MOBILE COMPUTATION OF QUERIES IN AN UNCERTAIN ENVIRONMENT[*]

Sarah Monisha Pulimood, Boumediene Belkhouche and Adnan Yazici

Department of Electrical Engineering and Computer Science,
Tulane University, New Orleans, LA - 70118, USA
Tel: (504) 865-5840, Fax: (504) 862-3293
{pulimood, bb, yazici}@eecs.tulane.edu

Abstract

The World Wide Web (WWW) is exploited by the average user for communication, data access, business transactions, and entertainment. In a competitive environment, it is essential for (possibly mobile) users to be able to retrieve relevant information from databases geographically distributed over the Internet, regardless of their own location. Such information retrieval entails computation of queries in an uncertain environment. We examine some of the issues involved like the uncertainty in data, and in criteria like complexity or computation size of the query, current network traffic, etc. which are evaluated by the system while deciding when and where a mobile computation should move. This paper proposes a framework for a system that incorporates concepts from fuzzy logic into a plan language to formulate and execute a plan for a mobile query computation that will process a large query efficiently in an uncertain environment like the Internet.

Keywords: *Fuzzy Logic, Logic & Databases, Information Retrieval, Reasoning Under Uncertainty*

1 Introduction

The influence of the Internet has permeated almost all aspects of society, due in great part to the World Wide Web (WWW) and the quantum advances in information technology equipment and applications. The changing requirements of today's economy and the success of the Internet are driving research in uncertainty and mobility in several areas of information technology, including database applications, with a vision of being able to harness the power of the Internet by sharing available resources in a seamless, secure and efficient manner. Mobile computations have much to offer in an uncertain environment, like the Internet, by enabling the use of portable, low cost, and personal communications devices; secure communication on public networks; and efficient, economic use of low bandwidth, high latency and error-prone communications channels [1]. Autonomous mobile computations can invoke resources locally eliminating network transfer of intermediate data. We distinguish between the two main aspects of mobility - mobile computing and mobile computation. *Mobile computing* refers to physical mobility [2], i.e. mobile hardware like laptops, palm tops, etc. *Mobile computation,* on the other hand, refers to virtual mobility or mobile software [2], that can suspend execution on one machine, migrate to another machine, and then resume execution on the new machine from the point at which it left off [3], [1], [4]. In this paper, a mobile computation denotes *"software that travels on a heterogeneous network, crosses protection barriers, and automatically executes upon arrival at the destination"* [5].

Many of the issues that come into play when processing queries on the Internet, like complexity, processing power, etc. are inherently imprecise and have domains with uncertainty. The databases involved in the query could contain fuzzy data. For example, the age of a person may be stored as "old" or "very young", instead of with a crisp value like "67" or "19". A query language (like the one proposed in [6]) can support fuzzy queries by incorporating fuzzy logic to handle uncertainty [7]. Although fuzzy databases are not very commonly used commercially, we take into consideration that some databases in the system may support fuzzy data and fuzzy queries.

It is desirable to perform transactions with a minimum of data transfer over the network but this consideration should be balanced by the availability of resources. For example, if a query requires resource intensive computations, it may be cheaper and more efficient to transfer parts of the databases to another much faster host and compute it there, rather than avoid the overheads of network transfer and lose on performance. The Internet is composed of heterogeneous distributed systems, so

[*]Proceedings of First International Conference on Intelligent Computing and Information Systems, ICICIS 2002, Cairo, Egypt, June 24 - 26, 2002.

there has to be some standard method for identifying and understanding a computation, i.e. the computation should be portable online onto any architecture [8]. We use mobile computations [8] that visit sites that can understand and execute them.

In this paper we present a framework for a Query Processing System that utilizes a fuzzy meta-database, fuzzy if-then rules, and concepts from fuzzy set theory to express imprecise information, and mobile computations to process large queries efficiently on a heterogeneous and uncertain network. We have implemented a prototype for a system that generates and executes mobile computations in heterogeneous environment.

Section 2 provides a summary of related work. Section 3 introduces the Query Processing System that creates and manages mobile query computations. Our conclusions and future directions are discussed in section 4.

