

An Efficient Data Broadcasting Scheduler for Energy-Constraint Broadcast Servers

Jen-Jou Hung¹ and André Seifert²

¹ National Taiwan University of Science and Technology
Department of Information Management
No. 43, Section 4, Keelung Road, Taipei 10672
Taiwan R.O.C.

`jhung@juno.cs.ntust.edu.tw`

² University of Konstanz, Database Research Group
P.O. BOX D188
78457 Konstanz
Germany
`Andre.Seifert@uni-konstanz.de`

Abstract. In this paper, we study the problem of maximizing the data broadcasting server’s service rate, defined as the ratio of satisfied vs. unsatisfied time-constrained data requests in an energy-limited on-demand data dissemination environment. The paper’s main contribution is an algorithm, called PBS (standing for *Power-aware Broadcast Scheduler*), with $\mathcal{O}(n)$ space and time requirements, which significantly outperforms state-of-the-art on-demand real-time scheduling algorithms in terms of its primary performance metric, i.e., the service rate. Additionally, even though designed to maximize the service rate of the server, it only slightly underperforms other schedulers in terms of the mean response time which use this metric as the main performance criterion. Our further contribution is the $\text{PBS}_{\text{light}}$ algorithm, which is a low-overhead, approximate and parameterized variant of PBS. Even though still running in linear time, $\text{PBS}_{\text{light}}$ significantly reduces the scheduling costs of PBS. Additionally, it provides performance results comparable to or even slightly better than those of PBS, however, at the expense of servicing requests for less frequently requested data items less often. The third contribution is that both proposed algorithms are built upon the criteria of providing continuous broadcasting services to mobile clients until either the estimated service period is completed or battery resources are recharged. This objective is achieved by calculating a broadcast area schedule which maps regional communication areas located within the communication coverage area of the server to upcoming broadcast cycles in accordance to the durability of the available energy resources and the workload conditions of the server.

1 Introduction and Motivation

In the course of the proliferation of mobile devices and the potential of anytime, anywhere information exchange, multiple new applications arise that in-

volve dissemination of data to a large population of mobile clients. Many of the dissemination-oriented applications such as news distribution systems, traffic information systems, entertainment delivery, etc. [29], have data access and data flow characteristics that differ from those of traditional client-server applications. Service load asymmetry, data volume asymmetry, and overlapping data requirements of the client population typically characterize the former class of applications [1].

Data broadcasting is a widely accepted and efficient approach to address the asymmetry problem prevalent in such systems. Since data broadcast has the ability to simultaneously satisfy the data needs of multiple clients, it efficiently utilizes the available downstream communications capacity and thereby contributes to the scalability of the system. Three data broadcasting models can be distinguished, namely *static* and *on-demand* broadcast, and a *hybrid* combination of the two. We adapt *on-demand* data broadcasting as the data delivery model in our work, since it allows us to generate tailor-made data broadcast programs built upon the clients' current data needs and QoS requirements (clearly, any of the above mentioned and even more advantages would apply to a hybrid scheme, but because the latter is much more complex, it would aggravate our model's presentation). In particular, we strive towards the challenging objective of maximizing the *request service rate* and *service duration period* of battery-powered on-demand data broadcasting servers. Mobile broadcast servers which do not rely on plug-in power supply are useful for facilitating information-centered applications including battlefield applications, emergency and disaster response operations, expeditions, etc. To provide further evidence for the necessity of designing power-aware data broadcast schedules, consider the following three application scenarios:

- As you might remember, on August 14, 2003, shortly after 4 p.m., a power outage affected dozens of cities in the eastern United States and Canada and subsequently causing the communications facilities to breakdown. During such critical time periods, emergency information such as the cause and length of the outage, status information about the current state of the public transport system, evacuation notices, etc. are of urgent need to the affected public, to the media and also to the instructed emergency response teams. Since a lot of users in such a situation are expected to request the same information at the same time, it is useful to disseminate the respective data through a battery backed-up mobile or stationary broadcast server. As the amount of energy stored in a battery is not infinite, the broadcast server is required to efficiently utilize its available battery power so as to provide services until the power system is restored and the emergency operations are completed.
- Consider events such as music and drama festivals in the woods at which mobile broadcast servers would be useful for disseminating information on the festival program, catering, locations of first aid stations, evacuation paths, etc. to the visitors. Additionally, battery-powered broadcast servers could be

set-up along the highways leading to the festival giving direction and parking information to the arriving participants.

- Another promising application domain for power-aware data dissemination scheduling are distributed wireless sensor networks in which typically a relatively large number (on the order of a couple of hundreds or thousands) of low cost, tiny-sized, battery-powered sensors collaborate with each other. For example, in an ad-hoc smart dust environment [17, 22], such as an earthquake-damaged building, a large number of specialized disposable sensors (specialized to detect heat, smoke, gas leakage, injured and trapped casualties, etc.) are densely scattered within the building, establish a mobile ad-hoc network (MANET) and cooperate with each other to gather information helping the emergency response teams to locate patients/casualties, to find safe evacuation paths, to inform the fire brigade about the possible spreading of fire, etc. In order to accomplish these tasks in an energy-efficient way and to evaluate abnormal situations accurately, the highly specialized sensors need to exchange their sensor data with multiple or even all other smart dust sensors in the system. As the collaborating sensors are typically distributed in a small geographical region (within a few dozens of meters), single-hop broadcasting rather than multi-hop broadcasting would be the most efficient transmission strategy [30].

All aforementioned scenarios have in common that broadcast technology would be helpful for information dissemination, but plug-in power supply is not available to the server/sensors and, therefore, energy becomes a precious commodity. Under the realistic assumption that the period during which the broadcast server is forced to rely on battery power is a-priori known or somehow predictable and that client requests contain *firm deadlines* [37] specifying the maximum tolerable response time for each request, two tightly linked objectives can be defined for such systems: (a) to maximize the number of client requests served within their specified deadlines, and (b) to provide broadcasting support until the communications services are no longer required anymore (e.g., the emergency operations are completed).

To the best of our knowledge, previous research work has not been directed towards solutions within the specified problem space. To rectify this, we propose two broadcast scheduling algorithms optimized to provide both low *request failure rates* and *response times* while guaranteeing communications availability until the end of the *planned service period*. This can be achieved by dynamically varying the radius of the communication area serviced by the broadcast server in accordance to its remaining energy resources as well as the timing constraints and origins of the requests received. Obviously, due to the characteristic of the wireless medium that the communication signal attenuates proportional to D^α , where D denotes the Euclidian distance and α is a constant, servicing clients' requests originating from further distances is less efficient than those received from clients close to the server. Therefore, the proposed scheme tends to favor the requests from clients located within closer proximity to the server.

The rest of this paper is organized as follows. Work related to this research is presented in Section 2. Section 3 briefly describes the system environment for which PBS is designed, the work’s underlying assumptions, and the criteria used to evaluate the algorithm’s performance. In Section 4, we describe the PBS scheduling algorithm and its approximate, parameterized version PBS_{light}. Section 5 presents experimental results of evaluating PBS and PBS_{light} in terms of their performance, computational overheads, and scalability under various system settings and workload conditions. Results of a comparison study of PBS against other related scheduling algorithms is also illustrated there. Finally, Section 6 concludes the paper and gives directions for future work.

2 Related Work

The problem of minimizing the energy consumption of energy-constrained data dissemination systems can be approached from the viewpoint of hardware and software. Hardware-oriented research has identified a rich collection of possible techniques to enable wireless communication system designers to trade-off energy vs. performance. Prominent low-power design techniques related to the system hardware include *variable clock speed CPUs* [4, 16, 35, 45], *flash memory* [10, 28], and *disk spin-down* [6, 11, 12, 15, 26, 51]. Another popular energy-tuning knob, making our software-based solution to the energy consumption problem feasible at all, is an *adjustable power amplifier* that allows energy to scale with transmission range [21, 31, 32, 38]. Adjusting the output power of the transmitter ensures that energy is not wasted by transmitting beyond the intended communication range. Commercially available network cards are already capable of gradually adjusting transmission power. For example, the Cisco Aironet 350 series card [8] offers fine-grained transmit power control by having six different transmit power levels: 1mW, 5mW, 20mW, 30mW, 50mW, and 100mW. An equally important energy-preserving technique is to reduce the transmit time of the radio signal. This can be accomplished by sending multiple bits per transmitted symbol, i.e., by using *M-ary modulation*. M-ary modulation provides higher data throughput since each transmitted symbol comes from a set of M, rather than 2 (as in binary modulation), distinct waveforms. Using M-ary modulation, however, does not come for free, but it results in more complex modulation/demodulation circuitry and increased radio power consumption [39, 40, 44]. Another hardware-oriented tuning knob that has a direct and substantial effect on the energy efficiency of data dissemination systems is the error control strategy. Error control is required to ensure the reliability of a communication link, and is typically enforced using either *forward-error correction* (FEC) or *automatic repeat request* (ARQ). As each of the two methods has limitations that render it inappropriate for achieving the goal of minimizing power consumption, a hybrid scheme in which the most energy-efficient combination of FEC and ARQ is chosen and adapted over time has been proposed in [25].

There are two software-oriented areas of work related to the research presented in this paper: (a) the first area concentrates on broadcast scheduling

strategies to improve the data response times and/or to meet timing constraints specified by the requests, and (b) the second area is concerned with strategies to preserve and optimally use the scarce energy of mobile hosts. Various on-demand broadcast scheduling policies have been proposed for both real-time and non-real-time data dissemination systems. Well-known on-demand schedulers belonging to the latter class include *First-Come-First-Serve* (FCFS), *Most Requested First* (MRF), *Most Requests First Lowest* (MRFL), *Longest Wait Time First* (LWF) [13, 48]. Among those algorithms, LWF excels in minimizing the response time. However, due to having a computational overhead proportional to the number of outstanding requests, LWF is expensive to implement. A low-overhead alternative to LWF is the *RxW algorithm* [3] which is fast and does only marginally underperform LWF. Another alternative algorithm is *PIP* [42] which shows even slightly better performance results than LWF and can be implemented to run with constant time complexity. However, it assumes a-priori knowledge of the clients access requirements which is not a reasonable assumption. On demand schedulers for applications with variable data item sizes was studied in [2]. To evaluate the performance for items of different sizes, a new performance metric called *stretch* was introduced, defined as the ratio of the access time of a request to its service time. Based on the stretch metric, four different algorithms, namely the *Preemptive Longest Wait First* (PLWF), *Shortest Remaining Time First* (SRTF), *Longest Total Stretch First* (LTSF), and *MAX algorithm*, have been quantitatively investigated with the result that none of them is superior to the others in all studied scenarios. In summary, however, the MAX algorithm performs quite well in both the worst and average cases in access time and stretch measures.

