A Probabilistic Push-Pull Hybrid Scheduling Algorithm for Asymmetric Wireless Environment

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

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

Sajal K. Das
CReWMaN Lab
Comp. Science & Engg. Dept.
University of Texas at Arlington
Arlington, Texas, USA
das@cse.uta.edu

***Abstract*— The vision of mobile computing lies in a seamless connectivity with the mobile clients and transmission of data in precise quality of service (QoS) guarantee. In order to endow such mobile applications with advanced data processing capabilities, efficient, dynamic scheduling algorithms are necessary. In this paper we have introduced a new hybrid scheduling algorithm that probabilistically combines the number of push and pull operations depending on the number of items present in the system and their popularity. The access probabilities of the data items are computed dynamically, without any prior knowledge. The basic elixir of our work lies in the efficiency of the algorithm in obtaining an improved data access-time, even with high system load and items having equivalent degree of access probabilities. The expected waiting time spent by a client is evaluated and compared analytically. Simulation results point out sufficient improvement in average waiting time than pure push systems and some existing hybrid systems.**

I. Introduction

The essence of mobile and ubiquitous computing lies in guaranteeing a pervasive connectivity while delivering a huge set of data within a precise time. The ever-increasing growth of these ubiquitous services require smart scheduling techniques to efficiently transmit the overwhelming data sets to a large number of mobile clients. The asymmetry in uplink and downlink capacity make this scheduling problem more challenging. Broadly all data transmission schemes are divided into two categories: (1) *Push* – where the clients continuously monitor a broadcast process from the server and retrieve their required data items; (2) *Pull* – where the clients initiate the data transfer by sending requests and the server makes an on-demand schedule to satisfy the clients' requests. But both of these push and pull based scheduling possess their own advantages and disadvantages. A fundamental, but very important result in [1] demonstrates that neither push nor pull based scheduling alone can achieve the optimal performance. Efficient combination of both push and pull scheduling are necessary for achieving an (near) optimal performance.

In this paper, we have performed a comparative study of various push and pull based scheduling schemes and subsequently introduced a hybrid scheduling algorithm which achieves best performance in terms of minimum access-time. A new hybrid scheduling is proposed, which probabilistically combines the best scheduling technique for broadcasting push data and on-demand dissemination of pull data, depending on the data items currently present in the system. At any instant of time, the item to be broadcast is selected by applying a PFS-based pure-push scheduling, while the item to be pulled is the one selected from the pull-queue by applying MRF-based pull scheduling. More specifically our contributions include:

1) We compare the performance of various push and pull scheduling algorithms and select the scheme providing the best performance.
2) A dynamic hybrid algorithm is proposed, which does not combine the push and pull in a static, sequential order. Instead, it combines the push and pull strategies *probabilistically depending on the number of items present and their popularity*. In practical systems, the number of items in push and pull set can vary. For a system with more items in the push-set (pull-set) than the pull-set (push-set), it is more effective to perform multiple push (pull) operations before one pull (push) operation. We claim that our algorithm is the first work which introduces this concept in a dynamic manner. This has the power to change the push and pull lists on real time and the minimize the overall delay.
3) The cut-off point used to separate the push and the pull sets is changed dynamically to get the optimal performance. This cut-off point is chosen so that the overall waiting time of the system is also minimized.
4) Extensive performance evaluation is carried out to analyze the average measures of different scheduling schemes.
5) Simulation experiments are performed without any previous knowledge of the data access probabilities. Later the access probabilities are re-computed dynamically, thereby computing the cut-off-point to separate the push and the pull sets dynamically. Simulation results closely match the results of performance evaluation.

The rest of the paper is organized as follows: Section II reviews the existing works on scheduling techniques. Our new hybrid algorithm is presented in Section III. A suitable queuing model is developed in Section IV to analyze the performance of our hybrid system. Simulation results in Section V corroborates the system behavior. Finally, Section VI concludes the paper by summarizing our work and pointers to future research.

II. Overview of Existing Works

Given the wide variety of existing works in scheduling and data transmission, we only highlight the major rele-

0-7803-8798-8/04/$20.00 ©2004 IEEE

vant strategies. In order to achieve optimal expected access time, the push-based broadcast scheduling has been related to the *packet fair queuing* problem [7]. A three-player *client-provider-server* model [3], which logically separates the servers from service providers, is proposed for asymmetric communication environments. The solution proposed in [11] performs dynamic adjustment of *channel allocation trees* to adapt with the changes of clients' data access frequencies. In another adaptive data dissemination scheme [8] for asymmetric wireless environment, the dynamics of broadcast information is subsumed into groups and an online slot-exchange policies between different groups is proposed. Asymptotically optimal algorithms for obtaining data access-costs, which are not proportional to their own waiting time, are provided in [4]. 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 [2] exists in literature. A combination of these two as suggested in [2], 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 [15] is proposed as a more appropriate performance metric. The pull-based real-time data dissemination system, discussed in [5], proposes Aggregated Critical Requests (ACR) scheduling algorithm to meet the predetermined timing constraints.

