

Specification – Based Approach for Implementing Atomic Read/ Write Shared Memory in Mobile Ad Hoc Networks Using Fuzzy Logic

Sherif El-etriby¹ and Reham Shihata²

¹ Faculty of Computers and Information,
Computer Science Dept., Menoufiya University, Egypt
sherif.el-etriby@ci.menofia.edu.eg

² Faculty of Science, Math's. Dept., Menoufiya University, Egypt
rehamteacher@yahoo.com

Abstract. In this paper we propose an efficient fuzzy logic based solution for the specification and performance evaluation depending on generation of fuzzy rules. A new property matching mechanism is defined. The requirement with attributes is channelled in the following manner: the basic functionality is ensured, matching properties names according to the classical reading/writing strategy. The preliminary solutions are selected and hierarchies according to the degree of attribute matching. Consequently, we describe the basic principles of the proposed solutions and illustrate them for implementing atomic read write shared memory in mobile ad hoc network. This is done by fuzzy logic, which is considered a clear solution to illustrate the results of this application in distributed systems. The results are approximate but also, they are very good and consistent with the nature of this application.

Keywords: Specification, Distributed Systems, Mobile Ad Hoc Network, Fuzzy Logic.

1 Introduction

A software system is viewed as a set of components that are connected to each other through connectors. A software component is an implementation of some functionality, available under the condition of a certain contract, independently deployable and subject to composition. In the specification approach, each component has a set of logical points of interaction with its environment. The logic of a component composition (the semantic part) is enforced through the checking of component contracts. Components may be simple or composed [1] [11]. A simple component is the basic unit of composition that is responsible for certain behavior. Composed components introduce a grouping mechanism to create higher abstractions and may have several inputs and outputs. Components are specified by means of their provided and required properties. Properties in this specification approach are facts known about the component. A property is a name from a domain vocabulary set and may have refining sub-properties (which are also properties) or refining attributes that

are typed values [1]. The component contracts specify the services provided by the component and their characteristics on one side and the obligations of client and environment components on the other side. The provided services and their quality depend on the services offered by other parties, being subject to a contract. A component assembly is valid if it provides all individual components are respected. A contract for a component is respected if all its required properties have found a match. The criterion for a semantically correct component assembly is matching all required properties with provided properties on every flow in the system [11]. In this specification approach, it is not necessary that a requirement of a component is matched by a component directly connected to it. It is sufficient that requirements are matched by some components that are presented on the flow connected to the logical point; these requirements are able to propagate.

2 Fuzzy Attributes

A property consists of a name describing functionality and attributes that are either type values or fuzzy terms. The names used for the properties and for the attributes are established through a domain-specific vocabulary[2][11]. Such a restriction is necessary because a totally free-text specification makes the retrieval difficult, producing false- positive or false-negative matching due to the use of a non-standard terminology[2][11]. In this work, the domain specific vocabulary must also describe the domains of the fuzzy attributes (linguistic variables) for each property as well as the membership functions for the fuzzy terms. The membership functions for all linguistic variables are considered of triangular shape as shown in Fig.1.

For each linguistic variable, first the number and the names of the terms of its domain must be declared, and after that the values of the parameters $a_1, a_2 \dots a_n$ must be specified.

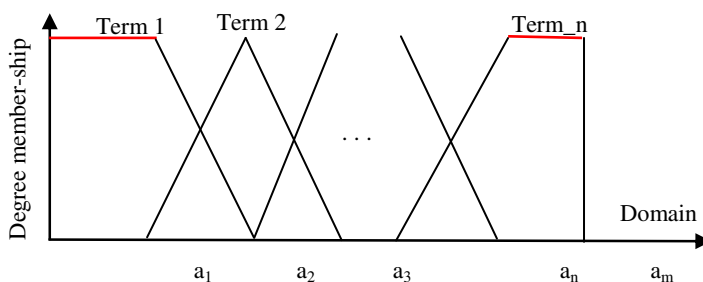


Fig. 1. The Shape of the Membership Functions

3 Implementing Atomic Read/Write Shared Memory

In this paper the Geoquorum approach has presented for implementing atomic read/write shared memory in mobile ad hoc networks. This approach is based on