If information is to be delivered in a timely manner, real-time scheduling algorithms are applied. Proposed strategies that consider deadlines attached to the requests in order to make scheduling decisions are *EDF* and *EDF_{Batch}* [27, 50]. *EDF_{Batch}* is superior to *EDF* as it exploits batching and handles multiple requests of the same item via a single broadcast transmission. Both algorithms ignore information on the number of pending requests for the same data item and are therefore expected to yield performance results inferior to those of PBS as the latter makes use of them for its scheduling decisions.

An effective approach to save the scarce energy resources of mobile hosts is to switch them as much as possible into low power states (*doze* or *sleep mode*). To this end, Imielinski et al. [20] proposed to enrich the broadcast program by some index in order to make the broadcast channel self-descriptive and to enable clients to operate in doze mode until the required data item arrives. To reduce the number of index probes for frequently requested objects, Chen et al. [7] and Shivakumar et al. [41] considered unbalanced tree structures that optimize the tuning time for non-uniform data accesses. More precisely, *k-ary Alphabetic Huffman trees* were proposed that minimize the average index search cost by reducing the number of index node probes for data objects with high access probabilities at the expense of spending more on those with a low popularity. Tan et al. [43] proposed efficient index structures facilitating selective tuning

for non-uniform broadcasts in which data items are disseminated with various frequencies. To allow system designers to trade-off between access latency and tuning time based on the respective application-specific requirements, the *flexible* and *exponential indexing* method has been proposed [19, 49]. Both methods achieve this goal by providing the system designer with tuning parameters that offer great flexibility in trading access time against tuning time and vice versa. While all the strategies presented above are directed towards reducing the power consumption of mobile hosts which request information rather than provide it, our work focuses on the side of the information provider and addresses the issue of generating power-aware broadcasting schedules for mobile energy-constrained data dissemination servers.

3 System Environment and Assumptions

3.1 System Architecture and Data Delivery Model

We consider an on-demand broadcasting system as shown in Figure 3.1. It consists of an energy-constrained *broadcast server* with a *database* keeping the data expected to be of interest to a large client population, numerous *mobile clients* located inside and outside of the server’s communication coverage area, an unidirectional broadband *broadcast channel* and multiple dedicated low-bandwidth *uplink channels*. The uplink and downlink communication links are expected to be physically independent with the former being used by clients for sending requests to the server and with the latter being a “listen only” channel for receiving the requested data from the server.

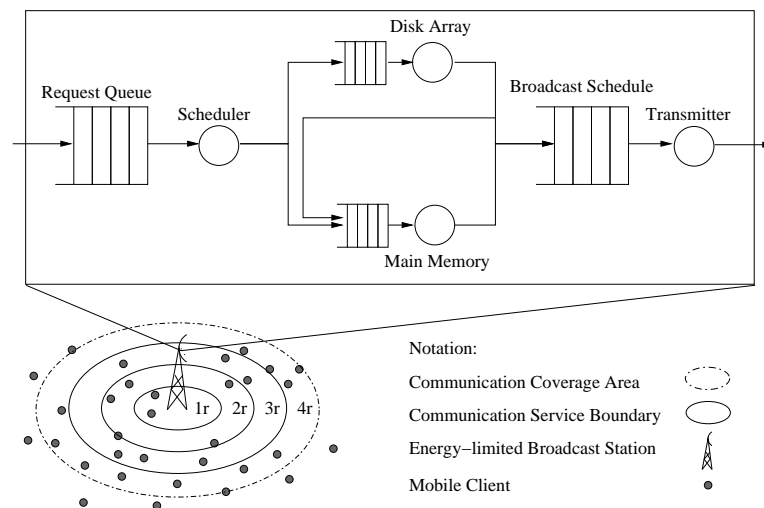


Fig. 3.1. On-demand broadcast server architecture.

Client requests are initiated as a response to local cache and memory misses and are efficiently queued at the broadcast server upon their arrival. Requests corresponding to the same data item i are grouped together forming a multi-request. Requests transferred from the client to the server are pairs $\langle ID_i, DL \rangle$ where ID_i represents the unique identifier of the data item i and DL denotes the point in time in the future (i.e., the broadcast cycle number) until which the requested data item is required, at the latest, by the user. Our model is built upon *firm deadlines* for data delivery, i.e., once the deadline expires, the requested data item will be useless for the client and can therefore be evicted from the broadcast request queue. The *server request queue* is organized in a multi-threaded manner consisting of three independent doubly-linked and sorted lists which are efficiently accessible and maintainable through the associated hash index (see Figure 3.2). The index is built by applying a collision-free or an imperfect hash function $f(k)$ on the identifying key k of any queued data item i . The sorted hash table maintains for any requested data item i a triple containing pointers to the item's entries in the *Expiration Time List* (ET-List), the *Request List* (R-List), and *Starvation List* (S-List), respectively. The three lists themselves keep track of aggregated and non-aggregated statistics collected for any recently requested data item i since its last dissemination. The gathered request information is used by the scheduler in a way as explained in Subsection 4.3 so that the request service rate is maximized and the planned service duration period of the broadcast server is met.

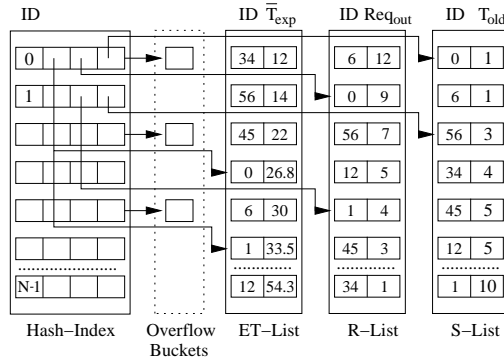


Fig. 3.2. Data structures of the broadcast server's request queue.

Further, we make the following assumptions: For reasons of simplicity, we assume that there exists only one broadcast server capable of disseminating information to a large population of mobile clients within the server's communication coverage area. We assume the broadcast server to use an omni-directional antenna to transmit messages to mobile clients, thus enabling it to transfer data requested by multiple clients by a single transmission — of course only provided that those clients are within the transmission range of the server. Additionally,

we assume that the server is able to adjust its communication distance by changing the transmission power of the broadcast signal. We expect the server to have a maximum transmission power of P_{max} , enabling it to serve a circular area with radius r_{max} . Because of signal attenuation due to the communication medium and the antenna characteristics, the power required by the server to transmit the signal to distance r_{max} is given by:

$$P_{max} = P_{thresh} \cdot r_{max}^\alpha, \quad \alpha \geq 2, \quad (1)$$

where α denotes the channel loss exponent³ and P_{thresh} presents a technology-specific threshold value which the power of the signal needs to exceed in order to be received correctly by the client. Note that in reality, the power required by the server to transmit the signal to distance r_{max} is not only a function of the signal attenuation over r_{max} and the signal quality threshold P_{thresh} , but also of the ambient noise level of the receiver. However, as the ambient noise level of the receiver is independent of the distance of the source and the receiver and depends purely on the operating conditions of the receiver, it is essentially a constant and can be neglected in the energy cost model.

From Equation 1 it follows that when the broadcast server BS_i intends to disseminate information to a set of mobile clients $ClientBase$, it needs to transmit by a power P_i given by:

$$P_i = P_{thresh} \cdot D_{i,k}^\alpha, \quad \text{where } k = \max\{D_{i,j} | j \in ClientBase\}, \quad (2)$$

where $D_{i,j}$ denotes the distance from the mobile broadcast server BS_i to the mobile client j . Since we are only concerned with the relative performance of different broadcast scheduling policies, we can set the transmission quality threshold parameter P_{thresh} to 1 (similar to [34, 46]) without loss of generality. For ease of presentation, we further assume that the broadcast server BS_i does not disseminate requested information to arbitrary distances within its communication coverage area, but rather may transmit data to one of n possible different areas being centered around BS_i with a radius of $1r, 2r, \dots, (n-1)r$, and nr , respectively, whereas nr equals r_{max} . In what follows, the circular area with radius kr will be denoted as $area(k)$ where k is an integer number between 1 and n , and the set of all those variably-sized areas located within the server's coverage area is presented by $ServiceArea$. Further, we will refer to the spatial region between the boundaries of two neighboring areas $area(m)$ and $area(n)$ with $m < n$ by $region(m, n)$. Finally, we assume the indexing of the contents of the broadcast channel to be an integral part of our on-demand data delivery model. Adding an index such as a bit-vector or one of the tree-based schemes proposed in [20] to the channel helps clients to save tuning time and, more importantly, conserves their power resources as they do not need to continuously monitor the broadcast medium after sending a request to the server.

³ The value of the attenuation parameter α depends on the environmental conditions and is typically around 2 for short distances and omni-directional antennae, and around 4 for longer distances (> 100 meters) [33, 36].

