Semantic Data Caching in Mobile Computing

CMSC-628. Project Report

Ajay Prabhu
Gaurav Jolly
Sachin Bhatkar

{jolly1, sachin1, ajay1} @cs.umbc.edu

Advisor: Dr. Anupam Joshi

Computer Science Department
University Of Maryland Baltimore County
May 2002

Semantic Data Caching in Mobile Computing

Ajay Prabhu, Gaurav Jolly and Sachin Bhatkar
Computer Science department
University Of Maryland Baltimore County
Baltimore, MD 21227.
{sachin1, ajay1, jolly1}@csee.umbc.edu

Abstract:

Getting released from the wired world comes at a cost. Mobile devices have constraints like limited power and memory, low network bandwidth and high disconnection rates which keep them from transmitting or receiving data all the time. This signals the need to adopt a futuristic, context-aware approach in data exchanges. Therefore we need to employ some intelligent data management techniques (intelligent caching) to improve performance and quality. The standard caching techniques like LRU or MRU may not be sufficient, as they do not reflect the user's priorities. We propose that *profile driven semantic caching* is a better approach in the ad-hoc mobile environment. In this report, we discuss this approach and its advantages in the ad-hoc peer-to-peer mobile environment. We try to modify the Hybrid Caching Algorithm[10] so that it uses information in the user profile and semantic description of cached data to perform cache replacement. Moreover we validate these ideas with experiments and comparison of the results with that of LRU and Hybrid cache replacement algorithms.

1. Introduction:

With the advent of short-range ad-hoc connectivity (Bluetooth and WLANs), mobile devices will be more autonomous and will be able to assume the dual roles of information seekers as well as that of information gatherers. However, ephemeral vicinities of these devices mean that a data provider may not be always reachable. Also the type of information needed by the mobile devices vary with location, time etc. For example: If Roberto were a stockbroker, collecting data on the current stock-market prices would be one of the primary data collection tasks for his palmtop. However, if facts about Roberto (like his meal-times, his movie tastes, etc) are stored in a profile, the device could dynamically change the priorities assigned to the data collection tasks depending on the time and context. (If Roberto takes lunch at a fixed time everyday, the palmtop would start caching ads of restaurants that serve his favorite dishes just before his lunch or dinnertime).

Motivation:

Mobile devices have limited resources (network bandwidth, memory, computation power) at their disposal. However, advances in technology have made them powerful enough to support reasonable caching and query processing. It is extremely important for mobile devices to decide what data to store locally and what its value (priority) should be depending upon the context because gathering irrelevant information could be very inefficient and could degrade the performance. Present caching techniques do not take into consideration the changing user requirements.

Profile-driven data management is a framework in which data management policies are based on analysis of user data requirements specified in profiles. So, combining caching with the user profiles and context (location, time) also known as Semantic Caching could be extremely useful in an ad-hoc environment. Such a technique could improve the cache performance thereby lessening the number of data exchanges needed.

Devices implementing such a technique would be immensely useful in real-life scenarios such as a door-to-door salesman accessing data on his mobile device in response to his customer's queries, drivers accessing online road routes, etc.

2. Related Work:

Traditionally caching has been done based on temporal and spatial parameters. From the discussion above it is clear that in ad-hoc environment data does not merely depend on time and space. User preferences and the context (location, time, period, activity, etc.) are equally or sometimes more important. Thus recently the focus has shifted on developing Semantic Caching techniques. One of the main issues in caching is the criterion for replacement of the existing data in cache. In [4], semantic locality is used for caching. For each cached item, the value assigned is the negative of Manhattan distance between the "center of gravity" of that region and the "center of gravity" of most recent query. With this distance function, semantic regions that are "close" to the most recent query have a small negative value irrespective of when they were created. Thus cache replacement is based upon the semantic relation with the current query. Moreover their experiments show that semantic caching performs better than page caching and tuple caching. In this paper, the authors have used a client-server model. Moreover they use just one parameter (location) as a cache replacement criterion. In [2] also, the focus is on location dependent data caching. Here semantic caching is used in Location Dependent Data applications, with every cached item having an attached semantic description. In [3], authors argue that strict control exercised by traditional DBA based data management approaches may not be appropriate in applications where data to be stored is huge and dynamic (For example - Internet). They suggest a profile-driven data management in such a scenario. In their architecture, profile-driven data management is implemented as a middle layer that receives profiles from the users and processes those profiles with respect to a set of data resources. They also present a resource allocation algorithm, called "greedy algorithm" that allocates resources based upon the value of a resource. Basically all these approaches more or less assume a client-server model and take into consideration either profile parameters or temporal and spatial parameters.

