Mobile Networks and Applications 9, 703–714, 2004

On-Demand Data Broadcasting for Mobile Decision Making *

MOHAMED A. SHARAF and PANOS K. CHRYSANTHIS
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA

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 user's analysis interests. In this paper, we are addressing the issue of time and energy efficient delivery of summary tables to mobile users with 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 mobile users and reduces the broadcast length for satisfying a set of requests compared to the already existing techniques. The algorithm effectiveness with respect to access time and energy consumption is evaluated using simulation.

Keywords: broadcast scheduling, broadcast pull, mobile computing, mobile commerce, OLAP, power-aware computing, wireless communication

1. Introduction

With the rapid growth in mobile technologies and the cost effectiveness in deploying wireless networks, mobile devices are quickly becoming the standard platform for accessing time-sensitive data. This, in combination with the increased popularity of hand-held computers as well as the availability of light yet powerful laptop computers, shows that mobile computers will become the preferred front-end devices for hosting sophisticated business applications.

Time-sensitive data is crucial for facilitating informed decision making. Without an effective decision support system, decision makers will not be able to exploit opportunities as they appear anywhere and anytime. Decision makers need to count on up-to-date, business-critical data being instantly available to their hand-held and wireless computers. Such data are maintained in data warehouses or data marts [14] and may be privately or publicly accessible. For example, a public data mart deployed within the stock market galleries or the campus of a trade fair may be responsible for gathering stock market and other financial information. This data is typically in the form of summarized reports tailored to suit the users' analysis interests.

In this paper, we are addressing the issue of time and energy efficient delivery of summary tables to mobile clients (i.e., mobile devices) equipped with simple or sophisticated OLAP (On-Line Analytical Processing) front-end tools. A simple OLAP tool will allow a client to submit a queries to a *base OLAP server* and display the downloaded report. A sophisticated one is a *personal OLAP server* such as PowerPlay [6], Express OLAP Access Pack [5], and BI/Analyze [7], that stores data on the mobile computer and provides OLAP functionality locally. Thus, these personal servers allow mobile users to carry out off-line data analysis and refresh their data when they are connected to base OLAP servers. The data dissemination technique proposed in this paper will enable both simple and sophisticated OLAP tools to efficiently support the on-line analysis using up-to-date information within a wireless and mobile network.

In wireless and mobile networks, broadcasting is the primary mode of operation for the physical layer. Thus, broadcasting is the natural method to propagate information within wireless links and guarantees 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 their operation relies 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 it arrives 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 [1,15,16].

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 requests, 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 [2,10,28,29].

* This work is supported in part by NSF award ANI-0123705, the National Center for Disease Control and the Pennsylvania Department of Health Award ME-01-737. The first author is supported in part by the Andrew Mellon Predoctoral Fellowship.

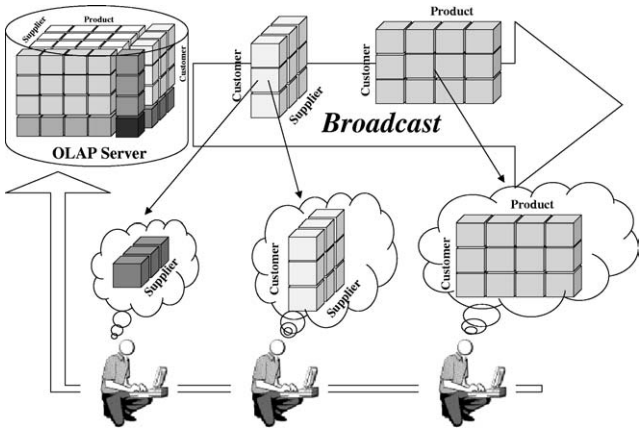

Figure 1. Mobile OLAP system.

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. Requests for the same table are aggregated and the corresponding table is broadcast. This scheme scales well with the increase in number of clients, especially in the cases where the access pattern is skewed towards few tables which are highly popular among users. As an example consider the case of brokers accessing the stock market gallery data mart. At the open and close times many traders will be analyzing the performance of different stocks in different financial dimensions. Some of these stocks are more popular than others, similarly, some analytical dimensions are more important than others. In this situation, a data mart equipped with a broadcast gateway will give higher priority to answering requests to popular tables. This will provide better response time and higher scalability. Similar situations arise in other mobile and wireless applications with similar data requirements. An example is decision making in the context of sensor network databases, where data is gathered from a large number of sensors and stored in a multidimensional structure. In the case of scientific exploration, scientists on the field rely on this data for data mining and pattern detection. These scientists will be using their wireless devices to analyze sensor data disseminated by a satellite or a close wireless station. A scalable data dissemination method is needed to meet the timely requirements of such analysis.

An interesting property in the wireless OLAP system, which we call *derivation dependency*, is that a table requested by a client may *subsume* a 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 this 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 matching of requests.

Our proposed on-demand scheduling algorithm, called *Summary Tables On-Demand Broadcast Scheduler* (*STOBS-*α), is non-preemptive and is based on the priority function $R \cdot W/S$. The formula $R \cdot W/S$ is a generic scheduling function which considers popularity (R), cost (S), and age (W) in prioritizing requests for broadcast. Variants of this function with different definitions of S appeared in [4,19,23,24,27]. The cost in *STOBS-*α captures the varying sizes of 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 at the client, *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 *RxW/S*. We use *RxW/S* to denote the scheduler that is using the basic $R \cdot W/S$ scheduling function. Our experimental results have shown that *STOBS-*α outperforms *RxW/S* scheduler, reducing the access time by up to 62%. Our experiments also show that *STOBS-*α can result in average access time that is within 36% of that provided by a hypothetical ideal on-demand data dissemination scenario in which each client is connected to the server through a dedicated channel.

