An Energy-Conserved On-Demand Data Broadcasting System

Jiun-Long Huang
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail:jlhuang@arbor.ee.ntu.edu.tw

Wen-Chih Peng
Dept. of Computer Sci. and Info. Eng.
National Chiao-Tung University
Hsinchu, Taiwan, ROC
E-mail:wcpeng@csie.nctu.edu.tw

ABSTRACT

We propose in this paper an energy-conserved on-demand data broadcasting system by employing the data indexing technique. We also propose algorithm AIDOA to adjust the degree of buckets according to system workload. Experimental results show that algorithm AIDOA is able to greatly reduce power consumption at the cost of slight increment in average access time and adjust the index and data organization dynamically to adapt to change of system workload.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems

General Terms

Algorithms, performance

Keywords

Data indexing, on-demand data broadcasting, energy conservation, mobile information system

1. INTRODUCTION

As shown in [5][6], only a modest improvement (about 20% $\sim$ 30%) in battery lifetime is expected in the next few years. Hence, energy conservation is raised as a key factor of the design of mobile devices. Most devices can operate in two modes: *active* mode and *doze* mode. Many studies show that the power consumed in active state is much higher than that consumed in doze mode. As a consequence, the mobile devices should stay in doze mode as long as possible to reduce power consumption.

To evaluate the effect of data indexing algorithms on energy conservation, *tuning time*, which is defined as the time that a mobile device operates in active mode in order to retrieve a data item, is introduced in [3]. Since employing data indexing will unavoidably introduce some overhead in access time[1], the data indexing algorithms should reduce tuning time as much as possible at the cost of producing an acceptable increment in access time. Fortunately, since the size of an index item is usually much smaller than that of a data item, such increment is usually small.

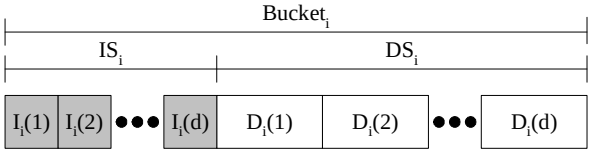


Figure 1: Index structure

The authors in [4] proposed an indexing approach for on-demand data broadcast systems. As shown in Figure 1, the proposed broadcast program is made up of a series of buckets and each bucket consists of one index segment and one data segment. The number of data items in a bucket is called the *degree* of the bucket. A data segment contains a series of data items, and an index segment consists of the index items of the data items in the corresponding data segment. The information in an index item, say $I_i(1)$, consists of the identifier of the corresponding data item $D_i(1)$, the data size of $D_i(1)$ and the time that $D_i(1)$ in bucket i will be broadcast on the broadcast channel. In addition, by the information in the current index segment, a mobile device is able to determine the broadcast time of the index segment of the next bucket.

Although it shows in [4] that inserting index items into the broadcast program is able to significantly reduce the average tuning time at the cost of slight increment in average access time, however, the proposed data indexing method proposed in [4] has the following drawbacks:

- *Does not consider power consumption of turning on and turning off the WNIs.*
- *Does not adapt to change of system workload*

[1]Average access time is defined as the average time elapsed from the moment a client issues a query to the point the desired data item is read.

