A Dynamic Hybrid Scheduling Algorithm for Heterogeneous Asymmetric Environments *

Navrati Saxena
Dept. of Informatics and Telecom.
University of Trento
Povo, Trento, Italy
navrati@dit.unitn.it

Cristina M. Pinotti
Dept. of Mathematics and Comp. Science
University of Perugia
Perugia, Italy
pinotti@unipg.it

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

Abstract

The rapid growth of web services has already given birth to a set of data dissemination applications. Efficient scheduling techniques are necessary to endow such applications with advanced data processing capability. In this paper we have developed a new hybrid scheduling scheme by effectively combining broadcasting of very popular (push) data and dissemination of less popular (pull) data to develop a new, ideal hybrid scheduling scheme. Our algorithm assumes no prior knowledge of the data access probability, i.e., which items are being pushed or pulled. Instead, data access probabilities and the cut-off-point to segregate the push and the pull sets are computed dynamically. The data items are assumed to be of variable lengths. The clients send their requests to the server, which ignores the requests for the push items but queues those for the pull items. An ideal behavior of the clients is assumed: a client, after making a request, patiently waits without taking any action. Every client is assumed to have a priority. The priority of a data item is determined by adding the priorities of all the clients requesting that item. At every instant, the item to be broadcast is selected with the help of a pure-push scheduling. On the other hand, the item to be pulled is the one stored in the pull-queue, having the optimal stretch value (max-request min-service time). When more than one data item have the same stretch value, the influence of the priorities of different clients on data dissemination is considered. A suitable modelling technique using the birth-and-death process is developed to analyze the performance of the system. Simulation results corroborate the average system performance and exhibit significant improvement over a pure push and existing hybrid systems in terms of average waiting time spent by a client.

1 Introduction

Over the past few years, the overwhelming growth of web services has resulted in the emergence of applications capable of storing and processing a huge set of information. The prime objective of such *data dissemination* applications lies in delivering the data to a very large population of clients with minimum delay. In such dissemination-based systems, there often exists asymmetry due to any of the following factors:

- The downstream communication capacity (bandwidth from server to client) may be much higher than the upstream communication capacity (bandwidth from client to server);
- The number of clients is significantly larger than the number of servers.

*This work was done when Navrati Saxena was a visiting researcher at CReWMaN Lab, CSE Dept., U.T. Arlington, USA

- In information retrieval applications, the clients make requests to the server through small request messages that results in the transmission of much larger data items. In other words, asymmetry remains in the size and amount of messages in uplink and downlink transmission.

Broadly, all data dissemination applications have two flavors. In a *push-based system*, the clients continuously monitor a broadcast process from the server and obtain the data items they require, *without making any requests*. In contrast, in a *pull-based* system, the clients initiate the data transfer by sending requests *on demand*, which the server schedules to satisfy. Both push- and pull-based scheduling have their own advantages and disadvantages. A detailed overview of the published research works on wireless data broadcast can be found in [31]. However, neither push nor pull alone can achieve optimal performance [22]. Therefore, the search for efficient hybrid scheduling, which explores the efficiency of both push and pull strategies, continues. Example of hybrid push-pull systems include the Huges Network System DirecPC Architecture [14], that uses satellite technology to give a fast, always-on Internet connection; the Cell Broadcast Service (CBS) that enables to deliver short messages to all the users in a given cell in both GSM and UMTS systems [27]; and the Service Discovery Service in networks of pervasive devices [7].

In this paper, we propose an ideal hybrid scheduling that effectively combines broadcasting of more popular (i.e., push) data and dissemination upon-request for less popular (i.e., pull data) in *asymmetric* (where asymmetry is arising for difference in number of clients and servers), *heterogeneous* (different items have different lengths) environments. In this approach, the server continuously broadcasts one push item and disseminates one pull item. We have assumed an ideal system where the clients sends their requests to the server and waits for the necessary data item until they receive it. Hence, the client's impatience and departure from the system which might result from its prolonged wait, is assumed to be negligible.

The server queues the clients' requests up for the pull items. At any instant of time, the item to be broadcast is selected by applying a pure-push scheduling. On the other hand the item to be pulled is the one selected from the pull-queue using *stretch optimal* (i.e, *max-request min-service-time first*) scheduling principle. We argue that stretch is a more practical and better measure in heterogeneous system, where items have variable lengths and the difference in item-lengths results in the difference in service time of data items. Hence, apart from the client-requests accumulated, the system also needs to consider the service time of the items as items of larger size should wait longer than items of shorter length. This motivates us to use *stretch* (max-request min-service-time first) as the performance metric. The performance of our hybrid scheduler is analyzed to derive the expected waiting time. The cut-off point between push and pull items is chosen so as to minimize the overall waiting time of the system. A preliminary version of this work has

been presented in [24].

The rest of the paper is organized as follows. Section 2 reviews related work and develops the motivations behind our new hybrid algorithm presented in Section 3. To analyze the performance of the hybrid system, a queuing model is developed in Section 4. Simulation results in Section 5 corroborates the system behavior. Section 6 concludes the paper.

2 Related Work and Motivations

Wireless communications environments are characterized as asymmetric because the downlink capacity or bandwidth is much greater than the uplink capacity. Broadcast is an efficient and scalable way of taking advantage of the large downlink capacity to disseminate data to an arbitrary number of clients. In fact, the broadcast employs a server in a base-station, which continuously transmits data items from a given set over a wireless channel, in order to reach clients that passively listen to the shared channel waiting for their desired item. The server follows a broadcast schedule to decide upon which data item to be transmitted at any time instant. From now on, the problem of finding an efficient broadcast schedule, which minimizes the clients' expected delay (i.e., the average amount of time spent by a client before receiving the item) will be termed as the *Broadcast Problem*.

A broadcast time unit is generally defined as the time taken to transmit a unit length item. Several solutions for the Broadcast Problem have been proposed in literature. The most basic strategy is the *flat* schedule in which the transmission of data items is performed one at a time in a round-robin fashion, and the expected delay is the same for all data items. More precisely, on average, this delay is half of the sum of lengths all the items in the data set. Naturally, this flat schedule becomes impractical for a large set. Researches have, then, revealed that for an efficient scheduling strategy, every data item needs to be characterized by its access probability, i.e., the measure of popularity of an item amongst the clients [1, 2]. Indeed, these access probabilities guide the selection of the item to be broadcast. A preliminary solution for the Broadcast Problem was studied in the context of teletext systems by [4]. Intuitively, a *hot* item (whose access probability is close to 1) is expected to be selected in the broadcast schedule more frequently than a *cold* item (whose access probability is close to 0). The same problem is also known as the *Broadcast Disk Problem*, first studied in [1]. The solution of the Broadcast Disk Problem groups data items in disks, thus assigning items in the same range of access probabilities to the same disk. The broadcast schedule is then generated by interleaving one item from each disk. The disks with smaller size rotate faster and contains items having higher access probabilities. These helps the smaller disks to provide more instances of their

data to the broadcast schedule. In literature, the most common solution for the Broadcast Problem is based on the *square root rule (SRR)* [13, 4, 5, 16]. SRR produces a broadcast schedule where each data item appears with equally spaced replicas, and each data item has a frequency proportional to the square root of its access probability and inversely proportional to the square root of its length. Although the Broadcast Problem has been widely studied and modelled as a special case of the Maintenance Scheduling Problem and Multi-Item Replenishment Problem, its tractability is still open for the uniform case where all items are of the same length [6, 18]. On the other hand, the non-uniform case (i.e., items of variable length) of the problem has been shown to be strong NP-hard [5, 16]. Hence different approaches using non-linear programming, greedy heuristics and golden-ratio-based approximation algorithms [5] have been proposed to obtain near-optimal bounds on the scheduling performance. A set of randomized and approximation algorithms are also proposed to extend this problem into a more realistic setting of non-uniform message transmission times [16]. Indeed, broadcasting multiple files over the web environment often needs to consider the inherent dependencies among the files. The approach proposed in [19] is focused on the major problems and issues in accessing multiple data items over a broadcast channel.