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 [21]. *STOBS-*α achieves power reductions up to 12% less than *RxW/S*, while reducing the average access time by 48%. There is a trade-off between access time and energy consumption which the value α of the optimizer also controls. For example, α can be set so that the reduction in average access time could be increased to 60% while decreasing the average energy consumption to only 2% less than *RxW/S*.

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. In section 7, we experimentally highlight the importance of broadcasting as a data dissemination technique and we conclude in section 8.

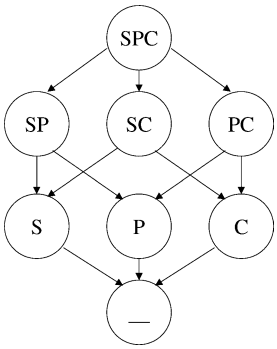

Figure 2. Data cubes lattice.

2. Background and related work

In this section, we will first provide a short introduction to OLAP concepts and then present several scheduling policies that have been proposed for broadcast-pull data dissemination.

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 [18]. 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., *supplier*, *product*, *customer*, *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 [11]. 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 pre-computed and stored as summary tables at the server.

Basically, 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, the OLAP server maintains the three-dimensional table (*supplier*, *product*, *customer*). By adding the measure values across the *customer* dimension, the first client can use the detailed table (*supplier*, *customer*) to extract the abstract table (*supplier*).

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 [12–14]. To facilitate the selection process, the search lattice was introduced in [14]. 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 a simple 3-dimensional schema where the dimensions for analysis are *Supplier* (S), *Product* (P), and *Customer* (C). In the figure, the edges from (SC) to (S) and (C) indicate the dependency of both (S) and (C) on (SC). The symbol '–' is used to represent the summary table with no *Group-By* attributes (that is, everything).

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 [10] and *Most Requests First* (MRF) and *Longest Wait First* (LWF) were proposed as efficient alternatives [10,29]. 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. RxW has been proposed in the context of homogeneous data, where all data objects are of the same size. For the general case, where different data items incur different transmission times, we proposed the priority function $R \cdot W/S$ that incorporates transmission cost which is proportional to the table size [23].

Preemptive scheduling policies have been introduced to minimize the stretch measure in heterogeneous settings, 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 in wireless data communications 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 reading the required data. In a doze mode the mobile device is consuming power orders of magnitude less than that in the active mode. Power conservation indexing methods for single-attribute and multi-attribute based queries in broadcast push environments appeared in [3,15,16].

The idea of merging queries with overlapping answers to reduce broadcast communication cost has been introduced in the context of a multicast subscription environment [9]. 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 [20] for broadcast push in wireless environments, 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 or they have to request the missing items in the form of a supplementary query. The server uses popularity-based scheduling policy to select the data chunks on a broadcast. This assumes that the server has been informed about the clients' queries prior to the beginning of each broadcast cycle.

In our previous work [23,24], we proposed broadcast-based data dissemination techniques for the special case of summary tables (aggregation queries). In our work, we use on-demand data broadcasting that considers the current workload at the server and does not require prior knowledge of the request patterns. In [23], which is a preliminary version of this paper, we used the tables dimensionalities as the criterion to aggregate requests for subsuming summary tables, while in [24], we used the tables sizes as the criterion for aggregation. In [25], we showed how to utilize the Dwarf technology [26] to compress a view and all its subordinate views in order to achieve aggregation of multiple requests. Further, we combined the Dwarf-based scheme with our on-demand scheduling scheme to achieve further broadcast efficiency and scalability.

3. Mobile OLAP model

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 [14]) is an example of commercial product that materialize all the possible summary tables.

We are also assuming a typical mobile computing environment in which mobile devices retain their network connection through the support of specialized stationary hosts called *Base Stations* or *Mobility Support Stations* equipped with wireless communication capabilities. The logical or physical area served by a Mobile Support Station is called a *cell*.

In a single cell configuration, the OLAP server broadcasts the data directly to the mobile clients. In a multi-cell configuration the OLAP server broadcasts the data to the mobile support stations which, in turn, re-broadcast it to the mobile clients within the cell of their responsibility. Without loss of generality, in the rest of the paper we are assuming a single cell configuration.

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.

An uplink request Q is characterized by the set of its Group-By attributes D. Hence, we represent a request as Q^D

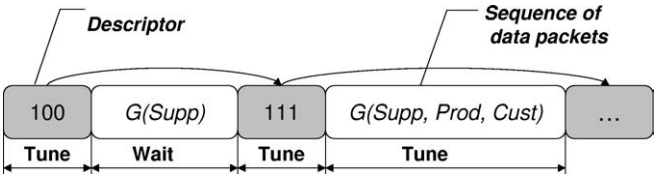

Figure 3. Client access to broadcast.

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 whether 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 $d_1, d_2, \ldots, d_n$. 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 (see figure 3; descriptor packets are shown with gray background). This scheme can be easily extended to include tables with more than one measure and different aggregation functions. In this paper, we limit the presentation to only one measure attribute and *sum()* as the aggregation function.

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.

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

Semantic matching. 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 subsuming match, while $B_2 =$ (*product*) and $B_3 =$ (*supplier*, *customer*) are examples of no match.

