CReaM: User-Centric Replication Model for Mobile Environments

Zeina Torbey, Nadia Bennani, Lionel Brunie
Université de Lyon, CNRS
INSA-Lyon, LIRIS
Lyon, France
{zeina.torbey, nadia.bennani, lionel.brunie}@insa-lyon.fr

David Coquil
Chair of Distributed Information Systems
University of Passau
Passau, Germany
david.coquil@uni-passau.de

Abstract: **Data replication can improve data availability in mobile networks; but due to typical resource limitations in such environments, specific replication mechanisms are needed. In this paper, we propose CReaM, user-Centric REplicAtion Model for mobile environment. This model puts users at the centre by letting them determine the amount of resources they are willing to share. CReaM is suitable for dynamic environments where each node needs a certain level of decision autonomy. In this paper, we present CReaM's general principles and focus on one of its core function, which is its autonomic behavior that generates replication requests based on resources monitoring and user settings.**

Keywords: Mobile Ad-Hoc Network, data availability, replication, user-centric.

I. Introduction

A mobile Ad hoc NETwork (MANET) is a self-configuring network; it consists of nodes communicate in an autonomous way without any centralized server; most often, these networks are deployed on devices with limited resources. MANETs have to face the problem of frequent networks topology changes that may lead to unanticipated network partitioning and cause incomplete data transfers between the nodes. In this context, data replication is one possible solution to maintain data availability, which is mandatory precondition for wide-spread deployment of distributed applications in this context.

Applying data replication in a decentralized manner is not a trivial task for many reasons: (1) the network topology changes frequently and unexpectedly (2) the data to be replicated must be carefully selected because of the limited storage space of the devices. (3) The data may be updated so mechanisms for maintaining data consistency are necessary. In addition, (4) the power of the device is limited, which means efficient methods must be adopted to reduce the communication between nodes. In view of all these issues specific data replication mechanisms for MANETs is required.

The proposed replication models replicate data by taking into account resource availability and the access frequency of shared data items. Most of them replicate periodically the important data items and place the replica on devices with the most available resources. Such top-down approaches may lead to an abuse of user resources, as they might go so far as to use all of a user's resources in order to reach their goal. To the contrary, we feel that users should be at the center of the systems. Hence we propose a user-centric approach in which users may are in control of the amount of resources that they share; these resources are then used to enhance data availability, while the rest of the resources are reserved for the users to be able to accomplish their tasks.

To illustrate the motivation of our work, let us consider the following scenario: George, Steve and Marie are three users waiting in the bus station and connecting to the MANET there. George is working on a laptop; Steve is using an advanced mobile device to download music, while Marie uses an old mobile phone to check the bus schedule. According to previous work criteria, the laptop would be selected for replica placement, because it has the highest resources capacity compared to the other devices. However, from another point of view, Marie’s device is a better choice even with its low resources, simply because she is not using her phone while George needs its device's resources to achieve his work. In case Marie receives a call or she needs her resources to execute a task, the replication mechanism should evolve cleverly to let Marie use her device in optimal conditions and choose dynamically an alternative target device(s) to hold the replica.

Several challenges need to be overcome to develop such a system. First, the system must react dynamically to the resources’ consumption on each node to keep all users satisfied. However, reacting each time a change occurs might be ineffective. It is therefore necessary to identify the right factors on which to react as well as the right granularity of reaction. Furthermore, if the system react starting from the users’ need, another challenge appears: the system must take local decisions to satisfy the individual need, but it must also consider the interest of the whole system. Finding a balance between these two factors is necessary to avoid problems like replica duplication, network saturation and free riding.

To address these issues, we propose the user Centric REplicAtion Model (CReaM). This model places users in the centre by letting them define their level of participation in the system. Thus, the model operates with respect to the user desire; it replicates automatically when the user is overloaded and places replica on other users’ devices that can support the load. Thus, the user participates in the system which is, in our view, a key requirement of a realistic approach. To do so, our model is based on a monitoring mechanism that gathers locally and periodically the consumption of peer features like memory, battery, etc, and attributes a status to the peer that reflects the user activity level implied by the monitored values. Each status conditions the peer's local decision about whether to accept or reject other peers' replica demands and about what data to replicate if its activity is increasing.

