# A Dynamic Hybrid Scheduling Algorithm with Clients' Departure for Impatient Clients in Heterogeneous Environments

Navrati Saxena, Kalyan Basu and Sajal K Das
Comp. Science & Engg. Dept.
University of Texas at Arlington
Arlington, Texas, USA
{nsaxena, basu, das}@cse.uta.edu

Christina M. Pinotti,
Dept. of Mathematics and Informatics,
University of Perugia,
Perugia, Italy,
pinotti@unipg.it

***Abstract*— The essence of efficient scheduling and data transmission techniques lies in providing the web-applications with advanced data processing capabilities. In this paper we have efficiently combined the push and the pull scheduling to develop a new, practical, dynamic, hybrid scheduling strategy for heterogenous, asymmetric environments. The proposed algorithm dynamically computes the probabilities and the optimal cutoff-point to separate the push and the pull data sets. The data items are also assumed to be of variable lengths. While the push strategy uses the flat, round-robin scheduling, the pull items are determined by stretch-optimal (max-request min-service time) scheduling policy. In order to make the scheduling more practical, we have considered the impact of the impatience of the clients waiting to get the service of a particular data item. The effects of this impatience can lead to departure of specific client(s) from the system. Our proposed hybrid scheduling strategy takes care of these effects to capture a real portrayal of the system dynamics. These scenarios are modelled by suitable birth and death process to analyze the overall expected delay of the system. Subsequently, simulation results corroborate the average system performance and points out significant improvement over existing hybrid systems in terms of average waiting time spent by a client.***

## I. Introduction

The recent advancement and ever increasing growth in web technologies has resulted in the need for efficient scheduling and data transmission strategies. The emergence of wireless communication systems have also added a new dimension to this problem by providing constraints over the low-bandwidth upstream communication channels. Guaranteeing precise quality of service (QoS), such as the expected access-time or delay, is perhaps the most salient feature of such data services. To extract the best performance and efficiency of a data transmission scheme, one needs a scalable data broadcasting technology.

Broadly, data dissemination applications can be categorized into two basic systems: (1) *push-based* and (2) *pull-based*. In a *push-based* broadcast system, the server periodically transmits the data items. The clients continuously monitor this broadcast process and obtain the data items they require, *without making any requests*. On the other hand, in a *pull-based* system, the clients explicitly sends requests, and the server then makes a schedule *on demand* to satisfy these requests. Both push- and pull-based scheduling have their own advantages and disadvantages. While push-based scheduling solves the problem of uplink channel constraints, it might waste valuable bandwidth by repeatedly transmitting less popular (cold) items. Moreover, for large number of data items, the delay associated with the cold items becomes half of the entire broadcast cycle-length. On the contrary, the pull-based scheduling transmits data items on demand, but suffers from uplink resource constraints. Thus, a better performance could only be achieved when the two scheduling approaches are combined in an efficient manner. Although, there exists some works on hybrid scheduling schemes in the literature [1], [8], [5], 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 make a priori assumption of data access probabilities.
- *Client Impatience :* In practical systems, the clients often looses their patience, while waiting for a particular data item. This results in forcing the client to leave the system after waiting for a certain time. Hence, the effects of client's impatience needs to be carefully considered to capture a more accurate, practical behavior 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

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

overall waiting time in heterogenous, asymmetric 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 the chosen from the pull-queue by applying *stretch-optimal (max-request min-service-time first)* scheduling algorithm. 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 stretch-optimal scheduling proposed in [17] works for pure push systems. The $R \times W$ algorithm [3] also considers data items with uniform size for on demand, pull-based systems.
2) The *cut-off point* that separates the push and pull sets is computed dynamically to better analyze the system behavior and more practical scenarios. This has the power to achieve optimal performance.
3) In most practical systems, the clients often get impatient while waiting for the designated data item. After a tolerance limit, the client may depart from the system, thereby resulting in a drop of access requests. This behavior significantly affects the system performance, which needs to be properly addressed. Although an introduction of impatience is investigated in [10], the work considers only pure push scheduling. One major contribution of our work lies in minimizing the overall drop request as well as the expected waiting time.
4) Simulation experiments are conducted starting with a pure pull system, without prior knowledge of data access probabilities. Later, the probabilities are re-computed dynamically to determine the cut-off-point between the push and pull sets. Simulation results closely match with analytical results.