Table retrieval. Depending on the matching result and the scheduling algorithm used (as we will see in section 4), the client will either switch to the final retrieval phase or it will stay in the semantic 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 for the next descriptor packet, 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.

Returning to our running example, the access protocol for accessing the complete cube (*supplier*, *product*, *customer*) is shown in figure 3. Here, it is assumed that the search starts by tunning into the descriptor packet for table (100).

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.* It is 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.

$$T_{\text{Total}} = T_{\text{Wait}} + T_{\text{Tune}} + T_{\text{Processing}}.$$

Accordingly, the *total energy consumption* (E_{Total}) is formulated as:

$$E_{\text{Total}} = E_{\text{Doze}} + E_{\text{Active}} + E_{\text{Processing}},$$

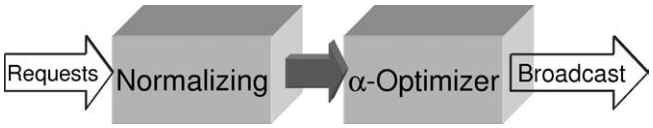

Figure 4. *STOBS-α*.

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.

As mentioned in the introduction and elaborated below, a trade-off between access time and energy consumption may arise in some cases. In section 6, we are introducing a metric to measure the overall gain whenever there is such a trade-off between access time and energy consumption.

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
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 presenting in this section, consists of two components: the *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 (figure 4).

The normalizing component

Given that we are scheduling requests of different sizes and consequently different transmission times, we propose using a prioritizing function $R \cdot W/S$ which considers popularity, size, and age in making the scheduling decision.

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

- R_X: the number of requests for T^X. This value is incremented with every arrival of a request for T^X.
- A_X: the arrival time of the first request Q^X for table T^X currently in the queue. A_X is used to compute the value of W_X which is the waiting time for the request Q^X.
- S_X: 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 $R_X \cdot W_X / S_X$ value for each request in the queue. The request with the highest value is selected to be broadcast.

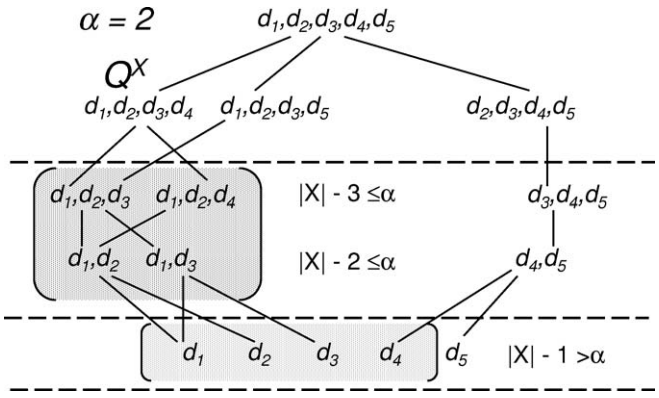

Figure 5. Flexibility and search lattice.

The α-optimization component

The α-optimization component is responsible for aggregating pending requests with the table selected by the normalizing component. 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, a request for T^Y can be discarded and T^Y is not broadcast if $Y \subset X$ and $|X| - |Y| \leqslant 2$. Consequently, a client will use a table T^X that subsumes the table T^Y which the client originally requested and which 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 $|X| - |Y| \leqslant 2$.

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

Q^Y is discarded iff
T^X is broadcast and $Y \subset X$ and $|X| - |Y| \leqslant \alpha$.

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 which has been discarded at the server and would not be broadcast. The value of α is made known to the clients by including it as part of each table descriptor information along with the dimensionality of the broadcast table.

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 *RxW/S* and is a *strict algorithm*, while the family of *STOBS*-α, where $\alpha > 0$, are *flexible algorithms*.

As an example, consider the search lattice shown in figure 5, in which nodes are summary tables. The nodes shown in the figure represent tables for which there exist at least one request. Also, let us 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.

Table 1
Example settings.

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

4.1. Discussion

The intuition for *STOBS*-α is to capture all the specific features of summary tables access in an on-demand broadcast environment. As shown above, the scheduling function $R \cdot W/S$ encapsulates all the basic factors affecting a response access time. The α parameter controls the degree of flexibility. The advantage of the flexibility is to find another aspect of common interest other than the exact strict one. This has the effect of increasing the frequency by which requests for a certain data item are satisfied over a period of time. Further, for a given set of requests it decreases the broadcast length by removing some of the redundancy.

Extra time and energy is 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 [14], 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 $|X| - |high| \leqslant 2$. If the $R \cdot W/S$ 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 for further processing it locally.

To recap, 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, W_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

Table 2
Example results.

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

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_4 is 60 units and its access time using *STOBS*-0 is equal to $W_4 + S_1 + S_4$, where $W_4 + S_1$ is the wait time and S_4 is the tune time.

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 simulated an environment that consists of a single OLAP server and a set of clients. The server receives clients requests for summary tables and disseminates the answers back to clients. We simulated two data dissemination models:

1. *Broadcast model.* In this model, there is a single downlink broadcast channel and a single uplink channel. These channels are shared between all clients. Clients use the uplink channel to send requests to the server and the server broadcasts tables on the downlink channel.
2. *Point-to-point model.* In this model, each client is connected to the server through a dedicated bidirectional channel. On this channel, a client sends its requests to the server and receives the corresponding table.

We used the broadcast model to evaluate the potential gains of using the *STOBS*-α algorithm by comparing it to the strict *RxW/S* (*STOBS*-0). We used the point-to-point model to highlight the efficiency of data broadcasting in general and *STOBS*-α in particular (see section 7).

