Facilitating Mobile Decision Making[*]

Mohamed A. Sharaf
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
msharaf@cs.pitt.edu

Panos K. Chrysanthis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
panos@cs.pitt.edu

ABSTRACT

The wide spread of mobile computing devices is transforming the newly emerged e-business world into a mobile e-business one, a world in which hand-held computers are the user's front-ends to access enterprise data. For good mobile decision making, users need to count on up-to-date, business-critical data. Such data are typically in the form of summarized information tailored to suit the clients' analysis interests. In this paper, we are addressing the issue of efficient delivery of summary tables to mobile users' hand-held computers equipped with OLAP (On-Line Analytical Processing) front-end tools. Towards this, we propose a new on-demand scheduling algorithm, called *STOBS*, that exploits the derivation semantics among OLAP summary tables. It maximizes the aggregated data sharing between clients and reduces the broadcast length compared to the already existing techniques. The algorithm effectiveness with respect to access time and energy consumption is evaluated using simulation.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Distributed databases, Query processing*; H.4.2 [**Information Systems Applications**]: Types of Systems —*Decision support*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Design, Experimentation, Performance, Theory

Keywords

Broadcast Scheduling, Broadcast Pull, Power-aware Computing, OLAP, Mobile Computing, E-Commerce

*This work is supported in part by National Science Foundation award ANI-0123705 and by National Center for Disease Control.

1. INTRODUCTION

With the rapid growth in wireless technologies and the cost effectiveness in deploying wireless networks, wireless devices are quickly becoming alternative platforms for accessing enterprise data. This combined with the increased popularity of hand-held computers as well as the availability of light yet powerful laptop computers, mobile computers will become the normal front-end devices hosting sophisticated business applications.

One such sophisticated business application which is central to the success of any enterprise, is the support of decision making. Without an effective decision support system, enterprises will be unable to exploit opportunities as they appear anywhere and anytime. For good decision making, executives and managers need to count on up-to-date, business-critical data, being instantly available on their hand-held and wireless computers. Such data are typically in the form of summarized information tailored to suit the users' analysis interests.

In this paper, we are addressing the issue of efficient delivery of summary tables to mobile clients (e.g., on a company wireless intranet) equipped with OLAP (On-Line Analytical Processing) front-end tools. Decision makers use OLAP tools to execute decision support queries on the enterprise data warehouse or data mart [10]. In wireless networks, broadcasting is the primary mode of operation for the physical layer. Thus, broadcasting is the natural method to propagate information in wireless links and guarantee scalability for bulk data transfer. Specifically, data can be efficiently disseminated by any combination of the following two schemes: *broadcast push* and *broadcast pull*. These exploit the asymmetry in wireless communication and the reduced energy consumption in the receiving mode. Client devices are assumed to be small and portable, and most often rely for their operation on the finite energy provided by batteries. Servers have both much larger bandwidth (downlink) available than client devices and more power to transmit large amounts of data.

In broadcast push the server repeatedly sends information to the clients without explicit client requests. Any number of clients can monitor the broadcast channel and retrieve data as they arrive on the broadcast channel. If data is properly organized to cater to the needs of the clients, such a scheme makes an effective use of the low wireless bandwidth and is ideal to achieve maximal scalability [1, 13, 12].

In broadcast pull, the clients make explicit requests for data. If multiple clients request the same data at approximately the same time, the server may aggregate these re-

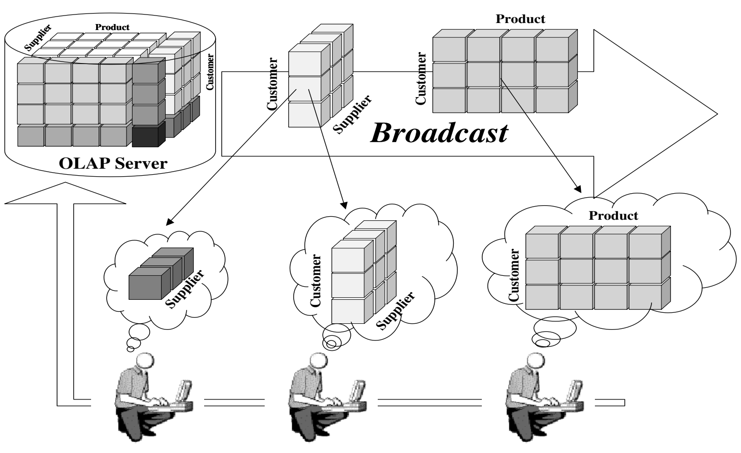

Figure 1: Mobile OLAP System

quests, and only broadcast the data once. Such a scheme also makes an effective use of the low wireless bandwidth and clearly improves user perceived performance. Several scheduling algorithms have been proposed that attempt to achieve maximum aggregation [2, 6, 23, 24].

Assuming the traditional OLAP server basic functionality, the broadcast pull or *on-demand* environment as shown in Figure 1 is the most suitable for supporting wireless OLAP query processing. Every client request is for one of the summary tables. An interesting property in the wireless OLAP system which we call *derivation dependency* is that a table requested by a client may *subsume* the table requested by another client. Since request aggregation is commonly used by general content delivery scheduling algorithms for efficient data dissemination, the derivation dependency property adds a new optimization dimension to the request aggregation process that allows further broadcast efficiency and scalability.