3.2 Energy Preservation Approach and Performance Metrics

As microwave signal propagation obeys the 2-nd to 4-th power path loss law, shorter-range communication is generally much cheaper in terms of power consumption than longer-range one. As a result, transmitting information to clients within close proximity is severely less energy intensive for the broadcast server than, for example, to the clients located at the boundary of the communication coverage area. This observation allows us to fine-tune the power consumption of the broadcast server in such a way that its power resources will either last until the planned service period is completed or the utilized energy storage facilities (e.g., batteries) can be recharged or replaced. If we realistically assume that the server has only a limited amount of energy at its disposal and the available energy would not be enough to disseminate data to the entire communication coverage area throughout the service duration, an energetically feasible approach would be to reduce the frequency of servicing clients located at farther distances from the server. Clearly, this approach does not come for free, as it might increase the mean request response time of the clients not always within the dynamically changing geographic scope of the server's broadcast area. Consequently, our approach trades-off increased *service durability* for an increase in the *mean request response time*. As the latter metric is time-independent in the sense that its value does not reflect the duration of the server's service period, we will only use it as a secondary performance indicator in what follows. Instead, and in contrast to traditional non-real-time scheduling approaches [3, 13, 42, 48], we rather rely on the *service rate*, defined as the ratio between timely satisfied data requests and those whose deadline constraints were failed by the server, to be our primary performance metric. Using the service rate as main comparison metric allows us to quantify the energy and scheduling efficiency of the PBS scheduler. Since PBS trades-off timely data delivery for request response time, it is expected to be outperformed by access time-efficient algorithms such as LWF [13] or RxW [3] w.r.t. responsiveness, but a better service rate may be anticipated due to improved service availability in comparison to the above mentioned and other related scheduling approaches.

4 The PBS Scheduling Algorithm

In this section, we describe the *Power-aware Broadcast Scheduler*, or PBS, consisting of two orthogonal parts, namely *Broadcast Area Selector* and *Data Item Selector*. As the name implies, the Broadcast Area Selector determines how frequently and at what particular broadcast cycle(s) any of the n equally-spaced communication areas within the server's communication coverage area will be serviced. Based on the calculated broadcast area schedule (BAS) and the information contained therein, as to which communication area, $area(k)$, in the next broadcast cycle (BC) is to be transmitted, the Data Item Selector then chooses from the set of all outstanding, un-expired requests of $area(k)$ those that yield the highest service rate to the client population.

4.1 The Broadcast Area Selection Algorithm

In contrast to previously proposed real-time and non-real-time on-line broadcasting schedulers, instead of assuming that the broadcast server is connected to continuous power sources, we expect it to have only a limited amount of energy at its disposal. Consequently, and, of course, depending on its available energy resources, the server might be unable to transmit the requested data to its maximum possible communication distance throughout its service period, and, therefore, will be forced to service those areas with larger radii less frequently or even not at all. The decision on the frequency of servicing a particular communication area is primarily influenced by the *amount of energy* available to the server, and secondarily by the *number of pending requests* and their associated *timing constraints* of the respective area. In what follows, we describe how the PBS scheduler calculates the BAS, and in order to facilitate its presentation, we refer to the following notions:

T_{ser}	Estimated or expected service period of the broadcast server in BCs.
$T_{ser}(k)$	Fraction of T_{ser} reserved for broadcasting data to $area(k)$.
$\overline{T}_{arr}(k)$	Mean arrival time between two consecutive data requests originating from $area(k)$.
$\overline{T}_{exp}(k)$	Weighted (by the request arrival rate) mean expiration time of all requests coming from $area(k)$.
E_{avail}	Amount of energy available to the server.
$E(BAS)$	Amount of energy required to deploy the calculated BAS.
$E(k)$	Amount of energy needed to transmit data to $area(k)$ for the duration of one BC.
$P(k)$	Power required to transmit data to $area(k)$.
$S_{loss}(k)$	Service loss per unit of energy if $T_{ser}(k+1)$ is assigned to $T_{ser}(k)$.
$S_{inc}(k)$	Service increase per unit of energy if $T_{ser}(k-1)$ is assigned to $T_{ser}(k)$.

By assuming the independence of the data request rate from the requester's location, i.e., that an item i is equally likely to be requested by a client located in $region(m, n)$ and in $region(n, k)$ with $m \neq k, m \neq n$, the benefit of servicing outstanding requests of $area(k)$ is proportional to the expiration arrival ratio (EAR) being computed as follows:

$$EAR(k) = \frac{\overline{T}_{exp}(k)}{\overline{T}_{arr}(k)}. \quad (3)$$

Clearly, the higher the value of $EAR(k)$, the more rewarding it becomes to satisfy outstanding requests of $area(k)$. Therefore, and in order to increase the service rate of the server, areas with higher EARs should be selected for data dissemination more frequently. However, as the power consumption for sending microwave signals to longer distances grows exponentially, the frequency

of broadcasting information to $area(k)$ should be inversely proportional to $P(k)$. Consequently, the fraction of T_{ser} used to transmit data to $area(k)$, denoted by $T_{ser}(k)$, should be assigned in proportion to $\frac{EAR(k)}{P(k)}$ and can be computed by the following equation:

$$T_{ser}(k) = \frac{\frac{EAR(k)}{P(k)}}{\sum_{i=1}^N \frac{EAR(i)}{P(i)}} \cdot T_{ser}, \quad (4)$$

with N denoting the cardinality of $ServiceArea$.

Once the Broadcast Area Selector has determined $T_{ser}(k)$ for all areas $area(k) \in ServiceArea$, it needs to verify whether the deployment of the calculated BAS is efficient and energetically feasible, depending on whether it *exceeds* or does *not fully utilize* the energy resources of the server. Based on the amount of energy at the server's disposal, it may find itself in one of following situations: (a) First, and most conveniently, the amount of energy required to implement the preliminary BAS equals the energy resources of the server. (b) Second, the available energy resources (E_{avail}) of the server do not suffice to provide uninterrupted broadcasting service throughout T_{ser} . In that case, the BAS needs to be adjusted to make the server consume less energy per time unit. To address the energy shortage problem, a viable solution is to either partially or even completely cease broadcasting activity to those communication areas whose coverage is associated with a relatively low service rate loss per energy unit spent. The $area(k)$ with the *lowest service loss* per unit of available energy can be determined by calculating $S_{loss}(k)$ for all areas $\{area(k) \in ServiceArea | k > 1\}$ according to the following formula:

$$S_{loss}(k) = \frac{EAR(k) - EAR(k-1)}{P(k) - P(k-1)}. \quad (5)$$

After the $area(k)$ with the lowest $S_{loss}(k)$ is found, the scheduler would use the following formula to determine the fraction of $T_{ser}(k)$ that needs to be assigned to $T_{ser}(k-1)$ in order to decrease and possibly overcome the energy deficit:

$$T_{ser}(k-1) = \begin{cases} T_{ser}(k-1) + T_{ser}(k), & \text{if } E_{avail} < (E(BAS) - T_{ser}(k) \cdot (E(k) - E(k-1))) \\ T_{ser}(k-1) + \frac{E(BAS) - E_{avail}}{E(k) - E(k-1)}, & \text{otherwise.} \end{cases} \quad (6)$$

Depending on the difference between $E(BAS)$ and E_{avail} as well as on the energy savings potential of re-assigning service time from $area(k)$ to $area(k-1)$, the scheduler might be forced to suspend the broadcasting activity to other communication areas than $area(k)$. That is, the scheduler will continue to reduce longer-range communication duties in favor of shorter-range ones until the condition that E_{avail} equals E_{BAS} holds.

(c) Finally, E_{avail} might exceed the amount of energy required to deploy the

preliminarily calculated BAS, in which case the server is able to increase the portion of longer-distance broadcasts which certainly helps increase its service rate. Contrary to the previous situation, this time the scheduler needs to determine the $area(k)$, which would cause the highest service rate increase per energy unit consumed (compared to the other areas), if served more frequently by re-allocating $T_{ser}(k-1)$ to $T_{ser}(k)$. More precisely, the scheduler has to determine the $area(k)$ with the *highest service rate increase* per unit of energy deployed, which is done by computing $S_{inc}(k)$ for all areas $\{area(k) \in ServiceArea | k \geq 2\}$ according to the following equation:

$$S_{inc}(k) = \frac{EAR(k) - EAR(k-1)}{P(k) - P(k-1)}. \quad (7)$$

Based on the result of Equation 7, the scheduler needs to determine whether it is affordable (energy-wise) to re-allot either all parts of $T_{ser}(k-1)$ to $T_{ser}(k)$ or only fractions thereof. To do so, the PBS scheduler uses the following formula:

$$T_{ser}(k) = \begin{cases} T_{ser}(k) + T_{ser}(k-1), & \text{if } E_{avail} > (E(BAS) + \\ & T_{ser}(k-1) \cdot (E(k) - E(k-1))) \\ T_{ser}(k) + \frac{E(BAS) - E_{avail}}{E(k) - E(k-1)}, & \text{otherwise.} \end{cases} \quad (8)$$

If E_{avail} still surmounts $E(BAS)$ after $T_{ser}(k-1)$ has been completely added to $T_{ser}(k)$, the scheduler would again calculate Equation 7 for all areas $\{area(k) \in ServiceArea | k \geq 2, T_{ser}(k) > 0\}$ followed by applying Formula 8. This process repeats until eventually E_{avail} equals $E(BAS)$. As a result of the re-allocation operations, a few communication areas located within the server's broadcasting distance might not be serviced "directly" anymore, but their associated requests would be satisfied during broadcasts to areas with larger radii than their own. In what follows, we denote the set of remaining areas experiencing direct broadcast support as $ServiceArea_{direct}$, so that $ServiceArea_{direct} \subseteq ServiceArea$.

Up to this point, we only know which areas need to be broadcast to and how frequently each of those areas should be serviced. What remains to be discussed is the problem of building a concise schedule based on that information. For this purpose, the scheduler deploys Algorithm 4.1 which maintains for each $area(k) \in ServiceArea_{direct}$ an additional variable B_k representing the earliest logical time $area(k)$ is to be serviced next time. The algorithm uses a number of variables keeping track of temporary results and the algorithm's progress status, respectively. Additionally, a new parameter, termed $thresh_{re-calc}$, is introduced to represent the upper bound on the number of forthcoming BCs up to which the algorithm needs to calculate the broadcast area schedule. The intuition behind $thresh_{re-calc}$ is the observation that the distribution of mobile clients within the server's communication coverage area as well as their request behavior typically change over time and therefore the broadcast area schedule should be subject to periodic re-calculations to compensate for those changes.

```

 $\bar{T}_{ser} \leftarrow$  average value of  $T_{ser}(k)$  over  $area(k) \in ServiceArea_{direct}$ ;
 $B_k \leftarrow 0$  for all  $area(k) \in ServiceArea_{direct}$ ;
 $i \leftarrow 1$ ;
 $B_{min} \leftarrow 1$ ;
while  $i \leq thresh_{re-calc}$  do
  foreach  $area(k) \in ServiceArea_{direct}$  do
    if  $B_{min} \geq B_k$  then
       $B_{min} \leftarrow B_k$ ;
       $D_i \leftarrow k$ 
     $B_k \leftarrow B_k + \frac{\bar{T}_{ser}}{T_{ser}(k) \cdot \bar{T}_{ser}}$  for  $k \in D_i$ ;
     $B_{min} \leftarrow 1$ ;
   $i++$ 

```

Algorithm 4.1: The broadcast area scheduling algorithm.