We generated a synthesized lattice for an n-dimensional data cube. The values of n is in the range between 4 and 12, with $n = 6$ being the default. Setting n to 6 results in 64 (2^6) possible queries and 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 sizes of lattice subcubes is computed as in [17], 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.

Table 3
Simulation parameters.

Parameter	Value	Default
Base cube dimensionality	4–12 dimensions	6
Possible requests	16–4096 requests	64
Zipf parameter (θ)	0.0–0.9	0.5
Simulation length	100 requests/client	
Number of clients	10–250 clients	
α-optimization	0-dimensionality	0, 1, 2, 3, 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.

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

To test the system under a typical workload, requests are generated by the clients according to Zipf distribution with the Zipf parameter (θ) with default value being 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 algorithms performance. In the next section, we use the broadcast model to compare the *STOBS*-α algorithm to *RxW/S*, which is equivalent to the strict *STOBS*-0. Recall that setting α to zero limits requests aggregation to exact matching only, while values of $\alpha > 0$ provide more flexibility in aggregating requests by allowing subsuming matching. We compare the algorithms with respect to access time, energy consumption, overall reduction, impact of database size on performance, and skewness of access.

6. Results

For our evaluation, we took extensive performance measurements. The time reported throughout is in seconds and the energy consumption is in Joules. We considered a wireless LAN where the broadcast channel has a bandwidth of 1 Mbps. We assumed clients are using the IBM ThinkPad laptop [8] that is equipped with Pentium 4 mobile processor with a 100 MHz RAM and 64 bits bus. The processing of a summary table is

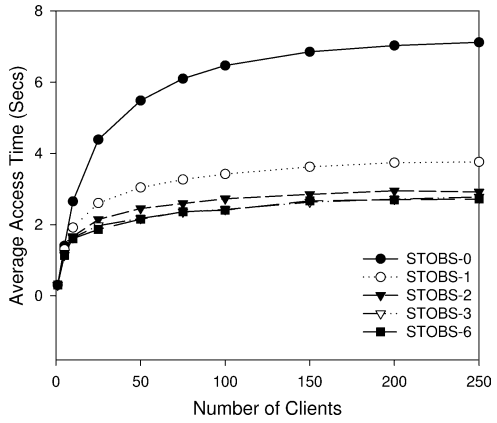

Figure 6. Access time.

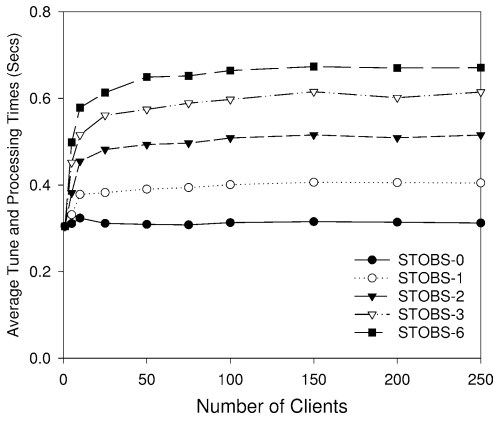

Figure 7. Tune and processing times.

basically a scanning process, and hence we used the memory transfer time to bound the processing time.

6.1. Access time

Figure 6 shows the average access time for 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 250 clients where $\alpha = 2$ (*STOBS*-2). In the case of 10 clients, the average access time decreased by 38% compared to *STOBS*-0 (the basic *RxW/S*), while in the case of 250 clients *STOBS*-2 achieved 60% reduction in the access time compared to the strict *STOBS*-0. In the case of *STOBS*-6, and population of 250 clients, the average access time is 62% less than *STOBS*-0.

Figures 7 and 8 depicts separately the tune, processing and wait times demonstrated in figure 6. The strict algorithm *STOBS*-0 exhibits the minimal tune and processing times. 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 8. It is worth mentioning here, that as the requests arrival rate increases with the increase in the number of clients, 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.

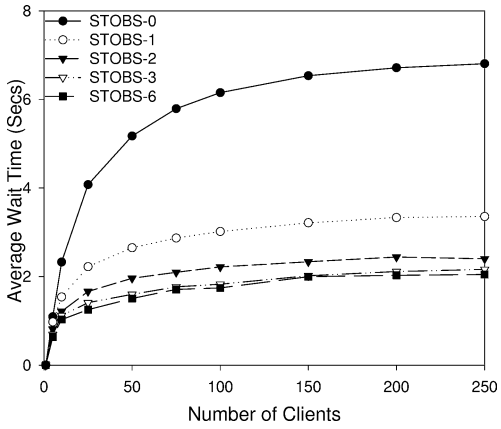

Figure 8. Wait time.

6.2. Energy consumption

In this section, we are using the same experimental settings described earlier. We assume that the processor accesses the downloaded summary table from memory and the memory transfer rate determines the energy needed for processing.

The assumed Pentium 4 mobile processor consumes 2 Watts on average, with a 100 MHz RAM and 64 bits bus. Hence, processing a packet (100 bytes) will take 0.125 μs and the processor energy consumed during this duration equals $0.125\ \mu\text{s} \cdot 2\ \text{W} = 0.25\ \mu\text{J}$.

Let clients be equipped with the ORiNOCO World PC Card [22]. The card operates on a 5 V power supply, using 9 mA at doze mode and 185 mA at receiver mode. Hence, dozing for one packet time will consume $0.8\ \text{ms} \cdot 9\ \text{mA} \cdot 5\ \text{V} = 36\ \mu\text{J}$, while being active tuning to one packet will take $0.8\ \text{ms} \cdot 185\ \text{mA} \cdot 5\ \text{V} = 740\ \mu\text{J}$.