CReaM is suitable for highly mobile environments where network topology changes rapidly. Indeed, replication decisions in CReaM are made in a totally distributed manner. Each node has a certain level of decision autonomy; it decides to replicate data starting from its tolerance to answer other nodes' queries. Our solution gives a priority to the user needs but considers also the global network status and the global data distribution for an application in a MANET context. In this paper, the local autonomous decision of the node is explained in detail since it is the start point of our replication model. However, the replication based on the global network decisions will be discussed in the near future.

The paper is organized as follows. Section 2 is an overview of related work. In section 3 we present the proposed model CReaM; we introduce some definitions, detail the model in itself, the architecture, and we detail one component in the architecture that is the kernel of our approach. Finally, we conclude the paper and present directions of future work in section 4.

II. Related Work

Several replication strategies have been proposed to increase data availability in mobile environment. A first criterion to categorize them is the level of autonomy of the peers. In this regard, one can distinguish between centralized (requiring a fixed host) and decentralized solutions. The latter are divided into group-based and fully decentralized strategies.

From the group based strategies, [7] proposes an economic model for dynamic allocation in M-P2P networks where the price of a data item depends on its access frequency among other values; the solution deploys a super-peer architecture where groups are formed and each group is managed by a service provider that collects information and makes the replication decision. In [1], group based data replication techniques were proposed. The replication is done periodically based on the access frequency. Three methods are proposed: in the first one, the most accessed data item is replicated in priority, while the other two reduce replica duplication among neighboring hosts or those in stable groups. In [4], the replica allocation methods are extended by considering the correlation among data items; correlated data items are replicated together in one node. All these techniques have the drawback of requiring that all hosts have a global view on the available data items and the corresponding access frequencies. Such an assumption is not adapted to highly mobile environments; it requires that all nodes broadcast information to all other nodes, which will cause significant undesirable network traffic overhead. DRAM [8] is also a group based replication solution where the group mobility is studied to avoid the broadcast of information to all nodes. One example of fully decentralized strategy is REDMAN [9]; it is a middleware that manages, retrieves, and distributes replicas and maintains approximately the desired resource replication degree. However, this solution is restricted to dense MANETs where the number of connected node in one region is high.

From another point of view, replication strategies need a trigger to start the process and criteria to decide where to place the replicas. In [1, 2, 3, 4], the replication is performed regularly during specific periods, where all nodes identify the most accessed data of last period and decide to replicate them on suitable hosts. Moussaoui et al. [11] propose two replication processes: *primary replication,* for new data items, and *dynamic replication,* executed periodically to relocate replicas near the interested nodes. Tsuchida et al. [10] handle location-dependant queries in their method Skip Copy; data are replicated on hosts within a specific area using the protocol Geocast. Other research works [2, 12, 9 13] consider other criteria to choose data items to replicate and the target hosts like the stability of the radio link, the available storage space, the remaining power, and so on. Boulkenafed et al. [12] calculate the expected time within the group. They use it in addition to the available storage space and the available energy, to avoid weak hosts. Hara et al. extend their work presented in [1] by considering the network partitioning and the host's battery power. In [2], if the radio link between two nodes is weak, the nodes are not considered as neighbors and are allowed to hold replicas of the same data item. In [4], the idea is to decrease the data transmission by increasing the number of replicas but in the same time the methods don't place replicas on nodes with low battery. Chen et al. [13] use advertising messages to communicate available data. These messages include some parameters useful to choose the replica holders, e.g., the free storage space, the remaining energy, the processor idle time, and so on.

