A Game Theoretic Approach to Power Aware Wireless Data Access

Mark Kai Ho Yeung, *Student Member*, *IEEE*, and Yu-Kwong Kwok, *Senior Member*, *IEEE*

Abstract—We consider a basic scenario in wireless data access: a number of mobile clients are interested in a set of data items kept at a common server. Each client independently sends requests to inform the server of its desired data items and the server replies with a broadcast channel. We are interested in studying the energy consumption characteristics in such a scenario. First, we define a utility function for quantifying performance. Based on the utility function, we formulate the wireless data access scenario as a noncooperative game—*wireless data access (WDA) game*. Although our proposed probabilistic data access scheme does not rely on client caching, game theoretical analysis shows that clients do not always need to send requests to the server. Simulation results also indicate that our proposed scheme, compared with a simple *always-request* one, increases the utility and lifetime of *every client* while reducing the number of requests sent, with a cost of slightly larger average query delay. We also compare the performance of our proposed scheme with two popular schemes that employ client caching. Our results show that *caching-only* benefits clients with high query rates at the expense of both shorter lifetime and smaller utility in other clients.

Index Terms—Wireless data access, game theory, caching, invalidation reports, wireless protocol design, simulations, utility.

1 Introduction

Nowadays, we are accustomed to using different types of mobile equipments such as laptop computers, hand-held devices, smart cellular phones, etc. With the popularity of personal area networks (PANs), such as 802.11x [16] and Bluetooth [2], there has been increasingly large amount of information being delivered over the wireless medium. Furthermore, the world-wide deployment of third generation cellular systems (3G) [15] is going to complement the coverage limitations in PANs. 3G, together with PANs, makes ubiquitous information access a reality [25], [27]. One fundamental support to many wireless data applications is to provide efficient *on-demand information access*. Instead of the "push" approach in traditional radio and television services, on-demand information access enables mobile clients to selectively "pull" the desired data from the server.

The major challenge in supporting efficient on-demand information access is conserving clients' battery resources with minimum performance degradation. Indeed, power awareness has become an indispensable issue in the protocol design for wireless networks [4], [6], [13], [20], [28], [30], [33]. In the context of wireless data access, researchers have proposed two techniques to meet the power conservation challenge: 1) data broadcasting and 2) client caching. Data broadcasting allows the server to make a piece of information available to all clients in one single transmission [1], [7], [11]. Client caching means that some data items are cached in clients' local storage for possible future uses. Since data items are updated asynchronously, we require a cache consistency scheme to ensure the validity of the caches.

However, previous research studies [1], [4], [5], [6], [18], [19], [22], [30] mainly focused on quantifying performance improvement from client caching. The effect of maintaining consistency on mobile clients is largely ignored. Specifically, every cache consistency algorithm involves communication overheads between server and clients. For example, the most popular class of consistency schemes mandates that each client retrieves periodic cache invalidation information *indefinitely* from the server. This is not always desirable since the extra energy cost incurred may not justify the amount of energy conserved from caching. Thus, we are interested in designing a power efficient on-demand data access scheme, which takes advantage of data broadcast but avoid the hassle of maintaining cache consistency.

A key motivation of our study is that clients could possibly send duplicated query requests to the server. Since only one query request is needed to make the server broadcast a data item, it is possible for clients to conserve uplink power consumption by reducing these duplicated requests. As such, in this paper we devise a distributed scheme to achieve such objective. First, we design a power aware utility function, which takes client's power consumption in transmit, receive and idle modes into consideration. Specifically, utility is expressed as the number of queries that can be completed given a fixed energy source (see (2) in Section 3.2). Based on the utility function, we formulate our power aware wireless data access scheme as a noncooperative game—*wireless data access (WDA) game*. In the WDA game, each client independently determines its request probability, in the hope of conserving its own uplink power consumption. Although our proposed scheme does not rely on client caching (called "without-cache"), game theoretical analysis shows that clients do not

- *Y.-K. Kwok is with the Department of Electrical and Electronic Engineering, Room 604, Chow Yei Ching Building, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: ykwok@hku.hk.*
- *M.K.H. Yeung is with the Department of Electrical and Electronic Engineering, Room 807, Chow Yei Ching Building, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: mark@eee.hku.hk.*

Manuscript received 12 Nov. 2004; revised 20 Apr. 2005; accepted 22 June 2005; published online 15 June 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0306-1104.

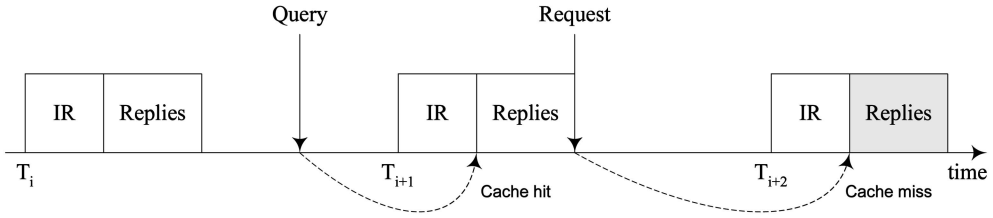


Fig. 1. The IR cache invalidation scheme.

always need to send requests to the server. Simulation results also confirm that our proposed scheme, compared with a simple always-request one, increases the utility and lifetime of every client while reducing the number of requests sent, with a cost of slightly larger average query delay.

We have also compared the performance of our proposed scheme with two popular schemes that employ client caching (with-cache). Our results show that client caching does not always conserve energy for every client. In particular, each client dissipates a significant proportion of energy in maintaining the validity of its cache entries. It is observed that *caching-only* benefits clients with high query rates but results in both shorter lifetime and smaller utility in other clients.

The rest of the paper is organized as follows: Section 2 gives a brief review of the two popular cache invalidation schemes, which are used in Section 5 for performance comparison. In Section 3, we describe the system model used in our study. Based on this system model, we formulate our power aware wireless data access scheme using game theory. We evaluate the performance of our proposed scheme in Section 4. In Section 5, the performance results of with-cache and without-cache approaches are compared. Finally, we discuss the practicality of our proposed scheme in Section 6 and provide some concluding remarks in Section 7.

2 Background

Existing wireless data access schemes are largely based on client caching (called "with-cache"). In this section, we review the two popular cache invalidation schemes for wireless data access, which are used in Section 5 for performance comparison.

Barbará and Imieliński [1] have proposed the basic cache invalidation scheme. The server broadcasts an invalidation report (IR) every L seconds, indicating which data items are updated in the last ωL seconds, where ω is the broadcast window size. Formally, each IR contains the current timestamp, T_i, and a list of pairs (d_x, t_x) such that $t_x > (T_i - \omega L)$, where t_x represents the most recent timestamp of the data item, d_x. To answer a particular query, a client is required to wait for the next IR to determine whether its cache is valid or not. If a valid cached copy exists, the query can be served locally; otherwise, the client issues an uplink query request to the server and waits for the reply in the next IR broadcast period. For example, in Fig. 1, a new query arrives at a time instance between T_i and T_{i+1}. If the client has a valid copy in its cache, the query can be served locally after the IR broadcast at T_{i+1}; otherwise, an uplink request will be issued to the server. The client is expected to obtain the requested data item in the IR broadcast period at T_{i+2}.

Advantages of IR-based approaches include high scalability and energy efficiency: The size of each IR is independent of the number of clients; and IRs are scheduled to be broadcast periodically. As such, clients can switch to doze mode operation between successive IRs to save battery power. The major drawback is that each client must at least wait for the next IR before answering a query to ensure consistency, which potentially causes large query delay.

To reduce the long query delay associated with the basic IR cache invalidation approach, Cao [7] has done some pioneering work in that he proposed the addition of Updated Invalidation Report (UIR) to the original IR scheme (IR+UIR). Each UIR contains invalidation information for those data items that have been updated since the last IR. Formally, an UIR consists of the timestamp T_i of the last IR and a list of pairs (d_x, t_x) such that $t_x > T_i$. In other words, UIRs reflect the most up-to-date view of the database. Clients use UIRs to invalidate their cache and either answer queries locally for cache hits, or otherwise, issue uplink requests. As shown in Fig. 2, the same query arrives at a time instance between T_i and T_{i+1}. For cache hit, the query can be served locally after the next UIR broadcast; otherwise, the client issues an uplink request to the server and waits for the reply after the next IR at T_{i+1}. Therefore,

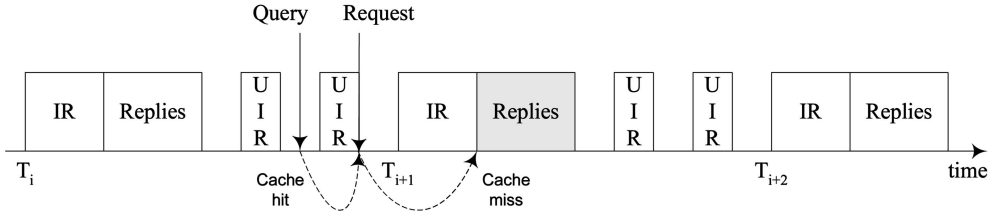


Fig. 2. The IR+UIR cache invalidation scheme.

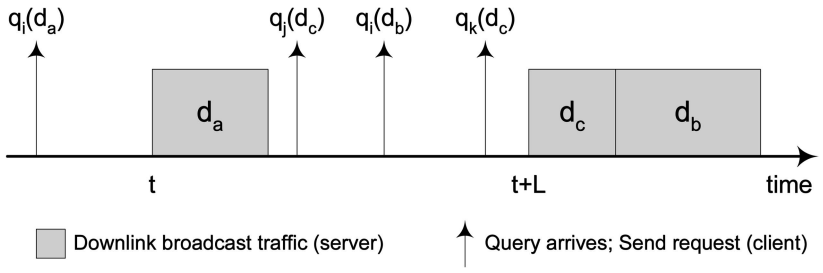


Fig. 3. System model for wireless data access.

IR+UIR reduces the average query latency at the expense of slightly more broadcast overhead in UIRs [6], [7].

Both IR and IR+UIR employ client caching to meet the bandwidth budget and limited lifetime challenges, attempting to improve performance in a wireless environment. A number of enhancement schemes [19], [18], [33] have been proposed to mitigate some of their limitations. We have also done some work in this area [31], [32]. However, previous research results did not quantify the performance improvement from client caching. Specifically, every cache consistency scheme involves different communication overheads between server and clients. In particular, both IR and IR+UIR mandate that each client retrieves periodic cache invalidation information *indefinitely* from the server. However, this is not always desirable since the extra energy cost incurred may not justify the amount of energy conserved from caching. Thus, we are interested in designing a *power efficient* on-demand data access scheme that does not rely on client caching. Clients do not need to process consistency information and maintain their cache entries. This could possibly lead to reduced power consumption among clients.

Finally, we have witnessed that game theoretic analysis has recently been widely applied in various resource management problems in wireless communications [10], [12], [17], [26].

