A Wireless Stateful Power-Aware Pre-fetch Scheme

A thesis presented
by

Ian Scatliff

to
The Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of

Computer Science

The University Of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2
June 2004

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Wireless Stateful Power-Aware Pre-fetch Scheme

BY

Ian Scatliff

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

A Wireless Stateful Power-Aware Pre-fetch Scheme

Abstract

Wireless computing allows users to access data while allowing them to move about, free of a physical connection to the data they are accessing. The increased mobility of the user comes at a performance cost, because data must now be transferred over relatively slower and less reliable, wireless, networks. This research investigates a pre-fetching data caching solution that stores mobile unit metadata (information about data) at the base station in order to optimize what is pre-fetched from the mobile unit. Another issue with wireless networking is that transmission from a mobile unit is significantly more costly, in terms of energy utilization, than message reception. This proposed technique will consider the effects of power consumption, and try to maximize the battery life of the mobile unit. The techniques proposed here are largely an extension of the *value-based adaptive pre-fetch* scheme, developed by Yin, Cao, Das and Ashraf [Yin et al., 2002], using ideas developed in the *asynchronous and stateful* scheme proposed by Kahol, Khurana, Gupta and Srimani [Kahol et al., 2000].

Table of Contents

Acknowledgements

Thank you to Dr. Peter Graham for his input, support and review of my work. Thank you to Dr. Neil Arnason for his help in determining the proper statistical measures needed to analyze the results of this thesis. Thank you to Dr. John Anderson for his encouragement and support over the course of my masters degree. Thank you to Protegra Technology Group (www.protegra.com) for funding part of my degree and for granting me use of their computer lab for the purpose of running the simulations in this thesis.

Most of all thank you to Chantal and Noah, for the support and encouragement you have given me over the last five years.

Dedication

This thesis is dedicated to my beautiful bride Chantal and to my son "Noah-Balboa".

1. Introduction

Wireless computing allows users to access data while allowing them to move about, free of a physical connection to the data they are accessing. Mobile units such as laptops with wireless connections and wireless handhelds communicate with "base stations" (or "access points") via wireless protocols. The base station provides direct connectivity with a higher-speed networking infrastructure that supplies access to user data. As the user roams out of the range of its current base station, a hand-off will occur between the user's current base station and the base station providing support in the geographical area that the user has just entered. This hand-off is transparent to the user, who will have access to their data throughout the transition.

The benefits of wireless computing come at a price. The wireless connection between the mobile unit and the base station is low-bandwidth, prone to a high error-rate and may not always be available (leading to temporary disconnection). In addition, the mobile unit itself is typically a smaller portable device with limited memory and computing resources. To support user roaming, the mobile unit's power supply is most often battery powered. Sending transmissions from the mobile unit to a base station consumes a relatively large amount of energy.

1.1. Thesis Objective

The objective of this thesis is to investigate ways in which the use of the low-bandwidth connection between the mobile unit and the base station can be maximized while minimizing energy expenditure. This objective will be achieved by combining two recently developed wireless communication optimization techniques described in [Yin et

al., 2002] and [Kahol et al., 2000] to further optimize bandwidth usage and the mobile unit's energy expenditure.

Specifically the goals of this thesis are as follows:

- Reproduce the simulation of [Yin et al., 2002] to provide a baseline for the improvements offered by this thesis.
- Leverage ideas provided in [Kahol et al., 2000] to improve the results of [Yin et al., 2002] by reducing the energy expenditure of the mobile unit, while providing the same service level.

1.2. Organization

The organization of this thesis is as follows:

- **Section 1 Introduction** presents an introduction to this thesis and the thesis objective.
- **Section 2 The Problem and Domain** describes the domain and the problems associated with the wireless computing environment.
- **Section 3 Related Work** provides related research in this domain that pertains to the work of this thesis.
- **Section 4 Solution Strategy** describes the simulation model and processing architecture.
- **Section 5 Simulation Evaluation** evaluates the work done in this thesis and directly compares it to the work presented by [Yin et al., 2002].

- **Section 6 Conclusion** provides a summary of this thesis and the findings of the simulation.
- **Section 7 References** provides a bibliography of the work referenced within this thesis.

1.3. Terminology

In the research related to this thesis, the terms "base station" and "server" are typically used interchangeably. This thesis will distinguish between these two terms. The term "base station" will refer to the wireless networking infrastructure and hardware that must exist to support this wireless domain. The base station is able to broadcast information to mobile units that are within its geographical area. The term "server" will be used to denote the strategies and schemes (either in software or hardware) that reside at the base station and provide service to mobile units. The server will determine what and when to send information to the mobile units via that base station's infrastructure. In addition to these two terms, a third term called "database server" will be used to describe the entity that serves data to the mobile units. The database server can reside on any machine connected to the base station via a high-speed land-line. For simplicity, this thesis will assume that the database server resides at the base station.

2. The Problem and Domain

This thesis will concern itself with certain key issues within wireless computing environments. This section will describe these key issues and discuss their impact on mobile computing.

There are many low-level networking issues that affect wireless computing, however this proposal concerns itself with only one of these issues, because the others do not directly affect it. The issue is that communication between the mobile unit and base station is over a wireless low-bandwidth connection, as opposed to a wired high-bandwidth connection. Traditional data caching and pre-fetching schemes used for high-bandwidth connections must be altered to support this low-bandwidth link. The necessary changes are required mainly due to the increased cost of a cache miss. When a cache miss occurs, the modified cache item must be downloaded over the low-bandwidth link, greatly increasing the latency of data access. Therefore it is imperative that the cache remain as current as possible. Amplifying the cache consistency problem is the fact that pre-fetching is also more expensive. The cache items must be pre-fetched over the low-bandwidth link, utilizing precious network resources as well as power. Mobile units must therefore carefully determine what to pre-fetch. Another interesting characteristic of the wireless connection is that the bandwidth from the base station to the mobile unit is typically an order of magnitude larger than the bandwidth from the mobile unit to the base station.

One main technical issue that this thesis will focus on is the fact that transmission from a mobile unit is significantly more costly than reception. Cost is measured by the

amount of energy required to transmit or receive information. Given that mobile units typically operate on battery power, there is a fixed amount of energy available to the mobile unit during a working session. As [Forman and Zahorjan, 1994] suggest, there may be gains to be had by having the base station automatically transmit information (the information might be metadata or the data itself) to the mobile units in its geographical reception area without waiting for mobile units to request it. Information may be broadcast to all mobile units in the area, or directed to one particular mobile unit, depending on the mobile unit's data access characteristics. These characteristics, which include, among other things, available or remaining power on the mobile units should be made known to the base station to allow it to optimize communication.

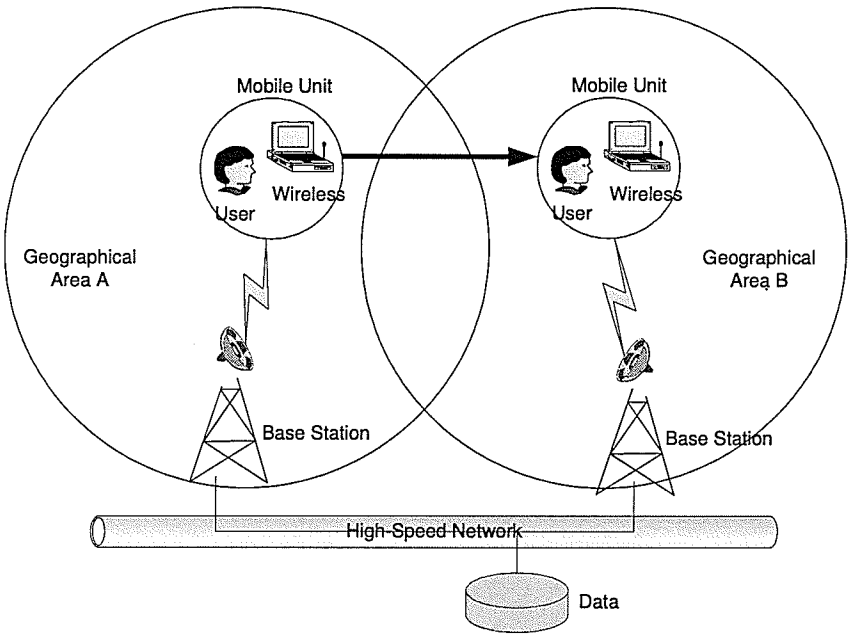


Figure 1 – Wireless computing environment

As Figure 1 shows, mobile units communicate with base stations to gain access to their data. A simple domain example would be a taxi cab equipped with a wireless console capable of communicating with the central dispatch office database. In this

scenario the taxi cab is constantly requesting available pickups in the vicinity. Instead of waiting for the driver to request this information from the central dispatch office, this information could be pre-fetched onto the vehicle and be ready for immediate use by the driver. As the vehicle moves into another geographical area, the vehicle may become supported by another base station assigned to that area. However, both the previous and current base station will have access to the central database server being housed at the dispatch office, and therefore will still be accessible by the mobile vehicle.

3. Related Work

Over the last decade, a significant amount of research has been done in latency minimization for network queries and mobile unit power optimization for wireless communications. This research is mainly centered on optimizing the low-bandwidth connection between the mobile unit and the base station, while maximizing the life of the mobile unit's battery.

[Ebling et al., 1994] proposed that to overcome the network bottleneck in mobile computing, designers must sacrifice CPU and storage resources to implement smart caching and network communication scheduling to reduce network cost. Mobile computing requires that the network no longer be treated as an inexhaustible resource. They concluded that the network needs to be treated as a first-class resource and that other computing resources such as the CPU and storage should be used to compensate for low-bandwidth connections. However, they noted, this usage must be adaptive and should aggressively exploit cheap network resources when they are available.

3.1. The Invalidation Report

In 1995, Barbará and Imielinski published a paper [Barbará and Imielinski, 1995] that pioneered research in the area of mobile computing database caching strategies and supported the ideas presented by [Ebling et al., 1994]. They proposed the periodic broadcast of information from the server to all mobile units currently subscribed to it for the purpose of keeping the mobile units' caches consistent. The broadcast, called an *invalidation report* (IR), contains information about all modified data items since the last IR. The mobile unit then uses the information in the IR to invalidate stale items in its

cache, and to determine what information it needs to download from the server to satisfy client queries that have been issued since the previous IR.

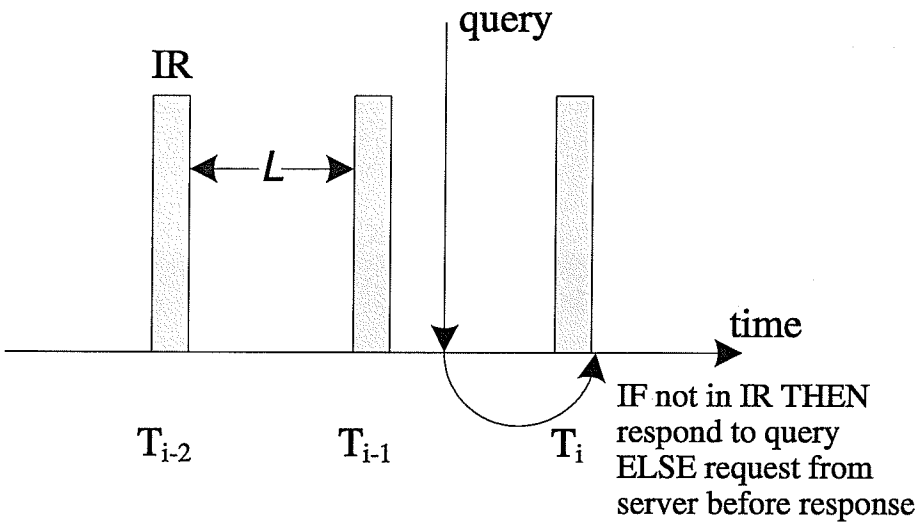


Figure 2 – The Invalidation Report

As Figure 2 shows, if a user initiates a query between T_{i-1} and T_i the mobile unit will wait until the next IR is read at time T_i to determine if the pending query can be answered immediately because the cache is then consistent. If the cache is inconsistent then the modified data item must be retrieved from the server before the query can be satisfied. This increases the latency of the query, because of the time spent requesting and downloading the data item from the server.

The information sent in the IR could be either state-based or history-based. A state-based IR contains information about values of the modified items. This might take the form of checksums, bit sequences indicating modified/not-modified, or the passing of the actual data values themselves. In the latter case this eliminates the need to fetch the modified data item from the server.

A history-based IR identifies all data items that have been modified at the server, and for each modified data item supplies the timestamp of its last update. The system parameter w is used to configure the IR to contain all update information in the last w IR intervals. If a client is disconnected for less than w time it can invalidate its cache given the information in the IR without having to discard its entire cache. In the case of the history-based IR, the mobile unit(s) must explicitly fetch the modified data they require. These two methods are roughly analogous to the use of update and invalidate protocols in conventional caching.

[Barbará and Imielinski, 1995] proposed three different IR formats: Broadcasting Timestamps (TS), Amnesic Terminals (AT), and Signatures (SIG). The TS IR format is equivalent to the history-based algorithm already discussed. The AT IR format is very similar to the TS strategy except that w is set to L, where L is the time between IRs. The SIG algorithm uses checksums over pages of data that are compared with cached data to determine the need for invalidation. Since mobile units do not have all the database information required to compute the checksum for comparison, the server periodically broadcasts the computed checksums of the entire database to the clients. The mobile unit and server agree beforehand on the appropriate granularity of the checksums. Using this technique the parameter w does not apply so there is no disconnection time constraint on the mobile unit. Unlike TS and AT, SIG is a compressed report, as checksums represent pages of data instead of individual data items.

The TS invalidation report contains a timestamp T_i indicating the time the IR was created. The invalidation report also contains a set of tuples (d_x, t_x), where d_x is the identifier of the data item and t_x is the last update timestamp of data item d_x. The set of

tuples contained in the report is restricted such that $t_x > (T_i - w * L)$, where w, as mentioned earlier, is a system parameter that configures the report to contain all update information in the last w IR intervals and L is the time interval between invalidation reports.

The simulations that [Barbará and Imielinski, 1995] performed involved two types of mobile units called "sleepers" and "workaholics". As their names imply, "sleepers" are mobile units that tend to be disconnected often for some length of time, whereas "workaholics" are mobile units that never disconnect from the base station and are constantly keeping the caches valid via the IRs. For "workaholics" the AT format provides the best performance. This is mainly due to the fact that the IRs are smallest using this strategy and no history information is required since the mobile units are constantly connected and receiving IRs. When the model uses "sleepers" the TS and SIG strategies generally outperform AT, and TS outperforms AT when the database update rate is a small. However, for very large values of sleep time a no-caching strategy works best.

[Jing et al., 1997] expands on the work done by [Barbará and Imielinski, 1995] and offers another compressed state-based IR strategy called Bit Sequences (BS). The BS IR represents all database items using a sequence of bits. The n^{th} bit represents the n^{th} item in an indexed database, where the index sequence is agreed to beforehand by the mobile unit and the server. In addition to the bit sequence, one timestamp is associated with the entire sequence of bits and corresponds to the time interval that these bits represent. A "1" bit indicates that a value has been updated at the database and a "0" bit indicates that

the data item has not been updated. Mobile units that disconnect after the time indicated by the timestamp can use the bit sequence to invalidate their caches.

In general, mobile units will be disconnected for varying amounts of time. As a result, a mobile unit that has disconnected for five minutes will have to invalidate a larger than necessary portion of its cache given a bit sequence with a timestamp of 5 hours prior. To handle mobile units with varying disconnection times, the BS strategy employs an ingenious hierarchy of bit sequences that allows the mobile unit to determine with precision which data items have been updated since its disconnection while minimizing the invalidation of valid cached items.

The BS algorithm assumes that there are N data items in the database such that $N = 2^n$ for some integer n. Let there exist $\log_2(N)$ bit sequences in the invalidation report, where the sequence Bn contains N bits, and the Bn-1 sequence contain half as many bits, and the Bn-2 sequence contain half as many bits and Bn-1, etc... until the last sequence contains 2 bits. The client determines which sequence to use by choosing the sequence with a timestamp that is less and nearest to the time that the client disconnected. If the disconnection time of the client is less than the timestamp of the oldest sequence Bn, then the report cannot be used. When constructing the report, the server will only allow N/2 updates to be reported per sequence. This does not mean that some updates will not be reported, rather that the timestamps of the bit sequences are dependent on the frequency of updates that occur at the server.

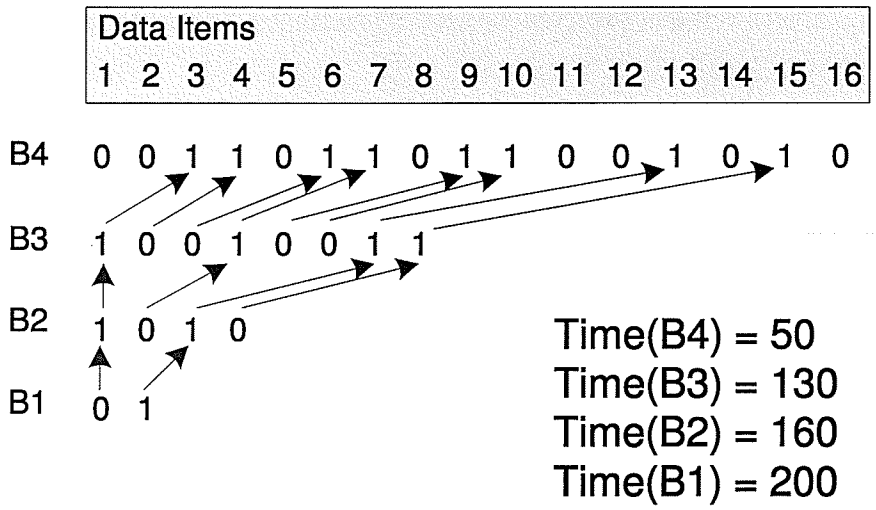


Figure 3 – Example BS report at time 200

An example report in the BS scheme is shown by Figure 3. If the client disconnected at time 170, then the B2 sequence will be used to invalidate the mobile unit's cache. The client will invalidate the 2 data items denoted by the two "1" bits in the B2 sequence. To locate the first "1" bit in B2, find the first "1" in bit in B3. From there find the first "1" bit in B4. From here the mobile unit will know that data item 3 has been updated. The second "1" bit in B2 is found by locating the third "1" bit in B3 and then the 7^{th} "1" bit in B4, which indicates that data item 13 has been updated.

The BS scheme is able to reduce the size of the IR to half of that required by the TS strategy and much smaller than the SIG IR, while retaining the same reporting value for clients that have been disconnected.

3.2. Relaxing Cache Consistency

As [Barbará and Imielinski, 1995] suggest, relaxing the consistency of the mobile unit's cache may allow for a smaller invalidation report. This may be possible depending on the application in question. For instance, it may be perfectly acceptable for an

application to serve a stock quote from its local cache that has already been updated at the server. In this situation, the user doesn't need to have the latest stock quote value, therefore the stock quote only needs to be retrieved periodically and not every time it is updated at the server.

[Yuen et al., 2000] expand on this concept and introduce a cache invalidation scheme they call *Invalidation by Absolute Validity Interval* (IAVI). This strategy identifies an *Absolute Validity Interval* (AVI) for each cached data item, and represents the useful life-span of the data item. A cached data item is invalid when its life-span has been exceeded, which is determined by using its last update time and AVI.

The IAVI scheme relies on the proper estimation of the AVI, which is based on the past update pattern of the data item. To measure the effectiveness of the AVI scheme, [Yuen et al., 2000] introduce two new metrics: *False Valid Period* (FVP) and *False Invalid Period* (FIP). The FVP measures when the AVI has been over-estimated, and therefore when the life-span of the data item has been over-estimated. In this scenario, the user will be using a data item whose value has been updated on the server but which has not yet been replicated to the mobile unit. The FIP measures when an AVI has been under-estimated. In this case, the mobile unit will request a refresh from the server when a refresh is not required, because the mobile unit still has the most up-to-date copy of the data item. Both FVP and FIP can be minimized by choosing a suitable AVI for the data item.

The server will use the IR to let the mobile unit know when the AVI for a data item has decreased. This will prevent a client from using a stale data item. When the AVI for a data item has increased the server will not let the client know so as to minimize the size

of the IR. In this scenario there is no danger in not telling the mobile unit because the data will still be valid if requested by the user. By employing this strategy, the IAVI IR size is much less than the BS and TS approaches described earlier.

The performance of IAVI was slightly better than BS and TS in terms of response time, due to the fact that the IR was much smaller, and users could get their answers quicker. However, BS and TS have higher cache hit ratios than IAVI.

3.3. The Updated Invalidation Report

One major drawback of the IR approach is that a user query must wait (on average) until half of the next IR has been received before the mobile unit can determine if the item is in its cache or not. For instance in the case of a cache miss, the client must wait on average ½ L, which is the interval between IRs, plus half of the time it takes to receive the IR, plus the time it takes to send a query to the server and receive a reply. In the case of a cache hit, it is the same as the cache miss minus the time to send the query to the server and receive a reply.

To reduce this latency, [Cao, 2002b] proposes a new strategy called the *Updated Invalidation Report* (UIR). The UIR indicates the data items that have been updated since the last IR. The UIR does not contain the historical update information of the last w intervals as the IR does and, as a result, the size of the UIR is significantly less than the size of the IR. Therefore the UIR can be transmitted a number of times between two successive invalidation reports without increasing overall bandwidth usage.