All currently available on-demand scheduling algorithms are *strict* in the sense that they restrict sharing among clients to matching requests. In the paper, we propose a new family of *flexible* scheduling algorithms that aggregate requests by exploiting their semantics to increase sharing among clients that goes beyond the exact match of requests.

Our proposed on-demand scheduling algorithm is called *Summary Tables On-Demand Broadcast Scheduler* (*STOBS-α*) which is based on the *RxW* algorithm [4]. STOBS-α is non-preemptive and considers the varying sizes of the summary tables. The unique characteristic of STOBS-α is its α-optimizer that exploits the derivation dependency among the summary tables to increase sharing among clients. Because each table satisfying a particular request incurs a different processing cost, STOBS-α considers this cost when selecting the set of requests to be aggregated into the specific table which is broadcast at a given point. This cost is captured by the selected value of α. By considering that each different value of α yields a different scheduler, *STOBS-α* can be thought of as a family of scheduling algorithms as well.

The effectiveness of our new heuristic that is based on derivation dependency was evaluated experimentally using simulation by comparing *STOBS-α* to the RxW algorithm. To the best of our knowledge, RxW is currently the best performing non-preemptive scheduler reported in the literature [4]. Our experimental results have shown that STOBS-α outperforms the RxW, reducing the access time by up to 83%.

For mobile clients, savings in power consumption is particularly important since they operate on batteries. Power consumption is also becoming a key issue for all other computer products given the negative effects of heat. Heat adversely affects the reliability of the digital circuits and increases costs for cooling [17]. *STOBS* achieves power reductions up to 30% less than RxW, while reducing the average access time by 70%. The latter saving could be increased to 80%, while decreasing the energy consumption to 23% less than RxW by adjusting the value α of the optimizer.

The rest of this paper is organized as follows. The next section presents an overview of the related work in OLAP technologies and broadcast-based data dissemination techniques. In Section 3, we discuss our assumed wireless OLAP environment and in Section 4, we present *STOBS*, our new on-demand scheduling algorithm. Our simulation testbed and experiments are presented in Sections 5 and 6, respectively.

2. BACKGROUND AND RELATED WORK

2.1 OLAP and Summary Tables

In a decision support environment, sets of facts are analyzed along multiple dimensions. This led to the development of the multidimensional data model that represents a set of facts in a multidimensional space in a way that facilitates the generation of summarized data and reports [15]. In this model, data is typically stored using a star schema. The star schema consists of a single fact table storing the measures of interest (e.g., *sales*, or *revenue*) and a table for each dimension (e.g., *product*, *time*, or *region*).

OLAP queries typically operate on summarized, consolidated data derived from fact tables. The needed consolidated data by an OLAP query can be derived using the *data cube* operator [7]. The data cube operator is basically the union of all possible *Group-By* operators applied on the fact table. A data cube for a schema with N dimensional attributes, will have 2^N possible subcubes. Given that the data cube is an expensive operator, often subcubes are precomputed and stored as summary tables at the server. Ba-

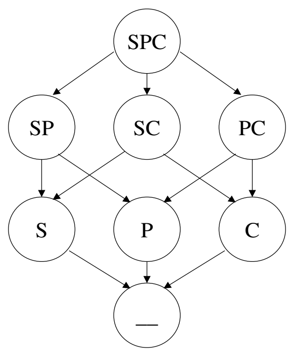

Figure 2: Data Cubes Lattice

sically, a summary table can be modeled as an aggregation query, where the dimensions for analysis are the *Group-By* attributes and the measures of interest are the aggregation attributes. A detailed summary table T_d can be used to derive a more abstract one T_a. In such a case, the abstract table T_a has a derivation dependency on T_d. For example, in Figure 1, by adding the measure values across *supplier*, the detailed table (*supplier*, *product*, *customer*) can be used by a client to extract the abstract table (*product*, *customer*).

The idea of using summary tables to derive one from another has been widely used in materialized views selection. The objective is to select the appropriate set of tables for storing (materialization), so that to speed up future query processing, while meeting the space constraints [10, 8, 9]. To facilitate the selection process, the search lattice was introduced in [10]. The search lattice is a directed graph to represent the subcubes space that captures the derivation dependencies among subcubes. For example, Figure 2 shows the lattice for the (*Supplier(S)*, *Product(P)*, *Customer(C)*) schema.

In this paper, we also use the property of derivation dependency of the summarized tables and the idea of search lattice in selecting the appropriate tables to broadcast over wireless links, such that the user perceived latency is minimized.

2.2 Broadcast-Pull

Several scheduling policies have been proposed in the broadcast pull literature. These policies can be classified as either *non-preemptive* or *preemptive*. In a non-preemptive environment, it has been pointed out that that *First Come First Serve* (FCFS) scheduling would provide poor access time in broadcast pull [6] and *Most Requests First* (MRF) and *Longest Wait First* (LWF) were proposed as efficient alternatives [6, 24]. The *RxW* algorithm [4] combines the benefits of MRF and FCFS, where the intuition underlying RxW is that "hot" or popular data items are disseminated as soon as possible yet it avoids starvation of "cold" or less popular data items by means of an aging scheme.

Preemptive scheduling policies have been introduced to handle the heterogeneous requests problem, i.e., requests for data items of varying sizes [2]. Three preemptive algorithms have been proposed, namely, *Longest Total Stretch First* (LTSF), an off-line algorithm called *BASE* and its online approximation *MAX*.