3 Power Aware Wireless Data Access

3.1 System Model with Energy Consideration

Fig. 3 depicts the system model for wireless data access. It consists of a server and a set of clients, N. The clients are interested in a common set of data items, D, which are kept at the server. To request a specific data item, d_a, client i is required to inform the server by sending an uplink request, represented by $q_i(d_a)$. The server then replies with the content of the requested data item, d_a, in the common broadcast channel. This allows the data item to be shared among different clients. As illustrated in Fig. 3, both clients j and k request the same data item, d_c, in the second interval. However, the server is required to broadcast the content of d_c only *once* in the next broadcast period, which reduces the bandwidth requirement. This system model has attracted much attention [1], [4], [5], [6], [7], [18], [19], [29], [31], [32], [33], [34]. However, most previous work focused on the design of cache invalidation schemes. Little has been done on the query process. We would like to investigate the energy consumption characteristics of the above data access model. Specifically, we are interested to conserve energy consumed by the clients.

To successfully complete a query, a client expends its energy in two different parts: 1) informing the server of the desired data item, E_{UL}, and 2) downloading the content of the data item from the common broadcast channel, E_{DL}. The energy cost of sending a request to the server is represented by E_s. If a client sends a request for each query, then E_{UL} would be the same as E_s. However, we show that clients need not to send requests for each query even without caching (see Section 3.2). This implies that E_{UL} does not necessarily equal E_s. In general, the total energy required to complete a query, E_Q, is given by:

$$E_Q = E_{UL} + E_{DL}. \tag{1}$$

It is assumed that E_{DL} is the same for all clients, but its value depends on the size of a data item. In practice, E_s is a function of various quantities [21], [24], including spatial separation, speed, instantaneous channel quality, bit-error rate requirement, etc. For simplicity, however, E_s is also assumed to be a fixed quantity.

3.2 Problem Formulation

Based on the energy consumption model in Section 3.1, we formulate our power aware wireless data access scheme as follows: Define E_{total} as the amount of energy available to each client. We use the *number of queries that can be completed* to quantify the performance of a wireless data access scheme. Mathematically, the *utility* of client i's is given by:

$$U_i = \frac{E_{total}}{E_Q^i}. \tag{2}$$

Our objective is to reduce the amount of energy consumed in the query process such that *every* client's utility ((2)) is increased.

For a simple data access scheme, defined as "simple access," the client is required to send an uplink request to inform the server whenever a new query arrives, i.e., $E_{UL} = E_s$. Thus, we have:

$$U_i = \frac{E_{total}}{E_s + E_{DL}}. \tag{3}$$

Due to data locality, some data items are more popular than the others [3]. It is possible that more than one client independently requests the same piece of "hot" data item. However, these duplicated requests are a waste of 1) battery energy and 2) uplink bandwidth. In fact, these requesting clients, but *not* all, are more advantageous to "back-off" and let some other to send the request to the server on behalf of themselves. Ideally, only one request is needed to trigger

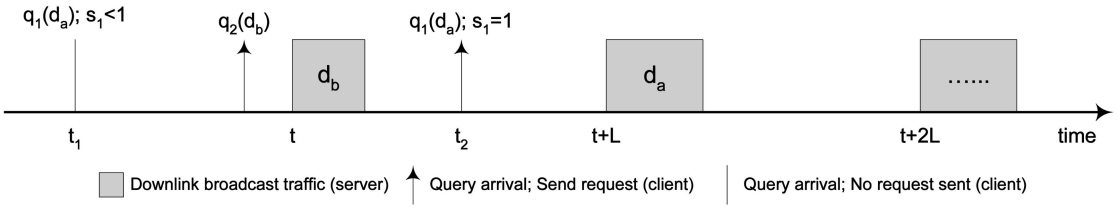


Fig. 4. Client 1 chooses not to request at t_1, but cannot find d_a in the first server broadcast. Then, it requests with probability 1 at t_2.

the server to broadcast the data item. However, this would require *explicit* coordination among clients. Such extra communication overheads induce more interference and may not justify the energy conserved from uplink requests. More importantly, the client, who actually sends the request, expends its own battery energy and possibly bears monetary cost for the goodness of others. Without appropriate incentive measures, there is obviously *no* client willing to take up the requesting role.

To analyze the above conflicting situation, we model the data access scenario as a static noncooperative game—*wireless data access (WDA) game*—each *player* (client) maximizes its own *utility* (2) with *no* explicit communication with one another.

3.3 System Analysis

Denote $N = \{1, \ldots, n\}$ as the set of players (clients). Each player determines its *request probability*. The strategy space of player i is given by, $S_i = \{s_i | 0 \leq s_i \leq 1\} \subset \Re^1$. The strategy combination is denoted as, $s = (s_1, \ldots, s_n) \in S$, where $S = \times_{j \in N} S_j \subset \Re^n$ is the Cartesian product of the n players' strategy spaces. Furthermore, define

$$s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n) \in S_{-i},$$

where $S_{-i} = \times_{j \in N \setminus \{i\}} S_j \subset \Re^{n-1}$, as the strategy combination of all the players, except i. $U_i(s) \in \Re^1$ represents the utility of player i when the strategy combination is s. A strategy combination, s^*, is said to achieve the state of *Nash equilibrium* when:

$$U_i(s^*) \geq U_i(s^*_{-i}, s_i) \quad \forall s_i \in S_i, i \in N. \tag{4}$$

Notice that the utility function, U_i, depends on every player's strategy, s_i, which in turn affects the strategies of all the other players, s_{-i}. In other words, a strategy combination is said to achieve the state of *Nash equilibrium* when *no* player can improve its utility by *unilaterally* deviating from its own strategy. A salient feature is that there is *no* coordination among the set of players. In general, a game may have multiple equilibria or even none at all.

In the WDA game, the set of clients represents game players. Client i's strategy, s_i, is to determine its request probability, which is the probability of sending an uplink request to the server. In particular, "$s_i = 1$" represents that client i always send a request upon query arrival. Similarly, "$s_i = 0$" means the client never sends any requests but wait. A strategy combination, s, is a vector, with the ith element represents client i's request probability.

In general, the strategy combination: $s = (1, \ldots, 1)$, i.e., always request, does not constitute a Nash Equilibrium because it is beneficial for client i to deviate. If client i withholds sending its requests, the server would still broadcast the reply due to requests from other clients. On the other hand, the strategy combination: $s = (0, \ldots, 0)$, i.e., "always wait," should not be a Nash Equilibrium. This is because the server would not broadcast any query reply. Clients expends all their energy in waiting and result in zero utility.

3.3.1 WDA Game—Two-Person Version

To study the performance of the WDA game, we start by analyzing the two-person version, i.e., $N = \{1, 2\}$. There are two clients in the system, each of which determines its request probability, s_i, independently. If client 1 chooses *not* to request, there are two possible consequences:

1. Client 2 sends a request for the same data item. Server then broadcasts the data item in the next scheduled period. Thus, the uplink cost for client 1 is zero, i.e., $E_{UL} = 0$;
2. Client 2 *does not* send a request for the same data item. Client 1 cannot complete the query but dissipates energy in waiting.

In the second case, if client 1 is not patient enough to wait for another period, its strategy is to send the request with probability 1, as illustrated in Fig. 4. The uplink costs for clients 1 and 2 are, respectively,

$$E^1_{UL} = s_1 E_s + (1 - s_1)(1 - s_2)(E_s + E_w), \tag{5}$$

$$E^2_{UL} = s_2 E_s + (1 - s_1)(1 - s_2)(E_s + E_w), \tag{6}$$

where E_w is the energy cost of waiting per broadcast period.

Thus, the utility function ((2)) takes power consumption in transmit, receive and idle modes, into consideration. From (5) and (6), we observe that the utility functions are symmetrical. This motivates us to search for symmetric equilibrium strategies:Let $E_w = \alpha E_s$ and differentiate U_1 with respect to s_1 gives:

$$\frac{\partial U_1}{\partial s_1} = -\frac{E_{total} E_s}{E^1_{UL} + E_{DL}} \{s_2(1 + \alpha) - \alpha\}. \tag{7}$$

Depending on the value of s_2, $\frac{\partial U_1}{\partial s_1}$ takes on different values:

1. $s_2 < \frac{\alpha}{1+\alpha} \Rightarrow \frac{\partial U_1}{\partial s_1} > 0$.
   Client 1's best-reply strategy is, $s^*_1 = 1$, which reduces to the original simple access scheme.
2. $s_2 > \frac{\alpha}{1+\alpha} \Rightarrow \frac{\partial U_1}{\partial s_1} < 0$.

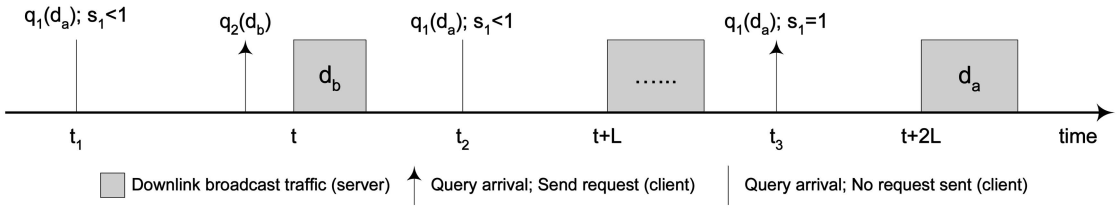


Fig. 5. Client 1 chooses not to request at t_1 and t_2 but it cannot find d_a in both the first and the second server broadcasts. Then, it requests with probability 1 at t_3.

Client 1's best-reply strategy is, $s_1^* = 0$, i.e., client 2's request probability is so large that client 1 is always advantageous to wait for a broadcast period.

3. $s_2 = \frac{\alpha}{1+\alpha} \Rightarrow \frac{\partial U_1}{\partial s_1} = 0$.
   The best-reply for client 1 is any feasible strategy, i.e., player 1 is indifferent between request and wait. In particular, the strategy combination, $(s_1^*, s_2^*) = (\frac{\alpha}{1+\alpha}, \frac{\alpha}{1+\alpha})$, achieves a *weak Nash equilibrium*.

To avoid the weak equilibrium and the associated degenerated solutions, consider the case that client 1 is patient enough to wait for one extra broadcast period. In other words, client 1 waits for a maximum of two server broadcasts before resolving to request with probability one (see Fig. 5). The possible values of uplink energy cost are:

1. 0,
2. E_s,
3. $E_s + E_w$, and
4. $E_s + 2E_w$.

Recall that the request probabilities of client 1 and client 2 are s_1 and s_2, respectively. In particular, the term $(1 - s_1)(1 - s_2)$ represents the probability that neither of them send a request. Thus, the expected uplink energy cost, E_{UL}, becomes:

$$\begin{aligned} E_{UL} &= s_1 E_s + (1 - s_1)(1 - s_2) \\ &\quad \{s_1 E_s + E_w + (1 - s_1)(1 - s_2)(E_s + E_w)\} \\ &= E_s\{s_1 + (1 - s_1)(1 - s_2)(s_1 + \alpha) \\ &\quad + (1 - s_1)^2(1 - s_2)^2(1 + \alpha)\}. \end{aligned} \tag{8}$$

Fig. 6 shows a plot of client 1's utility (2) against different values of s_2 using the numerical values: $E_{total} = 100$, $E_s = 0.0278$, $E_{DL} = 0.0111$, and $\alpha = 0.5$. There are a number of observations:

1. If client 1 always requests, i.e., $s_1 = 1$, its utility is independent of client 2's strategy, s_2.
2. The *maximum* utility of client 1 increases with s_2.
3. For small values of s_2, the optimal strategy for client 1 is to always request, $s_1^* = 1$.
4. For large values of s_2, the optimal strategy for client 1 is to always wait, $s_1^* = 0$.
5. For some values of s_2, the optimal strategy for client 1 is in the interior of the strategy space, i.e., $s_1^* \in (0, 1)$.