Moreover, it has been observed that 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. An introduction of impatience is investigated in [15, 29] for push-based scheduling. The work in [15] is focused on modelling the user impatience. Precisely, the client can make a request (a request arrives in the system) and the client can leave the system before being served (a request leaves the system). Assuming an exponential distribution of the time that a client waits before dropping a request, a broadcast schedule is devised in [15] with high service ratio and low mean waiting time for requests.

Similarly, for the pull-based scheduling, many preemptive and non-preemptive algorithms like First Come First Served (FCFS), Most Requests First (MRF) and Request-Waiting-time ($R \times W$), exist in the literature [23, 3]. While the performance of the on-demand pull systems are often estimated by their response time, recently the average of access time cost, tuning time cost and cost of handling failure have been proposed as more appropriate performance metric [26]. The pull-based real-time data dissemination 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 [25] to compare the performance of push and pull based systems.

Modelling the push-based system as a deterministic Markov Decision Process (MDP), near-optimal

scheduling policies are identified. Similarly, the pull-system is separately modelled as a stochastic MDP and performance comparison points out that in certain cases the performance of push and pull systems are pretty close. Then, a hybrid approach that use the flavors of both push-based and the pull-based scheduling algorithms in one system, appears to be more attractive. At the best of our knowledge, the first hybrid technique for scheduling and data transmission in asymmetric environment is proposed in [2]. In [2], the server pushes all the data items according to some push-based scheduling, but simultaneously the clients are provided with a limited backchannel capacity to make requests for the items. Such an introduction of pull-based transmission into the existing push-based systems significantly reduces the overall expected waiting time, and increases the system's performance efficiency. Dynamic channel allocation strategies for broadcast push and on-demand pull systems are investigated in [17]. The adaptive real-time data transmission strategy in [9, 20] combines broadcast push and on-demand pull strategies to achieve a near-optimal system response time. In order to exploit the advantages of both the push and the pull based scheduling, a lazy data request approach is recently proposed [21]. Instead of transmitting the request for data items, 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. Apart from reducing the overall access time, the scheme also minimizes the total messaging overhead of the system. The adaptive hybrid scheduling in [22] dynamically adjusts the data items and amount of bandwidth assigned to the push and pull sets. The system assigns more bandwidth to the frequently requested data items and less bandwidth to the items having fewer requests. The strategy reduces the overall average access time of the hybrid system. Another hybrid approach is presented in [12] which divides the data items into two disjoint sets: one for push and the other for pull. Here, push-one and pull-all approach is followed, but ensuring that on average no more than one item is in the pull queue at a particular instance of time. This restriction on the pull-queue size is removed in [23] and a push-one and pull-one approach is adopted improving on the performance of [12]. Finally, the work in [29] has considered the user retrial phenomenon to incorporate the impatience of the clients on hybrid systems that transmit push and pull items on separate channels. In [29], it is proved the existence of a broadcast scheme that ensures optimal system performance, with respect to the system throughput, the grade and the quality of service, under the retrial phenomenon.

2.1 Motivations

A careful look into the existing research works on hybrid scheduling that use the same channel to transmit both push- and pull-sets, reveals that most of the hybrid scheduling algorithms consider only unit length

data items, which is not the case in heterogeneous systems where items may have variable lengths. Also, none of the hybrid scheduling strategies have considered the priorities (classification) between the clients, which often play significant role in today's differentiated services offered by the service providers. At this point of time, some questions arise:

1. Can we develop a hybrid scheduling algorithm for asymmetric environment with variable length data items, such that the cut-off (segregation) point between the push and pull system is independent of the build-up point, used to control the maximum length of the pull queue in [12]?

2. Is it possible to stop the trend of the system proposed in [12] of going into pure push scheduling even at a high load and/or all items having same degree of probability?

3. What will be the most efficient and fair measure for disseminating an item from the pull system? While response time is often used for such measures in homogeneous environments where all the data items have the same length, optimizing the *stretch* (i.e., *max-request min-service-time first*), is more fair than optimizing the response time alone for heterogeneous environments where every data item can have different length, as service time for every item maybe different. The stretch ($\mathcal{S}_i$) for an item is defined as $\mathcal{S}_i = \frac{RequestCountforitemi}{Length_i^2}$. Although the stretch scheduling algorithm is used for both pure push [30] and pure pull [3] systems, according to our knowledge no hybrid scheme has explored its efficiency.

4. Can we explore the priorities associated with the clients to improve the performance of the hybrid scheduling and make it more practical? Obviously, higher priority clients should be provided their requested items more quickly than the lower priority ones. Thus, the client priorities also influence the service of data items.

The above set of questions motivates us to develop a new, and more efficient hybrid algorithm.

2.2 Our Contributions

Our proposed solution partitions the data items in the push-set and the pull-set, but it chooses the value of the cut-off point between those two sets independent of the pull-queue size. After each single broadcast, we do not flush out the pull-queue, instead, we just pull one single item which has the largest stretch-value. Note that simultaneously with every push and pull, a certain number of new access/requests arrive to the server. However, only one item, the item that has accumulated the largest stretch-value is extracted from the queue to be pulled. Thus the pull-queue grows up fast at the beginning, but it can never grow more than the size of pull set. The request-count of every pull item is increased with the arrival of the new instances of

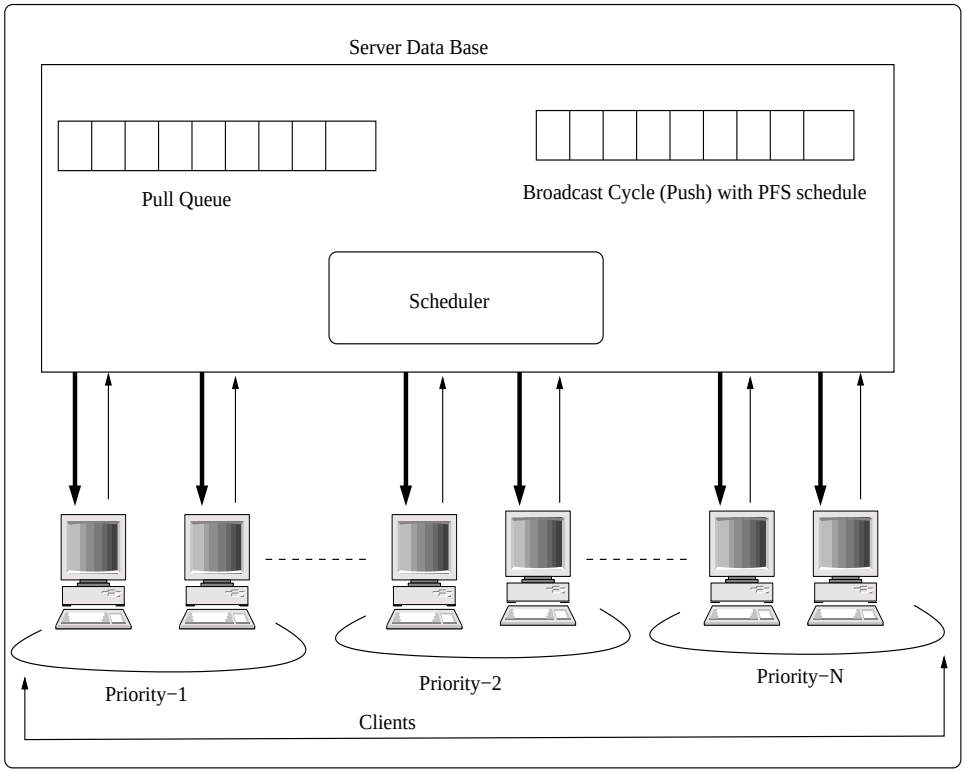

Figure 1: Asymmetric Hybrid Scheduling System

same data item. A max-heap data structure is used to capture these requests and maintain the pull-queue as a priority-queue.