Our work is based on the framework suggested in [1] which consists of an ad-hoc mobile network based on the Bluetooth protocol. Devices are able to interact with each other. Thus every device is an independent entity and can consume as well as produce information.
We simulate such an environment and observe the caching for a single node/device. The other device/devices are assumed to be only information providers in our simulated environment. Devices communicate with each other using DAML [1] requests and responses. The user-profile is maintained in a DAML file. We use Jess to describe rules based on the user-profile maintained facts.

3. Some Languages/Tools/APIs used :

3.1 DAML:

In our implementation, user profiles and data file instances are represented in DAML (DARPA Agent Markup Language) [7].
(The DAML description below has been referenced from [7]).
DAML is a joint effort started by DARPA and W3C, aimed at developing an extension to XML (Extensible Mark-up Language). The goal of the DAML effort is to develop a language and tools to facilitate the concept of the Semantic Web. DAML supports new structures that have the capability to describe relationships between objects. Therefore unlike XML, DAML provides a basic infrastructure that allows machine to make simple inferences without human intervention. DAML ontology's provide a very powerful way of describing relationships between objects. DAML provides constructs called properties that define relationships to connect two items. The properties can describe relation between two objects as well as a relation between an object and a data type. For example the relationship between a person and his shoe size can be described by first defining a class as person and than defining a data type property shoe size for that class.

3.2 Jess:

In our implementation, rules using the facts defined in the user profile are written in Jess[].
(The Jess description below has been referenced from [8]).
Jess is an acronym for Java expert system shell. It is a rule engine and a scripting environment written entirely in Java. It is used to build an expert system that is a set of rules (rule engine) that can be applied to a collection of facts.
A rule-based system maintains a collection of knowledge nuggets called *facts.* This collection is known as the *knowledge base.* Rules can take actions based on the contents of one or more facts. A Jess rule is something like an `if... then` statement in a procedural language, but it is not used in a procedural way. Jess rules are executed whenever their `if` parts (their *left-hand-sides* or *LHSs*) are satisfied, given only that the rule engine is running. This makes Jess rules less deterministic than a typical procedural program.

3.3 SiRPAC :

(The Jess description below has been referenced from [9]).
SiRPAC is the W3C RDF reference implementation that parses the XML encodings of RDF into the triple data model, to be used in applications. SiRPAC can be run as a command-line tool or embedded in Java applications. SiRPAC uses SAX 1.0 to parse documents, which means that it can work with nearly any Java XML parser.

4. Profile Implementation:

The user-profile is a DAML file, which contains the various activities that the user does daily and their time intervals (basically intervals slotted by start time and end time. For eg. activities Breakfast, News in interval 8:00 to 9:00).
A snippet from the user-profile is :

```
<profile:Time rdf:ID="time1">
    <profile:starthrs>08</profile:starthrs>
    <profile:startmins>00</profile:startmins>
    <profile:startsecs>00</profile:startsecs>
    <profile:endhrs>09</profile:endhrs>
    <profile:endmins>00</profile:endmins>
    <profile:endsecs>00</profile:endsecs>
    <profile:hasLocation>
    <profile:Location rdf:ID="locn1">
         <profile:city>baltimore</profile:city>
         <profile:state>MD</profile:state>
         <profile:zipcode>21227</profile:zipcode>
    </profile:Location>
    </profile:hasLocation>
    <profile:hasActivity>
    <profile:Activity rdf:ID="act1">
         <profile:name>news</profile:name>
    </profile:Activity>
    </profile:hasActivity>
</profile:Time>
```

Also, all the requests and responses are DAML files.
We use SiRPAC to generate 3-tuples for the DAML data in each of the DAML files whenever needed. These 3-tuples are essentially the facts which can be asserted as knowledge in the Jess-engine knowledge base. Rules are defined based on these facts. So, whenever a combination of facts is asserted (loaded) into the Jess knowledge base, if there is a rule, which has all the facts on its LHS in the knowledge base, that rule executes and its RHS is carried out.
So, we assert the various facts about the user (his activity and its time-slot) in the knowledge base. Whenever, the system time lies between this interval (tested by means of a function liesBetween() written by us in Jess), we assert a fact that a match is found. Thus we calculate the utility values for each activity based on its "nearness" to the current system time (context).