From (8), the best-reply strategy for client 1 is shown to be:

$$s_1 = \frac{2(1+\alpha)(1-s_2)^2 - (1-\alpha)(1-s_2) - 1}{2(1+\alpha)(1-s_2)^2 - 2(1-s_2)}. \tag{9}$$

Fig. 7 shows a plot of (9) when $\alpha = 0.5$. It is observed that there is only one *fixed-point* solution within the feasible strategy space, i.e., $s_1^* = s_2^* = 0.4131$, which is the unique equilibrium strategy of the game.

Now, let us try to determine the optimal symmetric strategy. To do so, we take $s_1 = s_2 = s$ in (8) which is then substitute into the utility function. Setting $\frac{\partial U}{\partial s} = 0$, we get:

$$4(1+\alpha)s^3 - 3(3+4\alpha)s^2 + 2(4+7\alpha)s - 2(1+3\alpha) = 0. \tag{10}$$

Solving this cubic equation for s, we obtain only one real root also: $s = 0.6567$. Fig. 8 shows the variation of utility with symmetrical strategies ($s_1 = s_2$). The optimal strategy is $s_1 = s_2 = 0.6567$; while the equilibrium strategy is

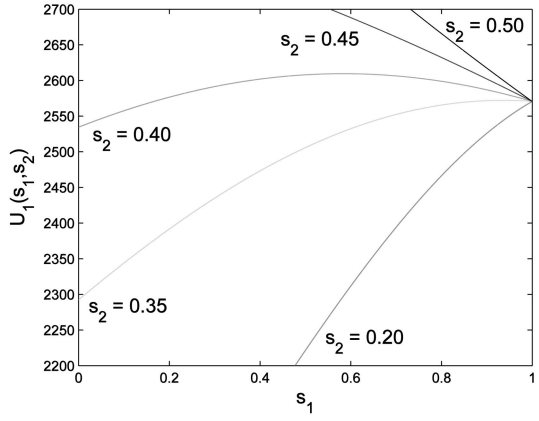


Fig. 6. $U_1(s_1, s_2)$ versus s_2.

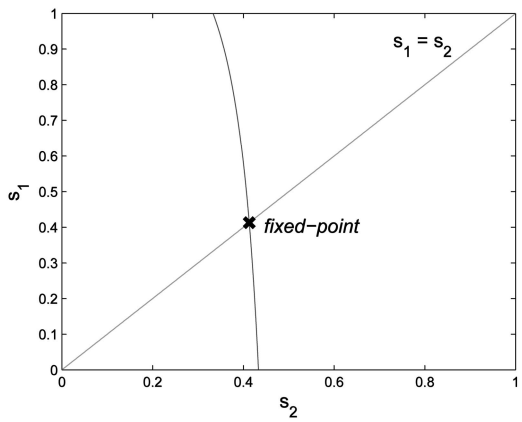


Fig. 7. s_1 versus s_2.

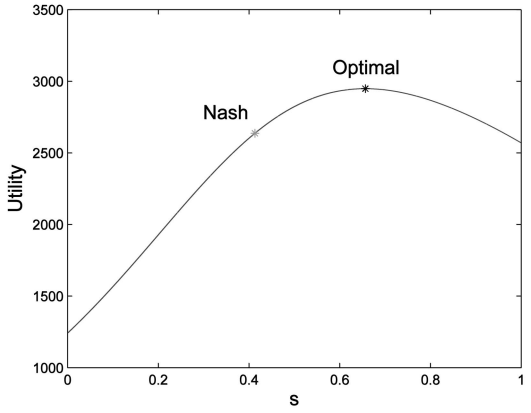


Fig. 8. Utility versus s.

$s_1 = s_2 = 0.4131$. Thus, the equilibrium utility is *Pareto inefficient*, which is a common characteristic in noncooperative game models. In the case of our WDA game, it is possible to leverage the presence of the central server such that each client plays the optimal strategy (see Section 3.4).

Finally, we give a justification for the terminating condition: client 1 sends request with probability one after two serverbroadcasts. First, it is intuitive for clients to bound the query delay. Second, if client 1 does not impose the terminating condition, the uplink energy cost would become:

$$E_{UL}^1 = E_s \sum_{i=0}^{\infty} (1-s_1)^i (1-s_2)^i (i\alpha + s_1) = E_s\{\alpha \sum_{i=1}^{\infty} i\Omega^i + s_1 \sum_{i=0}^{\infty} \Omega^i\} = E_s\{\frac{\alpha\Omega}{(1-\Omega)^2} + \frac{s_1}{(1-\Omega)}\}, \tag{11}$$

where $\Omega = (1-s_1)(1-s_2)$.

To determine the optimal symmetric strategy, we take $s = s_1 = s_2$ in (11). Setting $\frac{\partial U}{\partial s} = 0$ and after some manipulations, we have:

$$s^4 - 2(\alpha+1)s^3 + 6\alpha s^2 - 8\alpha s + 4\alpha = 0. \tag{12}$$

Solving (12) for s, we obtain the only feasible strategy with $s = 0.6579$, which is similar to the result obtained from using the terminating condition. Therefore, the terminating condition results in a comparable strategy and achieves bounded query delay.

3.3.2 WDA Game—n-Person Version

Based on the analysis in Section 3.3.1, the 2-person WDA game is transformed to the n-person version as follows:

If client i has a query pending for the data item, d_x, the equilibrium strategy, s_i^*, is used. Otherwise, $s_i = 0$ for that data item, since client i is not interested in d_x. Comparing with (8), client i's uplink energy cost for d_x is given by:

$$E_{UL}^i = E_s\left\{ s_i + (s_i+\alpha)\prod_{j\in M}(1-s_j) + (1+2\alpha)\prod_{j\in M}(1-s_j)^2 \right\}, \tag{13}$$

where $M \subseteq N$ (i.e., $|M| = m \le |N| = n$) is the set of clients interested in d_x.

TABLE 1
Optimal Request Probabilities with and without the Terminating Condition

m	s^* (with termination)	s^* (without termination)
2	0.6567	0.6579
4	0.4136	0.4066
6	0.3159	0.3075
8	0.2599	0.2520
10	0.2228	0.2157

Thus, the symmetric equilibrium strategy is given by:

$$s^* = \frac{2(1+2\alpha)\chi^2 - (1-\alpha)\chi - 1}{2(1+2\alpha)\chi^2 - 2\chi}, \tag{14}$$

where $\chi = (1-s^*)^m$.

Theorem 1. *There exists an equilibrium strategy for the n-person WDA game.*

Proof. First, each player's strategy space, $S_i \in \Re^1$, is nonempty, convex and compact. Second, the utility function, U_i, are continuous on S, $\forall i \in N$. Furthermore, the best-reply mapping is single-valued. Thus, we can conclude that there exists an equilibrium strategy for the n-person WDA game [14], [23]. □

Although Theorem 1 does not rule out the possibility of more than one equilibrium, we observe, from simulations, that the equilibrium strategy obtained from (14) appears to be unique.

Similar to the 2-person's case, we can determine the optimal symmetric strategy by considering a homogeneous set of s_i in the uplink energy costs. With the symmetric homogeneous uplink costs, setting $\frac{\partial U}{\partial s} = 0$ leads to:

$$1 - m(s+\alpha)(1-s)^{m-1} + (1-s)^m - 2m(1+2\alpha)(1-s)^{2m-1} = 0. \tag{15}$$

It can be shown that this equation has only one legitimate real root, which is the optimal strategy. If we remove the terminating condition, the uplink energy cost for client 1 in the n-person version becomes: $E_{UL}^1 = E_s\{\frac{\alpha\Omega}{(1-\Omega)^2} + \frac{s_1}{(1-\Omega)}\}$, where $\Omega = \prod_{j\in M}(1-s_j)$. In this case, the optimal symmetric strategy is the solution to the following equation:

$$1 - [(\alpha m + 2) + (m-2)s](1-s)^{m-1} + [(1-\alpha m) + (m-1)s](1-s)^{2m-1} = 0. \tag{16}$$

The unique feasible solution to (15) and (16) represents the optimal strategy with and without the terminating condition, respectively. Table 1 shows the optimal request probabilities with and without the terminating condition. Since both cases result in comparable strategies, we adopt the terminating condition in our protocol to achieve bounded query delay.

3.4 The Data Access Protocol

To achieve the stated strategy, we have designed algorithms for server and clients, which are formalized as follows:

Each client executes Algorithm 1 to determine its request probability. Clients do not solve (9) every time. Instead, the solutions are calculated and stored to perform table lookup. Each client determines its strategy independently based on the m value announced by the server (see below). It should be emphasized that there is no coordination among clients. If there is more than one query pending, the client plays with the joint request probability.

Using Algorithm 2, the server keeps track of the set of m and announces the values to each client via periodic broadcast. For $m = 2$, the request probability is 0.4132 (see Section 3.3.1). If the server intentionally announces $m = 1.4668$, each client would play 0.6567, which corresponds to the optimal strategy. Thus, clients plays with the optimal request probability as if it is an equilibrium strategy. This requires the server to perform the actual-to-optimal m mapping, which is implemented as table lookup for efficiency. The mapping process is analogous to introducing a cost function to the original utility [26] such that the new equilibrium strategy achieves *Pareto improvement* over the original one. To estimate the values of m, it is assumed that the server knows 1) the mean query generation time of a typical client, e.g., obtained from previous usage statistics, and 2) the number of alive clients, e.g., from the network layer. From the above information, the server can estimate the values of m as indicated in Algorithm 2.

Server runs Algorithm 3 to reply clients' queries. In addition to the data content, the server also broadcasts an index list, similar to the one used in [6], [7], [29]. Using the index list, clients can tune to their desired data item(s) and ignore the others. This further conserves clients' energy in monitoring the broadcast channel, which is also costly [13].