We have developed a suitable analytical model to capture average measures of the system delay (expected access time), where the access probabilities are governed by Zipf's distribution, and the arrival at the server by Poisson's distribution. The analysis is then enhanced to capture the client priorities. Since the analytic model of the system performance has a good precision, it is possible by varying the value of K among all possible values to determine, for a given arrival rate and Zipf's distribution, the value of the cut-off point that minimizes the average system delay.

3 Proposed Hybrid Algorithm

We assume an ideal environment with a single server serving multiple clients, thereby imposing asymmetry. Figure 1 shows the schematic diagram of such an asymmetric environment consisting of a single server and multiple clients with different priorities. The uplink bandwidth is much less than the downlink bandwidth. The database at the server consists of a total number of D distinct items, out of which, let us say, K items are pushed and the remaining $(D - K)$ items are pulled. The items have variable lengths, and each item j has a different access probability P_i. The service time for an item is dependent on the size of that item. The

larger the length of an item, the higher is its service time. We say that a client *accesses* an item if that item is *pushed* whereas a client *requests* an item if that item belongs to the *pull set*.

A *Packet Fair Scheduling (PFS) Algorithm* [13] has better performance than other push based schemes and has been studied well analytically [10]. PFS makes a schedule based on the popularity of the items: transmitting popular items more often than the less popular ones. The strategy schedules the data items in an order such that two consecutive instances of the same data items are as equally spaced as possible. The expected spacing ς_i between two occurrences of item i is given by $\varsigma_i = \left[\sum_{i=1}^{M} \sqrt{\hat{P}_i\, l_i}\right] \sqrt{\frac{l_i}{\hat{P}_i}}$ where P_i is the probability of i^{th} data item and $\hat{P}_i = \frac{P_i}{\sum_{j=1}^{K} P_j}$ is the probability of item i normalized with respect to the sum of the probabilities of the items of the push set. We have adopted PFS in our hybrid algorithm as the push mechanism. In the rest of this paper, the term push-scheduling will refer to the cyclic scheduling produced by the PFS algorithm applied to the push set.

On the other hand, for the pull mechanism, we select the item to be pulled as that one that has maximum stretch $\mathcal{S}_i = \frac{Request\ Count\ for\ item\ i}{Length_i^2}$. We have assumed an ideal environment, where the client needs to send the server its request for the required item i along with its unique ID and waits until it listens for i on the channel (see Figure 2). Note that the behavior of the client is independent of the fact that the requested item belongs to the push-set or the pull-set. As mentioned earlier, the Huges Network Network Systems DirecPC architecture [14] is a suitable example for such broadcast system.

```
Procedure CLIENT-REQUEST:
begin
     send to the server the request for a particular item;
     wait until listen for that item on the channel
end
```

Figure 2: Client side algorithm

The server maintains the database of all items. The system starts as a pure pull-based scheduler (i.e., the push set is empty) assuming that all the items have the same access probability and few requests occur. Then, based on the requests received for each item during a certain interval of time, it dynamically moves to a hybrid system with the data items separated into the push set and the pull set. Precisely, at regular interval of time, the server monitors the data access probabilities of the items and the arrival rate of the requests. If the values registered by the server significantly deviate by the values for which the current segregation point between the push and the pull sets has been computed, the cut-off point must be recomputed. This will be done, as explained later in Section 4.2, by the analytic model devised in Section 4.

Once the cut-off point is fixed, the server continuously alternates a push-phase, in which a single item

```
Procedure Hybrid Scheduling;
while (true) do
begin
     Push-Phase:
         broadcast an item selected according to the Packet Fair Scheduling;
         handle the requests occurring during the push-phase;
     if  the pull-queue is not empty then
     Pull-Phase:
        extract from the pull-queue the item whose stretch is maximum;
        if tie
          extract the item whose sum of the clients' priority is high;
          if tie
            extract the item with the smallest index;
        clear the number of pending requests for this item, and pull-it;
     handle the requests occurring during the pull-phase;
end;
```

Figure 3: Hybrid scheduling algorithm

is broadcasted, and a pull-phase, in which a single item is disseminated, when there are clients waiting for pull items. Between each push- and pull-phase, the server handles the arrived requests. Precisely, the server simply collects statistics about the requests for the push items. Indeed, the decision about which item to broadcast is taken according to the PFS algorithm, and independent of any current request. While, if the request is for a pull item, other than collecting statistics for such an item, the server inserts it into the pull queue with its arrival time and updates the stretch value of the requested item. After every push, if the pull queue is not empty, the server chooses the item whose stretch value is maximum. It might happen that more than one item have same stretch value. In that case, the server considers the item that has maximum priority. Priorities of the items are estimated by adding the priorities of the clients requesting that particular item, and then normalizing it. The ID of the client is used by the server to calculate its priority. If there is still a tie, the server selects the item having smallest index. It now pulls that item and clears the pending requests for that item in the pull-queue which is implemented as a max-heap. Figure 3 provides the pseudo-code of the hybrid scheduling algorithm executing at the server-side while the push and pull sets do not change.

4 Modelling the System

In this section we evaluate the performance of our hybrid scheduling by developing suitable analytical models. The goal of this analysis is twofold: it is used (i) to estimate the minimum expected waiting time (delay) of the hybrid system when the size, say K, of the push set is known, and (ii) to determine the cut-off point between the push-set and pull-set when the system conditions (arrival rate and access probabilities) change. Indeed, since the waiting time is dependent on the size K of the push set, we investigate, by the analytical model, into the delay dynamics for different values of K in order to derive the cut-off point, that is the value

of K that minimizes the system delay.

Before proceeding further, let us enumerate the parameters and assumptions used in our model:

1. The database consists of $\mathcal{D} = \{1, \ldots, D\}$ distinct items, sorted by non increasing access probabilities $\{P_1 \geq \ldots \geq P_D\}$. Basically, the access probability gives a measure of item's popularity among the clients. We have assumed that the access probabilities (P_i) follow the *Zipf's distribution* with *access skew-coefficient* θ, such that $P_i = \frac{(1/i)^\theta}{\sum_{j=1}^{n}(1/j)^\theta}$. Every item has different length randomly distributed between 1–L^*, where L^* is the maximum length.

2. Let C, K and $\varrho_{(cl)}$, respectively, denote the maximum number of clients, the size of the push set and priority of client cl. The server pushes K items and clients pull the remaining $(D - K)$ items. Thus, the total probability of items in push-set and pull-set are respectively given by $\sum_{i=1}^{K} P_i$ and $\sum_{i=K+1}^{D} P_i = (1 - \sum_{i=1}^{K} P_i)$.