2 Related Work

Cardelli [9] describes all aspects of mobility within a single framework that encompasses mobile agents, the ambients where agents interact, and the mobility of the ambients themselves. (An ambient is a bounded place where computations take place. It can be nested within other ambients, and can be moved as a whole [10].) Murase et al. [11] proposed the active mobile database system (AMDS), with a data handling method for asynchronous events in a mobile computing environment. Saygun et al. [12] merge mobile, active, and fuzzy databases on a common platform to construct a system capable of supporting mobility of data and computers as well as active and fuzzy features. (An active database management system allows users to specify actions to be executed when specific events are signaled under certain conditions [12], while a mobile database is defined as one whose servers and / or clients run on mobile computers [11].) Imielinski et al. [13] present a number of data management problems that arise due to mobility of both data sources and data consumers. The D'Agent system [3] supports mobile agents that are spawned onto the local machine and move to a different machine based on information about the network, like bandwidth of the network link, uptime / downtime history, etc. obtained from network-sensing agents. Papastavrou et al. [14] propose the "DBMS-Aglet Framework", which launches Java mobile agents into the unstructured network to roam around and gather information.

There is a large body of work on mobile computing, mobile agents, fuzzy databases and fuzzy theory as distinct areas. However to the best of our knowledge there is no work, so far, that performs mobile computations [8] on large database queries, using fuzzy logic and set theory to manage the uncertainty [7] of the environment and for planning and improving performance, as presented in this paper.

3 Query Processing System

In this section we discuss our framework for a Query Processing System (QPS) that manages the processing of a large user-generated query. The system assumes that all the databases required to run a query are connected on the Internet. There are also several computing hosts that the user has access to, which allow mobile computations to be performed on them. We also assume that all the databases and computing hosts in the system understand the mobile query computation.

3.1 An Example Scenario

We now present an example scenario which will be used throughout the rest of the paper to illustrate the concepts discussed. Using some of the insights gained from this example, we merge concepts from fuzzy set theory and planning to develop a fuzzy plan language. We then go on to formulate the plan for the example using this fuzzy plan language.

Let us take the example of a senior marketing manager traveling around the world exploring the marketing opportunities for a new product recently released by her company. The company maintains information on the demographics of their existing customer base. Through strategic alliances with other corporations they have access to their databases as well. The manager uses wireless or telephone communication to connect her portable notebook computer to the Internet and thus to her company network. She is currently in Chennai, India, hoping to set up a manufacturing facility and a marketing network for the company's latest line of products. To ensure that the negotiation meetings with high ranking government officials and powerful business people are successful, she requires a lot of data at her fingertips. For example, she may need an analysis of the customer base for the current product line in other countries where it is being marketed, or she may want to estimate the likelihood of families with different income levels buying the new product, based on their purchasing patterns over the last six months etc. Fuzzy queries will extract more meaningful data for her purposes. Some of the required databases are on mobile computers, and it is uncertain whether these computers are online and what their current locations are. All the databases required to compute the queries are geographically far apart. Since the manager requires a lot of comparative analysis, the queries will entail massive join operations. A join operation that has to be performed on databases located far apart will require considerable resources and generate network traffic. Hence the query computation should be moved to some other faster,

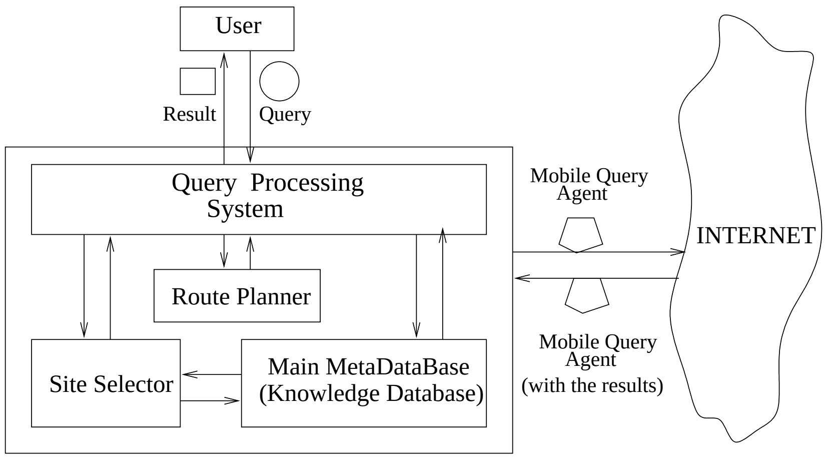

Figure 1: Query Processing System (QPS)

and preferably stationary host, which has the necessary resources to complete the operations in the shortest possible time. The results of the query can then be transferred back to the manager's computer when she comes back online and is ready to receive the data. In this way resource (including network) usage can be optimized for performance.

