On-line Balanced K-Channel Data Allocation with Hybrid Schedule per Channel

Navrati Saxena
Department of Informatics and Telecommunication
University of Trento
Trento, Italy
navrati@dit.unitn.it

M. Cristina Pinottti
Department of Math. and Computer Science
University of Perugia
Perugia, Italy
pinotti@unipg.it

ABSTRACT

Broadcast is an efficient and scalable technique for disseminating data over wireless channels to an arbitrary large number of clients. The prime objective of broadcast schedules, both on-line and off-line, is to minimize the average amount of time a client needs to wait before receiving its desired item. In off-line schedules, it is assumed that both the entire set of data items and the demand probability for each data is known in advance. Such schedules are guaranteed to obtain the minimum average waiting time only when a skewed partition of the data among multiple-channels and flat schedules per channel are assumed. On the contrary, the on-line broadcast schedules decide which item to transmit without any a-priori knowledge of either the entire set of data or the data demand probabilities. Although no optimal solutions are known for the latter schedules, the volatility of the set of data items to be transmitted makes on-line schedules much more desirable than its off-line counterparts. In this paper, a new on-line broadcast schedule for broadcast over multiple channels is presented. The strategy partitions the data among the broadcast channels in a balanced way, and adopts a hybrid push-pull broadcast schedule per channel. Simulation experiments point out that the results of our new algorithm outperforms the minimum average waiting time achieved by the optimal off-line schedule (skewed partition of data among channels and flat schedule for channel), and the on-line broadcast schedule based on the square root rule.

Categories and Subject Descriptors
H.4 Information Systems Applications, C.2 Computer Communication Networks

General Terms
Algorithms, Experimentation

Keywords
Broadcast, Hybrid Systems, Wireless Multiple Channels

1. INTRODUCTION

In wireless communication environments a server of a base-station continuously transmits data over a given set over wireless downlink channels. Clients listen to the channels waiting for desired items, and might send requests to the server through an uplink channel, whose capacity is much smaller than the capacity of downlink channels. This brings an inherent asymmetry in the communication environment. Each data item is assumed to be of *unit length*, and requires a specific *time unit* to be transmitted. Every data item is also associated with a *demand probability*, reflecting the popularity of the item among the clients. The server maintains a *broadcast schedule* to decide which item of the data set will be transmitted at any *time unit.* The amount of time units a client waits before obtaining its desired data is called the *item delay.* The *Single Broadcast Problem* is the problem of finding an efficient single broadcast schedule on a particular downlink channel which minimizes the clients' waiting time. Let us first consider the *total push* systems, where the server autonomously decides the broadcast schedule and transmits the data items to all clients listening to a particular channel. For such systems, the measure to minimize is the *Average Expected Delay* (AED), which is the mean item delay over all items, assuming that clients have the same probability to start listening to the broadcast channel at any instant of time. Under such conditions, the simple broadcast strategy is the *Flat* schedule, which transmits the data items one at a time, in a cyclic, round robin fashion, with a prefixed order. The AED performance of the flat schedule is equal to half of the size of the data set to be transmitted. However, this is quite inefficient even for medium size data items. In order to reduce the *AED* of such a naive approach, several solutions of the Single Broadcast Problem have been proposed that exploit the demand probabilities of the data items. Intuitively, a *hot* item, i.e. an item whose demand probability is close to 1, is expected to appear in the broadcast schedule more frequently than a *cold* item, i.e. an item with demand probability close to 0. Indeed, a hot data item should be less delayed than a cold data. The first solution under these assumptions for the Single Broadcast Problem has been given in the context of teletext systems in [3] and analyzed in [1, 4]. It divides data items in groups, with items in the same range of demand probabilities assigned to the same group. The broadcast schedule is then generated taking items from groups with higher demand probabilities more frequently than from groups with lower demand probabilities. In [10], the on-line *Square Root Rule* solution is presented, whose aim is to provide a broadcast schedule where every data item appears with replicas equally spaced in such a way that the distance between the replicas of a certain item is inversely proportional to the square root of its demand probability.