The rest of the paper is organized as follows: Section II reviews related works. The hybrid scheduling strategy is proposed in Section III. To analyze its performance, a queuing model is developed in Section IV. Simulation results in Section V corroborates the performance analysis and points out the efficiency of the hybrid scheduling strategy. Finally, Section VI concludes the paper with pointers to future researches.

## 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 [8]. However, the optimal solution for data broadcast problems with non-uniform transmission time and multiple data items [11] are shown to be NP-hard, thus leading to the development of various approximation techniques. A three-player *client-provider-server* model [4], 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 [14], [9]. Asymptotically optimal algorithms for obtaining data access-costs, which are not proportional to their own waiting time, are provided in [5]. 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 [10] is the only work that considered this effect for push-based 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 [16] is proposed as a more appropriate performance metric. The pull-based real-time data dissemination system, discussed in [6], proposes Aggregated Critical Requests (ACR) scheduling algorithm to meet the predetermined timing constraints.

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. In [2], 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. The adaptive real-time data transmission strategy in [12] 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 [13], 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. Our previous work on hybrid scheduling [15] 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. Dynamic Hybrid Scheduling

In this section we first provide the basic definitions and preliminary concepts, and then describe our new hybrid scheduling strategies with practical considerations.

### A. Definitions and Assumptions

We assume that the server has a total number of $D$ data items of variable sizes, stored in the memory, so that they could be retrieved immediately by the server. The length of an item $i$ is denoted by $L_i$. Each item is again divided into equal-sized pages, where each page takes 1 unit of time to be broadcasted/disseminated in the air. The cutoff point $K$ divides the entire set of $D$ items into the push and pull set such that $K$ items are pushed and remaining $D-K$ are pulled.

There are a certain number of clients in the system and different items can be accessed or requested by different clients. Depending on clients' requests, a particular data item is associated with certain *degree of popularity* or *access probability*. Access probability for an item $i$ is denoted as $\varrho_i$. We have assumed that the access probabilities obey the Zipf's distribution with *skew coefficient* $\theta$ such that $\varrho_i = \frac{(1/i)^\theta}{\sum_{j=1}^{D}(1/j)^\theta}$. For small values of $\theta$, the access probabilities $\varrho_i$, for $1 \leq i \leq D$ are well balanced but for increasing values of $\theta$, they become skewed. Data items are arranged in decreasing order of their access probabilities, i.e., $\varrho_1 \geq \varrho_2 \geq \varrho_3 \geq ... \geq \varrho_D$.

### B. Hybrid Scheduling Algorithm

As highlighted before, one major novelty of our strategy lies in its consideration for clients' impatience. The impatience of a client results in a departure from the system. This strategy is termed as *Hybrid Scheduling with Clients Departure*. In general, the system begins with operating as a pure pull system providing on-demand service for every client. When the number of client's access request rate increases and broadcasting the same item to different clients causes downstream bandwidth wastage, the algorithm shifts to the hybrid mode. The items are now divided into two disjoint sets: the push set of cardinality $K$ and the pull set of cardinality $D-K$. The items to be pushed are governed by *flat* round-robin scheduling. On the other hand, the item which maximizes stretch (max-request min-service time) is selected to be pulled by the server. Every push is followed by a pull, provided that the pull-set (queue) is not empty. If there are no items in the pull queue, then the server simply continues pushing the items using flat schedule. However, after transmitting each page the server attempts $\lambda$ more access requests arriving into the system. If the request is for a push item, the server simply ignores the request as the item would be pushed anyway according to the broadcast schedule. If the request is for a pull item, then the server first checks whether the request is for a new item or an already requested item. If it is for a new item, the item is inserted into the pull queue and its stretch value is calculated. Next, the server checks for the client's impatience and tolerance.