3. The service times of both the push and pull systems are exponentially distributed with mean μ_1 and μ_2, respectively.

4. The arrival rate in the entire system is assumed to obey the Poisson distribution with mean $\lambda_{arrival}$.

Table 1 lists the symbols with their meanings used in the context of our analysis. Now, we are in a position to analyze the system performance for achieving the minimal waiting time.

Table 1: Symbols Used for Performance Analysis

Symbols	Descriptions
D	Maximum number of items
C	Maximum number of clients
i	Index of data item
K	Size of the the push set
P_i	Access Probability of item i
L_i	Length of item i
λ	Pull Queue Arrival Rate
$\lambda_{arrival}$	System Arrival Rate
μ_1	Push Queue Service Rate
μ_2	Pull Queue Service Rate
S_i	Space between the two instances of data item i
$\varrho_{(cl)}$	Priority of client cl
ϱ_i	Priority of data item i
$E[W_{pull}]$	Expected Waiting Time of Pull System
$E[W_{pull}^q]$	Expected Waiting Time of Pull Queue
$E[\mathcal{L}_{pull}]$	Expected Number of items in the Pull system
$E[\mathcal{L}_{pull}^q]$	Expected Number of items in the Pull queue

4.1 Minimal Expected Waiting Time

Figure 4 illustrates the birth and death process of our system model, where the arrival rate in the pull-system is given by $\lambda = (1 - \sum_{i=1}^{K} P_i)\lambda_{arrival}$. First, we discuss the state space and possible transitions in this model.

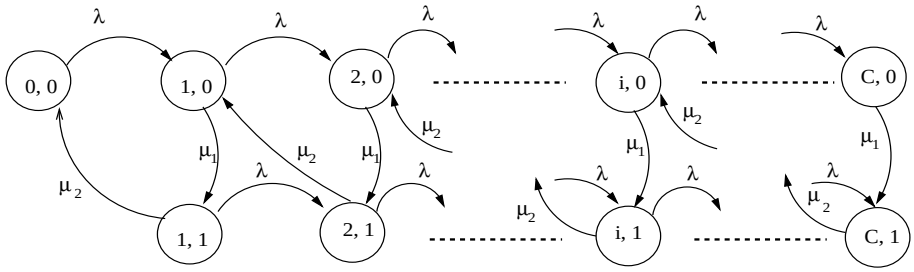

Figure 4: Performance Modelling of Our Hybrid System

1. Any 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.

2. The arrival of a data item in the pull-system results in the transition from state (i, j) to state $(i+1, j)$, for $0 \leq i \leq C$ and $0 \leq j \leq 1$. However, the service of an item results in two different actions. Since the push system is governed by packet fair scheduling, the service of an item in the push-queue results in transition from state $(i, 0)$ to state $(i, 1)$, for $0 \leq i \leq C$. On the other hand, the service of an item in the pull queue results in transition from state $(i, 1)$ to the state $(i-1, 0)$, for $1 \leq i \leq C$.

3. Naturally, the state (0, 0) of the system represents that the pull-queue is empty and any subsequent service of the items in the push system leaves it in the same $(0, 0)$ state. Obviously, state $(0, 1)$ is not valid because the service of an empty pull-queue is not possible.

In the steady-state, using the flow-balance conditions of *Chapman-Kolmogrov's* equation [11], we have the following equation for representing the initial system behavior:

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

where $p(i, j)$ represents the probability of state (i, j). The overall behavior of the system for push (upper chain in Figure 4) and the pull system (lower chain) are given by the following two generalized equations:

$$p(i,0)(\lambda + \mu_1) = p(i-1,0)\lambda + p(i+1,1)\mu_2 \tag{2}$$

$$p(i,1)(\lambda + \mu_2) = p(i,0)\mu_1 + p(i-1,1)\lambda \tag{3}$$

The most efficient way to solve of Equations (2) and (3) is using the *z-transforms* [11]. The resulting solutions are of the form:

$$P_1(z) = \sum_{i=0}^{C} p\,(i,0)\,z^i \tag{4}$$

$$P_2(z) = \sum_{i=0}^{C} p(i,1)\, z^i. \tag{5}$$

Now, dividing both sides of Equation (2) by μ_2, letting $\rho = \frac{\lambda}{\mu_2}$ and $f = \frac{\mu_1}{\mu_2}$, performing subsequent z-transform as in Equation (4) and using Equation (1), we obtain

$$\begin{aligned} P_2(z) &= p(1,1) + z(\rho + f)[P_1(z) - p(0,0)] - \rho\, z^2\, P_1(z) \\ &= \rho\, p(0,0) + z(\rho + f)[P_1(z) - p(0,0)] - \rho z^2\, P_1(z) \end{aligned} \tag{6}$$

Similarly, transforming Equation (3) and performing subsequent derivations we get,

$$P_2(z) = \frac{f\,[P_1(z) - p(0,0)]}{(1 + \rho - \rho\, z)} \tag{7}$$

Now, estimating the system behavior at the initial condition, we state the following normalization criteria:

1. The occupancy of pull states is the total traffic of pull queue and given by: $P_2(1) = \sum_{i=1}^{C} p(i,1) = \rho$.
2. The occupancy of the push states (upper chain) is similarly given by: $P_1(1) = \sum_{i=1}^{C} p(i,0) = (1 - \rho)$.

Using these two relations in Equation (6), the idle probability, $p(0,0)$, is obtained as follows:

$$\begin{aligned} P_2(1) &= \rho\, p(0,0) + (\rho + f)\,[P_1(1) - p(0,0)] - \rho\, P_1(1) \\ \rho &= \rho\, p(0,0) + (\rho + f)\,[1 - \rho - p(0,0)] - \rho\,(1 - \rho) \\ &= f(1 - \rho) - f\, p(0,0) \\ f\, p(0,0) &= f\,(1 - \rho) - \rho \\ p(0,0) &= 1 - \rho - \frac{\rho}{f} \\ &= 1 - 2\,\rho, \text{ (if } \mu_1 = \mu_2) \end{aligned} \tag{8}$$

Generalized solutions of Equations (6) and (7) to obtain all values of probabilities $p(i,j)$ become very complicated. Thus, the best possible way is to go for an expected measure of system performance, such as the average number of elements in the system and average waiting time. The most convenient way to derive the expected system performance is to differentiate the z-transformed variables, $P_1(z)$ and $P_2(z)$ and capture their values at $z = 1$. Thus, differentiating both sides of Equation (6) with respect to z at $z = 1$, we estimate the expected number of items in the pull-system, $E[\mathcal{L}_{pull}]$, as follows:

$$\begin{aligned} \left[\frac{dP_2(z)}{dz}\right]_{z=1} &= (\rho + f)\left[\frac{dP_1(z)}{dz}\right]_{z=1} + P_1(1)\,(f - \rho) \\ &\quad -p(0,0)\,(\rho + f) - \rho\left[\frac{dP_1(z)}{dz}\right]_{z=1} \end{aligned}$$