Algorithm 1 Client—Query
1: INPUT: L; $\{m_j\}$
2: OUTPUT: Q (query list); s (request probability)
3:
4: **for** each new query, $q_i(d_j)$ **do**
5: Use m_j to determine s^* from (14);
6: $s = 1 - (1 - s)(1 - s^*)$;
7: $Q = Q \cup d_j$;
8: Send(s);
9: **end for**
10: **for** each server broadcast period, L **do**
11: **if** $Q \neq \emptyset$ **then**
12: Check the index list, I;
13: Download the desired data item(s);
14: **end if**
15: **if** $Q \neq \emptyset$ **then**
16: **if** a query has missed two server broadcast **then**
17: s = 1;
18: **end if**
19: Send(s);
20: **end if**
21: **end for**
22: Function Send(s)
23: **if** $rand < s$ **then**
24: Send a request;
25: $Q = \emptyset$; $s = 0$;
26: **end if**

TABLE 2
Simulation Parameters—Wireless Data Access

Parameter	Value
Database size ($\|D\|$)	1,000
Data item size	1KB–10KB
Broadcast interval (L)	20s
Client population size ($\|N\|$)	200
Transmit power	0.5W
Receive power	0.1W
Relative cost of waiting (α)	0.5
Bandwidth	384Kbps
Energy available[1] (E_{total})	100J
Minimum query generation time ($T_q^{\min}$)	5s
Maximum query generation time ($T_q^{\max}$)	50s
Uplink request size	1KB

Algorithm 2 Server—m-values
1: INPUT: $p_j(\theta); L; T_q^i, i \in N' \subseteq N$ (*alive* clients)
2: OUTPUT: $\{m_j\}, j \in D$
3:
4: **for** $j = 1$ to $|D|$ **do**
5: $m'_j = p_j(\theta) \sum_{i \in N'} \frac{L}{T_q^i}$;
6: $m'_j \rightarrow m_j$, using actual-to-optimal m table;
7: **end for**

Algorithm 3 Server—Reply
1: OUTPUT: I (index list); Query replies
2:
3: **for** each received query $q_i(d_x)$ **do**
4: $I = I \cup d_x$;
5: **end for**
6: **for** each server broadcast period, L **do**
7: Broadcast I;
8: Broadcast the content of data items in I;
9: **end for**

4 Performance Evaluation

We evaluate our proposed game theoretic power aware wireless data access scheme using MATLAB simulation. In particular, system performance is quantified in terms of:

1. *Utility*—number of queries completed;
2. *Lifetime*—time taken to deplete the available energy;
3. *Uplink traffic*—number of uplink requests sent; and
4. *Average query delay*—time between query arrival and complete reception of server's reply.

We also study the effect of:

1. Client population size, $n = |N|$,
2. Query generation time, T_q, and
3. Access Skew, θ,

on the above four performance metrics.

4.1 Simulation Configuration and Parameters

Table 2 shows the values of major simulation parameters, most of which are similar to those used in [7], [22], [33]. Each client generates a stream of exponentially distributed

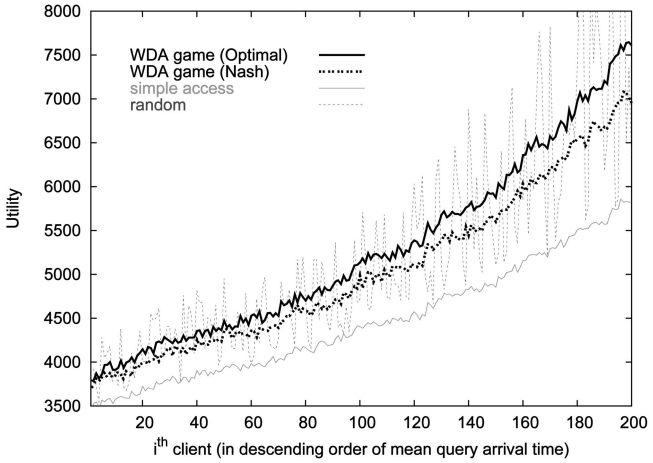


Fig. 9. Utility distribution across all clients.

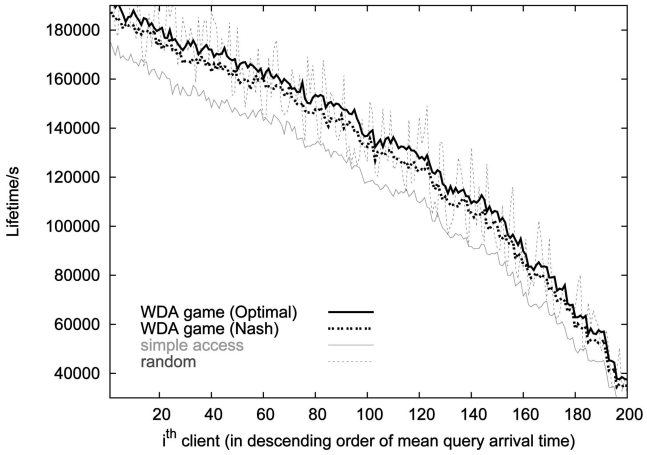


Fig. 10. Lifetime distribution across all clients.

queries with mean generation time, T_q, drawn uniformly between $T_q^{\min}$ and $T_q^{\max}$. This models different levels of interest on the database among the set of clients. We use the *Zipf-like* distribution [13], with $\theta = 0.9$, to model the nonuniform access pattern. Each client queries the jth data item with probability given by:

$$p_j(\theta) = \frac{1}{j^{\theta} \sum_{k=1}^{|D|} \frac{1}{k^{\theta}}}, \tag{17}$$

where $0 \leq \theta \leq 1$.

The value of θ determines the "skewness" of the access pattern. $\theta = 1$ gives the strict *Zipf* distribution while $\theta = 0$ results in the uniform distribution. To focus on the data access process, the effect of client caching is not considered. Thus, clients do *not* need to expend energy to obtain and process any consistency information. Server replies clients' queries every L seconds and announces m-values whenever the set of *alive* clients changes. It is assumed that both the uplink and downlink bandwidth are used exclusively for data access.

In the figures, "WDA game (Optimal)" and "WDA game (Nash)" represents the results with and without the mapping of m values discussed in Section 3.4, respectively. "simple access" represents the simple case that clients always send a request upon query arrival. "random" means that clients choose their request probabilities randomly.

4.2 Utility Results

Fig. 9 shows the number of queries completed (*utility*) across all clients. Notice that client indices are in descending order of their mean query arrival time, i.e., the first client has the lowest query rate. First, we observe that each client's utility is increased in the WDA game compared to the simple access scheme. This is because each client individually and independently optimizes its utility function (2) to determine its equilibrium request probability. If a client chooses not to request, the client expends energy to wait for server's reply. Although the reply may not appear, our analysis shows that the decision would be advantageous in terms of expectation, which is confirmed by the simulation results. Second, clients with higher utility values show larger improvements, since they benefit more due to their frequent queries. On the other hand, low-utility clients dissipate the majority of their energy in idle state and do not show significant improvement in utility. Third, clients using the random strategy suffer from large discrepancy in terms of utility. Some achieves larger utility values while some become worse. These fluctuations are obviously undesired.

The lifetime of *each client* is shown in Fig. 10. The lifetime of a client is defined as the time taken to deplete its available energy, E_{total}. First, the lifetime of *every client* is increased in the WDA game compared to the simple access scheme. Second, we observe that high-utility clients have relatively shorter lifetimes. This is because of their high query generation rates: Energy dissipates more rapidly in sending requests to the server. As such, they deplete their energy source earlier. Third, low-utility clients achieve larger improvements in lifetime compared to the high-utility ones. It is because the energy conserved from uplink requests has a more significant effect in the low-utility group, which is mainly in the idle state with low power consumption. Similar to the case of utility, "random" strategy gives large fluctuations in terms of lifetime.

Fig. 11 shows the number of requests sent for the ith client. We observe a similar result: *each client* sends

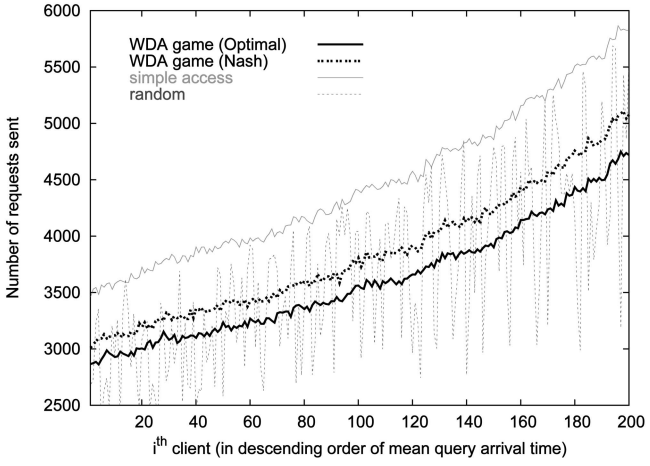


Fig. 11. Number of requests for all clients.

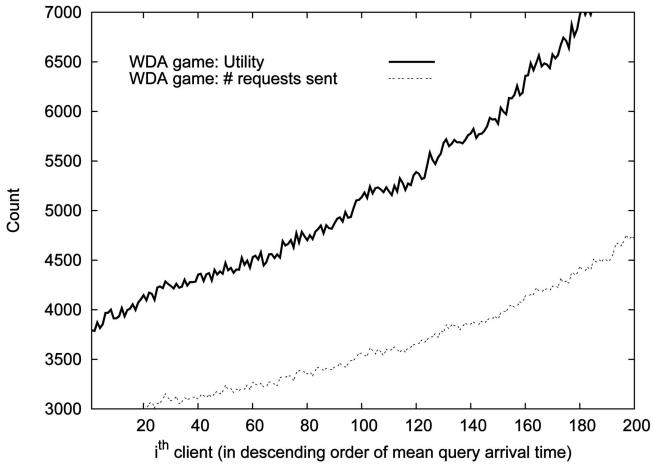


Fig. 12. Performance of the WDA game: utility and number of requests sent.

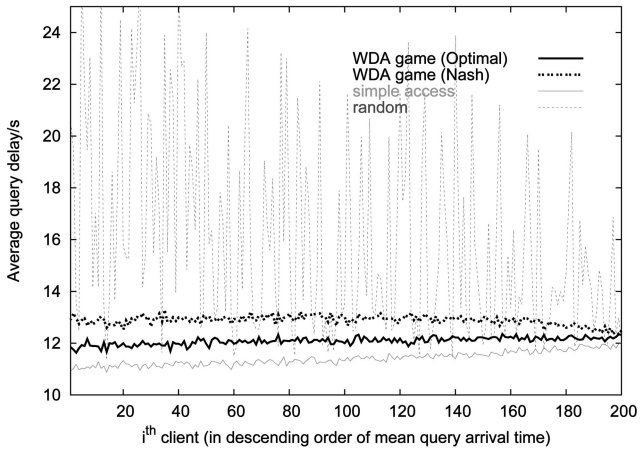


Fig. 13. Average query delay distribution across all clients.

fewer requests to the server. This provides an explanation for the increase in lifetime of *every client*. Unlike the previous two metrics, each client shows a very similar reduction in the number of requests sent. This is due to the common access pattern, i.e., the $Zipf$-like distribution, used in the simulation model. The result also suggests that similar amount of energy is conserved among the clients. As such, all clients benefit from playing the game. Besides the energy conservation effect, fewer requests sent also reduces: 1) uplink bandwidth requirement, 2) collision probability, and 3) interference to other users.

Fig. 12 shows more details of the clients in the WDA game. On average, each client requires to send only about 70-80 requests to complete 100 queries. The actual amount of energy conserved depends on the relative power consumption in transmit and receive modes, which depends on the types of the wireless networks [20].

The performance improvements come with a cost: increase in average query delay, which is defined as the time between arrival of a query and complete reception of the reply from the server. Fig. 13 shows the clients' average query delay in the WDA game and simple access scheme. As expected, clients playing the WDA game experience slightly longer average query delay. This is inherent in the design of the game: wait for some one to make the request. However, the delay is bounded by the additional requirement: If the desired data item does not appear in the immediate *two* server's broadcasts, the client is forced to send a request with probability one. For the case of "random," we observe that clients suffered from significantly larger delay. Although the random strategy may give better results in terms of utility and lifetime, the performance degradation in terms of delay is very large. Thus, clients would not arbitrarily choose its request probability.