**Hybrid Scheduling with Impatience:**
If the request is for an existing item, the server checks whether one or more clients are getting impatient and loosing there tolerance limit. Anticipating departures of such clients, the server drops their requests and stores their previous waiting time (departure time − arrival time). It then updates the stretch value of the data items in the pull queue considering only the request of existing clients which are not impatient. A pseudo-code of the strategy is depicted in Figure 1. The procedure *Take-Access-with-Drop()* considers the set of arriving requests, process them and insert in the pull queue, after considering the number of requests dropped due to the client's departure. A pseudo-code of this procedure is shown in Figure 2.

```
HYBRID SCHEDULING with IMPATIENCE;
while true do
begin
Broadcast all the pages of an item, selected
according to the flat scheduling;
After broadcasting each page
Take-Access-with Drop();
if  the pull-queue is not empty then
   extract an item from the pull-queue
   that optimizes the stretch;
   clear the number of pending requests for
   that item and pull it;
Take-Access-with-Drop() /*procedure call */
end;
```

Fig. 1. Hybrid Scheduling with Client's Departure

```
Procedure: Take-Access-with-Drop();
Take λ more accesses;
if the request is for push items then
   ignore the requests;
if the request is for pull items then
   Compute number of impatient clients
   leaving and remaining;
   insert request in pull queue (with arrival time);
   update its stretch value based on
   number of remaining clients;
```

Fig. 2. Take Access with Drop-requests

## IV. Performance Modelling

In this section we analyze the performance of our hybrid scheduling algorithm. Before proceeding further, let us enumerate the parameters and assumptions used.

### A. Assumptions

1) The arrival rate in the entire system is assumed to obey the Poisson distribution with mean $\lambda'$. This includes the

arrival rate in both push and pull systems. Although the arrival rate of the push system is assumed fixed, the departure of impatient clients and/or their spurious requests changes the arrival rate of the pull system at every step. The initial arrival rate of the pull system is assumed to be $\lambda$. The pull queue contains data items which are yet to be served. Thus by the term pull system, we mean the items waiting in pull queue, together with the item(s) currently getting service.

2) The service times of both the push and pull systems are exponentially distributed. Again, the mean service time of push system is fixed, however, the clients' impatience changes the service time of the pull system. We represent the initial service time of pull system by $\mu_2$.
3) Let $C$, $D$ and $K$ respectively represent the maximum number of clients, total number of distinct data items, and the cut-off point. The server pushes $K$ items while clients pull the remaining $(D-K)$ items. Thus, the total probability of items in the push- and pull sets are respectively given by $\sum_{i=1}^{K} \varrho_i$ and $\sum_{i=K+1}^{D} \varrho_i = (1 - \sum_{i=1}^{K} \varrho_i)$, where $\varrho_i$ denotes the access probability of item $i$. Basically, it gives a probabilistic measure of item's popularity among the clients. We have assumed that the access probabilities follow the *Zipf's distribution* with *access skew-coefficient* $\theta$, such that $\varrho_i = \frac{(1/i)^\theta}{\sum_{j=1}^{n}(1/j)^\theta}$. Items are numbered from 1 to $D$ and are arranged in the decreasing order of their access probabilities, i.e., $\varrho_1 \geq \varrho_2 \geq ... \geq \varrho_D$. Table I lists the symbols with their meaning used in the context of our analysis.

Let us now analyze the system performance for achieving the minimal waiting time.

TABLE I
SYMBOLS USED FOR PERFORMANCE ANALYSIS

| *Symbols* | Meanings |
|---|---|
| $D$ | Maximum number of data items |
| $C$ | Maximum number of clients |
| $i$ | Candidate data item |
| $K$ | Cut-Off Point separating push and pull sets |
| $P_i$ | Access Probability of item $i$ |
| $L_i$ | Length of item $i$ |
| $\lambda'$ | Overall System Arrival Rate |
| $\lambda$ | Initial Arrival Rate in pull queue |
| $\mu_1$ | Push Queue Service Rate |
| $\mu_2$ | Initial Service Rate in Pull Queue |
| $E[W_{pull}]$ | Expected Waiting Time of Pull System |
| $E[W^q_{pull}]$ | Expected Waiting Time of Pull Queue |
| $E[\mathcal{L}_{pull}]$ | Expected Number of items in the Pull system |
| $E[\mathcal{L}^q_{pull}]$ | Expected Number of items in the Pull queue |

