Performance Evaluation for CReaM: User-Centric Replication Model

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—**The deployment of applications in mobile networks is hindered by limited resources and frequent network disconnection. Data replication can improve data availability in mobile networks but also introduces the problem of adequately disseminating data without abusing user and network resources. In this context, we present CReaM, a user-Centric REplicAtion Model for mobile environment that privileges the users by letting them determine the amount of resources they assign to the system. In this paper, we focus on CReaM's autonomic behavior that generates replication requests based on resource monitoring and user settings. Then, we present a simulation-based evaluation of CReaM, which shows the efficiency of the model.**

Keywords-component; 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 that communicate in an autonomous way without any centralized server. Most often, these networks are deployed on devices with limited resources. Moreover, 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. These characteristics make it difficult to guarantee data availability, hindering the wide-spread deployment of distributed applications over MANETs. In this context, a possible solution is the use of data replication techniques.

However, applying data replication in MANETs 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 devices have limited power, which means that efficient methods must be adopted to reduce the communication between nodes. In view of all these issues, specific data replication mechanisms for MANETs are required.

Previously proposed MANET replication models replicate data by taking into account resource availability and the access frequency of the data items. Most of them replicate periodically the important data items and place the replica on the devices with the most available resources. Such top-down approaches may lead to an abuse of user resources, as in the worst case they might use all of a user's resources in order to achieve their goal. To the contrary, we feel that users should be at the center of the systems. We propose a user-centric approach in which users 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 local MANET. 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 the criteria of previous work, 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 reacts starting from the users' needs, another challenge appears: the system must take local decisions to satisfy the individual needs, but it must also consider the interest of the whole system. Finding a balance between these two factors is necessary to avoid problems such as 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 replicas on other users' devices that can support the load instead. Thus, the system is driven by the wishes of the

978-1-4577-0729-2/11/$26.00 ©2011 IEEE

users, 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 such as 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, whether to generate replication requests, and if need be about what data to replicate.

In this paper, we have focused on the local autonomous replication decision of the node, based on user satisfaction criteria and evaluate the impact of the local decision on user satisfaction and on the global query satisfaction.

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. Section 4 contains the performance evaluations. Finally, we conclude the paper and present directions of future work in section 5.

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. We focus on the latter which can be further 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 periodically at specific time points, at which 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; the 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 such as 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 do not place replicas on nodes with low battery. Chen et al. [13] use advertising messages to communicate available data. These messages include some parameters that can assist the choice of replica holders, e.g., the free storage space, the remaining energy, the processor idle time, etc.

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, our model is user-centric. The users are in control of their level of participation in the replication process and of the amount of resources that they make available. Then the replication system acts automatically to keep them satisfied; it adapts to the user's needs by replicating when resource consumption exceeds the chosen limit. In this paper, we will first present our model then an overall view of the architecture with a focus on its main component (PSM) that implements the key ideas of our model. In a second part, we will present our experiments that validate and confirm the efficiency of our model with user need satisfactions.

III. CREAM: USER-CENTRIC REPLICATION MODEL

In this section, we describe our user-Centric REplicAtion Model for mobile environment (CReaM). Cream is a decentralized user-centric replication that takes replication decisions at the node level with the goal of increasing data availability. The model is user-centric as it is driven by user-chosen threshold to decide when to replicate. Indeed, it depends on monitoring 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 is also decentralized as it takes its decision based on local information: the consumption of the above mentioned resources, the user-specified thresholds, but also the requests observed by the node. It uses this information to select the data to be replicated and requests other peers to hold it. Indeed, CReaM handles the read-only data items. It also manages incoming replication requests in a user-centric way by deciding whether to accept replica placement depending on its effect on user satisfaction with respect to the consumption of its resources. Before detailing the model, we first define some important concepts that are related to it.

A. *Definitions*

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.

Temperature Degree (TD): It indicates the current importance of a data item; the importance is defined for a given time period and from the point of view of neighbor nodes. TD starts from 0; it is updated periodically at specific time points based on a predefined time window. If AF increased during the last time interval, TD increases by the same value. However, if AF remained at the same level, 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 is updated based on the following rules: (1) TD increases by X if AF increased 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, TD starts decreasing. In the following illustrative example (Fig. 1), AF remains at 7 between T_4 and T_6, but TD starts decreasing by Y=1 at T_5 until T_7 when AF starts increasing again.