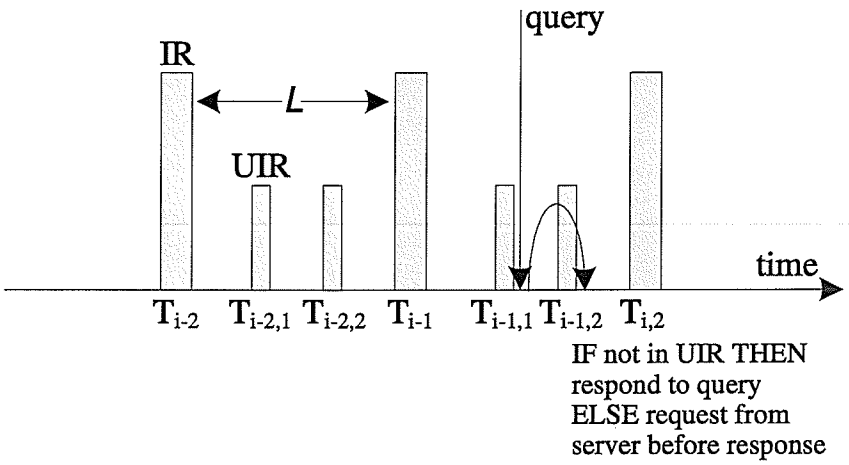


Figure 4 – The Updated Invalidation Report

In Figure 4, $T_{i,k}$ represents the time of the k^{th} UIR after the i^{th} IR. When a mobile unit receives a user query between $T_{i-1,1}$ and $T_{i-1,2}$, it cannot answer the query until T_i in the IR-based approach, but it can answer the query at time $T_{i-1,2}$ in the UIR-based approach (assuming a cache hit). Thus latency is reduced in the UIR-based approach. However, since the UIR does not contain history information, a mobile unit that awakens between IRs must wait until the next IR before validating its cache.

The UIR approach achieved higher cache hit ratios and lower query delays than the TS approach, while only requiring slightly higher broadcast overhead.

3.4. Enhancing Disconnection Strategies

One of the significant focal points in this problem domain is in determining how to gracefully handle long disconnections by the mobile unit. While the mobile unit saves energy by being disconnected from the server, it is not receiving invalidation reports and in turn keeping its cache consistent. [Wu et al., 1998] present one approach to handling long disconnections called *Grouping with COld update-set REtention* (GCORE).

One simple approach to updating the mobile unit's cache after awakening is to send all cached data ids to the server and have the server return a report of invalidated data ids back to the mobile unit. However, this is costly because the number of cached items may be quite large. Another simple approach is to group cached data items and send the group ids to the server. The restriction with this approach is that all items in the group must be discarded even if only one item in the group has been updated.

Using the GCORE approach, the mobile unit requests a group validity report from the server. When compiling the report the server examines the validity of a group, but ignores invalid data items that will be sent in the next IR. The items that are being sent in the next IR tend to be the hot data items, i.e. those data items which are most frequently updated. In this way, a group with invalid items may be considered valid and when the next IR arrives at the mobile unit it will invalidate those specific hot data items that are invalid. This allows the mobile unit to retain the cold data items, and not discard them from its cache unnecessarily.

GCORE has better performance, in terms of reduced energy consumption and bandwidth requirements, over sending all ids to the server and over the approach of invalidating whole groups without considering the hot data set items.

3.5. Optimizing Power Consumption

Another grouping approach was provided by [Cai and Tan, 1999], who offered a group scheme based on the invalidation report strategy. In their scheme, a mobile unit organizes its cache into groups and invalidation at the cache is performed at the group

level. Three schemes are proposed: *Group-based Cache Invalidation* (GCI), *Hybrid Cache Invalidation* (HCI), and *Selective Cache Invalidation* (SCI).

In the GCI scheme, the invalidation report takes the form of a *group invalidation report* (GIR) that issues groups ids and update timestamps for each group instead of the traditional invalidation report format. The advantage of this approach is that the GIR is a fixed size based on the number of groups the database has been partitioned into. In addition, the GCI scheme can handle arbitrarily long disconnections because it does not depend on the parameter w (see Section 3.1). When a mobile unit awakens from a disconnection, it only needs to listen for the next GIR to determine what groups in its cache are invalid. Of course what data items belong to what group must be agreed upon previously by the mobile unit and the server.

Not only can the client tune the size of the GIR based on energy consumption but, since the GIR is broadcast in group order, the mobile unit need only listen to relevant groups. The mobile unit can stay in doze mode, conserving energy, while the GIR is broadcast and only awaken to listen to portions of the GIR it is interested in, given that the mobile unit knows the group ids it has cached.

One of the issues with the GIR schemes is that false invalidations will occur, because even if only one data item in the group becomes invalid the entire group is considered to be invalid. To minimize the impact of false invalidations, the size of the database groups must be carefully chosen and tuned. The HCI scheme addresses this problem by issuing two invalidation reports from the server. The first report is the group invalidation report, and the second is the traditional invalidation report. The IR is configured with a small w to handle short mobile unit disconnections. This hybrid approach allows for the mobile

unit to pick up all data updates at the server when it is connected. In addition, the format of the HCI allows the mobile unit to recover from long disconnections and not have to invalidate everything in its cache.

The final scheme that [Cai and Tan, 1999] propose is the SCI scheme. The SCI scheme is very similar to the HCI scheme with a few exceptions. Instead of issuing the traditional IR first, the GIR is broadcast first and then the IR. This allows the mobile unit to tune to specific portions of the IR based on invalidation information within the GIR, thereby reducing the energy requirements of listening to the entire invalidation report as in the HCI scheme. This scheme requires that the invalidation report be ordered by group id so that all updates within the same group will appear consecutively. Since the mobile unit knows which group ids are contained in its cache it can tune into only the portions of the invalidation report that it is interested in, and doze for the rest of the report.

The decision of how to group the data items is based on the update rate and access rate of the data item in question. It would not make sense to place a data item that is frequently updated amongst a group that contains data items that are very often accessed, but infrequently updated. (This would create many false invalidations.) Thus, the groups were partitioned into four categories, Hot Update/ Hot Access, Hot Update/ Cold Access, Cold Update/ Hot Access, Cold Update/ Cold Access. In this way data items with the same characteristics would be grouped together.

All three schemes discussed achieved better performance than TS, while SCI provided the best overall performance of the three schemes.

3.6. Introducing Pre-fetching

A simple, broadcast only, pre-fetch scheme is presented in [Grassi, 2000]. No pre-fetch request is required from the mobile unit, instead the server determines what items the mobile units should have and broadcasts that information to all clients. Grassi proposed a function, $fv(i)$ to be calculated for each data item i in the database. This function is based on the access rate of the data item in question. Given that the size of the cache is a constant, the data items with the highest $fv(i)$ values will be pre-fetched from the client.

To improve the cache hit ratio of the IR/UIR scheme and to minimize the energy expenditure on the mobile unit, [Cao, 2002a] proposes a power-aware cache management algorithm. This algorithm uses pre-fetching to improve the cache hit ratio and thereby minimize the query response time.

After broadcasting the IR, an id list is sent to the mobile unit. The id list contains identifiers of data items whose values will be sent in a subsequent broadcast. After the list has been sent, the actual data values of those items in the id list are broadcast to the mobile unit. To save power, the mobile unit can sleep between invalidation reports and awaken only to receive the id list and the data values that it is interested in refreshing. In this way, mobile units can stay asleep much of the time.

[Cao, 2002a] presents two strategies: *stateful* server and *stateless* server, to determine what data items to include in the id list. In the stateful approach, the server maintains a counter for each data item. The data item's counter is incremented each time the data item is requested by the mobile units. The content of the id list is determined from these counters and the hot data items are then broadcast to the mobile units from the server.

In the stateless approach the server does not maintain a request counter per data item. Instead, when a query arrives at the server, the server does not immediately answer the query. Once the server has sent the next IR, it broadcasts a list of the identifiers of all data items that have been requested in the last IR period and then, like the stateful server approach, broadcasts the data values of the id list. The mobile unit will accept all data items whose ids match items in its cache and it will also look for data items that may not be in mobile unit's cache but that have been requested by it in the last IR interval. In the latter case, this will allow the mobile unit to respond to the request without having to wait until the next IR.

The stateless server strategy achieved a higher cache hit ratio and lower query delay than the TS approach. The stateful server strategy achieved similar results to the stateless server approach and was also able to serve more queries per time period than TS.

As mentioned previously, it takes significantly more power for the mobile unit to send a message than for the mobile unit to receive a message. The pre-fetching schemes presented in [Cao, 2002a] and [Cao, 2004] dynamically optimize power consumption against performance based on the available system resources (e.g. battery power) and performance requirements. [Cao, 2002a] introduces the notion of *pre-fetch access ratio* (PAR) which is the number of pre-fetches for a given data item divided by the access rate for that data item. When PAR < 1 pre-fetching the data item is useful because it is likely that the data item will be accessed again before another pre-fetch is required. If PAR > β, where β is a system parameter, then the item is *not* pre-fetched. This allows the mobile unit to adjust the number of pre-fetches it makes in order to conserve power at the client. When β is small, energy will be conserved at the expense of increasing the query latency

at the client. When β is large, query latency will be reduced at the expense of reducing mobile unit power levels more rapidly.

To handle items that are required frequently at the client (i.e. that have a high access rate) the mobile unit will fetch N_p data items with a PAR > β. In this scenario, the energy requirements to pre-fetch the items is not that great considering that it may significantly increase the cache hit ratio and reduce query latency. Therefore, the trade-off between performance and power is also determined by the value of N_p.

[Cao, 2004] proposes two methodologies for managing N_p. The first scheme called AVP_T is an adaptive value-based pre-fetch which adapts N_p to achieve a certain battery life at the client. A scenario that [Cao, 2004] proposes is where a commuter knows how long it will take them to get from source to destination where, presumably, they are able to re-charge their mobile unit. This scheme allows the mobile unit to determine the proper value of N_p based on past energy consumption rates and the current value of N_p. If the mobile unit is consuming energy at a rate which will not allow it to reach the specified destination, then N_p is reduced appropriately. Conversely, if the rate of energy consumption will allow the mobile unit battery life to extend significantly beyond the time frame for reaching the destination, then N_p can be increased to reduce query latency and thereby increase performance. Although this methodology is intuitive, [Cao, 2004] provides no clear algorithm for determining past energy consumption rates nor how significantly N_p should be increased or decreased to hit certain targets.

The other scheme presented by [Cao, 2004] is called AVP_P. This scheme is an adaptive value-based pre-fetch scheme which adjusts N_p using a threshold function based

on the current energy levels at the client. This is shown in Equation 5 which is described in the following Section.

3.7. Value-Based Adaptive Pre-fetch

The work presented by [Grassi, 2000] and [Cao, 2002a] does not account for varying data sizes of the data items and there is no consideration of the update rate of the data items at the server. In addition, [Grassi, 2000] does not address the mobile unit's power constraints. While [Cao, 2002a] does address power constraints, his work doesn't provide a clear determination of when β should be altered.

Expanding on the work of [Grassi, 2000], [Cao, 2002a] and [Cao, 2002b] is the research of [Yin et al., 2002] which demonstrates that by considering power constraints, data item size, data access rate, and data update rate, further efficiencies can be achieved. These considerations are balanced in a scheme called *value-based adaptive pre-fetch* (VAP), which optimizes the pre-fetch cost to achieve better performance. VAP focuses on pre-fetching items of small size, low update rate and high access probability.

A value-based function calculates the worth of each cached data item. This information is then used to determine what data items should be pre-fetched from the server. The *d-value* function, shown in Equation 1, is described in terms of the following variables:

p_i : the access probability of data item i

s_i : the size of data item i

b_i : the retrieval delay from the server for data item i

$\overline{a_i}$: the mean access arrival rate for data item i

$\overline{u_i}$: the mean update arrival rate for data item i

v : the cache validation delay

$$d-value(i)=\frac{p_i}{s_i}\left(\frac{b_i\cdot\overline{a_i}}{\overline{a_i}+\overline{u_i}}-v\right)$$

Equation 1 – VAP function to calculate data item worth

Since b_i, $\overline{a_i}$ and $\overline{u_i}$ are not constant, estimation equations must be provided. An exponential aging method is used to calculate b_i, provided by Equation 2, where $\alpha < 1$ is used as a factor to lessen the impact of old values of b_i.

$$b_i=\alpha\cdot b_i^{new}+(1-\alpha)\cdot b_i^{old}$$

Equation 2 – Estimation of b_i

The estimation equations for $\overline{a_i}$ and $\overline{u_i}$ are shown in Equation 3 and Equation 4, where T represents the current time, and $T_{\overline{a_i}}(K)$ and $T_{\overline{u_i}}(K)$ represent the K^{th} most recent samples of the mean access rate and mean update rate, respectively. Note that these values must be stored for each item in the database, so to keep the number of samples to a reasonable size, K is set to 2.

$$\overline{a_i}=\frac{K}{T-T_{\overline{a_i}}(K)}$$

Equation 3 – Estimation of $\overline{a_i}$

$$\overline{u_i}=\frac{K}{T-T_{\overline{u_i}}(K)}$$

Equation 4 – Estimation of $\overline{u_i}$

Only the N_p highest *d-value* data items are pre-fetched from the server to the mobile unit, where N_p is the pre-selected number of items to pre-fetch. The VAP scheme uses a discrete function to calculate the number of data items that should be pre-fetched as a function of the energy remaining on the mobile unit. As the power in the mobile unit decreases, pre-fetching is performed less aggressively in order to save energy. Only the highest *d-value* items are pre-fetched from the server. Equation 5 describes the VAP pre-fetch threshold function, where the variable a_k represents the percentage of energy remaining on the mobile unit.

$$f(a_k) = \begin{cases} 100\% & 0.5 < a_k \leq 1.0 \\ 70\% & 0.3 < a_k \leq 0.5 \\ 50\% & 0.2 < a_k \leq 0.3 \\ 30\% & 0.1 < a_k \leq 0.2 \\ 10\% & a_k \leq 0.1 \end{cases}$$

$$N_p = N_p \cdot f(a_k)$$

Equation 5 – VAP pre-fetch discrete threshold function

To evaluate the performance of VAP versus the UIR approach presented in [Cao, 2002a], two metrics called *stretch* and *energy-stretch* were introduced and are detailed below.

One of the most common metrics used to evaluate schemes in a client/server environment is the *response time* of a request. The response time is the time between a client issuing a query and receiving a response from the server. This metric is most useful in environments where all data items are homogenous in size. However, in cases were data items may be heterogeneous in size, the *service time* of the request should also be

factored into evaluation metrics. The service time is the time to complete the request as if it were the only request being handled by the server. [Acharya and Muthukrishnan, 1998] propose the use of a metric called *stretch*, defined in [Bender et al., 1998], to evaluate the performance of schemes in the wireless computing domain. Stretch is defined to be the ratio of the response time of a request to its service time (Equation 6). This metric more fairly accounts for the servicing of jobs of different sizes – users requesting larger data items should be willing to wait longer for their requests to be serviced.

$$stretch = \frac{reponse\ time\ of\ request}{service\ time\ of\ request}$$

Equation 6 – Definition of stretch

Energy-stretch, as shown in Equation 7, is defined to be the stretch of a query multiplied by the total energy consumed by the query. [Yin et al., 2002] believes this metric provides a better measure of system performance than average stretch alone because it factors in energy consumption. However, since the purpose of stretch is to normalize the response time by the service time of the request, then adding energy consumed per query is effectively negating this normalizing effect. More energy will typically be consumed by queries that request larger data items due to the increased download costs. Regardless, given that [Yin et al., 2002] uses this metric to evaluate experiments, this thesis will also use this metric for comparison purposes.

$$energy\ stretch = stretch * energy\ consumed$$

Equation 7 – Definition of energy-stretch

[Yin et al., 2002] shows that VAP reduces energy consumption and improves system performance over the UIR approach in terms of both stretch and energy-stretch. A

detailed description of the results of this research is presented in Section 5 and compared with the results of the technique proposed in this thesis.

3.8. Applying Stateful Servers

By storing the mobile unit's cache information on the server, [Kahol et al., 2000] was able to show, using a scheme called *asynchronous stateful* (AS), that arbitrary disconnection patterns could be supported while improving bandwidth usage over TS. Each mobile unit is assigned one home *mobile support station* (MSS) that contains the mobile unit's *home location cache* (HLC). The HLC contains the last update timestamp for each cached data item at the mobile unit. As the mobile unit moves into different geographical locations and is supported by a different MSS the new MSS will maintain a replica of the HLC from the home MSS. When a data item is updated on the server, every MSS receives notification that this data item has been updated. Instead of a periodic broadcast of the IR, AS broadcasts an IR only when information on the server is updated. In this way, invalidation reports are only sent when necessary.

As mentioned, AS supports arbitrary disconnection times. When a mobile unit reconnects to the base station after being in sleep mode, it can send a *probe* message, along with its cache timestamp, to the server indicating that it has just connected. The server can then tailor an IR for the recently connected mobile unit. All user queries at the mobile unit must wait until the first IR has been received, before they can be answered.

4. Solution Strategy

This thesis proposes a new cache management scheme called *stateful value-based adaptive pre-fetch* (SVAP). SVAP is an enhanced version of VAP proposed by [Yin et al., 2002] using ideas developed in the AS scheme proposed by [Kahol et al., 2000]. SVAP shows improved results over VAP in terms of energy consumption and, in addition, supports arbitrary disconnection patterns, which is a characteristic drawn from the AS scheme. While arbitrary disconnection patterns are supported, the main thrust of the evaluation of the results of this thesis center around energy consumption. The results of VAP, published in [Yin et al., 2002] have been duplicated in a simulation and are used as a baseline with which to compare SVAP. This thesis will directly compare the stateless server approach of VAP with the stateful server of SVAP in terms of energy consumption. The two implementations (VAP and SVAP) are discussed in the following sections.

4.1. The VAP Client

The VAP client is a mobile unit that produces a steady stream of read-only queries. The data access pattern follows the Zipf distribution [Zipf, 1929], which has often been used to model non-uniform distributions [Breslau et al., 1999]. The access probability of the i^{th} data item, where $1 <= i <= n$, is shown by Equation 8. This equation permits $0 <= \theta <= 1$, where $\theta = 0$ is the uniform distribution and when $\theta = 1$ provides the strict Zipf distribution.

$$P_i = \frac{1}{i^{\Theta} \sum_{i=1}^{n} \frac{1}{i^{\Theta}}}$$

Equation 8 – The probability of accessing the i^{th} data item in the Zipf distribution

In a strict Zipf distribution of a 2000 item data set with $\theta = 1$, the leading two most frequent items are accessed approximately 18% of the time, while the top 5 items are accessed approximately 28% of the time (see Table 1). All experiments in this thesis use $\theta = 0.9$ and the distribution of the first 7 elements are shown by Table 2.

Data Item	Probability
1	12.2%
2	6.1%
3	4.1%
4	3.1%
5	2.4%
6	2.0%
7	1.7%
…	

Table 1 – Zipf distribution of 2000 items with $\theta = 1$

Data Item	Probability
1	8.4%
2	4.5%
3	3.1%
4	2.4%
5	2.0%
6	1.7%
7	1.5%
…	

Table 2 – Zipf distribution of 2000 items with $\theta = 0.9$

The mobile unit can generate two types of messages that are sent to the server. The first message, called `send_prefetch_request`, allows the mobile unit to request the top N_p most valuable items for its cache from the server. The value of the data items are determined by the *d-value* of the data item, as described in Section 3.7. The second

message that the mobile unit can generate is `send_query_request`. This message is used when the user has requested a data item that is either not in the cache or is in the cache but has just been deemed invalid by the incoming IR.

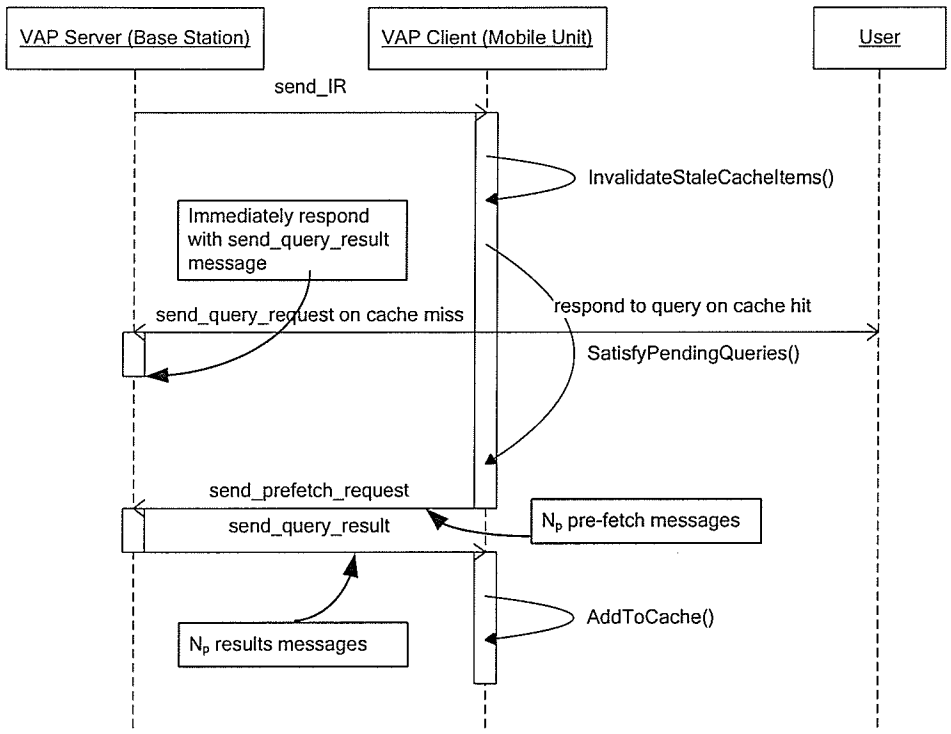


Figure 5 – VAP IR sequence diagram

In addition to outbound messages, the VAP client must be able to handle three types of incoming messages from the server using three corresponding event handlers. The first, called `send_IR` (see Figure 5), processes the incoming IR message from the server. When this event handler is triggered, the mobile unit will invalidate all stale items in its cache based on the information in the invalidation report. It will then attempt to satisfy all pending queries that have been issued since the last IR. If the query has requested a cached item that is still valid, the query result can be returned immediately

from the local cache. If the cached item is now invalid then the mobile unit must generate a `send_query_request` message to the server to get the latest value.

