BANDWIDTH ALLOCATION IN A WIRELESS BROADCAST SYSTEM

ASLIHAN CELIK and STEVE NAHMIAS

Department of Operations and Management Information Systems
Santa Clara University, 500 EL Camino Real, Santa Clara
CA 95053, USA

RHONDA RIGHTER*

Department of Industrial Engineering and Operations Research
University of California, Berkeley
CA 94720, USA
RRighter@IEOR.Berkeley.edu

We develop a model for broadcasting data to wireless appliances. Any data item may either be “pushed”, or broadcast without waiting for requests for the item, or it may be “pulled”, or broadcast in response to a request. Pushing items is cheaper than handling the individual requests, but if pushed items are not wanted, the bandwidth for broadcasting them is wasted. We determine the optimal allocation of bandwidth between pushed and pulled items.

Keywords: Wireless broadcasting; publish/subscribe; broadcast scheduling; optimization.

1. Introduction

Consumers are increasingly relying on wireless appliances, such as PDA’s and cellular telephones, to receive data on a real-time basis. Subscribers to data services may access stock prices, weather information, or vital documents. Software vendors are developing applications that facilitate mobile access to data. In these applications, clients request data from an application server. In some cases, the data changes rapidly and client requests for the same item could be frequent. Moreover, there may be “hot” items that many different users want to receive. To efficiently use bandwidth, data items are broadcast so that multiple users may access the same data item.

Processing requests for items can be expensive and time consuming. In order to alleviate the load on the server, data providers may consider a “push” strategy. In this scenario, the server pushes (by broadcasting or multicasting) information to

*Corresponding author.
a shared channel, without waiting for requests. If the information required by the client is not part of the broadcast, the client must make a request to the server. These requests are broadcast on a “pull” basis and at a higher cost. The strategic issue addressed here is to determine an optimal allocation of bandwidth between “pushed” and “pulled” broadcasts.

We develop a mathematical model to capture key features of the bandwidth allocation problem. We give the stochastic dynamic program to determine the optimal broadcast configuration, and derive properties that enable us to determine the configuration efficiently.

2. An Architecture for Data Broadcasts in a Wireless Network

We consider a cellular network composed of mobile clients, base stations, and an information server. The network is geographically divided into smaller cells. This type of architecture is pictured in Fig. 1. A wireless cell is a circular area whose radius ranges from tens of meters to several kilometers. Every base station is responsible for facilitating communication of the mobile clients in its cell with the rest of the network. When a mobile client moves from the influence of one base station to another, the two base stations in question must complete a process called a “handoff” to register the client with the new cell. Note that the network has several fixed (as opposed to mobile) hosts, and a wired network connects the fixed hosts and the base stations.

Fig. 1. Basic wireless architecture.
In this system, mobile clients wish to receive the most recent set of data items. The information server is responsible for preparing the collection of data items in the form of a broadcast and relaying it to the base stations. The base stations then transmit the broadcast to the mobile clients in their areas. When the broadcast is completed, a new broadcast with updated data is prepared and sent. The broadcast may be encrypted to be sure only valid subscribers receive the information. The structure of a broadcast is shown schematically in Fig. 2.

A broadcast consists of an index segment (or table of contents) and a collection of data blocks. When a client receives the broadcast, the index is first downloaded so the client knows the data items that are included in that broadcast. If the client requires data not contained in the broadcast, that client submits a request to be included in a future broadcast. Thus, each broadcast consists of items pushed by the server as well as items previously requested.

### 3. Related Literature

Much work has been done on scheduling data broadcast in wireless networks. We can classify the existing research based on the three architectures proposed by Wong:\(^{12}\) (1) one-way (push-based) broadcast, (2) two-way interaction (pull-based), and (3) hybrid one-way broadcast/two-way interaction. See Katzela and Naghshineh\(^ {13}\) for a survey, and Zomaya,\(^ {14}\) for a genetic-algorithm approach.

