Increasing data availability in mobile ad-hoc networks : A community-centric and resource-aware replication approach

Zeina Torbey Takkouz

Ordering N°: 2012ISAL0089

Year 2012

Thesis

Increasing Data Availability in Mobile Ad-hoc Networks:

A Community-Centric and Resource-Aware Replication Approach

Submitted to the
National Institute of Applied Sciences
(INSA of Lyon)

In fulfillment of the requirement for
Doctoral Degree

Doctoral School of Computer Science and Mathematics (InfoMaths)

Affiliated Area: Computer Science

Prepared by
Zeina Torbey Takkouz

Defended on September 28th 2012

Examination Committee

Prof. Abdelkader Hameurlain	Paul Sabatier University, Toulouse	Reviewer
Prof. Claudia Roncancio	Polytechnic Institute of Grenoble	Reviewer
Dr. Philippe Roose	IUT of Bayonne	Examiner
Dr. David Coquil	University of Passau, Germany	Examiner
Prof. Lionel Brunie	INSA of Lyon	Supervisor
Dr. Nadia Bennani	INSA of Lyon	Co-Supervisor

Abstract

A Mobile Ad-hoc Network is a self-configured infrastructure-less network. It consists of autonomous mobile nodes that communicate over bandwidth-constrained wireless links. Nodes in a MANET are free to move randomly and organize themselves arbitrarily. They can join/quit the network in an unpredictable way; such rapid and untimely disconnections may cause network partitioning.

In such cases, the network faces multiple difficulties, which in turn can dramatically impact the applications deployed on top of it. One major problem is data availability. Indeed, when the network gets divided into disconnected sub-networks, the hosts in one partition can no longer access the data located in others.

Data replication is a possible solution to increase data availability. However, implementing replication in MANET is not a trivial task due to two major issues. The first issue is the MANET resource-constrained environment: nodes have poor CPU, battery and storage capabilities, and the available bandwidth on the network is limited. Consequently, a replication system for MANETs should strive to consume as little resources as possible.

The second major issue is that the dynamicity of the environment makes making replication decisions a very tough problem. The constantly evolving topology of a MANET makes it impossible to compute an optimal replica placement, and the unpredictable distribution of requests complicates the selection of the "best" data items to replicate. Still, a replication framework for MANETs must be able to make beneficial decisions in spite of these problems.

In this thesis, we propose a fully decentralized replication model for MANETs. This model is called CReaM: "**C**ommunity-Centric and **R**esource-Aware R**e**plic**a**tion **M**odel". CReaM is designed to cause as little additional network traffic as possible. To this end, it works in an autonomous mode: contrarily to other replication systems, replication decisions are made at the node level without any global synchronization. This avoids a costly periodical exchange of negotiation messages.

To preserve device resources, CPU, storage space and battery levels are monitored on each node. When the consumption of one resource exceeds a predefined threshold, replication is

initiated with the goal of balancing the load caused by requests over other nodes. Thresholds are dynamically adapted depending on the actual user activity on the node.

The data item to replicate is selected depending on the type of resource that triggered the replication process. The best data item to replicate in case of high CPU consumption is the one that can better alleviate the load of the node, i.e. a highly requested data item. Oppositely, in case of low battery, rare data items are to be replicated (a data item is considered as rare when it is tagged as a hot topic (a topic with a large community of interested users) but has not been disseminated yet to other nodes). To this end, we introduce a data item classification based on multiple criteria e.g., data rarity, level of demand, semantics of the content. Furthermore, we present a rule-based inference engine that, based on the resources status, selects the best data item to replicate.

To select the replica holder (i.e., the node that will receive a data item replica), CReaM integrates a prediction engine that estimates which location should provide the best load balancing and the best data usability. To this end, we propose a lightweight solution to collect information about the interests of participating users. Users interested in the same topic form a so-called “community of interest”. Through a tags analysis, a data item is assigned to one or more communities of interest. Based on this framework of analysis of the social usage of the data, replicas are placed close to the centers of the communities of interest, i.e. on the nodes with the highest connectivity with the members of the community.

We evaluated the performance of CReaM by a series of experiments using the OMNet++ simulation environment. The results show that CReaM has positive effects on its main objectives. In particular, CReaM allows maintaining a level of resource consumption compliant with the user requirements. Furthermore it imposes a dramatically lower overhead than that of traditional periodical replication systems (less than 50% on average), while it maintains the data availability at a level comparable to those of its adversaries.

Keywords: Mobile ad-hoc network (MANET), Vehicular ad-hoc network (VANET), Data Availability, Replication, Resources Monitoring, Communities of Interests, User Satisfaction.

Résumé

Les réseaux ad hoc mobiles ou MANETs sont des réseaux qui se forment spontanément grâce à la présence de terminaux mobiles, appelés communément nœuds, dotés de technologies de communication adaptées. Ces réseaux sans fil sont de faible capacité. Les nœuds se déplacent librement et de manière imprévisible au sein du réseau. De plus, ils se déchargent très rapidement. En conséquence, un réseau MANET est très enclin à subir des partitionnements fréquents.

Les applications déployées, sur de tels réseaux, souffrent de problèmes de disponibilité des données induits par ces partitionnements. La réplication des données constitue un des mécanismes prometteurs pour pallier ce problème. Cependant, la mise en œuvre d'un tel mécanisme dans un environnement aussi contraint en ressources (faible bande passante du réseau, capacité de calcul et de mémoire réduites, faible charge électrique) constitue un réel défi. L'objectif principal est donc de réaliser un mécanisme peu consommateur en ressources.