3.2 System Overview

The Internet is basically a system of independent heterogeneous networks of arbitrary design interconnected through open architecture networking [15]. The QPS views the hosts available for its use as nodes on a graph representing the network.

Figure 1 shows a schematic of the Query Processing System.

The QPS accepts a query from the user. It then sends a fuzzy query to the meta-database (Knowledge Database) to obtain the locations of each of the databases required to process the user's query. Our hypothetical manager, for example, could send a query like 'Group all the customers of the company by their household income'.

The meta-database contains all the relevant information about the databases and the environment, like name of the host, locations where each database is maintained, network conditions etc. Relevant information regarding previous computations may also be stored for use in estimating the size of the computation. The results, in this case, are the names and locations of the database servers at Tokyo, Los Angeles, and New York. These are all fairly large databases and the join operations on them would be computation intensive, so this query would be termed a "very large" computation.

This information is passed to the Site Selector which is basically a fuzzy if-then rule system that, in consultation with the meta-database, selects and returns the best choice for the computation site along with a list of possible alternatives based on the weights given to various criteria. The rules are of the form:

```
If ComputationSize is 'very large' then
   From the meta-database get all processors that are 'very powerful'
   with a threshold of 0.9
```

In our example, the most important criteria for the manager are speed in processing the query and security. Hence she assigns very high weights to "processing power" and "security" level. In another situation, minimizing the cost of using outside resources may be more important than security, so a high weightage would be given to "usage cost" and a low one to "security". As per the meta-database, the computing hosts at Sydney and Chennai are the most powerful ones that the user has access to.

Depending on the tasks to be carried out, and the current information available about the environment, an initial plan is generated by the Route Planner to most efficiently process the query. The parameters for efficiency may be predefined or modified by the user and may include minimizing the time taken for the computation, minimizing the amount of network transfers, maximizing security precautions, etc.

Since the computation has to travel to different sites to collect information, and the computation is very large a Mobile Query Computation (MQC) is created and deployed over the Internet to perform the query computation as per the plan. The plan generated in this example is for the MQC to visit the database server at Tokyo (A) to obtain the information resident there. Since the computing station at Sidney (B), a very powerful host, is 'nearby' (in terms of network distance), the MQC can go there to perform some partial computations. The MQC can then go to the database servers at Los Angeles (C) and New York (D) to gather further information before going finally to the computing host at Chennai (E). Once all the computations are

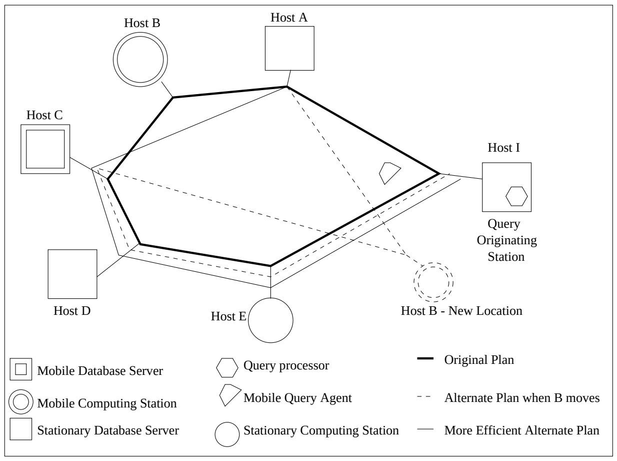

Figure 2: Schematic of example scenario

Table 1: Sample Similarity Relation Matrix

	very large	large	average	small	very small
very large	1.0	0.9	0.6	0.2	0.0
large	0.9	1.0	0.8	0.5	0.1
average	0.6	0.8	1.0	0.8	0.3
small	0.2	0.5	0.8	1.0	0.8
very small	0.0	0.1	0.3	0.8	1.0

completed, the MQC can return to the initiating host, i.e. the manager's laptop at Chennai (I). The thick solid line in Figure 2 shows the initial plan.

The QPS receives the end results and presents them to the user in the form required.