### B. Effect of Client's Impatience

Here we assume that a client's impatience results in its departure from the system before the item is actually serviced. This impatience generally takes two forms [7]: (1) The reluctance of the customer to remain in the system is known as *reneging*; (2) Excessive reluctance might restrain the customer to even join the system, which is known as *balking*. These two behaviors significantly affect the arrival/service rate and average system performance. In our analysis, we have assumed the duration of the waiting time of a client (before leaving) to follow exponential distribution with mean $1/\tau$. If $\bar{\lambda}_m$ represents the request arrival rate for $m^{th}$ data item, then $\bar{\lambda}_m = \varrho_m \lambda$, where $\lambda$ is the initial request arrival rate of the entire pull system. If the request arrives at time $t$ and does not depart the system before servicing the $m^{th}$ data at time $\Gamma$, then expected number of requests, $E[R_i]$, satisfied by transmission of $m^{th}$ item is given by:

$$\begin{aligned} E[R_m] &= \int_0^{\Gamma} \bar{\lambda}_m e^{-\tau(\Gamma - t)} dt \\ &= \frac{\varrho_m \lambda}{\tau}(1 - e^{-\tau\Gamma}) \end{aligned} \quad (1)$$

Also, for Poisson arrival, the expected number of requests arriving in time period $\Gamma$ is given by $\lambda\Gamma$. Thus, the expected number of drop requests, $E[R_d]$, is measured as:

$$\begin{aligned} E[R_d] &= \lambda\Gamma - E[R_m] \\ &= \lambda\Gamma - \frac{\varrho_m \lambda}{\tau}(1 - e^{-\tau\Gamma}) \end{aligned} \quad (2)$$

Our next objective is to estimate the expected waiting time of our hybrid system considering the clients balking and reneging due to client's impatience.

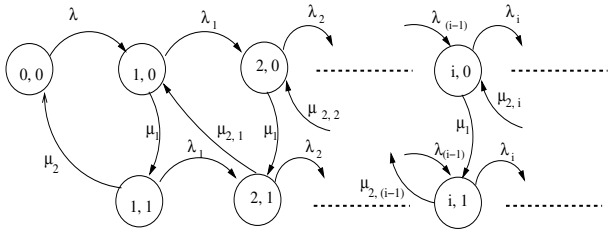


Fig. 3. Performance Modelling of the System

Figure 3 illustrates the birth and death model of our system. For any variable $i$, the $i^{th}$ state of the overall system is represented by the tuple $(i, j)$, where $i$ represents the number of items in the pull-system and $j = 0$ (or 1) respectively represents whether the push-system (or pull-system) is being served. The arrival of a data item in the pull-system results in the transition from state $(i, j)$ to state $(i+1, j), \forall i \in [0, \infty]$ and $\forall j \in [0, 1]$. The service of an item results in transition of the system from state $(i, j = 0)$ to state $(i, j = 1), \forall i \in [0, \infty]$. On the other hand, the service of an item in the pull results in transition of the system from state $(i, j = 1)$ to the state $(i - 1, j = 0), \forall i \in [1, \infty]$. Note that, the arrival and service rates in the pull system are both different at each state. Naturally, the state of the system at $(i = 0, j = 0)$ represents that the pull-queue is empty and any subsequent service of the elements of push system leaves the system in the same $(0, 0)$ state. Obviously, state $(i = 0, j = 1)$ is not valid because the service of an empty pull-queue is not possible. The arrival rates at different states are now represented by $\lambda_0, \lambda_1, \ldots, \lambda_i, \ldots$, where $\lambda_0 = \lambda$. Furthermore, $\lambda_i$ is different from $\bar{\lambda}_m$ discussed before. While $\bar{\lambda}_m$ represents the request arrival rate for $m^{th}$

data item, $\lambda_i$ denotes the total arrival rate of requests for all $i$ items present in the system, i.e., $\lambda_i = \sum_{m=0}^{i} \lambda_m^-$. Similarly, the service rates at different states are denoted by $\mu_{2,j}$ where $1 \leq j \leq n$ and $\mu_{2,1} = \mu_2$.