Acharya, Alonso, Franklin and Zdonik\(^ {1}\) introduced the Broadcast Disk approach for pushing data items to the clients. Essentially, the broadcast is scheduled by “spinning” disks that hold data items. Then these disks are multiplexed on the broadcast channel. Therefore, depending on the size and the rotational speed of the disks, the data items may be repeated in the broadcast. Jiang and Vaidya\(^ {8}\) suggest an optimality condition for a broadcast schedule that minimizes mean waiting time of the users and maximizes the percentage of requests served in a pure push scenario. Su, Tassiulas and Tsotras\(^ {11}\) formulate the broadcast schedule as a deterministic dynamic optimization problem that minimizes the average response time. They demonstrate by a numerical study that as the request generation rate increases, the achievable performance of the pull- and push-based systems becomes almost identical.

Pure pull-based scheduling has also received interest from researchers. Aksoy and Franklin\(^ {3}\) propose an algorithm to prioritize data items based on the number of requests and how long since their last broadcast. A similar measure, called the *Ignore Factor*, was proposed by Datta, VanderMeer, Celik and Kumar.\(^ {5}\)
Although much work has been devoted to pure push- and pull-based systems, there is an increasing interest in hybrid broadcasting systems. Research in hybrid broadcasting was initiated by Acharya, Franklin and Zdonik. They explore efficient ways of combining push and pull systems by integrating a back channel for the clients to send messages to the server. Their performance metric is the response time (i.e. the delay until a client obtains a needed data item). They observe that under a heavily loaded system, pure push is preferable. If the system load is very light, then pure pull performs better. The hybrid approach, however, performs more uniformly over the entire load space.

Gao, Das and Pinotti propose a similar approach. The authors divide the set of data items into more popular and less popular items, and forcibly include the popular items into the automatic broadcast. The less popular items are sent as soon as clients request them and are interleaved with the push-based scheduled items. The authors use an additional index channel to transmit the position of the data items in the broadcast. They derive the cut-off point for deciding which items to place in the broadcast so that the queue of outstanding requests is stable, by using the access probabilities of the data items. They are primarily concerned with scheduling the items to minimize the mean access time.

Lee, Hu and Lee consider the problem of dynamically assigning channels for broadcast or on-demand services based on system workload, assuming that users may request data items from the server, including items that are to be broadcast. They are interested in minimizing the mean access time for the data items, using light and heavy traffic approximations based on queueing models. They assume that all on-demand requests must be satisfied individually, even if there are multiple requests.

We propose a hybrid system that prepares a push-based broadcast without consulting the clients, yet accepts requests from the clients whose data items are not included in the broadcast. All on-demand data items that are requested in our model are broadcast, so that broadcasting the item once satisfies multiple requests. Unlike prior work that focuses on minimizing access times for the clients, we address the problem from the point of view of the server (the mobile support station), and obtain an allocation that minimizes the cost of providing the service.

4. Wireless Broadcast Model

There are $n$ items (or information packets or pages) of equal size available to broadcast, and time is slotted such that the length of a slot equals the length of time to broadcast an item on a channel. Broadcasts are done in cycles, where the cycle length is $M$ slots, $M < n$. For each cycle the probability that the $i$th item will be requested (at least once) is $p_i$, where $p_1 \geq p_2 \geq \cdots \geq p_n > 0$, and where different items are requested independently. A subset of the slots, of size $l \leq M$, will be used for push-based broadcast, such that $l$ items are automatically broadcast, without requiring requests from users. Other requested items may be broadcast during the remaining $M - l$ slots, i.e. these slots are reserved for pull-based, or on-demand,
broadcast. For convenience, we assume a single channel, and that a cycle consists of \( M - l \) pull-based slots followed by \( l \) push-based slots, but our model also applies to an arbitrary number of channels with a total of \( M \) slots in the cycles for all channels, and the \( l \) push-based slots may be distributed arbitrarily among the channels and within the cycles. During the first \( M - l \) slots of a cycle, items that are not part of the push-based broadcast and that were requested during the last cycle are broadcast; the remaining \( l \) slots contain the push-based items. If fewer than \( M - l \) distinct non-push-based items were requested, the remaining slots are used for some other independent application. If more than \( M - l \) distinct non-push-based items were requested, the excess are broadcast on another, more expensive, channel. At the beginning of a cycle, a table of contents for the cycle is given. Note that we can think of the system as providing batch service, so that when an item is broadcast, all outstanding requests for that item are served at once. For this reason, for each item \( i \) we need only the probability that it is requested at least once during the cycle, \( p_i \), and we do not need to worry about the entire distribution of the number of requests for the item.

