat-attempt queue is taken as $x\xi\lambda$, where ξ is the scaling factor based on per item's average repeat attempt probability. We denote the transitional probability associated with transition from any state (x, y, z) to any another state (x', y', z') by $P_{(x,y,z);(x',y',z')}$. A careful insight into the system, shown in Figure 4 demonstrates the following major transitions:

1) There is only single transition possible from initial (idle) state $(0, 0, 0)$. This happened with probability $P_{(0,0,0);(1,0,0)}$ during the arrival of any item in the pull system.
2) Arrival of any item in the pull queue results in transition of state in both the push and pull system from $(x, y, 0)$ and $(x, y, 1)$ to $(x + 1, y, 0)$ and $(x+1, y, 1)$ with probabilities $P_{(x,y,0);(x+1,y,0)}$ and $P_{(x,y,1);(x+1,y,1)}$ respectively.
3) Similarly, arrival of any item in the repeat-attempt queue results in transition of states in the repeat-attempt system from $(x, y, 0)$ and $(x, y, 1)$ to $(x, y + 1, 0)$ and $(x, y + 1, 1)$ with probabilities $P_{(x,y,0);(x,y+1,0)}$ and $P_{(x,y,1);(x,y+1,1)}$ respectively.
4) Service of an item in the push system results in transition of states from $(x, y, 0)$ to $(x, y, 1)$ with probability $P_{(x,y,0);(x,y,1)}$. However, depending on the number of repeated attempts, the service of an item in the pull system can result in transition of states from $(x, y, 1)$ to $(x - 1, y, 0)$, $(x - 1, y - 1, 0)$, ..., $(x - 1, 0, 0)$ with probabilities $P_{(x,y,1);(x-1,y,0)}$, $P_{(x,y,1);(x-1,y,0)}$, ..., $P_{(x,y,1);(x-1,0,0)}$ respectively. When the pull system contains only a single element, the service of an item results in transition from $(1, y, 1)$ to $(0, 0, 0)$ with probability $P_{(1,y,1);(0,0,0)}$.

For example, referring to the states $(2, 0, 0)$ (push with 2 items) and $(2, 0, 1)$ (pull with 2 items) in Figure 4, the arrival of a new pull-item with arrival rate λ in the system, leads to the transition into state $(3, 0, 0)$ and $(3, 0, 1)$ with probability $P_{(2,0,0);(3,0,0)}$ and $P_{(2,0,0);(3,0,1)}$ respectively. Similarly, arrival of a repeat-attempt item at these two states with an arrival rate $2\xi\lambda$ results in transition into the state $(2, 1, 0)$ and $(2, 1, 1)$ with probability $P_{(2,0,0);(2,1,0)}$ and $P_{(2,0,1);(2,1,1)}$ respectively. We have assumed strictly reciprocal service of a push and pull item. The average service rate of the push system is assumed to be μ'. Such a service of an item from the push system, indicates that the next service will be from the pull system. Referring to the same state, i.e., $(2,0,0)$ in Figure 4, the service of the item results in transition from state $(2,0,0)$ to state $(2,0,1)$ with probability $P_{(2,0,0);(2,0,1)}$ and service rate μ'. However, the service of an item results in different possibilities, because the item currently getting serviced might be present or absent in the repeat-attempt queue. If it is present in the repeat-attempt queue, then the number of entries of that particular item in the repeat-attempt queue also needs to be cleared. Hence, service from state $(2,1,1)$ results in transition to either of the states $(1,0,0)$ or $(1,1,0)$ with probabilities $P_{(2,1,1);(1,0,0)}$ and $P_{(2,1,1);(1,1,0)}$ with service rates μ_1 and μ_2 respectively.