In the steady-state, using the flow-balance conditions of *Chapman-Kolmogrov's* equation [7], the initial system-behavior is represented by:

$$p(0,0)\ \lambda = p(1,1)\ \mu_2 \tag{3}$$

where $p(i,j)$ represents the probability of state $(i,j)$. The overall behavior of the system for push (upper chain in Figure 3) and the pull system (lower chain) are given by the following two generalized equations:

$$p(i,0)(\lambda_i + \mu_1) = p(i-1,0)\lambda_{i-1} + p(i+1,1)\mu_{2,i+1} \tag{4}$$

$$p(i,1)(\lambda_i + \mu_{2,i}) = p(i,0)\mu_1 + p(i-1,1)\lambda_{i-1} \tag{5}$$

Balking [7] is generally estimated by using a series of monotonically decreasing functions of the system size multiplying by the initial arrival rate, $\lambda$. If $b_i$ is the balking function at $i^{th}$ state, then $\lambda_i = b_i\lambda$, where $0 \leq b_{i+1} \leq b_i \leq 1, (\forall i > 0,\ b_0 = 1)$. The most practical discouragement (balking) function is $b_i = e^{-i\alpha}$, where $\alpha$ is a constant. This takes the queue size into account and discourages the customers from joining in large-sized queues. However, in practical systems, the discouragement does not always arrive from excessive queue sizes. These customers might instead join the system and continuously retain the prerogative to *renege* if the waiting time is intolerable. This reneging function $r(i)$ [7] at $i^{th}$ state is defined by:

$$r(i) = \lim_{\Delta t \to 0} \frac{Pr[\text{unit reneges during } \Delta t]}{\Delta t} \tag{6}$$

The service rate of pull queue now takes the form: $\mu_2 = \mu_2 + r(i)$. A good possibility of the reneging function is: $r(i) = e^{i\alpha/\mu_2}$. Note that both balking and reneging functions are assumed to follow exponential distribution, which is in accordance with the distribution obeyed by request's waiting time.

From Equations (4) and (5) we get,

$$\begin{aligned} p(i,0)(e^{-\alpha i}\lambda + \mu_1) &= p(i-1,0)\lambda e^{-\alpha(i-1)} + \\ &\quad p(i+1,1)\mu_2 + e^{(i+1)\frac{\alpha}{\mu_2}} \end{aligned} \tag{7}$$

$$\begin{aligned} p(i,1)\lambda e^{-\alpha i} + p(i,1)\mu_2 &+ p(i,1)e^{\alpha\frac{i}{\mu_2}} = p(i,0)\mu_1 \\ &+ p(i-1,1)e^{-\alpha(i-1)} \end{aligned} \tag{8}$$

The most efficient way to solve of Equations (7) and (8) is using *Z-transforms* [7]. From the definition of $z$-transforms, the resulting solutions are of the form:

$$P_1(z) = \sum_{i=0}^{C} p\,(i,0)\,z^i \ \text{ and } \ P_2(z) = \sum_{i=0}^{C} p(i,1)\,z^i. \tag{9}$$

Using subsequent Z-transforms, Equation (7) yields:

$$\begin{aligned} &\lambda\left[P_1\left(\frac{z}{e^\alpha}\right) - p(0,0)\right] + \mu_1\left[P_1(z) - p(0,0)\right] \\ &= \lambda z\left[P_1\left(\frac{z}{e^\alpha}\right)\right] + \frac{1}{z}\left[P_2(z) - p(0,1) - p(1,1)\right] \\ &+\frac{1}{z}\left[P_2\left(ze^{\frac{\alpha}{\mu_2}}\right) - p(0,1) - p(1,1)\right] \end{aligned} \tag{10}$$

Similarly, transforming Equation (8) leads to:

$$\begin{aligned} \lambda P_2\left(\frac{z}{e^\alpha}\right) + P_2\left(ze^{\frac{\alpha}{\mu_2}}\right) &= \mu_1 P_1(z) - p(0,0) \\ &\quad + zP_2\left(\frac{z}{e^\alpha}\right) \end{aligned} \tag{11}$$