4.3 Effect of Client Population Size, $|N|$

Fig. 14 shows the effect of client population size, $|N|$, on the four performance metrics—average utility per client, 20 percent-lifetime, number of requests sent per client, and average query delay.

The average utility per client with different number of clients is shown in Fig. 14a. We observe that the average utility remains unchanged in the simple access scheme.

We do not quantify the performance in terms of the first client to deplete its energy. Instead, the "20 percent-lifetime" is used, which is defined as the time taken for the first 20 percent of the total clients to deplete their energy sources. For example, if $|N| = 200$, "20 percent-lifetime" is the time of the 40th client to deplete its energy. This metric gives a more stable result and better estimate of the system's lifetime.

Fig. 14b shows the 20 percent-lifetime in different client population sizes. For the WDA game, the 20 percent-lifetime *increases* with client population size. This is because a larger client population size gives larger values of m in Algorithm 2. Thus, clients use a lower equilibrium request probability and complete more queries with its limited energy source. Finally, we can see that the WDA game is capable of increasing the "20 percent-lifetime" to a similar extent in different client population sizes.

As shown in Fig. 14c, client population size does not affect the average number of requests sent per client in a simple access scheme. On the other hand, a larger client population size reduces the number of requests sent from each client, which is consistent with the result in Fig. 14b.

Fig. 14d shows the variations of average query delay with client population size: WDA game always gives a larger average query delay than a simple access scheme. A smaller request probability is used when there are more clients. A larger client population size also makes the desired data item more likely to be requested by others. These two effects counteract each other and result in the similar difference in delay, compared with the simple access scheme.

4.4 Effect of Query Generation Time, T_q

Fig. 15 shows the effect of query generation time, T_q, on the four performance metrics—average utility per client, 20 percent-lifetime, number of requests sent per client, and average query delay. The query generation time is uniformly distributed from T_q^{min} to T_q^{max}. We fix T_q^{min} and vary T_q^{max} from 10 to 1,000.

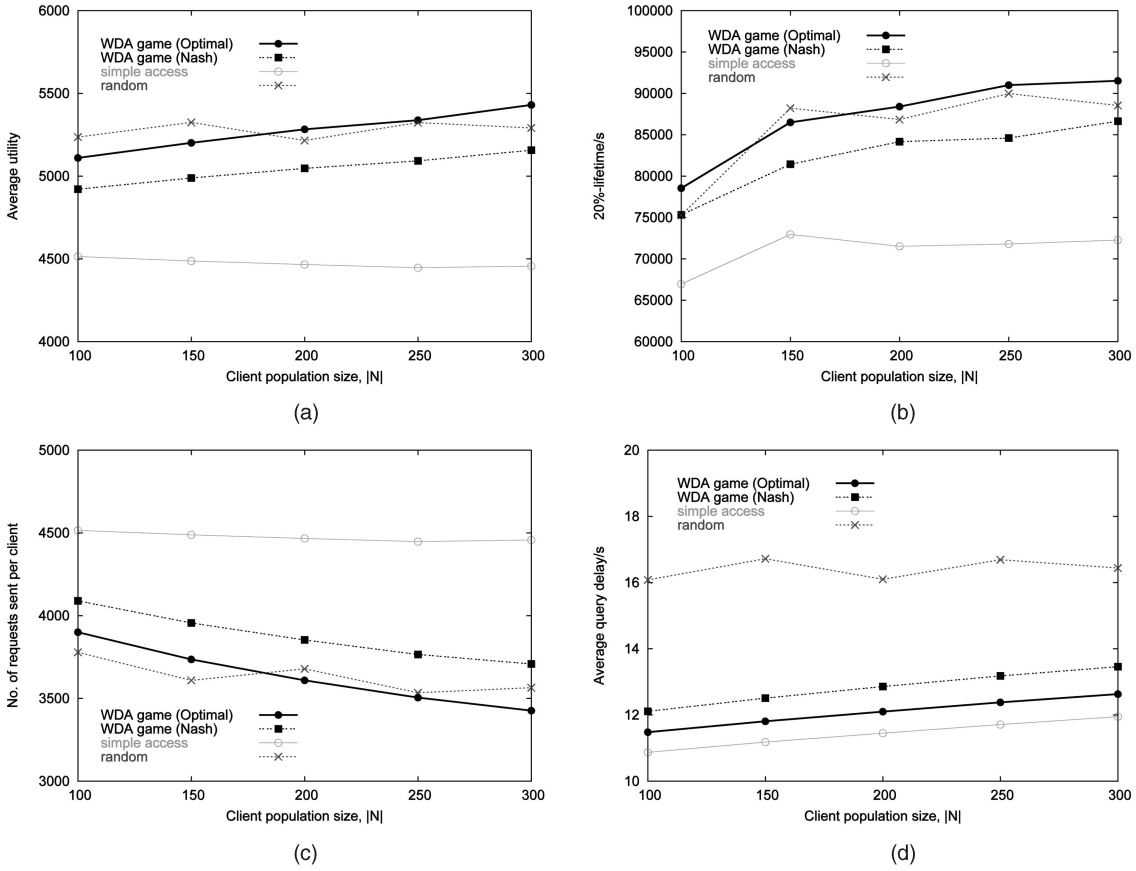


Fig. 14. (a) Average utility versus client population size. (b) Twenty percent-lifetime versus client population size. (c) Number of requests sent per client versus client population size. (d) Average query delay versus client population size. Effect of client population size, $|N|$, on various performance metrics.

The average utility per client with different number of clients is shown in Fig. 15a. In both schemes, the average utility drops with increasing query generation time. This is because the majority of energy is expended in waiting when the queries are rare. The amount of energy available for the query process is much smaller. Thus, we observe the declining trend in average utility. Another observation is that: The performance of the WDA game converges to that of a simple access scheme at high query generation time. When queries are infrequent, the values of m are small. From Algorithm 2, the equilibrium request probability approaches 1, i.e., a simple access scheme.

Fig. 15b shows the variations of 20 percent-lifetime in different query generation time. Similar to the results in Section 4.3, we observe an upward trend in both schemes. As mentioned before, most of the energy is expended in waiting at high query generation time. Since the energy cost of waiting is smaller than that of the query operations, the client's lifetime is extended. From Fig. 15a and Fig. 15b, the two metrics (average utility and 20 percent-lifetime) are conflicting with each other. Given a fixed energy source (E_{total}), if more energy is dissipated in waiting, less energy is remained for the query process. We can consider the waiting cost E_w as the cost of being "alive." However, the ultimate objective of being "alive" is to serve queries, which is used to quantify clients' utility.

Since the query generation time is directly related to the average number of requests sent per client, we observe the declining trend in Fig. 15c. Similar to the case in average utility, the performance of the WDA game converges to that of a simple access scheme. This is due to the equilibrium request probability, which approaches 1 at high query generation time. In fact, we can consider the WDA game as a general case of the simple access scheme. Each client independently determines the proportion between request and wait such that its utility is maximized.

Fig. 15d shows the variations of average query delay with query generation time. Similar to the results in Section 4.3, we observe that WDA game always gives a larger average query delay compared with a simple access scheme. Furthermore, average query delay drops with increasing query generation time. This is because fewer amount of uplink requests result in lower collision probability.

4.5 Effect of Access Skew, θ

To study the effect of access skew, the parameter, θ, is varied from 0.8 to 1.0. Fig. 16 shows the performance results in terms of average utility per client, 20 percent-lifetime, the number of requests sent per client, and average query delay.

From Fig. 16a, we can see that the average utility per client increases with θ in both simple access scheme and WDA game. As the access pattern becomes more skewed (θ

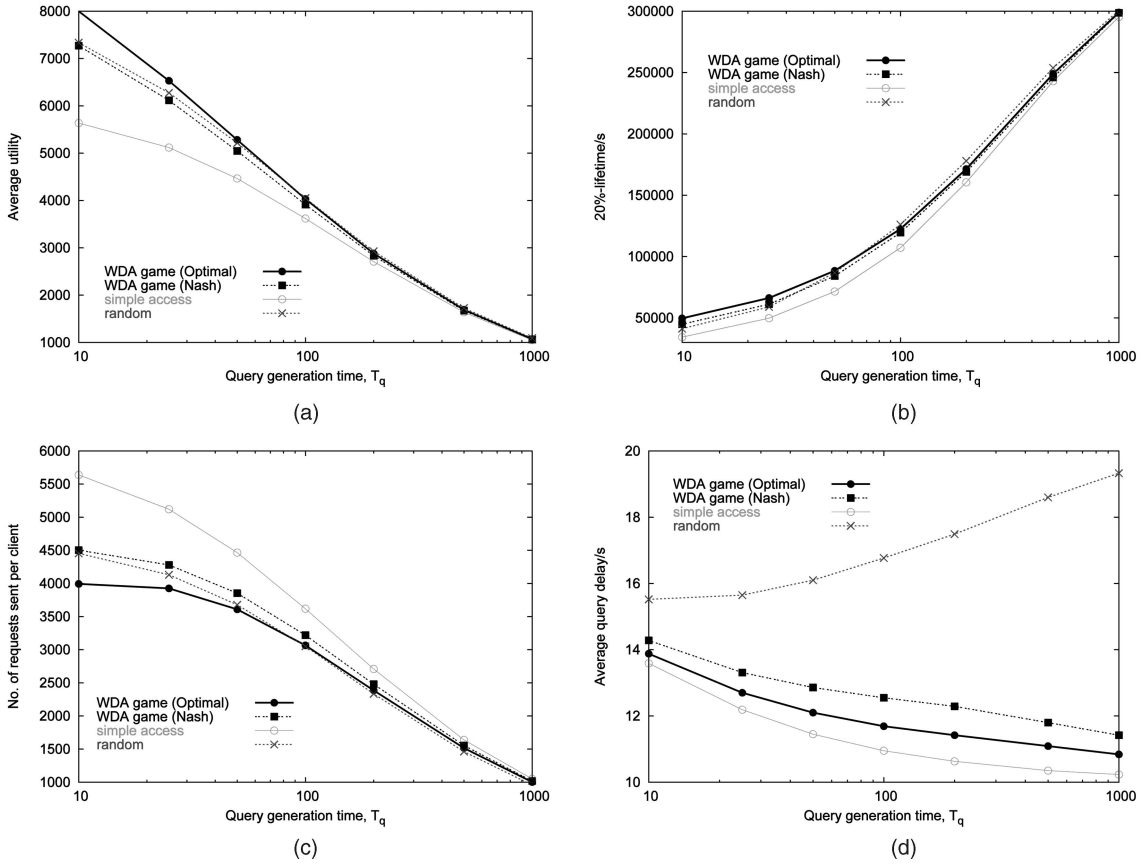


Fig. 15. (a) Average utility versus query generation time. (b) Twenty percent-lifetime versus query generation time. (c) Number of requests sent per client versus query generation time. (d) Average query delay versus query generation time. Effect of query generation time, T_q, on various performance metrics.

is large), it is more likely for a data item to be requested more than once between successive server broadcasts. This results in the general upward trend. In WDA game, the equilibrium request probability decreases when θ increases. This gives the larger improvement as shown. In terms of 20 percent-lifetime, the results are similar. If clients adopt a lower equilibrium request probability, the number of uplink requests sent would be reduced and the lifetime is lengthened. Therefore, the WDA game takes more advantage from the skewed access pattern and shows the better performance.