4.2 The Data Item Selection Algorithm

The Data Item Selector's task is to choose from the pending, un-expired requests those maximizing the server's service rate based on the mapping of broadcast areas to BCs and the requests currently queued at the server. Obviously, when examining outstanding requests in order to calculate the data item schedule for the next broadcast cycle, BC_i , the scheduler would only consider those requests that come from a broadcast area served during the respective BC_i . All other requests will be ignored at that instance of time. Provided the number of requests for distinct data items queued at the server exceeds the available slots in the data item schedule of BC_i , at least two criteria should be taken into account when generating it: (a) the *timing constraints* attached to the outstanding requests and, (b) the *popularity* of the requested data items. A well-known on-demand transmission scheduling policy that bases scheduling decisions solely on the deadline criterion is Earliest Deadline First (EDF) [27, 50]. This policy, however, is only suitable for non-bandwidth-constrained broadcasting environments, that is, it provides good (i.e., optimal) performance only if the available network bandwidth capacity does suffice to meet the deadlines of all pending requests [23]. However, requests in mobile environments are typically bursty in nature and attached timing constraints may vary drastically over time [9, 14, 24, 47]. Therefore, overload situations may arise, when *deadline misses*, i.e., some data requests cannot be fulfilled within their associated expiration time periods, become unavoidable. Under such circumstances, the scheduler needs to ensure efficient utilization of the available bandwidth capacity in order to prevent a severe drop in the service rate. An effective remedy for network overload is to *batch* multiple *requests* for the same data item and handle those requests together at a single (broadcast) transmission rather than at multiple ones. Applying this approach to our scheduling problem actually implies giving frequently requested

data items a higher scheduling priority compared to those with fewer entries in the request queue. The formula by which the PBS scheduler ultimately determines the scheduling priority of a data item x , denoted $Pri(x)$, is shown below:

$$Pri(x) = \left(\sum_{\substack{\forall Req_i^k(x): \\ k \leq D_i}} \frac{1}{T_{exp}^i(x) - T_{cur}} \right) \cdot (T_{cur} - T_{old}(x)), \quad (9)$$

where $Req_i^k(x)$ refers to the i -th schedulable data request for item x submitted to the server from $area(k)$ and D_i represents the broadcast distance of the next broadcast cycle BC_i . Additionally, $T_{exp}^i(x)$ is the broadcast cycle identifier when the i -th request for data item x expires, T_{cur} stands for the current logical time which is equal to the identifier of the current BC, $T_{old}(x)$ represents the time, i.e., broadcast cycle identifier, when the oldest (either expired or un-expired) request for item x entered the request queue of the server.

$T_{old}(x)$ has been added to Equation 9 to provide *scheduling fairness* among the waiting data requests and represents the third, not yet mentioned, scheduling selection criterion of the PBS scheduler. Scheduling fairness is especially important under conditions when the broadcast server is overloaded and the request workload is largely skewed in order to prevent requests for unpopular data items from not being served at all. To ensure that scheduling starvation does not occur, the PBS scheduler keeps track of T_{old} for any requested data item x and uses its value as a means to increase the relative scheduling priority of all those data items not disseminated for a long time. In this respect, it is important to note that while PBS discards data requests that have missed their deadlines from the request queue, it remembers for any requested data item x the identifier of the corresponding BC when the first request for x arrived at the server. This value will only be reinitialized in case item x gets selected for transmission.

4.3 The Lightweight PBS Algorithm

As the time complexity of the previously proposed PBS algorithm is $\mathcal{O}(n)$ with n representing the number of distinct, schedulable data requests, there are at least two reasons why it is desirable to reduce its computational overhead: (a) As the *power consumption* of the server is an issue of concern, reduction in the number of instruction cycles required to determine the scheduling priorities of all outstanding, un-expired requests would certainly help the server to save energy resources which can then be used elsewhere, e.g., to increase the mean communication distance. (b) The amount of time required to make scheduling decisions should be minimized since the scheduling process runs within a *critical section* during which period newly arriving requests cannot be processed and thus cannot be considered for scheduling.

To reduce the overhead of the PBS algorithm, we introduce its lightweight version, called PBS_{light} , based on the following two complementary approaches:

(a) reducing the *search space* when calculating the scheduling priority of requested data items, and (b) computing *approximate priority values* within the pruned search space. Both techniques are not new and were previously discussed within the framework of the RxW scheduling algorithm [3]. However, the RxW algorithm differs from the PBS_{light} algorithm by at least two facts: (a) RxW considers only two metrics when making scheduling decisions, whereas PBS_{light} has three metrics. (b) Even more importantly, because RxW operates on a broadcast-to-broadcast tick basis, it schedules only a single data item at a time. This contradicts the PBS_{light} algorithm which is expected to run orders of magnitude less frequently than RxW since it always schedules as much data items for transmission as there are available slots in the broadcast schedule of the forthcoming BC. Therefore, the proposed optimization techniques in [3] need to be adjusted to meet PBS_{light}'s requirements. As mentioned in Section 3.1, the PBS_{light} scheduler maintains three sorted doubly-linked lists, termed ET-List, R-List, and S-List, whose entries can be referred to through a hash table. All three lists contain pairs of values with the first member of each pair representing the identifier of the respective data item and the second value being the *average expiration time* of the particular item's pending requests, \overline{T}_{exp} , (ET-List), the *number of outstanding, un-expired requests*, Req_{out} , (R-List), and the identifier of the BC within which the *oldest outstanding request*, T_{old} , arrived at the broadcast server (S-List). While ET-List and S-List are ordered in ascending order of \overline{T}_{exp} , and Req_{old} respectively, the R-List is ordered by descending Req_{out} values as shown in Figure 3.2. As can be deduced from the described list structures, the first technique used by PBS_{light} to reduce the computational and storage overhead of PBS is to *aggregate* the *absolute expiration times* of all requests for the same data item by averaging them. Clearly, using the request's median expiration time value as basis for scheduling decisions would be more accurate than the mean value, especially if time constraints are largely skewed. However, as the calculation of the former requires the scheduler to keep track of any outstanding, un-expired request, we opted for the latter solution. Additionally, note that the applied aggregation technique, although efficient in reducing the complexity of the scheduling process, obviously comes at the expense of generating imprecise, i.e., sub-optimal schedules.

The second method applied by PBS_{light} is to *reduce* the set of *scheduling candidates* out of which the scheduler is to select those providing the highest service rate to the clients without causing waiting requests to starve. PBS_{light} proceeds by dividing the calculation process into two parts and starts by computing Pri_{no-age} values using the following equation:

$$Pri_{no-age}(x) = \frac{Req_{out}(x)}{\overline{T}_{exp}(x) - T_{cur}}. \quad (10)$$

In order to avoid unnecessary computations, PBS_{light} exploits the ordering property of the ET- and R-Lists and traverses them both in alternating fashion from the top to the bottom. At each traversal step it calculates the Pri_{no-age} value of the respective data item examined and inserts the computational result

along with the identifier of the data item into a separate list, termed P-List. Similarly to R-List, P-List is descendingly-ordered by the Pri_{no-age} value. But, and in contrast to the other three lists, P-List has only a limited number of storage slots available which is equal to the number of entries in the broadcast schedule of the next BC. Once all available entry slots within P-List are occupied for the first time, two lower bound parameters $LB(\overline{T}_{exp})$ and $LB(Req_{out})$ attached to the ET- and R-Lists, respectively, are used by PBS_{light} to decide whether more data items need to be examined, since they may yield a higher scheduling benefit to the clients than any other data item currently contained within the P-List, or whether the calculation can be terminated. The parameters' values have the property of separating the ET- and R-Lists in two variably-sized parts with the bottom part known to contain data items whose respective Pri_{no-age} values are smaller than the current minimum Pri_{no-age} value of the P-List. Consequently, those data items can be excluded from the calculation. By inversion, list entries with larger Req_{out} or \overline{T}_{exp} values than their corresponding lower bounds qualify as candidates for the calculation. The actual computation of those two parameters begins after the last available storage slot in the P-List has been occupied. Assume that the last data item examined by the scheduler stems from the ET-List and the minimum Pri_{no-age} value of the P-List is Min_{Pri} . Then, the scheduler calculates $LB(Req_{out})$ of the R-List by using the following equation:

$$LB(Req_{out}) = (ET_{next} - T_{cur}) \cdot Min_{Pri}, \quad (11)$$

where ET_{next} denotes the mean expiration time of the next unexamined entry in the ET-List. If the computed $LB(Req_{out})$ value exceeds the Req_{out} value of the next entry of the R-List, the algorithm stops computing Pri_{no-age} values and proceeds the scheduling process by examining the S-List as described below. Otherwise, the next entry (if there is any) of the R-List is examined and its associated Pri_{no-age} value is calculated. If this value turns to exceed Min_{Pri} , the data item with the lowest Pri_{no-age} value is removed from the P-List and the examined data item is inserted by preserving the list's sorting order. Further, Min_{Pri} is updated, and the scheduler calculates $LB(\overline{T}_{exp})$ based on the following formula:

$$LB(\overline{T}_{exp}) = T_{cur} + \frac{R_{next}}{Min_{Pri}}, \quad (12)$$

where R_{next} denotes the number of outstanding requests of the next still unexamined data item of the R-List. The algorithm continues until one of the lower bounds is passed by the actual calculation position on any of the associated lists. At this point, Min_{Pri} is confirmed to be the upper bound on the Pri_{no-age} value of all requested data items not included into the P-List.