The total push solutions for the Single Broadcast Problem reviewed so far do not explicitly consider the clients' needs, but they do implicitly through the demand probabilities. However, to handle the demand probabilities so that they correctly reflect the clients' status is extremely critical. It might happen that probabilities remain frozen long time and become obsolete, or that they change too frequently and reflect anomalies in the client behavior. Then, an alternative solution is to have the server that listen to explicit requests by the clients. This brings into the scenario, in the other extreme, the *total pull* broadcast schedules, where each data is transmitted only after explicit request by a client. Note that, due to the limited bandwidth of the uplink channel, total pull schedules are only feasible for not highly loaded systems. For pull systems, the *item delay* is measured as the *actual request delay*, averaged over all the requests; i.e., the interval of time between the client's request and the subsequent transmission of the item. For such systems the AED performance does not apply since, for lightly loaded systems, it is not worthy to assume that each item has a client waiting for it at any instant of time. Then, broadcast schedules have to minimize the *Average Access Time* (AAT). It is estimated in two steps: (1) First for a particular data item, the mean delay of actual requests are calculated over the entire set of requests (2) Secondly, the average of these mean delay estimates for every item, is obtained over the entire set of data items. Thus, the Single Broadcast Problem for the pull systems is the problem of finding an efficient schedule which minimizes AAT. Note that such schedules, which depend on the client requests and on the system conditions, must be devised on-line. Many preemptive and non-preemptive pull based algorithms like First Come First Served (FCFS), Most Request First (MRF) and Largest Delay Cost First (LDCF) exist in literature [2, 6, 11].

In spite of all this work, it is worthy to point out that in many realistic contexts the performance of total push systems and that of total pull systems are pretty close [8]. Hence, a hybrid strategy that combines both the push and pull systems appears to be more efficient. The actual essence of such hybrid scheduling lies in the partitioning of entire set of data items into two parts: *push set* and *pull set*. The hot items are kept in the push set, while the cold items in the pull set (transmitting only when requested by any client j). The cut-off point between the push and pull sets is chosen in such a manner that the waiting time of the clients for both push and pull items is minimized. In other words, the goal of hybrid systems is to minimize the AED of the push set as well as the AAT of the pull set [9, 7].

To further improve on the broadcast efficiency in asymmetric communications, one can divide the large bandwidth of the downlink channel in multiple disjoint physical channels. Then, for total push systems, the *Multiple Broadcast Problem* deals with finding the broadcast schedule on a multichannel environment which minimizes the *Multiple Average Expected Delay* (MAED), that is the mean of the Average Expected Delay measured over all channels a client can afford to listen. At the best of our knowledge, only total push schedules for multiple channels have been proposed so far. Such solutions may either transmit all data items on each channel or partition the data items in groups and transmit a group per channel. In the former case, MAED can be scaled up to the number of channels that clients can simultaneously listen by coordinating the solutions for each single channel. In the latter case, clients must afford to listen to all channels, but not necessarily simultaneously. When data items are partitioned among the channels, and the flat schedule is adopted to broadcast the subset of data assigned to each channel, the Multiple Broadcast Problem boils down to the *Allocation Problem* introduced in [5, 13]. For such a problem, the solution that minimizes MAED can be found in time polynomial in both the size of data and the number of channels [13, 12, 5]. However, the optimal schedule can only be computed off-line because it requires in input the data sorted by decreasing demand probabilities. Moreover, the strategy is not dynamic and the optimal solution has to be recomputed from scratch when the data demand probabilities change.

In this paper, a new on-line hybrid solution for the Multiple Broadcast Problem is investigated. The new strategy first partitions the data items among multiple channels in a balanced way. Then, a hybrid push-pull schedule is adopted for each single channel. Clients may request desired data through the uplink and go to listen to the channel where the data will be transmitted. In each channel, the push and pull sets are served in an interleaved way: one unit of time is dedicated to an item belonging to the push set; and one to an item of the pull set, if there are pending client-requests not yet served. The push set is served according to a flat schedule, while the pull set according to the Most Request First policy. No complete knowledge is required in advance of the entire data set or of the demand probabilities, and the schedule is designed on-line. The rest of the paper is so organized. Section 2 states the Multiple Broadcast Problem foundations, and review previous results under an unifying perspective. In particular, it is shown how to derive from the general expression of $MAED$, the MAED formulas studied in [10] and [13]. Section 3 proposes the new On-line Balanced K-Channel Data Allocation with Hybrid Schedule per Channel algorithm. The experimental tests in Section 4 reveal that the new solution is superior to both the optimal skewed allocation solution [13] and the on-line solution based on Square Root Rule [10]. Finally Section 5 concludes the paper offering discussions on future research directions.