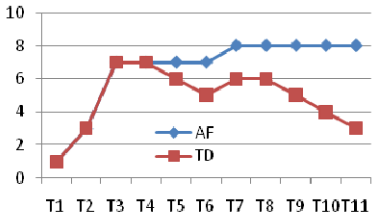

Figure 1. Example of evolution of AF and TD over time

The Threshold α: It is a numeric value related to TD used to identify important data items: when a data item DI has a TD value that reaches α, it is considered as *hot* on this node $TD(DI) \geq \alpha$.

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

These thresholds will be used to monitor the peer's status and take 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.

The User Tolerance degree (UTD): Our replication model enables the user to define 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. *The User Centric Replication 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: $MANET = \{M_1, M_2 \ldots M_n\} : n \in N$.

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 $(1 \leq i \leq n)$ must verify at least one of the following conditions in order to start the replication process:

- *Condition 1*: a DI_i becomes hot for node M_i. In this case replicating DI will increase its availability
- *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 (CPU, battery, storage). 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. For this purpose, we define the function $NoR(T_{i-1}\text{-}T_i)$ that returns the number of requests processed by the node during time interval $[T_{i-1}, T_i]$ that corresponds to the CPU load, the function BL(T) that returns the remaining battery at time T, and the function SS(T) that returns the available storage space at time T.

The answer to the "when" question depends on the set of the conditions named $CONDITIONS_M = \{TD(DI) \geq \alpha, BL(T) \leq \mu, SS(T) \leq \delta, NoR(T_{i-1}\text{-}T_i) \geq \beta\}$. When at least one condition becomes true the replication process starts.

Who: CReaM being a fully decentralized model, any mobile node that has at least one verified condition in CONDITIONSM starts the replication process.

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 most 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. We are currently working on an algorithm including all these aspects to take the replica placement decision.

What: The DI that must be replicated depends on the condition that has triggered the replication process (i.e. true conditions among the set CONDITIONSM). 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. These categories are defined 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 becomes *hot*.
DI_β: contains the requested DI(s) of the last period of time. It is constructed periodically each time T by adding the DI(s) with 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(s). A DI which its TD reaches zero is added to this category, and removed from it when its TD starts increasing.

In the following example, at T_3: $DI_\beta = \{DI_2, DI_3, DI_4\}$ 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

	DI_1		DI_2		DI_3		DI_4	
	AF	***TD***	***AF***	***TD***	***AF***	***TD***	***AF***	***TD***
T_1	15	8	10	1	1	1	15	5
T_2	15	8	10	0	2	2	16	6
T_3	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 $CONDITIONS_M$ becomes 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 requests with other nodes and to reduce its NoR. The candidate DI (**cdi)** must be a hot DI that was requested during last period ($\mathbf{cdi} \in DI_\alpha \cap DI_\beta$). If 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.

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

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 ($NoR(T_{i-1}\text{-}T_i) \geq \beta_1$) the node replicates a DI as explained in (1). If the number of requests reaches the ***hard*** threshold ($NoR(T_{i-1}\text{-}T_i) \geq \beta_2$), 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 replacement algorithms which is out of article's scope.

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 DI 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 remaining 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 $\mathbf{BC} = BL(T_{i-1}) - BL(T_i)$. 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:

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

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 Peer State Manager (PSM), a core component that controls Peer status, the replica dissemination Manager (RDM) that places replicas on neighbors and the Local Replica Placement Manager (LRPM) that deals with local replica placement. Both the RDM and the LRPM act based on the information provided by the PSM. Finally, the Access Frequency Manager (AFM) collects information about DIs. The four components are detailed below.

Peer State Manager (PSM): The PSM controls the replication process; it monitors the peer's status and decides 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 DI(s), it receives warnings from the AFM when a DI become hot. 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.

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 or reject the RQ. This additional information may concern TD of DI and the owner's availability. (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 best 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 will decide whether to accept or reject the RQ.

Local Replica Placement Manager (LRPM): The LRPM is responsible for placing the replica on the current node. It receives RQ(s) from other nodes and decides with the support of the PSM whether to accept them. It consists of two components. (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. It runs a cache replacement algorithm to decide which replicas should be deleted when no more space is available.

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 is the central component of 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. 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 $NoR(T_{i-1}\text{-}T_i)$. 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 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 on a forward chaining mode. 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$\leq \mu_2$ and (BL$\leq \mu_3$ or 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$\leq \mu_1$ or BC$\geq \mu_3$ or NoR$\geq \beta_1$	A_4: If $DI_\alpha \cap DI_\beta <> \varnothing$ Replicate from it else Replicate from DI_β
R_5	BL$>\mu_1$ and SS $> \delta_1$ and 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

IV. PERFORMANCE EVALUATION

A. Simulation design

In this section, we present the simulation that was carried out in order to validate the key functionalities of our proposed model and to evaluate its performance. The simulator has been developed using the OMNet++ and INETMANET frameworks[1].