MDM 2005 05 Ayia Napa Cyprus

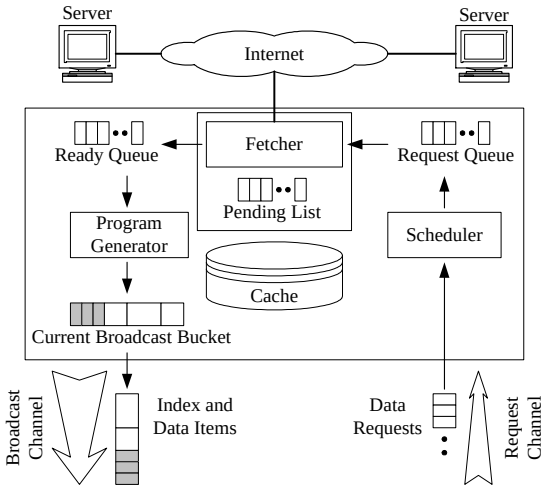


Figure 2: System architecture

In view of this, we propose in this paper an energy-conserved on-demand data broadcasting system by employing the data indexing technique. Different from the prior work on data indexing on on-demand data broadcasting, power consumption of turning on and turning off the WNIs is considered. We first analyze the access time and tuning time of data requests and propose algorithm AIDOA to adjust the degree of buckets according to system workload. In essence, algorithm AIDOA consists of two phases, statistics collection phase and adjustment phase, and switches back and forth between these two phases. The system collects some statistic information of all served data requests in statistics collection phase, and the information is used to adjust the degree of buckets in adjustment phase according to the derived analytical results. Several experiments are then conducted to evaluate the performance of algorithm AIDOA. Experimental results show that due to the dynamic adjustment on degree, scheme using algorithm AIDOA outperforms other schemes with static degree in most cases and is able to adapt to change of system workload.

The rest of this paper is organized as follows. Section 2 describes the power consumption model and client access protocol adopted in this paper. Based on the analytical model, we propose algorithm AIDOA in Section 3. Several experimental results are shown in Section 4 to evaluate the performance of algorithm AIDOA. Finally, Section 5 concludes this paper.

2. PRELIMINARIES

2.1 Power Consumption Model

Denote the time for a mobile device to switch the WNI from active mode to doze mode as T_{On} and the time to switch the WNI from doze mode to active mode as T_{Off}. To evaluate the power consumption of turning on and turning off the WNIs, we assume that the power consumption of a mobile device spending in time interval T_{On} (respectively, T_{Off}) is equal to that of a mobile device staying in active mode for time $\alpha_1 \times T_{On}$ (respectively, time $\alpha_2 \times T_{Off}$). Similar to [8], the values of α_1 and α_2 can be obtained by profiling.

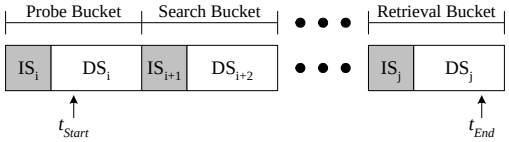


Figure 3: Categories of buckets

Denote the traditional (i.e., without considering the turning-on and turning-off time of WNIs) average tuning time of a data request as T_{Tuning}. For a data request, we also let T_{On} and T_{Off} be the average times for a mobile device spending in turning on and turning off the WNI. Hence, to evaluate the overall power consumption, we define the *effective tuning time* of a data request as $T^{Eff.}_{Tuning} = T_{Tuning} + n_1 \times \alpha_1 \times T_{On} + n_2 \times \alpha_2 \times T_{Off}$, where n_1 and n_2 are the numbers of times of turning on and turning off the WNI, respectively, and T_{Tuning} is the traditional tuning time. To ease the presentation, we use the term tuning time to represent effective tuning time, and assumes $\alpha_1 = \alpha_2 = 1$ in the rest of this paper.

2.2 Client Access Protocol

After submitting a data request, a mobile client will behave according to the employed client access protocol to retrieve the desired data item. In this paper, we adopt the client access protocol described in [7], and the protocol consists of the following phases.

- *Initial probe phase:* After submitting a data request, the mobile device tunes to the broadcast channel and listens on the broadcast channel to wait for the appearance of an index segment.
- *Index search phase:* The mobile device enters index search phase after receiving an index segment. In index search phase, the mobile device determines whether the desired data item will be broadcast in the corresponding data segment. If not, the mobile device will switch to doze mode and then switch back to active mode when the next index segment is broadcast. Otherwise, the mobile device will enter data retrieval phase.
- *Data retrieval phase:*

  If the desired data item will be broadcast in the current data segment, the mobile device will calculate the time that the desired data item will be broadcast from the current index segment and switch to doze mode. Then, when the desired data item is broadcast, the mobile device will switch back to active mode to retrieve the desired data item.

Consider the example shown in Figure 3 that a mobile device submits a data request. t_{Start} is the time that the mobile device starts to listen on the broadcast channel after submitting the data request, and t_{End} is the time that the mobile device receives the desired data item. From the perspective of the mobile device, the buckets within the time interval from t_{Start} to t_{End} can be divided into the following three categories according to the above three phases:

- *Probe bucket:* The bucket which t_{Start} lies on is called the probe bucket. In Figure 3, $Bucket(i)$ is the probe bucket. There is *exactly* one probe bucket for a data request.