2. DEFINITIONS AND METRICS

Let $D = \{1, 2, \ldots, N\}$ be a set of N data items of unit length, and let each item i be characterized by a demand probability p_i. To start, consider a system with a single broadcast channel. A *broadcast schedule* S of period P is an ordered sequence of P data items selected from the set D. Note that if S is cyclic then P is a positive integer, otherwise $P \to \infty$. Position t of S indicates the item of D that is broadcast at time $\tau \equiv t \bmod P$. The same item can be replicated in S. The *average spacing* between two consecutive instances of the same item i in S is termed s_i.

Note that if i appears only once in S, then $s_i = P$. For total push systems, the *expected item delay* t_i for item i on S is defined as the average time a client waits before receiving i, assuming that, at any instant of time, clients start to listen with the same probability. Hence, $t_i = \frac{s_i}{2}$, for $1 \leq i \leq N$. Thus, the *Average Expected Delay* is given by:

$$AED(D) = \sum_{i=1}^{N} t_i p_i = \frac{1}{2} \sum_{i=1}^{N} s_i p_i \qquad (1)$$

is the average over all items of D of their delay.

For the total pull systems, let $\delta_{i,r}$ be the actual delay between the request r for item i and the transmission time of item i. Let R_i and $\#_i$ respectively denote the set of requests for item i and its size. Then the *average item delay* is defined as $\frac{\sum_{r \in R_i} \delta_{i,r}}{\#_i}$ and the *Average Access Time*

$$AAT(D) = \sum_{i=1}^{N} \frac{\sum_{r \in R_i} \delta_{i,r}}{\#_i} p_i \qquad (2)$$

is the average over all items of D of their average item delay.

For the hybrid push-pull systems, let $D = \Pi \cup \Delta$, where Π and Δ are the push and pull sets, respectively. Then, their performance is measured as the *Hybrid Time*, represented by:

$$HT(D) = AED(\Pi) + AAT(\Delta) \qquad (3)$$

Finally, the Single Broadcast Problem is defined as the problem of finding the broadcast schedule S which minimizes Equations 1, 2, and 3 for push, pull and hybrid systems, respectively. Note that, for total push schedules, particular assumptions lead to simplified formulations. For example, for a flat schedule F, since $s_i = N$ for $1 \leq i \leq N$ and $\sum_{i=1}^{N} p_i = 1$, it holds that $AED_F(D) = \frac{N}{2}$

Consider now a system with K broadcast channels. A *multiple broadcast schedule* M consists of K single broadcast schedules, one per channel. For total push systems, the average delay t_i for item i is defined exactly as in the single channel environment, except that two item occurrences are considered consecutive if they happen to be close in the time, irrespective of on which channels they appear. Specifically, for a client listening simultaneously to the first j channels, two occurrences of i are consecutive and they are s_i apart if an occurrence of item i appears at time τ_i on channel j_1, the subsequent earliest occurrence of i appears at time τ_{i+s_i} on channel j_2, with $1 \leq j_1 \leq j_2 \leq j$, and no other occurrence appears in any other channel between 1 and j at the instants of time $\tau_{i+1}, \ldots, \tau_{i+s_i-1}$. Now, let $AED^j(D)$ denote the AED experienced by a client listening to the first j channels. It is easy to see that a lower bound for $AED^j(D)$ is $\frac{AED(D)}{j}$. Note that, such a lower bound holds either when all data items are transmitted on each channel or when only a group of the data items is transmitted on each channel. Finally, the *Multiple Average Expected Delay MAED* is defined as the AED averaged over all the subsets of channels that clients can afford to read. To simplify, let clients listen only to consecutive subsets of channels, starting from channel 1. Thus, if a client listen to j channels, with $j > 1$, it will listen to channels $1, 2, \ldots, j$. Denoting by π_j the probability that clients listen to j channels, and assuming that $\sum_{j=1}^{K} \pi_j = 1$,

$$MAED(D) = \sum_{j=1}^{K} AED^j(D) \pi_j \qquad (4)$$

Thus, MAED = AED when $K = 1$. As for AED, also simplified expressions of MAED hold.