Fig. 16c shows the variations of the number of requests sent per client with access pattern. In the simple access scheme, each client sends a larger number of requests, which increases the utility. In the WDA game, clients send fewer requests but results in a larger improvement in utility. This is again due to the access skew advantage, as illustrated in Fig. 16b.

As shown in Fig. 16d, the average query delay decreases with θ in both simple access and WDA game. The downward trend is explained by the access skew. When θ increases, query requests are more concentrated on a small sets of data items. As such, data items broadcast by the server are more likely to be shared by more than one client. This reduces the average query delay as observed.

4.6 Effect of Channel Quality Fluctuations

We have assumed that the energy cost of sending a query request, E_s, is fixed. In this section, we would like to investigate the effect of channel fluctuations, i.e., variable E_s, on various performance metrics.

We adopted a simple two-state first-order Markov chain described in [9], [35] to model the channel fluctuations. Specifically, the channel state is either *good* or *bad*. The transition probability matrix, M_c, is given by [9], [35]:

$$M_c = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}, \tag{18}$$

where p and $1-q$ are the probabilities that the channel state is good in the current timeslot, given that the channel state in the previous timeslot was good or bad, respectively. Both p and q depend on 1) fading margin, F, 2) timeslot duration, T, and 3) Doppler bandwidth $f_D = V/\lambda$, where V is the maximum mobile speed and λ is the carrier wavelength.

Consider a network model [31], [32] with carrier frequency, $f_c = 1,800\text{MHz}$, $V = 50\text{km/hr}$, and $T = 2.5\text{ms}$, the transition probabilities are given by, $(p, q) = (0.9100, 0.1446)$ for $F = 10\text{dB}$ and $(p, q) = (0.7665, 0.3722)$ for $F = 5\text{dB}$. The transmit power is set to 0.5W and 1.0W when the channel is good or bad, respectively.

Fig. 17 and Fig. 18 shows the performance of the WDA game in different channel conditions. $F = \infty$ corresponds to

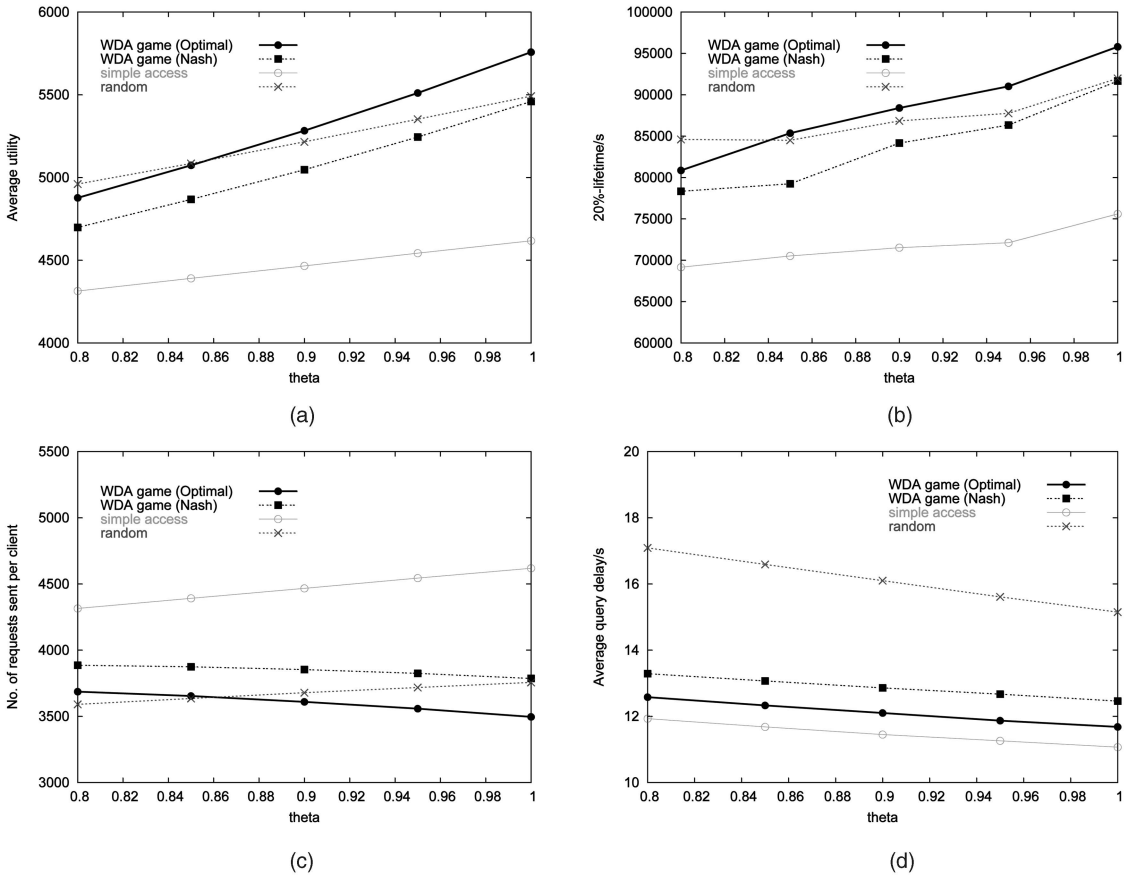


Fig. 16. (a) Average utility versus query generation time. (b) Twenty percent-lifetime versus query generation time. (c) Number of requests sent per client versus query generation time. (d) Average query delay versus query generation time. Effect of access skew, θ, on various performance metrics.

the results in previous sections, i.e., the channel is always good. On the other hand, $F = 0$ represents the worst-case scenario, i.e., the channel is always bad. From Fig. 17, it is interesting to note that the average query delay is smaller for a good channel. This is because clients dissipate more energy to send query requests when the channel is bad. Equivalently, the relative cost of waiting, α, is reduced. As such, waiting becomes a more preferable strategy because of the high energy cost incurred in sending. This results in the larger average query delay. Although more clients choose waiting, the overall utility still decreases as the channel deteriorates as shown in Fig. 18a. We also obtained similar results in terms of lifetime (see Fig. 18b). This suggests that waiting can only partially compensate for the effect of channel impairments. If a client can tolerate a larger average query delay, it is possible to wait longer. For example, a client can have an extreme strategy: *wait forever, unless the channel is good*. However, this strategy is undesirable for most practical situations due to its unbounded delay.

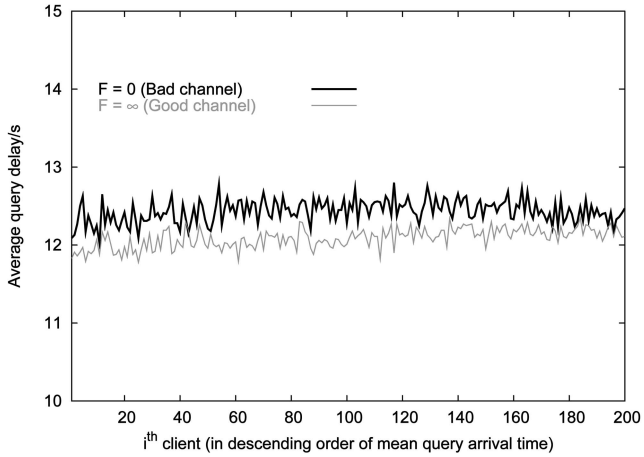


Fig. 17. Effect of channel fluctuations on the WDA game—query delay.

4.7 Effect of Initial Energy Distribution

Fig. 19 shows the effect of initial energy distribution in terms of normalized utility. There are 200 clients with $T_q = 25$s and E_{total} evenly distributed between 100J and 1,000J. The performance is measured in terms of normalized utility, which is defined as utility per Joule of energy. From the figure, it is observed that clients with a smaller E_{total} achieved a slightly larger normalized utility. This suggests that the WDA game is more efficient for low-energy clients, which is a desirable property. Similar results were observed in terms of lifetime (not shown).

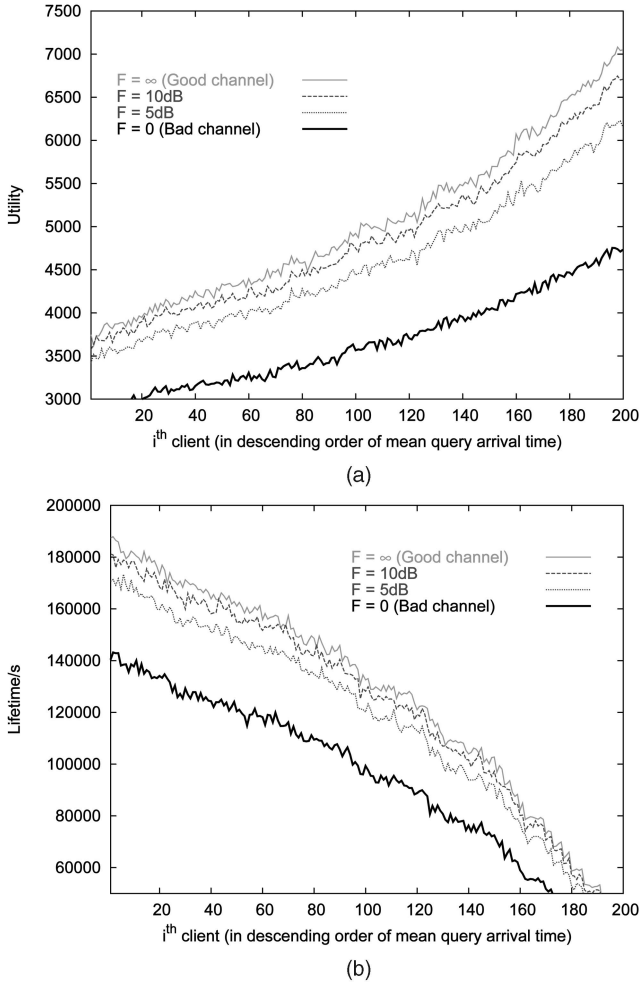


Fig. 18. (a) Utility distribution across all clients. (b) Lifetime distribution across all clients. Effect of channel fluctuations on the WDA game—utility and lifetime.

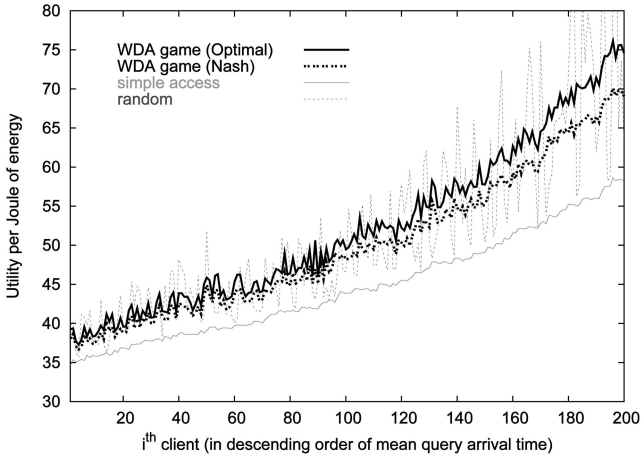


Fig. 19. Effect of initial energy, E_{total}.