Le second objectif de la réplication est de permettre le rééquilibrage de la charge induite par les requêtes de données. Le choix des données à répliquer ainsi que celui des nœuds optimaux pour le placement des futurs réplicas est donc crucial, spécialement dans le contexte du MANET, sujet à des changements de topologies incessants.

Dans cette thèse, nous proposons CReaM (**C**ommunity-Centric and **R**esource-Aware R**e**plic**a**tion **M**odel") un modèle de réplication adapté à un environnement mobile sans infrastructure. CReaM fonctionne en mode autonomique : les prises de décisions se basent sur des informations collectées dans le voisinage direct du nœud plutôt que sur des données globalement synchronisées impliquant tous les nœuds, ce qui permet de réduire le trafic réseau lié à la réplication.

Pour réduire l'usage des ressources induit par la réplication sur un nœud, les niveaux de consommation des ressources sont contrôlés par un moniteur. Toute consommation excédant un seuil prédéfini lié à cette ressource déclenche le processus de réplication. Pour permettre le choix de la donnée à répliquer, nous avons proposé une classification multi critères : rareté de la donnée (une donnée est rare si elle appartient à un sujet demandé mais qu'elle-même n'a pas été répliquée précédemment), sémantique, niveau de demande ; et un moteur d'inférence

qui prend en compte l'état de consommation des ressources du nœud pour désigner la catégorie la plus adaptée pour choisir la donnée à répliquer.

Pour permettre de placer les données répliquées au plus près des nœuds intéressés tout en privilégiant l'équilibre de la charge en requêtes, CReaM propose également un mécanisme autonomique et proactif pour l'identification et le maintien à jour des centres d'intérêt des nœuds. Les utilisateurs intéressés par un même sujet constituent une communauté. Par ailleurs, chaque donnée à répliquer est estampillée par le ou les sujets au(x)quel(s) elle s'apparente. Ainsi, chaque nœud est en mesure de calculer puis de diffuser autour de lui son potentiel sur un ensemble de sujets c'est-à-dire sa capacité à être connecté à une communauté importante sur ce ou cet ensemble de sujet(s). Un nœud désirant placer un réplica apparenté à un sujet choisira le nœud ayant la plus grande communauté sur ce sujet.

Nous avons évalué les performances de CReaM grâce à un simulateur développé sous OMNet++. Les résultats obtenus confirment d'une part la capacité de CReaM à améliorer la disponibilité des données au même niveau que les solutions concurrentes, tout en réduisant de moitié, la charge liée à la réplication. D'autre part, ces résultats prouvent que CReaM permet de respecter l'état de consommation des ressources sur les nœuds.

Mots clés: Réseau ad hoc mobiles (MANET), Disponibilité des données, Réplication, Monitoring des ressources, Communautés d'intérêt.

Table of Content

INTRODUCTION

1.1. Introduction

Wireless and mobile communications have experienced an unprecedented development during the past decade. Advances in computer hardware, telecommunication, and software technologies have rendered mobile devices and wireless networking commonplace and powerful. Several new wireless communication technologies have been introduced, such as Wi-Fi [87], Bluetooth [19] and ZigBee [1]. Most of the new mobile wireless applications that appeared have been deployed over networks associated with a fixed infrastructure through a base station. Nevertheless, the progress in hardware and software technology have also made possible a new alternative way for wireless communication, in which devices form a self-creating, self-organizing and self-administering wireless network. This new type of network is called Infrastructure-less Wireless Network, also commonly known as Mobile Ad-Hoc Network (MANET) [71].

Mobile ad-hoc networks are a very effective opportunity of truly pervasive computing. They can indeed be set up dynamically, anywhere, and anytime. MANET users are free to move randomly and organize themselves arbitrarily. As a consequence, the network topology may change rapidly and unpredictably, so the nodes are responsible for discovering each other in a dynamic way.

MANETs are usually constructed using scarce-resources devices equipped with a wireless interface with a limited communication range. Hence, users that are within each other's range can communicate directly with one hop connections; otherwise intermediate nodes must operate as routers to establish multi-hop connections.

MANETs are support environments for autonomous and decentralized applications used in many contexts: military battlefield, emergency services, commercial and civilian applications, entertainment applications, and the applications based on inter-vehicle communications [76].

However, the dynamic and unpredictable nature of MANETs creates several challenges, which in turn impact the applications deployed on top of them. In this

thesis, we investigate the issue of data availability, which represents a major and still unresolved problem for data sharing applications running on top of MANETs.

To motivate and illustrate our approach, we present in the next section two application scenarios in the context of mobile ad-hoc networks. The related research problems and our main research contributions in this area are identified in section 1.3. Finally, we conclude the introductory chapter by outlining the structure of the thesis in section 1.4.

1.2. Motivation

1.2.1. Application Scenarios

In the following, we present two application scenarios that illustrate the objectives of this thesis.

1.2.1.1. Application Scenario1: Mobile Social Network

Tonight is the night of the music festival in France[1]. *The administrator of the festival's web site has made the "Live Fiesta" application available for download. "Live Fiesta" is a mobile social application that works on any mobile network and does not* require any Internet connection. Its users can share self-produced information related to shows such as their location, information about the performers, comments, etc. These data are stored on the device of the user who produces them, but can be retrieved by other users connected with the data producer. The application also provides a query functionality to let users locate devices that hold information about a specific show.

Marie is a fan of rock music. She has agreed with her friends to go out and watch *some shows. Before going out, she downloads "Live Fiesta" on her smart phone.*

[1] This festival is called "La fête de la musique" in France, and it takes places each year on June 21.