associating abstract atomic objects with certain geographic locations. It is assumed that the existence of Focal Points, geographic areas that are normally "populated" by mobile nodes. For example: a focal point may be a road Junction, a scenic observation point. Mobile nodes that happen to populate a focal point participate in implementing a shared atomic object, using a replicated state machine approach. These objects, which are called focal point objects, are prone to occasional failures when the corresponding geographic areas are depopulated [3]. The Geoquorum algorithm uses the fault-prone focal point objects to implement atomic read/write operations on a fault-tolerant virtual shared object. The Geoquorum algorithm uses a quorum- based strategy in which each quorum consists of a set of focal point objects. The quorums are used to maintain the consistency of the shared memory and to tolerate limited failures of the focal point objects, which may be caused by depopulation of the corresponding geographic areas. The mechanism for changing the set of quorums has presented, thus improving efficiency. In general, the new Geoquorum algorithm efficiently implements read/write operations in a highly dynamic, mobile network. In this study the basic idea for the proposed approach is an ad hoc network uses no pre-existing infrastructure, unlike cellular networks that depend on fixed, wired base stations. Instead, the network is formed by the mobile nodes themselves, which cooperate to route communication from sources to destinations. Ad hoc communication networks are by nature, highly dynamic. Mobile nodes are often small devices with limited energy that spontaneously join and leave the network. As a mobile node moves, the set of neighbors with which it can directly communicate may change completely. The nature of ad hoc networks makes it challenging to solve the standard problems encountered in mobile computing, such as location management using classical tools. The difficulties arise from the lack of a fixed infrastructure to serve as the backbone of the network [3] [4]. Atomic memory is a basic service that facilitates the implementation of many higher level algorithms. For example: one might construct a location service by requiring each mobile node to periodically write its current location to the memory. Alternatively, a shared memory could be used to collect real – time statistics, for example: recording the number of people in a building here, a new algorithm for atomic multi writes/multi- reads memory in mobile ad hoc networks.

The problem of implementing atomic read/write memory is explained as we define a static system model, the focal point object model that associates abstract objects with certain fixed geographic locales. The mobile nodes implement this model using a replicated state machine approach [3] [4]. In this way, the dynamic nature of the ad hoc network is masked by a static model. Moreover, it should be noted that this approach can be applied to any dynamic network that has a geographic basis. The implementation of the focal point object model depends on a set of physical regions, known as focal points. The mobile nodes within a focal point cooperate to simulate a single virtual object, known as a focal point object. Each focal point supports a local broadcast service, LBCast which provides reliable, totally ordered broadcast. This service allows each node in the focal point to communicate reliably with every other node in the focal point. The focal broadcast service is used to implement a type of replicated state machine, one that tolerates joins and leaves of mobile nodes. If a focal

point becomes depopulated, then the associated focal point object fails [4]. (Note that it doesn't matter how a focal point becomes depopulated, be it as a result of mobile nodes failing, leaving the area, going to sleep, etc. Any depopulation results in the focal point failing). The Geoquorum algorithm implements an atomic read/write memory algorithm on top of the geographic abstraction, that is, on top of the focal point object model. Nodes implementing the atomic memory use a Geocast service to communicate with the focal point objects. In order to achieve fault tolerance and availability, the algorithm replicates the read/write shared memory at a number of focal point objects. In order to maintain consistency, accessing the shared memory requires updating certain sets of focal points known as quorums.

An important aspect of our approach is that the members of our quorums are focal point objects, not mobile nodes [3] [4]. The algorithm uses two sets of quorums (I) **get-quorums** (II) **put-quorums** with property that every get-quorum intersects every put-quorum. There is no requirement that put-quorums intersect other put-quorums, or get-quorums intersect other get-quorums. The Put/Get quorums implementing atomic read / write shared memory in mobile ad hoc networks shown in Fig. 2.

The use of quorums allows the algorithm to tolerate the failure of a limited number of focal point objects. Our algorithm uses a Global Position System (GPS) time service, allowing it to process write operations using a single phase, prior single-phase write algorithm made other strong assumptions, for example: relying either on synchrony or single writers [3][4]. This algorithm guarantees that all read operations complete within two phases, but allows for some reads to be completed using a single phase.

The atomic memory algorithm flags the completion of a previous read or write operation to avoid using additional phases, and propagates this information to various focal point objects[3][4]. As far as we know, this is an improvement on previous quorum based algorithms. For performance reasons, at different times it may be desirable to use different times it may be desirable to use different sets of get quorums and put-quorums.

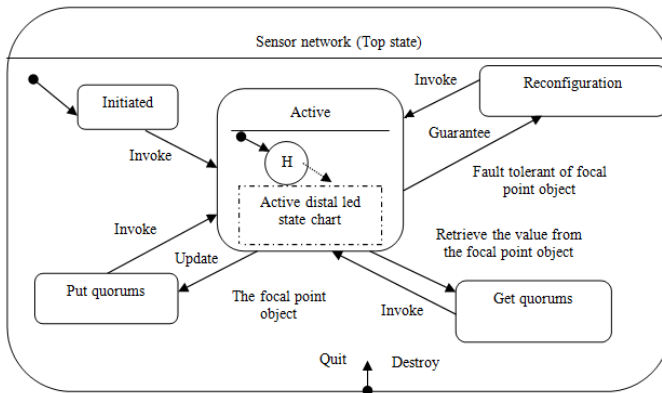


Fig. 2. Put/Get quorums implementing atomic read / write shared memory in mobile ad hoc networks