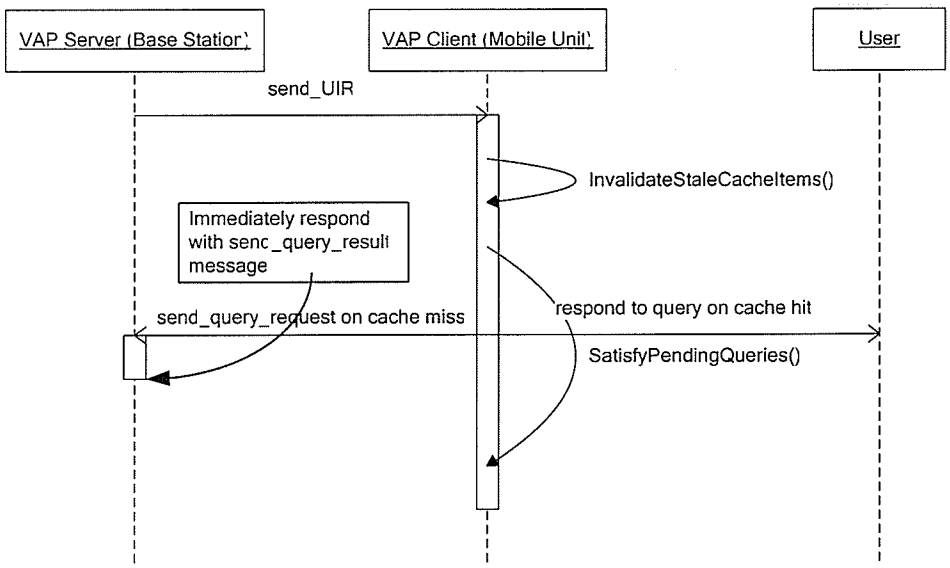


Figure 6 – VAP UIR sequence diagram

The mobile unit also provides a `send_UIR` event handler (see Figure 6). This handler is triggered when the server has sent a UIR to the mobile unit. The mobile unit will invalidate stale items in its cache given the content of the UIR, and will attempt to satisfy all pending queries as it did in the `send_IR` handler. The third message the VAP client can handle is the `send_query_result` message. The processing done in response to this message returns the result of a `send_query_request` message that was initiated by the mobile unit and which contains the updated data value of the data item in the database.

The power cost of the messages is different for uplink and downlink of messages. As discussed, the uplink (i.e. transmission) of messages from the mobile unit is more costly than reception of messages from the server.

The client also has the ability to use power consumption adaptation as described in Section 3.7. When the power level at the client reaches a certain threshold the number of items marked for pre-fetch, N_p, is reduced, as shown by Equation 5. This allows the mobile unit to temper the aggressiveness of the pre-fetch when there is less power available and therefore attempts to prolong the battery life of the mobile device.

In addition to differences in cost between uplink and downlink notifications, the bandwidth also varies between the two directions. The uplink bandwidth is assumed to be 14.4 kbps while the downlink bandwidth is assumed to be 144 kbps.

4.2. The VAP Server

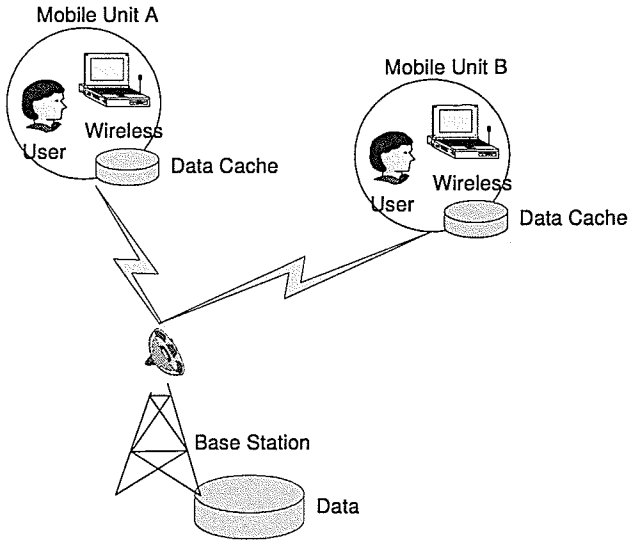


Figure 7 – VAP model

For simplicity, the simulation places the database at the base station instead of on another machine (see Figure 7). The database is composed of 2000 items with a data access pattern that follows the Zipf distribution where θ = 0.9. The database items progress linearly in size from the i = 1 data item equal to 0.5k (S_{min}) in size to the i = 2000 data item at 10k (S_{max}) in size. The database size, θ value and size distribution have been chosen because they are the values that [Yin et al., 2002] used in their experiments. As a result of the correlation between the data item access pattern and the distribution of size, the smallest data items will be accessed most frequently and the largest data items will be accessed most infrequently. This has been shown to be a valid assumption in the work of [Breslau et al., 1999].

Simulation Parameter	Value
Database size	2000 items
Number of clients	100
Smallest data item (S_{min})	0.5k
Largest data item (S_{max})	10k
Mean update time	100 seconds
Mean query generate time	100 seconds
Hot update probability	80%
Hot subset percentage	20%
Broadcast interval (L)	20 seconds
UIR interval	4x within L
Broadcast bandwidth	144 kbps
Uplink bandwidth	14.4 kbps
Cache size	20% of db size
Zipf distribution θ	0.9
Mobile unit receive power dissipation	0.2 watts
Mobile unit transmit power dissipation	0.5 watts

Table 3 – Simulation default parameters

The server produces a steady stream of database updates, where 80% of the updates are distributed randomly among elements in the hot data set. (The hot data set is assumed

to be the first 20% of the data items in the database, which are the smallest and most frequently accessed data items.)

Every 20 seconds, the IR is sent to the mobile unit in a `send_IR` message and UIRs are sent out 3 times within that interval. Each UIR is contained in a `send_UIR` message. The server must also handle two incoming events from the mobile unit, namely `send_prefetch_request` and `send_query_request` as described earlier.

4.3. The SVAP Client

The SVAP client follows the same pattern of processing as the VAP client except for the differences described in this section. The first difference is that the `send_IR` notification handler is modified to *not* request pre-fetch information from the server as shown by Figure 9. The handler will try and satisfy pending queries as usual.

The other difference in the SVAP client is that the adaptation calculation no longer happens at the client. After the IR has been received, the mobile unit sends a new notification message called `send_energy_remaining` to the server. This message tells the server how much power is left at the client, which allows the server to anticipate how many items the mobile unit would have requested in a pre-fetch. In this way, the power level at the mobile unit is further conserved and is not dissipated by receiving a large amount of data from the server.

4.4. The SVAP Server

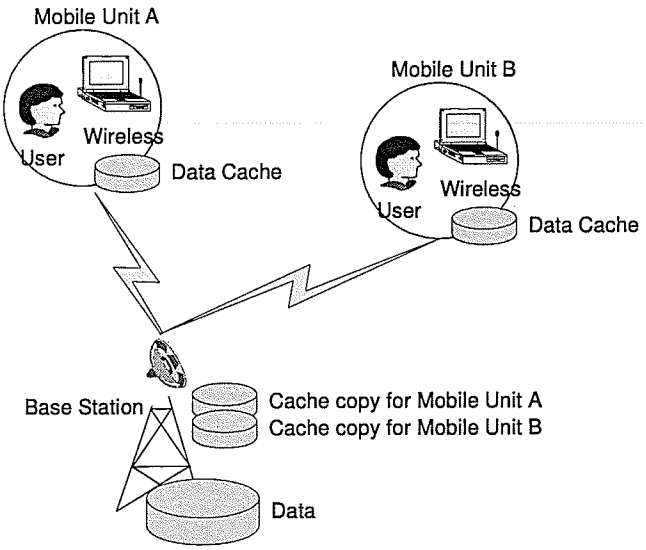


Figure 8 – SVAP model

The VAP server does not contain any information about the mobile units within its geographical area, and in this sense is deemed stateless. This is a significant distinction from the SVAP server that contains copies of each mobile unit's cache (see Figure 8). With this information the server is able to tailor the IR and UIR for specific mobile units.

When constructing the IR for a specific mobile unit, the server will only send information about the stale cached data items that it knows exist at the mobile unit (see Figure 9). In addition, it will maintain an up-to-date copy of the mobile unit's cache in the process by invalidating items in the cache copy it knows are invalid.

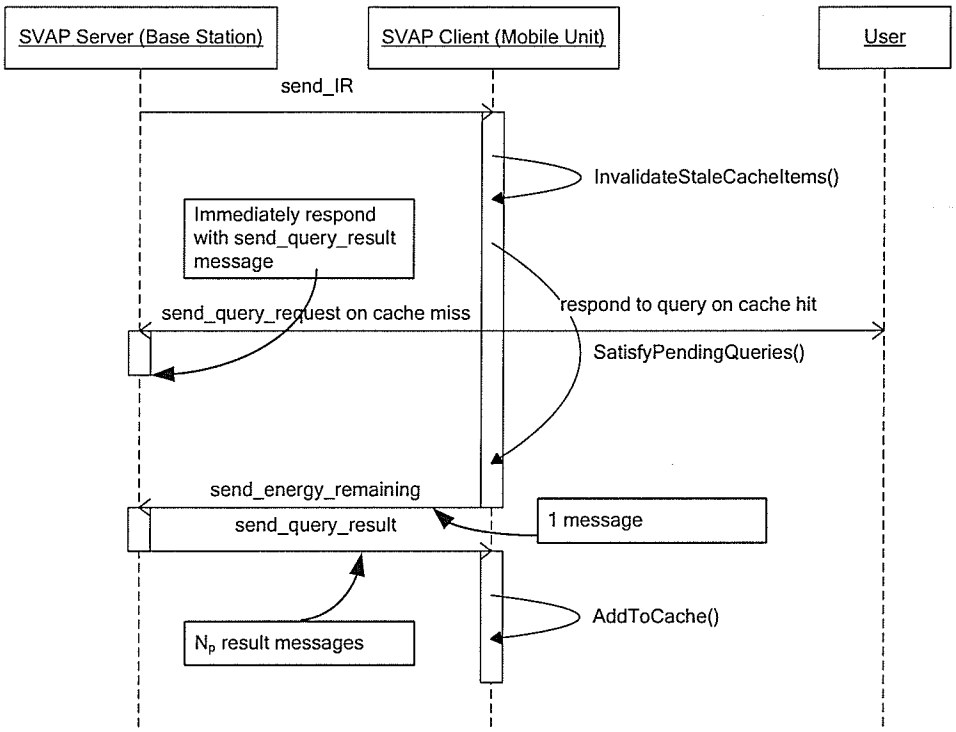


Figure 9 – SVAP IR sequence diagram

There is also a new notification handler at the SVAP server for the `send_energy_remaining` message from the mobile unit. When this handler is triggered, the server will use its knowledge of the power remaining at the mobile unit, sent in the `send_energy_remaining` message, to determine the correct value of N_p given the threshold function introduced in Equation 5. The server will then determine the top *d-value* items in the database and send these values to the client. In addition to sending these items to the client, it will also add them to the mobile unit's cache copy at the server. If the mobile unit already has some of the top *d-value* items, then the server will not send these data items back to the mobile unit thereby conserving downlink bandwidth.

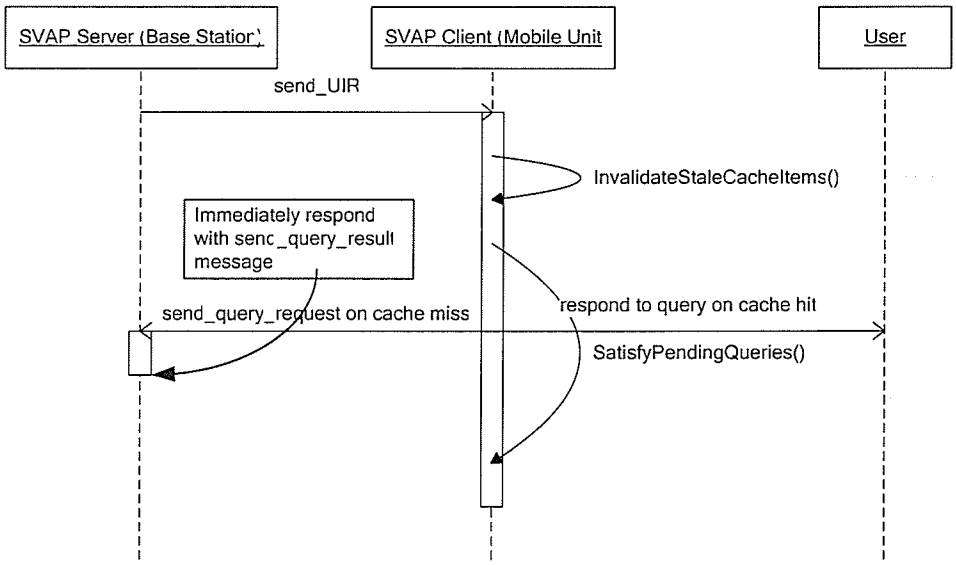


Figure 10– SVAP UIR sequence diagram

As Figure 10 shows, the SVAP UIR sequence is no different than the VAP UIR sequence shown in Figure 6.

5. Simulation Evaluation

The VAP simulation has been reproduced and will be used as a baseline for comparing the results of SVAP. Every figure presented in [Yin et al., 2002] will be discussed and compared against the new SVAP scheme. In addition to measuring power consumption, the simulation also compares bandwidth use between VAP and SVAP. Unfortunately [Yin et al., 2002] did not track this metric when evaluating VAP, however, since the VAP simulation has been re-created, these metrics are now being tracked by the replicated VAP simulator and are compared with the SVAP scheme directly.

The simulation models the performance of 100 clients connected to one base station. Most of the simulation parameters are set to the same values that [Yin et al., 2002] used in their experiments. Some parameters were set to different values to achieve similar trends as those reported by [Yin et al., 2002] - the parameter values used are documented below. A description of the differences between [Yin et al., 2002]'s VAP scheme and the replicated VAP scheme are described in the next section.

In all results shown in this chapter, each data point is the mean of a number of trials. The number of trials performed was chosen to ensure that the differences in the means between SVAP and VAP were statistically significant. A two-way analysis of variance was performed for each simulation configuration, and the results are shown below.

5.1. Reproducing VAP

Care was taken to reproduce the VAP simulation, yet the new simulation was not able to exactly reproduce the specific y-values shown in the figures presented by [Yin et al.,

2002]. However, the integrity of all the data trends have been reproduced, with the few exceptions described and explained below.

There are a number of possible reasons for differences in the specific y-values. One possible reason for value differences in the experiment described later (in Section 5.6) is that [Yin et al., 2002] does not describe precisely when N_p should be re-calculated. They write that N_p should be calculated periodically. This thesis experiment has chosen to recalculate N_p with the same periodicity as the invalidation report interval L.

Another potential discrepancy is that it is unclear whether the *d-value* is calculated at the server or on the client. [Yin et al., 2002] mention that the mean update rate will be piggy-backed on the IR so that the client may use it. They do not mention if this is to determine what items to remove from the cache when the cache is full or if this is to determine which items to pre-fetch next. In the replicated VAP simulation I have chosen to include the *d-value* for each item in each IR tuple. Thus, instead of the invalidation report containing tuples of the form (d_x, t_x), this simulation will use the form (d_x, t_x, *d-value*$_{dx}$) where *d-value*$_{dx}$ is the *d-value* for data item d_x. In the case of SVAP, the invalidation report will contain tuples of the form (d_x, *d-value*$_{dx}$) because the server will be sending pre-invalidated items to the client. The SVAP client need not perform a timestamp check against items in the cache, since the server knows its contents. Note that there is a timestamp at the SVAP client which indicates the time at which their cache is valid. This timestamp can be transmitted to the server after a disconnection to retrieve all invalidation information since that time.

In addition, storing the *d-value* at the client allows the mobile unit to choose judiciously which items to pre-fetch. Since the client knows the top *d-value* items it does

not need to request a pre-fetch from the server for a top *d-value* item that already resides in its cache. Instead the client can be satisfied that the cache has the top *d-value* items and it can then request fewer items to pre-fetch thereby expending less energy for both the uplink of the pre-fetch request and the downlink of the requested item.

Another difference is the reported values of stretch and energy-stretch. The reason for this is that it is not clear how [Yin et al., 2002] calculates the response time and service time variables necessary to determine stretch and therefore energy-stretch. The replicated VAP stretch values are 3 to 4 orders of magnitude higher than [Yin et al., 2002]'s reported results. A hint at why this discrepancy exists may be that Guohong Cao, one of the authors of [Yin et al., 2002], writes in [Cao, 2004] that he has removed the constant bandwidth factor in the stretch calculation's denominator, i.e. the service time calculation. However, this would only increase the service time, and therefore decrease stretch even further. It is probable that [Yin et al., 2002] has also removed some constants from the stretch numerator, i.e. response time, for the purpose of computational efficiency.

One last difference is that the replicated simulation of VAP does not include w IR update notifications in its IR. The reason for this simplification is that no experiments below consider the effects of disconnection time and no variables in [Yin et al., 2002] were given to describe at what frequency mobile units go to sleep and wake up. With no evaluation of the performance given w, it was considered a constant factor that could be removed from the equation. Also, it can be safely said that VAP would only perform worse than SVAP if w were considered, since SVAP doesn't require the server to broadcast w IR update notifications. The extra size of the VAP IR would use more energy

at the client and require higher downlink bandwidth. When a client awakes in the SVAP scheme they send a request to the server with the latest timestamp to retrieve all invalidations since that time, as in the AS scheme.

5.2. Varying the Update Arrival Time

This experiment compared the effect of varying the mean update arrival time between VAP and SVAP. The mean update arrival time is the time between updates arriving at the server. Therefore, the shorter the interval, the more updates are occurring. For every interval cycle, 100 updates will occur at the database, given that the experiment models 100 clients. The determination of what items get updated is implemented as described in Section 4.2 and is based on the partitioning of the database into a hot and cold dataset. Determining what item is updated by a mobile unit is independent of updates that other mobile units are performing. In other words, if mobile unit 20 updates data item 22 at time interval 1200, this has no bearing on the updates done by other mobile units. Hence, mobile unit 33 may also update data item 22 at time interval 1200. This is done in order to simulate the possibility of multiple mobile units updating the same data item within the same update interval.

Simulation Parameter	Value
Simulation length	10000s
Mean query generate time	10s
Number of pre-fetches (N_p)	50
Number of trials	15

Table 4 – Varying update arrival time simulation variables

This experiment replicates that of [Yin et al., 2002]'s Figure 1 (Performances under varying update arrival time) parts (a), (b) and (c).

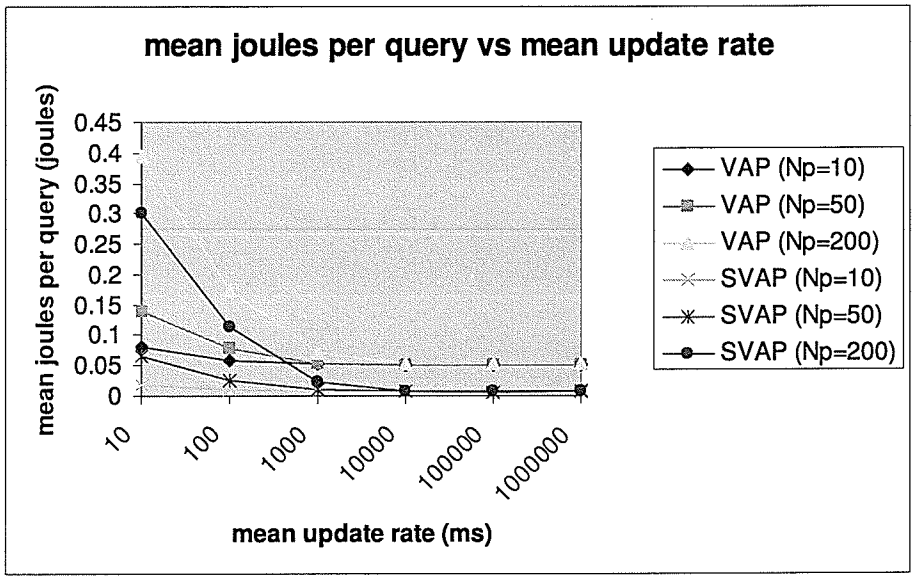


Figure 11 – Energy consumption under varying update arrival time

Mean joules per query is calculated by totalling all energy used at the mobile unit and dividing this number by the total number of queries (data requests) from the mobile unit. The queries counted include both cache hits and cache misses. As Figure 11 shows, SVAP consistently uses less energy per query than VAP in all cases when N_p is equal (recall that N_p specifies the number of items to pre-fetch). When the mean update time grows larger, there are fewer and fewer updates happening at the server. This in turn increases the cache hit ratio and lowers the number of joules spent per query. The number of joules per query decreases because when the mobile unit achieves a cache hit the client does not have to request the new data value from the server and does not incur the cost of an uplink request and the cost of downloading the new value.