$$
\begin{aligned}
E[\mathcal{L}_{pull}] &= (\rho + f)\,\mathcal{N} + (1-\rho) - (\rho + f)\left(1 - \rho - \frac{\rho}{f}\right) - \rho\,\mathcal{N} \\
&= \left(\frac{\mu_1}{\mu_2}\right)\mathcal{N} + \left(1 - \frac{\lambda}{\mu_2}\right) - \left(\frac{\lambda + \mu_1}{\mu_2}\right)\left(1 - \frac{\lambda}{\mu_2} - \frac{\lambda}{\mu_2}\right) \\
&= \mathcal{N} + \left(1 - \frac{\lambda}{\mu}\right) - \left(1 + \frac{\lambda}{\mu}\right)\left(1 - 2\,\frac{\lambda}{\mu}\right), (\text{if}\,\mu_1 = \mu_2 = \mu) \qquad (9)
\end{aligned}
$$

where $\mathcal{N}$ is the average number of users waiting in the pull queue when push is being served. Once we have the expected number of items in the pull system from Equation (9), using Little's formula [11] 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:

$$
\begin{aligned}
E[W_{pull}] &= \frac{E[\mathcal{L}_{pull}]}{\lambda} \\
E[\mathcal{L}^q_{pull}] &= E[\mathcal{L}_{pull}] - \frac{\lambda}{\mu_2} \\
E[W^q_{pull}] &= E[W_{pull}] - \frac{1}{\mu_2} \qquad (10)
\end{aligned}
$$

Note that, there is a subtle difference between the concept of pull system and pull queue. While the pull queue considers only the items waiting for service in the queue, the pull system also includes the item(s) currently being serviced. However, the expected waiting time for the pull system discussed above does not consider the priorities associated with the individual data items. Such estimate can suffice the need for average system performance when every item in the pull queue has accumulated different number of requests. However, when any two items contain the same number of pending requests, the priorities associated with those two items come into consideration. This will affect the arrival and service of the individual data items. Thus, a smart system should consider the priorities of the data items influenced by the client priorities.

4.1.1 Role of Client Priorities

Every client cl is associated with a certain priority $\varrho_{(cl)}$ that reveals its importance. The priority of a particular data item is derived from the total normalized priorities of all the clients currently requesting for that data item. Thus, if a particular data item i is requested by a set $\mathcal{C}$ of clients, then its priority is estimated as: $\varrho_i = \frac{1}{|\mathcal{C}|} \times \sum_{\forall cl \in \mathcal{C}} \varrho_{(cl)}$. The lower the value of $\varrho_{(cl)}$, the higher is the priority. When two items have the same stretch value, the item with higher priority is serviced first. This is also practical since such an item is requested by more important clients than its counterpart. Considering a non-preemptive system with many priorities, let us assume the data items with priority ϱ_j have an arrival rate λ_j and service time μ_{2_j}. The occupancy arising due to this jth data item is represented by $\rho_j = \frac{\lambda_j}{\mu_{2_j}}$, for $1 \leq j \leq max$, where max represents maximum possible value of priority. Also, let $\sigma_j = \sum_{i=1}^{j} \rho_i$. In the boundary conditions we have, $\sigma_0 = 0$ and $\sigma_{max} = \rho$. If we assume that a data item of priority i arrives at time t_0 and gets serviced

at time t_1, then the waiting time is $t_1 - t_0$.

Let at t_0, there be n_j data items present having priorities j. Also let, S_0 be the time required to finish the data item already in service, and S_j be the total time required to serve n_j. During the waiting time of any data item, n'_j new items having the same number of pending requests and higher priority can arrive and go to service before the current item. If S'_j be the total time required to service all the n'_j items, then the expected waiting time will be,

$$E[W_{pull}^{q(i)}] = \sum_{j=1}^{i-1} E[S'_j] + \sum_{j=1}^{i} E[S_j] + E[S_0] \tag{11}$$

In order to get a reasonable estimate of $W_{pull}^{q(i)}$, three components of Equation (11) needs to evaluated individually.

(i) *Estimating* $E[S_0]$: The random variable S_0 actually represents the remaining service time, and achieves a value 0 for idle system. Thus, the computation of $E[S_0]$ is performed in the following way:

$$\begin{aligned} E[S_0] &= Pr[\text{Busy-System}] \times E[S_0|\text{Busy-System}] \\ &= \rho . \sum_{j=1}^{max} E[S_0|\text{Serving items having priority-j}] \\ &\quad \times Pr[\text{items having priority j}] \\ &= \rho \sum_{j=1}^{max} \frac{\rho_j}{\rho \mu_{2_j}} \\ &= \sum_{j=1}^{max} \frac{\rho_j}{\mu_{2_j}} \end{aligned} \tag{12}$$

(ii) *Estimating* $E[S_j]$*:* The inherent independence of Poisson process gives the flexibility to assume the service time $S_j^{(n)}$ of all n_j customers to be independent. Thus, $E[S_j]$ can be estimated using the following steps:

$$\begin{aligned} E[S_j] &= E[n_j S_j^{(n)}] = E[n_j] E[S_j^{(n)}] \\ &= \frac{E[n_j]}{\mu_{2_j}} \rho_j E[W_{pull}^{q(j)}] \end{aligned} \tag{13}$$

(iii) *Estimating* $E[S'_j]$*:* Proceeding in a similar way and assuming the uniform property of Poisson,

$$E[S'_j] = \frac{E[n'_j]}{\mu_{2_j}} \rho_j E[W_{pull}^{q(i)}] \tag{14}$$

The solution of Equation (11) can be achieved by combining the results of Equations (12)–(14) and using Cobham's iterative induction [11]. Finally, the new overall expected waiting time of the pull system

$(E[\widehat{W^q_{pull}}])$ is achieved in the following manner:

$$E[W^{q(i)}_{pull}] = \frac{\sum_{j=1}^{max} \rho_j/\mu_{2_j}}{(1-\sigma_{i-1})(1-\sigma_i)} \tag{15}$$

$$E[\widehat{W^q_{pull}}] = \sum_{i=1}^{max} \frac{\lambda_i E[W^{q(i)}_{pull}]}{\lambda} \tag{16}$$

Thus, the expected access-time, $E[T_{hybacc}]$, of our hybrid system is given by:

$$E[T_{hybacc}] = \left(\sum_{i=1}^{K} \varsigma_i \hat{P}_i\right)\left(\sum_{i=1}^{K} P_i\right) + E[\widehat{W^q_{pull}}] \times \sum_{i=K+1}^{D} P_i = \sum_{i=1}^{K} \varsigma_i P_i + E[\widehat{W^q_{pull}}] \times \sum_{i=K+1}^{D} P_i, \tag{17}$$

where according to the packet-fair-scheduling, $\varsigma_i = \left[\sum_{i=1}^{M} \sqrt{\hat{P}_i\, l_i}\right] \sqrt{\frac{l_i}{\hat{P}_i}}$ and $\hat{P}_i = \frac{P_i}{\sum_{j=1}^{K} P_j}$. The above expression provides an estimate of the average behavior of our hybrid scheduling system.