Now, putting $z = 1$ in Equation (10), we can obtain the probability $p(0,0)$ of the idle state as:

$$\begin{aligned} &\lambda\left[P_1\left(\frac{1}{e^\alpha}\right) - p(0,0)\right] + \mu_1[P_1(1) - p(0,0)] = \\ &\lambda\left[P_1\left(\frac{1}{e^\alpha}\right)\right] + \mu_2[P_2(1) - p(1,1)] + P_2\left(e^{\frac{\alpha}{\mu_2}}\right) - p(1,1) \\ &p(0,0) = \frac{\mu_2\rho - \mu_1(1-\rho) + \frac{\rho}{1-e^{\frac{\lambda}{\mu_2}}}}{\frac{\lambda}{\mu_2} - \mu_1} \end{aligned} \tag{12}$$

Deriving closed form solutions of Equations (10) and (11) to evaluate the state probabilities seems not possible. Instead we measure the expected performance of the overall system. In order to estimate the average number of items in the pull system, Equation (10) is differentiated (at $z = 1$). Now, the occupancy of push and pull states are respectively given by $P_1(1) = \sum_{i=0}^{\infty} p(i,0) = 1-\rho$ and $P_2(1) = \sum_{i=0}^{\infty} p(i,1) = \rho$, where $\rho = \frac{\lambda_{eff}}{\mu_{eff}} = \frac{\sum_{i=0}^{\infty}\lambda_i p(i,1)}{\sum_{i=1}^{\infty}\mu_{2,i}p(i,1)}$. Differentiating Equation (10) and using these values of $P_1(1)$ and $P_2(1)$, we get

$$\begin{aligned} &\mu_2\frac{dP_2(z)}{dz} + \frac{dP_2}{dz}\left(ze^{\frac{\alpha}{\mu_2}}\right) = \mu_1 P_1(1) + \mu_1\frac{dP_1}{dz} \\ &-(\lambda+\mu_1)\frac{\mu_1\rho - \mu_1(1-\rho) + \frac{1}{1-e^{\frac{\alpha}{\mu_2}}}}{\lambda/\mu_2 - \mu_1} - \\ &\lambda P_1(1/e^\alpha) - \lambda\frac{dP_1}{dz}(\frac{1}{e^\alpha}) + 2\lambda P_1(1/e^\alpha) + \lambda P_1(1/e^\alpha) \\ &E[\mathcal{L}_{pull}] = \frac{dP_2(z)}{dz}|_{z=1} = \left(\mu_1 + \frac{1}{1-e^{\frac{\alpha}{\mu_2}}}\right)^{-1} \\ &\left[\mu_1\rho + \mu_1 E[\mathcal{L}_{push}] - (\mu_1+\lambda)\frac{\mu_1\rho - \mu_1(1-\rho) + \frac{\rho}{1-e^{\frac{\alpha}{\mu_2}}}}{\frac{\lambda}{\mu_2} - \mu_1}\right] \\ &+\lambda E[\mathcal{L}_{push}]e^{\alpha/mu_2}, (\text{where } E[\mathcal{L}_{push}] = \frac{dP_1(z)}{dz}|_{z=1}) \end{aligned} \tag{13}$$

Once we have the expected number of items in the pull system from Equation (13), using Little's formula [7], we can easily estimate the average waiting time of the system ($E[W_{pull}]$), average waiting time of the pull queue ($E[W^q_{pull}]$) and expected number of items ($E[\mathcal{L}^q_{pull}]$) in the pull queue as follows:

$E[W_{pull}] = \frac{E[\mathcal{L}_{pull}]}{\lambda}$, $E[\mathcal{L}^q_{pull}] = E[\mathcal{L}_{pull}] - \frac{\lambda}{\mu_2}$ and $E[W^q_{pull}] = E[W_{pull}] - \frac{1}{\mu_2}$.