When the mean update time is very low, there are many updates occurring at the server. This reduces the number of cache hits, and increases the number of times the mobile unit will have to request an updated data value from the server.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean update rate	5	5	0.00125407	1158.255	<.0001
Type	1	1	0.03093569	142860.3	0.0000
Np	2	2	0.69224455	1598385	0.0000
mean update rate*Type*Np	10	10	0.00303050	1399.477	0.0000
Type*Np	2	2	0.00369034	8520.951	0.0000
mean update rate*Type	5	5	0.00237682	2195.222	0.0000
mean update rate*Np	10	10	0.54648795	252366.9	0.0000

Table 5 – Effect table for energy consumption under varying update arrival rates

A two-way analysis of variance (ANOVA) test was performed to compare the energy consumption of SVAP and VAP under varying mean update arrival rates. The variable Type is an ordinal variable indicating "VAP" or "SVAP". Note that there are significant interaction effects between all three control variables: Type, mean update rate and N_P. There is strong evidence that energy consumption under SVAP is consistently less than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 5 and Figure 12, respectively.

ANOVA tests the statistical significance of the difference between two means, calculated from random samples of the entire population. The effect table shows the impact of each model effect, and assesses its significance with respect to the differences in the mean. The effects that are combined with a "*" indicate combination effects when considering the two effects together. When the value of Prob>F is < 0.05 this indicates strong evidence that the model effect fits the ANOVA model. The profile plots provide a graphical representation of the significance of a model effect. The horizontal line shows the sample's mean value, while the dashed lines indicate the confidence curves. If the curves cross the mean then the effect is significant, while if the area between the curves

contains the mean (i.e. the confidence curves do not cross the mean), then the effect is not significant [Sall et al., 2000].

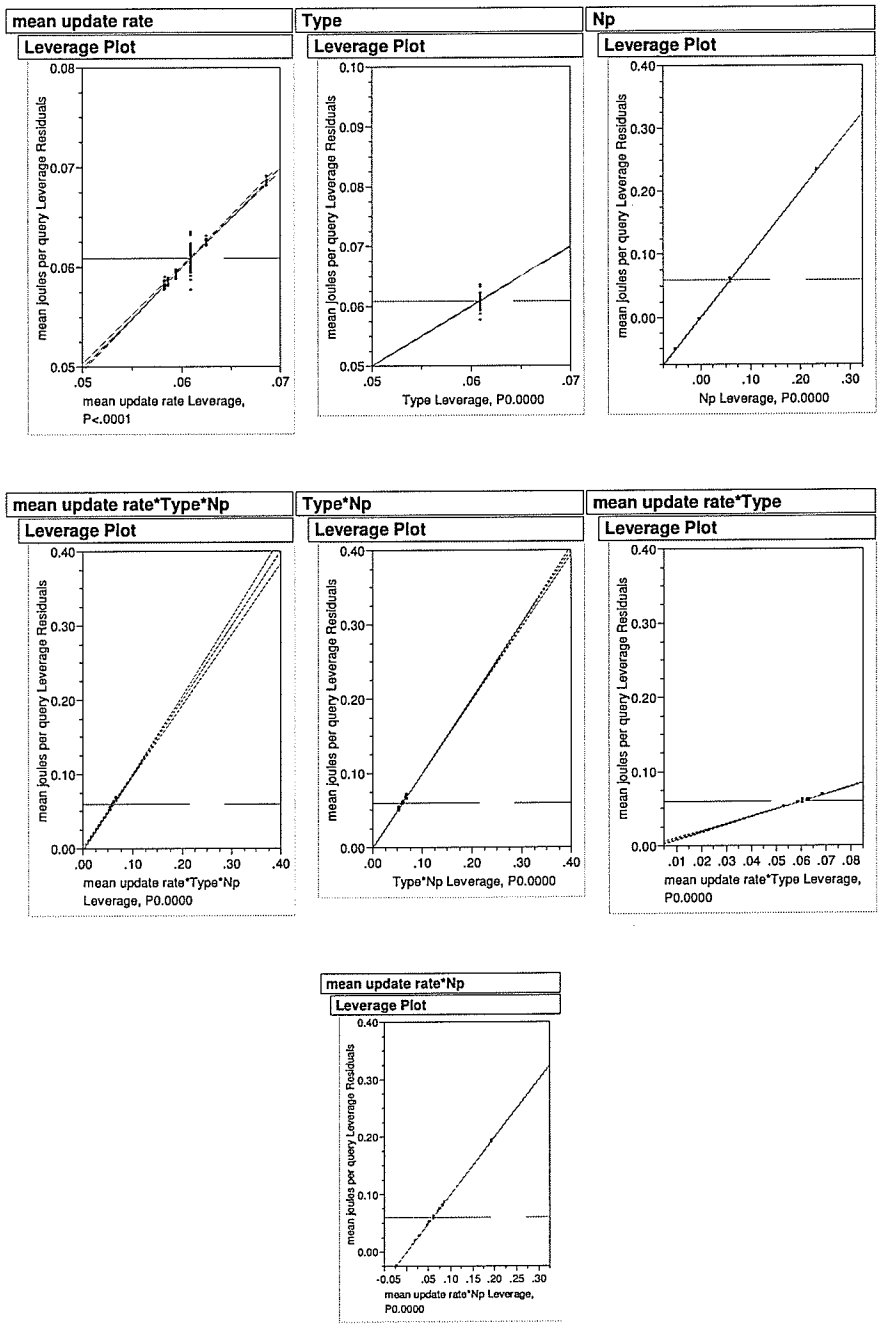


Figure 12 – Profile plots for energy consumption under varying update arrival rates

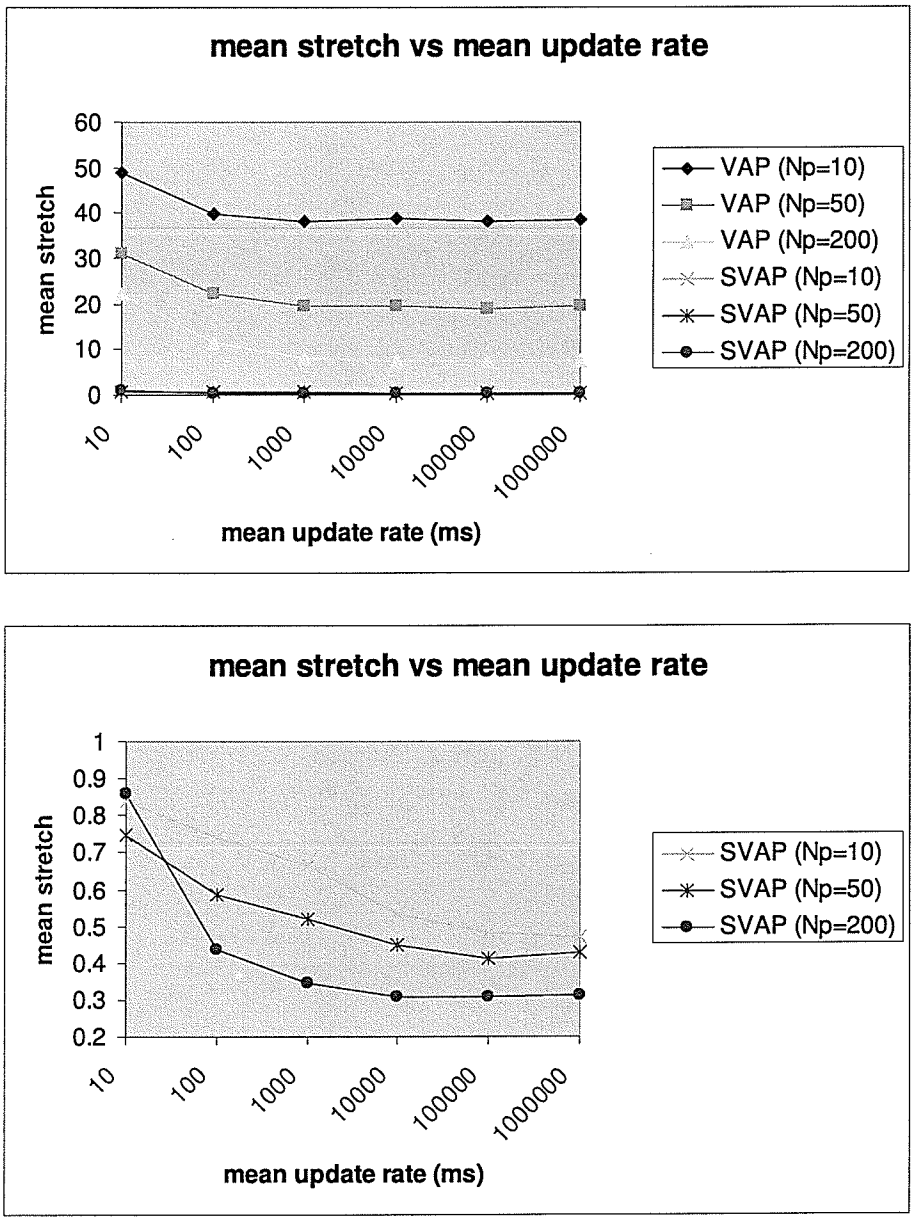


Figure 13 – mean stretch vs. mean update arrival time using two different y-axis scales

The VAP trends (see Figure 13) follow a slightly different trend than the results presented in [Yin et al., 2002]. While the replicated VAP results still show N_p = 200 having the lowest stretch and N_p = 10 having the highest stretch, the trends do not converge as the mean update rate increases. The explanation for these trends in the replicated VAP results can be explained as follows. As the mean update rate increases,

the cache will become saturated with the highest *d-value* items in the database. However, in this implementation of VAP, when the mobile unit cache already contains the highest *d-value* data items, it will not pre-fetch any other items from the database. Therefore, the cache hit ratio will begin to stabilize as the mean update rate increases, given that the cache contains the top N_p *d-value* data items. However, this doesn't necessarily mean that the entire cache is populated in the most optimal fashion. In other words, when $N_p = 50$ there may be 150 other items in the cache with non-optimal *d-values* given that the size of the cache is 20% of the database, or 400 items. When $N_p = 200$ there are more data items in the cache with higher *d-values* than in the $N_p = 50$ run. Thus, the cache hit ratio stabilizes at different stretch levels given the different number of items being pre-fetched from the database. Regardless, as can be seen by the trends of Figure 13, SVAP has a significantly lower mean stretch than VAP in all cases.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean update rate	5	5	1.707	0.4523	0.8117
Type	1	1	17242.303	22842.29	0.0000
Np	2	2	0.105	0.0695	0.9329
mean update rate*Type*Np	10	10	63.896	8.4648	<.0001
Type*Np	2	2	2730.152	1808.428	<.0001
mean update rate*Type	5	5	631.606	167.3479	<.0001
mean update rate*Np	10	10	0.713	0.0945	0.9999

Table 6 – Effect table for mean stretch under varying update arrival rates

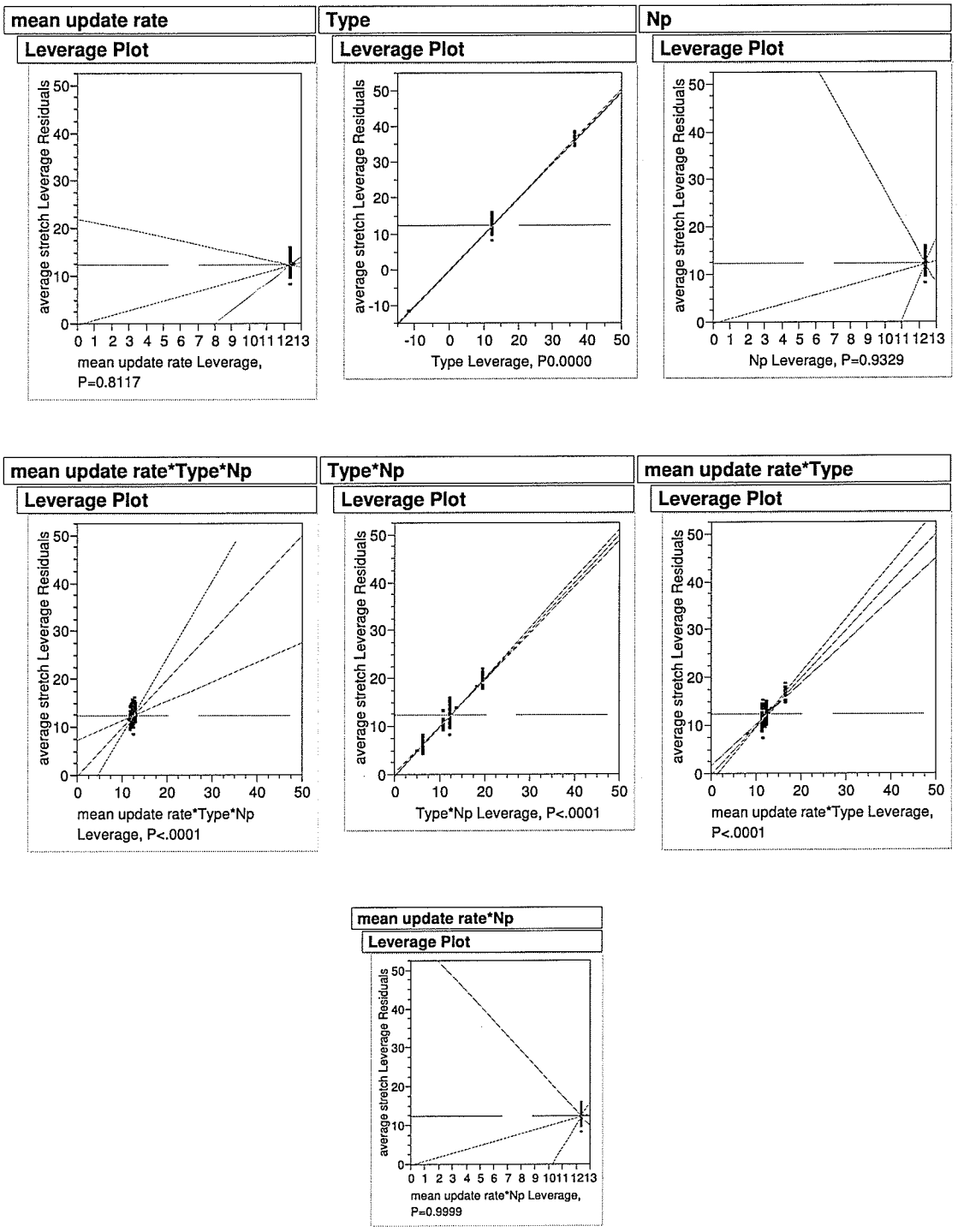


Figure 14 – Profile plots for mean stretch under varying update arrival rates

Again, a two-way analysis of variance (ANOVA) test was performed to compare the mean stretch of SVAP and VAP under varying update arrival rates. There is strong evidence that the mean stretch for SVAP is consistently less than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 6 and Figure 14, respectively.

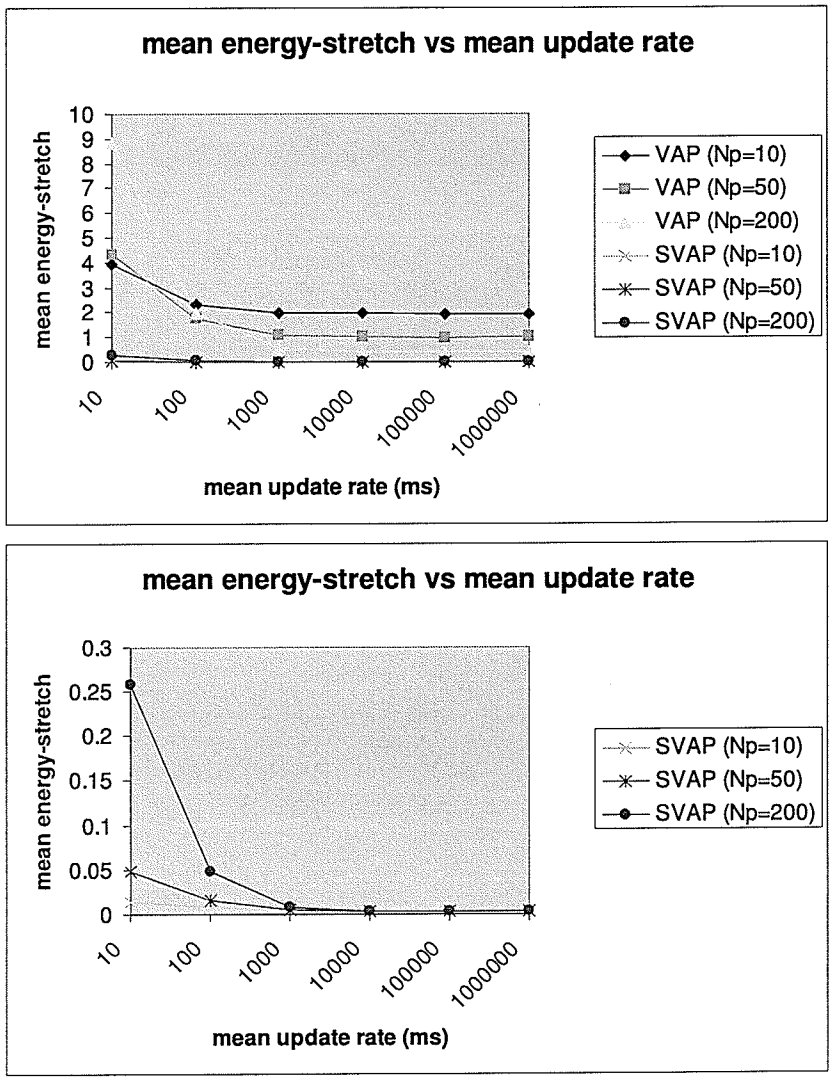


Figure 15 – Mean energy-stretch under varying update arrival time using two different y-axis scales

Since SVAP requires less information to be downloaded from the base station to the mobile unit, energy at the client is conserved, as shown in Figure 15. Less information must be downloaded because the SVAP server can tailor the UIR and IR given that it knows the contents of the mobile unit's cache. The server does not need to send update information about data items that it knows the mobile unit does not have in its cache. SVAP consistently uses less energy as the mean update time increases for a fixed value of N_p.

Note that the replicated VAP trends do not converge as do the results of [Yin et al., 2002]. This can be explained by the same reasoning discussed earlier for the mean stretch vs. mean update arrival time experiment.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean update rate	5	5	0.00136	0.0273	0.9996
Type	1	1	115.41835	11612.64	0.0000
Np	2	2	0.52520	26.4209	<.0001
mean update rate*Type*Np	10	10	125.53488	1263.049	0.0000
Type*Np	2	2	99.13311	4987.061	0.0000
mean update rate*Type	5	5	23.57664	474.4253	<.0001
mean update rate*Np	10	10	0.41915	4.2172	<.0001

Table 7 – Effect table for mean energy-stretch under varying update arrival rates

A two-way analysis of variance (ANOVA) test was performed to compare the mean energy-stretch of SVAP and VAP under varying update arrival rates. There is strong evidence that the mean energy-stretch for SVAP is consistently less than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 7 and Figure 16, respectively.