5 Comparison with Caching-Based Approaches

In this section, we compare the performance of with-cache schemes (IR and IR+UIR) and without-cache schemes (simple access and WDA game) in terms of the same performance metrics—*utility*, *lifetime*, *number of requests sent* and *average query delay*. The same set of simulation parameters are used as in Table 2. To better visualize the effect of query pattern, the maximum query generation time ($T_q^{\max}$) is changed from $50s$ to $100s$. For with-cache schemes, updates to each data item follow an exponential distribution with mean arrival time of $5,000s$. Each client has a cache size of $1,000KB$. We assume that the server performs the actual-to-optimal mapping described in Section 3.4.

5.1 Results

Figs. 20a, 20b, 20c, and 20d show the comparison results in terms of utility, lifetime, uplink traffic (number of requests sent) and average query delay, respectively.

Utility: Clients with high query rates (those with large index) perform much better than the without-cache counterparts in with-cache schemes (IR and IR+UIR). On the other hand, clients with low query rates are more advantageous to adopt a without-cache scheme. Based on this observation, we can conclude that client caching and the use of invalidation reports (IRs and UIRs) are more suitable for clients with frequent queries. In fact, utility improves significantly with increasing query rates. Between the two with-cache schemes, clients in IR generally achieve a larger utility than those in IR+UIR. Another observation is that the change occurs at about the 160th client, i.e., $T_q = 25s$, which is fairly demanding given the mean update time of $5,000s$. In our simulation settings, more than half of the clients achieve a higher utility value (number of queries completed) in a without-cache scheme.

Lifetime: Given the same query pattern, longer lifetime results in larger utility values. As such, we observe a similar pattern as in utility: High-query-rate (low-query-rate) clients generally have a longer lifetime in with-cache (without-cache) schemes. Therefore, a client's query rate determines its appropriateness of adopting a with-cache scheme such as IR or IR+UIR. In without-cache schemes, lifetime decreases gradually from $250,000s$ with increasing rate of query. However, clients in with-cache schemes generally have similar lifetime between $100,000s$ and $150,000s$. This is because *every* client is required to 1) download invalidation reports (IRs and UIRs) and 2) replace invalidated caches with updated ones. Both factors consume considerable proportions of energy (see Section 5.2). Low-query-rate clients expend most of their energy in cache consistency process, leaving little for actual query operations. In IR+UIR, *every* client is required to download UIRs in addition to IRs. Therefore, the clients generally have a shorter lifetime in IR+UIR than those in IR.

Uplink traffic: We observe that with-cache schemes *always* result in much lower amount of uplink traffic than without-cache schemes do. This is due to the aggregate effect in with-cache schemes: Each client has to wait for the next invalidation report (IR or UIR) before sending a request. By then, there may have more than one query pending at the client. As such, a single request can inform the server of multiple desired data items. For the simple access scheme, each client sends a request upon every new query. This results in the largest number of requests. WDA game exploits the possibility of duplicated requests, which reduces the amount of requests sent.

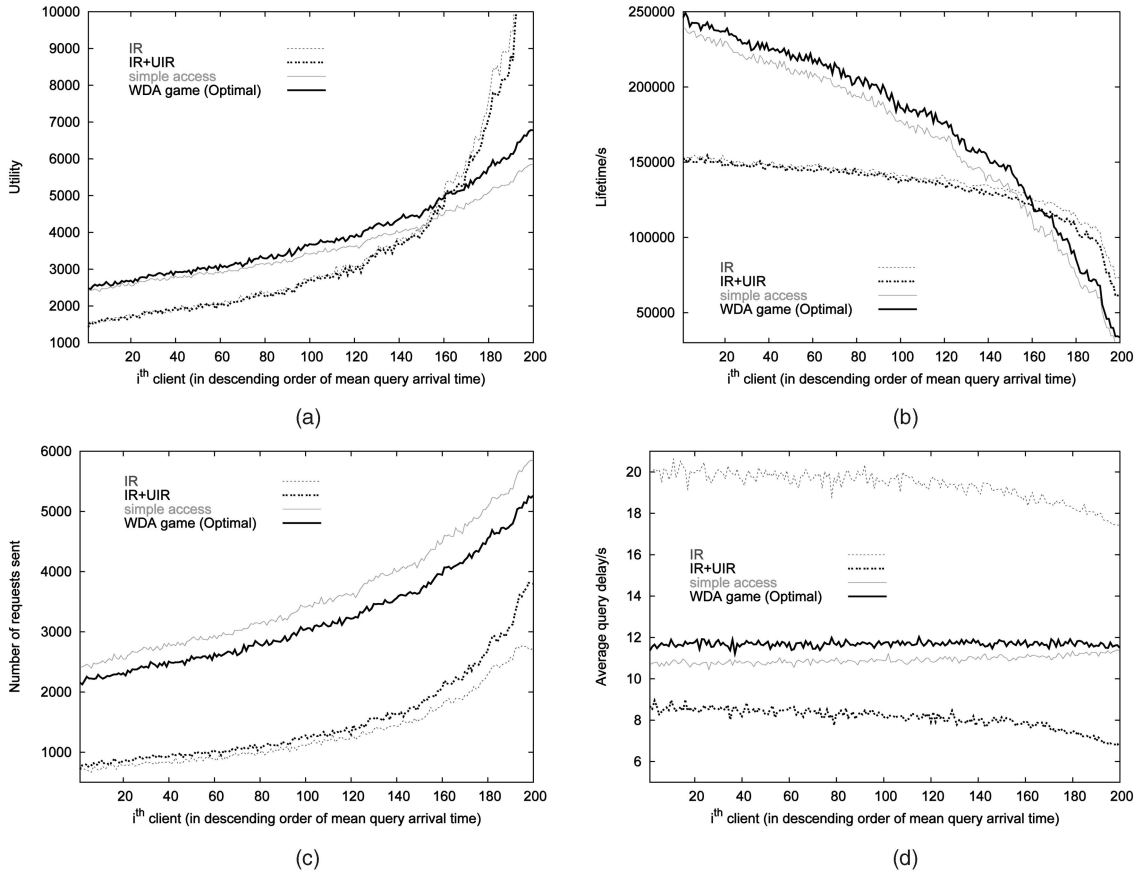


Fig. 20. (a) Utility. (b) Lifetime. (c) Number of requests sent. (d) Average query delay. Performance comparison.

Average query delay: Clients experience a much smaller average query delay in IR+UIR than those in IR. This is consistent to the primary aim of IR+UIR—to reduce the large query delay in IR. It is also observed that high-query-rate clients have shorter query delay for both IR and IR+UIR. This is due to the increase in the number of cache hits. For without-cache schemes, each client experiences more or less the same average query delay, which is about $10s$, half of the server broadcast time, i.e., $\frac{L}{2}$.

5.2 Energy Consumption Characteristics

Section 5.1 compares the performance of with-cache schemes and without-cache schemes in terms of different performance metrics. In this section, we take a closer look at the difference between the two approaches. Specifically, we look at the energy characteristics. In a without-cache scheme (simple access and WDA game), a client consumes its energy in three different ways:

1. sending query requests (Uplink),
2. receiving replies from server, and
3. waiting (energy cost of being "alive").

For a with-cache scheme (IR and IR+UIR), a client is also required to

1. receive IRs,
2. receive UIRs (IR+UIR only),
3. replace invaliidated caches with the update copies (cache maintenance).

Fig. 21 shows the average amount of energy consumed in different ways among the 4 schemes. The energy expended in waiting represents the energy cost of being "alive," which is proportional to the lifetime. Since clients have the longest lifetime in WDA game, the percentage of energy consumed in waiting is the largest. In the with-cache schemes, receiving invalidation reports (IRs and UIRs) takes up only about 5 percent of the available energy, which is insignificant. Client caching reduces both the uplink and downlink cost via cache hits. However, the energy cost of keeping the cache up-to-date, is considerable. As observed in Fig. 20a and Fig. 20b, these two types of extra energy cost outweigh the benefit from cache hits for most clients, except those with high query rates.

6 Discussion

We have proposed a power aware wireless data access scheme based on a game-theoretic formulation. Each client in the WDA game independently determines the request probability (strategy), in the hope of reducing its own uplink power consumption. From the simulation results presented in Section 4, we observe that every client the WDA game is capable of increasing its utility when compared with the simple always request scheme. These performance improvements are rested from the observation: Clients may send duplicated query requests to the

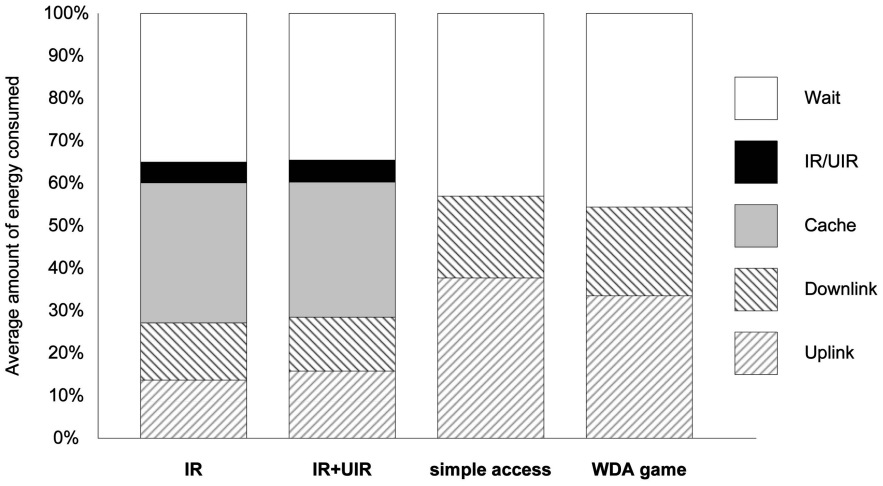


Fig. 21. Comparison—energy consumption characteristics.

server. The higher the chance of duplicates, the larger would be the improvements. This depends on a number of system parameters:

1. database size $|D|$,
2. client size $|N|$,
3. query rates $|T_q|$,
4. access skew θ, etc.

With the number of data items ever growing, it would become less likely to have duplicated requests. As such, each client would play with a request probability of one. Thus, the proposed scheme is not applicable to such applications with a huge number of data items, e.g., Web browsing from mobile devices. However, we do envision some application scenarios that our scheme would be useful. For instance, a number of subscribers (clients) query a set of data items such as stock quotes, traffic schedules, location-dependent information, etc. Another example is that a set of clients query the server for routing information of the network. In such a scenario, each data item represents a specific route between a pair of source and destination nodes. Furthermore, our scheme could be used to disseminate information about individual client, e.g., clients' reputation values [8].

7 Concluding Remarks

We have studied the following wireless data access scenario: a number of clients send requests to a server to query for a set of data items; the server replies with the requested content via the common broadcast channel. Specifically, we consider two classes of data access schemes: *with-cache schemes* use client caching while *without-cache schemes* do not.

