User Centric Replication Approach to Maintain Data Availability in MANET

Zeina Torbey[1], Nadia Bennani[1], David Coquil[2], and Lionel Brunie[1]

[1] Université de Lyon, CNRS, LIRIS,
Lyon, France
{zeina.torbey,nadia.bennani,lionel.brunie}@insa-lyon.fr
[2] 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 challenge 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 gives priority to 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 its efficiency comparing with a periodical model; indeed, CReaM gives the same rate of data availability while it causes 50 less overhead.

Keywords: Mobile Ad-Hoc network, Replication, User centric, Data availability, Simulation.

1 Introduction

A mobile Ad hoc NETwork (MANET) 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 several 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

N. Venkatasubramanian et al. (Eds.): Mobilware 2011, LNICST 93, pp. 195–208, 2012.

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: Marie, a journalist, is in a stadium. She connects to the local MANET, which provides a data sharing service. Marie records interviews with the players and takes photos of them on her notebook. Other users are interested in such photos, but as they sit in the back rows of the stadium, they cannot take them themselves, and thus send requests for such data to the data sharing service. The service connects them to Marie. However, after a while, the load on Marie's device becomes too high and she cannot answer any more requests. We can see here how a system that automatically creates copies of the data when the load on one device becomes too high due to other users' requests would be useful. Indeed, by copying on other devices of the MANET, it would provide relief to Marie as the next requests would be directed to those devices rather than to hers. Let us assume that such a system is in place; when creating a replica, it must decide on which node to place it. According to the criteria of existing replication model of the literature, the device of George (another journalist covering the event) would be selected, because his device has the highest capability. However, from another point of view, the mobile phone of another spectator may be a better choice, simply because it is currently not used, whereas George needs the resources of his device to accomplish his work. But later on, if the spectator needs his resources to accomplish a task, the replication mechanism should again evolve cleverly by dynamically choosing another 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 these 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. The system is therefore driven by the wishes of the

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 periodically gathers the consumption of 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 which data to replicate.

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

2 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], 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. Examples of fully decentralized strategy are REDMAN [9], and [15]; [9] presents 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 a region is high. The solution proposed in [15] distributes the storage, bandwidth and energy load through a resource-efficient adaptive caching scheme; each node flags a data item to be replicated when it discovered high bandwidth utilization for that data item. The primary difference between [15] and our method is the consideration of the user needs as a trigger for the replication process.

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. 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 good level of user satisfaction.

3 CReaM: User-Centric Replication Model

In this section, we describe our user-Centric REplicAtion Model for mobile environments (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, the model 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. CReaM 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.

3.1 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.

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 $TD(DI) \geq \alpha$.

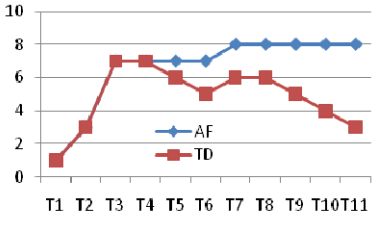

Fig. 1. Example of evolution of AF and TD over time

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.

3.2 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 $NoR(T_{i-1},T_i)$ 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 $CONDITIONS_M = \{TD(DI) \geq \alpha, BL(T) \leq \mu, SS(T) \leq \delta, NoR(T_{i-1},T_i) \geq \beta\}$. 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 peer participates also in the replica placement process; the system makes the decision to place/refuse the replicas using the tolerance thresholds configured according to the user needs. We are currently working on an algorithm including all these aspects to take the replica placement decision.

What. The data item that must be replicated depends on the condition that has triggered the replication process (i.e. true conditions among the set $CONDITIONS_M$). Thus, several cases need to be considered. For example, if $NoR(T_{i-1}, T_i) \geq \beta$, 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). A DI joins this category when its TD becomes equal or greater than the threshold α and leaves it when it is no longer *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 of time.
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.

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 (**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.

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

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, 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 a DI from the set DI_o by applying one of the well known cache replacement 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 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 **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 1 summarizes all cases. It contains the conditions, the peer's status and the executed actions. We proposed architecture to implement the model of CReaM; the details of this architecture are presented in details in [14].