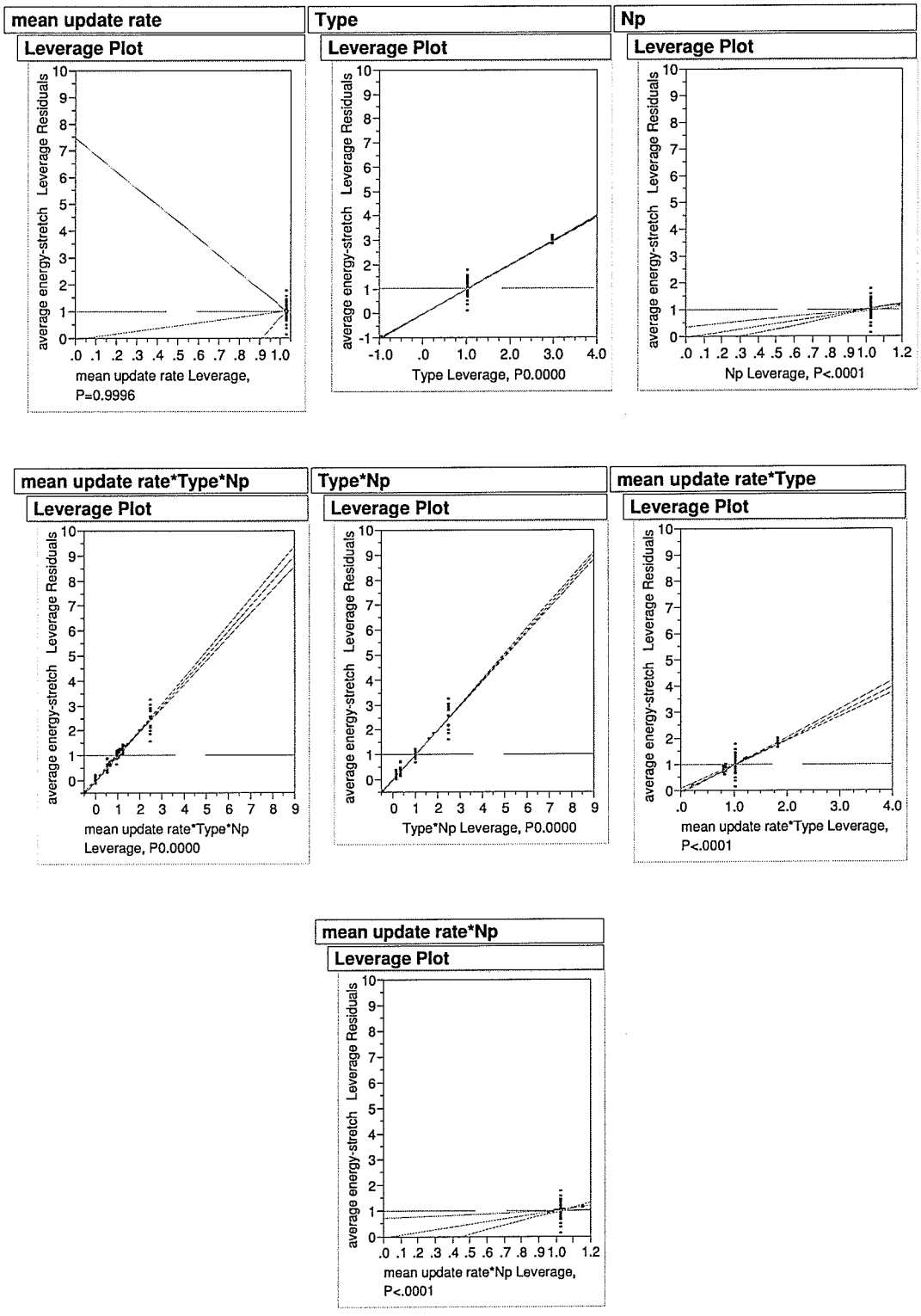


Figure 16 – Profile plots for mean energy-stretch under varying update arrival rates

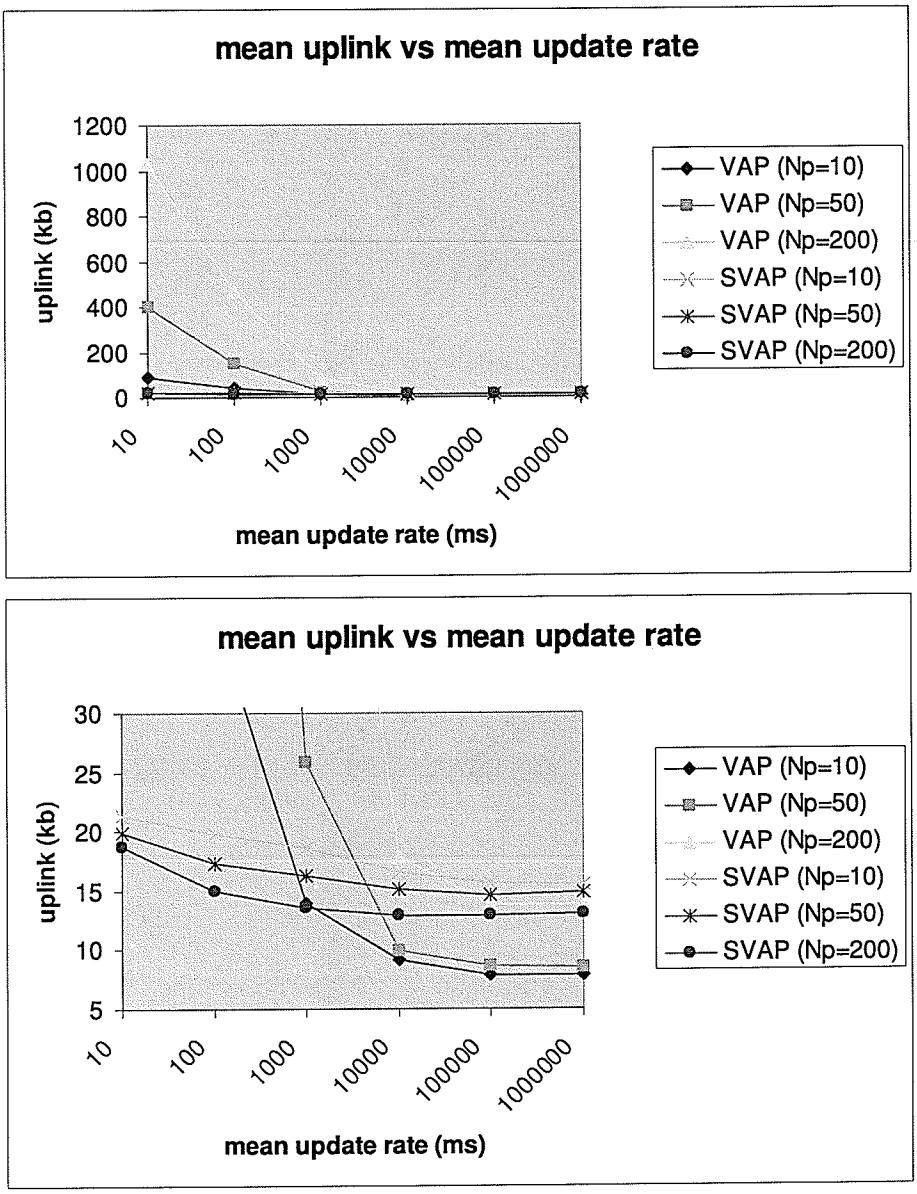


Figure 17 – Uplink totals under varying update arrival time

When the mean update rate is low and there are frequent database updates occurring at the server, the VAP $N_p = 50$ and $N_p = 200$ configurations have significantly higher data uplink requirements than SVAP, as shown in Figure 17. This is due to the fact that these schemes must aggressively pre-fetch the highest *d-value* items from the server, and these pre-fetches require uplink data requests. As the mean update rate increases, fewer pre-

fetches are required in both schemes thereby reducing the uplink requirements of the two schemes.

As the mean update time increases the VAP scheme requires less uplink bandwidth than the SVAP scheme. This can be explained by the fact that when the mean update rate is high, the mobile unit's cache will be saturated with the highest *d-value* items in the database. This means that fewer pre-fetches will be required and therefore the uplink requirements are lower. While this same optimization applies in the SVAP scheme, the SVAP client will still be sending the `send_energy_remaining` notification to the server. As discussed in Section 4.3, this message is sent to the server so the it will be able to decide how much information to send to the mobile unit, given the current power levels at the client. While small, these messages do have an impact over the course of the simulation.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean update rate	5	5	436.1	98.8292	<.0001
Type	1	1	36233.3	41053.85	0.0000
Np	2	2	52.6	29.7840	<.0001
mean update rate*Type*Np	10	10	2818444.6	319341.8	0.0000
Type*Np	2	2	3470629.1	1966186	0.0000
mean update rate*Type	5	5	35731.8	8097.131	0.0000
mean update rate*Np	10	10	54.8	6.2041	<.0001

Table 8 – Effect table for mean uplink under varying update arrival rates

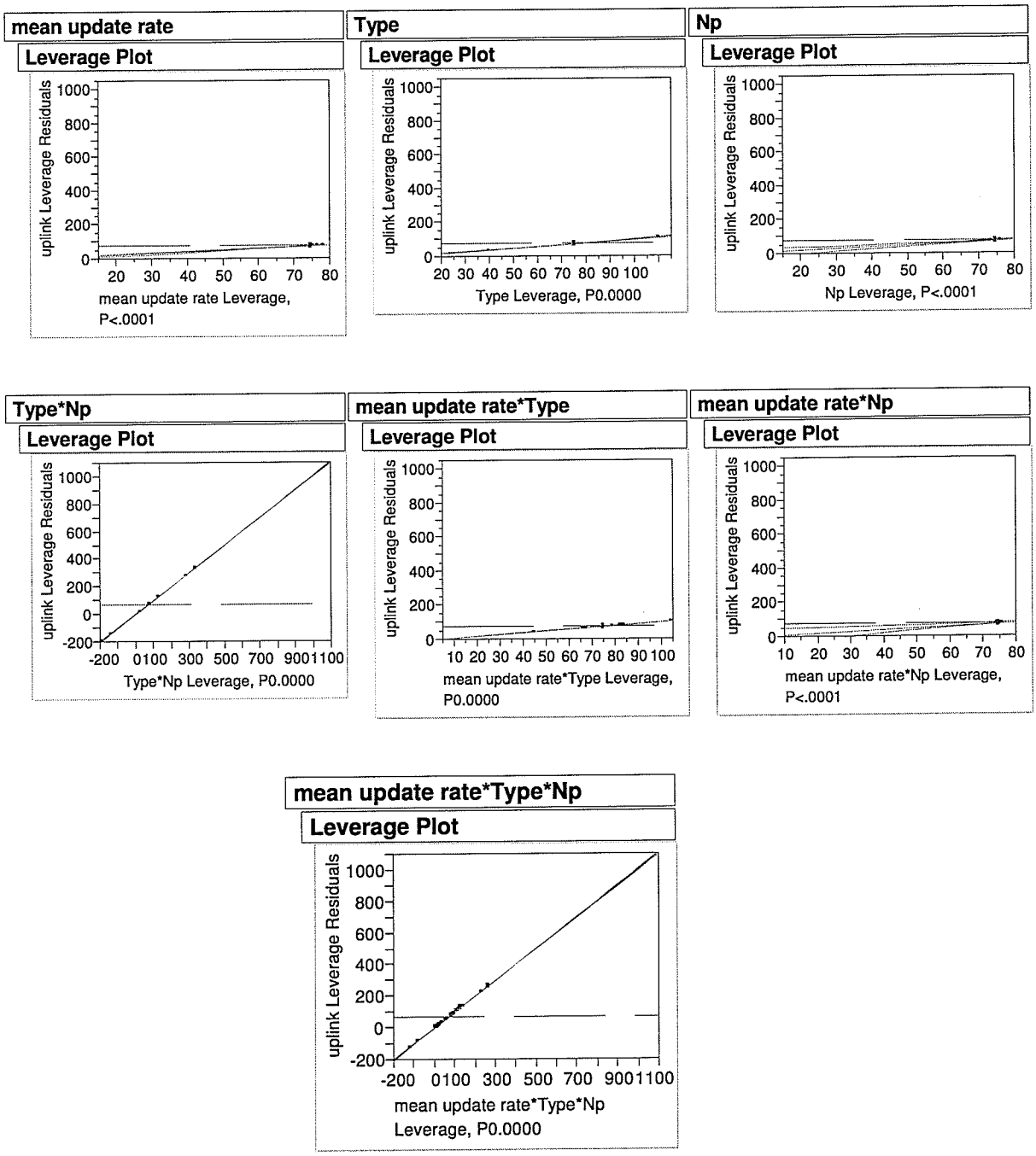


Figure 18 – Profile plots for mean uplink under varying update arrival rates

A two-way analysis of variance (ANOVA) test was performed, this time to compare the mean uplink totals of SVAP and VAP under varying update arrival rates. There is

strong evidence that the mean uplink totals for SVAP are different than VAP with a probability P>0.9999 as shown by the Type effect. SVAP performs better under lower mean update arrival rates, while VAP performs better when the mean update arrival rates are higher – i.e. when updates are less frequent. The effect table and profile plots are shown in Table 8 and Figure 18, respectively.

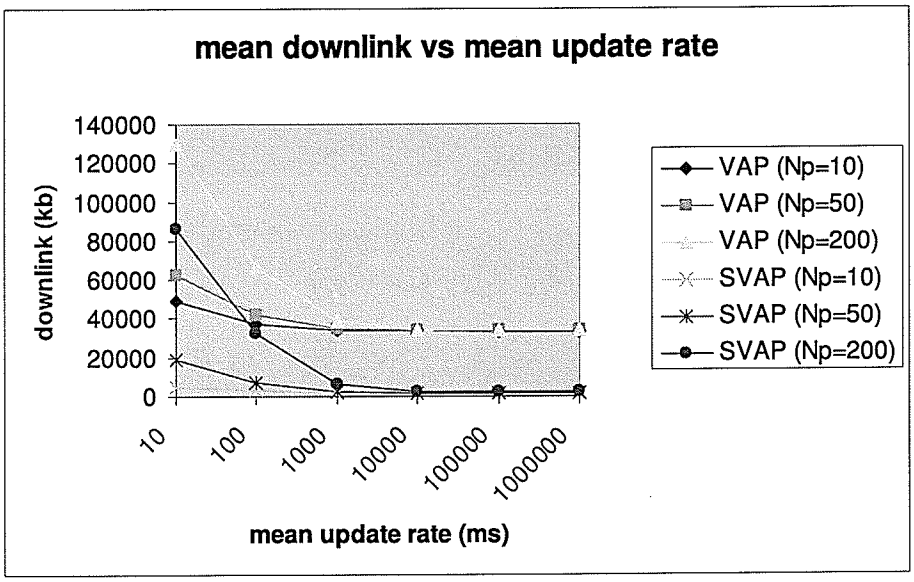


Figure 19 – Downlink totals under varying update arrival time

Figure 19 shows the downlink bandwidth utilization in kb for a particular mobile unit. In the SVAP scheme, the IR and UIR are tailored to a specific mobile unit whereas VAP only requires one invalidation report and UIR for all mobile units in its geographic area. Obviously, a direct comparison of totals from the server's perspective would show that VAP far outperforms SVAP because the VAP server need send only one invalidation report whereas the SVAP server needs to send one for each of the 100 clients. To get a more accurate picture, Figure 19 shows the average downlink totals from the perspective of the mobile units. In this way we focus on evaluating the performance of VAP and

SVAP from the perspective of conserving power at the mobile unit. From the perspective of the server, energy consumption is likely not an issue when broadcasting one IR per client, however, the larger amount of bandwidth consumed is a trade-off to achieve performance at the client level.

SVAP consistently requires less downlink bandwidth than VAP in all scenarios. This is due to the fact that the IR and UIR are tailored to the specific requirements of the mobile unit, given that the server knows the contents of the mobile unit's cache. The IR/UIR does not need to contain information about updated data items that are not part of the mobile unit's cache, thus conserving bandwidth over the VAP approach.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean update rate	5	5	102020334	1248.982	<.0001
Type	1	1	1.47432e10	902468.7	0.0000
Np	2	2	5.77351e10	1767051	0.0000
mean update rate*Type*Np	10	10	2358465.66	14.4367	<.0001
Type*Np	2	2	3053802.26	93.4653	<.0001
mean update rate*Type	5	5	905422835	11084.62	0.0000
mean update rate*Np	10	10	4.5554e+10	278846.6	0.0000

Table 9– Effect table for mean downlink under varying update arrival rates

A two-way analysis of variance (ANOVA) test was performed to compare the mean downlink totals of SVAP and VAP under varying update arrival rates. There is, again, strong evidence that the result are significant. In this case, that the mean downlink totals for SVAP are smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 9 and Figure 20, respectively.

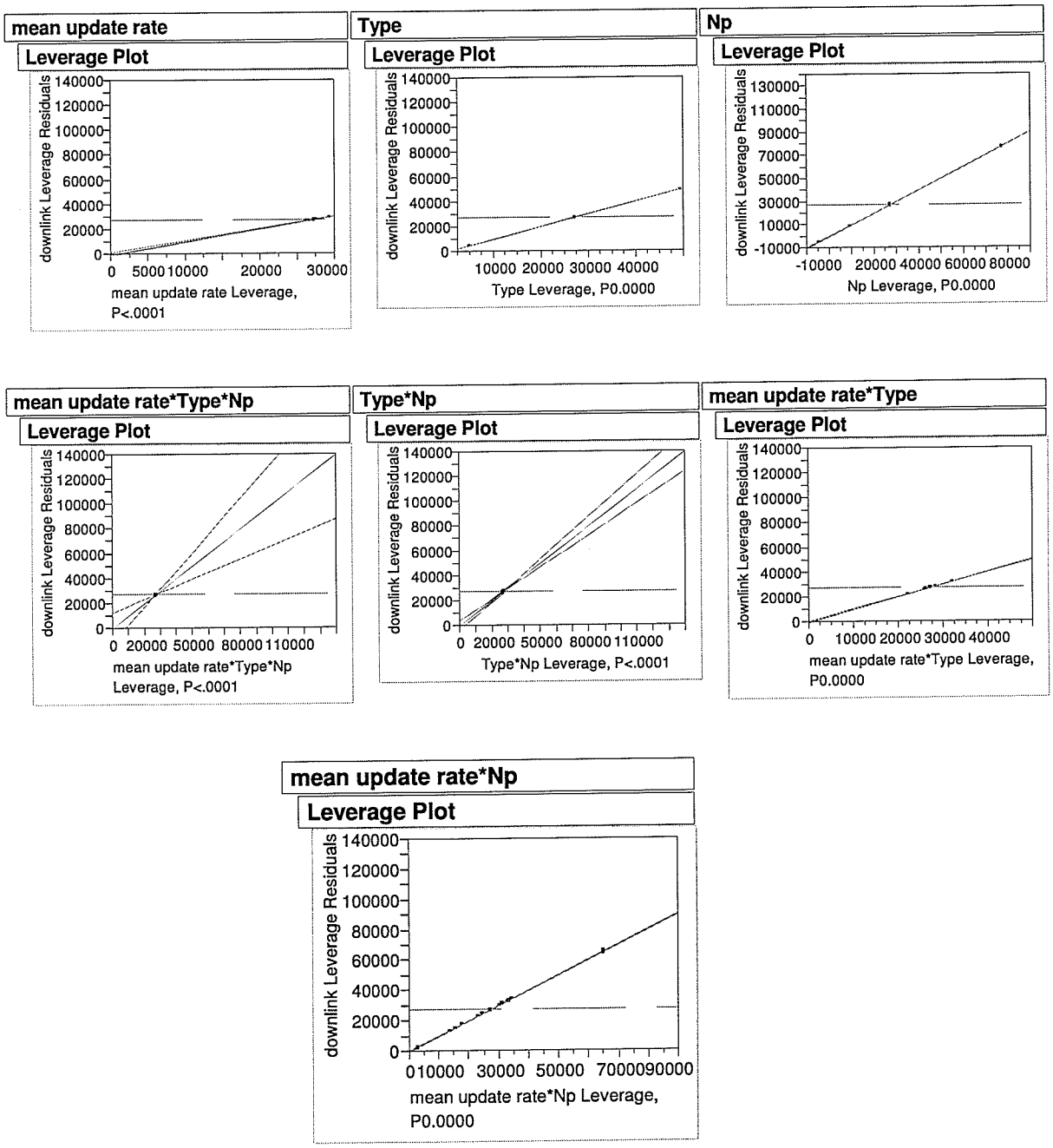


Figure 20 – Profile plots for mean downlink under varying update arrival rates

5.3. Varying the Cache Size

This experiment compared the effect of cache size on VAP and SVAP. The number of pre-fetches has been set at a constant $N_p = 50$ and the mean update arrival time interval has been set to 10,000 seconds. The specific variables for this experiment are shown in Table 10.

Simulation Parameter	Value
Simulation length	10000s
Mean update arrival time	100s
Mean query generate time	100s
Number of pre-fetches (N_p)	50
Number of trials	15

Table 10 – Varying cache size simulation variables

This experiment replicates that of [Yin et al., 2002]'s Figure 2 (Performance as a function of the cache size) parts (a), (b) and (c).

While [Yin et al., 2002]'s work shows a continuously decreasing trend in terms of energy consumed per query as a function of increasing cache size, the reproduced VAP simulation plateaus when the cache size hits 20% of the database size (see Figure 21). This difference can be explained by the fact that the database's hot data set is 20% of the database. When the size of the cache nears 20% of the database, it is likely that most of the hot data set has been cached at the mobile unit. When the size of the cache increases beyond 20%, the mobile unit will start to cache the database's cold data set items – the other 80% of the database. The benefit of caching the cold data set is not worth the effort and energy expended to pre-fetch the cold data items from the server because they will most likely not be requested by the user. As well, given the size distribution of the database, the cold data set items tend to be larger and more expensive to pre-fetch from the server. The caching of the cold dataset items manifests itself in the plateau of energy

consumed per query when the cache size hits 20% of the database. SVAP also displays this plateau.