4 The Specification of the Geoquorum Approach Using Fuzzy Logic

A component repository contains several implementations of components that have the functionality of the application, specified with the provided property reading/writing in mobile ad hoc networks. Let us considered two different components, C1 and C2, specified as follows:

Component C1:

Property reading / writing with attributes

Read/write_ACK_rate = Crisp(0.2)

Read/write_ACK_rate = Crisp(0.4)

Occurrence = fuzzy (connect, about right, Almost no-connect)

Component C2:

Property reading / writing with attributes

Read/write_ACK_rate = Crisp(0.6)

Read/write_ACK_rate = Crisp(0.8)

Occurrence = fuzzy (connect, about right, Almost no connect)

Each of these attributes is defined as a linguistic variable with these terms as follows:

Domain(read/write_ACK_status)={ACK_response,no change is needed,Almost no response}

Domain(occurrence)= {connect, about right, almost no connect}

For each linguistic variable set of the parameters a_1, a_2, a_3 defining the shape of the membership functions are defined. In our application, in case of the attribute reading/writing, these values are ($a_1=0.2$), ($a_2=0.4$), ($a_3=0.6$), ($a_4=0.8$), and random values are ($a_5=0.1$), ($a_6=0.3$). It is important to note that a linguistic variable that characterizes an attribute can have different meanings in the context of different properties. The domain and the shape of a linguistic variable can be redefined in the context of different properties.

4.1 Generation of Fuzzy Rules

A new property matching mechanism is defined. In general, a requirement as: Requirement property P with attributes $A1 = V1$ and $A2 = V2$ and $A_n = V_n$ is handled in the following manner: First, the basic functionality is ensured, matching properties names according to the classical reading/writing strategy. Usually several solutions result from this first step. Second, the preliminary solutions are selected and hierarchies according to the degree of attribute matching [5] [6] [11]. This is done by fuzzy logic. The given requirement is translated into the corresponding rule:

If $A1=V1$ and $A2=V2$ and ... $A_n=V_n$ then decision = select

The generation of the fuzzy rules is done automatically starting from the requirements. Very often, the required attributes are not values, but rather are required to be at least (or at most) a given value, $A \geq V$ or $A \leq V$. In general, a requirement containing the attribute expression $A \geq V$ will be translated into a set of I rules, for all $V_i \geq V$: If $A=V_i$ then decision = select

4.2 Extension of the Fuzzy Rules

Several rules are generated from one requirement. In order to relax the selection, it is considered a match even if one of the linguistic variables in the premises matches only a neighbor of the requested value (the predecessor or the successor) [5] [6] [11]. In this case the decision of selection is a weak one. In the case that more than one linguistic variable in the premise matches only neighbor values (while the rest match the requested fuzzy terms); the decision is a weak reject. In the extreme case that all linguistic variables in the premises match neighbor values, the decision is a weak reject [5] [7]. In all the other cases, the decision is a strong reject. For example, in the case of a requirement containing two attributes, $A1=V1$ and $A2=V2$, the complete set of generated rules is [7] [8]:

The directly generated rule is:

If $A1=V1$ and $A2=V2$ then decision=strong_ select

The rules generated if one of the linguistic variables in the premises matches only a neighbor of the requested value are:

If $A1 = \text{pred}(V1)$ and $A2=V2$ then decision = weak _ select

If $A1 = \text{succ}(V1)$ and $A2=V2$ then decision =weak _select

If $A1 = V1$ and $A2 = \text{pred}(V2)$ then decision = weak _ select

If $A1 = V1$ and $A2 = \text{succ}(V2)$ then decision =weak _select

In this case there are a maximum number of four generated rules for instance. If neither $V1$ nor $V2$ are extreme values of their domains, if a value is the first value in the domain it has no predecessor, if it is the last value in the domain it has no successor. The rules generated if more than one of the linguistic variables in the premises matches only a neighbor of the requested value are (maximum 4 rules):

If $A1= \text{pred}(V1)$ and $A2 = \text{pred}(V2)$ then decision =weak_ reject

If $A1= \text{succ}(V1)$ and $A2 = \text{pred}(V2)$ then decision =weak_ reject

If $A1= \text{pred}(V1)$ and $A2 = \text{succ}(V2)$ then decision =weak_ reject

If $A1= \text{succ}(V1)$ and $A2 = \text{succ}(V2)$ then decision =weak_ reject

For all the rest of possible combinations of values of $A1$ and $A2$ the decision is strong-reject [11][14] [15].