Table 1. 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

4 Performance Evaluations

4.1 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

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

and IEEE802.11n in the MAC layer. The bandwidth is set at 2 Mbit/s. For the parameters of CReaM, we have fixed the tolerance thresholds as summarized in Table 2, the number of replicas is determined experimentally to 2 after each PSM's reaction; and finally the selection of the replica holders has been done randomly, in the meantime of implementing the Replica Distributer behavior.

Table 2. 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

In the following, we explained successively, the mobility model of connected nodes, the data generation and distribution model, and finally the query distribution model.

Mobility Model: The nodes move according to the *Random Way Point* model which is widely used in MANETs simulations 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 3 seconds. 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[2], 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 reply using the reverse route of the request. The size of all reply messages was set to 1500 bytes including replying route in our simulations. Each requested host copies the replica to its local storage after it receives the replica. Actually, we assume the new-copied replicas are read-only.

4.2 Experimental Results

As explained before, CReaM starts the replication process on an ad-hoc basis, when the resources consumption exceeds the levels specified by the user. This is different

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

from most existing works, in which each node periodically decides whether to replicate some of its (hot) data items. Thus, to validate our model, we have developed a simulation with the goal of comparing it to a periodical replication model. Three metrics were defined for these experiments, namely data availability, overhead, and user satisfaction. For each particular setting, the simulation was executed 25 times. We now define each metric then, 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. The replication system's goal is to increase the DA as much as possible.

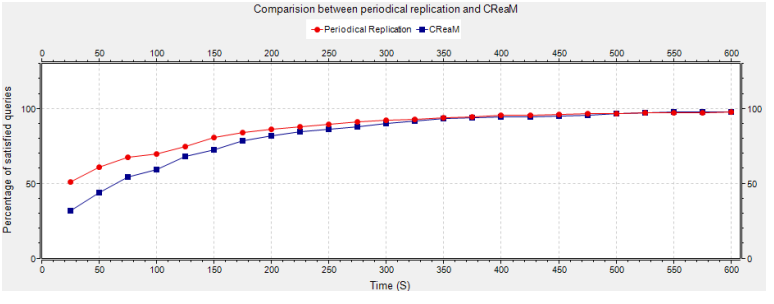

Fig. 2. Data availability with time

Some experiment studies of the positive effects of CReaM and the influence of the data item's number on DA were done; indeed, CReaM increases the data availability in a significant manner and it was proven experimentally that making two replicas in each replication request gives a compromise between the overhead and the augmentation in DA.

Fig. 2 shows the obtained results of comparing CReaM with a periodical replication system with respect to the time. We see here that the periodical replication model increases the data availability better than CReaM in the beginning of the simulation; but with time, CReaM increases also the DA as the periodical replication does. In other words, CReaM gives the same performance with time, because the replicas on the network are created only when it's needed. From other point of view, because CReaM creates the replicas when needed, it causes less overhead. From here the second metrics *overhead* is necessary to show the utility of CReaM.

Overhead (OVH): This metric represents the total number of exchange messages needed for the replication system during the simulation time. Formally, it is defined as $OVH=\sum_{i=1}^{n} NoMi$ where NoMi is the total number of messages needed for the replication system and sent from node i and n is the number of nodes in the system. The aim of any replication system is to decrease the overhead as much as possible.

A comparison between the overhead caused by CReaM and by the periodical replication was done. Fig. 3. shows that during the simulation, CReaM causes much less overhead than the periodical model. This is because CReaM only sends replication requests when one of the conditions monitored by the PSM is reached. We are planning to further study this when the component Replica Distributer will be ready. Indeed we are designing to this end an algorithm that does not require many additional messages to select the replica holders and expect that appropriate replica placement will contribute to maintain the overhead caused by our replication system low as possible with better data dissemination according to requests origins.

In a second series of experiments, we have studied the influence of the number of data items on the overhead created by CReaM. In Fig. 4, we can observe very different results between the overhead caused by CReaM and by the other model. Moreover, we note that the positive effect of CReaM on the overhead gets more significant as the number of data items increases. At the same time, as shown in Fig. 5 the global rates of data availability are similar in all cases. For example, when the number of data item is equal to 150, CReaM causes less than half overhead than the other model while they provide the same rate of data availability.