Figure 9 shows the average total energy consumption for the different algorithms. As expected, the extreme case of flexibility ($\alpha = 6$) leads to an increase in the overall energy consumptions. However, reduction in energy consumption is achieved by setting α to the values of 1 or 2. This reduction is more noticeable at higher loads where wait time is longer. In such high loads, doze energy is playing an important role. For instance, consider again the cases of 10 and 250 clients and $\alpha = 2$. In the case of 10 clients, energy consumption in *STOBS*-2 increased by 17% compared to *STOBS*-0, while it decreased by 2% in the case of 250 clients. The minimum energy consumption is provided by *STOBS*-1, which reached 12% less than *STOBS*-0 in the case of 250 clients.

This gain is better illustrated in figure 10 in which the total energy consumption is depicted by its active, process-

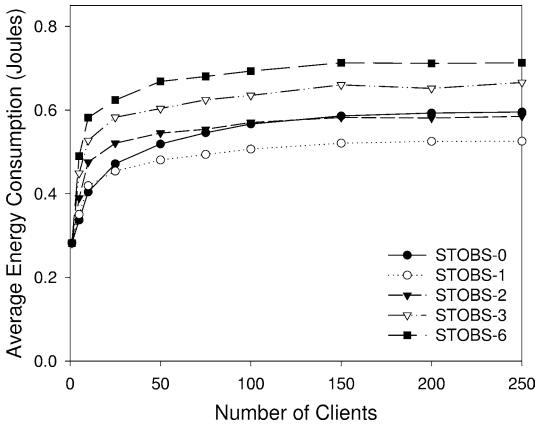

Figure 9. Overall energy consumption.

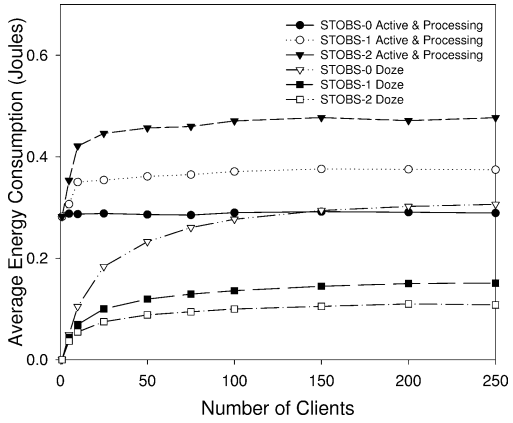

Figure 10. Detailed energy consumption.

ing, and doze components. At the population of 250 clients, *STOBS*-1 provided a 50% reduction in doze energy compared to *STOBS*-0, but, the active and processing energy increased by 30%, leading to the previously observed 12% overall reduction in energy consumption. Additionally, watching the *STOBS*-0 performance, we can see that doze energy consumption is growing with the increase in number of clients until it becomes equally important as the active one. This explains the gains obtained at high loads, where flexibility trades a limited increase in active energy for a substantial decrease in doze energy.

6.3. Overall reduction

Now, let us introduce a metric to measure the overall reduction in average access time and energy consumption, called *overall reduction* (*OR*):

$$OR_\alpha = \frac{T_\alpha/T_0 + E_\alpha/E_0}{2}, \tag{1}$$

where T_α and E_α are the average access time and the average energy consumption for a value of $\alpha > 0$, respectively; and T_0 and E_0 are the average access time and average energy consumption in the case of $\alpha = 0$. The case of $OR_\alpha = 1$ means that overall performance provided by a flexible version of *STOBS*-α is similar to *STOBS*-0. However, a value

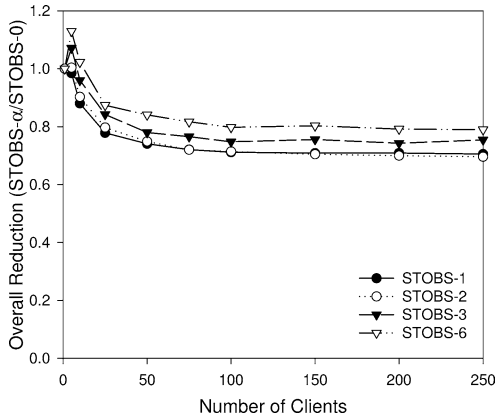

Figure 11. Overall reduction.

of $OR_\alpha > 1$ reflects a degrading in the overall performance, and a value of $OR_\alpha < 1$ shows a reduction in the net performance. Using this metric allows us to compare the relative performance for two settings of α, where the smaller the value of OR_α, the larger the net gain.

Figure 11 shows that the value OR_α decreases by increasing the number of clients. That is, more reduction in the overall performance. In the case of 5 clients, the reduction is insignificant. Moreover, in the case of 5 clients (i.e., extremely light loaded system), the overall performance provided by *STOBS*-3 and *STOBS*-6 is worse than that of the strict *STOBS*-0. This is because of the increase in active energy consumption has not been sufficiently compensated by a corresponding decrease in either doze energy or access time. However, by increasing the number of clients all versions of flexible *STOBS* provide significant reduction in overall performance. For example, in the case of 10 clients, OR_1 equals 0.88 and OR_2 equals 0.9, while in the case of 250 clients, OR_1 equals 0.7 and OR_2 equals 0.68.