4.2 Estimation of the Cut-off value

One important system parameter now needs to be investigated is the cut-off point, that is the value of K which minimizes the expected waiting time in the hybrid system. It is quite clear from Equations (9)–(17) that the dynamics of minimum expected waiting time (delay) is governed by K. Furthermore, Equation (17) has two components for the minimum expected waiting time. While $\sum_{i=1}^{K} \varsigma_i\, P_i$ provides an estimate for the delay accumulated from the push system, $E[\widehat{W^q_{pull}}] \times \sum_{i=K+1}^{D} P_i$ represents the delay component arising from the pull system. For different values of K, these two components change dynamically. Intuitively, for low values of K, most of the items are pulled and the significant delay is accrued from the pulled items. The scenario gets reversed for high values of K. It seems hard to derive a closed form solution for the optimal value of K. The cut-off-point can be obtained algorithmically by estimating both the delay components and overall expected delay at each iteration and preserving the value of K which provides minimum expected delay. Alternatively to derive the cut-off point, for a fixed value D, we analyze the pattern of the expected waiting time with different values of K and fit the values to obtain a closed form equation of the pattern. We have used *polynomial fit* with degree 3 to identify these patterns for 3 different values of the access skew coefficient, $\theta = \{0.40, 0.80, 1.20\}$. This leads to the equations for $E[T_{hybacc}] = f(K)$. For the sake of notational simplicity, we use y to represent $E[T_{hybacc}]$. We first differentiate y with respect to K to get the first derivative $\frac{\partial y}{\partial K}$. At the extreme points (maxima or minima) the derivative will be 0. Hence, the expression for $\frac{\partial y}{\partial K}$ is made equal to 0 to get the extreme values of K. As the polynomial is of degree 3, the slope of the curve $\frac{\partial y}{\partial K}$ is of degree 2. Hence, two possible values of K are possible. We have taken only that value of K which falls in the range $0 \leq K \leq D$, as the minimum and maximum possible values of cut-off point are 0 and D, respectively. At this particular value of K, we compute the value of y using

the original equation. This is the minimum expected access time with corresponding cut-off-point for a particular value of θ. In order to check the minima, we have also computed the second order derivative with respect to K and showed this derivative is positive (minimality condition) for that K. This is repeated for $\theta = \{0.40, 0.80, 1.20\}$.

For example, the following three optimal values of K achieves the minimum waiting time for different values of θ and $D = 100$. When $\theta = 0.40$,

$$\begin{aligned} y &= 27 \times 10^{-5} K^3 - 0.028K^2 - 0.5K + 160 \\ \left[\frac{\partial y}{\partial K}\right]_{min\_y} &= 81 \times 10^{-5} K^2 - 0.056K - 0.5 = 0 \\ K &= 77 \\ y &= 78.75191 \end{aligned} \tag{18}$$

When $\theta = 0.80$

$$\begin{aligned} y &= 13 \times 10^{-5} K^3 - 0.11K^2 - 0.34K + 100 \\ \left[\frac{\partial y}{\partial K}\right]_{min\_y} &= 39 \times 10^{-5} K^2 - 0.22K - 0.34 = 0 \\ K &= 69 \\ y &= 66.875 \end{aligned} \tag{19}$$

When $\theta = 1.20$

$$\begin{aligned} y &= 0.01K^2 - 4 \times 10^{-5} K^3 - 0.62K + 55 \\ \left[\frac{\partial y}{\partial K}\right]_{min\_y} &= 0.02K - 12 \times 10^{-5} K^2 - 0.62 = 0 \\ K &= 41 \\ y &= 43.633 \end{aligned} \tag{20}$$

Figure 5 shows the variation of expected access time with different values of the size of the push set. The overall expected waiting time always achieves more or less a parabolic (bell-shaped) form with the global minima occurring at $K = \{77, 69, 41\}$ for $\theta = \{0.40, 0.80, 1.20\}$, respectively. The corresponding minimum expected waiting time is $\{79, 67, 44\}$ time units.

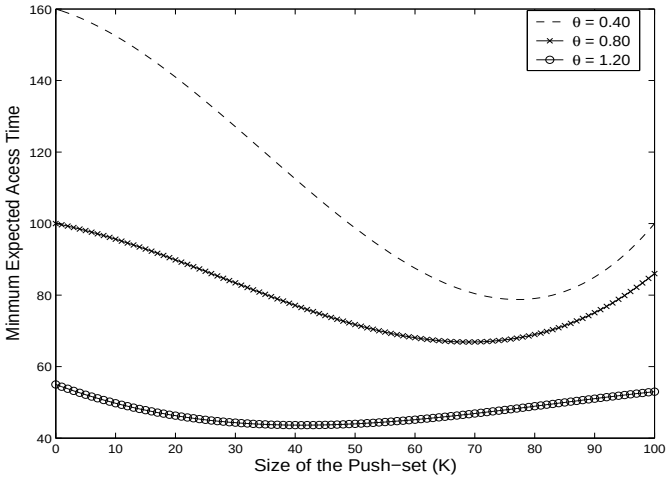

Figure 5: Variation of Expected Access Time with the Size of the Push Set

5 Experimental Results

In this section we validate our hybrid system by performing simulation experiments. The primary goal is to reduce the expected access time. We enumerate below the salient assumptions and parameters used in our simulation.

1. Simulation experiments are evaluated for a total number of $D = 100$ data items.

2. Arrival rate, $\lambda_{arrival}$, is varied between 5–20. The values of μ_1 and μ_2 are estimated as: $\mu_1 = \sum_{i=1}^{K}(P_i \times L_i)$ and $\mu_2 = \sum_{i=K+1}^{D}(P_i \times L_i)$.

3. Length of data items is varied from 1 to 5. An average length of 2 is assumed.

4. Every client is assumed to have some priority randomly assigned between 1 and 5. These priorities are so defined that the lower the value, the higher the priority.

5. To keep the access probabilities of the items from being similar to very skewed, θ is dynamically varied from 0.20 to 1.40.

6. To compare the performance of our hybrid system, we have chosen 4 different hybrid scheduling strategies [21, 22, 12, 25] as performance benchmarks.

Figures 6 and 7 respectively demonstrate the variation of the expected access-time with different values of K and θ, for $\lambda = 10$ and $\lambda = 20$, in our hybrid scheduling algorithm. In both cases, the expected access-time is minimum ($\sim$ 40 time units) for high values of θ ($\sim$ 1.3) and moderate K. With decreasing values of K, the expected access-time increases. This is because as K decreases, the number of items in the pull queue