During the evening, she attends several shows and enjoys herself by sharing her comments.

Paul is a Syrian PhD student and rock music fan. He came to France to participate to a scientific conference. The conference is now over, and his last night in France corresponds to the festival. Having *downloaded "Live Fiesta", he* decides to take a walk in the streets and check out some shows. In order to find interesting shows, Paul *launches "Live Fiesta", which connects* him to an exist*ing "Live Fiesta" MANET. He* starts reading the comments and looking for rock shows. Paul receives *Marie's* commentaries about a good show taking place three streets away from him, so he decides to go directly and attend this concert.

1.2.1.2. Application Scenario 2: Online Gaming

During the summer holiday period, Joseph is driving his car on the highway, heading from Lyon to Barcelona with his family.

Marc, Joseph*'s son, starts to get bored. He tries to connect to the Internet to play his favorite game, "Dream Land". It is a multiplayer game, which is available on the Internet for free. Users can play directly on the game's website, or they can download* it and install it on their personal device. In the second case, if the device has wireless capabilities it is possible to play the game over a MANET connection. The *synchronization process is done upon successful connection to the game's website.*

*In "Dream Land", eac*h player builds his own village; he constructs buildings for living, shops to earn money, and resorts for fun. The player creates missions in order to operate his shops and invites other players to participate. Thus, each player can announce his missions a*nd query other players' announcements. An announcement* comes in the form of an image or a short video describing the mission i.e., the mission itself, which consists in a mini-game such as a quiz or a puzzle, and the associated reward. The reward enables players to unlock levels and to progress in the game. When a player finds an announcement for a job that he likes, he applies for it. He can then start playing the game corresponding to the mission. When the player beats the

game, the mission is completed and the player can then collect his reward by connecting to the Dream Land Web server.

The 3G coverage on the road is bad, and Marc cannot connect to the game's server. *Thus, he decides to look for passengers in the cars around his father's car who might* have installed the game on their smart phones. He connects to the public ad-hoc network initiated by one of them, launches the game and is happy to identify three other players who have posted job announcements. This enables him to play as with the infrastructure-based version of the game; the only difference is that he will have to wait until he connects to the Internet to collect his reward.

1.2.2. Analysis: Constraints and Requirements

These scenarios expose examples of information sharing applications in MANETs, which is the type of applications targeted in this thesis. In the following, we analyze the common and specific features that can be identified in both scenarios in order to determine the requirements for operating such applications in this type of environment.

The common characteristics are:

- The network is constructed on the fly without any infrastructure. In addition, no central point can be used to manage the communication between different users.

- All users move freely at different speed. They can join and leave the network in an uncontrollable way. Thus, the topology of the network changes rapidly and unpredictably.

- Any device can be used as long as it has wireless connectivity. In the general case, devices are heterogeneous in terms of their capabilities and characteristics such as battery, screen size, computing power, memory size, etc. Usually, they are resources-constrained and battery-powered.

- The wireless link capacity may vary over time because of different factors e.g., the distance between nodes, the signal transmission power, obstacles. In general, the bandwidth is limited and much lower than which is available in wired networks.

However, the scenarios have also some different characteristics:

- In the first scenario, the size of the data is rather small. In the online gaming scenario on the other hand, the application requires exchanging larger multimedia files.

- The speed of the users varies between a pedestrian speed (in the first scenario) and a car speed (in the second scenario). A high speed movement significantly influences the communication; thus, in the second scenario, two cars moving at high speed will usually stay within the communication range for a very short time, and hence, be able to communicate only a small amount of information.

From this analysis, we can conclude that the constraints showed by MANETs impose a number of requirements for successfully deploying data sharing applications over such an environment. These requirements are the following:

- **Adaptation of the application to the MANET environment**

The application must be adapted to operate in a fully decentralized manner, without relying on a central server or services available from the infrastructure. It must support heterogeneous devices with poor capabilities (e.g. screen capacity, computation power, etc), and take into account the limited bandwidth and poor quality of wireless links, which affect data exchanges.

- **Mechanism to increase data availability**

Data availability is a critical issue for almost all applications running on top of a MANET. To run “Live Fiesta” or to play “Dream Land” under satisfying conditions, the shared information must be accessible at any time from any participating user. For

Live Fiesta (scenario 1), Marie's comments should be made available to interested users even after she disconnects or moves to another location. In Dream Land (scenario 2), the viability of the game depends on the collaboration and the diffusion of the available announcements. However, the players may disconnect unpredictably as a result of the cars' mobility. Hence, there is a need for a specific mechanism aimed at ensuring data availability and facilitating data sharing.

- **Efficient mechanism to exchange multimedia data**

Exchanging large size data in MANETs is difficult because this requires long connection times between nodes, which may not be possible if the nodes move and leave each other's communication range. The problem is particularly challenging when the mobility of the nodes is high. For example, in the second scenario, Marc may connect to another player in a car moving in a direction opposite to his own car; this car will stay inside the connectivity area only for a short time that will not be sufficient to completely transfer a video announcement. Thus a specific mechanism to support multimedia file transfer, and more generally large size, is required for such highly mobile MANETs.

- **Efficient routing and dissemination protocols**

Unlike traditional networks, MANETs cannot rely on a dedicated routing infrastructure to disseminate requests and responses. In the first scenario, routing the queries of Paul to Marie is already a challenging task. In addition, if several users hold the same data (for example Marie's friends) and several queries target them, it is desirable to distribute the queries over the whole set of devices holding the data in order to balance the corresponding load. This outlines the fact that efficient routing and dissemination algorithms are mandatory in order to efficiently share information in a MANET.