6.4. Impact of database size

In this experiment, we are testing the influence of changing the database size on performance, specifically the number of dimensions of the OLAP data cube, which has two effects: (1) it changes the ratio between the sizes of a summary table and its subsuming ones; and (2) it changes the number of generated subcubes. We experimented with number of dimensions varying from 4 to 12, while the number of clients is set to 50.

In figure 12, for clarity of presentation, we are normalizing the flexible versions of *STOBS* to the strict one (*STOBS*-0). As the parameter α can take different values depending on the data cube dimensionality, we picked the values 1, *HALF*, and *MAX*, where *HALF* corresponds to an α that is half the cube dimensionality, while in the case of *MAX*, α is equal to the cube dimensionality. For example, in the case of 10 dimensions, $HALF = 5$, while $MAX = 10$.

It is interesting to observe that all settings of α exhibit the same behavior, where the relative reduction in access time keeps increasing to a certain point where the normalized value is minimal and then it starts decreasing again. The explana-

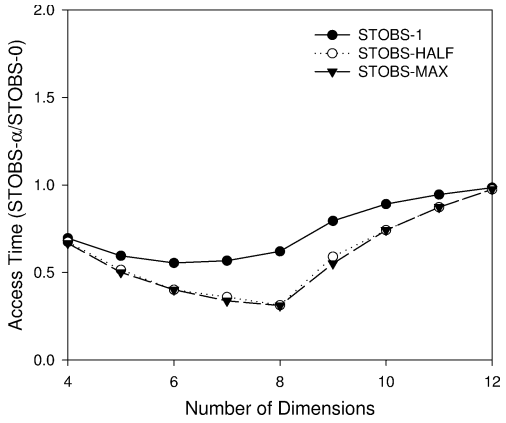

Figure 12. Access time vs. dimensionality.

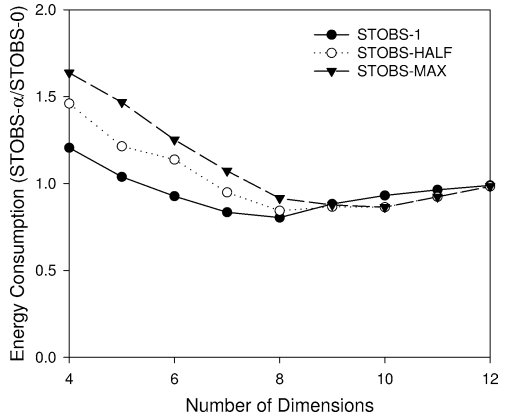

Figure 13. Energy vs. dimensionality.

tion for this is that with a constant number of clients and few dimensions the chances of exact matching between queries are high and the impact of any level of flexibility is small. As the number of dimensions starts growing, the clients interests are spread in a larger lattice of subcubes. In this case, the subsuming matchings offered by flexibility adds significantly to the exact matching ones, yielding to a high degree of sharing. However, as the number of dimensions keeps growing, the frequency of subsuming matches itself is decreasing. This is the case in which attempts to aggregate requests is not beneficial and the system should switch to serving each clients' request independently.

For instance, consider the performance of *STOBS-HALF* in the cases of 4 and 8 dimensions. In the 4 dimensions case, the average access time is 32% less than *STOBS*-0, while in the 8 dimensions it provided a 70% reduction where more flexibility is used to cater the diverging clients interests. However, beyond this minimum point (8 dimensions in our experiment), the chances of the subsuming matching itself start to diminish and the performance of the flexible *STOBS* algorithms is getting close to that of *STOBS*-0. Eventually at 12 dimensions, we can see the flexible *STOBS* algorithms performing similar to *STOBS*-0.

Figure 13 shows that in the case of low dimensionality there is enough exact matching that savings in wait time are not high enough to allow the flexibility to provide any gain

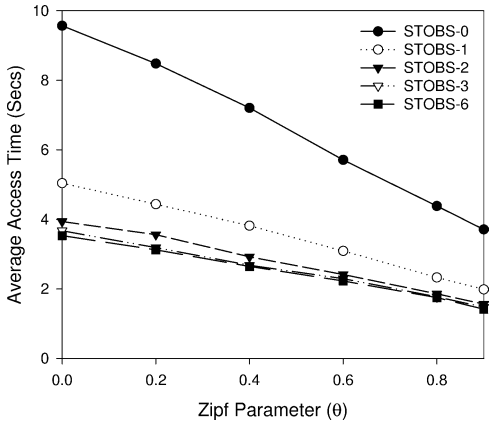

Figure 14. Access time vs. Zipf.

in energy savings. But, as the dimensionality increases, using flexibility starts to significantly reduce the total energy by reducing the doze component. This behavior is sustained until a minimum point, beyond which the savings are decreasing. The explanation is that at high dimensionality, the extra active power consumption due to the flexibility is high compared to the scarce saving in doze energy.

6.5. 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 14 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 access time by 60% less than the *STOBS*-0 for all θ values.

The energy consumption results presented in figure 15 came as expected. That is, the moderate use of flexibility by *STOBS*-1 and *STOBS*-2 showed a reduction in energy consumption. The achieved reduction is due to tackling the doze component. The figure shows that using *STOBS*-1 reduces the energy consumption by 10% less than *STOBS*-0.

7. Broadcasting versus point-to-point

For the sake of completeness, we studied two alternative models for data dissemination. These two models are based on point-to-point data communication, hence, no scheduling is required at the server side. Below, we are comparing these two models to our previously described broadcast data dissemination model. For the broadcast model, we are using *STOBS*-0 and *STOBS*-2 for on-demand scheduling.