Preemptive scheduling policies exhibit better performance than the non-preemptive ones for heterogeneous requests. However, preemptive schemes cannot in general support selective tuning. Selective tuning is the fundamental property for preserving energy where the main idea is: if sufficient indexing information is provided to clients, then the mobile device access pattern to the data stream can alternate between a *doze* mode waiting for data and an *active* mode tuning for required data. In a doze mode the mobile device is consuming power orders of magnitude less than that in the active mode. Power conservative indexing methods for single-attribute and multi-attribute based queries in broadcast push environments appeared in [13, 12, 3].

The idea of merging queries with overlapping answers to reduce broadcast data dissemination cost has been introduced in the context of a multicast subscription environment [5]. In this approach, a post-filtering is needed at the client side to obtain the answer to the original query. A similar proposal appeared in [16], where a semantic description is attached to broadcast unit, called a chunk, which is a cluster of data items. This allows clients to determine if a query can be answered based solely on the broadcast and to request the remaining items in the form of a supplementary query. A popularity-based scheduling policy was used to broadcast the data chunks. This assumes that the server has been informed about the clients' queries at the beginning of each broadcast cycle.

Our work carries some similarity with the work in [5, 16]. However, we are modeling an on-demand broadcast environment, where the server has no prior knowledge of the arriving requests. Additionally, in our case, the requests are for summary tables (aggregation queries) rather than queries with selection predicates as in [5, 16], where the relative sizes of dependent tables may vary tremendously. Hence, an efficient online scheduler is needed with an objective of reducing the access time and energy consumption. The design of this scheduler is our main contribution in this paper.

3. WIRELESS OLAP MODEL

In this section, we are presenting our model for the wireless OLAP environment. Our assumed architecture is based on broadcast pull scheme as shown in Figure 1. The OLAP server is responsible for maintaining and disseminating the summary tables. We are assuming that all the lattice subcubes are pre-computed and stored at the server, which is a reasonable assumption, specially for relatively small size data marts. The Essbase system (according to [10]) is an example of commercial product that materialize all the possible summary tables.

A client sends an uplink request for a table on the *uplink channel*. It can then be in one of two states: *tune* state or *wait* state. When the client needs to listen to the downlink channel, it enters the tune state and switches to active mode. Otherwise, it is in wait state and operates in doze mode. Clients depend on the server to satisfy all their requests; they are not accessing any local storage and previous answers are not locally cached for future use.

An uplink request Q is characterized by the set of its Group-By attributes D. Hence, we represent a request as Q^D and the corresponding table as T^D. A summary table T^{D1} *subsumes* table T^{D2}, and consequently T^{D2} is *dependent* on T^{D1}, if $D2 \subseteq D1$. We denote the number of dimensional attributes (table dimensionality) in the set D as $|D|$.

The smallest logical unit of a broadcast is called a *packet* or *bucket*. A broadcast table is segmented into equal sized packets, where the first one is a *descriptor* packet. Every packet has a header, specifying weather it is data or descriptor packet, the offset (time step) to the beginning of the next descriptor packet, and the offset of the packet from the beginning of its descriptor packets. The descriptor packet contains a table descriptor which has an *identifier* that captures the table aggregation dimensions, the number of attribute values or tuples in the table and the number of data packets accommodating that table. We are assuming that no single data packet is occupied by tuples from different tables. Each summary table is broadcast within a broadcast cycle that starts with the table descriptor packet.

By assuming each client knows the order in which attributes are defined in the database schema, we use bit encoding to represent the semantics of a client request and the table descriptor identifier. The representation is a string of bits; its length is equal to the number of the complete schema dimensions and each bit position is equivalent to one of the dimensions d_1, d_2, ..., d_n.

If a table T^D has dimension $d_x \in D$, then the bit at position x is set to 1, otherwise it is a zero. For example, assume the (*supplier*, *product*, *customer*) schema. The representation of the (*supplier*, *customer*) summary table will be 101. This scheme can be easily extended to include tables with more than one measure and different aggregation functions. But, without loss of generality, we are assuming only one measure attribute and *sum()* as the aggregation function in this paper.

When a client submits a request for table T^R on the uplink channel, it immediately tunes to the downlink channel, and goes through a three-phase access protocol until its request is satisfied:

1. *Initial probe*,
2. *Semantic matching* , and
3. *Table retrieval.*

In the initial probe phase, the client tunes to the downlink channel and uses the nearest packet header to locate the next descriptor packet.

The semantic matching phase starts when the client finds a descriptor packet, say for table T^B, then the client can semantically classify T^B as:

- *Exact match*: if the aggregation dimensions in T^B are the same as T^R (i.e., $R = B$).
- *Subsuming match*: if T^B subsumes T^R, and T^B is not an exact match for T^R (i.e., $R \subset B$ and $R \neq B$).
- *No match*: if it is neither an exact match nor a subsuming match (i.e., $R \not\subseteq B$).

For example, assume R is (*supplier*, *product*), then B_1 = (*supplier*, *product*, *customer*) is a subsumption match, while B_2 = (*product*) and B_3 = (*supplier*, *customer*) are examples of no match.

Depending on the matching result and the scheduling algorithm used (as we will see in Sections 4), the client will either switch to the final retrieval phase or it will stay in the matching one. In the former, the client stays in active mode tuning to the next sequence of data packets to read (download) table T^B. While in the latter case, it will wait switching to doze mode in order to reduce power consumption. Using the offset in the packet header, it wakes up just before the next broadcast cycle (i.e., descriptor packet of the next table on broadcast) where the semantic matching process is repeated. The access protocol is shown in Figure 3.