The experimental scenario is the following. A fixed set of nodes (100) interact for a given period of time; each node runs CReaM. The nodes move according to a given mobility model (see below) within a predefined region (square region of 1500m x 1500m). Each node is initialized with a set of data items chosen from a predefined list of 75 to 150 documents. The size of a data item is fixed at 1,5 kB. The interaction consists in nodes issuing requests for documents according to a requests generation model detailed below. As the nodes run CReaM, replication requests are also generated and processed during the simulation. The communication layer is simulated using AODV [19] to route requests and data, UDP broadcast to transmit all messages and IEEE802.11n in the MAC layer. The bandwidth is set at 2 Mbit/s. For the parameters of CReaM, we fixed the tolerance thresholds as summarized in Table IV, the number of replicas is a parameter that changes from run to another; and finally the selection of the replica holders was done randomly, waiting the end of the component Replica Distributer.

Some models are defined and used in our simulation; in the following, we explained the mobility model of connected nodes, the data generation and distribution model, and finally the query distribution model.

TABLE IV. CREAM'S PARAMETERS

Threshold of tolerance	β_1= 4R, β_2= 10R, μ_1=60%, μ_2=75%
Threshold Alfa	30
Time period for PSM	7s
Time life for each query	3s

Mobility Model: The nodes move according to the *Random Way Point* model which is widely used in MANETs simulation and as it seems to be the closest to typical movement patterns of the real mobile nodes. The moving speed of each mobile host *v* was chosen randomly in the range 0..3 m/s and pause time was 3s. The initial position of each host was also set randomly.

Data generation model: The number of data items on the network changes during simulation runs in order to study its impact on the different measurements. The data items are distributed on all mobile nodes in the beginning of the simulation based on the Zipf distribution [20] which has been frequently used to model non-uniform distribution.

Data requests generation model: is based on the Poisson model [19] with a mean of 4 requests each 2s. The packet length of a request message is 128 bytes including UDP and IP header. Periodically, the simulator selects randomly mobile nodes to send the requests and the subjects (DI) of the requests. The selected requestor nodes broadcast the requests to their neighbors and wait for responses. When a mobile host that receives a request message holds a replica of requested data item, it sends back a response message containing the replica. The response message may be simply forwarded to requesting node by unicast replying using the reverse of the requesting route. The size of all reply messages was set to 1500 bytes including replying route in our simulations. Each requesting host copies the replica to its local storage after it receives the replica. Actually, the new copied replicas are read only.

B. Experimental results

Two metrics were defined for the experiments namely data availability and user satisfaction. These metrics were used to evaluate the system's performance under specific settings. For each particular setting, the simulation was executed 25 times; the presented results correspond to the average of the 25 runs. We now define each metric, present and analyze the corresponding simulation results.

Data availability (DA): this metric represents the rate of data availability during the simulation (i.e.) the ratio of successful requests. Formally, it is defined by the following formula: DA=(NoSR/NoTR)*100, where NoSR and NoTR are respectively the number of successful requests and the total number of requests during the simulation. CReaM's goal is to increase the DA as much as possible.

To determine if CReaM increases the DA, we have measured it in three cases: 1- CReaM is not applied at all, 2- CReaM is applied and one replication request is generated after each PSM's reaction. 3- Cream is applied also but two replication requests are generated. In the three cases the other simulation parameters do not change.

[1] [1] http://www.omnetpp.org

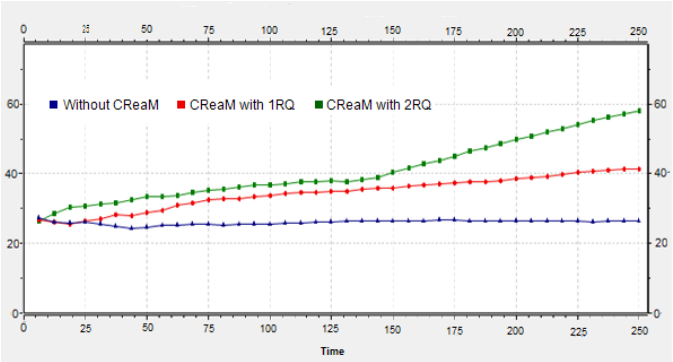

Figure 2. Data availability with time

Fig. 2 shows the obtained results with respect to the time. We see here that CReaM increases the data availability significantly, with the improvement increasing over time. We also note that the number of replicas seems to affect the data availability as the best results are obtained with RQ=2. Thus, in a second series of tests, we have tried to experimentally determine the best value for the number of replicas created by CReaM each time the PSM reacts. A compromise has to be found to this end; indeed increasing this parameter improves data availability, but also creates more overhead. From fig. 3 we conclude that after running the simulation for a while, making three replicas each time provides data availability similar to making 4 or more replicas, but surely it will cause less overhead. For the remaining of this paper, we will consider RQ=2 as the best number of replica number to disseminate for each request.