In order to get the estimates of these probabilities (P), first we need to derive the probabilities of selecting a particular item for service from the pull-queue and repeat-attempt queue. Subsequently, we need to obtain the relations between different service rates and measure for transition probabilities of the Markov Chain. We first proceed to find out the selection probabilities of different data items in the pull and Repeat Attempt queue. Since, there are x number of items currently present in the pull system, the actual items could be any combination of x elements chosen from total m data items in the system. Obviously, there are $\kappa = \binom{m}{x}$ number of combinations possible. We denote the combination by $\vec{C} = \{\vec{C}_1, \vec{C}_2, \ldots, \vec{C}_\kappa\}$, where every $\vec{C}_j$ is a x-element vector. Every element of this vector is a data item. We can select an element i from any of these vectors in $\binom{x}{1}$ ways. Now, once we have chosen i from a particular vector every other item of the remaining $x-1$ items can be chosen from any element of the available vectors. It should be noted that same items can not be repeated, as repeated items reside in the *repeat-attempt queue*. In other words, any item selected can not be re-selected again. Hence, if p_i represents the access probability of item i, then probability $Pr[i]_Q$ of choosing any item i from the pull queue (without repetition) is given by the relation:

$$Pr[i]_Q = \binom{x}{1}[p_i \sum_{j_1=1, j_1 \neq i}^{x} p_{j_1} \cdots \sum_{j_\kappa=1, j_\kappa \neq i, j_\kappa \neq j_z, \forall z<\kappa}^{x} p_{j_\kappa}] \quad (1)$$

However, it should be noted that since the pull queue does not contain the repeated instances of the items, the sum of total probability of the queue is less than 1. Hence all such probabilities $Pr[i]_Q$ need to be normalized. Hence the normalized probability is now given by:

$$Pr[i]_{norm} = \frac{Pr[i]_Q}{\sum_{j=1}^{\kappa} Pr[\vec{C}_j]} \quad (2)$$

where $Pr[\vec{C}_j]$ represents the probability of all the items belonging to the vector $\vec{C}_j$.

We now investigate into the Repeat-Attempt queue, where the elements can be repeated. They can be repeated once, twice or up to a maximum of m-times. We are looking to obtain the probability of this repetition of elements. Proceeding in the similar approach as in Equation (1), we can obtain the probability of a particular item i to be repeated any number of times in the Repeat-Attempt queue. Let, $(Pr[i]_{Repeat})_y$ denotes the probability that the item i is repeated y times in the Repeat-Attempt queue. Now, for the first time, the item i can still be selected in $\binom{x}{1} = x$ different ways. However, since i will be repeated once more, after choosing it for once, it can still be selected in x ways for the second time and there-after. The other terms for the remaining items can be chosen from any element of the available vectors. The restriction that the item can not be repeated (as in the pull queue) no longer exists in this repeat-attempt queue. Hence, proceeding in a similar way, the probability $(Pr[i]_{Repeat})_y$ that there are y number of repetition of the item i is given by the equation:

$$(Pr[i]_{Repeat})_y = [\sum_{j_1=1}^{x} \cdots \sum_{j_y=1, j_y \neq i}^{x} \cdots \sum_{j_\kappa=1, j_\kappa \neq i}^{x} p_{j_1} \ldots p_{j_y} \ldots p_{j_\kappa}] \times x^y p_i, \quad (\forall y, 1 \leq y \leq m) \quad (3)$$

The normalized probabilities of repeat-attempt states are now obtained by dividing the probability $(Pr[i]_{Repeat})_y$ by the total probability of all the elements in the repeat-attempt queue:

$$(Pr[i]_{Repeat})_{y_{norm}} = \frac{(Pr[i]_{Repeat})_y}{\sum_{i=1}^{x} \sum_{j=1}^{y} (Pr[i]_{Repeat})_j} \quad (4)$$