- *Search bucket:* The bucket whose index segment is retrieved by the mobile device and whose data segment is skipped by the mobile device is call the search bucket. In Figure 3, $Bucket(i+1)$, $Bucket(i+2)$, $\cdots$, $Bucket(j-1)$ are all search buckets. For a data request, there may be zero, one or multiple search bucket(s).

- *Retrieval bucket:* The bucket which t_{End} lies on is called the probe bucket. That is, retrieval bucket is the bucket where the mobile device retrieves the desired data item. In Figure 3, $Bucket(j)$ is the retrieval bucket. For a data request, there is *exactly* one probe bucket. In addition, the probe bucket and the retrieval bucket of a data request may be the same or different.

3. AIDOA: ADAPTIVE INDEX AND DATA ORGANIZING ALGORITHM

We propose in this section algorithm AIDOA (standing for Adaptive Index and Data Organizing Algorithm) to adjust the degree of buckets according to the system workload. Basically, algorithm AIDOA consists of two phases: statistics collection phase and degree adjustment phase, and switches between statistics collection phase and degree adjustment phase periodically. In statistics collection phase, the system will keep track of information of all data requests and the recorded information will be used to guide the adaptation procedure in the successive execution of adjustment phase.

3.1 Types of Data Requests

To facilitate the design of algorithm AIDOA in the following subsections, a data request is categorized by (1) the relationship between its probe bucket and its retrieval bucket and (2) the relationship between its t_{Start} and its t_{End}. According to the relationship of the probe and the retrieval buckets, a data request can categorized in to the following two types.

Type I: The probe and the retrieval buckets are the same.

Type II: The probe and the retrieval buckets are the different.

Now, consider the relationship of t_{Start} and t_{End} of a data request. According to the employed client access protocol, in a Type I data request, t_{Start} must be located in the index segment. Otherwise, t_{Start} and t_{End} will not be in the same bucket, and such result conflicts with the definition of Type I. On the other hand, a Type II data request can be further divided into the following two subtypes according to the location of t_{Start}.

Type II.I: t_{Start} is in the index segment.

Type II.II: t_{Start} is in the data segment.

3.2 Statistics Collection Phase

In each execution of statistics collection phase, the system will collect statistic information of all data requests which have been served in the current execution of statistics collection phase. A data request is *served* when the desired data item has been broadcasted.

Two data structures, $Stat_I$ and $Stat_{II}$, are defined to stored the collected information of data requests belonging to Type I, Type II (including Type II.I and Type II.II), respectively. The details of $Stat_I$ and $Stat_{II}$ are as follows.

Details of $Stat_I$

- *ReqNo:* The number of Type I data requests served in the current statistics collection phase

- *AggAT:* Aggregated Access Time of all Type I data requests served in the current statistics collection phase

- *AggTT:* Aggregated Tuning Time of all Type I data requests served in the current statistics collection phase

Details of $Stat_{II}$

- *ReqNo:* The number of Type I data requests served in the current statistics collection phase

- *AggATP/AggTTP:* Aggregated Access/Tuning Time of Probe buckets of all Type II data requests served in the current statistics collection phase

- *AggATS/AggTTS:* Aggregated Access/Tuning Time of Search buckets of all Type II data requests served in the current statistics collection phase

- *AggATR/AggTTR:* Aggregated Access/Tuning Time of Retrieval buckets of all Type II data requests served in the current statistics collection phase

Each field, except $ReqNo$ of $Stat_I$ and $Stat_{II}$, has an *average* version by replacing prefix Agg to Avg. For example, the field $AvgAT$ of $Stat_I$ indicates the *average* access time of all Type I data requests served in the current statistics collection phase. We also define the structure *Request* to indicate multiple data requests which are merged together. Elements in the request queue, pending list and the ready queue are all instance of structure *Request*. An instance of structure *Request* is said in the system when it is in the request queue, pending list or the ready queue. The details of structure *Request* are as follows.

Details of structure *Request*

- *ReqNo*: The number of data requests which are merged together and are represented by the *Request* structure.

- *AvgTIS*: Average Time In Search buckets of the data requests that the *Request* structure represents.