The replication solution proposed in this paper is a fully decentralized solution without any fixed point and does not require regular communication between neighbors. All replication decisions are made locally by each peer. We argue that this strategy is more suitable for highly mobile and dynamic environments where the communication between neighbors is not always possible. In addition, in our model we attach high importance to the user aspect. The users are in control of their level of participation in the replication process and their readiness to share their resources. Then the replication system acts automatically to keep them satisfied, it adapts with the user's needs by replicating when resource consumption exceeds the limit. The modeling of our system is the subject of this paper; we also present an overall view of the architecture with a focus on its main component (PSM) that implements the key ideas of our model.

III. CReaM: User-Centric Replication Model

Today many services are applied in MANET (like file sharing [6]) and all of them require data availability. The main goal of CReaM is to increase the availability by replicating based on user needs. Each user specifies to what level they want to participate in the system with their resources; this choice constitutes an important input to the model. The model depends on the consumption of three resources: the CPU, the battery and the storage. If CReaM notices some decreases in the available resources that are unacceptable with respect to the user level of satisfaction, it acts to decrease the resource consumption. The reaction is to replicate data in order to reduce the load on the peer; CReaM chooses the relevant data to be replicated and requests other peers to hold it. On the other hand, CReaM on requested peers accepts placing the replica if the user is still satisfied from the consumption of its resources.

A. *Definitions*

Before detailing the model, let us first define some important concepts that are related to it.

1) *Access Frequency (AF)*

It indicates the number of requests received by a specific node for a specific data item. It is an accumulation starting from zero and increased by 1 after each received request. It is initiated when the data item is created on the node.

2) *Temperature Degree (TD)*

It indicates the importance of a data item; the importance is defined for a given time period and from the point of view of neighbor nodes. The TD starts from 0 and increases each time AF increase with the same quantity of value. However, when AF stays at the same value for a while, TD starts decreasing to reflect the fact that the data item is important but not requested with the same intensity. At time T_i, TD changes (increases/decreases) based on the following rules: (1) TD increases by X if AF increases by X during the time interval $[T_{i-1}, T_i]$. (2) TD decreases by a parameter Y if AF remains unchanged between T_{i-1} and T_i.

In some cases, the data item might still be important to the nodes even if AF starts to be constant. To avoid decreasing TD too rapidly, we define a third rule as follows: TD remains unchanged at T_i if AF starts to be stable at T_{i-1}.

However, in all cases, when AF remains stable for more than two consecutive time periods, the TD starts decreasing. In the following illustrative example (Figure 1), AF remains at 7 between T_4 and T_6, but the TD starts decreasing by Y=1 at T_5 until T_7 when AF starts increasing again.

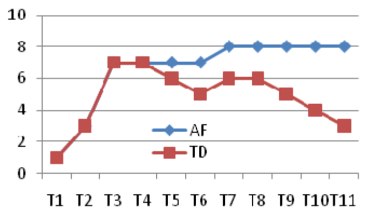

Figure 1. Example of evolution of AF and TD over time

3) *The Threshold* α

It is a numeric value related to TD used to identify important data items: when a data item's TD reaches α, it is considered "important" for neighbor nodes.

4) *The Tolerance Thresholds*

These thresholds represent the allowable level of resource consumption specified by the user. We define three thresholds: **β**: indicates the allowed load level on the node's CPU, **μ**: indicates the allowed level of remaining battery, and **δ**: indicates the allowed level of remaining storage space.

These thresholds will be used to monitor the peer's status and should influence the replication decisions; for example when the remaining battery reaches μ, outgoing replication requests will be generated, and when the remaining storage space reaches δ incoming replica placement requests will no longer be accepted.

5) *The User Tolerance degree (UTD)*

Our replication model enables the user to precise at which level her/his hardware will participate in the replication process. In this context, we define the **User Tolerance Degree (UTD),** which takes two values {Tolerant, Intolerant}, so that the more the user is tolerant the more it dedicates from her/his resources to the replication process. Thus, the UTD is used to determine the values of the tolerance thresholds. The following example explains the relation between the UTD and the tolerance thresholds: if the UTD is equal to 'Tolerant', the user dedicates 75% from its battery and the threshold μ takes the value 25%; otherwise, if UTD is equal to 'Intolerant' then μ=40% and so on.

