Coding and Scheduling for Efficient Loss-Resilient Data Broadcasting

Kevin Foltz
Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria, VA 22311
kfoltz@paradise.caltech.edu

Lihao Xu
Department of Computer Science
Washington University
St. Louis, MO 63130
lihao@cs.wustl.edu

Jehoshua Bruck
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125
bruck@paradise.caltech.edu

Abstract

We examine the problem of sending data to clients over a broadcast channel in a way that minimizes the expected waiting time of the clients for this data. This channel, however, is not completely reliable, and packets are occasionally lost. This poses a problem, as performance is greatly degraded by even a single packet loss. For example, one lost packet will increase our expected waiting time for an item from .75 to 2, or 167%, when sending two items with equal demands. We propose and analyze two solutions that attempt to minimize this degradation. In the first, we code packets and in the second we code packets and slightly modify our schedule. The resulting degradations are 67% for the first solution and less than 1% for the second. We conclude that using the second scheme is a very effective way to combat single packet losses, and we extend this solution to combat up to t packet losses per data item for any $t \ll k$, where k is the number of packets per data item.

1 Introduction

We examine the problem of scheduling data transmission over a broadcast channel. This problem has been studied by others. Vaidya and Hameed [8] proposed a method for nearly-optimal periodic broadcast scheduling. Jiang and Vaidya [4] look at combining the metrics of mean and variance of the waiting time, and give a scheduling method to minimize combinations of the two. Su and Tassiulas [6] also examine broadcast scheduling, including memory management of client caching [7]. Khanna and Zhou [5] look at combining waiting time and tuning time and minimizing them together. Aksoy et al. [1] discuss general issues relating to broadcast disks and push technology.

The basic model we consider is similar to that of Vaidya and Hameed, where data items are sent periodically, according to some fixed schedule. The expected waiting time for an item is the time a client will expect to wait to receive that item, averaged over all possible starting times in the schedule. Transmission time of the item is not included in the expected waiting time. Unlike Vaidya and Hameed, we look at splitting items into smaller pieces, as discussed in [2] and [3]. In this paper, we first consider two data items, and then show that the ideas extend to 3 or more items. We assume each data item has the same length, and each is split into k equal-sized pieces (packets). We examine the effect of packet losses on the expected waiting time of the clients. We show how MDS codes can help reduce expected waiting times when there are errors. In the remaining sections, we will use the term error to refer to a packet loss, and coding to refer to applying an error control code, in particular an MDS code.

In Section 2 we calculate expected waiting times as a function of the demands for the items. We then find the best schedules as a function of these demands. In Section 3 we examine the effects of a single packet loss on expected waiting times. We look at both coded and uncoded schedules and show that coding performs better. In Section 4, we consider a slightly modified schedule with coding. We show that this schedule is even better, with almost no penalty for a packet loss. In Section 5, we show how this work can be extended to a more general setting, and in Section 6, we draw some conclusions as to good scheduling methods to deal with packet loss.

2 Schedules of the form 12^n

In this work, our schedules are periodic, so we represent a schedule by one of its periods. We use a simple notation to represent schedules, where '1' means to send the first data item and '2' means to send the second data item. A schedule is written as a sequence of numbers. For example, if we alternate sending item 1 and item 2, we write the corresponding schedule as 12. Exponents indicate how much of each item to send, so 12^3 is the same as 1222, and means to alternately send data item 1 once and data item 2 three times.

Expected waiting time (EWT) is the average time clients must wait for data items. We assume clients start waiting for data items at times that are uniformly distributed over a broadcast schedule period. For item i ($i = 1, 2$), we compute how long a client must wait at each time during a schedule, and then average this waiting time over the entire schedule and get the *average waiting time* EWT_i for item i. We then average over the data items, with weights p_1 and p_2, the relative demands for the items, given to items 1 and 2, respectively, and get the expected waiting time of items 1 and 2, i.e., $EWT = p_1 EWT_1 + p_2 EWT_2$. Essentially, p_1 is the probability that the next item some client will want is item 1, p_2 is the probability it is item 2.