After receiving a data request, the system first determines the type of this data request. If the data request is belonging to Type I, the system calculates the contributions of the data request on the aggregated average and tuning time and updates $Stat_I$ accordingly. Since being able to be served by the current bucket, a Type I data request will neither be merged into a structure *Request* nor be inserted into the request and ready queues and pending list.

On the other hand, when the data request is belonging to Type II, the system first checks whether it can be merged into an instance of structure *Request* in the system. If yes,

the system updates the fields (i.e., $ReqNo$ and $AvgTIS$) of the instance of structure *Request* accordingly. Otherwise, the system creates a new instance of structure *Request* and inserts the instance into the request queue. Finally, the system calculates the contribution on aggregated access time and tuning time of the probe bucket of the data request, and updates $Stat_{II}$ accordingly.

While an instance of *Request*, say r, is retrieved from the ready queue, the system first calculates the average number of search buckets that each data request represented by r has by

$$AvgSBNo \leftarrow \frac{Bucket(j).start - r.ATIS}{d \times (S_D + S_I)}.$$

Then the system calculates the time that the desired data item of r can be retrieved (i.e., t_{End}). Finally, with t_{End}, the system calculates the aggregate contributions of the probe buckets of all data requests represented by r on the aggregated access time and tuning time, and updates $Stat_{II}.AggATR$ and $Stat_{II}.AggTTR$ accordingly.

3.3 Degree Adjustment Phase

In each execution of degree adjustment phase, the system will adjust the degree (i.e., the value of d) of buckets according to the statistic information collected in the precedent execution of statistics collection phase. Let $T_{Access}(d)$ and $T_{Tuning}(d)$ be the average access time and average tuning time, respectively, when the degree of the broadcast programs is d. For each field, the value of the *average* version is equal to the value of the *aggregate* version divided by the number of data requests. For example, the value of $Stat_I.AvgTT$ is equal to $\frac{Stat_I.AggTT}{Stat_I.ReqNo}$. Then, we have

$$\begin{aligned} & T_{Access}(d) \\ = \; & W_I \times (Stat_I.AvgAT) + W_{II} \times \\ & (Stat_{II}.AvgATP + Stat_{II}.AvgATS + Stat_{II}.AvgATR), \end{aligned}$$

and

$$\begin{aligned} & T_{Tuning}(d) \\ = \; & W_I \times (Stat_I.AvgTT) + W_{II} \times \\ & (Stat_{II}.AvgTTP + Stat_{II}.AvgTTS + Stat_{II}.AvgTTR), \end{aligned}$$

where W_I and W_{II} are the weights of Type I and Type II data requests, respectively. The values of W_I and W_{II} are defined as the ratios of the numbers of Type I and Type II data requests and are equal to $\frac{Stat_I.ReqNo}{Stat_I.ReqNo+Stat_{II}.ReqNo}$ and $\frac{Stat_{II}.ReqNo}{Stat_I.ReqNo+Stat_{II}.ReqNo}$, respectively. In addition, $T_{OverAll}(d)$ is adopted as the metric of the system performance when the degree of the broadcast programs is d, and is defined as

$$T_{OverAll}(d) = \beta \times T_{Access}(d) + (1 - \beta) \times T_{Tuning}(d),$$

where β is the weight of $T_{Access}(d)$. The objective of degree adjustment phase is to determine the value of d to minimize $T_{OverAll}(d_1)$. However, since globally minimizing $T_{OverAll}(d)$ is difficult, algorithm AIDOA is designed to find the new value of d, say d_1, where $T_{OverAll}(d_1)$ is local minimum where $T_{OverAll}(d_1)$ is smaller than $T_{OverAll}(d_1 + 1)$ and $T_{OverAll}(d_1 - 1)$. Since the exact values of $T_{Access}(d_1)$ and $T_{Tuning}(d_1)$ where $d_1 \neq d$ cannot be obtained from the statistic information, we adopt the following approximation method to estimate $T_{Access}(d_1)$ and $T_{Tuning}(d_1)$.