Since the push system is governed by flat scheduling, the average cycle time of the push system is given by: $\frac{K}{2(1-\rho)\mu_1} \sum_{i=1^K} \varrho_i$. Thus, the overall minimum expected access-time, ($E[T_{hyb-acc}]$, of our hybrid system is:

$$E[T_{hyb-acc}] = \frac{K}{2(1-\rho)\mu_1} \sum_{i=1}^{K} \varrho_i + E[W_{pull}] \sum_{i=K+1}^{D} \varrho_i \qquad (14)$$

This gives a suitable measure of the performance of our hybrid, heterogeneous system when the clients get impatient and leave the system at certain intervals.

## V. Simulation Experiments

In this section we validate the performance of our hybrid system through simulation experiments The primary objective of the hybrid scheduling with client's departure considers reducing the service drop, apart from minimizing 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 $\lambda'$ is varied between 1–4 arrivals per unit time. The value of $\mu_1$ and $\mu_2$ is estimated as: $\mu_1 = \sum_{i=1}^{K}(P_i \times L_i)$ and $\mu_2 = \sum_{i=K+1}^{D}(P_i \times L_i)$ where $P_i$ and $L_i$ are the access probability and length of data item $i$, respectively.
3) The length of data items are varied from 1 to 4.
4) In order to keep the access probabilities of the items from similar to very skewed, $\theta$ 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 [10], 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.

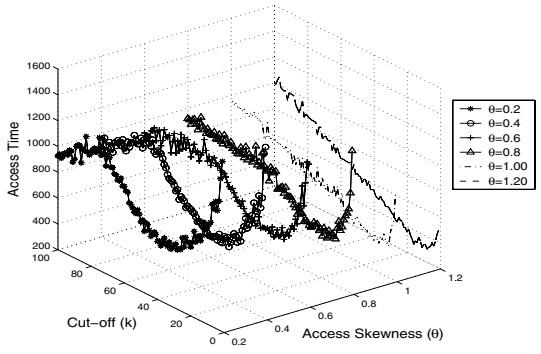


Fig. 4. Expected Access Time with Cutoff Point

Figure 4 demonstrates the variation of expected access time with cutoff-points ($K$) for different values of access skewness, $\theta$. For all values of $\theta$, with increasing $K$ the expected access time initially decreases up to a certain point and then increases again. The reason is that with lower values of $K$, the access time for push items are pretty low while those for pull items are very high. The scenario gets reversed when the value of $K$ is pretty high. The curve for the expected access time takes a bell-shaped form, with the minimum value obtaining for certain cutoff-point, termed as optimal cutoff.

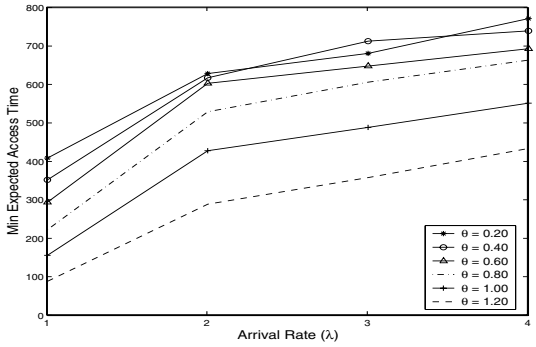


Fig. 5. Minimum Expected Access Time

The different arrival rates of data items have significant impact on the minimum expected access time achieved by the system. Figure 5 shows that for different access skewness and with increasing arrival rates, the expected access time increases. For an arrival rate of 1 and 4, the average access time is in the range 100–400 and 400–750 time units respectively.