Example 1 *EWT of Schedule 12*
If a client wants item 1, the waiting time will be 1 if it starts listening during the broadcast of item 1. It simply receives item 1 starting somewhere in the middle, waits 1 time unit while item 2 is sent, and then receives the rest of item 1, for a total wait of 1 (we don't include transmission of useful data in the waiting time). If the client starts listening during the broadcast of item 2, it waits through the remainder of item 2, and then receives item 1, for a total wait between 0 and 1, depending on the initial listening time. Thus the average waiting time, EWT_1, for item 1 is the average of 1 and $\frac{1}{2}$ (the average of the values between 0 and 1), i.e., $EWT_1 = \frac{3}{4}$. The analysis for a client wanting item 2 is similar, and $EWT_2 = \frac{3}{4}$. So the expected waiting time, EWT, for schedule 12 with demands $p_1 = p_2 = \frac{1}{2}$ is $EWT = p_1 EWT_1 + p_2 EWT_2 = \frac{3}{4}$. □

For the schedule 12^n, we compute the expected waiting times for each item, EWT_1 and EWT_2, and the overall expected waiting time, EWT^n. We use the formula for EWT^n to compute the expected waiting time, EWT^{n+1}, for 12^{n+1}. Here 12^n means to send all of item 1 followed by n copies of item 2. These items are sent in packets, where each item consists of k packets. For simplicity of calculation, we assume $k \longrightarrow \infty$. For any sufficiently large k, such as $k > 100$, the results are similar. The calculations follow:

$$\begin{aligned}
EWT_1 &= \frac{1}{n+1}(n) + \frac{n}{n+1}\left(\frac{n}{2}\right) \\
&= \frac{n^2+2n}{2(n+1)} \\
EWT_2 &= \frac{1}{n+1}\left(\frac{1}{2}\right) + \frac{n-1}{n+1}(0) + \frac{1}{n+1}(1) \\
&= \frac{3}{2(n+1)} \\
EWT^n &= \frac{n^2+2n}{2(n+1)}p_1 + \frac{3}{2(n+1)}(1-p_1) \\
&= \frac{3}{2(n+1)} + \frac{n^2+2n-3}{2(n+1)}p_1 \\
EWT^{n+1} &= \frac{3}{2(n+2)} + \frac{n^2+4n}{2(n+2)}p_1
\end{aligned}$$

We find the values of p_1 for which 12^n and 12^{n+1} have the same expected waiting times, and we then compute the optimal n as a function of p_1.

$$\begin{aligned}
\frac{3}{2(n+2)} + \frac{n^2+4n}{2(n+2)}p_1 &= \frac{3}{2(n+1)} + \frac{n^2+2n-3}{2(n+1)}p_1 \Longrightarrow \\
p_1 &= \frac{3}{n^2+3n+6} \Longrightarrow \\
n &= \left\lceil -\frac{3}{2} + \frac{1}{2}\sqrt{\frac{12}{p_1} - 15} \right\rceil
\end{aligned}$$