It should be noted that when a departure occurs from a repeat-attempt state, the next state always depends on the probabilities of the number of repeated attempts occurred. Let, μ and μ' be the overall service rate associated with the pull and push system. Also, let μ_0, μ_1, ..., μ_y represents the fraction of overall pull service rate (μ) associated with $0, 1, \ldots, y$ number of repetitions. Now each of this fractional service rate is responsible for servicing the particular item from the pull-queue and the corresponding items repeated in the repeat-attempt queue. Hence, the fractional service rate can be estimated by multiplying the probability of item-selection from the pull queue and from the repeat-attempt queue. Thus, we have:

$$\begin{aligned}
\mu_y &= Pr[i]_{norm}\ (Pr[i]_{Repeat})_{y_{norm}}\,\mu, \\
\mu_0 &= (Pr[i]_{norm}\,[1-\zeta])\,\mu,\ \text{where} \\
\zeta &= (Pr[i]_{Repeat})_{1_{norm}} + \ldots + (Pr[i]_{Repeat})_{y_{norm}} \qquad (5)
\end{aligned}$$

We are now in a position to compute the transitional probabilities in the Markov Chain. The transitional probabilities between any two states are estimated as the ratio of the transition rate between the initial and the final state with the total transition rate from the initial state. Hence, the expression for different transitional probabilities of the Markov Chain is now given as:

$$\begin{aligned}
P_{(x,y,0);(x+1,y,0)} &= \frac{\lambda}{\lambda + x\xi\lambda + \mu'} \\
P_{(x,y,0);(x,y+1,0)} &= \frac{x\xi\lambda}{\lambda + x\xi\lambda + \mu'} \\
P_{(x,y,0);(x,y,1)} &= \frac{\mu'}{\lambda + x\xi\lambda + \mu'} \\
P_{(x,y,1);(x+1,y,1)} &= \frac{\lambda}{\lambda + x\xi\lambda + \sum_{i=0}^{y}\mu_i} \\
P_{(x,y,1);(x,y+1,1)} &= \frac{x\xi\lambda}{\lambda + x\xi\lambda + \sum_{i=0}^{y}\mu_i} \\
P_{(1,y,1);(0,0,0)} &= \frac{\mu_0}{\lambda + x\xi\lambda + \mu_0} \\
P_{(x,y,1);(x-1,y,0)} &= \frac{\mu_0}{\lambda + x\xi\lambda + \mu_0} \\
P_{(x,y,1);(x-1,y-1,0)} &= \frac{\mu_1}{\lambda + xk\xi\lambda + \mu_0} \\
(\forall x \geq 1, \forall y \geq 0) & \\
\ldots\ldots & \quad \ldots\ldots\ldots \\
P_{(x,y,1);(x-1,0,0)} &= \frac{\mu_y}{\lambda + x\xi\lambda + \mu_0} \qquad (6)
\end{aligned}$$

The transitional probabilities of the Markov Chain obtained in this manner now forms the transitional matrix, containing the necessary information of the hybrid system. Any entry corresponding to $(x, y, z), (x', y', z')$ in the transition matrix, actually contains the state transition probability $P_{(x,y,z);(x',y',z')}$ from (x, y, z) to (x', y', z'). Representing all the steady states by the vector $\vec{\pi}$ and the transition matrix by $\mathbf{P}$, an approximate measure of the steady state probabilities can be obtained by solving the following matrix equations associated with the Markov Chain:

$$\begin{aligned}
\vec{\pi} &= \vec{\pi}\mathbf{P} \\
\vec{\pi}\mathbf{e} &= 1, \qquad (7)
\end{aligned}$$

where $\mathbf{e}$ is a unit column vector. Solving the above equations helps us in obtaining the state probabilities $\pi = \{\pi(0,0,0), \ldots, \pi(x,y,z)\}$. The average number of items in the system and the average waiting time is now estimated as:

$$\begin{aligned}
E[Items] &= \sum_{x=0}^{D-K}\sum_{y=0}^{x}[\pi(x,y,0) + \pi(x,y,1)] \\
E[W] &= E[Items]/\lambda. \qquad (8)
\end{aligned}$$

This provides an average behavior of our newly proposed hybrid scheduling system, which considers repeated-attempts from the clients.