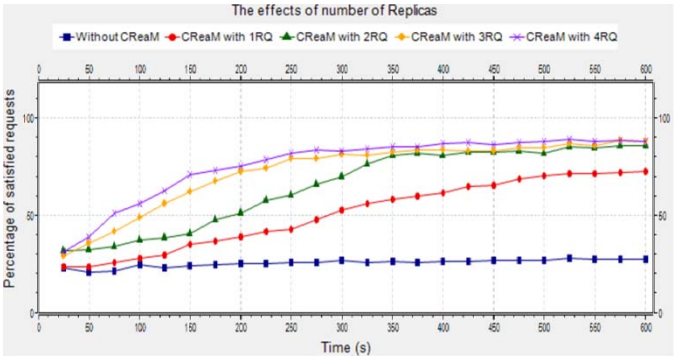

Figure 3. The effects of the number of replicas

The third test studied the influence of the number of data items on the effectiveness of CReaM evaluated based on DA. Fig. 4 shows that as the number of data item increases, the positive effect of the replication on data availability gets less significant; Indeed, in our setting the number of requests remains constant whereas the dispersion of query targets increases; consequently, the impact of each replica decreases.

User satisfaction (US): this metric aims to determine the fraction of simulation time during which a user remained satisfied from their resources' consumption; indeed, the idea of maintaining resource consumption within user defined boundaries is at the centre of the design of our model, as it distinguishes it from other replication models. Thus, user success is a critical criterion to evaluate the success of the approach. Formally, the metric is defined for each user as $US_i = (NoTS_i/T)*100$, where NoTS is the total time during which the i^{th} user remains satisfied over the whole simulation, and T is the duration of the simulation.

Fig. 5 shows individual results for a sample of 10 users. We can note that CReaM increases the user satisfaction even when only one replica is created with each replication request (with one exception). Moreover in most cases the improvement is even more significant with RQ=2. This tends to show that CReaM has the desired effect of better distributing the load of data requests on all less busy connected nodes, which helps keeping the user satisfied.

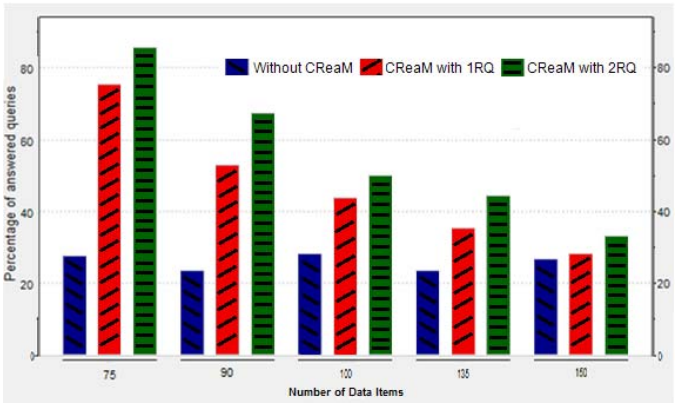

Figure 4. Influence of data items' number on Data Availability

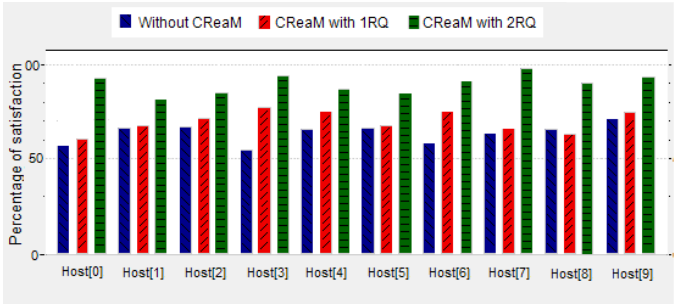

Figure 5. User Satisfaction

In addition to the US, we define the Total User Satisfaction (TUS) to observe the effect of CReaM on the whole network. It is defined as the $TUS=\Sigma_t(NoUSt)$, where the NoUS is the number of satisfied users during time period T. Fig. 6 shows that with CReaM, in general, the number of satisfied users is higher than without CReaM. That confirms the results obtained for 10 nodes; thus, considering the whole set of users, with time, CReaM also has the desired effect of distributing the load of data requests on all connected nodes, thus keeping the users satisfied. Moreover, when the RQ=2 the TUS is very high during all the simulation.