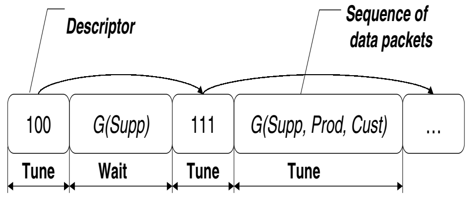

Figure 3: Client Access to Broadcast

3.1 Cost Model

The time interval a mobile client spends since issuing a request until the summary table is made ready for either displaying or further processing, can be expressed by the following three components:

- **Wait Time:** The total period of time a client spends waiting for a descriptor packet to appear on the downlink channel until it finds a matching one. A client network interface is switched to doze mode during the wait time.
- **Tune Time:** It is the total period of time spent by the client listening to the downlink channel either reading a descriptor packet or a stream of data packets containing the requested summary table. During tuning, the client network card is in active mode consuming energy orders of magnitude higher than that in doze mode.
- **Processing Time:** It is the total period of time needed to convert the downloaded data into the form of the requested summary table. During this phase, the processor is active, accessing the table in main memory and consuming full power.

Hence, we define the *Access Time* (T_{Total}) as the total period of the wait, tune, and processing times.

$$T_{Total} = T_{Wait} + T_{Tune} + T_{Processing}$$

Accordingly, the *Total Energy Consumption* (E_{Total}) is formulated as:

$$E_{Total} = E_{Doze} + E_{Active} + E_{Processing}$$

where the energy consumed during a certain period equals to the product of the power consumption during that period and the duration of the period.

4. SUMMARY TABLES ON-DEMAND BROADCAST SCHEDULER (*STOBS*-α)

The profile for OLAP summary tables access has the following key features:

1. *Heterogeneity:* summary tables are of different dimensionalities and varying sizes

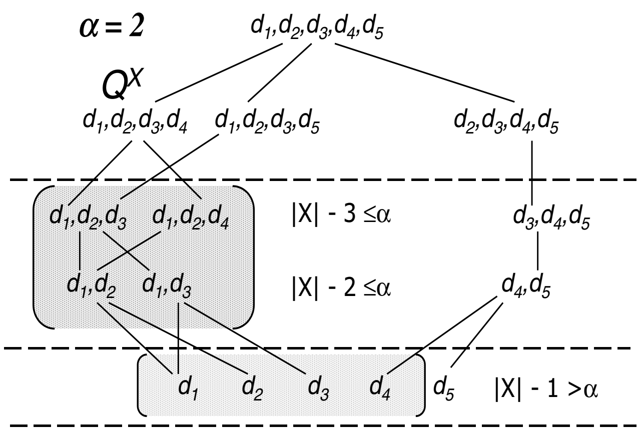

Figure 4: Flexibility & Search Lattice

2. *Skewed Access:* Request from OLAP clients usually form a hot spot within the data cubes lattice. Most of the time queries are accessing low dimensionality tables and they often drill down for detailed ones.

3. *Derivation dependency:* it is often possible to use one detailed table to extract other tables.

The *Summary Tables On-Demand Broadcast Scheduler* (*STOBS*-α) that we are proposing in this section, consists of two components: A *normalizing* (basic selection) component, which captures the first and second features above and the α*-optimizing* component that exploits the third feature above to control the degree of sharing. The basic selection component is an extension of RxW that takes into consideration the table size. It normalizes the value of RxW by dividing it by the size of the table ($\frac{R \times W}{S}$).

Specifically, the server queues up the clients requests as they arrive. For each request Q^X for a summary table T^X, the server maintains the following three values:

- R: The number of requests for T^X. This value is incremented with every arrival of a request for T^X.
- W: The time the first request Q^X has been waiting for table T^X.
- S: The size of table T^X.

When it is time for the server to make a decision which table to broadcast next, it computes the $\frac{R \times W}{S}$ value for each request in the queue. The request with the highest value is selected to be broadcast.

The parameter α defines the degree of flexibility in broadcasting a summary table and eliminating from the broadcast some of its dependent tables. For example, for $\alpha = 2$, if the server selects a table T^X to broadcast, then the server discards every request in the queue for a table T^Y that can be derived from T^X and is up to two levels lower in the data cubes lattice.

More precisely, T^Y can be discarded and it is not broadcast if $Y \subset X$ and $\mid X \mid - \mid Y \mid \leq 2$. Consequently, a client will use a table T^X that subsumes the table it originally requested T^Y and is up to two levels higher in the data cubes lattice. That is, a client requested T^Y will use T^X if $Y \subset X$ and $\mid X \mid - \mid Y \mid \leq 2$.

In general, the formal criterion at the server to discard a request for table T^Y and aggregate it into a request for table T^X is:

$$T^Y \textit{ is discarded iff } T^X \textit{ is broadcast and } Y \subset X \textit{ and } \mid X \mid - \mid Y \mid \leq \alpha$$

The same criterion is used at the client side to determine if it has to use table T^X that subsumes its original request for table T^Y that has been discarded at the server and will not be broadcast.

The value of α ranges from 0 to the maximum data cube dimensionality MAX. At $\alpha = 0$ there is no flexibility in using summary tables and the client access is restricted to exact match. At $\alpha = MAX$, a client will use the first subsuming matching table. STOBS-0 is basically $\frac{R \times W}{S}$ and is a strict algorithm, while the family of STOBS-α, where $\alpha > 0$, are *flexible algorithms.* The value of α is made known to the clients by including it as part of the table descriptor information along with the dimensionality of the broadcasted table.