B. *CReaM System Model*

As any replication model, CReaM answers the following questions: (1) **who** starts the replication process, (2) **when** to start it (3) **what** data to replicate and (4) **where** to place the replica. Let us consider a MANET consisting of n mobile nodes: MANET = $\{M_1, M_2\ldots M_n\}$: $n \in N$.

1) *When*

The replication model starts the replication process depending on the peer's status (its available resources, the temperature of data items held by the node). Node M_i ($1 \le i \le n$) must verifiy at least one of the following conditions in order to start the replication process:

- Condition 1: a DI_i becomes hot for the nodes in MANET. By definition, a **hot data item** is a DI with TD equal or greater than the threshold α: $TD(DI) \ge \alpha$
- Condition 2: the user becomes unsatisfied from the availability of its resources. We define three functions to calculate the consumption of the three previously mentioned resources. The output of these functions is compared to the tolerance thresholds μ, δ and β respectively, to determine whether a user should be unsatisfied with the level of his/her resource. If this is the case, the system reacts to improve the situation. Therefore, we define the function BL(T) that returns the remaining battery at time T, the function SS(T) that returns the available storage space at time T, and finally we measure the load on the CPU with the function $NoR(T_{i-1}\text{-}T_i)$ that returns the number of requests processed by the node during time interval $[T_{i-1}, T_i]$.

The answer to the "*when*" questions depends on a set of the four conditions named $CONDITIONS_M = \{TD(DI)\ge\alpha, BL(T)\le\mu, SS(T)\le\delta, NoR(T_{i-1}\text{-}T_i)\ge\beta\}$. When at least on conditions become true the replication process starts.

2) *Who*

Any mobile node that has at least one verified condition in $CONDITIONS_M$ will start the replication process.

$\forall M_i \in$ MANET: $i \in [1,...,n] \wedge \exists con \in CONDITIONS_{Mi}$: con=true

3) Where

A good distribution algorithm must be applied in order to properly distribute the replicas and avoid DI duplication on two neighbors. At the same time, the replicas must be placed near the interested nodes. The user participates also in the replica placement process; the system makes the decision to place/refuse the replicas using the tolerance thresholds.

4) What

The data item that must be replicated depends on the condition that has triggered the replication process (i.e. condition true returned from the set $CONDITIONS_M$). Thus, several cases need to be considered. For example, if the number of requests during T_{i-1} and T_i is greater than the threshold β, then the appropriate solution is to replicate the most requested DI in order to decrease the requests coming to the node, consequently decreasing the CPU load and satisfying the user desire. In another example, if a DI becomes hot, it should be replicated in order to increase its availability.

We classify the DI(s) of each node into several categories; a DI may belong to one or more categories at the same time. We define the categories as follow:

- **DI_α**: includes the hot DI(s). It is modified by adding a DI when its TD becomes equal or greater than the threshold α or by removing a DI when its TD becomes less than α.
- **DI_β**: contains the most requested DI(s) of the last period of time. It is constructed periodically each T time by adding the DI(s) with most modified AF during last period.
- **DI_r**: includes the rare DI(s) that are important but rarely found on the peers; we are interested to consider such category in order to prevent the data lost occurred when the nodes containing such rare items disconnect.
- **DI_o**: includes "not important" DI. For a DI, when TD reaches zero, the data is added to the category, and it is removed when its TD starts increasing.

In the following example, at T_3: $DI_\beta = \{DI_3, DI_4\}$ because their AF has been increased by 3 between T_2 and T_3. $DI_\alpha = \{DI_1, DI_4\}$ if the threshold α=9. Finally, $DI_o = \{DI_2\}$.