A hybrid approach that combines the flavors of both push-based and pull-based algorithms in one system, appears to provide a better solution. Perhaps the first hybrid technique for scheduling and data transmission in asymmetric environment was proposed in [1]. In this work, the server pushes all the data items according to some push-based scheduling, but simultaneously the clients are provided with a limited back channel capacity to make requests for the items. The adaptive real-time data transmission strategy in [9] 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 [10], 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 [12], [13] 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) one item from the higher probable set of items and disseminating (pull) one item from the lower probable set in a reciprocal manner.

III. Our Proposed Hybrid Algorithm

In this section, we first take a look at the relative advantages and disadvantages of the different scheduling principles. Subsequently, we choose the scheduling schemes providing the best performance and introduce our new hybrid scheduling technique.

A. Scheduling Techniques:

Before going into the proposal of our new hybrid scheduling scheme, we look through the salient scheduling techniques for push and pull-based systems.

Scheduling for Push-based Broadcasting: The scheduling principle for push-based systems are generally based on either flat scheduling or packet fair scheduling schemes. In *flat scheduling* the items are broadcasted in a round robin fashion. Every data item is assigned a fix time slot and the server transmits the items in regular time slots. Such a scheduling scheme is quite simple and suffers from performance degradation, specially for system having items with skewed requests. On the other hand, *Packet Fair Scheduling* [7] makes a schedule based on the popularity of the items, i.e. transmitting popular items more often than the less popular ones. The instances of every items are equally spaced and the service time required to serve the item i is dependent on the size of the item. The larger the length of an item the higher is its service time.

Scheduling for Pull-based Dissemination: During data dissemination, the client sends requests for particular data item and the server queues these requests. Either *First Come First Serve (FCFS)* or Mo*st Request First (MRF)* scheduling principle is used by the server look into the queue and schedule the delivery of the particular items. In FCFS scheduling principle the item which arrived first in the pull-queue is also serviced first. If the requests for some less popular item precedes the request for a more popular item, the more popular item has to wait until all the less popular items get serviced, thus decreasing the system efficiency. In the MRF scheduling principle, the server checks the pull queue and chooses the item based on *max-requested item* from the pull-queue.

B. Dynamic Hybrid Scheduling Algorithm

Our simulation experiments (described in Section V) corroborates the existing research works in the aspect that PFS and MRF achieves best performance for push and pull scheduling principles respectively. Hence, motivated by the efficacy of hybrid scheduling disciplines, we have investigated to combine the best scheduling principles for push and pull-based systems. However, strict sequential combination of push and pull fails to explore the system's current condition. In practical systems, it is a better idea to perform more than one push operations followed by multiple pull operations, depending on the number of items currently present in the system. The algorithm needs to be smart and efficient enough to get a good estimate of these number of continuous push and pull operations. Our proposed hybrid scheduling scheme performs this strategy based on the number of items present and their popularity.

We have assumed a single server, multiple clients and a database consisting of D distinct items, of which K items are pushed and the remaining $(D - K)$ items are pulled. The access probability P_i of an item i is governed by the Zipf's distribution and depends on the access skew-coefficient θ. When θ is small (value close to 0), P_i is well balanced but as θ increases (value close to 1), the popularity of the data items becomes skewed. From time to time the value of θ is changed

dynamically for our hybrid system, which in turn, results in dynamic variation of P_i and the size of the push and pull sets. PFS and MRF techniques are used for selecting the item to be pushed and pulled respectively. After every push or pull operation, the next push or pull operation is probabilistically determined using the following equation:

$$\begin{aligned}
\gamma_1 &= Pr[push|push] = \frac{K}{D}\sum_{i=1}^{K} P_i \\
\gamma_2 &= Pr[pull|push] = 1 - \gamma_1 \\
\gamma_3 &= Pr[pull|pull] = \frac{D-K}{D}\sum_{i=K+1}^{D} P_i \\
\gamma_4 &= Pr[push|pull] = 1 - \gamma_3
\end{aligned} \tag{1}$$

```
Procedure HYBRID SCHEDULING;
while true do
      begin
      1. select an item using PFS and push it;
      2. consider new arrival requests;
      3. ignore the requests for push item;
      4. append the requests for items in the
      pull queue;
      5. compute probabilities of γ1 and γ2
      6. if (probability <= γ1) goto step 1
      7. else goto step 8
      8. if pull-queue is not empty then
            9. use MRF to extract an item from pull
            queue;
            10. clear pending requests for that item;
            11. Pull that particular item;
            12. consider new arrival requests;
            13. ignore the requests for push item;
            14. append the requests for items in
            pull queue;
      end-if
      15. compute probabilities of γ3 and γ4
      16. if (probability <= γ3) goto step 8
      else goto step 1;
end-while
```