3.3 Meta-database

Information about the parameters describing the state of the environment is maintained in the meta-database. The imprecise nature of these parameters is expressed using fuzzy terms characterized by similarity matrices [7]. *A similarity matrix describes the similarity between each pair in the same domain.* A sample similarity relation matrix [1] is shown in Table 1.

Computation Size One of the primary motivations for a mobile computation to move from one host to another is the size of the query computation, which can be classified in terms of the resources it requires. For example, a query that requires join operations on several large tables, selection of data based on certain criteria and then some calculations on the results, would be defined as "very large" since it is computation intensive and would require a very fast processor for efficient execution. In contrast, a simple select operation on one table requires very little computation and could be termed "very small". The domain for this parameter is D = {very large, large, average, small, very small}.

Processing Power of the Host It is logical for a "very large" query to move to another host only if this host is more powerful in terms of speed, availability of resources like memory etc. A "very powerful" host is one that has a very fast processor, considerable memory, and other resources available for use, while a "very weak" host has a very slow processor with very little or no resources available for use. The domain is formulated as D = {very powerful, powerful, average, weak, very weak}.

Location In the case of mobile databases, the exact current location of a host may be unknown or uncertain. These data can be maintained as "near" or "very far" with respect to some specific (possibly the originator's) location. The domain for this parameter is D = {very near, near, far, very far}.

[1]Fuzzy values included in the domains discussed in this paper are application specifi c and may vary depending on the designer.

Security Level Authenticity of the computation (who the originator is and what the access rights and privileges on the system are) and availability (the computation is not able to deny access to resources it uses) are important aspects of security for a computation host, so it will not easily open its resources to a foreign entity. Formal methods are evolving that can be used to prove that code written in a language will always be secure [16], [17], [18], [19]. The MQC generation language used in this system is assumed to provide adequate assurances on this front.

The MQC is interested in knowing which hosts it is allowed to execute on and use resources from. Security considerations include the fact that communication is over open and possibly insecure network connections. The computation also needs to know how secure its data will be on the host. The domain for the host's security level can be D = {very secure, secure, insecure, very insecure}.

Network Conditions Reliability of the network connections can be calculated based on statistics of current and past conditions, or based on assurances by the provider. A "very reliable" connection maybe one that has a history of being available even in the most adverse conditions, while a "very unreliable" condition may be one that frequently goes down regardless of the environment. The reliability of a connection to a stationary host can usually be expected to be more than that of a mobile host, since the latter can go off-line at any time. A mobile computer has the additional disadvantage of usually being run on batteries which could expire before the computation is completed, resulting in loss of data. So there could be a factor of unreliability assigned to any host that is mobile. The domain for the reliability of a network connection could be D = {very reliable, reliable, average, unreliable, very unreliable}. Traffic conditions on a network connection, e.g. "very high" traffic or "normal" traffic, can also be expressed in a similar manner.

Usage Cost A host that is willing to allow its resources to be used by outside computations may require compensation. One goal of the QPS could be to minimize the cost of resource usage for the MQC. The identification and pricing of such resource usage is still a matter of debate and research. However, if some method is being implemented the fuzzy domain is D = {very expensive, expensive, okay, cheap, very cheap, none}.

3.4 A Fuzzy Plan Language

Referring to the plan in Figure 2, we see that one of the preconditions for the first action step will be a reliable connection between I and A. This can be expressed by the predicate RelConnBet (I, A). Languages traditionally used in planning systems (e.g. STRIPS) assume discrete inputs and actions, so the value of the predicate would be either 'true' or 'false', since 'reliable' would have a crisp value described as being, say, 'between 1.5 and 2'. Let us assume the only available connection between hosts I and A has the value 1.4. This would not be considered to be 'reliable' and the precondition would fail. Similarly a value like 2.1 would cause the condition to fail. However, in a fuzzy system like the one described here, each predicate is a fuzzy set having a domain characterized by a membership function. The values 1.4 and 2.1 could thus be considered as 'reliable with a membership of (say) 0.9'. As long as the membership value is above a threshold, say 0.8, the precondition would still be satisfied in either of these cases. The plan language used in the Route Planner must be capable of generating plans that can utilize the information in the fuzzy meta-database and express the uncertain nature of the Internet environment. We incorporate fuzzy set concepts into the plan language to describe such preconditions and the effects for each action step of the plan.