As an example, consider the search lattice shown in Figure 4, in which nodes are summary tables. Assume the search lattice nodes shown in figure, are the tables for which there exist at least one request. Also, let assume that Q^X is a request to the 4-dimensional table T^X:(d_1, d_2, d_3, d_4) that is selected to be broadcast next. All the shaded tables in the figure can be derived from T^X. These are (d_1), (d_2), (d_3), (d_4), (d_1, d_2), (d_1, d_3), (d_1, d_2, d_3), and (d_1, d_2, d_4). However, if we assume that $\alpha = 2$ then clients' requests for tables (d_1, d_2), (d_1, d_3), (d_1, d_2, d_3), and (d_1, d_2, d_4) will be satisfied by T^X and hence the requests for these tables will be discarded; whereas the requests for tables (d_1), (d_2), (d_3), and (d_4) will remain in the queue for future consideration.

4.1 Discussion

It should be noted that our proposed $\frac{R \times W}{S}$ is different from the algorithm appeared in [21] under the same name. Contrast to our work, in [21] requests are assumed to be homogeneous and the S value refers to the service time required to extract a request at the server.

The intuition for *STOBS*-α is to capture all the specific features of summary tables access in an on-demand broadcast environment. The $\frac{R \times W}{S}$ encapsulates all the factors affecting a response access time. The α parameter controls

	Q_1	Q_2	Q_3	Q_4
R_i	2	1	1	2
A_i	5	4	10	14
S_i	20	25	50	60

Table 1: Example Settings

Algorithm	BSeq	AAT	BSize
RxW	T_4, T_1, T_3, T_2	102.8	155
STOBS-0	T_1, T_4, T_2, T_3	85.3	155
STOBS-2	T_1, T_4, T_3	77.0	130

Table 2: Example Results

the degree of flexibility. The advantage of the flexibility is to find another aspect of common interest other than the exact strict one. The drawback is the extra time and energy needed for tuning in and processing a detailed table rather than a summarized one. Picking a reasonable value for α will balance the trade-off between reducing the wait time and increasing the tune and processing times. As in [10], we are assuming a linear cost model for aggregate query processing, where a table scan is required to compute the result. During processing, the processor accesses the downloaded summary table from memory and the memory transfer rate determines the time taken for processing.

As an example for the flexibility trade-off, consider the case where α is set to 2. In case of request for table T^{high}, where $| X | - | high | \leq 2$. If the $\frac{RxW}{S}$ value for the request for table T^{high} is still not high enough, then disseminating T^X will reduce the wait time by a client requested T^{high}. On the contrary, a client requested table T^{low}, where $| X | - | low | > 2$, if T^X is disseminated, the client requested T^{low} would rather wait for the next broadcast cycles to avoid the costly tune time needed for downloading T^X and internally processing it.

Let us now consider a simple numeric example that highlights the differences in scheduling decisions and average access time between the strict scheme and our proposed flexible approach. Table 1 shows the example settings, where there are four pending requests Q_1, Q_2, Q_3, and Q_4 for four different tables T_1, T_2, T_3, and T_4. The R_i, A_i, and S_i values for request Q_i are as described above. Additionally, we are assuming that table T_2 is derivable from T_4 and T_2 is within two levels lower than T_4 in the subcubes lattice. Each scheduler has to make a decision what is the sequence of tables to broadcast given the queue status at each broadcast cycle. In this snapshot, the four requests constitute the whole workload, i.e., no more requests will arrive at the server.

Table 2 shows the broadcast sequence (BSeq) generated by each algorithm (left most table is the first to be broadcast), the corresponding average access time (AAT), and the broadcast size (BSize). Assume that the transmission time of a table is equal to its size, hence, the transmission time for table T_1 is 20 units and its access time using RxW is equal to $A_1 + S_4 + S_1$, where $(A_1 + S_4)$ is the wait time and S_1 is the tune time.

We can see in the sequence generated using RxW that the large table T_4 is the first to be broadcasted due to its high A and R values. In the case of the strict STOBS-0, sending the small popular table T_1 first, followed by T_4, gave an average access time 17% less than the strict RxW. As it is possible to derive T_2 from T_4, STOBS-2 selects T_4 for transmission and discards T_2, converting the wait time for T_2 into a tune time to T_4 and in addition eliminating part of the wait time for T_3. This broadcast sequence gave an average access time that is 10% less than that achieved by STOBS-0.

5. PERFORMANCE EVALUATION TESTBED

We implemented a system simulation model to evaluate the potential gains using the *STOBS*-α algorithm by comparing it to the RxW scheduling algorithm. We modeled the environment as a single server with a set of clients. There is a single downlink broadcast channel over which all data is disseminated to the clients and a single uplink channel that clients use to send uplink requests. We are assuming that clients are able to complete any uplink request in a single uplink packet.

We generated a synthesized lattice for a six-dimensional data cube. The sizes of lattice subcubes is computed as in [14], where a subcube is given a binary code C. The binary code is similar to the bit encoding we use for identifying cubes on the broadcast. The subcube size (number of tuples) is set to C^2. The final cube size is the product of generated number of tuples and the number of attributes (dimensional and measure attributes), hence, the unit for size is the number of attribute values in a table. Using a six-dimensional data cube results in a maximum value for α equals to 6. Due to similarity in performance between close values of α and for the sake of readability, we are only presenting results where α is set to 0, 1, 2, 3, and 6.