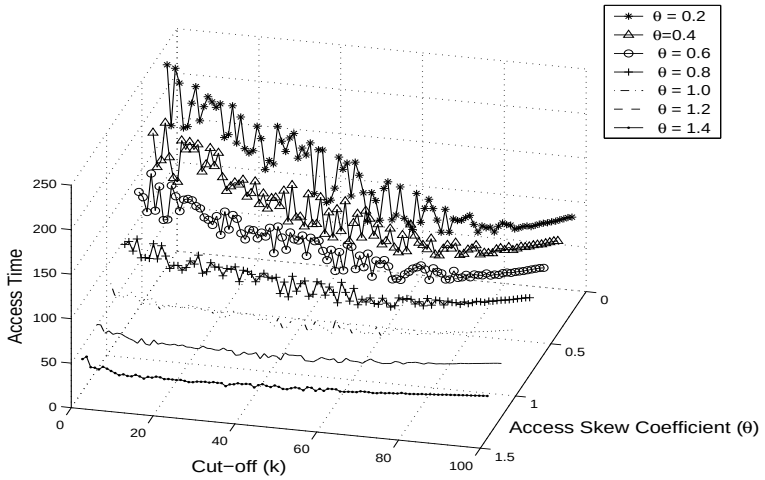

Figure 6: System Behavior with $\lambda = 10$

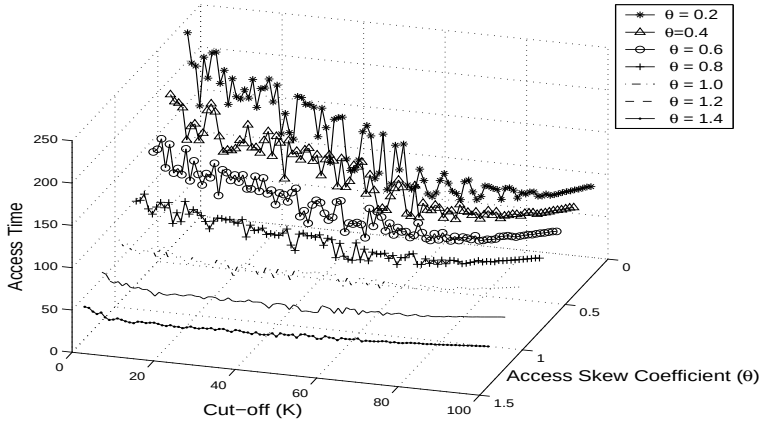

Figure 7: System Behavior with $\lambda = 20$

increases and those items take much more time to get serviced. On the other hand, the average access time also increases for very high values of K. This is because for pretty high K, the push set becomes very large and the system repeatedly broadcasts data items which are even not popular. Thus, the optimal performance is achieved when K is in the range 40–60.

Figure 8 shows the results of performance comparison, in terms of expected access time (in seconds), between our newly proposed hybrid algorithm with three existing hybrid schemes due to Su, et al. [25], Oh, et al. [22], and Guo, et. al. [12]. The plots reveal that our new algorithm achieves an improvement of $\sim 2 - 6$ secs. The possible reasons lie in the fact that while these existing scheduling algorithms use MRF and Flat scheduling to select an item for transmission from the pull and push systems, our new algorithm uses the *stretch*, i.e., *max-request min-service-time* based scheduling and *packet fair scheduling* for pull and push systems, respectively. The effective combination of these two scheduling principles result in the lower expected access time in our hybrid scheduling algorithm.

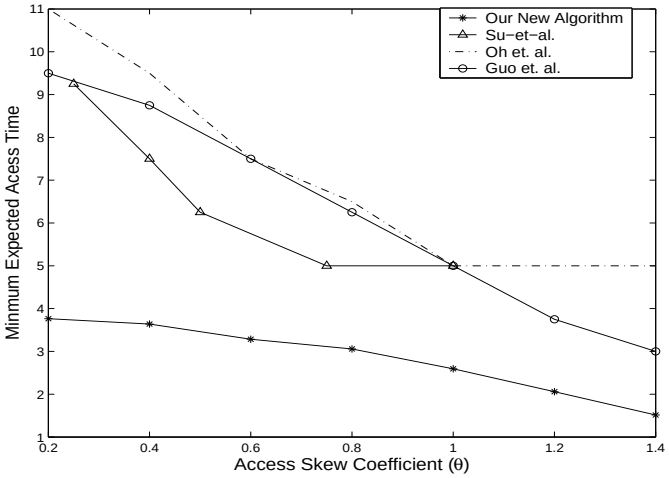

Figure 8: Performance comparison with varying skewness

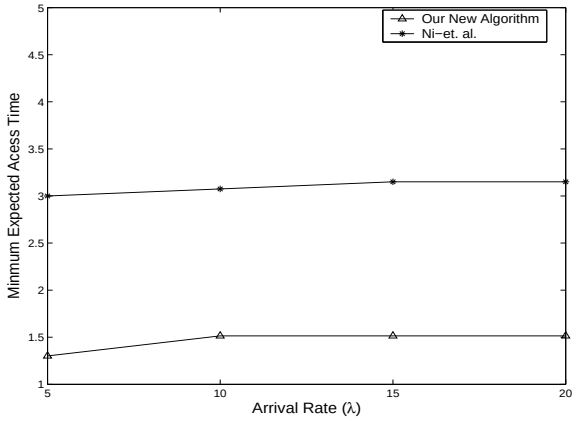

Figure 9: Performance comparison with different arrival

In order to demonstrate the performance efficiency of the proposed hybrid scheduling, we have also looked into the *minimum expected access time* (for a particular K and θ) with different arrival rates (λ). The hybrid scheduling algorithm due to [21] is chosen for comparison. Figure 9 points out that our algorithm consistently gains over existing hybrid scheduling [21] with different arrival rates. Note that the variation of expected access time with different arrival rates is pretty low. This also demonstrates the stability of our hybrid scheduling system.

Figure 10 depicts the comparative view of the analytical results, provided in Equation (17), with the simulation results of our hybrid scheduling scheme. The analytical results closely match with the simulation results for expected access time with almost $\sim 90\%4$ and $\sim 93\%$ accuracy for $\lambda = 5$ and $\lambda = 20$, respectively. Thus, we can conclude that the performance analysis is capable of capturing the average system behavior with good accuracy. The little ($\sim$ 7–10%) difference exists because of the assumption of memory-less property associated with arrival rates and service times in the system.

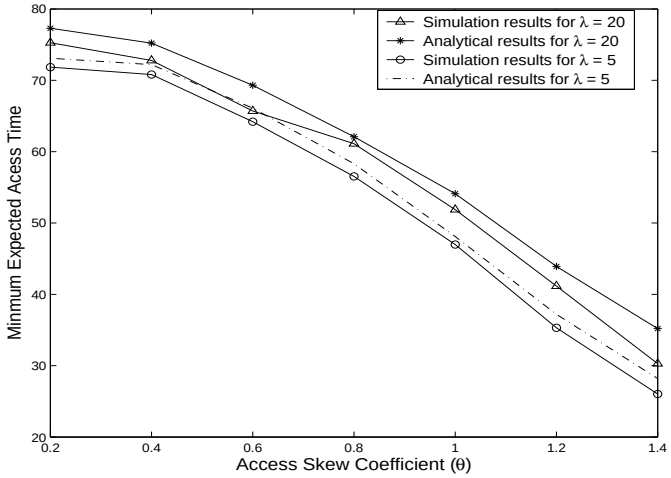

Figure 10: Simulation Vs Analytical Results

Let us now investigate the dynamics of the cut-off point (K) achieved by our hybrid scheduling strategy. Figure 11 demonstrates that K lies in the range of 40–60 for three different arrival rates such as $\lambda = [5, 10, 20]$. Intuitively, this points out that the system has achieved a fair balance between push and pull systems, thereby efficiently combining both the scheduling strategies to achieve the minimum expected access time.

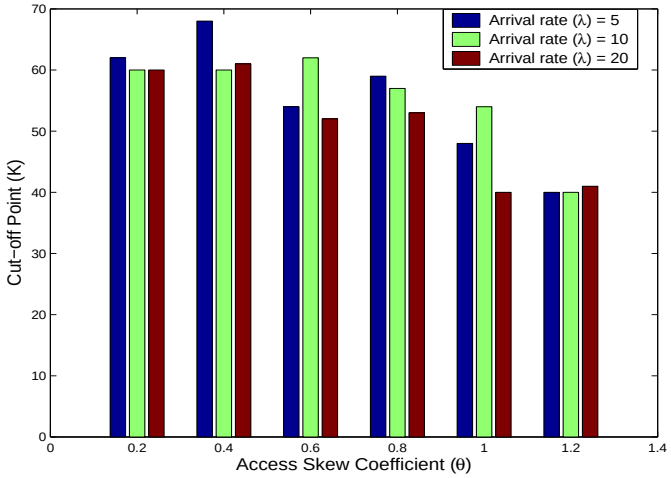

Figure 11: Variation of Cutoff-point (K)

Figure 12 provides the comparison of the variation of optimal cut-off point provided by simulation and analytical results, for different values of access skew coefficient, θ. The plots point out that the simulation and analytical results of optimal cut-off point closely matches with a difference of only $\sim$ 1%–2%.