V. Simulation Experiments

In this section we validate the performance of our hybrid system through simulation experiments. The primary goal of hybrid scheduling is to reduce the expected access time. Before presenting the details of simulation results, we enumerate the salient assumptions and parameters used in our simulation.

1) The simulation experiments are evaluated for a total number of $D = 1000$ data items.
2) The overall arrival rate λ is varied between 5–20 arrivals per unit time. The value of μ and μ' is estimated as: $\mu = \sum_{i=1}^{K}(P_i \times L_i)$ and $\mu = \sum_{i=K+1}^{D}(P_i \times L_i)$ where P_i and L_i are the access probability and length of data item i, respectively.
3) The length of the data items are varied from 1 to 4.
4) In order to keep the access probabilities of the items from similar to very skewed, θ is dynamically varied from 0.20 to 1.40.
5) To compare the performance of our hybrid scheduling strategy with client's impatience, we have chosen the work in [16], as according to our knowledge, this is the only existing broadcast scheme which considered client's impatience.

In the following, we discuss as series of simulation results to demonstrate the efficiency of our two hybrid scheduling strategies.

Figure 5 demonstrate the variation of the expected access-time with different values of K and θ, for $\lambda = 10$, in our hybrid repeat-attempt scheduling system. With increasing values of cutoff point K, the expected access time initially decreases, attains a minimum value and then starts increasing again. This minimum point also provides the optimum cut-off point for which the framework gets an exact balance between the push and pull systems. Figure 6 shows the results of performance comparison, in terms of expected access time (in seconds), between our newly proposed repeat-attempt hybrid framework with the existing hybrid scheme due to Oh, et al. [16]. The effective combination of PFS and Stretch-optimal scheduling strategies, together with the repeat-attempt functionality results in the reduced waiting time in our hybrid scheduling framework.

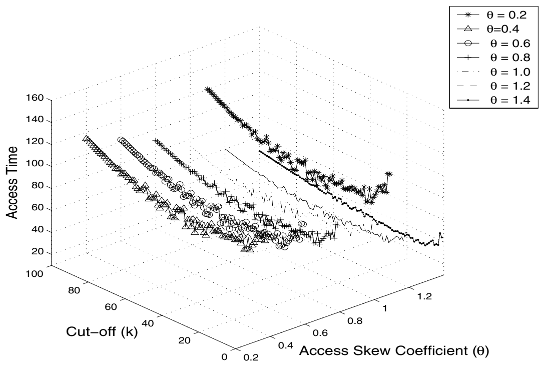

Fig. 5. Performance of Hybrid Scheduling

Figure 7 depicts the comparative view of the analytical results with the simulation results of our repeat-attempt hybrid scheduling framework. For the analytical results, we have numerically solved the Markov Chain in Figure 4 and the Equations 1– 8 to get an estimate of the average system performance. The analytical results closely match with the simulation results for expected access time with almost $\sim 95\%$ accuracy, thereby pointing out that the performance analysis is capable of capturing the average system behavior with good accuracy.

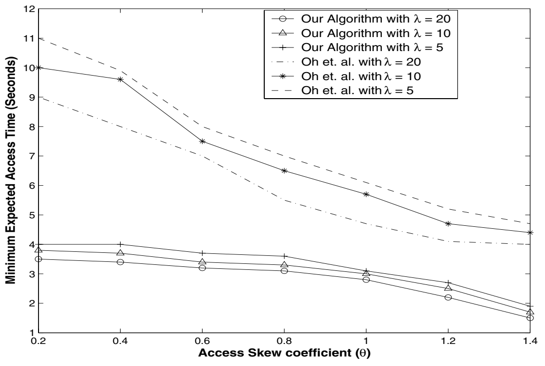

Fig. 6. Performance Comparison with [16]