1.2.3. Scope of the Thesis

From the previous section, we argue that there is a need for a **middleware** providing services to distributed data sharing applications deployed on top of MANETs.

Routing and dissemination protocols for MANETs have been extensively investigated – see for example the surveys [4, 74] and the review of routing protocols we provide in the state of the art (see section 2.4). We build on these works by assuming that such a service is available.

In this thesis, we focus on **data availability management**, which we consider as the most important service. We especially investigate how to use the replication mechanisms to increase the data availability. We also study the issue of the **exchange of multimedia files in highly mobile MANETs**.

In the next section, we take a closer look at the problem of increasing the data availability in MANETs, derive the related research issues and present the corresponding contributions presented in this thesis.

1.3. Research Problems and Contributions

1.3.1. Increasing Data Availability

Users in mobile ad-hoc networks move freely and join/quit the network in a random way. These characteristics lead to rapid topology changes and to unpredictable wireless disconnections. Such rapid and untimely changes and disconnections may cause network partitioning.

In fact, when network partitioning occurs due to the migrations of users (so-called mobile nodes), the network faces multiple difficulties, which in turn impact the applications deployed on top of it. One of the major issues is the problem of data availability. When the network is divided into disconnected sub-networks, the mobile nodes in one partition can no longer access the data held by nodes in others. This makes data availability in MANETs inherently lower than in fixed networks. Taking measures to prevent or at least reduce the deterioration of data availability in case of data partitioning is thus a key requirement for successfully implementing applications in MANETs.

In order to increase data availability in MANETs, two types of solutions can be considered. The first category is based on caching [106, 107]; it consists in keeping copies of the data that pass through mobile nodes while they act as routers. The second category is based on replication mechanisms [34, 59, 78, 82]; this solution consists in copying data on other mobile nodes. Indeed, when a disconnection occurs, there is a higher probability for a mobile nodes to find a data item if multiple copies of it exist in the network. This idea is illustrated in Figure 1-1. In fact, both categories of solutions improve significantly the data availability. Furthermore, a solution can be a combination of both mechanisms.

A: Data loss before replication

B: Data are available after replication

Figure 1-1: Increasing data availability using replication mechanism

In this thesis, we deal with the problem of data availability from two perspectives. First, we propose a model for MANETs, called CReaM, based on a replication mechanism. This model concentrates on the monitoring of the environment to achieve efficient replication.

Second, we propose a multimedia content exchange system for VANETs, called CoFFee. CoFFee, in the same way as CReaM, implements a replication mechanism to keep data available on the network. However, it addresses as well the problem of exchanging heavy multimedia data in highly mobile networks by partitioning the files and transferring the fragments instead of the complete file.

1.3.2. CReaM: Community-centric and Resource-aware Replication Model

Actually, replication in the context of mobile networks is applied in a way that differs from that in a centralized network. In the latter, specific "spots" can be statically identified as long-term good choices to allocate replicas due to their position in the network topology. In MANETs, the dynamicity of the network imposes a dynamic replication scheme. This imposes to overcome a number of specific issues. In the following, we detail each issue followed by a presentation of the contribution addressing it in the thesis.

- **Issue 1: Limitations of the network**

Since a MANET is a decentralized network, a central point to manage the replication does not exist. Thus, each node must participate in the replication service by collecting information about its vicinity and exchanging negotiation messages. On the other side, wireless connections have limited bandwidth which must be reserved in priority for the applications. Consequently, the resource overhead imposed by running the replication system must be minimized.

➢ **Autonomous and decentralized decisions**

To overcome this problem, we propose a replication model specifically designed to cause as little additional network traffic as possible. To this end, it works in an autonomous mode, in which replication decisions are made locally without explicit coordination with other nodes. This is an original point compared to existing work in the literature, which all implement a form of explicit coordination to make replication decisions.

The decision of the node to place the replica is made locally by the data owner and based on a partial view of the network. This partial view is maintained by each node using two types of messages: first the messages exchanged during the normal network activity. Second, small specific messages exchanged among neighbors only when necessary.

- **Issue 2: Limitations of the devices**

The second challenge lays in the limitation of resources of the devices. Nodes have limited CPU, battery and storage capabilities. This is an issue for the replication system. Indeed, when a node participates to the system, it carries replicas and answers data requests concerning them, and consequently consumes its scarce resources. Users would be unsatisfied if this additional resource consumption would hamper them when they are trying to accomplish personal tasks on their devices.

➢ **Resource-aware replication**

In the replication model that we propose, replication decisions are based on the monitoring of resource consumption. More precisely, replication is triggered when a high level of resource consumption is detected, with the goal of sharing the load of data requests processing with other nodes. The evaluation of the consumption takes the actual user activity on the node into account. This guarantees that a satisfactory amount of resources is reserved for the user's personal tasks. In addition, the model privileges nodes with available resources

as candidate replica holders and excludes nodes that could not assume the load of carrying replicas and processing queries.

- **Issue 3: Replica selection and allocation in a decentralized and unstable network**

Replica selection and allocation decisions are critical factors for the benefit provided by a replication system. This is even more the case in a resource-constrained environment in which only a limited number of replicas can be created. In addition, making such decisions locally and with only a partial knowledge of such a very dynamic network as we proposed above (issue1) is a very difficult problem.