4.3 Specifying the Application Using Fuzzy Rules

Let the rules generated for one different neighbor are:

[R1]If read/ write_ Ack_ status = Almost_ no response and occurrence = about right then decision = weak_ select.

[R2]If read/ write_ Ack_ status = Ack_ response and occurrence = about right then decision = weak_ select.

[R3]If read/ write_ Ack_ status = Ack_ response and occurrence = Almost no_ connect then decision = weak_ select.

[R4] If read/ write_ Ack_ status = no change need and occurrence = connect then decision = weak_select

The rules generated for two different neighbors are:

[R5] If read/write_Ack_status=Almost_no response and occurrence = almost no-connect then decision = weak_reject.

[R6] If read/write_Ack_status=Ack_response and occurrence = almost no_connect then decision = weak_reject.

[R7] If read/write_Ack_status=Almost no response and occurrence = connect then decision =weak_reject.

[R8] If read/write_Ack_status=Ack_response and occurrence = connect then decision = strong_select.

The method has been implemented using java programming, the code is consists of four phases: Analysis phase, Specification phase, Design phase, and Test phase. This paper proposes an efficient fuzzy logic based solution for the specification and performance evaluation depending on generation of fuzzy rules. As shown in Fig.3 to Fig. 6 we are discussing samples at instant values with a resulting controller output; the controller is sampling several times each second with a resulting “correction” output following each sample. So, we introduce a specification approach for the Geoquorums approach for implementing atomic read/write shared memory in mobile ad hoc networks which is based on fuzzy logic. The advantages of this solution are: a natural treatment for certain non-functional attributes that cannot be exactly evaluated and specified, and a relaxed matching of required/provided attributes that do not have to always be precise (see Fig. 9). Fig 3 -6 (a, b, c, d) illustrate how each of the generated rules is composed with the fact represented by the specification of component c_1 (with read/ write- Ack- rate= 0.1, 0.3, 0.4, 0.8 and occurrence= Almost no connect).

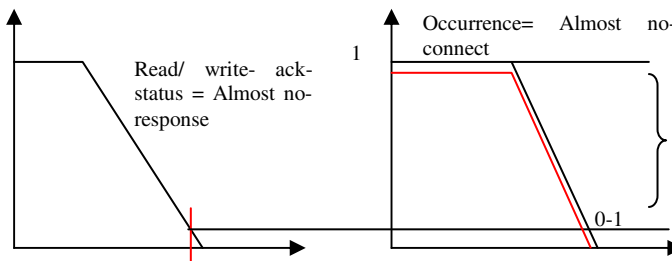


Fig. 3a. Rule: If read/ write- Ack- status = (Almost- no response) and occurrence= (almost-no-connect) then decision = weak-reject. Facts: read/ write- ack- rate= 0.1.

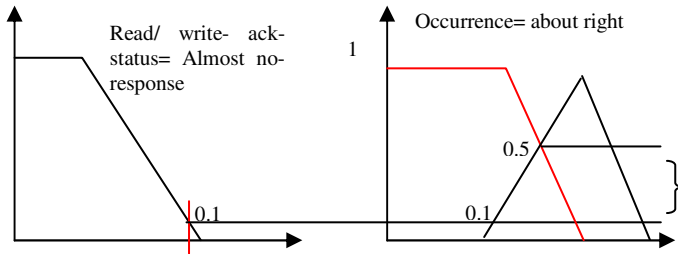


Fig. 3b. Rule: If read/ write- Ack-status = Almost no-response and occurrence= about right then decision= weak-reject. Facts: read/ write- Ack-rate= 0.1.

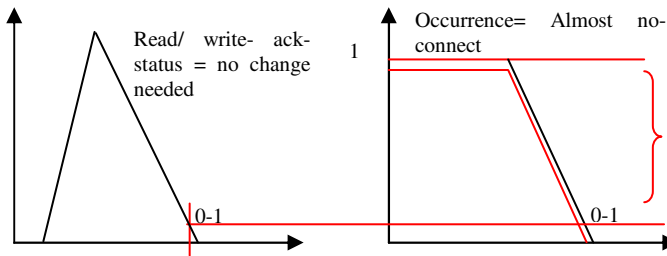


Fig. 3c. Rule: If read/ write-Ack-status = no change needed and occurrence=Almost no-connect then decision= weak-reject facts: read/ write- ack- rate= 0.1

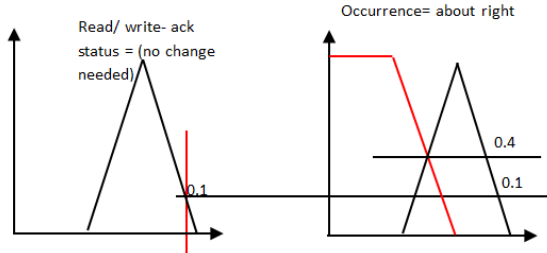


Fig. 3d. Rule: If read/ write- Ack-status = no change needed and occurrence= about right then decision= weak-reject. Facts: read/ write- Ack-rate= 0.1.