TABLE I. EXAMPLE

	DI₁		***DI₂***		***DI₃***		***DI₄***	
	AF	***TD***	***AF***	***TD***	***AF***	***TD***	***AF***	***TD***
T₁	15	8	10	1	1	1	15	5
T₂	15	8	10	0	2	2	16	6
T₃	16	9	10	0	5	5	19	9

As stated above, an action is initiated in case the user is unsatisfied i.e. a condition from $CONDITIONS_M$ become true. Below, each resource is studied separately in order to define what actions to take when necessary:

The CPU: when the number of requests exceeds the threshold β, the node selects a DI to replicate in order to share the load of the request with other nodes and to reduce its NoR. The candidate DI (**cdi**) must be a hot DI that was requested a lot during last period (**cdi** $\in DI_\alpha \cap DI_\beta$). In case this set is empty, the best choice is to select an element from DI_β that causes the load regardless if the elected element is hot or not.

$$cdi \in DI_\alpha \cap DI_\beta \text{ if } DI_\alpha \cap DI_\beta <> \{\} \text{ else } cdi \in DI_\beta . \quad (1)$$

However, if the load of the CPU keeps increasing, it would not be appropriate to keep replicating endlessly; instead, the node needs to take more radical actions in order to immediately preserve its resources. Therefore, another solution is proposed: we define two values for the threshold β, ***soft*** value β_1 and ***hard*** value β_2 ($\beta_1 < \beta_2$). If the ***soft*** threshold is exceeded ($NoR(T_{i-1}-T_i) \geq \beta_1$) the node replicates a DI as explained in (1). If the number of requests reaches the ***hard*** threshold ($NoR(T_{i-1}-T_i) \geq \beta_2$), the node will stop responding to any request.

The Storage space: following the same logic, we define two values for the threshold δ. If the available storage space becomes less than the soft threshold ($SS(T) \leq \delta_1$) the node accepts only the incoming urgent replication requests; the requests' urgency is evaluated in terms of data importance and requestor nodes' availability. If the hard threshold is exceeded ($SS(T) \leq \delta_2$), the node removes replicas from the set DI_o by applying one of the well known cache management algorithms.

The Battery: As with the other resources, it is necessary to define also two values for the threshold μ. Then, if the remaining battery becomes less than the soft threshold ($BL(T) \leq \mu_1$), the same action is applied as defined in (1). If the hard threshold is exceeded ($BL(T) \leq \mu_2$) the probability of disconnections becomes high, thus, it is more appropriate to replicate one or more rare item from the set DI_r in order to avoid data loss. However, unnecessary replication may occur, if each node replicates a rare DI and loads the network by data that might be unhelpful to the connected nodes. To avoid this situation, we give priority to a DI from the set $DI_\alpha \cap DI_r$ that is rare and hot at the same time.

In addition, a node might prevent critical situations of disconnection by reacting when noticing rapid battery consumption even before the thresholds (soft or hard) are reached. Thus, we define an additional value μ_3 for the threshold μ. The battery consumption between T_{i-1} and T_i is calculated using the function **BC =** $BL(T_{i-1}) - BL(T_i)$. When BC becomes less than the threshold μ_3, the node reacts preemptive by replicating data items according to formula (1).

Table II summarizes all cases. It contains the conditions, the peer's status and the executed actions.

C. CReaM Architecture

CReaM is a middleware layer located between the operating system and the application layer. Its functionalities are to create replicas on other nodes by disseminating replication requests RQ(s); and to manage the placement of replicas by accepting/rejecting incoming RQ(s). CReaM consists of the following components:

TABLE II. SUMMARY OF THE PEER'S STATUS

Condition	***Peer's status***	***Action***
$NoR(T_{i-1},T_i) \geq \beta_1$	CPU-Overloaded	Replicate from $DI_\alpha \cap DI_\beta$ / DI_β
$NoR(T_{i-1},T_i) \geq \beta_2$	CPU-Scarce	Stop responding
$SS(T) \leq \delta_1$	S-Overloaded	Response just to urgent RQ
$SS(T) \leq \delta_2$	S-Scarce	Delete replicas from DI_o
$BL(T) \leq \mu_1$	B-Overloaded	Replicate from $DI_\alpha \cap DI_\beta$ / DI_β
$BL(T) \leq \mu_2$	B-Scarce	Replicate from $DI_\alpha \cap DI_r$
$BC \leq \mu_3$	HB-Consumption	Replicate from $DI_\alpha \cap DI_\beta$ / DI_β