The utility values are basically calculated using the following logic :
First we calculate the differences of each activity's slot with respect to the system time as follows :
If the current time is not in the activity's slot and if it is a future interval (relative to the current day)
Diff = slot_start_time – curr_time
Else If the current time is not in the activity's slot and if it is not a future interval (relative to the current day),
Diff = (slot_start_time – curr_time) + one_day_in_secs.
else if the current time is in the activity's slot
Diff = slot_end_time – curr_time.

All the differences are in milliseconds.

We then normalize the above values using Min-Max normalization technique in the new range (1 to 10) to get the utility-values for each activity.

4.1 Implicit queries based on History:

In traditional caching algorithms queries are always explicit. Whenever a user requests certain information the caching algorithm first looks in to the cache, if the information is there in the cache it is returned to the user. Otherwise the query is forwarded to the information source.
With semantic caching in addition to explicit queries the Profile manager can make implicit queries as well. We maintain a history of the queries that the user generates and then use them using the Jess based rules to generate implicit queries some time before a related impending event. Thus, using the information from the context manager and the rules generated by the rule engine implicit queries can be generated. The assumption here is that the user ‘s daily habits are likely to produce the same queries at the same times everyday and so implicit querying will help in caching the response to queries which the user will “most likely” generate at particular times.

4.2 What we were not able to simulate in the User Profile but we would have liked to :

We would have also liked to incorporate rules for matching the location of the activity with the current location (context). We tried to do so, but the rules to be defined in Jess proved to be too complex and we ran out of time. However, we infer that if implemented the algorithm would definitely get more context-aware and better.
Also, for the experimental purpose, we used only a static profile. An ideal scenario would be to use the history that we have maintained to dynamically change the profile. This could also be done in combination with a BDI model, which will store the user’s actual likes and dislikes. It will require more complex Jess rules and complex parsing (using SiRPAC) .

5. Project details

5.1. Motivation and Ideas

The scenario we used for our project was largely based on the idea in MoGATU [1].
A brief description of the architecture as given in [1] is as follows:
The ad-hoc network uses the Blue tooth protocol stack developed by Axis communications Inc. [5] and Bluetooth modules developed by Ericsson. The framework consists of entities, which are autonomous in the sense that their actions are in no way dependent on their owners. Every entity implements InforMa, which is a local metadata repository that includes schema definitions for locally available information providers and facts such as queries and answers for local and non-local information providers. The devices discover each other, connect to each other, route data (whenever possible), and query each other. Each device should keep information about its data providers, remote data providers and particular answers to queries (i.e. data). For example, a device should know that it provides weather updates and it may even know that it is snowing (i.e. both meta-data and data). A user has a profile, which somewhat reflects his beliefs, desires and intentions. The device is able to create "standing queries" and other rules based on the profile, and keeps checking on the available information to make sure that user always gets the most necessary/useful information at all times. For example, while a user is driving, the device is caching traffic conditions but when he is at home the device is “snoring” or checking for the news updates.

5.2. Our Experimental Set-Up

We have simulated the following components :

5.2.1 Entities

They simulate the mobile devices (which can be both information seekers as well as providers) simulating the peer-to-peer mobile environment. For our experimental purposes we assume the “services” as belonging to certain fixed domains like video library, gas station, weather service, traffic information, etc. These entities are essentially Java threads continuously running and listening for requests.

MyNode is the entity, which simulates the user’s mobile device in question in our experiment. All experimental results are calculated for MyNode only.