The cost of the bandwidth is \( c_0 \) per slot, and is paid even if slots are unused for this application. We assume that broadcasting an item automatically on a push basis has negligible additional cost over the bandwidth cost, so that it is also \( c_0 \) per item. This cost is paid for the item regardless of the number of requests for the item. The cost of processing and broadcasting a requested non-pushed item is \( c_1 > c_0 \), where this cost is incurred only once for the item even if it is requested by more than one user. (Note that the item is only broadcast once, so we assume extra requests for the same item incur negligible additional cost.) The cost of processing a requested item that cannot be broadcast during the cycle is \( c_2 > c_1 \), where this cost is also incurred only once for the item even when there are multiple requests. We assume the “extra” requests are broadcast on an alternate channel. Our problem is to chose \( l \) to minimize the total expected cost per cycle. Without loss of generality we may assume that \( c_0 = 0 \); subtract \( c_0 \) from all the costs per slot (or item); then add \( M c_0 \) back after optimizing. There is obviously a trade-off in the bandwidth allocation problem: broadcasting on a push basis is cheaper, but some bandwidth may be wasted on items that are not requested at the expense of not being able to broadcast other requested items.

For our cost model it does not matter which requested items are broadcast on the main channel when the number of requested items exceeds \( M - l \). A reasonable choice would be to broadcast the most frequently requested items because this will create higher user satisfaction.

5. Optimal Bandwidth Allocation

First let us note the intuitively obvious result that the most popular items should be the ones that are broadcast on a push basis. The easy proof is omitted. Recall that for two random variables \( X \) and \( Y \), \( X \) is stochastically smaller than \( Y \) if \( P(X > x) \leq P(Y > x) \) for all \( x \).
Lemma 5.1. For a fixed $l$, $0 \leq l \leq m$, assigning items $1, \ldots, l$ for push-based broadcast stochastically minimizes the cost per cycle.

Note that this is a strong result. Not only does assigning the most popular items for push-based broadcast minimize the mean cost, it also minimizes the probability that the cost exceeds any given fixed cost level.

We will need the following definitions to formulate our problem as a dynamic program.

c_1 = \text{the cost of processing and broadcasting a requested non-pushed item} \\
c_2 = \text{the cost of processing a requested item that cannot be broadcast during the cycle} \\
\Delta c = c_2 - c_1 \\
n = \text{the total number of items} \\
M = \text{the total number of slots in a cycle} \\
p = (p_1, p_2, \ldots, p_n) = \text{the vector of probabilities of the items being requested at least once during the cycle} \\
p_i = \text{the } i\text{th largest } p_k \\
p_{(i)} = p_{n-i} = \text{the } i\text{th smallest } p_k \\
p_i^l = (p_1, \ldots, p_{i-1}, p_i+1, \ldots, p_n) \\
p_{(i)}^l = (p_{n-i+1}, p_{n-i+2}, \ldots, p_n) = (p_{(i)}, p_{(i-1)}, \ldots, p_{(1)}) \\
C_m(p) = \text{the total cost for a cycle under the optimal allocation when there are } m \text{ slots remaining to be allocated and the remaining items have probabilities } p \\
V_m(p) = \mathbb{E}[C_m(p)] = \text{the optimal value function} \\
W_m(p) = \text{the expected cost when it has been decided to fill the remaining } m \text{ slots only by request (so none of the remaining items will be pushed)} \\
Q(k|p) = \text{the probability that there are } k \text{ distinct requested (non-pushed) items for a set of items with probabilities } p \\
P_i(k) = \text{the probability that at least } k \text{ items are requested among the last } i \text{ items, i.e. those with probabilities } p_i, \\
l = \text{the number of slots to be used for push-based broadcast} = \text{the number of items to be pushed} \\
s = M - l = \text{the number of slots to be used for pull-based broadcast} \\
r = n - l = \text{the number of items to be broadcast on a pull basis.}

Note that $V_{m-1}(p^1)$ is the optimal value function given that it has been decided to push at least one item, since from our lemma above, item 1 will be pushed, so

$V_m(p) = \min\{V_{m-1}(p^1) \colon W_m(p)\} = \min_{l=0, \ldots, m}\{W_{m-l}(p_{n-l})\}$

$V_0(p) = W_0(p)$. 