1) Peer State Manager (PSM):

The PSM controls the replication process; it monitors the peer's status and decides based when to replicate DI and when to accept placing replicas on the node. The peer's status is determined by the consumption of its resources. Thus, to monitor the resource's consumption, the PSM receives, from sensors integrated in the operating system layer, warning about insufficient resources. On the other hand, to monitor the temperature of the data items, it receives from the AFM (III.C.4) the DI categories. When the peer's status is not compatible with the user desire, the PSM notifies the responsible components to start the replicating process by sending a RQ or to place replicas by accepting RQ coming from other nodes. The PSM will be explained in more details in the next section after the presentation of the whole architecture.

2) Replica Dissemination Manager (RDM):

Its responsibility is to replicate data on neighbors by creating and distributing RQ on the network. It consists of 3 subcomponents: (1). **Replica Decider (RDec)**: it starts working after receiving a notification from the PSM. It prepares the RQ and chooses the DI to be replicated. In the RQ, it adds the reason for replication provided by the PSM in the notification. Additionally, it adds some important indicators to help other nodes decide whether to accept the RQ. This additional information may concern TD of DI and the owner's disposability. (2). **Replica Controller (RC)**: it determines the number of replicas that will be created depending on the number of connected nodes, network bandwidth, and the urgency of the replication reason. The RC adds this number to the RQ coming from RDec then sends it to next component (3). **Replica Distributer (RDis)**: it sends the RQ on the network and follows the dissemination process to insure that the number of replicas is achieved. The RDis applies suitable algorithms to choose the better available nodes that may receive the replica in a way to guarantee a good replica distribution. The chosen nodes will be contacted and their LRPM (explained later) will decide whether to accept or reject the RQ.

3) Local Replica Placement Manager (LRPM):

The LRPM is responsible for placing the replica on the current node. It receives RQ(s) from neighbors and decides with the support of the PSM whether to accept them. It consists of (1). **Replica Placement (RP)**: decides to accept/refuse the RQ depending on the available resources assigned by the user to the replication process. If the node receives more than one RQ simultaneously, it decides to which RQ it must respond based on the information included in the RQ such as the replication reason, the TD of DI and the owner's availability. (2). **Memory Management (MM)**: manages the available storage space dedicated to host replicas. A replica replacement algorithm will be executed in order to keep the most important ones when no more space is available.

4) The access frequency Manager (AFM):

It manages the access requests to all the data presented on the node. It keeps track of the values of AF and TD, and warns the PSM whenever a DI becomes hot. In addition, the AFM collects for each DI some additional information such as the distance of the nodes that accessed it. This information may be helpful to the RDis in order to distribute the replicas in an optimal way.

D. Peer State Manager (PSM)

The PSM has a fundamental job in the architecture; it monitors the peer's status and helps other components in the architecture to accomplish the replication process. Depending on the values that the PSM monitors, the RDec starts preparing a RQ to be sent on the network, and chooses the appropriate DI to be included in the RQ. The LRPM also uses these values to decide whether to accept/refuse the incoming RQ(s). In this paper, we focus on the first part of the PSM task, which is to notify the RDec to start the replication process.

As explained previously, the PSM receives the indicators from three sensors integrated in the operating system layer (sensor for each resource). These indicators are the values returned by the functions defined previously in section III.B.1. Therefore, the indicators returned by the battery sensor, the storage space sensor and the CPU sensor are respectively the values of the functions BL(T), SS(T) and $NoR(T_{i-1}-T_i)$. The PSM keeps monitoring the node that still has a *stable status* by comparing these indicators to the tolerance thresholds. By definition, a status is stable when the user is satisfied and all the previously mentioned conditions (III.B.4) are false. If one of the above conditions becomes true, the PSM reacts by updating the status of the peer and initiating the suitable action.

