A Decentralized and Autonomous 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.

Categories and Subject Descriptors
H.3.4 [**Information Systems**]: Systems and Software – *Distributed Systems, Information network.*

General Terms
Algorithms, Performance, Design, Human Factors.

Keywords
Replication, User Centric, Data availability, Mobile Environment

1. INTRODUCTION
Data replication is one possible solution to maintain data availability and face the problem of frequent network topology changes in Mobile Ad hoc NETwork (MANET). In such environment, nodes' mobility leads to network partitioning and cause incomplete data transfer. However, applying replication in MANETs is not a trivial task for many reasons: (1) the network topology changes frequently, (2) the devices have limited resources (storage space, power, etc…), so the data must be carefully selected and the communication must be reduced, (3) the consistency of the data is difficult to maintain.

Most of the proposed replication models replicate periodically the important data items and take into account the resource availability by placing 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 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.

Several challenges need to be overcome to develop such a system. First, it must react dynamically to the resources' consumption on each node; however, not all changes are significant; 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, a balance between the individual need of connected user and the interest of the whole system 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 which is, in our view, a key requirement of a realistic approach; it replicates automatically when the user is overloaded and places replica on other users' devices that can support the load. 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 and dynamic environments where network topology changes rapidly. Indeed, replication decisions in CReaM are made in a totally distributed manner. 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. We worked on the local autonomous decision of the node, while the replication based on the global network decisions will be discussed in the near future.

The paper is organized as follows. In section 2 we present the CReaM model; we introduce some definitions, detail the model in itself, and its architecture. In section 3 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.

2. CReaM: User-Centric Replication Model
The main goal of CReaM is to increase the availability by replicating data items 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. During the monitoring process, if CReaM notices some decreases in the available resources that are unacceptable with respect to the user level of satisfaction, it reacts to decrease the resource consumption. The reaction is to replicate data items in order to reduce the load on the peer; CReaM chooses the relevant data to be replicated and requests other peers to hold this replica. On the other hand, CReaM on requested peers accepts placing the replica if the user is still satisfied from the consumption of its resources.

2.1 CReaM Model

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

Temperature Degree (TD): indicates the importance of a data item DI for a given time period and from the point of view of neighbor nodes.

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

The Tolerance Thresholds: represent the allowable level of resource consumption specified by the user. We define three thresholds: **β** for the load level on the node's CPU, **μ**: for the allowed level of remaining battery, and **δ**: for the allowed level of remaining storage space. These thresholds will be used to monitor the peer's status and should influence the replication decisions.

As any replication model, CReaM answers the following questions:

2.1.1 When

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

- A DI become 'Hot' for neighbor nodes $TD(DI) \geq \alpha$.

- The user becomes unsatisfied from the availability of its resources. We define three functions to calculate the consumption of the three resources: BL(T) returns the remaining battery at time T, SS(T) 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 outputs of these three functions will be compared to the thresholds μ, δ, and β respectively.

Therefore, the answer to the "*when*" questions depends on a set of the four conditions named $CONDITIONS_M = \{TD(DI) \geq \alpha, BL(T) \leq \mu, SS(T) \leq \delta, NoR(T_{i-1}\text{-}T_i) \geq \beta\}$ and it is used to trigger the replication process on each node on the network when at least one of the above conditions becomes true.

2.1.2 Who

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

2.1.3 Where

The distribution algorithm avoids data item duplication on two neighbor nodes and places replica near the interested nodes. The user participates also in the replica placement process; by using the tolerance thresholds.

2.1.4 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 $CONDITIONS_M$). We classify the data items of each node into several categories; a data item may belong to one or more categories at the same time: $\mathbf{DI_\alpha}$: includes the hot data items on the node. $\mathbf{DI_\beta}$: contains the most requested data items of the last period of time. $\mathbf{DI_r}$: includes the rare data items that are important but rarely found on the peers. $\mathbf{DI_o}$: includes "not important" data item with TD = 0.

An action is initiated in case the user is unsatisfied; each resource is studied separately in order to define what actions to take:

CPU: the action is to replicate a hot DI that was requested a lot during last period (i.e. the candidate DI must belong to the intersection set of $DI_\alpha \cap DI_\beta$) in order to share the load of the request with other nodes and to reduce its NoR. However, if the load of the CPU keeps increasing, it would not be appropriate to keep replicating data items endlessly; so we define two values for the threshold β, soft value β_1 and hard value β_2 ($\beta_1 < \beta_2$); if β_1 is exceeded the node replicates but when β_2 is exceeded the node stop responding to any request.

Storage space: following the same logic, we define two values for the threshold δ. When $(SS(T) \leq \delta_1)$ the node accepts only the incoming urgent replication requests (RQ) and when $(SS(T) \leq \delta_2)$ the node removes replicas from the set DI_o by applying one of the well known cache management algorithms.

Battery: with the other resources, when $(BL(T) \leq \mu_1)$ the same action is applied as with the CPU's soft threshold (an element from the $DI_\alpha \cap DI_\beta$). If the hard threshold is exceeded the probability of disconnections becomes high so the node replicates an element belonging to the set DI_r in order to increase its/their availability. However, unnecessary replication may occur, if each node replicates a rare data item and loads the network by data that might be unhelpful to the connected nodes. To avoid this situation, we give priority to a DI that is rare and hot at the same time belonging to the set $DI_\alpha \cap DI_r$.

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 preemptively by replicating data items from $DI_\alpha \cap DI_\beta$.