Now let us determine $W_m(p)$. If $m$ slots are used for pull-based requests, and the number of distinct requested (non-pushed) items is $k \leq m$, then the cost for each
requested item is \( c_1 \). If \( k > m \), then \( m \) requested items can be broadcast with cost \( c_1 \) each, and the remaining \( k - m \) items will incur cost \( c_2 \) each. Thus,

\[
W_m(p) = c_1 \sum_{k=1}^{m} kQ(k|p) + \sum_{k=m+1}^{n} [c_1m + c_2(k - m)]Q(k|p).
\]

The following lemma is easy to prove. The proof of Lemma 5.3 is given in the Appendix.

**Lemma 5.2.** \( C_m(p) \) is stochastically decreasing in \( m \), stochastically increasing in \( p_i \) for all \( i \), and stochastically increasing in \( n \) (meaning new \( p_i \)’s are added to the vector \( p \)).

**Lemma 5.3.**

\[
W_m(p) - W_{m-1}(p^1) = c_1p_1 - \Delta c(1 - p_1)P_{n-1}(m).
\]

We can gain some intuition for the difference between reserving \( m \) and \( m - 1 \) slots for requests in Lemma 5.3 as follows. If item 1 is requested, then the other items will have the same cost for both policies, but item 1 will cause an additional cost of \( c_1 \) when \( m \) rather than \( m - 1 \) slots are reserved. On the other hand, if item 1 is not requested, then if \( m \) or more of the other items are requested, one more of them can be broadcast in the cycle when \( m \) rather than \( m - 1 \) slots are reserved, resulting in a savings of \( \Delta c = c_2 - c_1 \).

From Lemma 5.3, for any \( m \), we will prefer to reserve all \( m \) remaining slots for requests, rather than assigning item 1 to a slot and reserving the remaining \( m - 1 \) slots for requests, if

\[
p_1 < \frac{\Delta cP_{n-1}(m)}{c_1 + \Delta cP_{n-1}(m)}.
\]

Let us now consider, for \( M \) available slots, whether we prefer to assign the first \( l < M \) items to slots for push-based broadcast, and reserve the remaining \( M - l \) slots for requests of the remaining \( n - l \) items, or to push the first \( l - 1 \) items, \( l = 1, \ldots, M \). We have from Lemma 5.3 (where now we have \( p_1 \) taking the place of \( p \)), that if

\[
p_i < f(l) := \frac{\Delta cP_{n-1}(M - l + 1)}{c_1 + \Delta cP_{n-1}(M - l + 1)} = \frac{\Delta cP_{s}(s + 1)}{c_1 + \Delta cP_{s}(s + 1)}
\]

then we prefer assigning \( l - 1 \) items to slots rather than \( l \). Since \( P_i(k) < P_i(k - 1) \) for all \( i \) and \( k \) (in order to have at least \( k \) requests among the last \( i + 1 \) items, we have to have at least \( k - 1 \) requests among the last \( i \) items) we have that \( f(l) \) is increasing in \( l \). Therefore, because \( p_i \) is decreasing in \( l \), we have that \( p_i < f(l) \) implies \( p_{i+1} < f(l + 1) \), and indeed \( p_{i+k} < f(l + k) \) for \( k = 1, \ldots, M \). Thus, if we prefer assigning \( l - 1 \) items to slots rather than assigning \( l \) items to slots, then we also prefer assigning \( l - 1 \) items to assigning any more than \( l \) items, and the optimal number of items to assign to slots is the smallest \( l \) such that \( p_{i+1} < f(l + 1) \), \( l = 0, \ldots, M - 1 \). This requires computing \( f(1), f(2), \ldots f(l + 1) \), where \( l \) is the
optimal number of items to assign to slots. Summarizing, we have the theorem below. Note that we have essentially shown that $W_{M-l}(p_{n-l})$ is convex in $l$.

**Theorem 5.4.** The optimal allocation of slots to push-based items, $l^*$, is the smallest $l$ such that $p_{l+1} < f(l+1)$, $l = 0, \ldots, M - 1$.