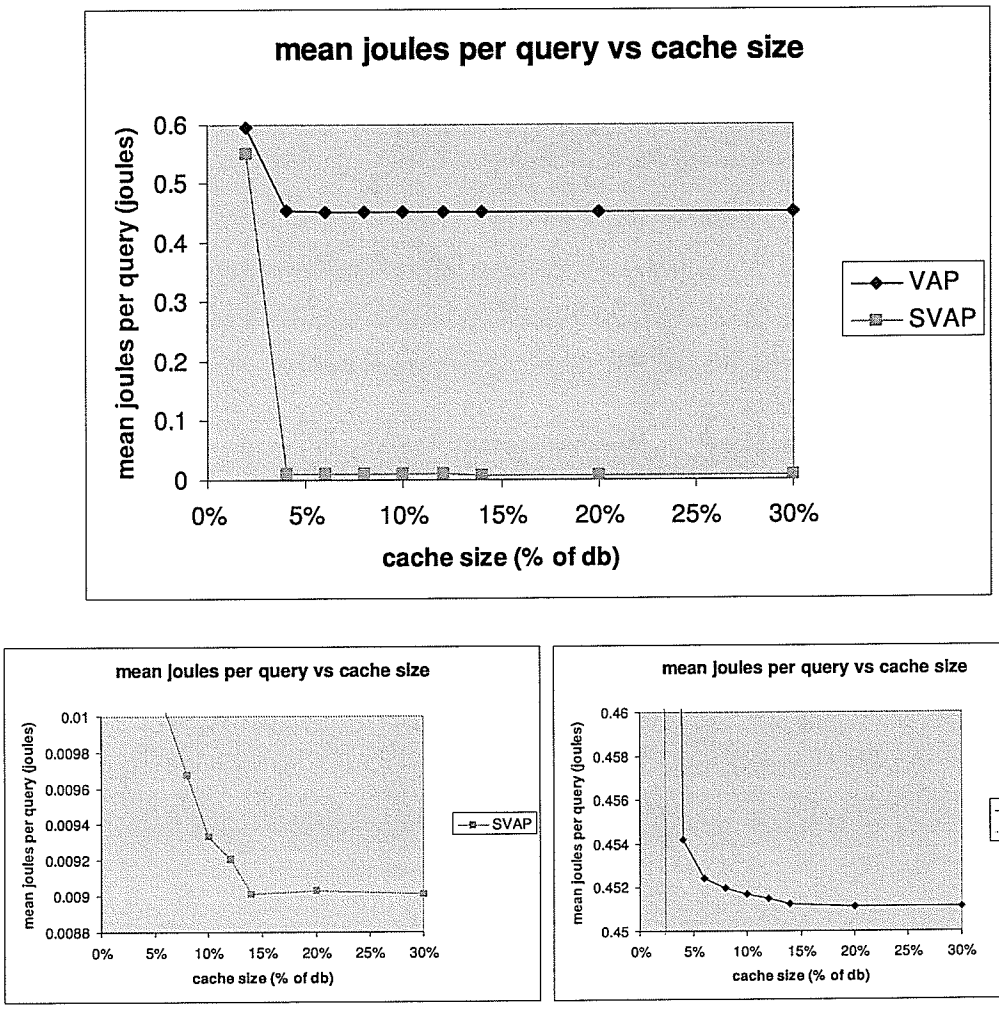


Figure 21 – Energy consumption as a function of cache size using three different y-axis scales

The results show that SVAP consistently uses a significantly smaller amount of energy per query than the VAP scheme.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
cache size	8	8	3.8847674	4981500	0.0000
Type	1	1	0.0147270	151077.4	0.0000
cache size*Type	8	8	1.0560264	1354160	0.0000

Table 11 – Effect table for mean joules per query as a function of cache size

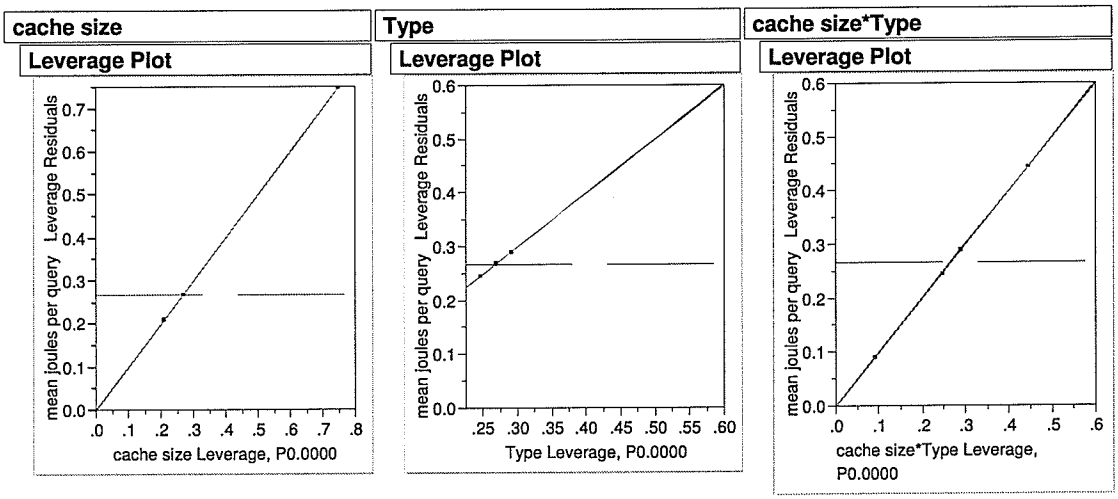
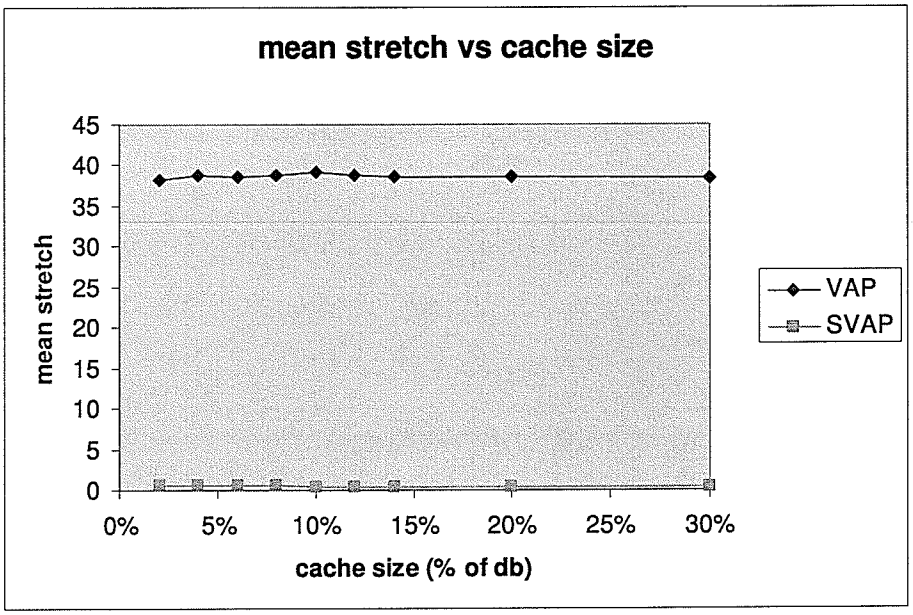


Figure 22 – Profile plots for mean joules per query as a function of cache size

A two-way analysis of variance (ANOVA) test was performed to compare the mean joules per query of SVAP and VAP under varying cache sizes. There is strong evidence that the mean joules per query for SVAP are smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 11 and Figure 22, respectively.



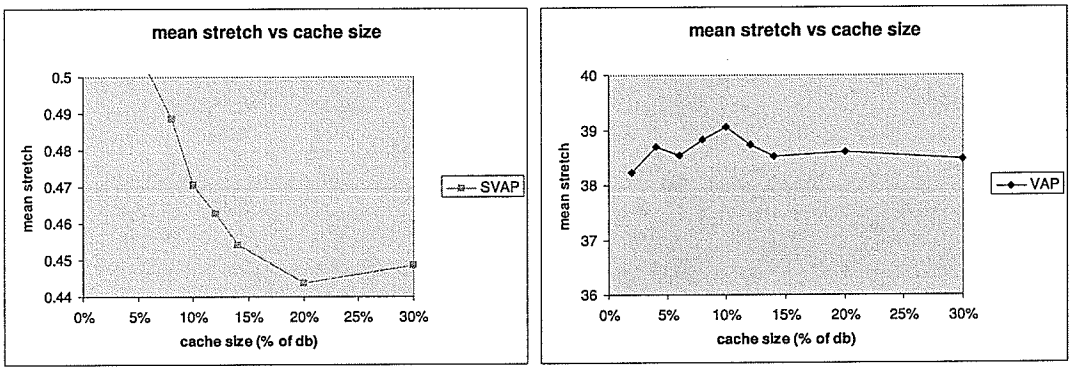


Figure 23 – Mean stretch as a function of cache size using three different y-axis scales

As Figure 23 shows, the average stretch as a function of cache size follows the same trend as the energy consumed per query. When the size of the cache hits 20% of the size of the database, the average stretch increases slightly for SVAP and plateaus for VAP. This can be explained by the same reasoning described for the trends of the energy consumed per query. The increase, in the case of SVAP, is due to the extra downloading required of the cold data set items that will likely not be used by the client. In any case, SVAP has a significantly lower mean stretch than VAP in all cases.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
cache size	8	8	0.167	0.0152	1.0000
Type	1	1	10649.356	7738.022	<.0001
cache size*Type	8	8	3.774	0.3428	0.9485

Table 12 – Effect table for mean stretch as a function of cache size

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
cache size	8	8	3.102	0.2875	0.9697
Type	1	1	98231.367	72849.99	0.0000

Table 13 – Effect table for mean stretch as a function of cache size without interaction effects

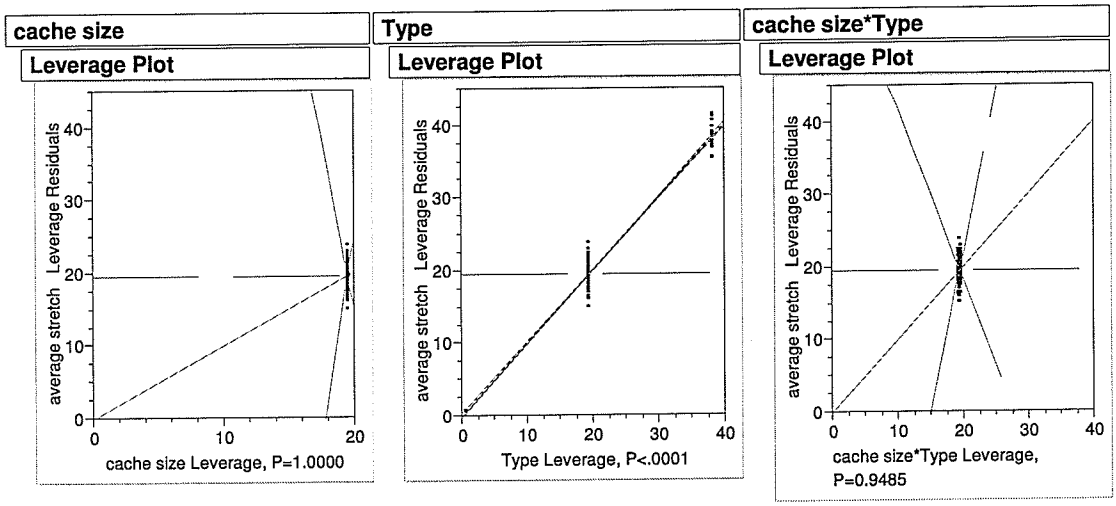


Figure 24 – Profile plots for mean stretch as a function of cache size

A two-way analysis of variance (ANOVA) test was performed to compare the mean stretch of SVAP and VAP under varying cache sizes. There is strong evidence that the mean stretch for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect in Table 13. Since there are no interaction effects it can also be said that the mean stretch of SVAP is 38.15 stretch units lower than VAP's with a standard error of

0.14 stretch units. The effect tables are shown in Table 12 and Table 13, while the profile plots are shown by Figure 24.

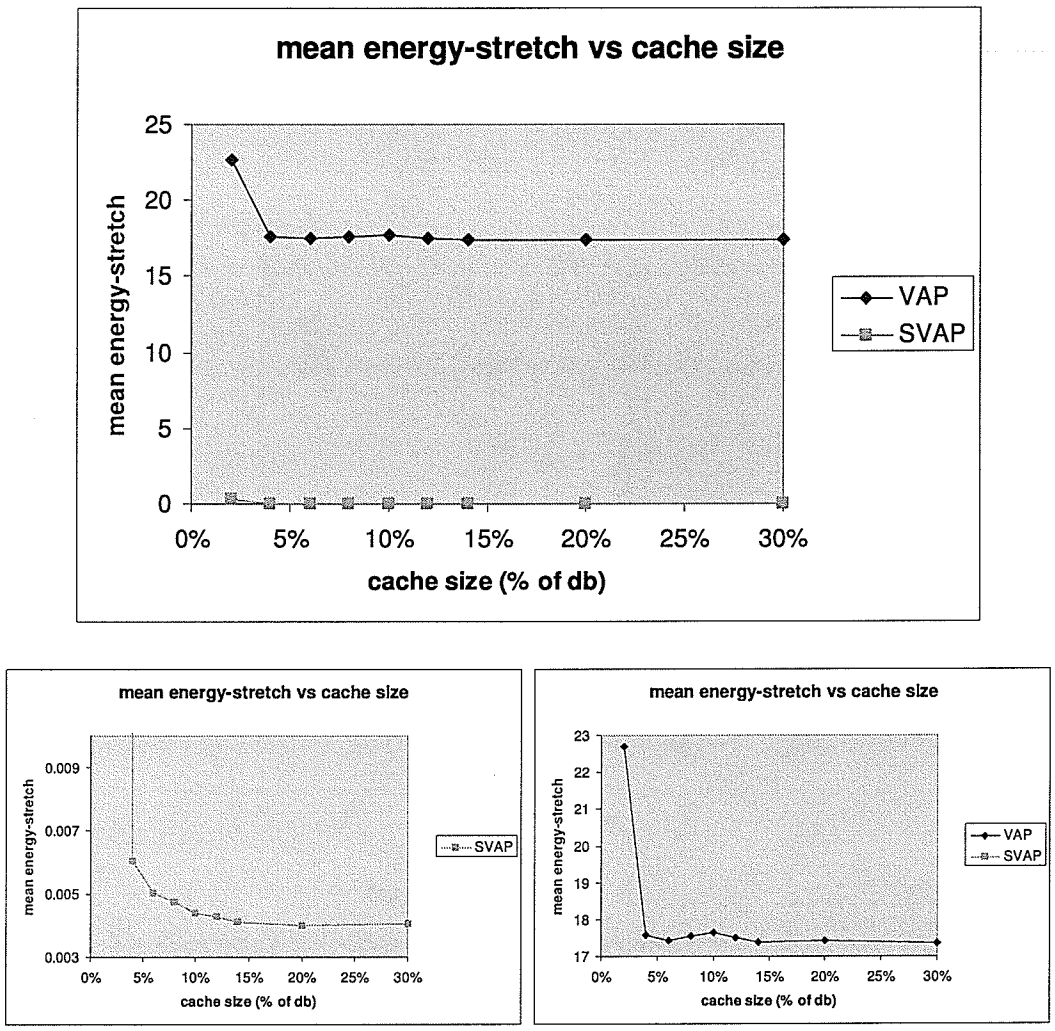


Figure 25 – Mean energy-stretch as a function of cache size using three different y-axis scales

SVAP consistently uses less energy-stretch than VAP as shown in Figure 25. As described earlier, this is due to the optimization of the IR and UIR content for each mobile unit. This allows the mobile unit to conserve energy by not having to download unwanted information.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
cache size	8	8	1.1843	0.4763	0.8724
Type	1	1	3761.6641	12102.5	<.0001
cache size*Type	8	8	161.9184	65.1180	<.0001

Table 14 – Effect table for mean energy-stretch as a function of cache size

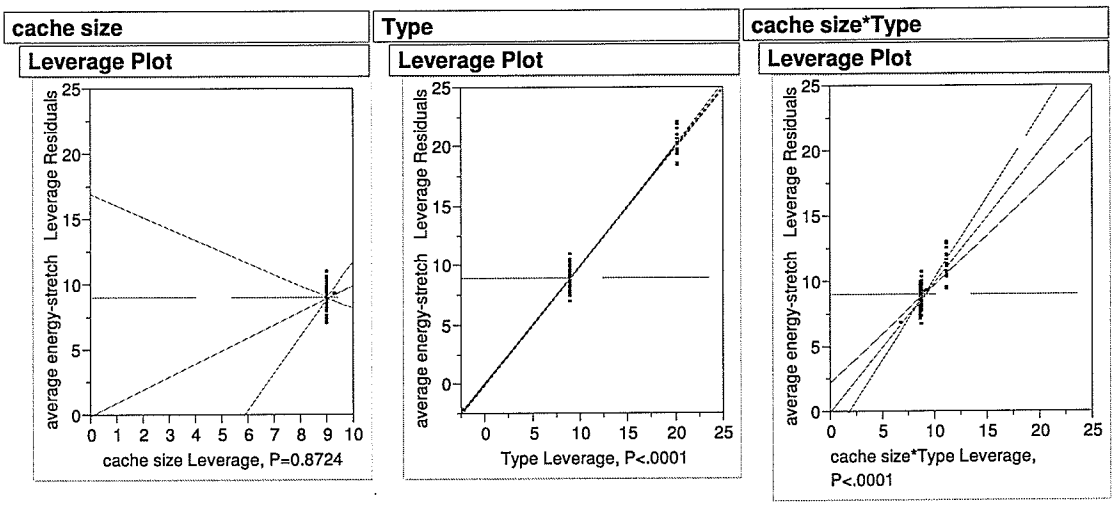


Figure 26 – Profile plots for mean energy-stretch as a function of cache size

A two-way analysis of variance (ANOVA) test was performed to compare the mean energy-stretch of SVAP and VAP under varying cache sizes. There is strong evidence that the mean energy-stretch for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 14 and Figure 26, respectively.

5.4. Varying Mean Query Generate Time

This experiment examined the effect of mean query generate time on VAP and SVAP. Mean query generate time specifies the time interval between when the mobile

unit requests data items from the server. The specific variables for this experiment are shown in Table 15.

Simulation Parameter	Value
Simulation length	10000s
Mean update arrival time	10s
Number of pre-fetches (N_p)	50
Number of trials	30

Table 15 – Varying mean query generate time simulation variables

This experiment replicates that of [Yin et al., 2002]'s Figure 3 (Energy-stretch value under varying mean query generate time).

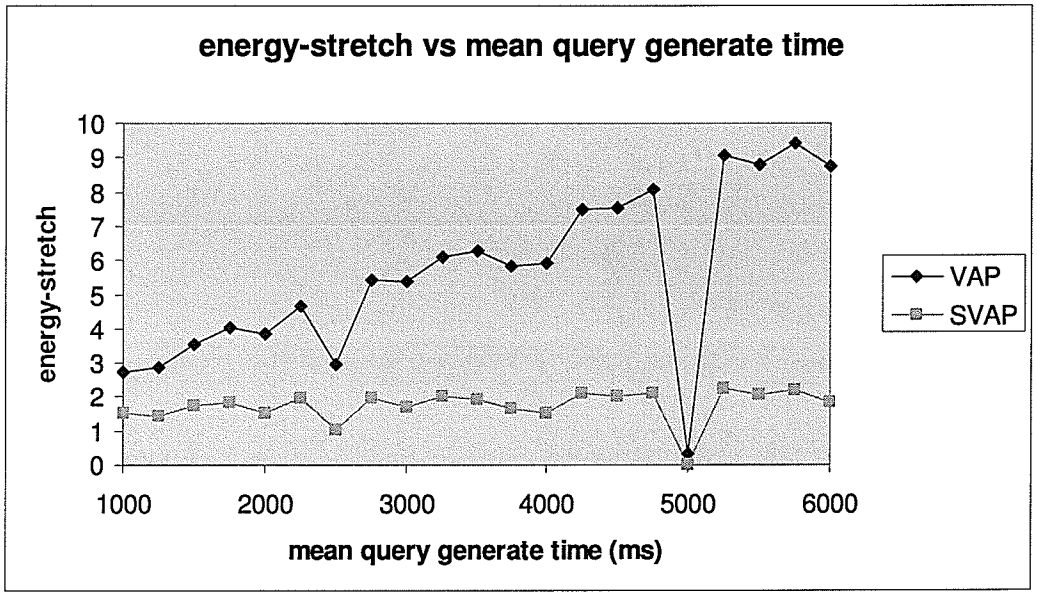


Figure 27 – Energy-stretch under varying mean query generate time

As shown in Figure 27, the energy-stretch increases as the mean query generate time increases. [Yin et al., 2002] explains that, all variables being equal, increasing the mean query generate time has the same effect as reducing the mean update arrival time at the server (as confirmed in Figure 15). An interesting attribute of the replicated VAP results

over [Yin et al., 2002]'s results are the extreme dips in energy-stretch around the 2500ms and 5000ms mark. This can be explained given the periodicity of the IR and UIR. The IR is issued every 20,000ms and the UIR is replicated 3 times within, at 5000ms, 10,000ms, and 15,000ms. The times of these dips divide evenly into the IR and UIR intervals. The effect of this manifests itself by having the mobile units issue queries in conjunction with an IR and UIR interval. Therefore, when the query interval coincides with the IR and UIR there is a very high chance that there will be a cache hit because the latest *d-values* items have just been downloaded to the client. The dip at the 5000ms mark is lower than the 2500ms mark because, in that case, the mobile unit will issue queries that coincide with twice as many IR intervals as the 2500ms mark given the same length of simulation. The IR interval has more impact that the UIR interval because the mobile unit performs its pre-fetching just after the IR arrives. This lowers the energy-stretch significantly at this point.

Although [Yin et al., 2002] explain that increasing the mean query generate time has the same effect as reducing the mean update arrival time, this isn't entirely true given the trend dips (i.e. performance hot spots) shown above. Even though these dips may be artificial given the periodicity by which queries are generated by the simulator, it is still an interesting phenomenon that should be considered when implementing this scheme. For instance, applications that require periodic updates of large amounts of data should have their queries batched transmitted by the subsystem at an interval that divides evenly into the UIR and IR interval.

SVAP consistently outperforms VAP in terms of energy consumption as a function of the mean query generate time. In addition, it is shown that queries should be issued, whenever possible, at intervals that coincide with the delivery of the IR and UIR.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
mean query generate time	21	21	146.8327	132.6351	<.0001
Type	1	1	22.6654	429.9498	<.0001
mean query generate time *Type	21	21	1451.6803	1311.313	0.0000

Table 16 – Effect table for mean energy-stretch as a function of mean query generate time

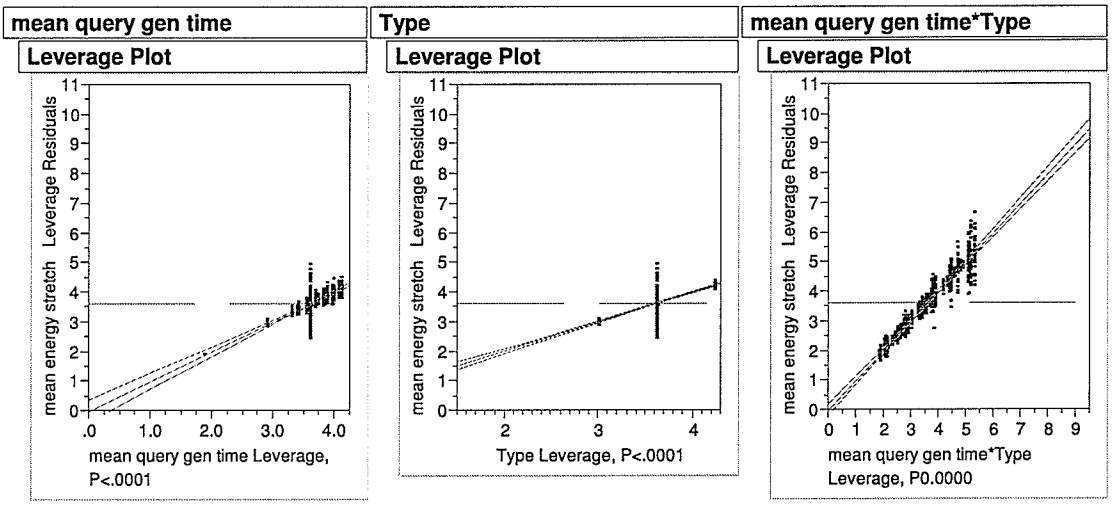


Figure 28 – Profile plots for mean energy-stretch as a function of mean query generate time

A two-way analysis of variance (ANOVA) test was performed to compare the mean energy-stretch of SVAP and VAP under varying mean query generate times. There is strong evidence that the mean energy-stretch for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 16 and Figure 28, respectively.