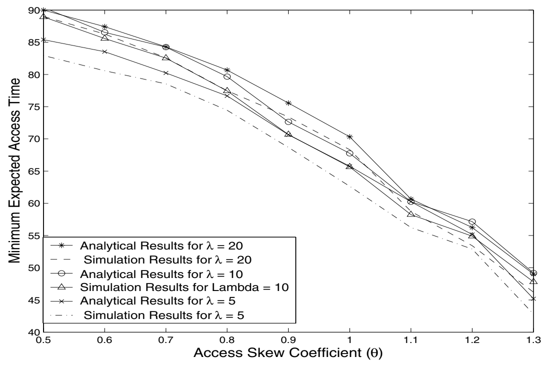

Fig. 7. Simulation Vs Analytical Results

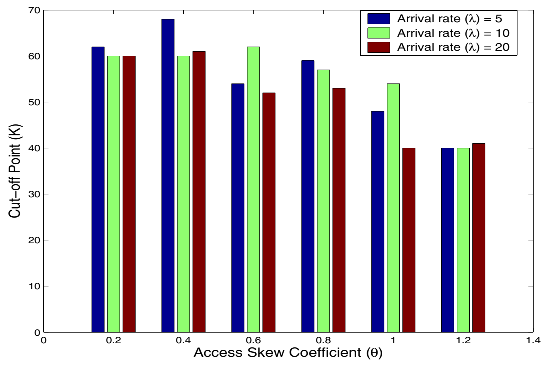

Fig. 8. Variation of Cutoff-point (K)

Figure 8 demonstrates that K lies in the range of 40–60 for three different arrival rates $\lambda = [5, 10, 20]$. Intuitively, this points out that the system has achieved a fair balance between push and pull systems to achieve the minimum expected access time.

VI. Conclusion

In this paper we proposed a new dynamic, hybrid scheduling strategy for heterogeneous environments. To incorporate the item's length (size) into the scheduling discipline, we have used the concept of *stretch optimal or max request min service time first* scheduling. We consider this as a more practical measure, since the clients waiting for lengthy data items should wait longer than those waiting for shorter data items. The scheduling strategy starts as a pure pull system providing on-demand service for every client. When the number of clients' access request rate increases, the algorithm shifts to the hybrid mode. With the dynamics of the system, the system estimates the new access probabilities and adjusts the cutoff-point between push and pull sets in a dynamic fashion, to achieve the minimum waiting time. The major novelty of our work is in modeling the client's impatience. An impatient client often sends multiple requests for the same data item. Using an optimal combination of push and pull scheduling strategies, our proposed algorithm explicitly takes care of the repeated attempts generated by the same client and offers a minimum average waiting time to the set of clients. Extensive analysis and simulation experiments are performed to show the efficiency of the system performance. Our future work will investigate the problem with different classes of clients (different priorities) with energy efficiency issues.

REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communication Environments, *Proceedings of ACM SIGMOD Conf.*, pp. 199-210, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing push and pull for data broadcast. *Proceedings of the ACM SIGMOD Conference*, pp. 183–193, May, 1997.

[3] D. Aksoy and M. Franklin. RxW: A scheduling approach for large scale on-demand data broadcast. *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, pp. 846-860, Dec. 1999.

[4] A. Bar-Noy and Y. Shilo, "Optimal Broadcasting for Two Files over an Asymmetric Channel", *Journal of Parallel and Distributed Computing*, vol. 60, no. 4, pp. 474-493, 2000.

[5] A. Bar-Noy, J. S. Naor and B. Schieber. Pushing Dependent Data in Clients-Providers-Servers Systems. In *Mobile Networks and Applications*, Vol. 9, pages 421-430, 2003.

[6] A. Bar-Noy, B. Patt-Shamir and I. Ziper, "Broadcast Disks with Polynomial Cost Functions", *ACM/Kluwer Wireless Networks*, vol. 10, pp.157-168, 2004.

[7] Z. Brakerski and B. Patt-Shamir, "Jitter Approximation Tradeoff for Periodic Scheduling", *4th IEEE WMAN*, 2004.