Let $Stat_I^{d_1}$ and $Stat_{II}^{d_1}$ be the approximations of the values of structure $Stat_I$ and $Stat_{II}$ when the degree of buckets is d_1. Then, we have the following lemmas:

Lemma 1 $Stat_I^{d_1}.AvgAT$ and $Stat_I^{d_1}.AvgTT$ can be approximated by

$$Stat_I^{d_1}.AvgAT = Stat_I.AvgAT + (d_1 - d) \times \frac{S_I}{B},$$

and

$$Stat_I^{d_1}.AvgTT = Stat_I.AvgTT,$$

respectively.

Lemma 2 $Stat_{II}^{d_1}.AvgATP$ and $Stat_{II}^{d_1}.AvgTTP$ can be approximated by

$$\begin{aligned} & Stat_{II}^{d_1}.AvgATP \\ = \; & \frac{S_I}{S_I + S_D} \times Stat_{II.I}^{d_1}.AvgATP + \frac{S_D}{S_I + S_D} \times Stat_{II.II}^{d_1}.AvgATP, \end{aligned}$$

and

$$\begin{aligned} & Stat_{II}^{d_1}.AvgTTP \\ = \; & \frac{S_I}{S_I + S_D} \times Stat_{II.I}^{d_1}.AvgTTP + \frac{S_D}{S_I + S_D} \times Stat_{II.II}^{d_1}.AvgTTP, \end{aligned}$$

respectively, where

$$Stat_{II.I}^{d_1}.AvgATP = Stat_{II}.AvgATP + (d_1 - d) \times \left(\frac{S_I}{B} + \frac{S_D}{B} \right),$$

$$Stat_{II.I}^{d_1}.AvgTTP = Stat_{II}.AvgTTP + (d_1 - d) \times \frac{S_I}{B},$$

$$Stat_{II.II}^{d_1}.AvgATP = Stat_{II}.AvgATP + (d_1 - d) \times \frac{S_D}{B}$$

and

$$Stat_{II.II}^{d_1}.AvgTTP = Stat_{II}.AvgTTP + (d_1 - d) \times \frac{S_D}{B}.$$

As mentioned in Lemma 2, setting the degree of buckets will increase the numbers of index and data items in each probe bucket of Type II data requests by $d_1 - d$. Suppose that these extra index and data items are from the search buckets. Then, we have

Lemma 3 $Stat_{II}^{d_1}.AvgATS$ and $Stat_{II}^{d_1}.AvgTTS$ can be approximated as

$$Stat_{II}^{d_1}.AvgATS = AvgSBNo_1 \times d_1 \times \frac{(S_I + S_D)}{B},$$

and

$$Stat_{II}^{d_1}.AvgTTS = AvgSBNo_1 \times \left(d_1 \times \frac{S_I}{B} + T_{Off} + T_{On} \right),$$

where

$$AvgSBNo_1 \quad = \quad \frac{Stat_{II}.AvgATS \times B}{d_1 \times (S_I + S_D)} - \frac{d_1 - d}{d_1},$$

respectively.

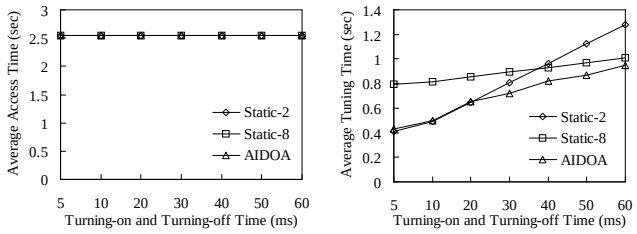
Lemma 4 $Stat_{II}^{d_1}.AvgATR$ and $Stat_{II}^{d_1}.AvgTTR$ can be approximated as

$$Stat_{II}^{d_1}.AvgATR = Stat_{II}.AvgATR + (d_1 - d) \times \frac{S_I}{B},$$

and

$$Stat_{II}^{d_1}.AvgTTR = Stat_{II}^{d_1}.AvgTTR,$$

respectively.



(a) Average Access Time (b) Average Tuning Time

Figure 4: The effect of turning-on and turning-off time

We then devise procedure DegreeAdjustment to find the value of d_1 where $T_{OverAll}(d_1)$ is local minimum. In procedure DegreeAdjustment, the system first checks whether increasing or decreasing the value of degree will reduce the value of $T_{OverAll}(d_1)$. After that, the system repeatedly increases or decreases the value of degree by one until $T_{OverAll}(d_1)$ is local minimum. Finally, the system sets the value of degree to the return valued of procedure DegreeAdjustment.