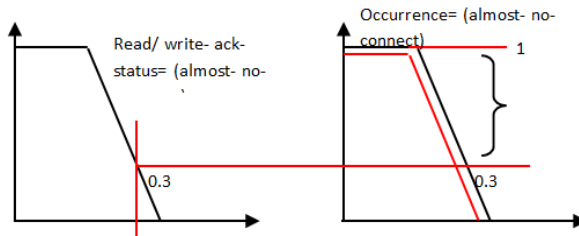


Fig. 4a. Rule: If read/ write-Ack-status = (Almost no- response) and occurrence= Almost no-connect then decision= weak-reject Facts: read/ write-Ack-rate= 0.3.

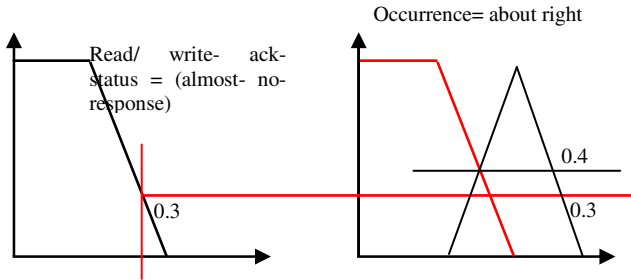


Fig. 4b. Rule: If read/write-Ack – status = (Almost no- response) and occurrence= about right then decision= weak–select. Facts: read/ write- Ack-rate= 0.3.

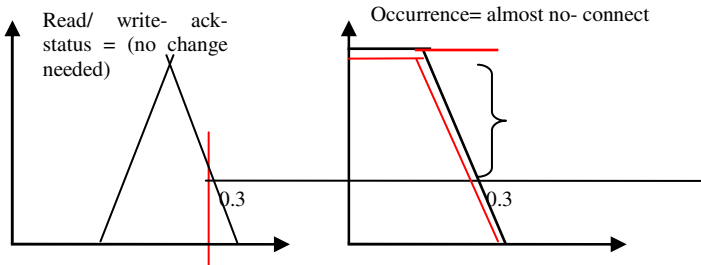


Fig. 4c. If read/ write- Ack – status= no change needed and occurrence= Almost no- connect then decision= weak–reject. Facts: read/ write- Ack- rate= 0.3.

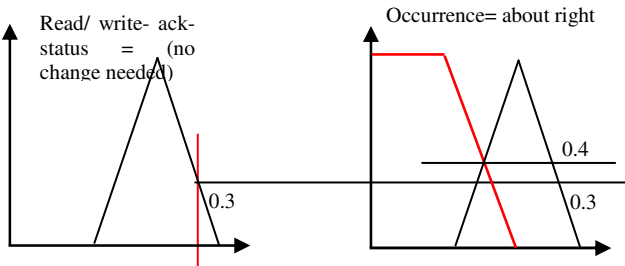


Fig. 4d. Rule: If read/ write- Ack – status = no change needed and occurrence= about right then decision= strong- select. Facts: read/ write- Ack- rate= 0.3.

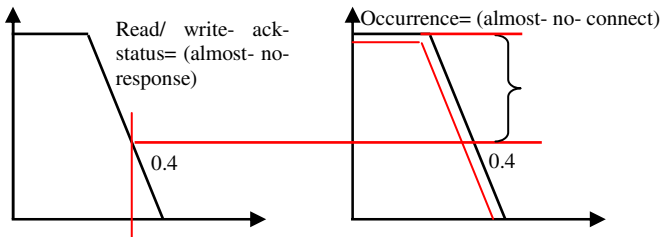


Fig. 5a. Rule: If read/ write-Ack – status= (Almost no- response) and occurrence= Almost no- connect then decision= weak–reject facts: read/ write- Ack-rate= 0.4

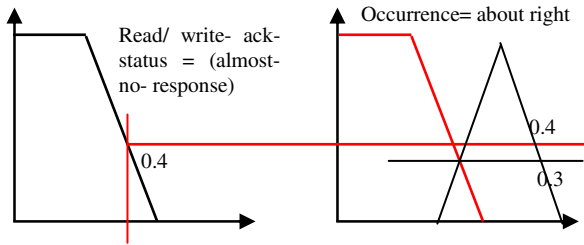


Fig. 5b. Rule: If read/write- Ack-status = (Almost no- response) and occurrence= about right then decision= weak -reject. Facts: read/write- Ack- rate= 0.4