5.5. Energy Level and Performance

This experiment examined the performance of the mobile unit when pre-fetching adaptation is turned on. The number of pre-fetches in the case of VAP and the number of anticipated pre-fetch data items in the case of SVAP, are determined by the threshold function shown by Equation 5. Each query transmitted and received dissipates the power at the mobile unit. As the power of the mobile unit decreases, the aggressiveness of the pre-fetching will be tempered in an attempt to prolong the life of the battery powered mobile unit.

Since the replicated VAP scheme does not consider disconnections of the mobile unit from the base station, the replicated VAP server does not broadcast $w * L$ server updates to the clients, instead only L server updates are sent to the mobile units (refer to Section 3.1 for a description of w and L). In this way, the replicated VAP invalidation report is smaller in size than the VAP invalidation report produced in [Yin et al., 2002]. Since the VAP invalidation report is smaller in size, it takes longer to reduce the energy at the mobile unit than the VAP scheme in [Yin et al., 2002]. Given this, the initial joules at the mobile unit has been decreased from 5000 joules, as used by [Yin et al., 2002], to 2600 joules to allow the simulation to complete in a shorter, more reasonable, time. Even doing this, it took 24 hours running trials on 5 different 500 MHz machines at once to generate enough representative data points to achieve statistical significance.

Simulation Parameter	Value
Simulation length	300000s
Number of pre-fetches (N_p)	50
Mobile unit initial joules	2600 joules
Transmit dissipation	1.5 watts
Receive dissipation	0.6 watts
Mean update arrival time	1000ms
Mean query generate time	1000ms
Number of trials	12

Table 17 – Energy level and performance simulation variables

This experiment replicates that of [Yin et al., 2002]'s Figure 4 (The energy level and system performance as a function of time) parts (a), (b) and (c).

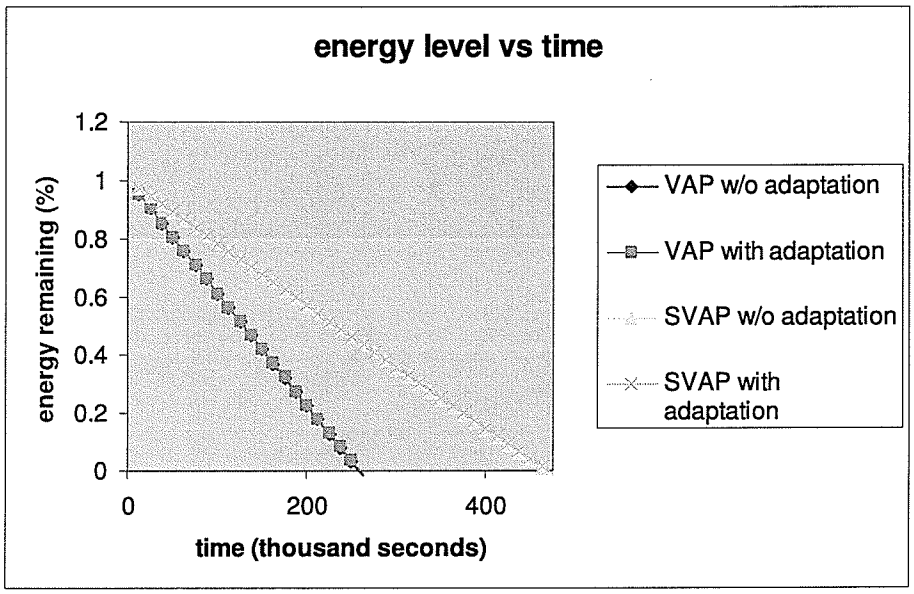


Figure 29 – Energy level as a function of time

Figure 29 shows that the SVAP client uses significantly less power than the VAP client and therefore allows the client to have a significantly longer life. SVAP has approximately a 55% longer lifespan than VAP. Figure 29 seems to show that adaptation has relatively little effect on the lifespan of the mobile unit, however by examining the numeric results of the simulation, VAP with adaptation has an 18 minute longer lifespan

than VAP without adaptation. SVAP with adaptation has a 61 minute longer lifespan than SVAP without adaptation.

The replicated VAP simulation differs from [Yin et al., 2002]'s result which shows that VAP without adaptation and VAP with adaptation diverge at the 70% level rather than at the 50% level. However, this is most likely an error in their simulation given that the first step in the threshold function occurs at 50%. The replicated simulation does not display this error and correctly shows the divergence of VAP without adaptation and VAP with adaptation at the 50% mark.

A two-way analysis of variance (ANOVA) test was performed to compare the percentage of client energy remaining between SVAP and VAP over time. There is strong evidence that the percentage of client energy remaining for SVAP is larger than VAP given the same unit of time with a probability $P>0.9999$ as shown by the Type effect. The effect table and profile plots are shown by Figure 30 and Table 18, respectively.

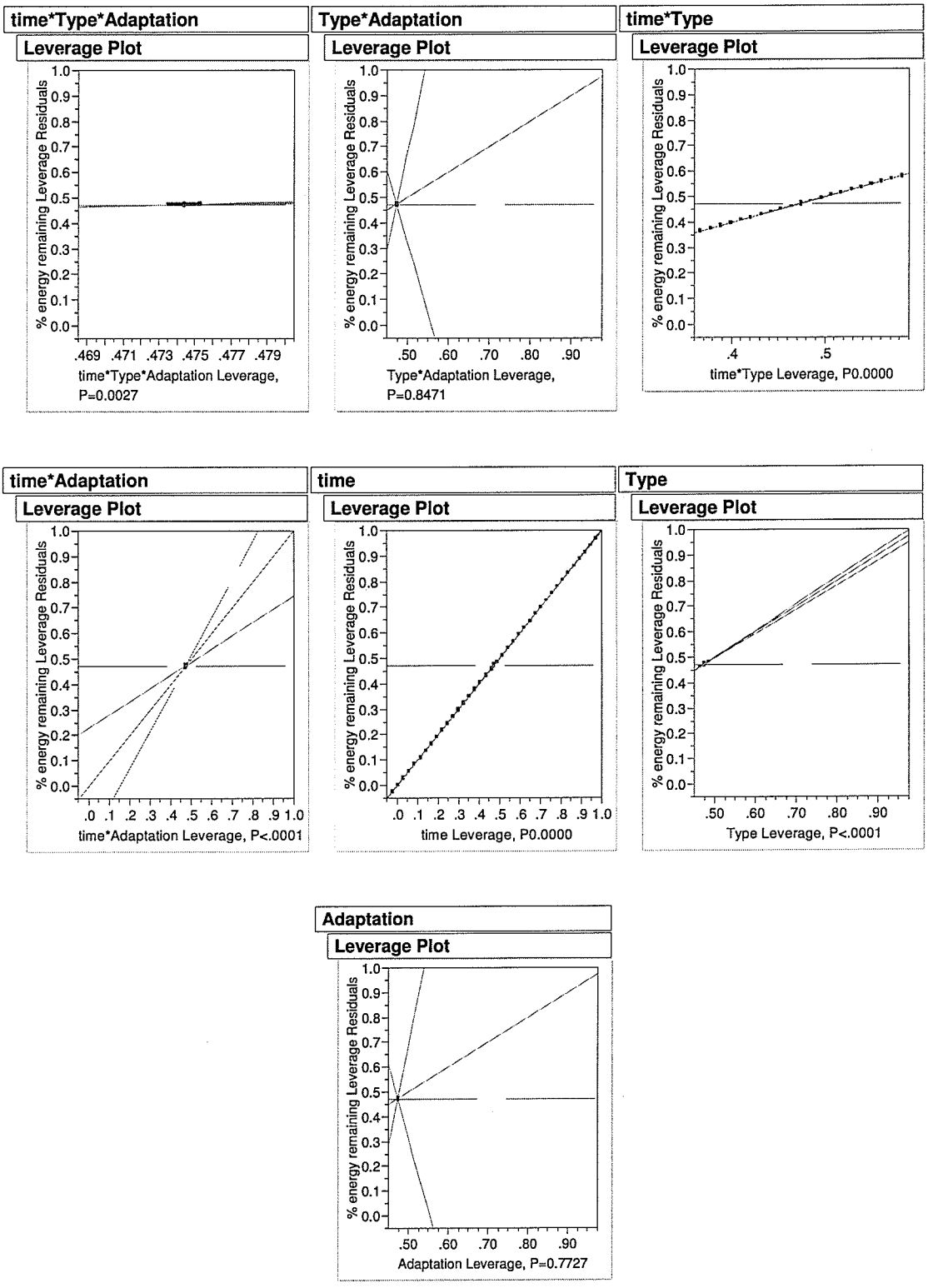


Figure 30 – Profile plots for energy level as a function of time

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F	
time*Type*Adaptation	37	20	0.000112	2.1408	0.0027	LostDFs
Type*Adaptation	1	1	0.000000	0.0372	0.8471	
time*Type	37	20	1.251224	23810.99	0.0000	LostDFs
time*Adaptation	37	37	0.000666	6.8461	<.0001	
time	37	37	22.943236	236007.2	0.0000	
Type	1	1	0.001590	605.0902	<.0001	
Adaptation	1	1	0.000000	0.0835	0.7727	

Table 18 – Effect table for energy level as a function of time

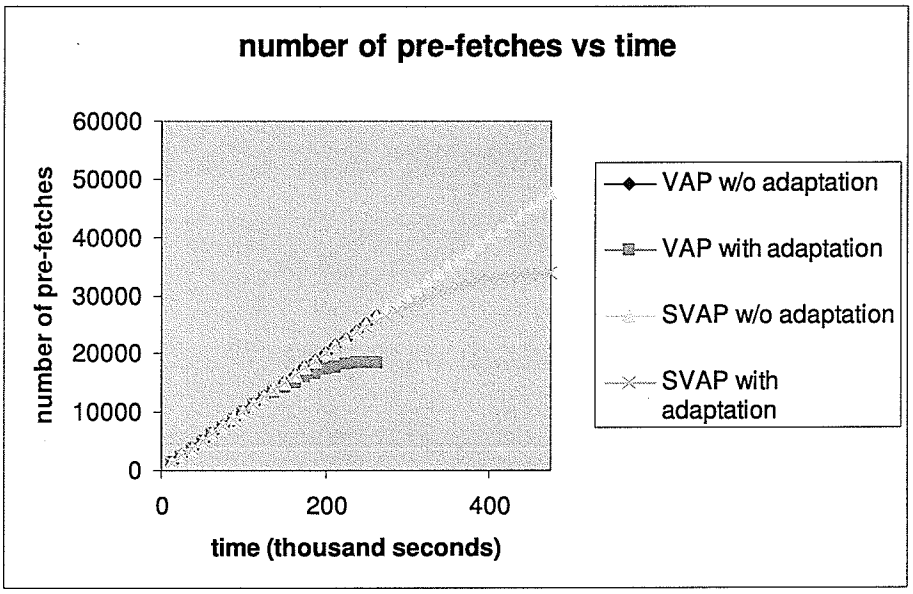


Figure 31 – Number of pre-fetches as a function of time

Figure 31 shows that the rate of pre-fetching tapers off in the schemes that use adaptation once the power thresholds are reached. This reduction in pre-fetching enables the mobile unit to conserve more energy.

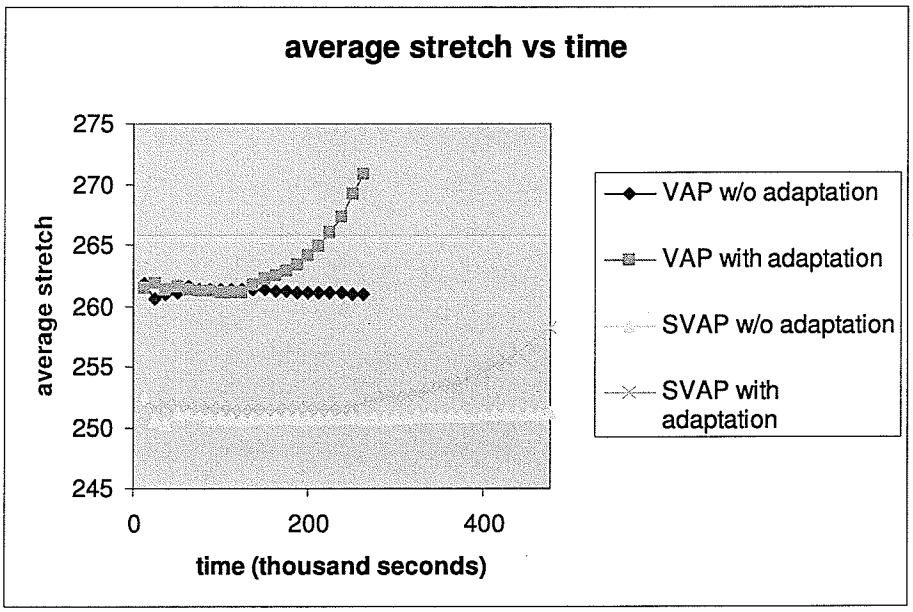


Figure 32 – Average stretch as a function of time

The reduction in pre-fetching comes at the cost of higher average stretch as illustrated in Figure 32. Once the thresholds have been reached, the VAP scheme with adaptation trends toward higher average stretch. Note that SVAP does not follow this trend. Interestingly, SVAP with adaptation has a mean response time of approximately 877 ms, while SVAP without adaptation has a mean response time of approximately 850 ms. This means that queries are taking longer in the SVAP with adaptation scheme. Since stretch is a function of response time and service time, it follows that since response time is significantly different, the service time must be offsetting this difference, resulting in a similar average stretch. Therefore, SVAP with adaptation is more frequently requesting large data items from the server than SVAP without adaptation, likely due to the increased cache misses in the SVAP with adaptation scheme. This is not a factor in the VAP scheme, likely because the trends of VAP with and without adaptation diverge for less time than the trends for SVAP. The longer divergence in SVAP causes a significantly larger number of cache misses. The mean actual values from the trials show

that VAP without adaptation has approximately 451 more cache hits than VAP with adaptation, while SVAP without adaptation has 6943 more cache hits than SVAP with adaptation.

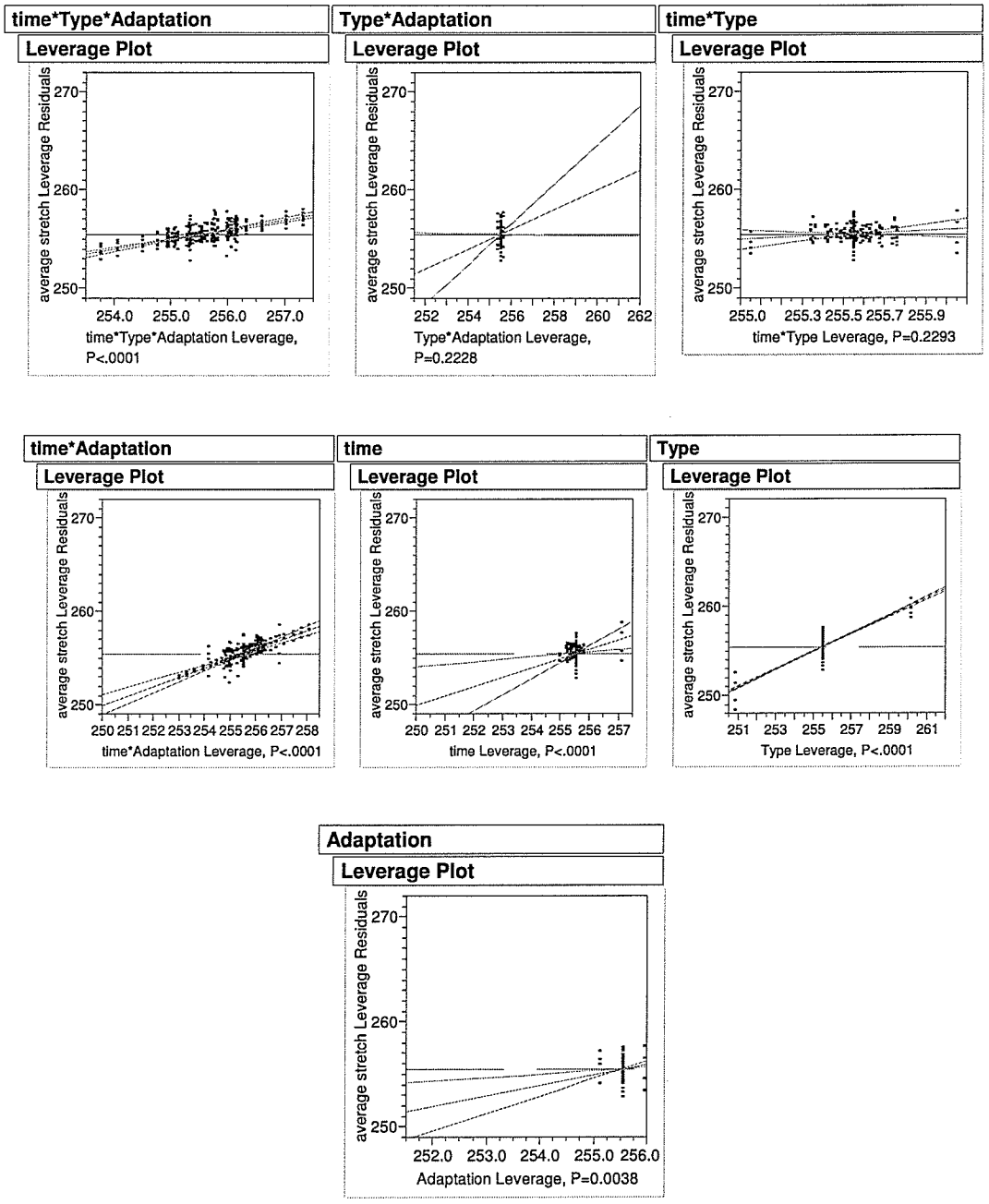


Figure 33 – Profile plots for average stretch as a function of time

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F	
time*Type*Adaptation	37	20	285.73446	49.5563	<.0001	LostDFs
Type*Adaptation	1	1	0.42915	1.4886	0.2228	
time*Type	37	20	7.03816	1.2207	0.2293	LostDFs
time*Adaptation	37	37	428.97137	40.2155	<.0001	
time	37	37	27.31675	2.5609	<.0001	
Type	1	1	303.76425	1053.667	<.0001	
Adaptation	1	1	2.42512	8.4120	0.0038	

Table 19 – Effect table for average stretch as a function of time

A two-way analysis of variance (ANOVA) test was performed to compare the average stretch between SVAP and VAP over time. There is strong evidence that the average stretch for SVAP is smaller than VAP given the same unit of time, with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Figure 33 and Table 19, respectively.

5.6. Changing Access Patterns

This experiment measured the effect of a drifting hot data set (due to changing user access patterns) on the VAP and SVAP schemes. Recall that the hot data set is that part of the database that is accessed 80% of the time and comprises 20% of the data items in the database. Let the hot data set drift 1 data item every δ invalidation reports, where δ represents the mean access pattern change interval, delta. The simulation was altered to add a variable Δ which tracks the current offset of the hot data set. The variable Δ is initialized to 0 and incremented by 1 every δ invalidation reports. The variable Δ is used as an offset in determining what data item is requested by the mobile unit. For example, if $\Delta = 4$ and the client determines they wish to access data item 3, the offset will be applied and the mobile unit will request data item 7. This offset is also applied to the data item updates at the server so that the access and update patterns are consistent.

Simulation Parameter	Value
Simulation length	10000s
Mean update arrival time	100s
Number of pre-fetches (N_p)	50
Number of trials	60

Table 20 – Changing access patterns simulation variables

This experiment replicates that of [Yin et al., 2002]'s Figure 5 (The performance when the access pattern changes), parts (a), (b) and (c).