4. PERFORMANCE EVALUATION

4.1 Simulation Model

In order to evaluate the performance of the proposed degree adjustment method in algorithm AIDOA, the algorithm proposed in [4] (referred to as algorithm Static) is modified to integrated into the proposed system. Hence, the difference between algorithm AIDOA and algorithm Static is on the ability of adjusting the degree of buckets. Based on algorithm Static, we devise two schemes, Static-2 and Static-8 which set the degree of buckets to two and 8, respectively, and the values of degree of buckets are fixed throughout the simulation. In addition, scheme AIDOA employs algorithm AIDOA and initializes the degree of buckets to two. Hence, scheme AIDOA will dynamically adjust the degree of buckets according to system workload. Note that all these three schemes employ server cache and asynchronous I/O to eliminate performance degradation caused by the data item fetch time.

4.2 Effect of Turning-on and Turning-off Time of WNIs

The effect of turning-on and turning-off time of WNIs is measured in this subsection and the experimental results are given in Figure 4. In this experiment, we assume that $T_{On} = T_{Off}$ and set the value of T_{On} and T_{Off} from 5ms to 60ms. As shown in Figure 4a, the values of T_{On} and T_{Off} do not affect average access time in scheme Static-2 and scheme Static-8. It is because that in these two schemes, degrees of broadcast buckets are static, and the values of T_{On} and T_{Off} do not affect the organizations of broadcast programs. On the other hand, although scheme AIDOA is able to dynamically adjust the degree of buckets, the influence of T_{On} and T_{Off} on average access time of scheme AIODA is small since the size of index items is much smaller than that of data items.

Consider average tuning time of these schemes shown in Figure 4b. Since the benefit of increasing the value of degree is in proportion to the values of T_{On} and T_{Off}, scheme Static-2 performs well when T_{On} and T_{Off} are small. On the contrary, scheme Static-8 outperforms scheme Static-2 in the case with large T_{On} and T_{Off}. Although producing more power consumption in the probe bucket than scheme Static-2 does, scheme Static-8 is still able to reduce overall power consumption since being able to greatly reduce the power consumption on turning-on and turning-off the WNIs by reducing the average number of search buckets. On the other hand, with dynamic adjustment in the degree, scheme AIDOA is able to determine a suitable value of degree for current system workload, and hence outperforms scheme Static-2 and scheme Static-8 in most cases.

5. CONCLUSION

In this paper, we analyzed the access time and tuning time of data requests and proposed algorithm AIDOA to adjust the degree of buckets according to system workload. To facilitate scheme AIDOA, we also devised an approximation method to estimate the effect of increasing and decreasing the values of degree. Experimental results showed that algorithm AIDOA is able to greatly reduce power consumption at the cost of slight increment in average access time and dynamically adjust the index and data organization to adapt to change of system workload.

6. REFERENCES

[1] M. Agrawal, A. Manjhi, N. Bansal, and S. Seshan. Improving Web Performance in Broadcast-Unicast Networks. In *Proceedings of the IEEE INFOCOM Conference*, March-April 2003.

[2] D. Aksoy and M. J. Franklin. Scheduling for Large-Scale On-Demand Data Broadcasting. In *Proceedings of IEEE INFOCOM Conference*, pages 651–659, March 1998.

[3] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, 9(9):353–372, June 1997.

[4] S. Lee, D. P. Carney, and S. Zdonik. Index Hint for On-demand Broadcasting. In *Proceedings of the 19th IEEE International Conference on Data Engineering*, March 2003.

[5] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Bettery Operated Devices. In *Proceedings of the 8th ACM/IEEE International Conference on Mobile Computing and Networking*, September 2002.

[6] M. A. Viredaz, L. S. Brakmo, and W. R. Hamburgen. Energy Management on Handheld Devices. *ACM Queue*, 1(7):44–52, October 2003.

[7] J. Xu, W.-C. Lee, and X. Tang. Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air. In *Proceedings of the 2nd ACM/USENIX International Conference on Mobile Systems*, June 2004.

[8] H. Zhu and G. Cao. A Power-Aware and QoS-Aware Service Model on Wireless Networks. In *Proceedings of IEEE INFOCOM Conference*, March 2004.