To illustrate the operation of the algorithm, consider the following initial situation as shown in Figure 4.3. The figure depicts the status of the scheduler's data structures after item 6 has been assigned to the last available entry slot of the P-List. At this point, the scheduler calculates the $LB(\overline{T}_{exp})$ value for the first time, which returns $10 + \frac{9}{0.5} = 28$. The scheduler goes on by examining

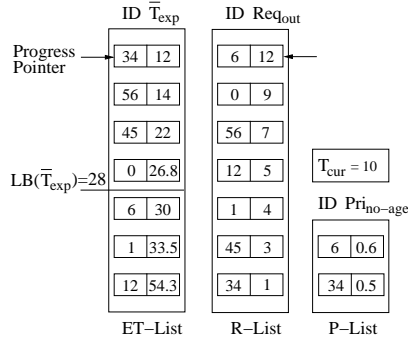


Fig. 4.3. Example of PBS_{light} determining the P-List entries.

the next entry in the ET-List whose Pri_{no-age} value is larger than the current Min_{Pri} value ($0.5 < 1.75$). As a consequence, item 54 is inserted at the top of the P-List shortly after item 34 was evicted from the list. Further, Min_{Pri} is updated to 0.6 and $LB(Req_{out})$ is computed for the first time and set to 7.2. The PBS_{light} scheduler then checks item 0 and computes its Pri_{no-age} value which turns out to be lower than Min_{Pri} ($0.54 < 0.6$). Hence, data item 0 does not qualify to be inserted into the P-List, and $LB(\bar{T}_{exp})$ is updated to 21.7. Since \bar{T}_{exp} of the next entry in the ET-List (item 45) is larger than the computed $LB(\bar{T}_{exp})$ ($22 > 21.7$), the algorithm terminates and exits at that point.

The last technique applied by PBS_{light} to reduce the scheduling costs is enforced at the computational stage and is aimed at solving the request starvation problem. To this end, the scheduler maintains for each requested data item x the logical time, T_{old} , of the item's oldest, but not necessarily schedulable, request in the S-List, and computes its priority value, $Pri(x)$, according to the following formula:

$$Pri(x) = Pri_{no-age}(x) \cdot (T_{cur} - T_{old}(x)). \quad (13)$$

To reduce the computational costs incurred by calculating priority values, the PBS_{light} scheduler computes them only for two (not necessarily disjoint) sets of items. The first set consists of all the data items currently stored within the P-List and the second comprises all those requested data items that have not been broadcast for a long time. Thereby, what time actually is considered as long, is configurable by the server through a time threshold parameter, T_{thresh} , whose value represents the period of time after which outstanding requests are considered to suffer from starvation. T_{thresh} can be set to any value from T_{cur} to some point in time in the past, and the PBS_{light} scheduler will compute priority values only for those data items of the S-List with T_{old} at least as old as T_{thresh} . Consequently, the lower T_{thresh} is set, the fewer Pri values need to be computed, but the longer clients might need to wait until less frequently requested items appear on the broadcast channel. The calculation of priority values starts by first considering the data items listed in the P-List and is continued by examining all those data items recorded in the S-List for which the condition $T_{old} \leq T_{thresh}$

holds and for which there exists at least one outstanding, un-expired request in the server queue. If some item x in the S-List ($x \notin$ P-List) has a higher Pri value than any other item in the P-List, x is inserted into the latter list causing the item with lowest Pri value to be discarded. Once the scheduler has completed the calculation, the P-List contains the data items with the highest calculated priority values and those will be broadcast in the next BC.

5 Experimental Results

In this section, we describe results of an extensive simulation study conducted to evaluate the performance of the PBS and PBS_{light} schedulers. We also compare PBS's performance with other scheduling algorithms. Let us begin by presenting the simulator's setup and workload model.

5.1 System Model and Workload Parameters

To simulate an on-demand data broadcasting system accurately, it is necessary to model all the components which significantly impact the scheduling performance. The modeled simulation system is similar to the one presented in Subsection 3.1 and is therefore only briefly described below. The broadcast server including the *broadcast scheduler* lies at the heart of the simulator and has the capability to send requested data items to 10 broadcast areas each one 1 radius step apart from the other area starting with radius $1r$ and the channel path loss exponent α is set to 3. In the default setup of the simulator, the amount of energy available to the server is restricted to $4 \cdot 10^7$ units and the expected service period amounts to 1,000 BCs. The amount of energy required to broadcast one data item to *area*(1) is set to 1 unit, i.e., the server consumes $1 \cdot 10^3$ units to transmit an item to *area*(10). The broadcast server maintains a database of 10,000 items in size and 100 data items are disseminated within each broadcast cycle. The assignment of broadcast distances to broadcast cycles is periodically re-calculated at intervals of 50 BCs. Mobile clients are not modeled as separate facilities within the simulator; instead their behavior is simulated through a *request generator* which produces a sequence of data requests. Each request contains information on its *spatial origin*, the *identifier* of the requested data item, and its *expiration time*. Thereby, the generated values for those three parameters are derived by using three different statistical distributions described below.

The probability of a specific data item being requested follows a Zipf distribution [52] with parameter θ set to the default value of 0.8, meaning that approximately 75% of all requests apply to 25% of the database which approximately equals the data reference behavior experienced in production database systems [18]. Note that in Zipf distribution, the probability of accessing the i -th most popular object is:

$$p_i = \frac{1}{i^\theta \cdot \sum_{j=1}^N \frac{1}{j^\theta}}, \quad (14)$$

where N is the number of objects in the database and θ is the skewness parameter [5]. To illustrate the sensitivity of PBS and the other scheduling policies studied to changes in the frequency distribution of data requests sent to the server, we varied θ and reduced it from 0.8 to 0.5.

The probability of a request to originate from $region(x-1, x)$, denoted $p^r(x)$, approximates a Zipfian distribution and is computed as follows:

$$p^r(x) = \frac{(10 - |x - RRC|)^\Delta}{\sum_{i=1}^{10} (10 - |i - RRC|)^\Delta}, \quad x = 1, 2, \dots, 10, \quad (15)$$

where Δ determines the skewness of the distribution in the request frequencies of various broadcast regions and RRC (standing for Region Request Center) denotes the identifier of the $region(y-1, y)$ with the highest probability of issuing requests. Note that if Δ equals 0, the regional request frequency distribution downgrades to a uniform distribution and the value of $p^r(x)$ behaves reversely proportional to the value of $|x - RRC|$. Throughout our experiments, Δ is set to 0.8 and RRC to 1, i.e., almost 80% of the data requests come from only 20% of the regions and those are located within close proximity to the server.

The request's expiration time of $region(x-1, x)$ follows a uniform distribution in the range $[\frac{1}{2}\overline{T}_{exp}^r(x), \frac{3}{2}\overline{T}_{exp}^r(x)]$, where $\overline{T}_{exp}^r(x)$ represents the region's mean request expiration time. To model the assumption that mobile clients located at farther distances from the server may have weaker timing constraints, $\overline{T}_{exp}^r(x)$ is set to $\overline{T}_{exp}^r(1) \cdot x^\beta$ where β specifies the relative difference in the mean request expiration time between the broadcast regions. In the basic experiments, β is set to 0.5 and $\overline{T}_{exp}^r(1)$ is set to 4 BCs. The inter-arrival time between two consecutive data requests is modeled as a Poisson distribution with the mean rate λ set to 40 requests per broadcast tick. The main parameters and workload settings of the simulator are once more summarized in Table 5.1.

5.2 Experiment 1: Service Rate and Power Usage

In the first experiment, we investigated the impact of the *amount of energy available* to the server on its *service rate* when using PBS as scheduling policy along with three variants of the PBS scheduler which do not dynamically adjust their broadcasting activity according to the server's energy resources but which rather continuously transmit the requested data to a predefined area. The PBS variant that steadily broadcasts data only to $area(1)$ is termed $PBS_{fixed}(1)$, whereas the other two policies each servicing $area(7)$ and $area(10)$, respectively, are called $PBS_{fixed}(7)$ and $PBS_{fixed}(10)$. The performance results of the experiment are shown in Figure 5.4. As the service period of the server is set to 1,000 BCs and each BC has a length of 100 broadcast ticks, it takes 100,000 units of energy to serve $area(1)$ during the server's service period. If the amount of energy at the server's disposal is less than, or equal to, 100,000 units, the PBS scheduler broadcasts data items to $area(1)$ until its energy is dissipated. Therefore, and as the results in Figure 5.4 show, the performance results of PBS and $PBS_{fixed}(1)$ are identical under such conditions. If the system has more than 100,000 units of

Parameter	Description	Default Value	Sensitivity Range
λ	Mean request arrival rate (requests per broadcast tick)	40	-
DB_{Size}	Database size (number of data items)	10,000	[1,000-10,000]
T_{ser}	Estimated service period (broadcast cycles)	1,000	-
E_{avail}	Amount of energy available to the server	$4 \cdot 10^7$	$[10^3-10^9]$
θ	Skewness in the frequency distribution of data references	0.8	[0.5, 0.8]
Δ	Skewness of the distribution in the requests' origins	0.8	-
RRC	Region with the highest probability of issuing requests	1	[1-10]
$\bar{T}_{exp}^r(1)$	Mean expiration time of $region(0, 1)$	4	-
β	Exponential weight factor	0.5	[0-2]

Table 5.1. System and workload parameters.

energy available, the service rate of PBS increases significantly and it performs at least as well or even considerably better than any static PBS variant. Further, the results shows that the service rate increase of $PBS_{fixed}(1)$ and $PBS_{fixed}(7)$ policy tops out before the maximum of 1 is reached even if energy resources are available in abundance. The reason is, obviously, related to the communication distance constraint being imposed on both policies. Therefore, if the available energy exceeds a certain threshold value, the service rate does not increase anymore. The number of energy units used by the broadcast server as a function of time is shown in Figure 5.5. As can be seen in the diagram, the deployment of the $PBS_{fixed}(10)$ scheduling policy incurs high energy costs and therefore, the server's energy reserves last only for a period of about 400 BCs. Further, both

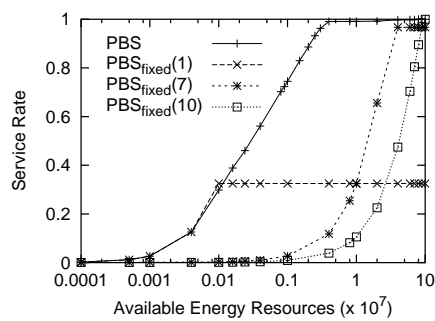


Fig. 5.4. Impact of the available energy resources on the service rate with and without adjustment of the broadcast distance.

the $\text{PBS}_{\text{fixed}}(1)$ and $\text{PBS}_{\text{fixed}}(7)$ policies do not completely use their available energy resources during the service period of the broadcast server which contrasts the PBS scheduler.