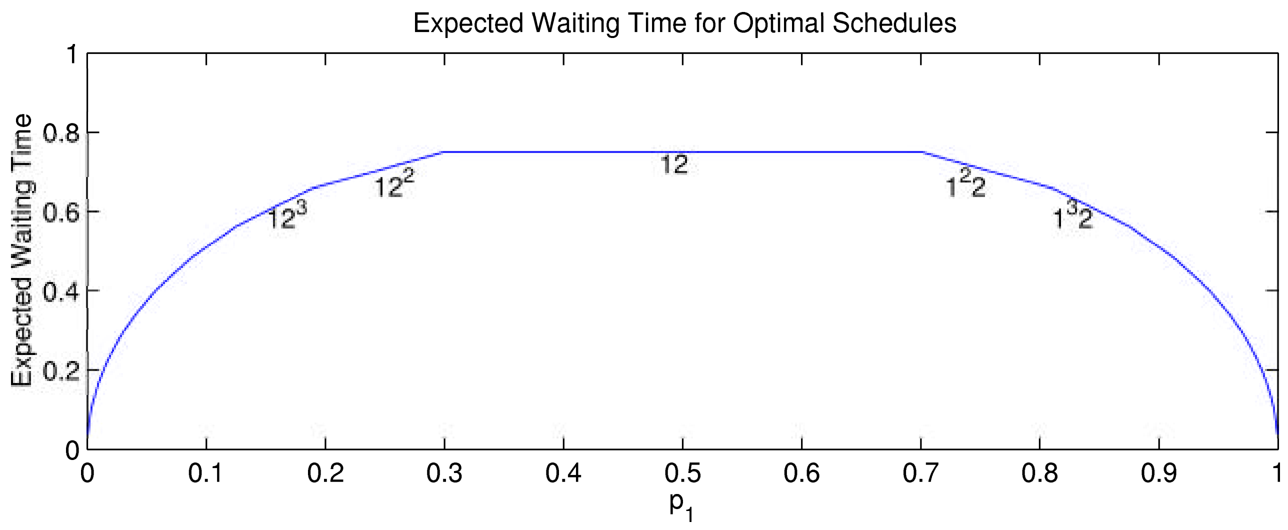

Figure 1: Expected waiting time vs. p_1 for the optimal schedules of the form 12^n and 1^n2. The individual optimal schedules are labeled for $n \leq 3$.

We use the ceiling function for n because at the boundary where 12^n and 12^{n+1} are optimal, we want to switch from n to $n+1$. Without the ceiling function, we would only go from n to $n+\epsilon$. Figure 1 shows a plot of expected waiting time (EWT) as a function of p_1 when we schedule with these values of n. Each linear segment of the curve corresponds to a particular schedule (value of n).

3 Performance with 1 error

We look at what happens with the schedule 12^n when a client loses one packet of the item it wants. We look at the two cases of coding and no coding. For our baseline, we use the EWT with no errors. Notably, this EWT is the *same* for coding and no coding. For coding, we assume an MDS code where any k of n coded packets are needed to reconstruct the original k. We do not specify n, but a value such as $n = 2k$ will suffice for all examples presented. If we do not code, we must wait until the next broadcast of the *specific* packet that we missed. If we do code, we need only wait until the broadcast of the *next* piece of the desired item, since we can reconstruct the item from any k of the n coded packets. We examine the advantage this gives coding over no coding when we lose one packet.

We look at what happens when we wait for item 1. Without coding, we have an expected waiting time of

$$EWT_1^{nc} = \frac{3n^2+4n+1}{2(n+1)}.$$

With coding, we have an expected waiting time of

$$EWT_1^{c} = \frac{3n^2+2n}{2(n+1)}.$$

For item 2, without coding we have an expected waiting time of

$$EWT_2^{nc} = \begin{cases} \frac{n+5}{2(n+1)} & \text{if } n \geq 2 \\ 2 & \text{if } n = 1 \end{cases}$$

With coding, we have an expected waiting time of

$$EWT_2^{c} = \begin{cases} \frac{3}{2(n+1)} & \text{if } n \geq 2 \\ \frac{5}{4} & \text{if } n = 1 \end{cases}$$

Combining the expected waiting times for items 1 and 2, we see that we get overall expected waiting times, for $n \geq 2$, of

$$EWT^{nc} = \frac{n+5}{2(n+1)} + \frac{3n^2+3n-4}{2(n+1)}p_1$$

$$EWT^{c} = \frac{3}{2(n+1)} + \frac{3n^2+2n-3}{2(n+1)}p_1$$

For $n = 1$, we add $\frac{1}{2}(1-p_1)$ to each.

Using our earlier value of $n = \left\lceil -\frac{3}{2} + \frac{1}{2}\sqrt{\frac{12}{p_1} - 15} \right\rceil$, we can plot EWT as a function of p_1, as in Figure 2, for coding and no coding. We note that our EWT increases about half as much with coding as with no coding. For example, at $p_1 = \frac{1}{2}$, we have an expected waiting time of $\frac{3}{4}$ with no errors. With one error and no coding, this jumps to 2, but with coding, it only increases to $\frac{5}{4}$. However, in either case, EWT increases noticeably.

4 Changing the Schedule Slightly for Improved Performance

We try a slightly different set of schedules, where we send one more packet of item 1 in the schedule 12^n, or one more packet of item 2 in the schedule 1^n2 (or one more packet of each item for the schedule 12). Essentially, we are using the schedule $1^{1+\frac{1}{k}}2^n$ or $1^n2^{1+\frac{1}{k}}$ (or $1^{1+\frac{1}{k}}2^{1+\frac{1}{k}}$), where $k \longrightarrow \infty$. These schedules have slightly worse performance than the originals when there are no errors, but with one error perform much better.