Fig. 1. Hybrid Scheduling Algorithm at the Server

At the server end, the system starts as a pure pull-based scheduler. If the request is for a push item, the server simply ignores the request as the item will be pushed according to the PFS algorithm sooner. However if the request is for a pull item, the server inserts it into the pull queue with the arrival time and updates its stretch value. Figure 1 provides the pseudo-code of the hybrid scheduling algorithm executing at the server-side.

IV. Analytical Underpinnings

In this section we investigate into the performance evaluation of our hybrid scheduling system by developing suitable analytical models. The arrival rate in the entire system is assumed to obey the Poisson distribution with mean λ_1. The service times of both the push and pull systems are exponentially distributed with mean μ_1 and μ_2, respectively. The server pushes K items and clients pull the rest $(D-K)$ items. Thus, the total probability of items in push-set and pull-set are respectively given by $\sum_{i=1}^{K} P_i$ and $\sum_{i=K+1}^{D} P_i = (1 - \sum_{i=1}^{K} P_i)$, where P_i denotes the access probability of item i. We have assumed that the access probabilities P_i follow the *Zipf's distribution* with *access skew-coefficient* θ, such that, $P_i = \frac{(1/i)^\theta}{\sum_{j=1}^{n}(1/j)^\theta}$. After every push the server performs another push with probability γ_1 and a pull with probability γ_2. Similarly, after every pull it performs another pull with probability γ_3 and a push with probability γ_4.

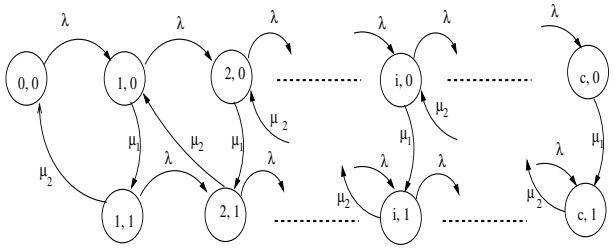

Fig. 2. Performance Modelling of Hybrid System

Figure 2 illustrates the underlying birth and death process of our system, where the arrival rate in the pull-system is given by $\lambda = (1 - \sum_{i=1}^{K} P_i)\lambda_1$. Any state of the overall system is represented by the tuple (i, j), where i represents the number of items in the pull-system. On the other hand, j is a binary variable, with $j = 0$ (or 1) respectively representing whether the push-system (or pull-system) is currently being served by the server.

The arrival of a data item in the pull-system, results in the transition from state (i, j) to state $(i+1, j), \forall i$, such that $0 \leq i < \infty$ and $\forall j \in [0, 1]$. The service results in different actions. The service of an item in the push-queue results in transition of the system from state $(i, j = 0)$ to state $(i, j = 1)$, with probability γ_2, $\forall i$ such that $0 \leq i < \infty$. With probability γ_1 the push-service makes the system to remain in same state. 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 $(i-1, j = 0)$, with probability γ_4 and state $(i-1, j = 1)$ with probability γ_3, $\forall i$, such that $1 \leq i < \infty$. 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, the state $(i = 0, j = 1)$ 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 [6], we have the following three equations representing the system's behavior:

$$p(i,0) = \frac{p(i-1,0)\lambda + p(i+1,1)\gamma_4\mu_2}{(\lambda + \gamma_2\mu_1)} \tag{2}$$

$$p(i,1) = \frac{p(i,0)\gamma_2\mu_1 + p(i-1,1)\lambda}{(\lambda + \gamma_3\mu_2 + \gamma_4\mu_2)} \tag{3}$$

$$p(0,0)\ \lambda = p(1,1)\ \mu_2, \tag{4}$$

where $p(i, j)$ represents the probability of state (i, j). While the first two equations represents the overall behavior of the

system for push (upper chain) and the pull system (lower chain), the last equation actually represents the initial condition of the system. The most efficient way to solve the above Equations is using *z-transforms* [6]. Performing z-transforms of Equation 2 and Equation 3 and using the initial condition, we get a pair of transformed equations:

$$P_2(z)\gamma_4\mu_2 = z[P_1(z) - p(0,0)](\lambda + \gamma_2\mu_1) - z^2\lambda P_1(z) + p(1,1)\gamma_4\mu_2 \quad (5)$$

$$P_2(z) = \frac{\gamma_2\mu_1[P_1(z) - p(0,0)]}{(\lambda + \gamma_3\mu_2 + \gamma_4\mu_2 - z\lambda)} \quad (6)$$