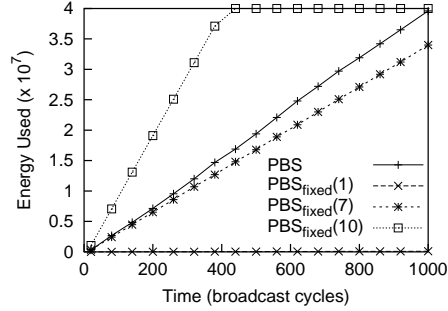


Fig. 5.5. Energy usage: Amount of energy consumed as a function of time.

5.3 Experiment 2: Effect of the Distribution of the Requests' Origins and β on the Service Rate

To study the effect of different *distributions in the origins* of the *client requests* on the scheduling performance of PBS, the *RRC* parameter is first set to 1, 4 and then to 10. As expected and can be seen in Figure 5.6(a), when *RRC* is equal to 1, i.e., the majority of the data requests come from regions with close proximity of the server, the service rate of PBS reaches its optimum with the lowest energy consumption for all examined distributions. Additionally, the more requests come from farther distances of the server, the more energy is required to satisfy them and to achieve an acceptable service rate. If *RRC* is set to 10, i.e., about 75% of the total number of requests come from *area(9)* and only about 8% stem from *area(4)*, the energy consumption required for providing the same service rate is on average about one order of magnitude higher than if *RRC* is equal to 1.

In another set of experiments, we varied the setting of β from 0 to 1.5 in increments of 0.5 to investigate the effect of the *tightness* of the *requests' time constraints* on the schedulers performance. The results are shown in Figure 5.6(b), and, as expected, the service rate improves with increasing mean request expiration time. The reason is that the longer the time period to satisfy outstanding requests, the more requests for the same data item can be batched and satisfied together, thus improving the network bandwidth capacity and, consequently, the service rate.

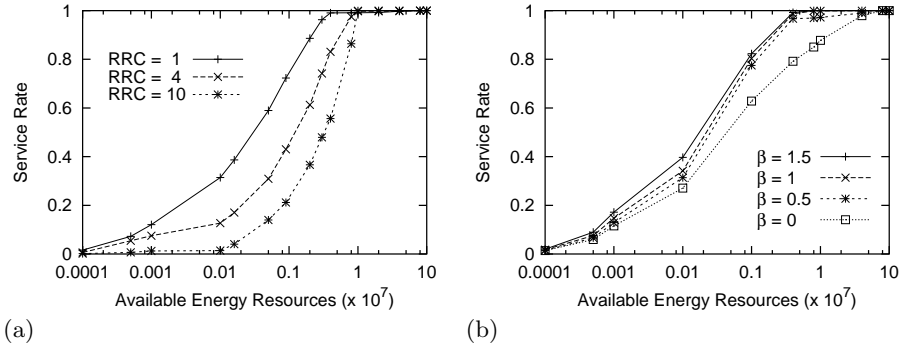


Fig. 5.6. (a) The impact of varying RRC and (b) β on the performance of PBS.

5.4 Experiment 3: Varying the Request Expiration Time

In the previous series of experiments, the impact of changing the mean expiration time on the performance of PBS was investigated. To this end, the simulator was run with three distinct values of the parameter β . However, in those experiments β remained unchanged throughout the simulation runtime, i.e., timing constraints associated with requests of the same spatial region were constant. To change this and to examine the adaptiveness of PBS to the changes in the *mean request expiration time*, we conducted additional experiments by dynamically varying the setting of β every 50 broadcast cycles. This was done by setting the request generator to periodically change β according to a uniform distribution in the range from 0 to β' where β' is a decimal value set at increments of 0.5 in the interval of $[-2, 2]$. To quantify the performance penalty experienced in environments with fluctuating request deadlines, where the broadcast area schedule is not periodically re-calculated, we ran the simulator using two different PBS versions: (a) The first variant determines the BAS only once at the simulator initialization time and is called PBS_{once} . (b) The second version represents the original PBS algorithm as proposed in Subsection 4.1, and periodically re-calculates the BAS whenever β' is changed. The performance results of the experiments are shown in Figure 5.7. For comparison reasons, the diagram additionally contains experimental results of $PBS_{\text{fixed}}(10)$.

The results nicely confirm the findings of prior experiments that the service rate increases if deadline constraints imposed on requests are relieved. Further, it can be observed that the relative benefit of re-calculating BASs grows with increasing request expiration times. For example, if $\beta' = 2$, the service rate of PBS is significantly higher (by about 5%) than that of PBS_{once} . For much lower values of β' (≤ 0), however, the relative performance difference between both algorithms is negligible. Additionally, PBS and PBS_{once} significantly outperform $PBS_{\text{fixed}}(10)$, and more interestingly, the absolute and relative performance gap widens with increasing request expiration time. The reason for $PBS_{\text{fixed}}(10)$'s underperformance is the fact that the broadcast server runs out of energy at

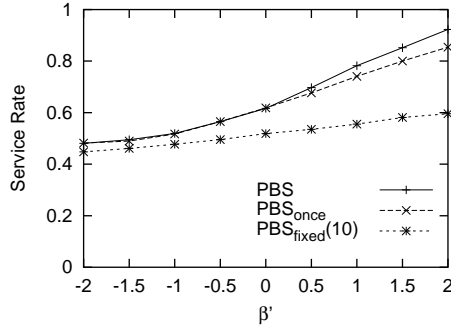


Fig. 5.7. Performance impact on PBS, PBS_{once}, and PBS_{fixed}(10) if β is varied periodically.

some point thus causing the service rate to drop to zero for the rest of the simulated period. Therefore, PBS_{fixed}(10) benefits only disproportionately from an increase in the request expiration time.

5.5 Experiment 4: Performance Impact of Avoiding Request Starvation

One of PBS’s objectives is preventing requests from “starving” while waiting to be served. However, and as argued before, its enforcement penalizes the system’s performance and, therefore, needs to be quantitatively investigated. To this end, we conducted experiments by using, on the one hand, the original PBS scheduler for making scheduling decisions and, on the other hand, a variant of PBS, called PBS_{no-age}, that builds the data item schedule on the basis of the requested items’ *Pri_{no-age}* values, i.e., it does not favor data items with long-outstanding requests for scheduling. In order to evaluate the performance of both schedulers under various BASs, we varied the amount of energy available to the server from 1,000 to 10^7 units.

The experimental results, shown in Figure 5.8(a), confirm the expectation that PBS_{no-age}’s performance would be superior to that of PBS. Moreover, it can be seen that if the server operates under tight energy constraints ($< 10^6$ units), the observable performance gap between both schedulers is only marginal since only a small portion of the available network capacity will be used by PBS to serve “cold” data items. However, as more energy is provided to the server, long-range broadcast communication becomes dominant ultimately leading to a significant increase in the service rate. As a side effect of longer communication distances, the scheduling search space grows in size, i.e., potentially more data items need to be examined for computing optimal broadcast schedules. Additionally, the portion of requested data items detected as starvation victims, and qualified for broadcasting increases resulting in a relative performance drop of PBS compared to PBS_{no-age}. The performance difference peaks at the

point where the available server energy approaches $2 \cdot 10^6$ units reaching approx. 5%. Thereafter, the performance gap gradually falls back as the service rate approaches its optimal level thus eliminating the starvation problem.

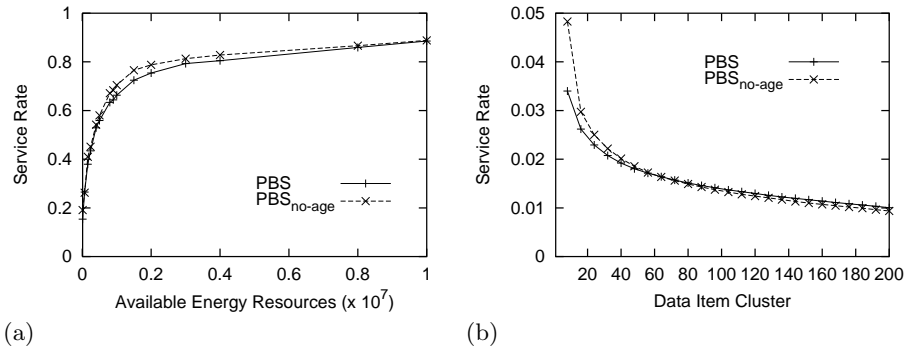


Fig. 5.8. Performance penalty for solving the starvation problem: (a) Service rate of PBS and PBS_{no-age} as a function of the amount of available server energy and (b) service rate of PBS and PBS_{no-age} broken down by data item clusters.

Although the performance of PBS is on average about 2% lower than that of PBS_{no-age}, on a median basis, however, it might provide even superior scheduling results. The reason is that PBS treats outstanding un-expired requests more fairly than PBS_{no-age} as the former tends to schedule less frequently requested data items more often. As a consequence, and as can be observed in Figure 5.8(b), the service rate provided by PBS for the majority of data items is slightly higher than the one provided by PBS_{no-age}. Note that the results presented in Figure 5.8(b) were generated by clustering the items of the database (10,000) into 200 groups according to their request probabilities (i.e., items with similar request probabilities are grouped into one cluster). Further, note that the clusters were assigned to the x-axis in descending order of their reference probabilities.