➢ **Community centric approach**

In order to improve the quality of the replica selection and allocation decisions in our model, we propose to take **semantic properties** of data items and **user interests** into account. Indeed, users do not query data at random; their goal is to retrieve documents that match their interests. Thus, considering previous accesses and topics that interest users and matching this information with the topics of data items available for replication can give hints on the data items that should be replicated and on the nodes where they should be placed.

To this end, we propose a lightweight solution to collect the interests of participating users. This information is used to build communities of interest i.e., set of users interested in the same topic. Hence, during the replica placement process, the replica is assigned to one or more communities based on tags that describe its content. The replica is then placed on the node with the highest connectivity with the community members.

The selection of the data item to be replicated favors data that are tagged by topics that interest a large number of users (identified by the monitoring of previous accesses) and are rare in the network.

To summarize, CReaM is a decentralized replication model designed for mobile environments. It is characterized by its autonomous operating intended to decrease as much as possible the bandwidth overhead. To preserve the devices resources, CReaM integrates a monitoring engine that initiates the replication process when resources are no longer sufficient to complete the user's tasks. The selection of the data items is based on the semantics of the data content and on the monitoring and analysis of previous accesses. The replica placement process selects the node to carry the replica based on two conditions: its connectivity with the community of interest is high and it has enough resources to support the load of the replica management. A middleware implementing CReaM has been developed and validated by experiments that evaluated its performance.

1.3.3. CoFFee: Collaborative and Infrastructure Free Peer-to-Peer System for VANETs

In this thesis, we have also dealt with the problem of multimedia content exchange in highly mobile networks such as VANETs (Vehicular Ad-hoc NETworks, formed by vehicles interconnecting in an ad-hoc fashion).

The main challenge when exchanging a large-size file is to make sure that the exchanged file will be fully delivered to its destination. This is especially difficult in MANET environments, as the connectivity between nodes can be interrupted after a very short time because of the high mobility of VANET nodes.

CoFFee is a multimedia content distribution system tailored for VANETs. It is achieved in collaboration with Talar Atechian, a PhD student at LIRIS laboratory and it bases on her research works. CoFFee partitions large-size files into fragments to facilitate the transmission process and to optimize the use of the bandwidth. The file

fragments are distributed over the network so that when a request is disseminated each node that holds a fragment can participate in the transmission process.

In addition, CoFFee is characterized by the cooperation between the nodes in order to ease the fragments transmission among the nodes. Indeed, nodes cooperate in order to detect redundant fragments and so avoid their transmission multiple times and consequently reduce the bandwidth usage. Finally, CoFFee addresses data availability issues, as nodes cooperate to detect the fragments considered as rare fragments. These rare fragments are then replicated in order to increase the fragments availability in the network.

1.4. Structure of the Thesis

The remainder of the thesis is organized as follows:

Part I: State of the Art

- Chapter 2 "Technological Background": presents the state of the art of the technologies on which our work is based.
- Chapter 3 "Survey: Replication Systems for MANETs and Contribution Positioning": surveys replication systems proposed for mobile ad-hoc networks in the literature, in addition to the contribution positioning.

Part II: CReaM

- Chapter 4 "CReaM: Triggering Replication and Selecting Data": describes the first two processes of our replication model: the replication initiation process and the data selection process.
- Chapter 5 "CReaM: Replica Placement": details the replica placement in CReaM, as well as the protocol used to manage the information used by the replica placement process.

- Chapter 6 “Simulation and Experimentation Results”: covers the simulation that has been carried out to evaluate the performance of CReaM.

Part III: CoFFee

- Chapter 7 “CoFFee: Cooperative and Infrastructure-Free peer-to-peer System”: presents our second contribution CoFFee, a peer-to-peer system for multimedia data exchange in VANETs.

Conclusion

- Chapter 8 “Conclusion and Future Works”: presents a summary of our contributions, a conclusion and highlights future work.

Finally, the bibliography and the list of the abbreviations used in the thesis with their descriptions are presented.

I

STATE OF THE ART

TECHNOLOGICAL BACKGROUND

Research efforts have been employed for solving problems related to mobile ad-hoc networks in the different layers of the OSI model. In this thesis, we target the application layer. However, we present in this chapter a brief description of the technologies implemented in other layers that are necessary for application layer. We divide the chapter as follow: section 2.1 provides a general presentation of mobile ad-hoc networks and vehicular networks. The target applications for ad-hoc networks are presented in section 2.2. Wireless technologies of the physical layer of the OSI model are presented in section 2.3. Finally section 2.4 is dedicated to present taxonomy of routing protocols. We conclude the chapter in section 2.5.

2.1. Mobile Ad-hoc Networks

Ad-hoc is a Latin expression meaning “for this”, with the further meaning “for this purpose only”. It provides a good and emblematic description of the reason of the need for ad-hoc networks. Indeed, MANETs can be set up anytime and anywhere without the need of external infrastructures (like wires and base stations) [71].

A Mobile Ad-Hoc network, known also as MANET, is a self-configured infrastructure-less network of autonomous mobile devices (named also nodes), that communicate over bandwidth constrained wireless links [71]. A MANET can be set up dynamically anywhere and anytime. The nodes in a MANET are free to move randomly and organize themselves arbitrarily; thus, the network’s topology may change rapidly and unpredictably; so nodes are responsible for discovering each other in a dynamic way [96].