6 Conclusions and Future Works

Since neither push nor pull based scheduling alone can optimize the system performance, a new hybrid scheduling algorithm that combines the advantages of both push and pull scheduling has been developed in this paper for an asymmetric heterogeneous environment. Our algorithm takes into consideration data items of variable lengths and clients with different priorities. By separating the most demanded items from the less

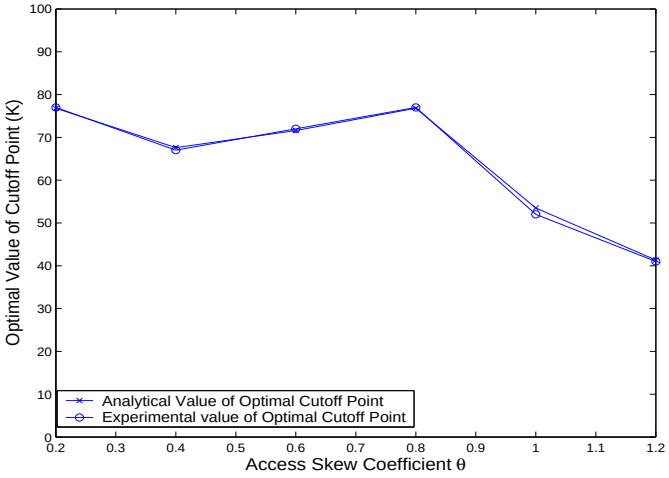

Figure 12: Simulation Vs Analytical Results of the Optimal Cut-Off point (K)

demanded ones, thus broadcasting the former and disseminating the latter, the total overall expected waiting time of the system is shown to decrease. At any instant of time, the item to be broadcast is designated by applying packet fair scheduling principle to the push items, while the item to be pulled is the one stored in the pull-queue with optimal stretch, i.e., max-request min-service-time value. For items having the same stretch value, we have chosen the item with higher priority for service. The priority of an item is determined from the priorities associated with the individual clients requesting that particular data item. The cut-off point between push and pull items is chosen in such a way that the overall expected access time of the hybrid system is minimized, and is independent of the number of pending requests arriving in a time unit. Performance analysis and simulation results demonstrate the improvement of our scheme over pure push and existing hybrid scheduling systems. Moreover by choosing an optimal cut-off point K a better trade off between the expected access time (push) and expected response time (pull) can be obtained.

Our future interest lies in making our system assumptions more realistic, and in particular in incorporating practical issues like clients impatience. Indeed, as already noticed for the total push systems in [15], impatient clients might leave the system, or as described in [29] impatient clients might reiterate the same request. In our system, multiple requests for a single item by the same client lead to the server an anomalous picture of the system. The server might consider the data item as a popular item, due to the high accumulated number of requests, which is not correct. We would also like to focus on the priority-based service classification for different clients to support differentiated QoS in wireless data networks.

Acknowledgment

The authors would like to thank Prof. A. Boukerche for open-wise discussions on this topic.

References

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik. Braodcast 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] M. H Ammar and J.W Wong. "The design of teletext broadcast cycles", *Performance Evaluations*, vol. 5, No. 4, pp. 235-242, 1985

[5] A. Bar-Noy, R. Bhatia, J. Naor and B. Scieber, "Minimizing Service and Operation Costs of Periodic Scheduling", *Proc. of. ACM-SIAM Symp. on Discrete Algos.* (SODA), pp. 11-20, 1998.

[6] E. K. Burke and A. J. Smith, "Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem", *IEEE Transactions on Power Systems*, Vol. 15 , No. 1, pp. 122 - 128, Feb. 2000.

[7] S.E. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph, and R.H. Katz, "An Architecture for a Secure Service Discovery", *Proc. 5th Int. Conf. Mobile Computing (Mobicom)*, pp. 24-35, Aug. 1999.

[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] J. Fernandez and K. Ramammritham, "Adaptive Dissemination of Data in Time-Critical Asymmetric Environments", *Proc. of* 11^{th} *Euromicro Conf. on Real-Time Systems*, pp. 208-215, 1999.

[10] J. Gecsei. The Architecture of Videotex Systems, *Englewood Cliffs, NJ Publisher: Prentice-Hall*, 1983.

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

[12] Y. Guo, S. K. Das and M. C. Pinotti. A new Hybrid Broadcast scheduling Algorithm for Asymmetric Communication Systems: Push and Pull Data based on Optimal Cut-Off Point. *Mobile Computing and Communications Review (*MC^2R*)*, Vol. 5, No. 4, pp. 39-54, 2001.

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

[14] http://www.direcpc.com, 1997

[15] S. Jiang and N. Vaidya, "Scheduling Data Broadcast to Impatient Users", *Proc. of ACM MobiDE*, pp. 52-59, 1999.

[16] C. Kenyon and N. Schabanel, "The Data Broadcast Problem with Non-Uniform Transmission Times", *Proc. of* 10^{th} *Annual ACM-SIAM Symp. on Discrete Algorithms* (SODA), pp. 547-556, 1999.

[17] 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* (MONET), vol. 4, pp. 117-129, 1999.

[18] F-C. Lee and M-J. Yao, "A global optimum search algorithm for the joint replenishment problem under power-of-two policy", *Computers and Operations Research*, Vol. 30, No. 9, pp. 1319-1333, 2003.

[19] G. Lee and S. C. Lo. Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments. *Mobile Networks and Applications*, Vol. 8, pp. 365-375, 2003.

[20] 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.

[21] 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*, (ICDCSW), pp. 790-796, 2003.

[22] J.H 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.

[23] 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.

[24] N. Saxena, K. Basu and S. K. Das, "Design and Performance Analysis of a Dynamic Hybrid Scheduling for Asymmetric Environment", *IEEE Intl. Workshop on Mobile Adhoc Networks*, WMAN, 2004.

[25] C-J. Su, L. Tassiulas and V. J. Tsotras, "Broadcast Scheduling for Information Distribution", *ACM/Kluwer Wireless networks (WINET)*, vol. 5, No. 2, pp. 137-147, 1999.

[26] W. Sun, W. Shi, B. Shi and Y. Yu, "A Cost-Efficient Scheduling Algorithm for On-Demand Broadcasts", *ACM/Kluwer Wireless Networks (WINET)*, Vol. 9, No. 3, pp. 239-247, 2003.

[27] "Support of Third Generation Services Using UMTS in a Converging Network Environment", *UMTS Forum*, 2002.

[28] N. H. Vaidya and S. Hameed, "Scheduling Data Broadcast in Asymmetric Communication Environments" *ACM/Kluwer Wireless Networks (WINET)*, Vol. 5, No. 3, pp. 171-182, 1999.

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

[30] Y. Wu and G. Cao. Stretch-Optimal Scheduling for On-Demand Data Broadcasts, *Proceedings of Tenth International Conference on Computer Communications and Networks*, pp. 500–504, 2001.

[31] J. Xu, D.L. Lee, Q. Hu, and W.C.Lee, "Data Broadcast", *Handbook of Wireless Networks and Mobile Computing* , I. Stojmenovic, Ed., New York: Wiley, 2003