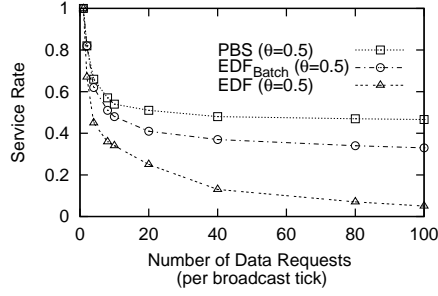
5.6 Experiment 5: Comparative Performance of PBS against other proposed Algorithms

In this section, we investigate the performance of PBS in comparison to other broadcast scheduling algorithms which either do or do not consider time constraints. Related scheduling policies of the first group are EDF and EDF_{Batch} [27, 50]. Remember EDF is a policy that always selects the items with the *earliest expiration times* for broadcasting and ignores no longer schedulable requests. EDF_{Batch} builds upon EDF extending it by employing *batching* to serve multiple requests of the same item via a single transmission. We conducted the *comparative simulations* in an energy-unconstrained environment setting the server's

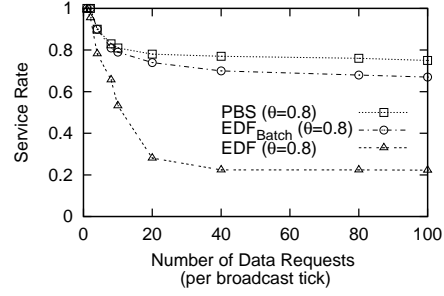
energy resources to $1 \cdot 10^9$ units, i.e., the server is able to broadcast to $area(10)$ throughout the simulation runtime. We again use the *service rate* as performance measure and gather the performance results by varying the mean request arrival rate in the range of 1 to 200. To investigate the policies sensitivity to the skewness in the frequency distribution of the data references, we set the Zipfian parameter θ to 0.5 and 0.8, respectively.

As can be observed from Figures 5.9(a) and (b), the performance of EDF and EDF_{Batch} is, on average, about 74% and 18%, respectively, lower than that of PBS. Additionally, and in accordance to the results presented in [50], the relative performance penalty of using EDF instead of EDF_{Batch} averages around 70%. The reason of PBS outperforming the other two scheduling policies is that the latter base scheduling decisions merely on the criteria of the requests' expiration times and do not include information on the number of outstanding, un-expired requests referring to the same data items. As a consequence, EDF and EDF_{Batch} always select the data items with the tightest deadline constraints for broadcasting and those items are likely to provide either no or only a minor batching effect. Therefore, in environments with non-uniform data request patterns, PBS tends to use the available broadcasting bandwidth much more efficiently than the other two scheduling approaches. Further, and as noted in [50], EDF_{Batch} is superior to EDF since after broadcasting the scheduled items, it removes other requests for the same data items from the request queue. Figures 5.9(a) and (b) also illustrate that the relative performance gap between the investigated scheduling policies is narrowing as the frequency distribution in the data requests becomes more skewed. For example, the relative performance difference between PBS and EDF_{Batch} diminishes from about 27% to 10% by increasing the parameter value θ from 0.5 to 0.8.

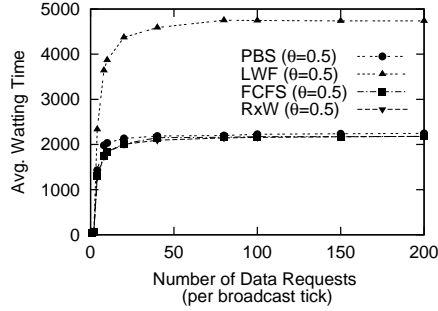
Prominent representatives of non-real-time scheduling policies are FCFS (First-Come-First-Served), LWF [13], and RxW [3]. Since all three algorithms were not devised to operate in an energy-limited environment, and in order to ensure a fair performance comparison with PBS, we again modeled an energy-unconstrained server. This time, however, we opt for the *average waiting time* to serve as comparison metric as it is traditionally used in performance studies of non-real-time schedulers. The simulation results are illustrated in Figures 5.9(c) and (d). Since the FCFS scheduler chooses data items for broadcasting irrespective of their access probabilities, it performs poorly compared to the other three algorithms. LWF and RxW provide the best performance with LWF outperforming RxW by a few decimal percentage points. PBS performs slightly worse than LWF and RxW and its average waiting time is, on average, about 4% higher than that of the other two algorithms. The reason for PBS's underperformance is that it is primarily concerned with maximizing the service rate and only secondarily with minimizing the average waiting time, which causes a minor, but nevertheless noticeable performance loss. Besides, and contrary to previous experiments, the relative performance difference between the studied scheduling policies now widens in terms of the



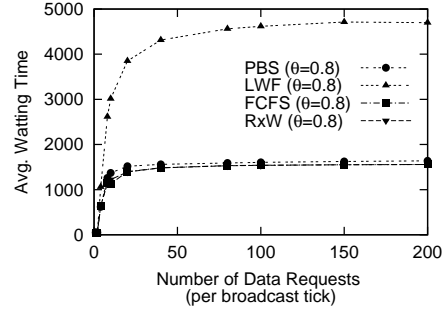
(a) Zipfian parameter $\theta = 0.5$



(b) Zipfian parameter $\theta = 0.8$



(c) Zipfian parameter $\theta = 0.5$



(d) Zipfian parameter $\theta = 0.8$

Fig. 5.9. Performance comparison: Service rate ((a) and (b)) and average waiting time ((c) and (d)) of various scheduling policies by varying the frequency distributions in the data requests.

average amount of time it takes the server to reply on data requests as the skewness in the frequency distribution of the data requests increases.

5.7 Experiment 6: The Performance of PBS vs. PBS_{light}

Besides the scheduling performance, an essential criteria for evaluating scheduling policies is their processing overhead. PBS_{light} has been proposed as a computationally less intensive alternative to PBS. In what follows, we examine PBS_{light}'s efficiency in reducing the scheduling overhead as well as its performance characteristics. We proceed by varying the database size in the range from 1,000 to 10,000 and evaluate the scheduling overhead of PBS vs. PBS_{light}. To observe the effect of T_{thresh} on the scheduling performance and on the overhead of PBS_{light}, we run the experiments with T_{thresh} set to $T_{cur} - \{0, 10, 100, 1, 000\}$. In what follows, and for the sake of simplicity, we refer to the different variants of PBS_{light} by using the shortcut $T_{thresh} = \{0, 10, 100, 1, 000\}$. The results are shown in Figure 5.10 and demonstrate that the scheduling costs of both PBS and PBS_{light} grow only sub-linearly with increasing database size. PBS has, as

expected, the highest scheduling overhead, followed by $\text{PBS}_{\text{light}}$ with T_{thresh} set to 0, 10, 100, and 1,000, respectively. As PBS is a computationally exhaustive algorithm, its scheduling overhead is proportional to the number of distinct pending requests. Consequently, under long periods of heavy load there exists for almost any item of the database at least one outstanding request in the request queue, and, therefore, the computational overhead may become a system bottleneck. $\text{PBS}_{\text{light}}$ exploits *pruning* in order to reduce the *candidate search space* for calculating maximal *Pri_{no-age}* values, computes *approximate Pri_{no-age}* values and may exclude parts of the S-List from calculating *Pri* values by means of the T_{thresh} parameter. The results show that when applying the first two measures alone (i.e., if $T_{\text{thresh}} = 0$), an average savings of about 67% at a load of 40 requests per broadcast tick is achieved. The savings in the computational overhead can even be increased up to a maximum of 98%, if S-List entries are completely left-out of the calculation (i.e., if $T_{\text{thresh}} = 1,000$).

However, and as can be seen in Figure 5.10(a), the reduction in the scheduling overhead of $\text{PBS}_{\text{light}}$ does not necessarily come for free, but may rather be accompanied by a drop in the service rate. For example, if the ratio of the number of items scheduled per broadcast cycle to the size of the database is relatively high (≥ 0.03), PBS slightly outperforms $\text{PBS}_{\text{light}}$, independent of the setting of the T_{thresh} parameter. However, if only a small portion of the database is broadcast per cycle, $\text{PBS}_{\text{light}}$ performs superior to PBS. To interpret the observed performance results, it is important to remember that $\text{PBS}_{\text{light}}$'s scheduling performance is dwarfed by imprecision arising from calculating *Pri* values on the basis of averaged and not the detailed request deadline data. On the other hand, however, $\text{PBS}_{\text{light}}$ is able to trade-off scheduling performance for scheduling fairness via the T_{thresh} parameter. Therefore, and in close dependence on the actual T_{thresh} setting, the performance of $\text{PBS}_{\text{light}}$ may even be superior to that of PBS.

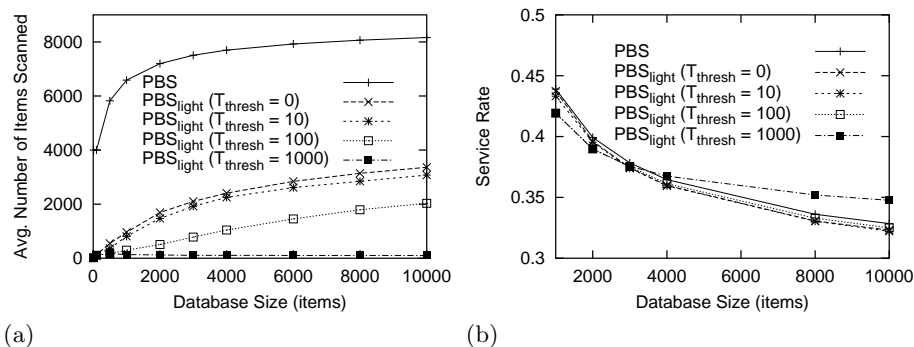


Fig. 5.10. (a) Scheduling overhead and (b) scheduling performance of PBS and $\text{PBS}_{\text{light}}$.