The way we generated the lattice ensures diversity in subcubes sizes and significant size difference between a cube and all its dependent cubes. In the generated lattice, cubes at the bottom left area have small sizes while those at top right have larger sizes. This setting will results in 64 (2^6) possible queries.

Derived summary tables are of different sizes, i.e., they have different degrees and cardinalities. In the simulation, we are assuming that attributes values have the same sizes and a data packet capacity is 10 attribute values and one attribute value is 10 bytes.

To test the system under a typical workload, requests are generated by the clients according to Zipf distribution with the Zipf parameter (θ) default value is equal to 0.5. Queries are sorted according to their sizes, so that queries to small size tables occur with higher probability than queries to detailed ones. These settings will create a hot spot at the bottom of the data cubes lattice.

We control the simulation by establishing a fixed number of requests, that is, each client was required to complete a certain number of requests before the experiment would terminate. A client will pose a new request as soon as it gets an answer to its previous one.

Table 3 summarizes our simulation parameters and settings reported in this paper. The combination of these parameters allows us to examine the scalability of the system as well as the impact of a changing workload on the algorithm performance.

6. RESULTS

For our evaluation, we took extensive performance measurements. The time reported throughout is in *Seconds* and the energy consumption is in *Joule*. We considered a wireless LAN where the broadcast channel has a bandwidth of

Parameter	Value
Base Cube Dimensionality	6 dimensions
Possible Requests	64 requests
Packet Capacity	10 attributes values
Zipf Parameter (θ)	0.0 – 0.9
Simulation Length	100 requests/client
Number of Clients (Request Rate)	10 – 200 clients
α-optimization	0,1,2,3,6

Table 3: Simulation Parameters

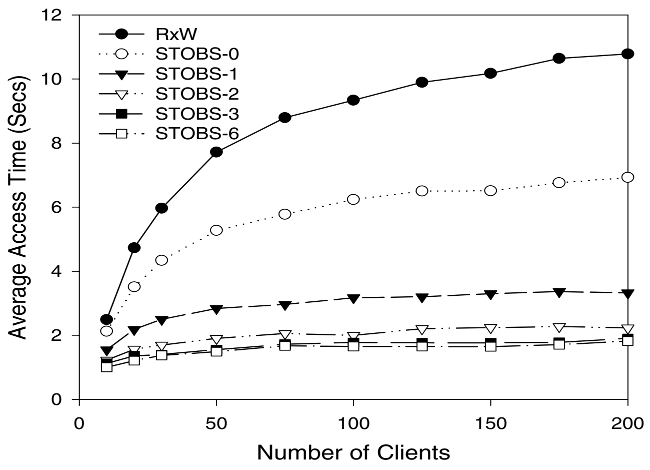

Figure 5: Access Time

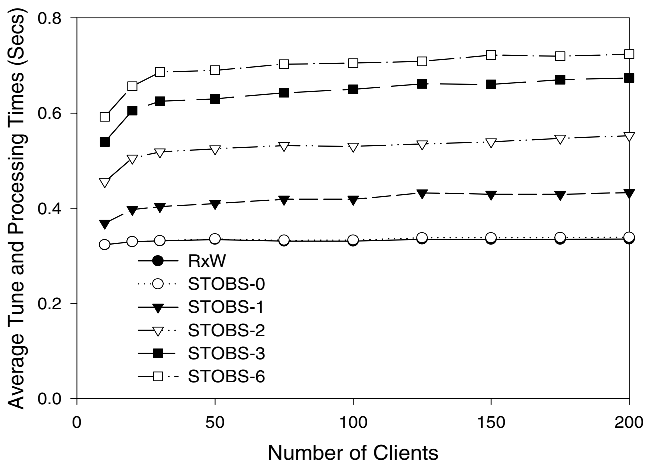

Figure 6: Tune and Processing Times

1Mbps. We assumed clients are using the IBM ThinkPad laptop [11] that is equipped with Pentium 4 mobile processor which consumes 2 Watts on average, with a 100 MHz RAM and 64 bits bus. The wireless card operates on 5 Volts using 9mA at doze mode and 185 mA at receiver mode. The processing of a summary table is basically a scanning process, and hence we used the memory transfer time to bound the processing time.

6.1 Access Time

Figure 5 shows the average access time for RxW and the STOBS family of algorithms. All algorithms exhibit a similar behavior, that is, the average access time increases but ultimately levels as the number of clients is increased. This behavior is normal for broadcast data delivery to clients with shared interests. The figure shows how the access time is decreasing with increasing α for the same number of clients. Furthermore, this reduction in access time is more significant as the load increases and more flexibility is needed to handle the high request rate. For instance, consider the cases of 10 and 200 clients where $\alpha = 2$ (STOBS-2). In the case of 10 clients, the average access time decreased by 43% compared to $\alpha = 0$ (STOBS-0), while in the case of 200 clients STOBS-2 achieved 68% reduction in the access time compared to the strict STOBS-0 and a reduction of 80% compared to RxW. In the case of STOBS-6, and population of 200 clients, the average access time is 83% less than RxW.

Figures 6 and 7 depicts separately the tune, processing and wait times demonstrated in Figure 5. The two strict algorithms (RxW and STOBS-0) exhibit the same minimal tune and processing times. This is reflected in Figure 6 where their corresponding curves coincide.