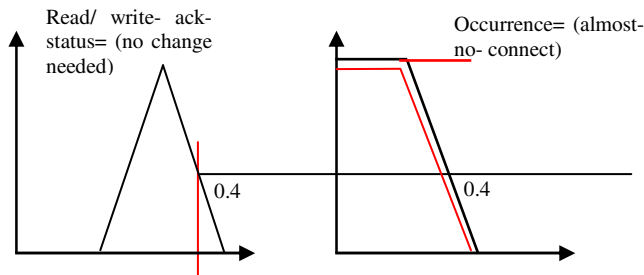


Fig. 5c. Rule: If read/ write- Ack-status = no change needed and occurrence= Almost no- connect then decision= weak-reject Facts: read/ write- Ack- rate= 0.4

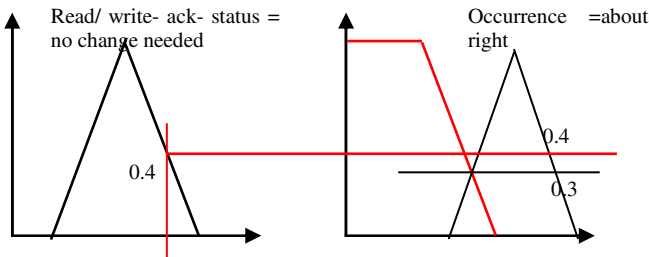


Fig. 5d. Rule: If read/ write- Ack-status = no change needed and occurrence about right then decision= strong-select. Facts: read/ write- Ack- rate= 0.4

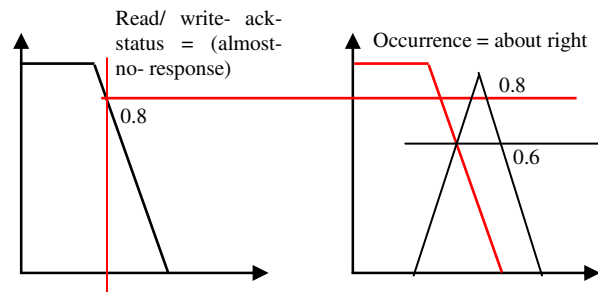


Fig. 6a. Rule: If read/ write- Ack-status= (Almost no- response) and occurrence= Almost no- connect then decision= weak-select Facts: read/ write- Ack- rate= 0.8

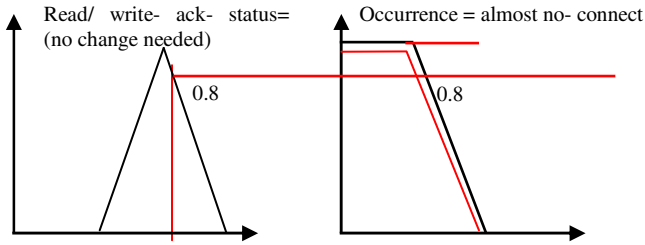


Fig. 6b. Rule: If read/ write- Ack-status= (Almost no- response) and occurrence= about right then decision= weak-select. Facts: read/write-Ack-rate= 0.8.

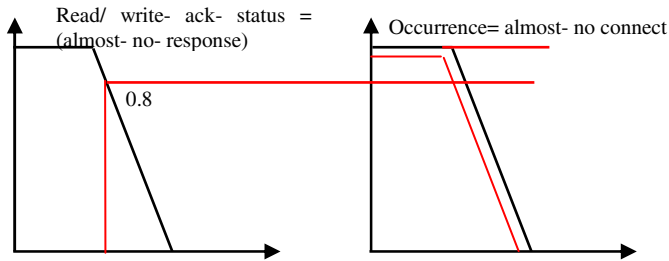


Fig. 6c. Rule: If read/ write-Ack – status= no change needed and occurrence= Almost no- connect then decision= weak-select Facts: read/write-Ack- rate= 0.8.

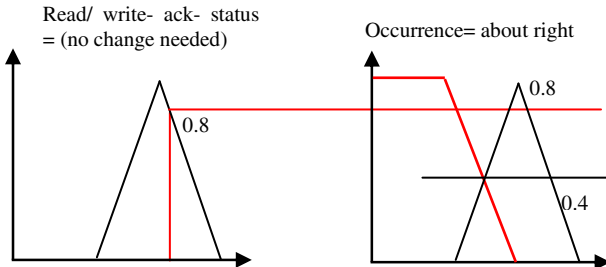


Fig. 6d. Rule: If read/write-Ack-status = no change needed and occurrence= about right then decision= weak-reject. Facts: read/ write-Ack-rate= 0.8.