A MANET can show various forms; it can operate in a standalone[2] fashion (node-to-node) or may be connected to an infrastructure such as Internet (node-to-Infrastructure). Ad-hoc networks are very suited to use as a free cost communication infrastructure. In addition, it is possible that the nodes connected to an Infrastructure contribute to propagate and to extend the network to other nodes localized outside the coverage of a given infrastructure. Usually, the devices used in a MANET are mobile

[2] A standalone MANET is a MANET with no connection to the Internet. This is the kind of MANET considered in this thesis.

computing devices like personal digital assistants (PDAs), laptops, smart phones, etc. They are equipped with a wireless interface with limited communication range. The nodes that are within each other's range can communicate directly with a one hop connection. However, in order to enable communication between nodes that are not directly connected (being within a same communication range), intermediate nodes can be selected to relay packets generated by other nodes to their destination. This is called a multi-hop connection.

Ad-hoc networks have been also widely studied in highly mobile environments, particularly in inter vehicles communications. These environments are known as VANETs (Vehicular Ad-hoc NETwork); they aim to allow wireless communication among nearby vehicles. Thus, vehicles that are approximately 100 to 300 meters one to other can be interconnected using a wireless link. The high mobility of the nodes is the main characteristics of VANETs which leads to rapid changes in the network topology.

Inter vehicular communication targets two main categories of applications: safety applications and non-safety applications also known as infotainment applications. As an example of safety and comfort driving, the Intelligent Transportation Systems (ITS) that share traffic information between connected vehicles; this may help avoiding collisions and traffic jams on road networks. Indeed, many projects have been developed for VANETs like FleetNet [33], CarTALK [69], PReVANET [70], etc. Major European car manufacturers (e.g. Volvo, BMW, Audi, Renault, etc…) are actually working on the Car2Car project [26], which aims to be a key contributor to the specification and the standardization of the technologies used in vehicular ad-hoc networks.

Infotainment applications for VANET are interesting and promising applications [91]. They are mainly used for sharing multimedia files (e.g. video, audio, pictures) and to create an environment for multiplayer games. Figure 2-1 shows an example of such a VANET, composed of mobile vehicles and fixed base stations. The illustrated scenario explains that it is possible to enable a Vehicle-to-Vehicle (V2V)

communication, directly between the vehicles as well as Vehicle-to-Infrastructure (V2I) communication through existing base stations on the road network.

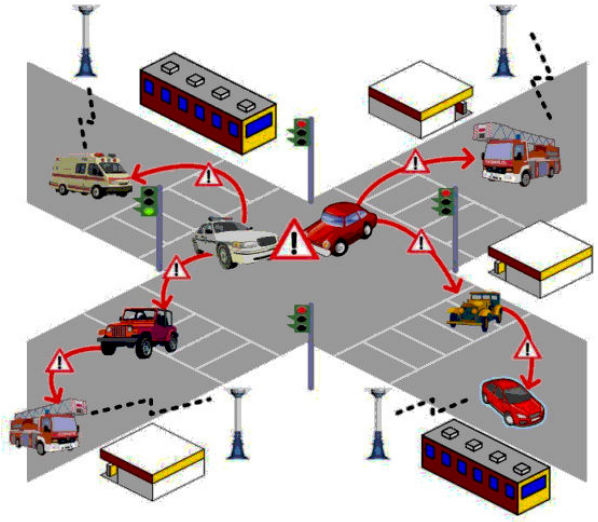


Figure 2-1 : Vehicle ad-hoc network (http://monet.postech.ac.kr/research.html)

2.2. Mobile Ad-hoc Network Applications

The target application areas for ad-hoc networks are manifold:

- The military battlefield and the emergency services (e.g. firefighting or disaster recovery) are the main application areas. When a lack of infrastructure is detected, a MANET is created and organized in an independent and autonomous manner.

- Another type of application area concerns commercial and civilian applications. A lot of examples for such applications and services can be provided: distributing advertisement in shopping malls or in sports stadiums, distributing trip schedules and notifications in train stations or in airports, distributing shared documents in university campus, etc.

- One Laptop per Child Project Application [65]: this non-profit project aims to sell cheap and durable laptops to governments such as Africa. The laptop is a

special prototype designed for the project which serves as learning tool for its recipient. One particular aspect of the design is that the computers work in an ad-hoc network mode and can share an Internet connection. The computers are equipped with a long life battery, but are limited by their low processing capability and storage space.

- In addition, several interesting applications are proposed especially for highly dynamic ad-hoc environments (VANETs); these applications are mainly classified into safety (Intelligent Transportation Systems) and non-safety applications (Infotainment) as mentioned in the section above [91,69, 70].

The deployment of these aforementioned applications needs an existing ad-hoc network, created between mobile nodes. Therefore, it is crucial when designing an application for MANETs to consider the radio communication established between the nodes. In addition, since the objective is to connect the nodes to each other, routing protocols of are important.

In the next two sections, an overview is presented regarding wireless technologies and routing protocols for ad-hoc networks.

2.3. Wireless Technologies

The establishment of an ad-hoc network requires some basic equipment at the physical layer of the OSI model. The most popular wireless technologies standardized and known worldwide are Bluetooth [19], ZigBee [1] and Wi-Fi technologies [87][3].

Bluetooth allows devices to communicate over short distance but with fast transmission speeds. The standard frequency band is equal to 2.4 GHz. It has the shortest range of coverage (5 to 10 meters) comparing to ZigBee and Wi-Fi. However,

[3] Other candidate wireless technologies for VANETs are Cellular, Satellite, and WiMAX (IEEE 802.16).

it is a standard and it offers a communication protocol designed for low power consumption.

ZigBee's coverage range reaches 100 meters without obstacles. It is a low-cost, low-power, wireless mesh networking proprietary standard. This low cost allows the technology to be widely deployed in wireless control and monitoring applications; its low power-usage allows a longer life with smaller batteries, and the mesh networking provides high reliability and larger range. ZigBee operates within three different frequency ranges 868MHz, 915MHz, and 2.4GHz.