MyNode will consist of the following modules:

5.2.2 Profile Manager

The Profile Manager parses the DAML representation of the user profile. It initializes the Jess rule-engine (Rete based) and asserts the facts from the user-profile in the database. It then defines the rules for the facts. The engine will keep on running and whenever a particular rule is satisfied, it will be executed.

5.2.3 Context Provider:

This module is actually in-built within the Profile Manager. We keep on changing the current time (forward it by 30 minutes) every 1-minute to go through each time interval of the user’s activities.

5.2.4 Cache Manager:

This is part of each client entity and it uses the semantic caching algorithm for replacement. The caching algorithm uses profile information and certain other parameters (profile utility value, file size, number of references, etc) to calculate the final utility values (importance or priority) of each data stored in the cache. The formula used to calculate the utility value is loosely based on the Hybrid Cache removal algorithm, which is motivated by Bolot and Hoschka's algorithm [2]. The parameters for our algorithm are:

- Profile utility value
- Timestamp
- No of references for this type of queries
- Size of the reply

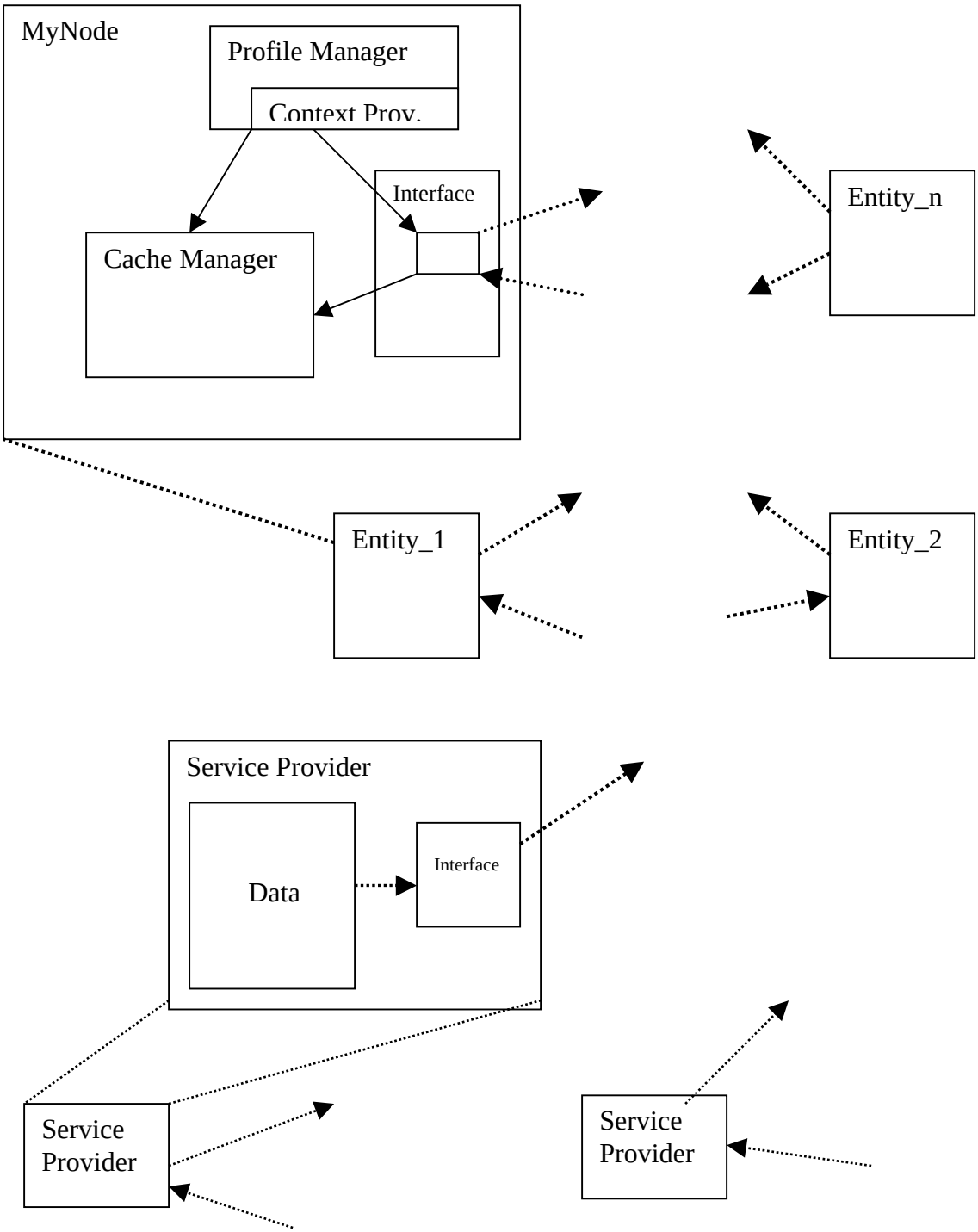
All these parameters have weights associated with them. Thus the final weight value for a file is calculated as:

Weight value = (Profile utility value * wt for profile + Distance * wt for distance + No of references * wt for references) / (reply size * wt for size.)

The important thing to be noted is that the final weight values for the cache entries keeps on changing. This is because the Profile Manager bases its decision on the location and time at that moment (simulated by the Context Manager) and in turn calculates its utility value based on its set of rules. The cache is a Hash Table structure with the key being a combination of object predicate and subject and the value being replies that have been got for queries related to this service and for each of those replies, an object containing information like the DAML file path, number of references, time of creation and other parameters.

At present, we have kept the query processing logic very simple. The query is in the form of a tuple (subject, predicate, object). For e.g. : (traffic , hasRoute, route1). We implemented a simple logic as query processing was not in the scope of the project.

5.2.5 Architecture:



6. Experimental Results:

We have carried out experiments to calculate the following parameters :

- Avg. Turn-around time
- Hit Ratio.

We have plotted the following graphs :

- Hit Ratio Vs Cache Size
- Avg. Turn-around time (for 68 requests) Vs Cache Size

for 3 cache replacement algorithms : LRU, Hybrid algorithm and Profile-Driven Semantic Caching Algorithm.

One basic assumption on which our motivation towards developing a semantic caching algorithm was that the user tends to follow a set pattern in his/her daily activities. So, we tailored the requests to be in a sequence that would almost match the user's schedule. The same set of requests was run twice and the average response time and hit ratio were noted.

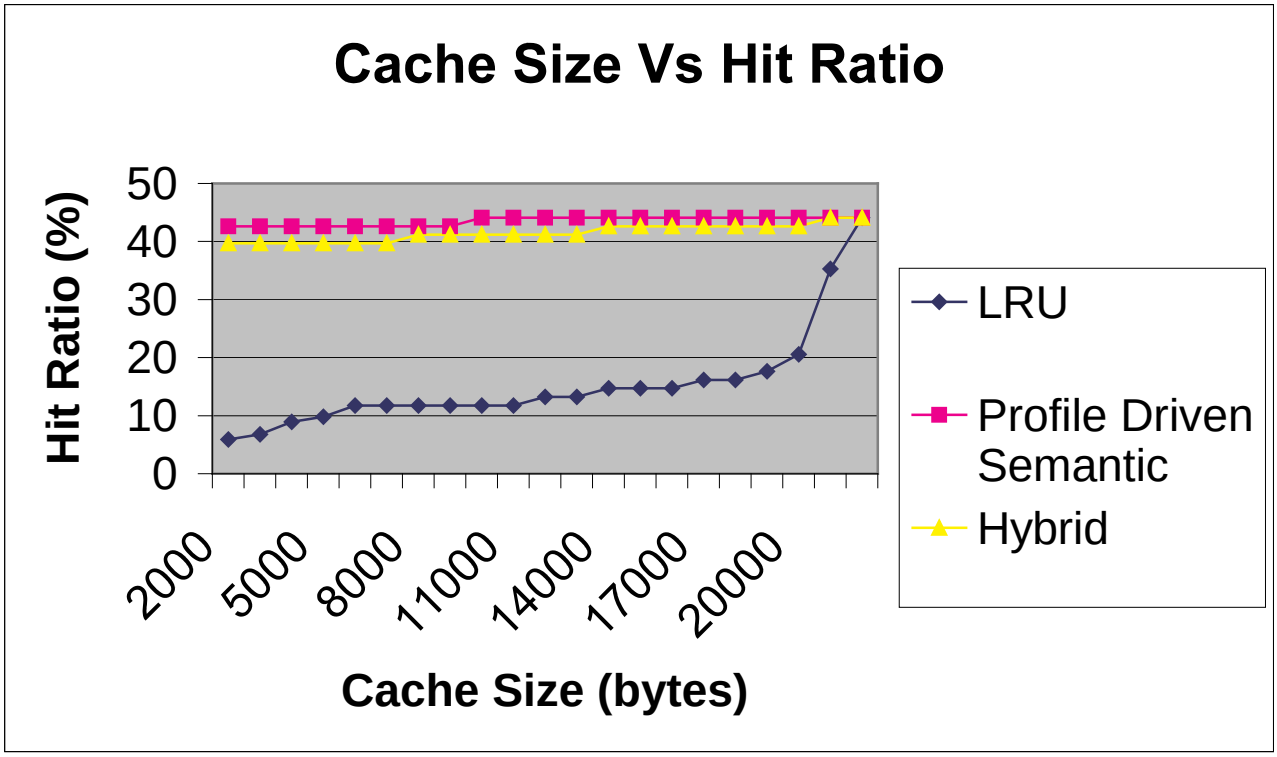
While giving out requests for the Profile-Driven Semantic Caching Algorithm, we took care to give requests related to an activity in the corresponding time interval (that was the only way we could think of for closely simulate the daily request pattern of an user). This was done by running the system time 30 minutes ahead after every 1-minute of the actual physical time and giving related queries every 30 minutes.