Rather than using Theorem 5.4 directly, we can find the optimal $l$ by first computing $f(M)$ and working backwards, so the optimal $l$ is the largest $l$ such that $p_l \geq f(l)$. The latter may be computationally easier, because we can use the probability calculations that we need for $f(l)$ to compute $f(l+1)$, using

$$P_t(j+1) = p_{(i)}P_{t-1}(j) + (1 - p_{(i)})P_{t-1}(j+1) \quad (1)$$

$$P_t(i) = \prod_{k=1}^{i} p_{(k)} \quad (2)$$

$$P_t(1) = 1 - \prod_{k=1}^{i} (1 - p_{(k)}) \quad (3)$$

and $P_t(k) = 0$ for $k > i$. Algorithm A below will give us the optimal allocation. To make the algorithm a little clearer, let us consider the case for $M = 10$ slots and $n = 15$ items. Then in step 1 of the algorithm, for $l = M = 10$, $r = n - M = 5$, and $s = M - l = 0$, we compute $P_5(1)$ from Eq. (3) and use this to compute $f(10)$. We also compute $P_2(1)$, $P_3(1)$, $P_4(1)$, and $P_5(1)$ from Eq. (3) for use in the next step. In the first iteration of step 2, for $l = M - 1 = 9$, $r = 6$, $s = 1$, we compute $P_5(2)$ from Eq. (2), and then compute $P_3(2)$, $P_4(2)$, $P_5(2)$, and $P_6(2)$ from Eq. (1). Then $f(9)$ is computed from $P_6(2)$. In the next iteration of step 2, for $l = 8$, $r = 7$, $s = 2$, we compute $P_3(3)$, $P_4(3)$, $P_5(3)$, $P_6(3)$, and $P_7(3)$, and so on until $p_l \geq f(l)$.

**Algorithm A**

1. Set $l = M$, $r = n - M$, $s = 0$. For $k = 2, \ldots, r$, compute

$$P_k(1) = 1 - \prod_{i=1}^{k} (1 - p_{(i)}).$$

Compute

$$f(M) = \frac{\Delta c P_r(1)}{c_1 + \Delta c P_r(1)}.$$  

2. Do while $p_l < f(l)$ and $l \geq 1$.

Set $l = l - 1$, $r = r + 1$, $s = s + 1$. Compute

$$P_{s+1}(s + 1) = \prod_{i=1}^{s+1} p_{(i)}.$$
For $k = s + 2, \ldots, r$, compute

$$P_k(s + 1) = p_{(k)}P_{k-1}(s) + (1 - p_{(k)})P_{k-1}(s + 1).$$

Compute

$$f(l) = \frac{\Delta c P_r(s + 1)}{c_1 + \Delta c P_r(s + 1)}.$$

3. It is optimal to push the first $l$ items, $l^* = l$.

We now consider the effects of changes in the parameters on the optimal bandwidth allocation. In particular, the amount of bandwidth that should be allocated to push-based items is increasing in the cost of processing and broadcasting requested items, and decreasing in the cost of processing items that cannot be broadcast at all and in the total number of items. The proofs are easy, with the corollary following from the lemma and the fact that $l^*$ is the largest $l$ such that $p_l \geq f(l)$.

**Lemma 5.5.** The function

$$f(l) = \frac{\Delta c P_{n-l}(M - l + 1)}{c_1 + \Delta c P_{n-l}(M - l + 1)}$$

is decreasing in $c_1$ and increasing in $c_2$, $n$, and $p_{n-l}$.

**Corollary 5.6.** The optimal allocation of bandwidth to push-based items, $l^*$, is increasing in $c_1$ and decreasing in $c_2$ and $n$.

6. Conclusion

In this paper, we have developed an approach based on stochastic dynamic programming to determine the optimal configuration of a wireless broadcast system in which items are either pushed by the broadcaster or pulled (i.e. requested) by the user. We assume pushed items are less expensive than pulled items, and determine the optimal configuration of pushed items based on minimizing the expected cost incurred for each broadcast cycle. While there is some literature on this problem, our cost-based approach appears to be new. One issue which we do not examine here, which could be the subject of future research, is determining the optimal cycle length, $M$. Our solution to the bandwidth allocation problem, given the cycle length, could be a component in an algorithm to determine $M$, or a heuristic to approximate it. The model would need to be expanded to incorporate the effect of the cycle length on the item request probabilities.