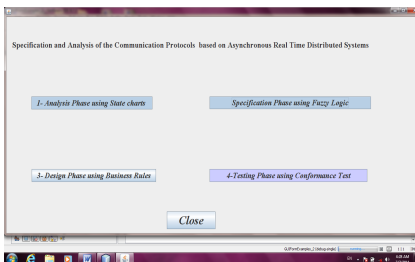


Fig. 7. First interface of the software development process

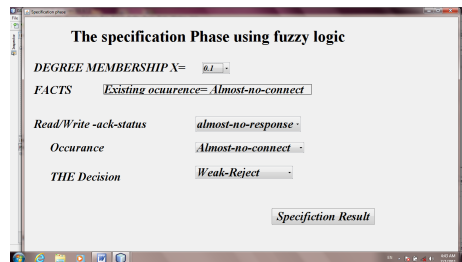


Fig. 8. The interface of the specification phase

Table 1. GUI bottoms and its corresponding results

Bottoms	Results
<ul style="list-style-type: none">Read/write- ack- status	<ul style="list-style-type: none">Almost no response, no change needed
<ul style="list-style-type: none">Occurrence	<ul style="list-style-type: none">Almost no connect, about right
<ul style="list-style-type: none">The decision	<ul style="list-style-type: none">Weak reject, strong select, weak select
<ul style="list-style-type: none">Specification result	<ul style="list-style-type: none">The specification phase is completed successfully

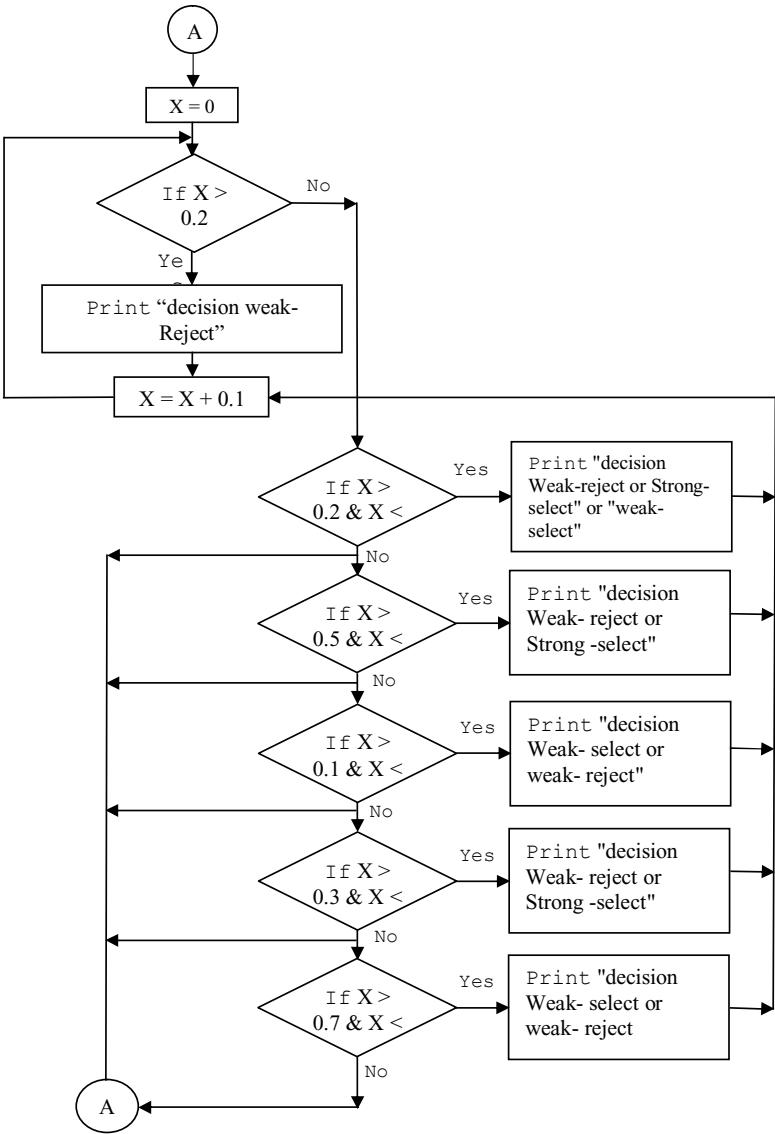


Fig. 9. Flow Chart for Specification Phase for Implementing Atomic Read/Write Shared Memory in Mobile Ad Hoc Networks

Consequently, the results of the previous figures (Fig. 3 to Fig. 6) are determined as follow: the status of read/write operation is always (almost no response or no change needed of connection), the future occurrence of connection is always (almost no connect or about right of the connection in the network connection). Also, in these figures we have two facts: the current status of occurrence is almost no connect and according to the fuzzy logic we assume values X which are determined at range from 0 to 1. So, the results of the connection by the network is either *weak-reject* or *weak-select*. Also, we can observe that when X is nearly 0.4 and the read/write-ack-status is no change needed, the current occurrence of connection is about right and the result of connection is may be *strong-select*, all possible output cases are shown in Table 1. The snapshot of the GUI is illustrated in Fig.7 and Fig. 8.