The results of predicates in the language would be one of a range of values from the fuzzy sets. The degree of membership of an element $u \in U$ in a fuzzy set F can be denoted by

$\mu_F(u)$.

For example, we can discover whether host B is a very powerful processor by computing the degree to which the processing power of B belongs to the fuzzy set 'VeryPowerful'. If the membership degree $\mu_{VeryPow}(B)$ is above a specified threshold, we can say that B is a very powerful processor.

Some predicates like At (M,D), indicating that the MQC M is at host D, may have crisp values - either M is at D or it isn't. To maintain homogeneity we can consider such predicates also to be fuzzy sets but with a domain of two values, e.g.

$\mu_{At}(M, D) = 1$ indicates that M is at host D and

$\mu_{At}(M, D) = 0$ indicates that it is not.

A set of preconditions for an action step in STRIPS is expressed as the conjunction of all the preconditions [20]. The 'maximum' and 'minimum' operators are commonly used to express disjunction and conjunction respectively in fuzzy sets. Thus we can express the preconditions that B should be close to A and B should be very powerful, as:

$\min\{\mu_{VeryPow}(B), \mu_{CloseTo}(A, B)\}$

The initial plan generated by the QPS for the on-going example is shown in Table 2. The preconditions for each action step expressed in the fuzzy plan language are also shown. The MQC is denoted by 'M'.

Table 2: Initial Plan shown schematically in Figure 2.

	Action	Preconditions	Annotations in fuzzy plan language	Effects
1	Go to host A	Connection between hosts I and A is *reliable*	$\min\{\mu_{RelConnBet}(I,A), \mu_{At}(M,I)\}$	$\mu_{At}(M,A)$
2	Get qry results W	Table T_A is available	$\min\{\mu_{At}(M,A), \mu_{Has}(A,T_A)\}$	$\min\{\mu_{Has}(M,W), \mu_{At}(M,A)\}$
3	Go to host B	Connection betn hosts A & B is *reliable*, B is *close* to A or C, B is *very powerful*	$\min\{\mu_{RelConnBet}(A,B), \mu_{At}(M,A),$ $\mu_{VeryPow}(B), \mu_{Has}(M,W),$ $\max\{\mu_{CloseTo}(A,B), \mu_{CloseTo}(C,B)\}\}$	$\mu_{At}(M,B)$
4	Perform partial computation X	W is available	$\min\{\mu_{At}(M,B), \mu_{VeryPow}(B),$ $\mu_{Has}(M,W)\}$	$\min\{\mu_{Has}(M,X),$ $\mu_{At}(M,B)\}$
5	Go to host C	Connection between hosts B and C is *reliable*	$\min\{\mu_{RelConnBet}(B,C), \mu_{At}(M,B)\}$	$\mu_{At}(M,C)$
6	Get qry results Y	Table T_C is available	$\min\{\mu_{At}(M,C), \mu_{Has}(C,T_C)\}$	$\min\{\mu_{Has}(M,Y), \mu_{At}(M,C)\}$
7	Go to host D	Connection between hosts C and D is *reliable*	$\min\{\mu_{RelConnBet}(C,D), \mu_{At}(M,C)\}$	$\mu_{At}(M,D)$
8	Get qry results Z	Table T_D is available	$\min\{\mu_{At(}M,D), \mu_{Has}(D,T_D)\}$	$\min\{\mu_{Has}(M,Z), \mu_{At}(M,D)\}$
9	Go to host E	Connection between hosts D and E is *reliable*, E is *very powerful*, E is *close* to D, M has Y and Z	$\min\{\mu_{RelConnBet}(D,E), \mu_{At}(M,D)$ $\mu_{VeryPow}(E), \mu_{Has}(M,Y),$ $\mu_{CloseTo}(D,E), \mu_{Has}(M,Z),$ $\max\{\mu_{Has}(M,W), \mu_{Has}(M,X)\}\}$	$\mu_{At}(M,E)$
10	Perform final computation Q	(W or X), Y, and Z are available	$\min\{\mu_{At}(M,E), \mu_{VeryPow}(E), \mu_{Has}(M,Y),$ $\mu_{Has}(M,Z), \max\{\mu_{Has(}M,W), \mu_{Has}(M,X)\}\}$	$\min\{\mu_{Compl}(Q),$ $\mu_{Has}(M,Q), \mu_{At}(M,E)\}$
11	Go to host I	Connection between hosts E and I is *reliable*	$\min\{\mu_{RelConnBet}(E,I), \mu_{At}(M,E)$ $\mu_{Compl}(Q), \mu_{Has}(M,Q)\}$	$\mu_{At}(M,I)$
12	Present results R to user U	Q completed in *minimal* time T	$\min\{\mu_{Compl}(Q), \mu_{MinTime}(T)\}$	$\min\{\mu_{Has}(U,R),$ $\mu_{At}(M,I)\}$