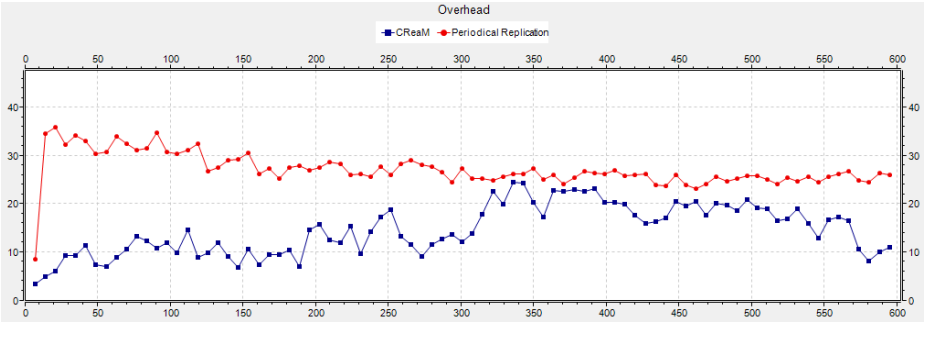

Fig. 3. Overhead with time

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 USi = (NoTSi/T)*100, where NoTSi 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. 6 shows individual results for a sample of 10 users. We can note that CReaM increases the user satisfaction. This tends to show that CReaM has the desired effect of better distributing the load of data requests on less busy nodes, which helps keeping the user satisfied.

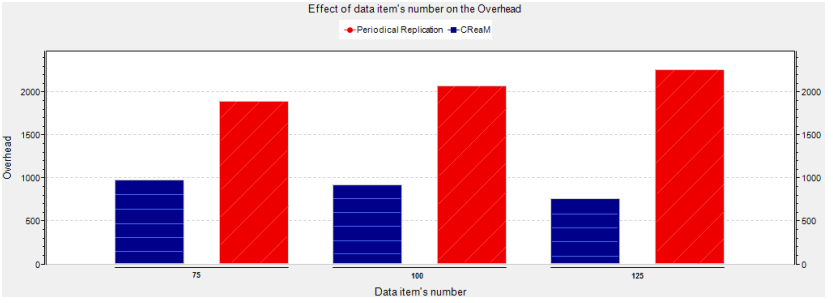

Fig. 4. Influence of data items' number on Overhead

Fig. 5. Influence of data items' number on data availability

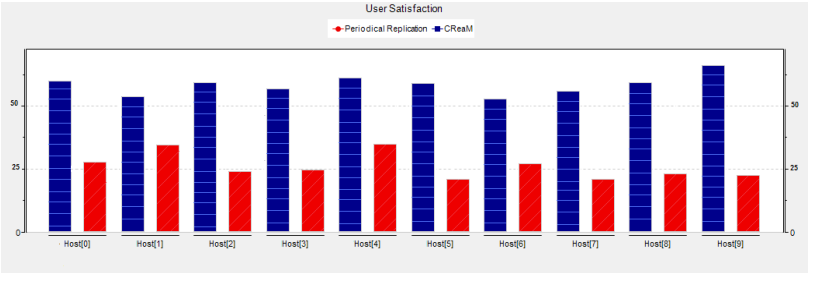

Fig. 6. User Satisfaction

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 $TUS=\Sigma_t(NoUS_t)$, where the NoUS is the number of satisfied users during time period T. Fig. 7 shows that with CReaM the number of satisfied users is higher than with the periodical replication model that not care in the user satisfaction but rather resources availability. 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.

To summarize, we can conclude from this experimental study that CReaM maintains data availability at level comparable to those of periodical replication systems but with an overhead that is significantly lower than that of the adversary.

This enable CReaM to reach its goal and save the available resources (network and devices resources) for the applications deployed on the MANET. At the same time, it keeps the users satisfied of the level of consumption of their resources, a point that is hardly considered by other replication systems, whereas we argue that this is a crucial and more realistic feature that a real large-scale system should provide.

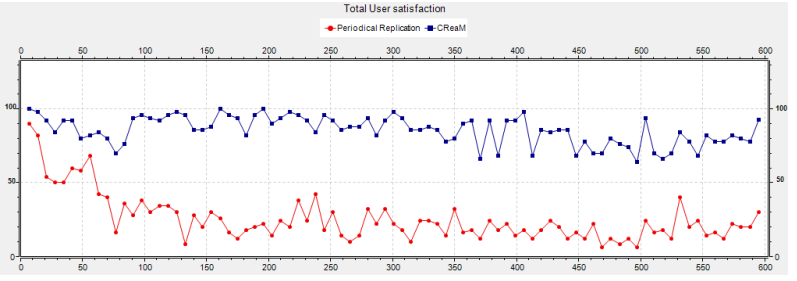

Fig. 7. The total user satisfaction

5 Conclusion and Future Work

In this paper we have presented CReaM, a user-centric 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 and available resources 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 and low level of overhead.