5 Conclusions

In this paper we introduce a specification Geoquorums approach for implementing atomic read/write shared memory in mobile ad hoc networks which is based on fuzzy logic. The advantages of this solution are: a natural treatment for certain non-functional attributes that can not be exactly evaluated and specified. In addition to a relaxed matching of required/provided attributes that do not have to always be precise.

References

1. Bachman, F., Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J., Seacord, R., Wallnau, K.: Technical concepts of component-based software engineering. Technical Report CMU/SEI-2000-TR-008, Carnegie Mellon Software Engineering Institute (2000)
2. Cooper, K., Cangussu, J.W., Lin, R., Sankaranarayanan, G., Soundararadjane, R., Wong, E.: An Empirical Study on the Specification and Selection of Components Using Fuzzy Logic. In: Heineman, G.T., Crnković, I., Schmidt, H.W., Stafford, J.A., Ren, X.-M., Wallnau, K. (eds.) CBSE 2005. LNCS, vol. 3489, pp. 155–170. Springer, Heidelberg (2005)
3. Dolv, S., Gilbert, S., Lynch, N.A., Shvartsman, A.A., Welch, A., Loran, J.L.: Geoquorums: Implementing Atomic Memory in Mobile Ad Hoc Networks. In: Proceedings of the 17th International Conference on the Distributed Computing, pp. 306–319 (2005)
4. Haas, Z.J., Liang, B.A., Wjghs, D.: Ad Hoc Mobile Management with Uniform GeoQuorums Systems. Proceeding of IEEE/ACM Transactions on Mobile Ad Hoc Networks 7(2), 228–240 (2004)
5. Koyuncu, M., Yazici, A.: A Fuzzy Knowledge-Based System for Intelligent Retrieval. IEEE Transactions on Fuzzy Systems 13(3), 317–330 (2005)
6. Sora, I., Verbaeten, P., Berbers, Y.: A Description Language For Composable Components. In: Pezzé, M. (ed.) FASE 2003. LNCS, vol. 2621, pp. 22–36. Springer, Heidelberg (2003)
7. Şora, I., Creţu, V., Verbaeten, P., Berbers, Y.: Automating Decisions in Component Composition Based on Propagation of Requirements. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 374–388. Springer, Heidelberg (2004)

8. Sora, I., Cretu, V., Verbaeten, P., Berbers, Y.: Managing Variability of Self-customizable Systems through Composable Components. *Software Process Improvement and Practice* 10(1) (January 2005)
9. Szyperski, C.: *Component Software: Beyond Object Oriented Programming*. Addison Wesley (2002)
10. Zhang, T., Benini, L., De Micheli, G.: Component Selection and Matching for IP-Based Design. In: *Proceedings of Conference on Design, Automation and Test in Europe (DATE)*, Munich, Germany, pp. 40–46 (2001)
11. Şora, I., Todinca, D.: Specification-based Retrieval of Software Components through Fuzzy Inference. *Acta Polytechnica Hungarica* 3(3) (2006)
12. Oliveira, R., Bernardo, L., Pinto, P.: Modeling delay on IEEE 802.11 MAC protocol for unicast and broadcast non saturated traffic. In: *Proc. WCNC 2007*, pp. 463–467. IEEE (2007)
13. Fehnker, A., Fruth, M., McIver, A.K.: Graphical Modelling for Simulation and Formal Analysis of Wireless Network Protocols. In: Butler, M., Jones, C., Romanovsky, A., Troubitsyna, E. (eds.) *Fault Tolerance*. LNCS, vol. 5454, pp. 1–24. Springer, Heidelberg (2009)
14. Ghassemi, F., Fokkink, W., Movaghar, A.: Restricted broadcast process theory. In: *Proc. SEFM 2008*, pp. 345–354. IEEE (2008)
15. Ghassemi, F., Fokkink, W., Movaghar, A.: Equational Reasoning on Ad Hoc Networks. In: Arbab, F., Sirjani, M. (eds.) *FSEN 2009*. LNCS, vol. 5961, pp. 113–128. Springer, Heidelberg (2010)
16. Lin, T.: *Mobile Ad-hoc Network Routing Protocols: Methodologies and Applications*. PhD thesis, Virginia Polytechnic Institute and State University (2004)
17. Tracy Camp, V.D., Boleng, J.: A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing* 2, 483–502 (2002)