**Appendix**

We give the proof of Lemma 5.3, that is, we show that

$$W_m(p) - W_{m-1}(p^1) = c_1 p_1 - \Delta c (1 - p_1) P_{n-1}(m).$$
Proof. Recall that for a set of items with probabilities $p, Q(k|p)$ is the probability that there are $k$ distinct requested (non-pushed) items. For notational convenience, let $Q(k) = Q(k|p^1)$. We can compute $Q(k|p)$ as follows.

$$Q(k|p) = p_1Q(k-1|p^1) + (1 - p_1)Q(k|p^1) = p_1Q(k-1) + (1 - p_1)Q(k)$$

$$Q(0|p) = \prod_{i=1}^{n}(1 - p_i); \quad Q(n|p^1) = Q(n) = 0.$$ We have

$$W_m(p) = c_1 \sum_{k=1}^{m} kQ(k|p) + \sum_{k=m+1}^{n} [c_1m + c_2(k - m)]Q(k|p)$$

$$= c_2 \sum_{k=1}^{n} kQ(k|p) - (\Delta c) \sum_{k=1}^{m} kQ(k|p) - (\Delta c)m \sum_{k=m+1}^{n} Q(k|p),$$

so

$$W_{m-1}(p^1) = c_2 \sum_{k=1}^{n-1} kQ(k|p^1) - (\Delta c) \sum_{k=1}^{m-1} kQ(k|p^1) - (\Delta c)(m - 1) \sum_{k=m}^{n-1} Q(k|p^1).$$

Now,

$$W_m(p) = c_2 \sum_{k=1}^{n} kQ(k|p) - (\Delta c) \sum_{k=1}^{m} kQ(k|p) - (\Delta c)m \sum_{k=m+1}^{n} Q(k|p)$$

$$= c_2 \sum_{k=1}^{n} k[p_1Q(k-1) + (1 - p_1)Q(k)] - (\Delta c) \sum_{k=1}^{m} k[p_1Q(k-1) + (1 - p_1)Q(k)]$$

$$+ (1 - p_1)Q(k)] - (\Delta c)m \sum_{k=m+1}^{n} [p_1Q(k-1) + (1 - p_1)Q(k)]$$

$$= c_2p_1 \sum_{k=0}^{n-1} (k + 1)Q(k) + c_2(1 - p_1) \sum_{k=1}^{n-1} kQ(k) - (\Delta c)p_1 \sum_{k=0}^{m-1} (k + 1)Q(k)$$

$$- (\Delta c)(1 - p_1) \sum_{k=1}^{m-1} kQ(k) - (\Delta c)m p_1 \sum_{k=m}^{n-1} Q(k)$$

$$- (\Delta c)m(1 - p_1) \sum_{k=m+1}^{n-1} Q(k)$$

$$= c_2p_1 \sum_{k=0}^{n-1} Q(k) + c_2 \sum_{k=1}^{n-1} kQ(k) - (\Delta c)p_1 \sum_{k=0}^{m-1} Q(k)$$

$$- (\Delta c) \sum_{k=1}^{m-1} kQ(k) - (\Delta c)(1 - p_1)mQ(m)$$
\[
- (\Delta c)(m - 1) \sum_{k=m}^{n-1} Q(k) - (\Delta c) \sum_{k=m}^{n-1} Q(k)
\]
\[
+ (\Delta c)(1 - p_1)nQ(m)
\]
\[
= W_{m-1}(p_1^1) + c_2p_1 - (\Delta c)p_1 \sum_{k=0}^{m-1} Q(k) - (\Delta c) \sum_{k=m}^{n-1} Q(k)
\]
\[
= W_{m-1}(p_1^1) + c_2p_1 - (\Delta c)p_1 \sum_{k=0}^{n-1} Q(k) - (\Delta c)(1 - p_1) \sum_{k=m}^{n-1} Q(k)
\]
\[
= W_{m-1}(p_1^1) + c_1p_1 - (\Delta c)(1 - p_1) \sum_{k=m}^{n-1} Q(k),
\]
where we use the fact that \(\sum_{k=0}^{n-1} Q(k) = 1\), and we are done. \(\square\)

Acknowledgements

Rhonda Righter’s research was supported by the Breetwor Fellowship of the Leavey School of Business at Santa Clara University.

References