In without-cache schemes, there is no need to maintain cache consistency. However, clients have to send an uplink query request for each new query. Due to data locality, it is observed that the server may receive more than one request for some popular data items. However, these duplicated requests are a waste of 1) battery energy and 2) uplink bandwidth. In fact, these requesting clients, but not all, are more advantageous to "back-off" and let some other to send the request on behalf of themselves. Ideally, only one request is enough to trigger the server to broadcast a desired data item. This would require *explicit* coordination among clients. Such extra communication overheads may not justify the energy conserved from uplink requests. More importantly, the client, who actually sends the request, expends its own battery energy and possibly bears monetary cost for the goodness of others. Without appropriate incentives, there is obviously *no* client willing to take up the requesting role.

To analyze the above conflicting situation, we model the without-cache data access problem as a static noncooperative game—*wireless data access (WDA) game*—each *player* (client) maximizes its own *utility* with *no* explicit communication with one another. The utility function (2) takes power consumption in transmit, receive, and idle modes into consideration. Our theoretical analysis shows that clients are *not always* necessary to send requests upon arrival of new queries. Instead, each client determines its equilibrium request probability without *any explicit* communication with one another. We have formalized and evaluated the algorithms for the server and clients in the WDA game. Simulation results confirm that our proposed power aware wireless data access scheme, comparing with a simple always-request one, increases the utility and lifetime of *every* client while reducing the number of requests sent, with a cost of slightly larger average query delay.

We also compare the performance of *with-cache* and *without-cache* schemes. Our results suggest that client caching does *not* always result in performance improvement. In particular, we observe that *every client* dissipates a significant amount of energy to maintain its client up-to-date via prefetching. For high-query-rate clients, the energy saved from cache-hits outweighs the energy cost of prefetching. However, the majority of low-query-rate clients suffer from shortened lifetime due to the increased energy consumption rate.

TABLE 3
Definitions of Notations

Notations	Definitions
$D; \|D\|$	A set of data items; total number of data items
$N; \|N\|$	A set of clients; client population size
$M \subseteq N$	A set of clients interested in a common data item
m	$m = \|M\|$
L	Server broadcast period
E_{total}	Energy available to a client
E_{UL}	Uplink energy required to complete a query
E_{DL}	Downlink energy required to complete a query
E_Q	Total energy required to complete a query
E_s	Energy required to send a request to the server
E_w	Energy required to wait for a broadcast period
α	Relative cost of waiting to sending ($E_w = \alpha E_s$)
s_i	Client i's strategy, i.e., request probability
S_i	Client i's strategy space ($s_i \in S_i$)
s_{-i}	Strategy combination of all clients, except i
s	Strategy combination of all clients
S	Strategy combination space
s_i^*	Equilibrium strategy of client i
s_{-i}^*	Equilibrium strategy combination of all clients, except i
s^*	Equilibrium strategy combination of all clients
U_i	Client i's utility (Equation (2))
T_q^i	Client i's mean query generation time
$p_j(\theta)$	Access probability of data item j

APPENDIX

A list of notations used throughput the paper and their definitions are summarized in Table 3.

ACKNOWLEDGMENTS

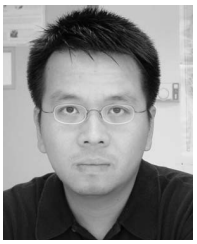
The authors would like to thank the anonymous reviewers for the insightful and constructive comments on our manuscript. Thanks are also due to Professor Sajal Das for his professional and timely handling of our submission. This research was supported by a grant from the Research Grants Council of the HKSAR under project number HKU 7157/04E.

REFERENCES

[1] D. Barbará and T. Imieliński, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *Proc. 1994 SIGMOD Int'l Conf. Management of Data,* pp. 1-12 May 1994
[2] Bluetooth Special Interest Group, "Bluetooth Specification," http://www.bluetooth.org, 2006.
[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. 18th Int'l Conf. Computer Comm.,* vol. 1, pp. 126-134, Mar. 1999.
[4] J. Cai and K.-L. Tan, "Energy-Efficient Selective Cache Invalidation," *Wireless Networks,* vol. 5, no. 6, pp. 489-502, Dec. 1999.
[5] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *Proc. Sixth Ann. Int'l Conf. Mobile Computing and Networking,* pp. 200-209, Aug. 2000.
[6] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 608-621, June 2002.
[7] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 5, pp. 1251-1265, Sept.-Oct. 2003.
[8] K. Chen and K. Nahrstedt, "iPass: an Incentive Compatible Auction Scheme to Enable Packet Forwarding Service in MANET," *Proc. 24th Int'l Conf. Distributed Computing Systems,* pp. 534-542, Mar. 2004.
[9] A. Chockalingam, M. Zorzi, L.B. Milstein, and P. Venkataram, "Performance of a Wireless Access Protocol on Correlated Rayleigh-Fading Channels with Capture," *IEEE Trans. Comm.,* vol. 46, no. 5, pp. 644-655, May 1998.
[10] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C.H. Papadimitriou, and J. Kubiatowicz, "Selfish Caching in Distributed Systems: A Game-Theoretic Analysis," *Proc. 23rd ACM Symp. Principles of Distributed Computing,* pp. 21-30, July 2004.
[11] A. Datta, D.E. VanderMeer, A. Celik, and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users," *ACM Trans. Database Systems,* vol. 24, no. 1, pp. 1-79, Mar. 1999.
[12] Z. Fang and B. Bensaou, "Fair Bandwidth Sharing Algorithms Based on Game Theory Frameworks for Wireless Ad-Hoc Networks," *Proc. 23rd Int'l Conf. Computer Comm.,* vol. 2, pp. 1284-1295, Mar. 2004.
[13] L.M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *Proc. 20th Int'l Conf. Computer Comm.,* vol. 3, pp. 1548-1557, Apr. 2001.
[14] D. Fudenberg and J. Tirole, *Game Theory.* MIT Press, Aug. 1991.
[15] V.K. Garg, *Wireless Network Evolution: 2G to 3G.* Prentice Hall, Aug. 2001.
[16] M.S. Gast, *802.11 Wireless Networks: The Definitive Guide.* O'Reilly and Associates, Apr. 2002.
[17] M. Goemans, L.E. Li, V.S. Mirrokni, and M. Thottan, "Market Sharing Games Applied to Content Distribution in Ad-Hoc Networks," *Proc. Fifth ACM Int'l Symp. Mobile Ad Hoc Networking and Computing,* pp. 55-66, May 2004.
[18] Q. Hu and D.L. Lee, "Adaptive Cache Invalidation Methods in Mobile Environments," *Proc. Sixth IEEE Int'l Symp. High Performance Distributed Computing,* pp. 264-273, Aug. 1997.
[19] J. Jing, A. Elmagarmid, A.S. Helal, and R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," *Mobile Networks and Applications,* vol. 2, no. 2, pp. 115-127, Oct. 1997.
[20] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and A. Manjeshwar, "Energy-Efficient Link Assessment in Wireless Sensor Networks," *Proc. 23rd Int'l Conf. Computer Comm.,* vol. 3, pp. 1751-1761, Mar. 2004.
[21] V.K. N. Lau, "Performance Analysis of Variable Rate: Symbol-By-Symbol Adaptive Bit Interleaved Coded Modulation for Rayleigh Fading Channels," *IEEE Trans. Vehicular Technology,* vol. 51, no. 3, pp. 537-550 May 2002.
[22] S. Lim, W.-C. Lee, G. Cao, and C.R. Das, "A Novel Caching Scheme for Internet Based Mobile Ad Hoc Networks," *Proc. 12th Int'l Conf. Computer Comm. and Networks,* pp. 38-43, Oct. 2003.
[23] M.J. Osborne and A. Rubinste, *A Course in Game Theory.* MIT Press, July 1994.
[24] J.D. Parsons, *The Mobile Radio Propagation Channel,* second ed. John Wiley and Sons, Nov. 2000.
[25] H. Schulzrinne, X. Wu, S. Sidiroglou, and S. Berger, "Ubiquitous Computing in Home Networks," *IEEE Comm. Magazine,* vol. 41, no. 11, pp. 128-135, Nov. 2003.
[26] V. Shah, N.B. Mandayam, and D.J. Goodman, "Power Control for Wireless Data Based on Utility and Pricing," *Proc. Ninth Int'l Symp. Personal, Indoor and Mobile Radio Comm.,* vol. 3, pp. 1427-1432, Sept. 1998.
[27] V. Stanford, "Pervasive Computing Puts Food on the Table," *IEEE Pervasive Computing,* vol. 2, no. 1, pp. 9-14, Jan.-Mar. 2003.
[28] K.-L. Tan, "Organization of Invalidation Reports for Energy-Efficient Cache Invalidation in Mobile Environments," *Mobile Networks and Applications,* vol. 6, no. 3, pp. 279-290, June 2001.
[29] K.-L. Tian, J. Cai, and B.C. Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 8, pp. 789-807, Aug. 2001.
[30] K.-L. Wu, P.S. Yu, and M.-S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing," *Proc. 12th Int'l Conf. Data Eng.,* pp. 336-343, Feb.-Mar. 1996.
[31] M.K.H. Yeung and Y.-K. Kwok, "Wireless Cache Invalidation Schemes with Link Adaptation and Downlink Traffic," *IEEE Trans. Mobile Computing,* vol. 4, no. 1, pp. 68-83, Jan.-Feb. 2005.
[32] M.K.H. Yeung and Y.-K. Kwok, "New Invalidation Algorithms for Wireless Data Caching with Downlink Traffic and Link Adaptation," *Proc. 18th Int'l Parallel and Distributed Processing Symp.,* pp. 222-229, Apr. 2004.

[33] L. Yin and G. Cao, "Adaptive Power-Aware Prefetch in Wireless Networks," *IEEE Trans. Wireless Comm.,* vol. 3, no. 5, pp. 1648-1658, Sept. 2004.
[34] B. Zheng, J. Xu, and D.L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," *IEEE Trans. Computers,* vol. 51, no. 10, pp. 1141-1153, Oct. 2002.
[35] M. Zorzi, R.R. Rao, and L.B. Milstein, "On the Accuracy of a First-Order Markov Model for Data Transmission on Fading Channels," *Proc. IEEE Int'l Conf. Universal Personal Comm.,* pp. 211-215, Nov. 1995.

Mark Kai Ho Yeung received the BEng degree (first class honors) in electrical and electronic engineering, and the MPhil degree both from the University of Hong Kong in 2002 and 2004, respectively. He is a PhD candidate at the University of Hong Kong. His research interests include cache invalidation, data access and cooperative caching in wireless networks with the emphasis of energy efficient and incentive compatiable operations. He is also interested in network-wide traffic analysis. He is a student member of the IEEE.



Yu-Kwong Kwok received the BSc degree in computer engineering from the University of Hong Kong in 1991, the MPhil and PhD degrees in computer science from the Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively. He is an associate professor in the Department of Electrical and Electronic Engineering at the University of Hong Kong (HKU). Before joining the HKU in August 1998, he was a visiting scholar for one year in the parallel processing laboratory at the School of Electrical and Computer Engineering at Purdue University. He recently served as a visiting associate professor at the Department of Electrical Engineering-Systems at University of Southern California from August 2004 to July 2005, on his sabbatical leave from HKU. His research interests include distributed computing systems, wireless networking, and mobile computing. He is a senior member of the IEEE. He is also a member of the ACM, the IEEE Computer Society, and the IEEE Communications Society. He received the Outstanding Young Researcher Award from HKU in November 2004.