[8] Q. Fang, V. Vrbsky, Y. dang and W. Ni, "A Pull-based Broadcast Algorithm that Considers Timing Constraints", *Appearing in Intl. Workshop on Mobile and Wireless Networks*, 2004.

[9] S. Hameed and N. H. Vaidya. Efficient algorithms for scheduling data broadcast In *Wireless Networks*, Vol. 5, pp. 183-193, 1999.

[10] C-L. Hu and M-S Chen, "Adaptive Information Dissemination: An Extended Wireless Data Broadcasting Scheme with Loan-Based Feedback Control, *IEEE Trans. on Mobile Computing*, vol. 2, no. 4, 2003.

[11] S. Jiang and N. H. Vaidya, "Scheduling Data Broadcast to "Impatient" Users, *ACM Intl. Workshop on Mobile Data Engineering*, 1999.

[12] C. Kenyon and N. Schabanel, "The Data Broadcast Problem with Non-Uniform Transmission Times", *Proc. of ACM-SIAM Symp. on Discrete Algorithms*, 1999.

[13] W-C Lee, Q. Hu and D. K. Lee, "A Study on Channel Allocation for Data Dissemination in Mobile Computing Environments", *ACM/Kluwer Mobile Networks and Applications*, vol. 4, pp. 117-129, 1999.

[14] C-W Lin, H. Hu and D-L Lee, "Adaptive Realtime Bandwidth Allocation for Wireless Date Delivery", *ACM/Kluwer Wireless Networks*, (WINET), vol. 10, pp. 103-120, 2004.

[15] W. Ni, Q. Fang, S. V. Vrbsky, "A Lazy Data Request Approach for On-demand Data Broadcasting", *Proc. of* 23^{rd} *Intl. Conf. on Distributed Computing Systems Workshops*, 2003.

[16] JH Oh, K A. Hua amd K. Prabhakara, "A New Broadcasting Technique for An Adaptive Hybrid Data Delivery in Wireless Mobile Network Environment", *Proc. of* 19^{th} *Intl. Performance, Computing and Communications Conference*, pp. 361-367, 2000.

[17] W. C. Peng and M. S. Chen. Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment. In *Proc. of ACM Conf. on Info. and Knowledge Management*, Nov. 2000.

[18] W. C. Peng, J. L. Huang and M. S. Chen. Dynamic Levelling: Adaptive Data Broadcasting in a Mobile Computing Environment. In *Mobile Networks and Applications*, Vol. 8, pages 355-364, 2003.

[19] M. C. Pinotti and N. Saxena. Push less and pull the current highest demanded data item to decrease the waiting time in asymmetric communication environments. *4th International Workshop on Distributed and Mobile Computing, Springer-Verlag, (LNCS)*, (IWDC), pp. 203–213, 2002.

[20] N. Saxena, K. Basu and S. K. Das, "Design and Performance Analysis of a Dynamic Hybrid Scheduling for Asymmetric Environment", *IEEE WMAN*, 2004.

[21] C-J. Su, L. Tassiulas and V. J. Tsotras, "Broadcast Scheduling for Information Distribution", *ACM/Kluwer Wireless networks*, vol. 5, pp. 137-147, 1999.

[22] W. Sun, W. Shi, B. Shi and Y. Yu, "A Cost-Efficient Scheduling Algorithm for On-Demand Broadcasts", *ACM/Kluwer Wireless Networks*, vol. 9, pp.239-247, 2003.

[23] N. Vlajic, C. D. Charalambous and D. Makrakis, "Performance Aspects of Data Broadcast in Wireless Networks With User Retrials", *IEEE/ACM Transactions on Networking*, Vol. 12, No. 4, pp. 620-633, 2004.

[24] Y. Wu and G. Cao. Stretch-Optimal Scheduling for On-Demand Data Broadcasts. In *Proc. of Intl. Conf. on Computer Communications and Networks*, pages 500–504, 2001,.