We omit the calculations, as they are fairly computationally intense, and give the results. For large k, the plot of EWT vs. p_1 with the new schedule and one error closely resembles the original curve for no coding and no errors. In Figure 2, the curve for this slightly modified schedule with one error would lie just above the curve for the original schedule without errors. As $k \longrightarrow \infty$ these two become indistinguishable. So, the bottom curve represents expected waiting time for the modified schedule with coding and one packet loss as well as the original schedule with no packet losses.

5 Extensions

Our work has focused on combating a single packet loss in two data items of equal size. However, the ideas of coding and slightly changing the schedule are extensible in a number of ways.

A simple extension would be to consider items of different sizes. Either the packets of the items would differ in size or, more likely, one would consist of more packets than the other. The main difference here is that the starting schedules would be different, since the optimal starting schedules (for no errors, no coding) depend on the demands and sizes of the items. The rest of the work and the resulting improvements in EWT closely parallel the work for the equal-length case.

Another extension would be to consider more than two items. Again, the starting schedules would be different, but the rest of the analysis is similar to the work for two items.

An interesting extension is to consider more than a single packet loss. In this case, we find that the EWT for no coding increases a significant, although diminishing, amount with each additional packet lost. The EWT with coding, however, does not increase significantly. In fact, for large k the change in EWT is close to zero. For the modified schedule with coding, we have an increase in EWT when going from one to two losses. The resulting EWT for two losses is approximately the same as the EWT for the schedule with coding but no modification. For more losses, the EWT stays nearly constant, as it does without the schedule modification. Figure 3 illustrates the expected waiting times for $p_1 = p_2 = \frac{1}{2}$ as a function of the number of packets lost, for each method.

Multiple errors thus seem to be a significant problem. However, we can reduce the EWT to nearly the no-error value by adopting a generalized version of our modified schedule. Instead of adding one additional packet to our schedule, we add t additional packets, where t is the number of errors we wish to combat. Using this (further) modified schedule, we reduce the EWT to nearly the no-error EWT for any number of packets lost up to t.

We must note, however, that this technique only works for small numbers of packet losses. If the number of lost packets is some appreciable fraction of the total number, these techniques will help, but the resulting performance will