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 $DI_\alpha \cap DI_r$
$BC \leq \mu_3$	HighB-Consumption	Replicate from $DI_\alpha \cap DI_\beta$ / DI_β

2.2 CReaM Architecture

CReaM is a middleware layer located between the operating system and the application layer. CReaM consists of the following components:

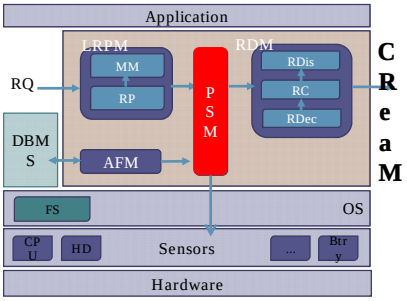

Figure 1. CReaM Architecture

2.2.1 Peer state Manager (PSM)

The PSM controls the replication process; it monitors the peer's status and decides based on this information when to replicate data items and when to accept placing replicas on the node. When the peer's status is not compatible with the user desire (e.g.: high battery consumption), (1) the PSM notifies the RDis to start the replicating process, (2) and it refuses placing the replica on the node. To monitor the peer's status it receives (1) warning about insufficient resources from the sensors and (2) the DIs categories from the AFM.

2.2.2 Replica Dissemination Manager (RDM)

Its responsibility is to replicate data on neighbor nodes by creating replication request and distributing it on the network. It consists of 3 subcomponents: (1) **Replica Decider (RDec)** that prepares a replication request (RQ) and chooses DIs to be replicated after receiving a notification from the PSM. (2) **Replica Controller (RC)** that determines the required number of replicas. (3) **Replica Distributer (RDis)** that sends the RQ on the network and follows the dissemination process to insure that the number of replicas is achieved.

2.2.3 Local Replica Placement Manager (LRPM)

The LRPM is responsible for placing the replica on the current node. It consists of two main components: (1) **Replica Placement (RP)** that decides to accept/refuse incoming RQ depending on the available resources (storage space, remaining battery, etc…) assigned by the user to the replication process. 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 temperature degree of DI and the owner's availability. (2) **Memory Management (MM)**: manages the available storage space dedicated to host replicas.

2.2.4 The access frequency Manager (AFM)

It monitors the access requests to all the data presented on the node. It keeps track of the values of TD, and warns the PSM whenever a data item becomes hot.

The subject of the next section is the PSM. It is the key component of both the architecture and the model. We thus started our implementation by developing it.

3. Peer State Manager (PSM)

The PSM helps other components in the architecture to accomplish the replication process. Depending on the values that the PSM monitors, the 'RDec' prepares 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 or refuse the incoming RQ. In this paper, we focus on the first part of the PSM task. Below we present in details the algorithm used by the PSM.

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. 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 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 (for example, low battery and high load on the CPU), which means that more than one status might be attributed to the node. 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>

The previous rules can be read in the following way: where the conditions are true then the actions are executed:

- **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 instead of BL(T), SS(T) and NoR (T_{i-1}, T_i) respectively
- **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, as outlined in Table 1. 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 1 such as replicating a DI or no longer responding to requests.

The process is initialized as follows: The PSM assigns values to the attributes, evaluates the conditions and checks if all conditions in a rule are satisfied. After verifying the conditions, 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 in the knowledge base 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 2.

Table 2 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_α
R_6	SS $\leq \delta_1$	A_6: Accepting just Urgent RQ
R_7	SS $\leq \delta_2$	A_7: Delete DI_o

Figure 2. 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.

4. CONCLUSION

In this paper we have presented CReaM which aims to increase data availability in MANETs. CReaM is a user-centric replication model that takes the user needs at high priority. A simulation-based implementation of this model has already been developed using the OMNet++ simulation environment. 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.

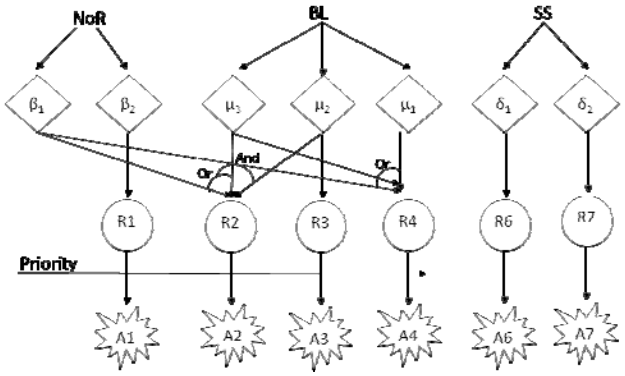

Figure 2. The Rule-Based Inference Engine of the PSM

5. REFERENCES

[1] Hara, T., Loh, Y.H., and Nishio S. 2003. Data replication methods based on the stability of radio links in ad hoc networks". In Proceedings of International Workshop on Mobility in Databases and Distributed Systems (Prague, Czech Republic, Sept. 2003).

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

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

[4] Nawaf, M.M., and Torbey, Z. 2009. Replica update strategy in mobile ad hoc networks. In Proceedings of the international ACM Student workshop on Management of Emergent Digital EcoSystems. (Lyon, France, Oct. 2009).

[5] Atechian, T., Torbey, Z., Bennani, N., and Brunie, L. 2009. CoFFee: Cooperative and InFrastructure-Free Peer-To-Peer System for VANET. In Proceedings of the 9th international ITS of Telecommunications. (Lille, France, Oct. 2009).

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

[7] Padmanabhan, P., and Gruenwald, L. 2006. DREAM: A Data Replication Technique for Real-Time Mobile Ad-hoc Network Databases. In the proceedings of the International Conference on Data Engineering. April,2006, p. 134- 134.

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

[9] Moussaoui, S., Guerroumi, M., and Badache, N. 2006. Data replication in mobile ad hoc networks. In Proceedings of the International conference on Mobile Ad Hoc and Sensors Networks MSN'06. (HongKong, December 2006).