MAED boils down to a much simpler expression when Skew allocation among channels and Flat schedules [13, 5], briefly *SF*-schedule, is adopted. Indeed, assume that the data items are partitioned into K groups $G_1, G_2, \ldots, G_K$, where the group G_j consists of the N_j data items transmitted by a flat schedule on channel j. Since each item is transmitted only by a channel, MAED is bounded by a constant only if clients listen to all channels. Hence, assuming $\pi_K = 1$ and $\pi_j = 0$, for any $1 \leq j \leq K-1$, it is easy to see that, for any skewed allocation,

$$MAED_{\mathrm{SF}}(D) \;=\; \frac{1}{2} \sum_{j=1}^{K} \left(N_j \sum_{i \in G_j} p_i \right) \qquad (5)$$

Hence, the Allocation Problem, proposed in [13, 5], consists in finding the values $N_1, N_2, \ldots, N_K$ such that Equation 5 is minimized. It is worthy to note that the optimal solution for the *SF*-schedule can be found applying a dynamic programming strategy in $O(NK \log N)$ time, as shown in [5]. Nonetheless, since the Allocation Problem is a special case of the Multiple Broadcast Problem, it is not known how far is the optimal MAED of the Allocation Problem from the optimal solution of the Multiple Broadcast Problem, even when it is considered restricted to the push systems. In conclusion, let us point out, that although *AAT* and *HT* performance measures can be generalized to the case of multiple channels, we are not aware of solutions already proposed in literature for the Multiple Broadcast problem for total pull or hybrid systems.

3. MULTI-CHANNEL HYBRID SCHEDULE

The above discussion suggests that many different schedules for the Multiple Broadcast Problem can be obtained by combining different data allocation strategies with different schedule strategies for single channels. The solution proposed in this paper for N data items and K channels combines a balanced allocation of data among channels with hybrid push-pull schedule per each single channel. The hybrid push-pull strategy guarantees that our solution adapts easily to changes of item demand-probability, while the balanced data allocation provides an easy way to incorporate new data items, without any data pre-processing. Moreover, since no more than $\lceil N/K \rceil$ items are assigned to each channel, by the flat schedule, MAED cannot go beyond $\lceil \frac{N}{2K} \rceil$. Finally, since each client knows in advance the channel on which the desired item will be transmitted, it can listen only to a channel per time.

The new *Balanced Allocation with Hybrid Schedule per Channel Schedule*, briefly *BH*-Schedule, first assigns to channel j, $1 \leq j \leq K$ the group of data items G_j, which consists of the N_j items $\{i|(i-1) \bmod K = j-1\}$, where

$$N_j = \begin{cases} \lceil \frac{N}{K} \rceil & \text{if } 1 \le j \le (N \bmod K), \\ \lfloor \frac{N}{K} \rfloor & \text{if } (N \bmod K) + 1 \le j \le K. \end{cases}$$

Then, for each channel j, the group G_j is partitioned in two sets: the *push-set* Π_j, whose items $\{1, \ldots, k_j\}$ will be broadcast according to a flat schedule and the *pull-set* Δ_j, whose items will be sent *on-demand.* Let k_j be termed the cut-off point for channel j. The BH-Schedule alternates between the transmission of one item extracted from the push-set and the transmission of one item from the pull-set. At each pull turn, the item to be sent on demand is the item most requested so far by clients. Note that the push set may gain several consecutive turns if there are no pending requests for items of the pull set.

A client, desiring to receive item i, sends to the server an explicit request through the uplink if $i > k_j$. Then, it goes to listen to channel $j = (i-1) \bmod K + 1$ to which item i has been assigned by the balanced allocation algorithm, and waits until i is transmitted. For each channel j, the server stores in F_j the flat schedule of Π_j, whose current length is k_j. For each item i of the pull set, the server maintains the number $\#_i$ of requests received between two consecutive transmissions of that item in a max-heap H_j. The requests are checked before each push turn. The item sent at the pull turn is the one stored in the heap root, that is the item that has received so far the largest number of requests. After the pull transmission of item i, $\#_i$ is always reset to 0. Note that each decision is made on-line and that it takes $O(\log \Delta_j)$ time. Finally, in order to have a schedule adaptive to noticeable changes of the demand probabilities, a mechanism for dynamically varying the push and pull sets is given based on the threshold σ. When item i is broadcast at the pull turn, if $\#_i > \sigma$, i is inserted in the push set as the last item of the flat schedule. Although the push set initially consists of consecutive items, it may become fragmented. Then, the client needs more information to learn to which set the desired item belongs. The server might supply the index of the changes occurred at the push sets in a compressed form along with each single data, and periodically defragmentation policies might be applied to globally renumber the data items.