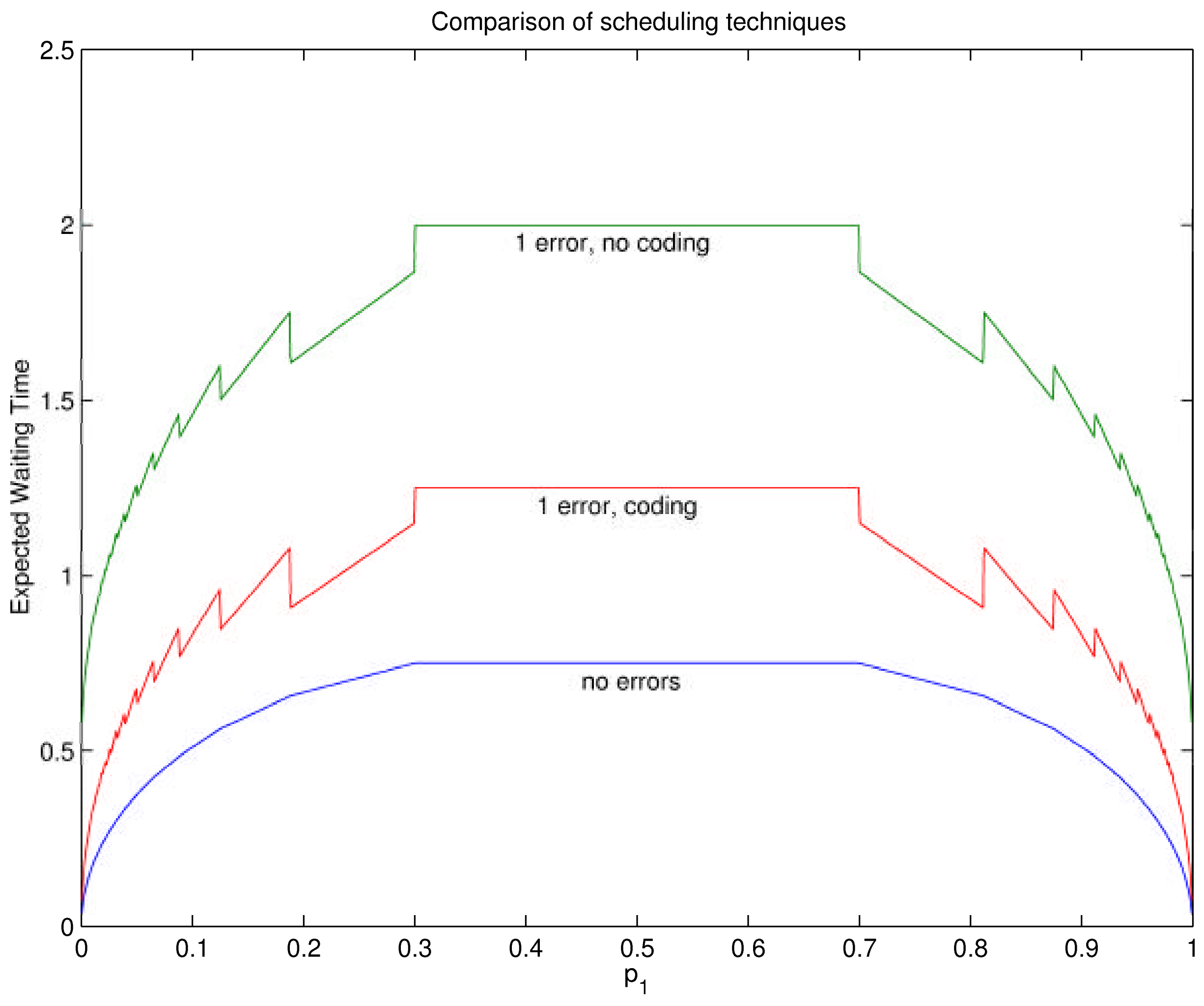

Figure 2: EWT for various scheduling methods. The bottom curve is for the original schedule (no coding) with no errors. The top one is for no coding with one packet error. The middle one is for coding and one error.

no longer approach the optimal time, as our assumptions about $t \ll k$ no longer hold. This is expected, though, since as the error rate grows, the expected waiting time must also grow. Our methods provide a way to prevent unnecessarily large increases in EWT due to a very small increase in error frequency.

6 Conclusions

We make two general conclusions based on the analysis presented. The first is that coding and slightly changing the schedule are good ways to combat errors in transmission. We see that we can essentially negate the effect of errors on expected waiting time by using these techniques. If changing the schedule is not possible, it would be advisable to at least code the items to gain some improvement. It is interesting to note that changing the schedule without coding offers no gains. Thus, coding is the key to making error recovery work.

Our second conclusion is that knowledge of how errors occur helps. We see that if we schedule to combat one error and we get two, our coding and modifying are no better than just coding. If a bound, b, on the number of packet losses can be established, where b is significantly less than k, then we can schedule for b packet losses and achieve near-optimal performance for any number of errors up to b.