Finally, the wireless LAN (Wi-Fi) technology is a data transmission system designed to provide location-independent network access between computing devices by using radio waves rather than a cable infrastructure. The Wi-Fi alliance defines Wi-Fi as any Wireless Local Area Network (WLAN) products that are based on the IEEE802.11 standards.

Wi-Fi devices communicate with each other with the help of a controller-device known as a wireless access point or "hot spot". Hot spots usually combine three primary functions; physical support for interfacing wireless and wired networking, routing between devices on the network and service provisioning to add and remove devices from the network. The Wi-Fi Alliance is nearing completion of a new specification, named Wi-Fi Direct, to enable Wi-Fi devices to connect to one another without wireless access points [109]. It allows devices equipped with Wi-Fi communication technology (IEEE 802.11a, 802.11g or IEEE 802.11n) to get involved in an ad-hoc network by embedding a software access point into these devices.

A typical wireless access point using 802.11b or 802.11g amendment has a range of 32 meters indoors and 95 meters outdoors. Other amendments have a wider range, such as IEEE802.11g (70 meters indoors and 250 meters outdoors). Typically, a Wi-Fi system transmits either in 2.4 GHz or 5 GHz.

Table 2-1 compares the three wireless technologies presented above:

Table 2-1: Wireless technologies

Technology	ZigBee	Bluetooth	Wi-Fi
IEEE	802.15.4	802.15.1	802.11a/b/g/n/p
Number of nodes	65000+	7	32
Transfer Speed	250Kb/s	1 Mb/s	11-45-108-320Mb/s
Range	100m	5-10m	250m
Frequency	868-915-204 MHz	2.4 GHz	2.4 or 5 GHz

The maturities of communication and computing technologies indicate the feasibility of MANETs to allow mobile devices to communicate with each other anywhere and anytime. Thus, in our thesis, we give more emphasis on higher layers in OSI model.

2.4. Routing Protocols

The communication in wireless ad-hoc networks is ensured through routing paths connecting a source node to a destination sink node. The routing path is established by a set of mobile nodes that cooperatively route the packet from one node to another. For this purpose, several routing protocols well-adapted for ad-hoc networks have been proposed. The main goal of MANET routing protocols is to find and maintain a path in a dynamic topology by using minimum resources. The performance of routing protocols is evaluated by several performance metrics [52], such as the end-to-end delay and overhead.

In this section, we basically concentrate the comparison of the routing protocols on these two metrics. Note that, the end-to-end delay is the time interval when the packet is sent until the time the packet is received at the destination. The overhead in the network is the amount of data transmitted; it is measured in terms of packets that are transmitted in excess of the payload (data).

MANET routing protocols can be classified into four main categories: reactive protocols, proactive protocols, hybrid protocols and geographic location-based protocols.

Reactive protocols, known also as on-demand routing protocols, aim to find a routing path from a source to a destination on-demand, which means, whenever a communication is needed. Thus, when a source node wants to transmit a message, it floods a query to discover the route to the destination. This disseminated discovery query is called the Route Request (RREQ) and the mechanism is called Route Discovery. The destination replied with a Route Reply (RREP) to confirm the establishment of the path in a proper manner. The discovered route is maintained until the end of the communication or when one of the nodes (source or destination) becomes inaccessible. Examples of reactive protocols are the Dynamic Source Routing (DSR) [45] and the Ad-hoc On-Demand Distance Vector (AODV) [22] protocols.

Oppositely, proactive protocols also known as Table-Driven protocols, aim to prepare in advance routing paths that can be used when needed. In proactive routing, each node maintains a routing table containing information of the possible routes to any node in the network. In order to update the information in the routing table, two kinds of table update are identified: periodic update and triggered update. In periodic update, each node periodically broadcasts its table to its neighbors. However, in triggered update, when a node detects a change in its neighborhood, it broadcasts entries in its routing table that have changed. Examples of proactive routing protocols are the Destination-Sequenced Distance-Vector protocol (DSDV) [46] and the Wireless Routing Protocol (WRP) [61].

The Hybrid protocols seek to join the advantages of proactive and reactive protocols. Thus, the goad of hybrid protocols is to use a reactive routing procedure at the global network level while employing a proactive procedure in the nodes local neighborhood. Zone Routing Protocol (ZRP) [12] is an example of such hybrid routing protocols.

Because of the mobility of the nodes, position-based geographic routing protocols have been proposed. These protocols consider the location of the nodes as a key factor for the routing algorithm. Therefore, the geo-positions of the nodes are exchanged and stored in the routing table of each node to be used during the routing path discovery.

In general, geographic routing protocols are more suitable for vehicular ad-hoc networks (VANETs). Examples of geographic position-aware routing protocols are Location Aided Routing (LAR) [49], DREAM [11], DG-CastoR [7] and GeoTORA [48].

The routing protocol must be suitable to the environment. When the mobility of the nodes is not considered high, a reactive or a proactive protocol can be used. However, it is important to choose a routing protocol that respond to the attended QoS. The route discovery process implemented in reactive protocols may increase the end-to-end transmission delay because of the exchange of the route requests and routes responses. In proactive table-driven protocols, the routing paths are established in advance in the routing table. However, generates an overhead in the whole network which may be high, because of the periodic exchange of some information needed to construct in advance all paths to all destinations in the network.