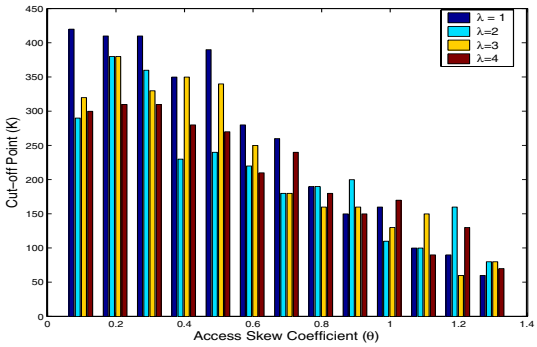


Fig. 6. Variation of Cutoff Point

Next we analyze the variation of the cutoff point with access skewness for different arrival rates. This is necessary to get a clear picture of the system dynamics, as the cutoff point plays the major role to minimize the expected access time. Figure 6 shows that the value of cutoff point decreases with increasing values of access skewness, $\theta$. For example, $K =$ 300–500 for lower skewness ($\theta \leq 0.6$) and $K =$ 100–150 for higher skewness ($\theta \geq 1.00$). The reason is that with increasing skewness, the items get more skewed and number of popular items decreases. Hence, fewer number of items are pushed, thus decreasing the cutoff point.

One major objective of our proposed hybrid scheduling is to reduce the dropped requests arising from client's impatience.

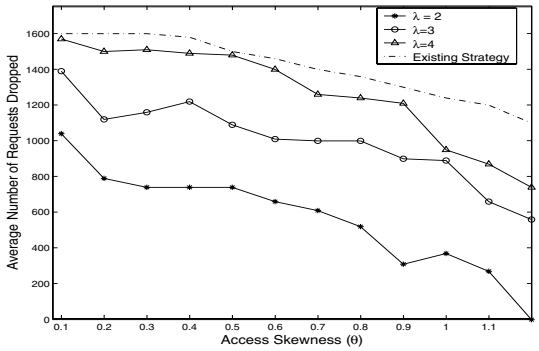


Fig. 7. Average Number of Requests Dropped

Figure 7 depicts the average number of requests dropped with access skewness for different arrival rates. The performance is compared with the existing strategy [10] for client's impatience in data broadcasting with an unit arrival rate. As expected, the number of drop-requests increases with increasing arrival rates. However, for all arrival rates the number of drop requests is significantly lower than the number of drop-requests in existing work. This is true even for higher arrival rates $\lambda' \geq 2$. This points out the efficiency of our hybrid scheduling strategy while considering client's departure due to impatience.

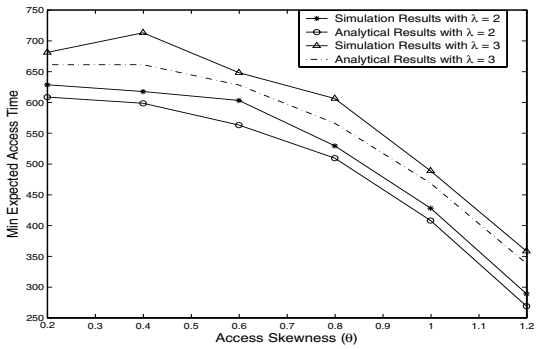


Fig. 8. Analytical Vs. Simulation Results

Figure 8 provides the comparative view between analytical and simulation results for hybrid scheduling with client's departure. The simulation results closely match with the analytical results. The minor $\sim 8\%$ difference is primarily due to the fact that analytical results only capture an approximate average value.

## VI. Conclusion

In this paper we have proposed a dynamic hybrid scheduling which simultaneously considers items of variable lengths, dynamic computation of access probabilities and clients' impatience. 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. 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. Another novelty of our work is in modelling the client's impatience. An impatient client can either leave the system prior to get serviced. In this case, the requests for the clients are dropped, which needs to be considered (minimized). The algorithms reduce the client's drop-request and minimizes the waiting time.

## References

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik. Braodcast Disks: Data Management for Asymmetric Communication Environments. *Proc. of ACM SIGMOD* , San Jose, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing Push and Pull for Data Broadcast. *Proceedings of 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, Dec, 1999.

[4] 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.

[5] 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.

[6] Q. Fang, V. Vrbsky, Y. Dang and W. Ni, "A Pull-based Broadcast Algorithm that Considers Timing Constraints", *Intl. Workshop On Mobile And Wireless Networking*", 2004.

[7] D. Gross and C. M. Harris, Fundamentals of Queuing Theory *John Wiley & Sons Inc.*

[8] S. Hameed and N. H. Vaidya. Efficient Algorithms for Scheduling Data Broadcast *Wireless Networks*, Vol. 5, pp. 183-193, 1999.

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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,.