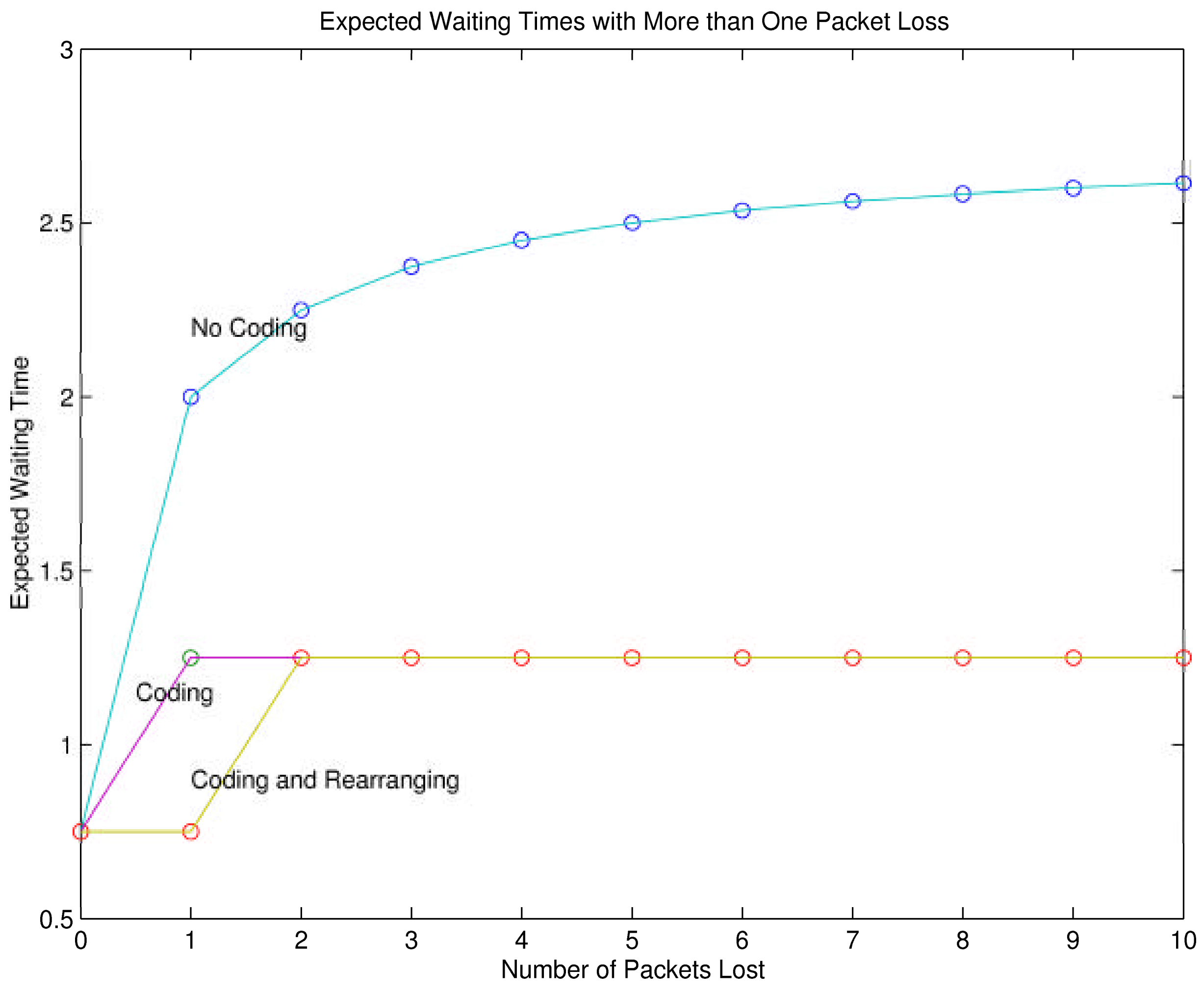

Figure 3: Expected waiting time as a function of the number of packet losses for no coding, coding, and coding with schedule modification.

References

[1] D. Aksoy, M. Altinel, R. Bose, U. Centemel, M. Franklin, J. Wang, S. Zdonik, "Research in Data Broadcast and Dissemination," *Proceedings of 1st International Conference on Advanced Multimedia Content Processing*, Osaka, Japan, Nov. 1998.

[2] K. Foltz, J. Bruck, "Robustness of Time-Division Schedules for Internet Broadcast," ISIT 2002, Lausanne, Switzerland.

[3] K. Foltz, J. Bruck, "Splitting Schedules for Periodic Internet Broadcast," *IEEE Trans. Info. Theory*, vol. 48, pp. 345-358, Feb. 2002.

[4] S. Jiang, N. H. Vaidya, "Scheduling Algorithms for a Data Broadcast System: Minimizing Variance of the Response Time," Technical Report 98-005, Computer Science, Texas A&M University, February 4, 1998.

[5] S. Khanna, S. Zhou, "On Indexed Data Broadcast," Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98), pp. 463-472, New York, 1998.

[6] C. J. Su, L. Tassiulas, "Broadcast Scheduling for Information Distribution," Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFO-COM'97), vol. 1, pp. 109-117, 1997.

[7] L. Tassiulas, C. J. Su, "Optimal Memory Management Strategies for a Mobile User in a Broadcast Data Delivery System," *IEEE JSAC Special Issue on Networking and Performance Issues of Personal Mobile Communications*, 15(7):1226-1238, September 1997.

[8] N. H. Vaidya, S. Hameed, "Data Broadcast in Asymmetric Wireless Environments," First International Workshop on Satellite-based Information Services (WOSBIS), Rye, NY, November 1996.