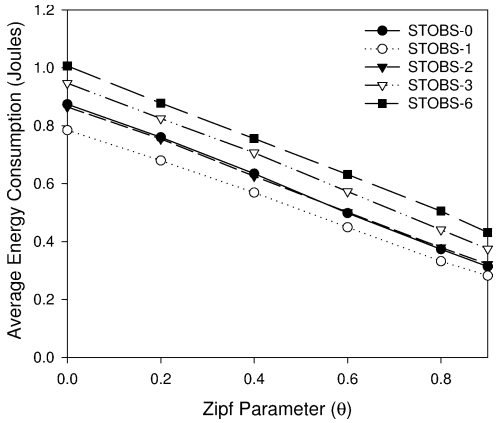

Figure 15. Energy vs. Zipf.

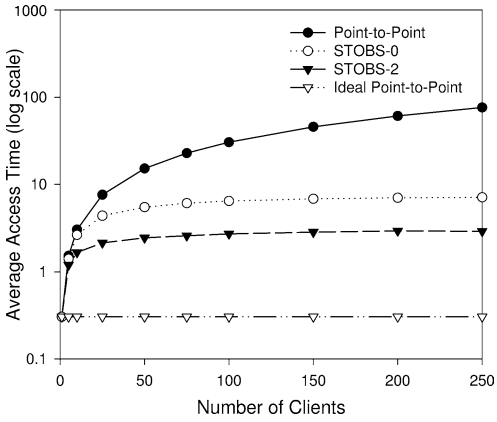

Figure 16. Broadcast vs. point-to-point: access time.

The first alternative model is a *basic point-to-point* model where each client is connected to the server through a dedicated channel. The server bandwidth is evenly divided between the different channels. In this model, adding more clients to the system results in a reduction of the bandwidth available for each and every client. The second one is a hypothetical *ideal point-to-point* model. As in the basic point-to-point model, each client is connected to the server through a dedicated channel. However, in the ideal point-to-point model the bandwidth of each of these channels is equal to our assumed server total bandwidth (i.e., 1 Mbps per channel). Additionally, the number of these channels is unlimited and new channels can be added as new clients join the system.

It is worth repeating that in these two point-to-point models no scheduling is required and hence a client will not experience any wait time. The server receives a client request and instantly sends the exact matching table on the dedicated channel. The access time in these models is just the time needed to tune to the disseminated table and processing it.

We are using the ideal system model as a yardstick to assess the gains provided by the STOBS algorithm with values of $\alpha > 0$ compared to the base case where $\alpha = 0$. Figure 16 shows that the on-demand broadcast data dissemination (*STOBS*-0 and *STOBS*-2) outperforms the point-to-point one. This is because data broadcast takes advantage of the commonality between requests. The figure also shows that

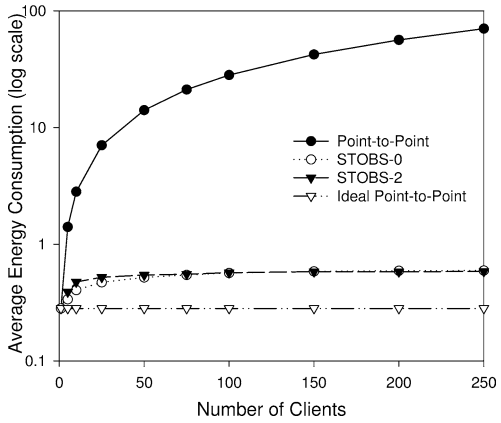

Figure 17. Broadcast vs. point-to-point: energy consumption.

in the case of 250 clients the average access time provided by the ideal setting is 96% less than that provided by the *STOBS*-0. Comparing this reduction to the one provided by *STOBS*-2 shows that *STOBS*-2 is only 36% away from the optimal performance given by that hypothetical ideal model. Figure 17 shows the superiority of on-demand data broadcasting over the point-to-point one in terms of energy consumption. Though point-to-point communication eliminates the doze energy component, however, the increase in tune time leads to an uncompensated corresponding increase in active energy dissipation.

8. Conclusions

In this paper, we focused on time and energy efficient delivery of summary tables to mobile devices equipped with OLAP front-end tools, consequently, enabling decision making anytime and anywhere.

In summary, our major contributions are:

- A new optimization that exploits the derivation semantics among OLAP summary tables 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 with respect to reduction in access time and savings in energy was demonstrated experimentally using simulation.
- We introduced a new metric called *overall reduction*, to capture the trade-off between energy consumption and access time. We used this metric to evaluate the performance of our proposed *STOBS*-α algorithm.
- We re-emphasized the role of broadcast based data dissemination in supporting efficient and scalable access to enterprise data by mobile users. Specifically, we experimentally demonstrated the advantages of using broadcast-based data dissemination over the point-to-point one when delivering OLAP summary tables in wireless and mobile environments.

Although the emphasis of our paper was on wireless and mobile computing environments, our result are applicable

in any networking environment which supports broadcasting/multicasting including wired networks. In fact, in a multicell environment, our scheme can be used by the OLAP server to multicast the summary tables to mobility support stations which in turn broadcast them to the mobile devices within their cells.

Currently, we are working on enhancing the α optimizer to take into consideration the energy levels available at the mobile clients. Specifically, as part of every request, each mobile client will specify its own α value that describes the client's maximum tolerance to increase in energy required for downloading and processing a subsuming table. The server will aggregate requests in a way that minimizes response time while not exceeding the tolerance given by the requesting clients. We are also working on techniques that strongly integrates the flexibility with the selection decision.