As a results summary, we conclude that CReaM increases the data availability significantly, especially with RQ=2, while at the same time keeping the users satisfied of the level of consumption of their resources. Compared to other replication models, CReaM is able to do so with minimal additional overhead as it does not require any exchange of messages between nodes to query for information to support the

replication decisions; its overhead is limited to replica placement messages.

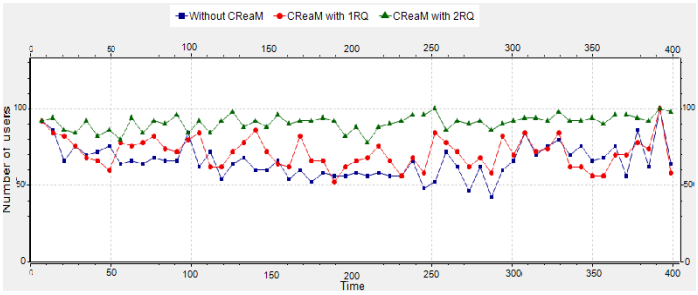

Figure 6. The total user satisfaction

V. CONCLUSION AND FUTURE WORK

In this paper we have presented CReaM, a replication model for MANETs. CReaM addresses the important problem of maintaining data availability in mobile environments. The model is user-centric, as each node only contributes under the condition that doing so does not cause excessive resource consumption. . In this paper, we have focused on the autonomous behavior of the nodes, according to which each node bases on its user needs to trigger replication requests. This process is based on settings chosen by the user and on monitoring of the resources that are at the user's disposal. CReaM has been evaluated using a simulation-based implementation using the OMNet++ simulation environment. The experimentations show that CReaM increases the data availability in a significant way, with high rate of user satisfaction.

Another series of experimentations are currently in progress. These experiments have two goals. The first one is to study the overhead introduced by CReaM and compare it with other replication models in terms of data availability, user satisfaction, and overhead. The second goal is to determine experimentally the best values for the tolerance thresholds that are important parameters of the model. In addition, we are working on the *Replica Distributer* component; our objective is to desing it so that it also enhances the data availability and the user satisfaction by a better choice of nodes that can hold new replicas, taking into account data distribution.

ACKNOWLEDGMENTS

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 proceedings of IEEE INFOCOM conference, pp. 1568–1576 April 2001.

[2] T. Hara, Y.H. Loh, and S. Nishio, "Data replication methods based on the stability of radio links in ad hoc networks," Proc. of 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 (MDM 2007), Manhaim, 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] T. Atechian, Z. Torbey, N. Bennani, L. Brunie, "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", 12th ACM International conference on Information and Knowledge Management CIKM-03, November 3–8 (2003), pp. 161–168.

[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 MSN'06, HongKong, December 2006, Springer LNCS 4325, pp. 685-698.

[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", In: SPIE international symposium on the Convergence of Information technologies and Communications, 2001, pp. 1–8.

[14] Z. Torbey, N. Bennani, D. Coquil, L. Brunie.: "CReaM: User-Centric Replication Model for Mobile Environments", the international Workshop on "Mobile P2P Data Management, Security and Trust (M-PDMST 2010)", IEEE ed. Kansas City, USA. pp. 348-353. ISBN 978-1-4244-7075-4. (2010).

[15] D. Harsch, S. Madria, "A Resource-Efficient Adaptive Caching Scheme for Mobile Ad Hoc Networks", 29th IEEE international symposium on Reliable Distributed Systems, (October 2010).

[16] T. Hara, S. Madria, "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks", the IEEE Transactions on Mobile Computing, vol.8, no.7, pp.950-967. (July 2009).

[17] A. Kanzaki, Y. Sawai, M. Shinohara, T. Hara, and S. Nishio, "Quorum-Based Consistency Management for Data Replication in Mobile Ad Hoc Networks", the international Workshop for Ubiquitous Networking and Enablers to Context-Aware Services, Turky, Finland, pp.357-360, (July 2008).

[18] Ad hoc On Demand Distance Vector, http://www.ietf.org/rfc/rfc3561.txt

[19] Rodriguez G.: Poisson Models for Count Data. September 2007.

[20] http://www.nslij-genetics.org/wli/zipf/