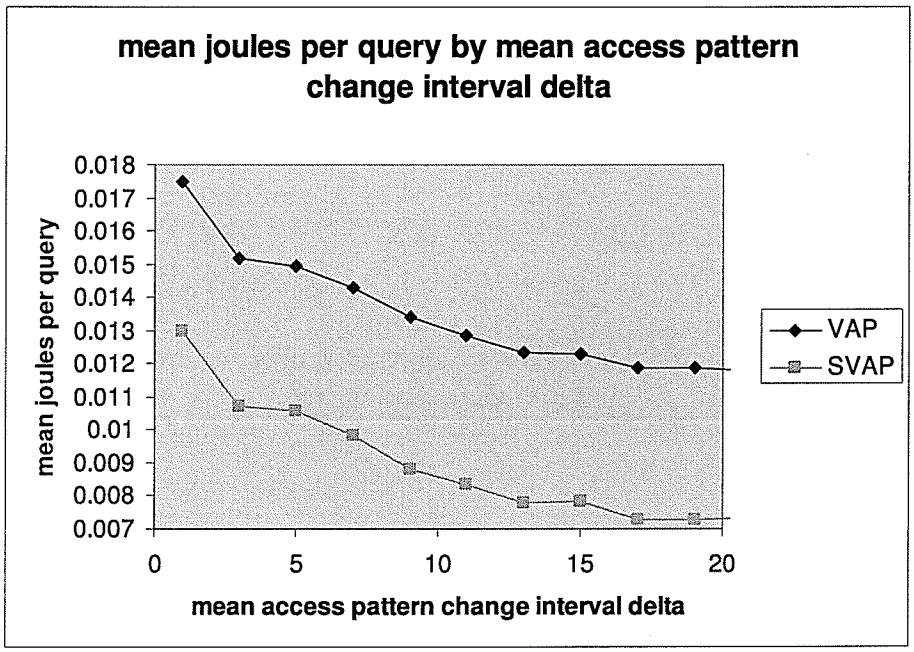


Figure 34 – Energy consumption as a function of mean access pattern change interval delta

As the mean access pattern change interval delta (δ) increases, a smaller hot data set drift occurs and both the VAP and SVAP schemes show a decrease in the energy consumed per query (see Figure 34). When δ is small, more cache misses occur and the energy consumed per query is higher. Note that SVAP consistently outperforms VAP in terms of energy consumption as δ increases.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Type	1	1	0.00060211	234.7531	<.0001
mean access pattern change	10	10	0.00203962	79.5220	<.0001
mean access pattern change*Type	10	10	0.00000141	0.0550	1.0000

Table 21 – Effect table for energy consumption as a function of mean access pattern change interval delta

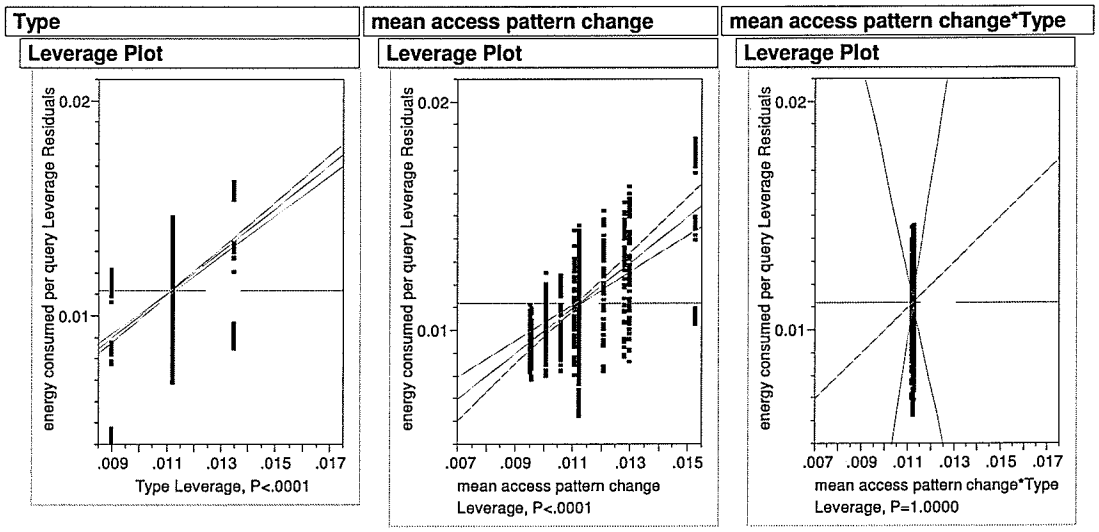


Figure 35 – Profile plots for energy consumption as a function of mean access pattern change interval delta

A two-way analysis of variance (ANOVA) test was performed to compare the mean joules consumed per query of SVAP and VAP under varying mean access pattern change interval delta. There is strong evidence that the mean joules consumed per query for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 21 and Figure 35, respectively.

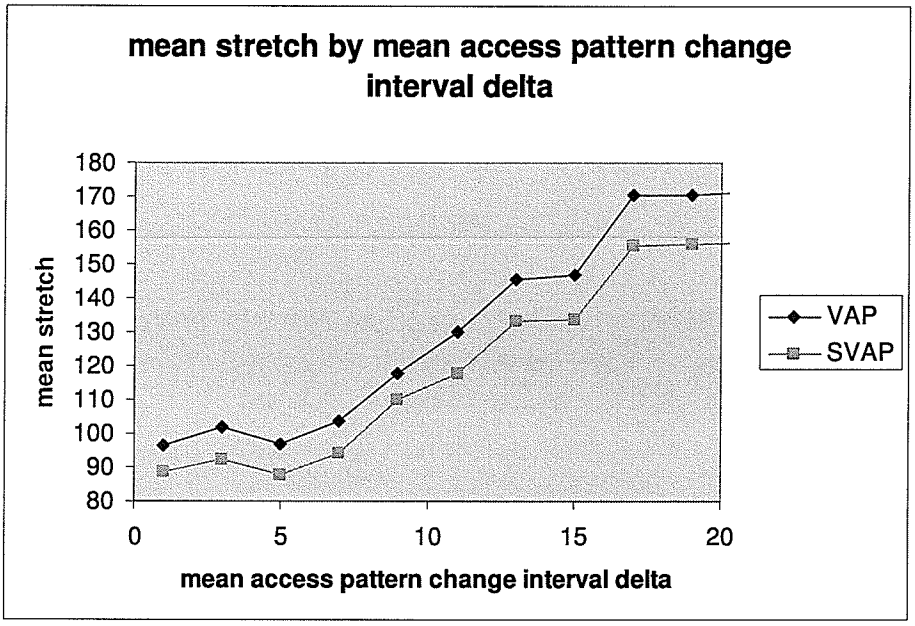


Figure 36 – Mean stretch as a function of mean access pattern change interval delta

The average stretch of SVAP and VAP follow a similar trend as δ changes (see Figure 36).

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Type	1	1	1959.42	0.8464	0.3577
mean access pattern change	10	10	470157.26	20.3100	<.0001
mean access pattern change*Type	10	10	2262.33	0.0977	0.9998

Table 22 – Effect table for mean stretch as a function of mean access pattern change interval delta

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Type	1	1	43316.9	18.8422	<.0001
mean access pattern change	10	10	1028476.7	44.7371	<.0001

Table 23 – Effect table for mean stretch as a function of mean access pattern change interval delta without interaction effects

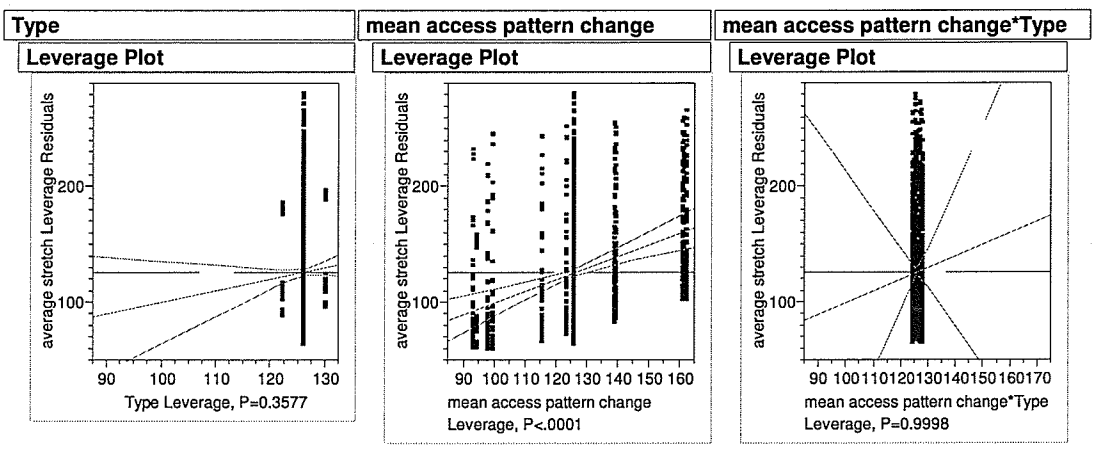


Figure 37 – Profile plots for mean stretch as a function of mean access pattern change interval delta

A two-way analysis of variance (ANOVA) test was performed to compare the mean stretch of SVAP and VAP under varying mean access pattern change interval delta. Since there are no interaction effects (Table 22), it can be said that there is strong evidence that the mean stretch for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect in Table 23. It can also be said that the mean stretch of SVAP is 11.46 stretch units lower than VAP's with a standard error of 2.64 stretch units. The effect tables in Table 22 and Table 23, while the profile plots are shown by Figure 37.

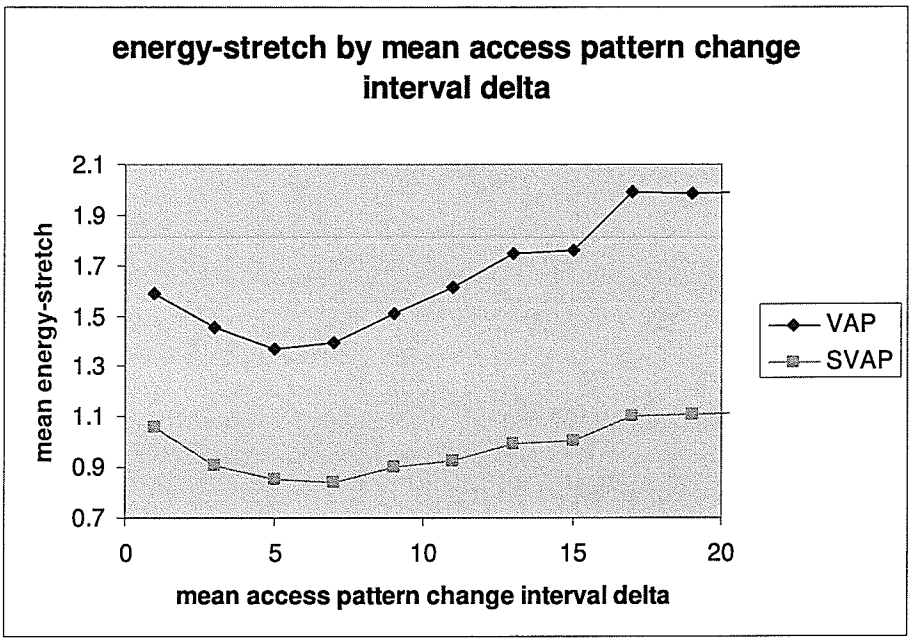


Figure 38 – Energy-stretch as a function of mean access pattern change interval delta

As expected, Figure 38 shows that SVAP consistently has a smaller energy-stretch than VAP as δ changes.

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Type	1	1	8.4600929	67.2635	<.0001
mean access pattern change	10	10	6.2856870	4.9976	<.0001
mean access pattern change*Type	10	10	6.6133137	5.2580	<.0001

Table 24 – Effect table for mean energy-stretch as a function of mean access pattern change interval delta

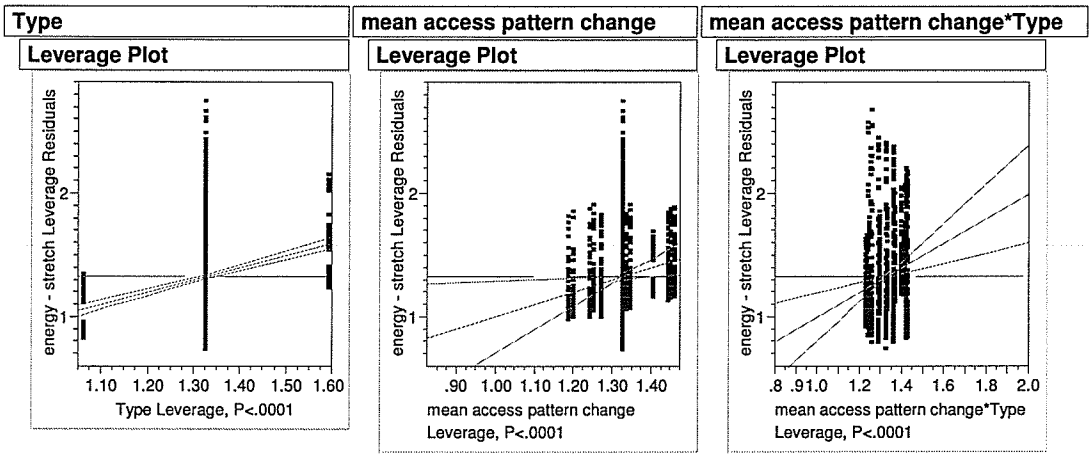


Figure 39 – Profile plots for mean energy-stretch as a function of mean access pattern change interval delta

A two-way analysis of variance (ANOVA) test was also performed to compare the mean energy-stretch of SVAP and VAP under varying mean access pattern change interval delta. There is strong evidence that the mean energy-stretch for SVAP is smaller than VAP with a probability P>0.9999 as shown by the Type effect. The effect table and profile plots are shown in Table 24 and Figure 39, respectively.

6. Conclusion

This section concludes the thesis by summarizing the findings of this research and laying out a course for future work.

6.1. Summary of Results

The work done in this thesis addressed problems within the mobile computing environment. Judicious use of bandwidth facilities was considered in order to minimize uplink and downlink activity at the mobile unit, and therefore minimize energy expenditure at the mobile unit. A stateful power-aware pre-fetch scheme, called SVAP, was proposed to maximize performance while minimizing mobile unit energy expenditure and a simulation-based performance assessment of the scheme was presented.

The stated objectives of this thesis have been met. The first objective, to reproduce the simulation of [Yin et al., 2002] to provide a baseline for the improvements offered by this thesis, was met by successfully reproducing the work of [Yin et al., 2002] with the exceptions noted in Section 5. The replicated simulation of VAP was used as a baseline to compare with the SVAP strategy presented in this thesis.

The second objective was to leverage ideas provided in [Kahol et al., 2000] to improve the results of [Yin et al., 2002] by reducing the energy expenditure of the mobile unit, while providing the same service level. This objective was met by improving over VAP using ideas from [Kahol et al., 2000]. [Kahol et al., 2000] proposed a stateful server cache management scheme for mobile environments. The new scheme presented in this thesis, SVAP, minimizes the number of pre-fetch requests the mobile client must perform

by moving the determination of what should be pre-fetched by the client to the server. The server anticipates what is required by each client and sends a tailored IR and UIR to each mobile unit.

The server anticipates what is required at the client and tailors the "broadcast" information it sends for each specific mobile unit within its geographical area. The tailoring of the broadcast information also minimizes the downlink overhead, from the perspective of the mobile unit, because the invalidation reports are not as large as they are in the VAP scheme. However, this comes at the cost of increasing the overall downlink requirements, from the perspective of the server, because the server must issue an invalidation report for every mobile unit. The VAP scheme only needs to broadcast one invalidation report for all mobile units.

SVAP also minimizes the uplink and downlink requirements, as compared to VAP, and can extend the life of the mobile unit by 55% as shown in Section 5.5. SVAP, like VAP, can use pre-fetch adaptation, which allows the scheme to tailor the aggressiveness of its pre-fetching based on the battery power remaining at the client. A function is used to reduce the number of items to pre-fetch as the energy level passes certain thresholds. The results of this thesis conclude that SVAP outperforms VAP in terms of stretch, energy-stretch and power consumption, while minimizing the uplink and downlink requirements at the mobile unit.

6.2. Future Work

Given that SVAP stores a copy of the mobile unit's cache at the server, the SVAP client, like AS, can handle arbitrarily long disconnections from the base station. While no

evaluation of this SVAP feature was performed in this thesis, the possibility exists given that it meets the data requirements present in the AS scheme. This would offer an improvement over the VAP scheme, which can only handle $w * L$ seconds long disconnections. While SVAP has been compared to VAP, a possibility for future research would be to replicate [Kahol et al., 2000]'s AS scheme and compare this scheme directly to SVAP. In addition to performance comparisons, arbitrary disconnection lengths could then be verified.

Another area of potential improvement to SVAP is reducing the amount of downlink traffic from the perspective of the base station. Since the server must now send an invalidation report to each mobile unit, the downlink requirements are much more significant than those of the VAP scheme. By splitting the invalidation report into two parts this might be improved. One part of the IR would then contain the common items of interest to all mobile units and would, correspondingly, be broadcast once to all mobile units in the area. The second section of the IR would be sent once for each mobile unit and would contain data items of interest only to the mobile unit it is sent to. The common area set would most likely be made up of the hot data item set in the database.

The final area of recommended future work is to develop and compare SVAP and VAP in a real wireless environment. This thesis has developed a simulation that shows the SVAP scheme performs significantly better than VAP from the perspective of the mobile unit, yet the only way to truly test these schemes would be to actually implement them in the real world.

7. References

[Acharya and Muthukrishnan, 1998], Swarup Acharya and S. Muthukrishnan, *Scheduling on-demand broadcasts: new metrics and algorithms*, Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, ACM Press, ISBN:1-58113-035-X, p. 43-54, Dallas, Texas, United States, October 1998.

[Barbará and Imielinski, 1995], Daniel Barbará and T. Imielinski, *Sleepers and Workaholics: Caching Strategies in Mobile Environments*, VLDB Journal, vol 4(4):567-602, October 1995.

[Bender et al., 1998], Michael Bender, Soumen Chakrabarti, and S. Muthukrishnan. *Flow and stretch metrics for scheduling continuous job streams*. In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, ISBN: 0-89871-410-9, p. 270-279, San Francisco, California, United States, January 1998.

[Breslau et al., 1999], Lee Breslau, Pei Cao, Li Fan, Graham Philips and Scott Shenker, *Web Caching and Zipf-like Distributions: Evidence and Implications*, Proceedings of IEEE Infocom '99, p. 126-134, New York, NY, March 1999.

[Cai and Tan, 1999], Jun Cai and Kian-Lee Tan, *Energy-efficient selective cache invalidation*, Wireless Networks: The Journal of Mobile Communication, Computation and Information, Kluwer Academic Publishers, ISSN: 1022-0038, vol 5(6):489-502, November 1999.

[Cao, 2002a], Guohong Cao, *Proactive Power-Aware Cache Management for Mobile Computing Systems*, IEEE Transactions on Computers, vol 51(6):608-621, June 2002.

[Cao, 2002b], Guohong Cao, *On Improving the Performance of Cache Invalidation in Mobile Environments*, ACM/Kluwer Journal of Mobile Networks and Application (MONET), vol 7(4):291-303, August 2002.

[Cao, 2003], Guohong Cao, *A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments*, IEEE Transactions on Knowledge and Data Engineering, vol 15(5):608-621, September/October 2003 (a preliminary version appeared in MOBICOM'00).

[Cao, 2004], Guohong Cao, *Power-Aware Cache Management in Mobile Environments*, Handbook of Mobile Computing, CRC Press, invited, 2004.

[Ebling et al., 1994], M. Ebling, L. Mummert, and D. Steere. *Overcoming the Network Bottleneck in Mobile Computing*. In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, pages 34-36, December 1994.

[Forman and Zahorjan, 1994], George H. Forman and John Zahorjan. *The Challenges of Mobile Computing*. IEEE Computer, vol 27(4):38-47, April 1994.

[Grassi, 2000], Vincenzo Grassi, *Prefetching Policies for Energy Saving and Latency, Reduction in a Wireless Broadcast Data Delivery System*, Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, ACM Press, ISBN:1-58113-304-9, p. 77-84, Boston, Massachusetts, United States, August 2000.

[Huntsberger and Billingsley, 1981], David V. Huntsberger, Patrick Billingsley, *Elements of Statistical Inference*, Fifth Edition, Allyn and Bacon Inc, ISBN:0-205-07305-0, 1981.

[Imielinski and Badrinath, 1994], T. Imielinski and B. R. Badrinath, *Mobile Wireless Computing: Challenges in Data Management*, Communications of the ACM, vol 37(10):19-28, October 1994.

[Jing et al., 1997], J. Jing, A. K. Elmargarmid, S. Helal, and R. Alonso. *Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments*. ACM/Baltzer Journal of Mobile Networks and Applications, vol 2(2):115-127, October 1997.

[Kahol et al., 2000], A. Kahol, S. Khurana, S. K. S. Gupta, and P. K. Srimani. *An Efficient Cache Maintenance Scheme for Mobile Environment.* IEEE International Conference on Distributed Computing Systems (ICDCS), p. 530-537, Taipei, Taiwan, April 2000.

[Moore and McCabe, 1993], David S. Moore and George P. McCabe, *Introduction to the Practice of Statistics*, Second Edition, W. H. Freeman and Company, ISBN:0-7167-2250-X, 1993.

[Sall et al., 2000], John Sall, Ann Lehman, Lee Creighton, SAS Institute Inc. *JMP Start Statistics*, Second Edition, Brooks Cole, ISBN:0-5343-5967-1, July 13, 2000.

[Wu et al., 1998], Kun-Lung Wu and Philip S. Yu and Ming-Syan Chen, *Efficient Caching for Wireless Mobile Computing*, Journal of Distributed and Parallel Databases, vol 6(4):351-372, October 1998.

[Yin et al., 2002], Liangzhong Yin, Guohong Cao, Chita Das, Ajeesh Ashraf, *Power-Aware Pre-fetch in Mobile Environments*, IEEE International Conference on Distributed Computing Systems (ICDCS), p.571-578, Vienna, Austria, July 2002.

[Yuen et al., 2000], Joe Chun-Hung Yuen, Edward Chan, Kam-Yiu Lam, H. W. Leung, *Cache invalidation scheme for mobile computing systems with real-time data*, ACM SIGMOD Record, v29(4):34-39, December 2000.

[Zipf, 1929], George Kingsley Zipf. *Relative frequency as a determinant of phonetic change*. Reprinted from the Harvard Studies in Classical Philology 15, p.1-95, 1929.