The $MAED$ performance of the BH-schedule is

$$MAED_{BH}(D) = \gamma \sum_{j=1}^{K} \sum_{i=1}^{k_j} \frac{k_j}{2} p_i + \sum_{j=1}^{K} \sum_{i=k_j+1}^{N_j} \frac{\sum_{r \in R_i} \delta_{r,i}}{\#_i} p_i$$

where γ is the *interleaving coefficient* and varies from 1, when only push turn occur (i.e., system light loaded), to 2, when every push turn is followed by a pull turn (i.e., system high loaded).

4. SIMULATION RESULTS

In this section, simulation experiments are reported in order to discuss the performance of our multi-channel scheduling strategies. Before going into the details of simulation results, the major assumptions and parameters used for our experiments are summarized.

1. The simulation experiments are evaluated for a total number of data items $N = 2000$.
2. The request arrival time is assumed to obey Poisson distribution with mean $\lambda = 10$, which simulates a middle load of the system. The item requests follow the Zipf distribution whose skew coefficient is θ; i.e., $p_i = \frac{(1/i)^\theta}{\sum_{i=1}^{N}(1/i)^\theta}$, for $1 \le i \le N$. The average service time for every request is assumed to be 1.
3. The number of channels varies between 2–4.
4. The demand probabilities follow the Zipf distribution with θ dynamically varied from 0.30 to 1.30.

Finally, observe that all the experiments involving BH schedule were executed 10 times, and the performance average has been reported.

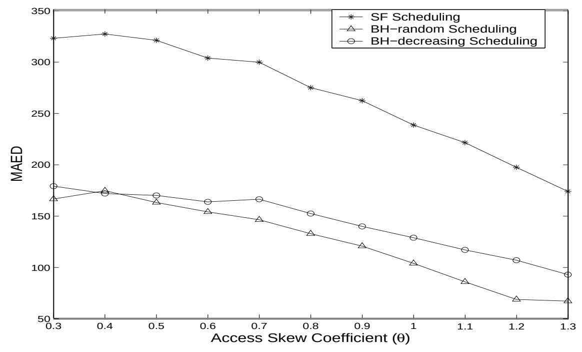

Figure 1: $MAED$ performances of the new multi-channel hybrid schedules vs the $MAED$ of the SF schedule.

Figure 1 demonstrates the performance efficiency of the BH schedule over the SF schedule. In addition to the BH schedule as described in Section 3, a variant that maintains in each channel the item of the push set sorted by decreasing demand probabilities is studied. Note that the ordering is local into each channel. In this way, the push set is initialized with hot items, and similarly the pull set with the cold items. So, from now on, let *BH-random* denote the basic hybrid schedule, while *BH-decreasing* the new variant. Both the BH schedules result in an improvement of almost half of the performance of the SF schedule. Figure 2 shows the gain achieved by BH schedules on different numbers of channels, for $\theta = 0.50$ or $\theta = 1.00$. Even for different number of channels the BH-schedules achieve almost half of the $MAED$ measure in comparison to the SF-schedule.

We have also taken a look into the distribution of items among different channels. While the Skewed Allocation is twisted, different sizes of the push sets of the BH schedule can be selected initially. Figure 3 and 4 show that, for a fixed value of the skew parameter θ of the Zipf distribution, and assuming all the push sets initially empty, when the BH schedule reaches a steady state (i.e., when the threshold mechanism does not move any items), the sizes of the push sets are skewed at least as much as that of the groups of the SF schedule when $\theta \le 0.4$ and much more skewed for larger values of θ.

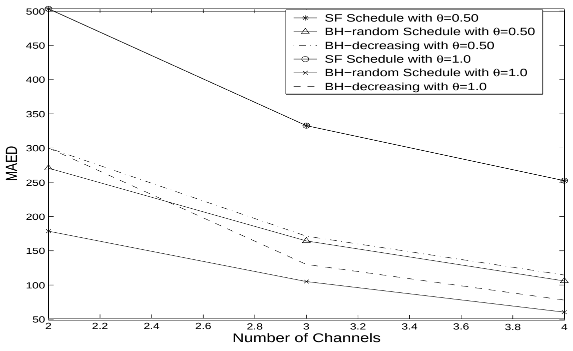

Figure 2: $MAED$ performances of the SF, BH-random and BH-decreasing schedules for $K = 2, 3$ and 4 channels