In some cases, more than one resource problem may be detected at the same time. This creates new challenges for choosing the appropriate course of action in such cases. To deal with these problems, we define a simple rule-based inference engine that uses some predefined rules to attribute a suitable status to the node, and to select the corresponding reaction.

Our Rules based inference engine operates by the methods of forward chaining. It consults a small knowledge base that stores the knowledge in the form of production rules. These rules are sequences defined as follows: ***If*** <conditions> ***Then*** <actions>.

- **Attributes**: are the basic elements that are used to build the conditions. In our case, three attributes are involved, and are equal to the predefined functions: BL(T), SS(T), $NoR(T_{i-1}, T_i)$. To simplify notations, in the remainder of the paper, the functions are written as BL, SS and NoR.
- **Conditions:** are Boolean expressions composed of atomic condition associated with logical connectors 'and' and 'or'. An atomic condition is a comparison between an attribute and a constant threshold. Thus an example of condition is the following: $BL \leq \mu_2$ and ($BL \leq \mu_3$ or $NoD \geq \beta_1$).

- **Actions**: are the tasks executed by the PSM when the conditions are true. They are the same actions as presented in Table II such as replicating a DI or no longer responding.

The process is initialized as follows: The PSM assigns values to the attributes, evaluates the conditions and checks to see if all conditions in a rule are satisfied. Then, the PSM executes the actions list of the rule. In order to decide which rules are fired first, a conflict resolution strategy is needed. In consequence, our method is to list the rules according to their priority and then fire the rule that is listed first.

We studied the conjunction of all conditions and we determined the suitable actions for each conjunction. A cleaning operation is done to reduce the rules and to join the similar ones with the same action list in one rule. However, we concluded to the next 6 rules presented in table III.

TABLE III. THE INFERENCE ENGINE RULES

	Conditions	***Actions***
R_1	$NoR \geq \beta_2$	A_1: Stop responding
R_2	$BL \leq \mu_2$ <u>and</u> ($BL \leq \mu_3$ <u>or</u> $NoR \geq \beta_1$)	A_2: If $DI_\alpha \cap DI_\beta \cap DI_r <> \varnothing$ Replicate from it else if $DI_r \cap DI_\beta <> \varnothing$ Replicate from it
R_3	$BL \leq \mu_2$	A_3: If $DI_\alpha \cap DI_r <> \varnothing$ Replicate from it
R_4	$BL \leq \mu_1$ <u>or</u> $BC \geq \mu_3$ <u>or</u> $NoR \geq \beta_1$	A_4: If $DI_\alpha \cap DI_\beta <> \varnothing$ Replicate from it else Replicate from DI_β
R_5	$BL > \mu_1$ <u>and</u> $SS > \delta_1$ <u>and</u> $NoR < \beta_1$	A_5: If $DI\alpha <> \varnothing$ Replicate from $DI\alpha$
R_6	$SS \leq \delta_1$	A_6: Accepting just Urgent RQ
R_7	$SS \leq \delta_2$	A_7: Delete DI_o

Fig. 2 below presents the Rules-Based inference engine of the PSM where we used the thresholds in order to present the corresponding conditions (as presented in Table 2). Therefore, the conditions are represented in the form of diamonds, the rules in the form of circles, and the actions in the form of stars.

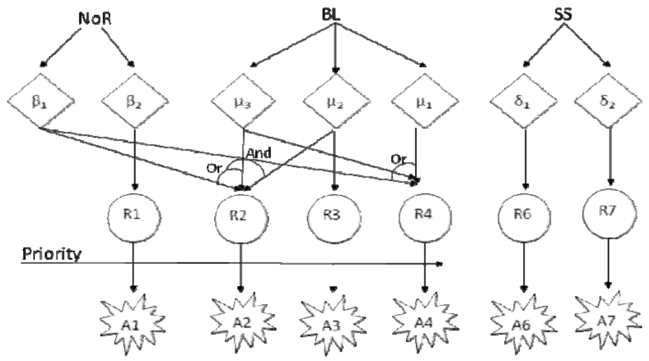

Figure 2. The Rule-Based Inference Engine of the PSM

IV. CONCLUSION AND FUTURE WORK

In this paper we have presented CReaM a user-centric model that takes the user needs at high priority. We focused on the autonomous behavior of the nodes, which is composed of the incoming replica acceptance process, and of the local decision of generating replication requests. The replication firing process depends on a set of conditions and thresholds, and is completed by an inference engine dealing with cases of simultaneously fulfilled conditions. A global architecture for the implementation of this model has been defined.