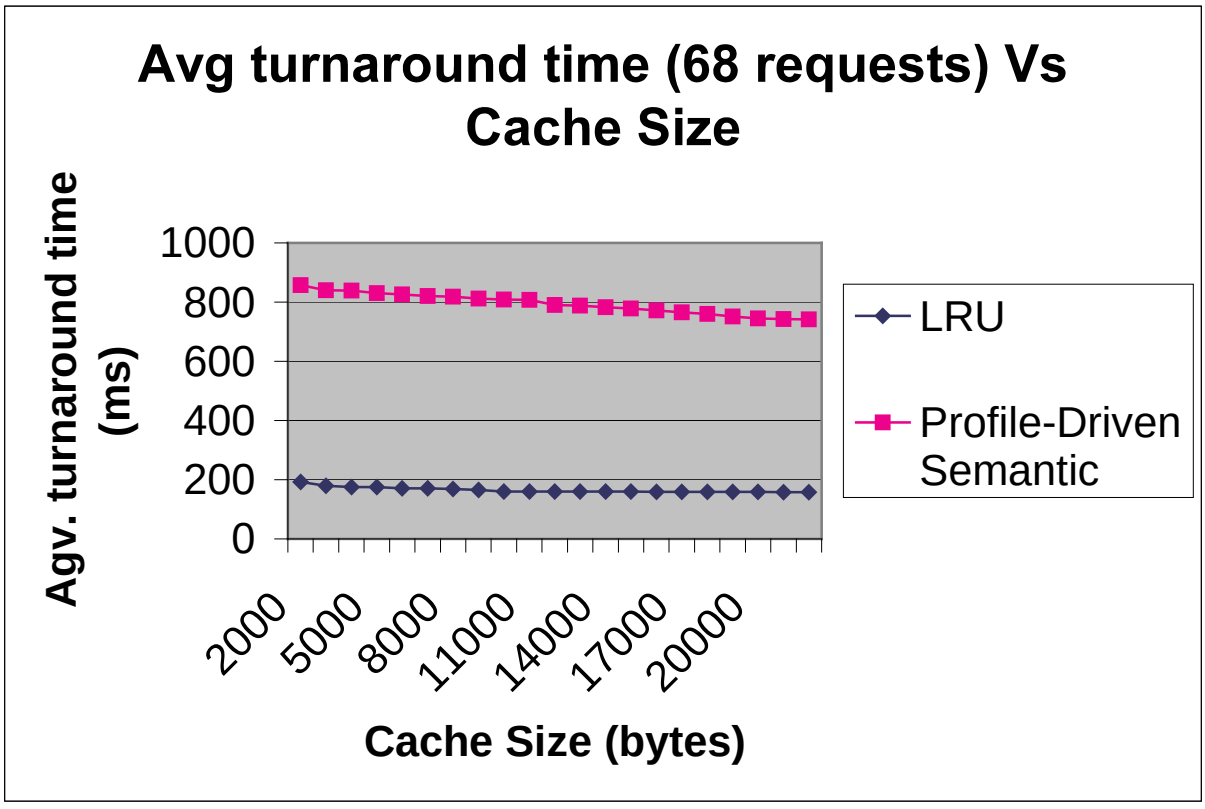
From the first graph, we see that the hit ratio for our profile-driven semantic caching algorithm is the highest and even at smaller cache sizes (which could be very much relevant in our mobile environment scenarios) the hit ratio comparatively (w.r.t LRU) very high.