Increasing the value of α leads to the increase in tune and processing times. However, this increase was successfully compensated by a larger decrease in wait time as shown in Figure 7. It is worth mentioning here, that as the requests arrival rate increases, the wait time becomes the dominant factor in the access time computation. This observation supports our idea of tackling the access time reduction by decreasing the wait time even if it yields to a moderate increase in the tune and processing times.

6.2 Energy Consumption

Figure 8 shows the average total energy consumption for the different algorithms. Recall that energy is computed by considering the power consumption during tuning, wait, and processing time.

In the case of the two strict algorithms RxW and STOBS-0, the tuning and processing times are the same and hence their corresponding energy consumptions. Consequently, the difference in their energy consumption is due to their difference in wait time. For example, this difference in wait time in the case of 200 clients translates into a 22% less total energy consumption by the STOBS-0.

As expected, the extreme case of flexibility (α=6) leads to an increase in the overall energy consumptions. However, reduction in energy consumption is achieved by setting α to the values of 1 and 2. This reduction is more noticeable at

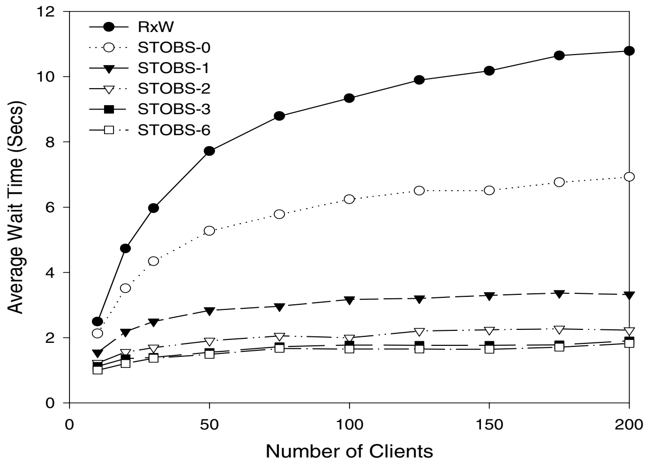

Figure 7: Wait Time

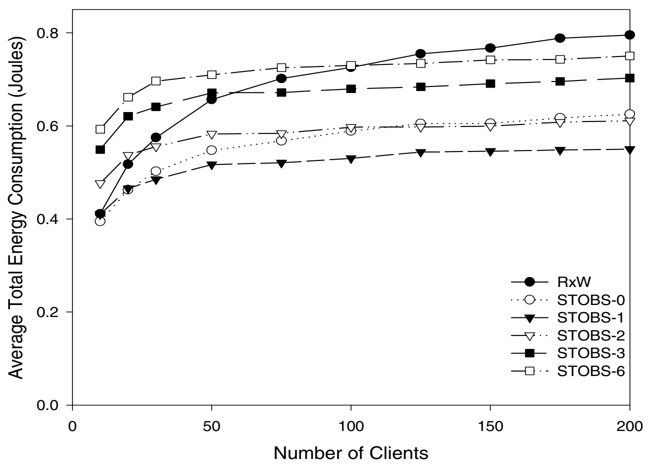

Figure 8: Energy Consumption

higher loads where doze energy (that is wait time is longer) is playing an important role. For instance, consider again the cases of 10 and 200 clients and $\alpha = 2$. In the case of 10 clients, energy consumption in STOBS-2 increased by 20% compared to STOBS-0, while it decreased by 3% in the case of 200 clients. The minimum energy consumption is provided by STOBS-1, which reached 12% less than STOBS-0 in the case of 200 clients.

6.3 Impact of Skewness

In all the previous comparisons, we used the default θ value of 0.5. In this section, we examine the performance for different values of θ, i.e., the degree of skewness of access. Figure 9 shows the average access time for a setting, where the number of clients equals to 100, each posing 100 requests. The Zipf parameter ranges from 0 to 0.9 where for $\theta = 0$, the distribution corresponds to the uniform one.

Since the number of clients (request rate) is kept constant, the increased overlap in client interests allows more efficient use of the broadcast bandwidth. Therefore, as the skew increases all algorithms provide improved reduction in access time. However, the STOBS-α schedulers, where $\alpha > 0$, are also taking advantage of the derivation dependency property between requested tables. Using STOBS-2 reduces the

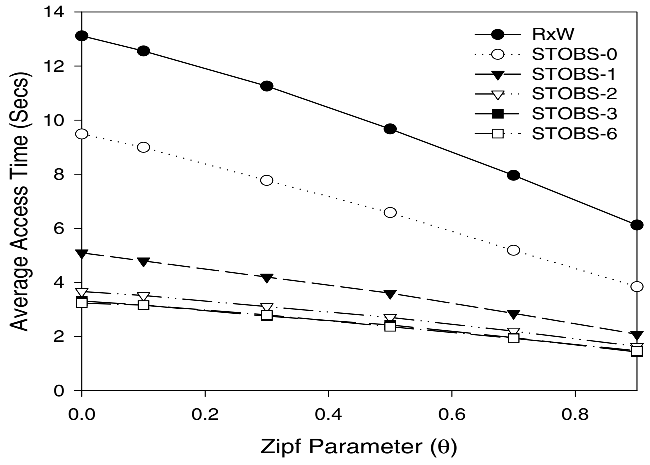

Figure 9: Access Time Vs. θ

access time by 60% less than the STOBS-0 for all θ values.