A simulation-based implementation of this model has already been developed using OMNet++. However, for a full evaluation of CReaM, it will be necessary to develop in the next step the component RDis from the architecture to answer the '*Where*' question. Of course, in a fully distributed environment, a locally optimized decision process does not imply that a satisfying global state has been reached. Work in this direction is thus needed, in order to provide peers with a rough idea about their surrounding environment in order to help them avoid making costly or contradictory actions.

ACKNOWLEDGEMENTS

This work has been partially supported by the German Academic Exchange Service (DAAD) and the Université Franco-Allemand (DFH-UFA) in the framework of the MDPS German-French doctoral college.

REFERENCES

[1] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility", In INFOCOM conference, pp. 1568–1576, 2001.

[2] T. Hara, Y.H. Loh, and S. Nishio, "Data replication methods based on the stability of radio links in ad hoc networks," International Workshop on Mobility in Databases and Distributed Systems (MDDS'03), Prague, Czech Republic, pp.969-973 (Sept. 2003).

[3] T. Hara, N. Murakami, and S. Nishio, "Replica allocation for correlated data items in ad hoc sensor networks," ACM SIGMOD Record, Vol.33, No.1, pp.38-43 (Mar. 2004).

[4] M.Shinohara, T.Hara, and S.Nishio, "Data replication considering power consumption in ad hoc networks," Proc. of International Conference on Mobile Data Management, Germany, pp.118-125 (May 2007).

[5] M.M. Nawaf, Z. Torbey, "Replica update strategy in mobile ad hoc networks", The international ACM Student workshop on Management of Emergent Digital EcoSystems (MEDES-SW), 2009, Lyon-France.

[6] Atechian T., Torbey Z., Bennani B., Brunie L., "CoFFee: Cooperative and InFrastructure-Free Peer-To-Peer System for VANET", 9th international ITS of Telecommunications, October 2009, Lille, France.

[7] A. Mondal, S.K. Madria, and M. Kitsuregawa, "EcoRep: An economic model for efficient dynamic replication in mobile-P2P networks", 13th International Conference on Management of Data COMAD, India 2006.

[8] J.L. Huang, M.S. Chen, and W.C. Peng, "Exploring group mobility for replica data allocation in a mobile environment", International conference on Information and Knowledge Management, 2003.

[9] P. Bellavista, A. Corradi, and E. Magistretti "REDMAN: A decentralized middleware solution for cooperative replication in dense MANETs", 3th IEEE Internationl conference on pervasive computing and communications workshops PerCom 2005, pp. 158- 162.

[10] G. Tsuchida, T. Okino, M. Tamori, T. Watanabe, T. Mizuno, and S. Ishihara, "Replica distribution of data associated with location on wireless ad hoc networks", Electronics and Communications in Japan (Part I: Communications), vol. 90, Issue 10, pp. 67-80, April 2007.

[11] S. Moussaoui, M. Guerroumi, and N. Badache, "Data replication in mobile ad hoc networks", International conference on Mobile Ad Hoc and Sensors Networks, HongKong, pp. 685-698, 2006.

[12] M. Boulkenafed, and V. Issarny, "A middleware service for mobile ad hoc data sharing, enhancing data availability", the 4th ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 2003, pp. 493-511.

[13] K. Chen, and K. Nahrstedt, "An integrated data lookup and replication scheme in Mobile ad hoc networks", International symposium on the Convergence of Information technologies and Communications, 2001.