Now, estimating the system behavior at the initial condition, we can state the following normalization criteria: The occupancy of pull states is the total traffic of pull queue and is given by: $P_2(1) = \sum_{i=1}^{C} p(i,1) = \rho$. 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 (5), we can obtain the initial probability, $p(0,0)$. The initial probability of the system, i.e. probability of an empty pull queue is given by the following equation:

$$p(0,0) = \frac{\mu_1(\gamma_2 - \gamma_2\rho - \rho\gamma_4\mu_2)}{\lambda + \gamma_2\mu_1 - \gamma_4\lambda} \quad (7)$$

Generalized solutions of Equations (5) 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 obtain this expected system performance is to differentiate the z-transformed equation (Equation (5)), and capture the values of the z-transformed variable at $z = 1$.

$$\gamma_4\mu_2\frac{dP_2(z)}{dz}|_{z=1} = \gamma_2\mu_1\frac{dP_1(z)}{dz}|_{z=1} + (1-\rho)(\gamma_2\mu_1 - \lambda) - p(0,0)(\lambda + \gamma_2\mu_1)$$

$$E[\mathcal{L}^q_{pull}] = \frac{dP_2(z)}{dz}|_{z=1}, \quad (8)$$

where $\frac{dP_1(z)}{dz}|_{z=1}$ gives the number of elements in push system in PFS. Once, we have the expected number of items in the pull system from Equation (8), using Little's formula [6], we can easily obtain the estimates of average waiting time of the system ($E[W_{pull}]$), and expected number of items ($E[\mathcal{L}^q_{pull}]$) in the pull queue as:

$$E[W^q_{pull}] = E[W_{pull}] - \frac{1}{\mu_2} = \frac{E[\mathcal{L}_{pull}]}{\lambda} - \frac{1}{\mu_2} \quad (9)$$

If K represents the number of items in the push system, then the expected cycle-time of the push system is given by: $\sum_{i=1}^{K} \frac{S_i P_i}{(1-\rho)\mu_1}$. Hence, the expected access-time ($E[T_{hyb-acc}]$) of our hybrid system is given by:

$$E[T_{hyb-acc}] = \sum_{i=1}^{K} S_i P_i + E[W^q_{pull}] \times \sum_{i=k+1}^{D} P_i, \quad (10)$$

where according to the packet-fair-scheduling, $S_i = \frac{\sum_{j=1}^{K}\sqrt{\hat{P}_j}}{\sqrt{\hat{P}_i}}$ and $\hat{P}_i = \frac{P_i}{\sum_{j=1}^{K} P_j}$. The above expression provide an estimate of the average behavior of our hybrid system.

TABLE I
PERFORMANCE COMAPRISON OF SCHEDULING PRINCIPLES

θ	push		pull	
	Flat	PFS	FCFS	MRF
0.50	433.32	403.30	445.54	413.30
0.60	409.94	377.78	423.35	387.78
0.70	381.14	353.23	401.01	372.23
0.80	331.08	323.36	373.31	343.36
0.90	319.96	293.38	341.12	303.38
1.00	278.83	259.95	306.64	278.95
1.10	236.73	229.95	275.55	223.95
1.20	216.69	199.90	242.29	201.90
1.30	193.35	171.04	207.86	191.04

V. SIMULATION EXPERIMENTS

In this section we first evaluate the comparative performance of the different scheduling principles. Subsequently, we perform the experiments to demonstrate the performance efficiency of our hybrid system. In order to compare the performance of our hybrid system, we have chosen our previous hybrid scheduling algorithm [12] as performance bench-marks. The prime goal of the entire scheme is to reduce the expected access time. Before going into the details of the simulation results, we enumerate the assumptions we have used in our simulation.

1) The simulation experiments are evaluated for $D = 1,000$ items. The system performs a push and pull operation in a reciprocal manner, unless the pull queue is empty.
2) In order to remain consistent with the analysis, the arrival and service rates are assumed to obey Poisson distribution. The average value of arrival rate (λ) is taken to be 10 and 20. The average value of service rates (push and pull), μ_1 and μ_2 are assumed to be 1.
3) In order to keep the access probabilities of the items from similar to very skewed, θ is dynamically varied from 0.50 to 1.50.