Another series of experimentations are currently in progress. Their 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 design it so that it also enhances proactively the data availability from a semantic point of view and the user satisfaction by a better choice of nodes that can hold new replicas, taking into account data distribution.

References

1. Hara, T.: Effective replica allocation in ad hoc networks for improving data accessibility. In: Proceedings of IEEE INFOCOM Conference, pp. 1568–1576 (April 2001)
2. Hara, T., Loh, Y.H., Nishio, S.: Data replication methods based on the stability of radio links in ad hoc networks. In: Proc. of International Workshop on Mobility in Databases and Distributed Systems (MDDS 2003), Prague, Czech Republic, pp. 969–973 (September 2003)

3. Hara, T., Murakami, N., Nishio, S.: Replica allocation for correlated data items in ad hoc sensor networks. ACM SIGMOD Record 33(1), 38–43 (2004)
4. Shinohara, M., Hara, T., Nishio, S.: Data replication considering power consumption in ad hoc networks. In: International Conference on Mobile Data Management, Germany (2007)
5. Nawaf, M.M., Torbey, Z.: Replica update strategy in mobile ad hoc networks. In: International Workshop on Management of Emergent Digital EcoSystems, Lyon-France (2009)
6. Atechian, T., Torbey, Z., Bennani, N., Brunie, L.: CoFFee: Cooperative and InFrastructure-Free Peer-To-Peer System for VANET. In: 9th International ITS of Telecommunications, France (October 2009)
7. Mondal, A., Madria, S.K., Kitsuregawa, M.: EcoRep: An economic model for efficient dynamic replication in mobile-P2P networks. In: 13th International Conference on Management of Data COMAD, India (2006)
8. Huang, J.L., Chen, M.S., Peng, W.C.: Exploring group mobility for replica data allocation in a mobile environment. In: International Conference on Information and Knowledge Management (CIKM 2003), Louisiana, USA, pp. 161–168 (November 2003)
9. Bellavista, P., Corradi, A., Magistretti, E.: REDMAN: A decentralized middleware solution for cooperative replication in dense MANETs. In: 3th IEEE Internationl Conference on Pervasive Computing and Communications Workshops PerCom, pp. 158–162 (2005)
10. Tsuchida, G., Okino, T., Tamori, M., Watanabe, T., Mizuno, T., Ishihara, S.: Replica distribution of data associated with location on wireless ad hoc networks. Electronics and Communications in Japan (Part I: Communications) 90(10), 67–80 (2007)
11. Moussaoui, S., Guerroumi, M., Badache, N.: Data Replication in Mobile Ad Hoc Networks. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) MSN 2006. LNCS, vol. 4325, pp. 685–697. Springer, Heidelberg (2006)
12. Boulkenafed, M., Issarny, V.: A Middleware Service for Mobile Ad Hoc Data Sharing, Enhancing Data Availability. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, pp. 493–511. Springer, Heidelberg (2003)
13. Chen, K., Nahrstedt, K.: An integrated data lookup and replication scheme in Mobile ad hoc networks. In: SPIE International Symposium on the Convergence of Information Technologies and Communications, pp. 1–8 (2001)
14. Torbey, Z., Bennani, N., Coquil, C., Brunie, L.: CReaM: User-Centric Replication Model for Mobile Environments. In: The International Workshop on "Mobile P2P Data Management, Security and Trust (M-PDMST 2010)", pp. 348–353. IEEE, Kansas City (2010) ISBN 978-1-4244-7075-4
15. Harsch, D., Madria, S.: A Resource-Efficient Adaptive Caching Scheme for Mobile Ad Hoc Networks. In: The 29th IEEE International Symposium on Reliable Distributed Systems (October 2010)
16. Hara, T., Madria, S.: Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks. The IEEE Transactions on Mobile Computing 8(7), 950–967 (2009)
17. Kanzaki, A., Sawai, Y., Shinohara, M., Hara, T., Nishio, S.: Quorum-Based Consistency Management for Data Replication in Mobile Ad Hoc Networks. In: 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/`