3.5 The Mobile Query Computation (MQC)

3.5.1 MQC Overview

Changes in the current status of the environment like location (a mobile host may change its location), security level (a host is discovered to have a breach of security making it no longer desirable for computing confidential data), network conditions (a reliable connection becomes unreliable), etc. are continuously monitored by the External Conditions Sensor. The input is received from the environment itself and other mobile computations it can communicate with. Feedback about the environment is evaluated as a degree of membership in the fuzzy sets for each parameter and is of the form "traffic on connection A-B is *'very high'*", or "host B has moved *'far'* away".

The MQC must be able to process the information received and respond appropriately. Such a response will sometimes require some amount of replanning. This is done by the Decision Maker with inputs from the other modules. The Site Selector module of the MQC is a subset of the fuzzy if-then rules of the Site Selector in the QPS. The Knowledge Base is also a relevant subset of the main fuzzy meta-database of the QPS containing information about the hosts and network conditions that satisfy the conditions required to complete the query. As new feedback is received from the environment, this database is updated with the current state. When the MQC returns to the initiating host, the information in the QPS meta-database and that in the MQC database are compared and reconciled. The Action Effector carries out the actions specified in the plan.

Figure 3 shows the architecture of the MQC.

3.5.2 Replanning

In the on-going example, let us assume that after the MQC has completed its tasks on host A it receives feedback that B, a mobile host, has moved to a location further away, as represented by the dashed line in Figure 2. The preconditions to Step 3, CloseTo (A, B) and CloseTo (C, B) fail. The next step in the plan cannot be carried out and the MQC now has to decide on the next course of action. The MQC could generate the new plan itself - in which case it would need to have a replanning module. Alternatively, the relevant information could be sent back to the initiating (or nearest) QPS where a new plan could be generated and sent back to the MQC to be implemented. The option selected will depend on various factors. If the QPS is 'fairly close' to the MQC's current position and the environmental conditions are favorable, it would be possible to have the QPS generate

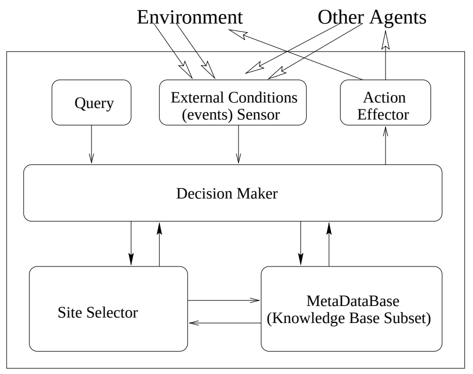
Environment
Other Agents
Query
External Conditions (events) Sensor
Action Effector
Decision Maker
Site Selector
MetaDataBase (Knowledge Base Subset)

Figure 3: Architecture of a Mobile Query Computation

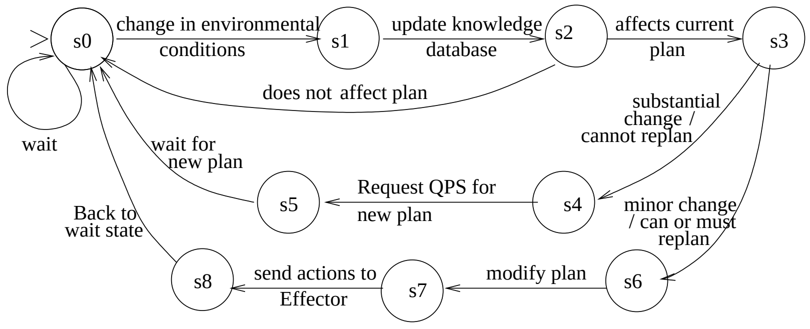

Figure 4: State Diagram for Decision Maker

the new plan. However if the MQC is 'far' from any QPS, this is not a feasible solution. By the time the information is sent to the QPS, and a new plan generated and sent back to the MQC, the environment could very well have changed again. In this case, it would be more practical for the MQC itself to perform the replanning. Other factors to be considered are whether the MQC has sufficient resources at the current host to be able to compute the plan, the extent of replanning involved, etc.