In evaluating our proposed heuristic algorithm, we have conducted a large number of experiments, many of which could not be included here. For experiments involving the effect of varying the cube dimensionality, as well as other parameters, see [18]. We also investigated the notion of fairness in scheduling . Our experiments illustrating fairness as performance metric appeared in [19].

7. CONCLUSIONS

In this paper, we re-emphasized the role of broadcast based data dissemination in supporting efficient access of enterprise data warehouse and consequently enabling good decision making anytime and anywhere. Although the emphasis of our paper was on wireless and mobile computing environments, our result are applicable in wired networks which support multicasting.

More specifically, the major contribution of this paper is a new optimization and a family of algorithms called STOBS-α that use this optimization to achieve better aggregation of OLAP requests that goes beyond the exact match. The superiority of the STOBS-α was demonstrated experimentally using simulation.

A close examination of our proposed $\frac{R \times W}{S}$ extension to the RxW has shown that it can be an approximation of the non-preemptive LTSF algorithm [2] and as such, in subsequent work we experimented with LTSF as well as with different definitions of the α-optimizer [20].

In our future work we are planning to compare the performance of $\frac{R \times W}{S}$ and LTSF when each is used as the basic selection component for our summary tables scheduler. An efficient implementation of the former as in [4] can exhibit a low computational overhead while retaining performance close to the latter.

Also we are working on techniques that strongly integrates the flexibility with the selection decision. We are also studying the problem in a push subscribe environment. We are planning to investigate the effect of deploying caching at the client side and what will be the appropriate caching mechanisms.

8. ACKNOWLEDGMENTS

The authors would like to thank Karl Gossett for his help with the initial phase of this work. They would also like to thank the anonymous reviewers for their very helpful comments.

9. REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proc. of the ACM SIGMOD Conf.*, pages 199–210, May 1995.

[2] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proc. of Fourth Annual ACM/IEEE Conf. MobiCom*, pages 43–54, October 1998.

[3] R. Agrawal and P. K. Chrysanthis. Efficient data dissemination to mobile clients in e-commerce applications. In *Proc. of the 3rd Int'l Workshop on E-Commerce and Web Information Systems*, pages 58–65, June 2001.

[4] D. Aksoy and M. Franklin. RxW: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions On Networking*, 7(6):846–860, December 1999.

[5] A. Crespo, O. Buyukkokten, and H. G. Molina. Efficient query subscription processing in a multicast environment (extended abstract). In *Proc. of the 16th ICDE Conf.*, February 2000.

[6] H. D. Dykeman, M. Ammar, and J. W. Wong. Scheduling algorithms for videotex systems under broadcast delivery. In *Proc. of the 1986 Int'l Conf. on Communications*, pages 1847–1851, June 1986.

[7] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. In *Proc. of the ICDE Conf.*, pages 152–159, February 1996.

[8] H. Gupta. Selection of views to materialize in a data warehouse. In *Proc. of ICDT*, pages 98–112, January 1997.

[9] H. Gupta, V. Harinarayan, A. Rajaraman, and J. Ullman. Index selection for olap. In *Proc. of the ICDE Conf.*, pages 208–219, April 1997.

[10] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. of the ACM SIGMOD Conf.*, pages 205–216, June 1996.

[11] http://www.ibm.com.

[12] Q. Hu, W.-C. Lee, and D. L. Lee. Power conservative multi-attribute queries on data broadcast. In *Proc. of the 16th ICDE Conf.*, pages 157–166, 2000.

[13] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy efficient indexing on air. In *Proc. of the ACM SIGMOD Conf.*, pages 25–36, May 1994.

[14] P. Kalnis, N. Mamoulis, and D. Papadias. View selection using randomized search. *DKE*, 42(1):89–111, 2002.

[15] R. Kimball. *The Data Warehouse Toolkit.* John Wiley, 1996.

[16] K. C. K. Lee, H. V. Leong, and A. Si. A semantic broadcast scheme for a mobile environment based on dynamic chunking. In *Proc. of the 20th IEEE Int'l Conf. on Distributed Computing Systems*, pages 522–529, 2000.

[17] T. Mudge. Power: A first class design constraint. *Computer*, 34(4):52–57, 2001.

[18] M. A. Sharaf. *STOBS: Summary Tables On-Demand Broadcast Scheduler.* Computer Science Dept., Univ. of Pittsburgh, April 2002.

[19] M. A. Sharaf and P. K. Chrysanthis. On-demand broadcast: New challenges and scheduling algorithms. In *Proc. of the 1st Hellenic Data Management Symposium*, July 2002.

[20] M. A. Sharaf and P. K. Chrysanthis. Semantic-based delivery of olap summary tables in wireless environments. *(To appear in) Proc. of the 11th International Conference on Information and Knowledge Management*, November 2002.

[21] P. Triantafillou, R. Harpantidou, and M. Paterakis. High performance data broadcasting: A comprehensive systems' perspective. In *Proc. of 2nd International Conference on Mobile Data Management*, January 2002.

[22] A. Vahdat, A. R. Lebeck, and C. S. Ellis. Every joule is precious: The case for revisiting operating system design for energy efficiency. In *Proc. of the 9th ACM SIGOPS European Workshop*, September 2000.

[23] N. H. Vaidya and S. Hameed. Scheduling data broadcast in asymmetric communication environments. *ACM/Baltzer Wireless Networks*, 5(3):171–182, 1999.

[24] J. W. Wong. Broadcast delivery. *Proc. of the IEEE*, 76:1566–1577, 1988.