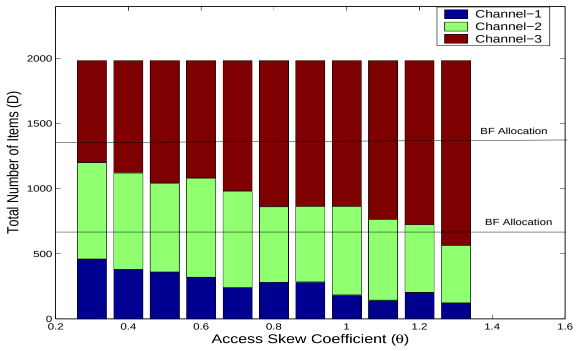

Figure 3: Size of the channel groups of the SF schedule when $K = 3$.

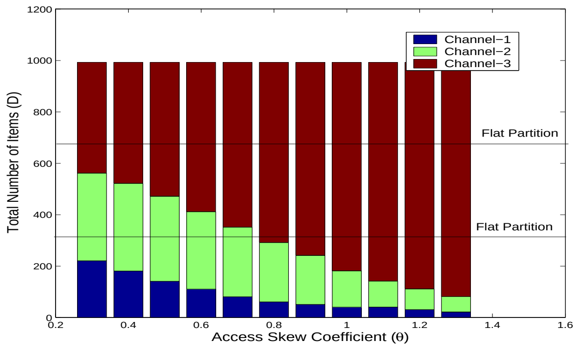

Figure 4: Size of the push sets of the BH schedule when $K = 3$.

5. CONCLUSION

In this paper we have investigated into the multi-channel scheduling strategies. We have presented a Balanced K-channel data allocation with Hybrid schedule that outperforms the well known Skewed Allocation with Flat Broadcast [13] and the Square Root Rule Broadcast [10]. Our solution is adaptive to changes in both number of items and demand probabilities, and fully on-line. Our future interest lies in the study of other combined strategies for the multiple channel environment in the perspective of optimally solving the Multiple Broadcast Problem for item of unit length.

6. REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: data management for asymmetric communication environments. In *Proc. of ACM SIGMOD*, pages 199–210.

[2] D. Aksoy and M. Franklin. Rxw: A scheduling approach for large scale on-demand data broadcast. *IEEE/ACM Transactions on Networking*, 7(6):846–860.

[3] M.H. Ammar and J.W. Wong. The design of teletext broadcast cycles. *Performance Evaluation*, 5(4):235–242.

[4] A. Bar-Noy, R. Bhatia, J.S. Naor, and B.Schieber. Minimizing service and operation costs of periodic scheduling. In *Ninth ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 11–20.

[5] A. A. Bertossi, M. C. Pinotti, S. Ramaprasad, R. Rizzi, and M. V. S. Shashanka. Optimal multi-channel data allocation with flat broadcast per channel. In *Proc. of IEEE IPDPS*, 2004.

[6] W. Ni, Q. Fang, and S. V. Vrbsky. A lazy data request approach for on-demand data broadcasting. In *Proc. of* 23^{rd} *Intl. Conf. on distributed Computing Systems Workshops (ICDCSW)*, pages 790–796.

[7] 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. In *4th International Workshop on Distributed and Mobile Computing, Springer Verlag (LNCS) (IWDC)*, pages 203–213.

[8] C-J. Su, L. Tassiulas, and V. J. Tsotras. Broadcast scheduling for information distribution. *ACM/Kluwer Wireless networks (WINET)*, 5.

[9] W. Sun, W. Shi, B. Shi, and Y. Yu. A cost-efficient scheduling algorithm for on-demand broadcasts. *ACM/Kluwer Wireless Networks*, 9.

[10] N. Vaidya and S. Hameed. Log time algorithms for scheduling single and multiple channel data broadcast. In *Proc. Third ACM-IEEE Conf. on Mobile Computing and Networking (MOBICOM)*, pages 90–99.

[11] Y. Wu and G. Cao. Stretch-optimal scheduling for on-demand data broadcasts. In *Proceedings of Tenth International Conference on Computer Communications and Networks*, pages 500–504.

[12] W. G. Yee and S. B. Navathe. Efficient data access to multi-channel broadcast programs. In *Proc. of ACM Conf. on Intl. Knowledge Management, (CIIKM)*, pages 153–160.

[13] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine. Efficient data allocation over multiple channels at broadcast servers. *IEEE Trans. on Computers*, 51(10):1231–1236.