The second graph gives the average turnaround time for 68 requests. The turn-around time includes the time when a request is sent , time to check in cache and time to get the request if not in cache. The time is high for our algorithm. This is because of the initial readings (empty cache) when there will be replacements. However, if the graph continues, we think that it should be low as the cache hit ratio is high and most of the requests will be satisfied quickly thus annulling any cache replacement delays.

We could probably reduce the time taken for the request by optimizing the code. The high time as compared to LRU is obviously because of the calculations involved in our algorithm each time the replacement takes place.

The purpose of using semantic caching is that user should always have access to data of his interest. Therefore while comparing such an algorithm with LRU or any other algorithm, the number of queries satisfied successfully (cache hits) should be more important a parameter than the response time.


Cache Size Vs Hit Ratio
Hit Ratio (%)
50
40
30
20
10
0
2000
5000
8000
11000
14000
17000
20000
Cache Size (bytes)
LRU
Profile Driven Semantic
Hybrid


Avg turnaround time (68 requests) Vs Cache Size
Agv. turnaround time (ms)
1000
800
600
400
200
0
2000
5000
8000
11000
14000
17000
20000
Cache Size (bytes)
LRU
Profile-Driven Semantic

7. Conclusions and Future Work:

We have implemented a profile-driven semantic data-caching algorithm that uses utility values based on the user's profile. We have also compared the performance of our algorithm with other caching algorithms LRU and Hybrid. Our plans for future work include developing logic for generating a dynamic user profile based by drawing inferential rules from the query history and incorporating a BDI (Beliefs, Desires and Intentions) model in the Profile Manager to make the cache content as close to the user's interests as possible.

Finally, we would like to thank Mr. Filip Perich who guided us in the initial phase of the project and helped us come up with scenarios for implementing the project.

References

[1] Filip Perich, Sashikanth Avancha, Anupam Joshi, Yelena Yesha, and Karuna Joshi. Query Routing and Processing in Mobile Ad-hoc Environment. Technical Report, November 2001. TR-CS-01-18.
[2] Qun Ren and Margaret H. Dunham. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. Proceedings of Mobicom 2000.
[3] Mitch Cherniack, Michael J. Franklin and Stan Zdonik. Profile-Driven Data Management. Brown University. Technical Report, Feb 2001.
[4] Shaul Dhar, Michael J. Franklin Bjorn T. Jonsson, Divesh Srivastava and Michael Tan. Semantic Data Caching and Replacement. In Proceedings of VLDP, pages 330-341, Bombay, India, September
1996.
[5] Mitch Cherniack, Michael J. Franklin and Stan Zdonik. Expressing User Profiles For Data Recharging. IEEE Personal Communications, July 2001.
[6] Axis Communications Inc. OpenBT: An open source blue tooth stack for Linux, 2001. http://sourceforge.net/projects/openbt.orks semantic caching has emerged as an important area of research.
[7] DAML (DARPA Agent Markup Language) www.DAML.org
[8] Jess the Rule engine for the Java Platform (http://herzberg.ca.sandia.gov/jess/)
[9] SiRPAC parser (http://www.w3.org/RDF/Implementations/SiRPAC/)
[10] Roland P. Wooster and Marc Abrams, "Proxy Caching that Estimates Page Load Delays" , WWW6, April 1997, pp. 325-334