The Decision Maker module combines 'execution monitoring' and 'simple replanning' with the capability to replan under uncertainty. Figure 4 shows the abstract model for the Decision Maker as a finite state automaton. The fuzzy plan language described in section 3.4 is used to perform any replanning required.

As contingencies arise the preconditions for the next action step in the plan are evaluated to gauge the extent of their effects and to verify that the remaining steps are completed successfully [21]. The Internet environment is uncertain with unforeseen changes taking place in the current state, so a contingency in this context is considered to be any change in environmental conditions that may affect the execution of the current plan [21]. For example, if there is a change in the usage cost of a host which does not need to be visited, then the current plan is not affected. However, if a host that needs to be visited changes its location, then the plan is affected. Depending on the quantum of change and its ramifications, a new plan may have to be generated. The characterization of a change in conditions as being substantial will depend considerably on the weight assigned to each parameter. For example, if the usage cost of hosts is immaterial to the user then even a considerable hike in the cost of using a computing host will not affect the plan.

The Site Selector (SS) and Knowledge Base (KB) are consulted while computing the new plan. Modify-Plan() (adapted from [21]) in Algorithm 1 keeps track of the remaining plan segment p and the complete plan q. The goal, G, is that the query has to be computed efficiently. The algorithm checks the current state of the environment as reported by the Sensor. If it finds that some preconditions for p are not met then the function Choose-Best-Continuation() is initiated which chooses a point in q such that there is a plan $p\prime$ that is the easiest to achieve from that point to the end of q. Now the new plan must achieve the preconditions of $p\prime$ and then execute it. The new plan is sent to the Action Effector for appropriate action.

In the example, the Decision Maker finds that the preconditions to Step 3, CloseTo(A,B) and CloseTo(C,B) fail. This means that Step 3 cannot be carried out and the Modify-Plan() algorithm is initiated. The Choose-Best-Continuation() algorithm finds that Step 6 is the point from where the rest of the plan $p\prime$ is easiest to achieve. Now the new plan must achieve the preconditions of Step 6. To be able to achieve this, it must create a new action step 'Go to Host C' with preconditions 'RelConnBet(A,C)' and 'At(M,A)'. This is appended to $p\prime$. The preconditions to this step are evaluated to see if they can be achieved. From the Knowledge Base, it finds that the connection between A and C is reliable and it also knows that the MQC is at A. This means

Algorithm 1 Algorithm to Modify Plan (Replan) [21]

```
Input: Change in parameter value
Output: New Plan

function MODIFY-PLAN (change-in-parameter-value)
   reference to: KB, the Knowledge Base,
                 SS, the Site Selector
   static: p, an annotated plan, initially NoPlan
           q, an annotated plan, initially NoPlan
           G, goal
current <- CURRENT-STATE(KB, t)
if p = NoPlan then
   p <- CREATE-PLAN(current, G, KB, SS)
   q <- p
   if p = NoPlan or p is empty then return NoOp
if PRECONDITIONS(p) not currently true in KB then
   p′ <- CHOOSE-BEST-CONTINUATION (current, q)
   p <- APPEND (CREATE-PLAN (current, PRECONDITIONS(p′), KB, SS, p′)
   q <- p
action <- FIRST (p)
p <- REST (p)
return q
```

that the preconditions are satisfied and the new plan is complete.

The thin solid line in Figure 2, shows the route as per the new plan generated by the Decision Maker.

4 Conclusion and Future Work

This paper presents a framework for a system that incorporates concepts from fuzzy logic into a plan language to formulate and execute a plan for a mobile query computation that will process a large query efficiently in an uncertain environment like the Internet.

The two technologies, fuzzy sets and databases, and mobile computation, incorporated here are still in their infancy and are each used in a very limited way. More work is required in each of these areas before such a system can become commercially viable. We have implemented a system that generates and manages mobile computations in a heterogeneous environment. Future work will include implementing the proposed query processing system in a test environment and further developing the algorithm for the selection of the computation sites to reflect the real-world environment.