6 Conclusion and Future Work

In this paper, we have studied the issue of efficiently scheduling time-constrained data requests in an energy-limited data broadcast environment. Scheduling on-demand data broadcast requests by an energy-constrained data broadcast server imposes many new research issues such as when and how frequently the server should service the various differently-sized circular communication areas centered around the broadcast server. Additionally, the problem of making scheduling decisions aimed at maximizing the service rate of the server needs to be addressed. As a response to those algorithmic challenges, we proposed a power-aware broadcast scheduling policy called PBS. It consists of two complementary parts: (a) the *Broadcast Area Selector* and (b) *Data Item Selector*. While the former solves the issue of mapping broadcast communication distances to broadcast cycles, the latter is responsible for constructing data item schedules. Besides, we designed a lower-overhead, approximate algorithm for PBS, termed PBS_{light}. It differs from PBS by cutting down the candidate search space when looking for requested data items with maximal priority values and also by relying on the parameter T_{thresh} to trade-off scheduling overhead for starvation fairness. To evaluate the performance of both algorithms and to compare them to the previously proposed scheduling policies, an extensive simulation study was conducted. The performance analysis showed that PBS significantly outperforms state-of-the-art real-time scheduling algorithms such as EDF and EDF_{Batch} and only slightly underperforms LWF and RxW designed to schedule non-real-time requests in terms of the average request waiting time measure. Further, we observed that the performance of PBS_{light} is comparable to that of PBS and may even be superior if T_{thresh} is chosen so as to allow only a few entries of the S-List to participate in the priority calculation.

There remain issues to be addressed in the future which may contribute to a performance improvement of the broadcast server. To increase the size of the area to which broadcast services are provided, *multiple broadcast servers* may be set up with partially overlapping communication coverage areas. To avoid that two or more energy-constrained broadcast servers disseminate the same data items to a commonly served region at equal times, synchronization between the broadcast area and data item schedules of the involved servers is required. As a result of the collaboration, the overall average response time and the service rate of the broadcast servers may be improved. Another possible performance enhancement measure aimed at reducing the fraction of client requests that miss their deadlines is by means of establishing a *MANET* between the clients. In MANETs, mobile clients can communicate directly with each other without the need of contacting a base station provided that they are located within each others communication coverage area. MANETs can be used to extend the communication capabilities of an energy-constrained broadcast server in various ways: (a) As the communication coverage area of an energy-limited broadcast server is typically restricted and varies periodically, mobile clients located at the border of the server's communication area can act as *relay nodes* to transfer requested data items to clients currently not within the server's broadcast radius;

(b) Rather than requesting any missing data item from the broadcast server, data requests can now also be transmitted to neighboring mobile clients in the hope that those either cache the requested data item themselves or can relay the request to some other mobile client which would be able to satisfy it. As a consequence, potentially fewer data requests need to be transferred to the broadcast server reducing the length of its request queue and ultimately increasing its service rate since more requests are likely to be processed before their deadlines are exceeded.

References

1. S. Acharya. Broadcast Disks: Dissemination-based Data Management for Asymmetric Communication Environments. Tech. Rep. CS-97-15, Brown University, Department of Computer Science, 1997.
2. S. Acharya and S. Muthukrishnan. Scheduling On-Demand Broadcasts: New Metrics and Algorithms. *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 43–54. ACM Press, 1998.
3. D. Aksoy and M. J. Franklin. RxW: A Scheduling Approach for Large-Scale On-Demand Data Broadcast. *TON* 7(6):846–860, 1999.
4. A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau. Profile-based Dynamic Voltage Scheduling using Program Checkpoints, 2002.
5. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. *Infocom 1999*, pp. 126–134, 1999.
6. E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. *Proceedings of the 17th Annual International Conference on Supercomputing*, pp. 86–97, 2003.
7. M. S. Chen, K. L. Wu, and P. S. Yu. Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing. *IEEE TKDE* 15(1):161–173, 2003.
8. Cisco Systems Inc. Cisco Aironet 350 Series Specifications, Nov 2004, <http://www.cisco.com/univercd/cc/td/doc/pcat/ao350ca.htm>.
9. P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Dissemination of Dynamic Data. *Proc. ACM SIGMOD Conf.*, p. 599, 2001.
10. F. Douglass, R. Caceres, M. F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber. Storage Alternatives for Mobile Computers. *Operating Systems Design and Implementation*, pp. 25–37, 1994.
11. F. Douglass, P. Krishnan, and B. N. Bershad. Adaptive Disk Spin-down Policies for Mobile Computers. *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, pp. 121–137, 1995.
12. F. Douglass, P. Krishnan, and B. Marsh. Thwarting the Power-Hungry Disk. *USENIX*, pp. 292–306, 1994.
13. H. D. Dykeman, M. H. Ammar, and J. W. Wong. Scheduling Algorithms for Videotex Systems under Broadcast Delivery. *ICC 1986*, pp. 1847–1851, 1986.
14. M. J. Franklin and S. B. Zdonik. Dissemination-Based Information Systems. *IEEE Bulletin of the Technical Committee on Data Engineering* 19(3):20–30, September 1996.
15. R. A. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. *USENIX*, pp. 201–212, 1995.

16. K. Govil, E. Chan, and H. Wasserman. Comparing Algorithm for Dynamic Speed-Setting of a Low-Power CPU. *Mobile Computing and Networking*, pp. 13–25, 1995.
17. R. Govindan, T. Faber, J. Heidemann, and D. Estrin. Ad hoc Smart Environments. *Proceedings of the DARPA/NIST Workshop on Smart Environments*, 1999.
18. W. W. Hsu, A. J. Smith, and H. C. Young. Characteristics of Production Database Workloads and the TPC Benchmarks. *IBM Systems Journal* 40(3):781–802, 2001.
19. T. Imielinski, S. Viswanathan, and B. R. Badrinath. Power Efficient Filtering of Data an Air. *EDBT 1994*, pp. 245–258, 1994.
20. T. Imielinski, S. Viswanathan, and B. R. Badrinath. Scheduling Data Broadcast in Asymmetric Communication Environments. *Knowledge and Data Eng., IEEE Trans.* 9(3):353–372, 1997.
21. E.-S. Jung and N. H. Vaidya. A Power Control MAC Protocol for Ad Hoc Networks. *MobiCom*, pp. 36–47, 2002.
22. J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next Century Challenges: Mobile Networking for “Smart Dust”. *MobiCom 1999*, pp. 271–278, 1999.
23. T. W. Lamyz, T. W. J. Nganzz, and K. K. To. Performance Guarantees for EDF under Overload. *Journal of Algorithms* 52(2):193–206, 2004.
24. E. R. Lassetre. Olympic Records for Data at the 1998 Nagano Games. *SIGMOD 1998*, p. 537, 1998.
25. P. Lettieri, C. Fragouli, and M. B. Srivastava. Low Power Error Control for Wireless Links. *MobiCom*, pp. 139–150, 1997.
26. K. Li, R. Kumpf, P. Horton, and T. E. Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. *USENIX*, pp. 279–291, 1994.
27. C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM* 20(1):46–61, 1973.
28. B. Marsh, F. Dougliis, and P. Krishnan. Flash Memory File Caching for Mobile Computers. *Proceedings of the 27th Hawaii Conference on Systems Science*, 1994.
29. Microsoft Corporation. DirectBand Network. Microsoft Smart Personal Objects Technology (SPOT), 2004, <http://www.microsoft.com/resources/spot/>.
30. R. Min and A. Chandrakasan. Energy-Efficient Communication for Ad-Hoc Wireless Sensor Networks. *35th Asilomar Conference on Signals, Systems, and Computers*, pp. 139–143, 2001.
31. J. Monks, V. Bharghavan, and W. W. Hwu. A Power Controlled Multiple Access Protocol for Wireless Packet Networks. *Infocom*, pp. 219–228, 2001.
32. B. Narendran, J. Sienicki, S. Yajnik, and P. Agrawal. Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems. *ICC*, pp. 349–355, 1997.
33. K. Pahlavan and A. Levesque. *Wireless Information Networks*. Wiley-Interscience, 1995.
34. P. Penna and C. Ventre. Energy-efficient Broadcasting in Ad-Hoc Networks: Combining MSTs with Shortest-Path Trees. *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 61–68, 2004.
35. P. Pillai and K. G. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. *18th ACM Symposium on Operating Systems Principles*, 2001.
36. J. G. Proakis. *Digital Communications*. McGraw Hill, 4th edition, 2000.
37. K. Ramamritham. Real-Time Databases. *Distributed and Parallel Databases* 1(2):199–226, 1993.
38. S. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Radio Networks using Transmit Power Adjustment. *Infocom*, pp. 404–413, 2000.

39. C. Schurgers, O. Aberthorne, and M. Srivastava. Modulation Scaling for Energy Aware Communication Systems. *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pp. 96–99. ACM Press, 2001.
40. E. Shih, S. Cho, F. S. Lee, B. H. Calhoun, and A. Chandrakasan. Design Considerations for Energy-Efficient Radios in Wireless Microsensor Networks. *J. VLSI Signal Process. Syst.* 37(1):77–94, 2004.
41. N. Shivakumar and S. Venkatasubramanian. Energy Efficient Indexing for Information Dissemination in Wireless Systems. *MONET* 1(4):433–446, 1996.
42. C. J. Su, L. Tassiulas, and V. J. Tsotras. Broadcast Scheduling for Information Distribution. *ACM Wireless Networks* 5(2):137–147, 1999.
43. K. L. Tan and J. X. Yu. Energy Efficient Filtering of Nonuniform Broadcast. *16th International Conference on Distributed Computing Systems*, pp. 520–528, 1996.
44. A. Wang, S. Cho, C. Sodini, and A. Chandrakasan. Energy Efficient Modulation and MAC for Asymmetric RF Microsensor Systems. *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pp. 106–111, 2001.
45. M. Weiser, B. Welch, A. J. Demers, and S. Shenker. Scheduling for Reduced CPU Energy. *Operating Systems Design and Implementation*, pp. 13–23, 1994.
46. J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. *Infocom 2000*, pp. 585–594, 2000.
47. O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases* 7(3):257–287, 1999.
48. J. W. Wong. Broadcast Delivery. *Proceedings of the IEEE*, pp. 1566–1577, 1988.
49. J. Xu, W.-C. Lee, and X. Tang. Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air. *MobiSys 2004*, pp. 153–164, 2004.
50. P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, and K. Ramamritham. Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments. *RTAS 1997*, pp. 38–48, 1997.
51. J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling Hard-disk Power Consumption. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, pp. 217–230, 2003.
52. G. K. Zipf. *Human Behavior and Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, 1949.