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.* (May 1995) pp. 199–210.

[2] S. Acharya and S. Muthukrishnan, Scheduling on-demand broadcasts: New metrics and algorithms, in: *Proc. of 4th Annual ACM/IEEE Conf. MobiCom* (October 1998) pp. 43–54.

[3] R. Agrawal and P.K. Chrysanthis, Efficient data dissemination to mobile clients in e-commerce applications, in: *Proc. of the 3rd Internat. Workshop on E-Commerce and Web Information Systems* (June 2001) pp. 58–65.

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

[5] `http://www.businessobjects.com`

[6] `http://www.cognos.com`

[7] `http://www.hummingbird.com`

[8] `http://www.ibm.com`

[9] A. Crespo, O. Buyukkokten and H.G. Molina, Efficient query subscription processing in a multicast environment (extended abstract), in: *Proc. of the IEEE ICDE Conf.* (February 2000) p. 83.

[10] H.D. Dykeman, M. Ammar and J.W. Wong, Scheduling algorithms for videotex systems under broadcast delivery, in: *Proc. of the Internat. Conf. on Communications* (June 1986) pp. 1847–1851.

[11] 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 IEEE ICDE Conf.* (February 1996) pp. 152–159.

[12] H. Gupta, Selection of views to materialize in a data warehouse, in: *Proc. of ICDT Conf.* (January 1997) pp. 98–112.

[13] H. Gupta, V. Harinarayan, A. Rajaraman and J. Ullman, Index selection for OLAP, in: *Proc. of the ICDE Conf.* (April 1997) pp. 208–219.

[14] V. Harinarayan, A. Rajaraman and J.D. Ullman, Implementing data cubes efficiently, in: *Proc. of the ACM SIGMOD Conf.* (June 1996) pp. 205–216.

[15] Q. Hu, W.-C. Lee and D.L. Lee, Power conservative multi-attribute queries on data broadcast, in: *Proc. of the IEEE ICDE Conf.* (March 2000) pp. 157–166.

[16] T. Imielinski, S. Viswanathan and B.R. Badrinath, Energy efficient indexing on air, in: *Proc. of the ACM SIGMOD Conf.* (May 1994) pp. 25–36.

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

[18] R. Kimball, *The Data Warehouse Toolkit* (Wiley, 1996).

[19] A. Labrinidis and N. Roussopoulos, Update propagation strategies for improving the quality of data on the Web, in: *Proc. of the VLDB Conf.* (September 2001) pp. 391–400.

[20] 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 IEEE ICDCS* (April 2000) pp. 522–529.

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

[22] ORiNOCO World PC Card, `http://www.orinocowireless.com`

[23] M.A. Sharaf and P.K. Chrysanthis, Facilitating mobile decision making, in: *Proc. of the ACM Workshop on Mobile Commerce* (September 2002) pp. 45–53.

[24] M.A. Sharaf and P.K. Chrysanthis, Semantic-based delivery of OLAP summary tables in wireless environments, in: *Proc. of the CIKM Conf.* (November 2002) pp. 84–92.

[25] M.A. Sharaf, Y. Sismanis, A. Labrinidis, P.K. Chrysanthis and N. Roussopoulos, Efficient dissemination of aggregate data over the wireless Web, in: *Proc. of the WebDB Workshop* (June 2003) pp. 93–98.

[26] Y. Sismanis, A. Deligiannakis, N. Roussopoulos and Y. Kotidis, Dwarf: shrinking the PetaCube, in: *Proc. of ACM SIGMOD Conf.* (June 2002) pp. 464–475.

[27] P. Triantafillou, R. Harpantidou and M. Paterakis, High performance data broadcasting: A comprehensive systems' perspective, in: *Proc. of the MDM Conf.* (January 2002) pp. 79–90.

[28] N.H. Vaidya and S. Hameed, Scheduling data broadcast in asymmetric communication environments, Wireless Networks 5(3) (1999) 171–182.

[29] J.W. Wong, Broadcast delivery, Proceedings of the IEEE 76 (1988) 1566–1577.

Mohamed A. Sharaf received the B.Sc. degree with honors in 1997 and the M.Sc. degree in 2000, both in computer engineering from Cairo University. Currently, he is a Ph.D. candidate in the Department of Computer Science at the University of Pittsburgh and is a graduate student research assistant in the Advanced Data Management Technologies Laboratory. His research interests include mobile and pervasive data management, data warehousing, OLAP, and sensor data management. He received the Orrin E. and Margaret M. Taulbee Award for Excellence in Computer Science in 2002 and the Andrew Mellon Predoctoral Fellowship in 2003.
E-mail: msharaf@cs.pitt.edu

Panos K. Chrysanthis is a Professor of Computer Science at the University of Pittsburgh and an Adjunct Professor at Carnegie Mellon University. He received his B.S. from the University of Athens, Greece, in 1982 and his M.S. and Ph.D. from the University of Massachusetts at Amherst, in 1986 and 1991, respectively. His current research focus is on mobile and pervasive data management including sensor networks. In 1995, he was a recipient of the National Science Foundation CAREER Award for his investigation on the management of data for mobile and wireless computing. Besides journal and conference articles, his publications include a book and book chapters on advances in transaction processing and on consistency in distributed databases and multidatabases. He is an editor of the VLDB Journal, and was program chair of several workshops and conferences related to mobile computing. He was the ICDE 2004 Vice Chair for the area of distributed, parallel and mobile databases and the General Chair of MobiDE 2003.
E-mail: panos@cs.pitt.edu