References

[1] "Mobile agent computing." White Paper, Horizon Systems Laboratory, Mitsubishi Electric ITA, Jan. 1998. http://www.meitca.com/HSL/Projects/ Concordia/documents.htm.

[2] Cardelli, L., "Mobility and security." Lecture Notes for the Marktoberdorf Summer School, 1999 from the works by Luca Cardelli and Andrew D. Gordon, 1999. Microsoft Research.

[3] Brewington, B., Gray, R., Moizumi, K., Kotz, D., Cybenko, G., and Rus, D., *Mobile agents in distributed information retrieval*, ch. 15, pp. 355–395. Intelligent Information Agents, Springer Verlag, 1999.

[4] Conde, J., "Mobile agents in java." http://wwwinfo.cern.ch/asd/rd45/white-papers/9812/agents2.html, Dec. 1998. CERN/IT/ASD/RD45/98/12.

[5] Thorn, T., "Programming languages for mobile code," *ACM Computing Surveys*, vol. 29, pp. 213–239, Sept. 1997.

[6] Rasmussen, D. and Yager, R., "Summary sql - a fuzzy tool for data mining," *Intelligent Data Analysis*, 1997.

[7] Yazici, A. and George, R., *Fuzzy Database Modelling*, vol. 26 of *Studies in fuzziness and soft computing*, ch. 4. Logical database models for uncertain data, pp. 113 – 214. Physica-Verlag, 1999.

[8] Pulimood, S., “Efficient support for mobile computations,” Tech. Rep. TR-EECS-00-52, Tulane University, Dec. 2000.

[9] Cardelli, L., “Mobile computational ambients,” tech. rep., Digital Equipment Corporation, 1997.

[10] Cardelli, L. and Gordon, A. D., “Mobile ambients,” in *FOSSACS’98, International Conference on Foundations of Software* (Nivat, M., ed.), vol. 1378 of *Lecture Notes in Computer Science*, pp. 140–155, Springer-Verlag, 1998.

[11] Murase, T., Tsukamoto, M., and Nishio, S., “Active mobile database systems for mobile computing environments,” *IEEE Transactions Information and Systems*, vol. E81-D, pp. 427 – 433, May 1998.

[12] Saygun, Y., Ulusoy, O., and Yazici, A., “Dealing with fuzziness in active mobile database systems,” *Information Sciences*, vol. 120, pp. 23 – 44, Nov. 1999.

[13] Imielinski, T. and Badrinath, B. R., “Querying in highly mobile distributed environments,” in *The 18th VLDB Conference, Vancouver, British Columbia, Canada*, 1992.

[14] Papastavrou, S., Samaras, G., and Pitoura, E., “Mobile agents for www distributed access,” 1999.

[15] Leiner, B., Cerf, V., Clark, D. D., Kahn, R., Kleinrock, L., Lynch, D., Postel, J., Roberts, L., and Wolff, S., “A brief history of the internet.” http://www.isoc.org/internet-history/brief.html, Feb. 1998. Last updated Aug 2000.

[16] Vitek, V. and Castagna, G., “Seal: A framework for secure mobile computations,” in *Internet Programming Languages, ICCL’98 Workshop, Chicago, IL, USA* (Bal, H. E., Belkhouche, B., and Cardelli, L., eds.), vol. 1686 of *Lecture Notes in Computer Science*, pp. 47 – 77, Springer, May 1998.

[17] Dean, D., “The security of static typing with dynamic linking,” *Fourth ACM Conference on Computer and Communications Security*, Apr. 1997.

[18] Volpano, D., “Provably-secure programming languages for remote evaluation,” tech. rep., Computer Science Dept., Naval Postgraduate School, Monterey, CA, 1997.

[19] Volpano, D. and Smith, G., “On the systematic design of web languages,” *ACM Computing Surveys*, vol. 28, pp. 315–317, June 1996.

[20] Russell, S. J. and Norvig, P., *Artificial Intelligence: A modern approach*. Artificial Intelligence, Prentice-Hall, Inc., 1995.

[21] Russell, S. J. and Norvig, P., *Artificial Intelligence: A modern approach*, ch. Planning and Acting, p. 402. Artificial Intelligence, Prentice-Hall, Inc., 1995.