First we show the performance comparison major push and pull scheduling algorithms. Subsequently, we show the performance efficiency of our newly proposed hybrid scheduling. Table I demonstrates the comparative performance of push and pull scheduling algorithms for 1000 items. It is quite clear that PFS outperforms Flat and MRF outperforms FCFS for push and pull scheduling respectively. This is the prime reason behind our motivation in choosing these two scheduling principles for push and pull-based data transmission respectively.

Figure 3 demonstrate the variation of expected access-time with different values of θ, for arrival-rates of 10 and 20 respectively, in our hybrid scheduling algorithm, for 1000 items. Note that, in both cases, the expected access-time for our new hybrid scheduling is sufficiently lower than the expected access time for existing hybrid scheduling. The prime

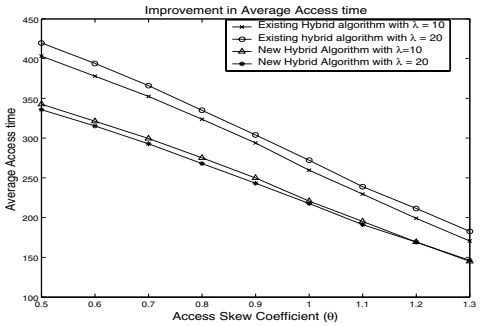

Fig. 3. Improvement in Average Access Time

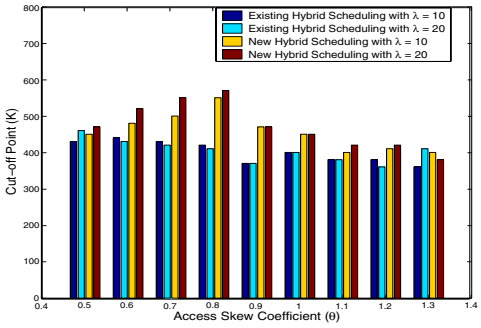

Fig. 4. Dynamics of Cutoff Point

reason behind this lies in the fact that the hybrid scheduling captures the requirement of the system by probabilistically combining push and pull-based scheduling principles. Figure 4 shows that the hybrid scheduling achieves a cut-off point in the range 360–430 and 360–460 respectively for arrival rates of 10 and 20 with 1000 data items. This explains the reason that our hybrid scheduling makes a fair combination of both push and pull systems, which is required to improve the expected access-time. Figure 5 depicts the comparative view of the analytical results with the simulation results, for 1000 data items. The analytical results closely match with the simulation results for expected access time with almost $\sim 90\%$ and $\sim 93\%$ accuracy for $\lambda = 10$ 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%) differences exist because of the assumption of memory-less property associated

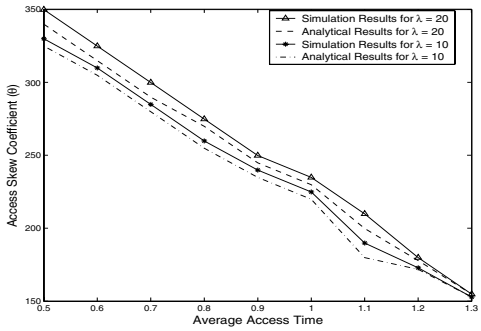

Fig. 5. Comparison of Analytical and Simulation Results

with arrival rates and service times in the system.

VI. Conclusions

In this paper we have proposed a dynamic, hybrid scheduling technique that probabilistically combines push and pull scheduling to achieve an improved access-time. The algorithm takes into consideration the number of items present in the push and pull subsystem and their popularity to perform push and pull operations dynamically in an efficient manner. This push and pull mechanism are guided by PFS and MRF scheduling principles. The cut-off point between push and pull items is chosen in such a way that the expected access time of the hybrid system is minimized. Performance modelling and simulation results points out the improvement in average access time even for highly skewed data items. Our future interests lie in further improving the performance by incorporating the classification of clients according to their probabilities.

References

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

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

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] M. C. Pinotti and N. Saxena. Push less and pull the current highest demaned data item to decrease the waiting time in asymmetric communication environments. In *4th International Workshop on Distributed and Mobile Computing*, (IWDC), Calcutta, India pages 203–213. Springer-Verlag 2002; LNCS 2571, Dec 28-31, 2002.

[13] 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.

[14] A. Seifert and M H Scholl. Processing Read-Only Transactions in Hybrid Data Delivery Environments with Consistency and Currency Guarantees. In *MONET*, Vol. 8, pages 327-342, 2003.

[15] 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.