If the mobility of the nodes is detected high in VANETs, it is more convenient to use geographic routing protocols that consider the location of the nodes to construct optimal paths from the source to the destination. Table 2-2 presents a summary of the routing protocols.

Table 2-2: Routing protocols

Category	Protocols	Properties	Mobility
Reactive	DSR - AODV	Low overhead	Low
Proactive	DSDV - WRP	Low end-to-end delay	Low
Hybrid	ZRP	-	Low
Geographic Location-based	DREAM – GeoTORA – DG-CastoR - LAR	proactive or reactive	High

2.5. Conclusion

In this chapter, we presented the basic technologies and protocols indispensable for implementing applications on top of a MANETs. A brief description of the existing

radio wireless technologies used to form ad-hoc network (Bluetooth, ZigBee and Wi-Fi) was given. Each technology has specific features that have to be considered for suitable scenarios. Concerning routing protocols, many research works have been done and which can be categorized in four classes: reactive protocols, proactive protocols and geographic protocols. We conclude that the advance in the wireless technologies and the routing protocols makes the deployment of the applications on top of MANETs and VANETs possible.

SURVEY: REPLICATION SYSTEMS FOR MANETS AND CONTRIBUTION POSITIONING

Data replication is an appealing technique to provide services on unreliable networks caused by poor connections, host failures, and network partitions. It has been studied for a long time in the areas of distributed file systems, distributed database systems, and recently in peer-to-peer information storage and mobile environments. This chapter presents a survey of data replication systems for mobile ad-hoc networks and positions the contribution of this thesis with respect to existing replication techniques. It is divided as follows: Section 3.1 presents the classification used to categorize the different works. The survey of replication systems for MANETs is exposed in section 3.2 where the systems are classified into 6 categories. The features used to compare all existing works are detailed in section 3.3. Section 3.4 positions our contribution with respect to the replication systems presented in this chapter. Finally, we conclude the chapter in 3.6.

3.1. Classification

The previous works can be classified according to two different classifications: constraint-based classification and architecture-based classification.

3.1.1. Constraints-Based Classification

A replication system designed for mobile ad-hoc network must consider the limitations of the environments. Actually, each proposed replication system considers one (or more) issue concerning those limitations (presented in section 1.2.2). After a general study of the existing works, we have identified the following issues as the most important ones to consider in the design of a replication scheme: the network limitations, the resources limitations, and the replicated data items. Based on these issues, we can define the following categories of replication techniques:

- ✓ **Resource-limitations-aware Techniques** consider the availability of at least one resource (CPU, storage space and battery) during the replication process.

- ✓ **Network-limitations-aware Techniques** support the network constraints; they are adapted to work in an infrastructure less network; they manage the mobility of nodes and consider the network partitioning and the limited bandwidth.

- ✓ **Data-aware Techniques** adapt the solution to consider the size of replicated data items or the content of the data items.

Actually, some techniques consider multiple issues at the same time; to this end, we distinguish more categories where the name of the category includes the considered issues. For example, a replication technique which considers the resources limitations and the network limitations belongs to the category Resource-Network-limitations-aware Techniques.

Furthermore, we identified a category of replication techniques that is proposed to increase data availability but does not consider any of the above issues; we called this category **"Non-constraints-aware**". The replication techniques classified into this category are primitives but they form the basis for many other works.

3.1.2. Architecture-Based Classification

Another aspect that distinguishes existing replication techniques is the architecture of the network that they consider. We mean by the architecture, the way the nodes communicate with each other. Some techniques assume the existence of a central point in the network and all queries pass through this point; some others assume the possibility of dividing the nodes into groups based on specific criteria and all queries are handled inside the groups.

Indeed, the selected architecture has a significant impact on the efficiency of the technique and on the scenarios that can use it. Four types of architectures can be differentiated:

- ✓ **Centralized architecture**: a central node exists in the architecture, and all mobile nodes send their queries to this node. This architecture offers the possibility of

global synchronization; however, it suffers from failure problems since the entire system fails when the central node fails.

- ✓ **Centralized group-based architecture**: mobile nodes are arranged into different groups based on various parameters (e.g. location, node interests, etc…). A master node is selected for each group. All nodes belonging to a group access data through the master node. The master nodes of different groups may be able to communicate with each other to synchronize data. This architecture usually provides accurate decisions and avoids the communication between the nodes; but it imposes overhead to complete the synchronization with the central point.

- ✓ **Decentralized group-based architecture**: in this architecture also, the nodes are arranged into groups but without central node. The distribution process is done so that each node belongs to only one group and communicates only with the other nodes in its group.

- ✓ **Decentralized architecture**: a mobile node sends its requests to any other nodes that are within its communication range. The decisions are made in an autonomous way. This architecture is adapted to dynamic networks but it is the most difficult architecture to manage.

We have chosen to classify the existing replication techniques using the "constraints-based classification" as this classification leads to more specific categories and shows clearly the difference between the different propositions. However, the category of the technique in the "Architecture-Based Classification" is mentioned in the description of each work. Table 3-1 presented at the end of the section shows a summary of the presented works.

3.2. Replication Systems for MANETs

3.2.1. Non-Constraints-Aware Techniques

3.2.1.1. SAF, DAFN, DCG [34]

Takahiro Hara worked from 2001 to 2004 on replication algorithms to improve data accessibility in MANETs. In his first paper [34], Hara proposes three replication schemes for unmodified data. The allocation is periodically executed to cope with dynamically changing network topology. In this work, the access frequency to a data item is the number of times the mobile node accesses this data item. The access frequencies to data items are used mainly to distribute the replicas. These frequencies are assumed to be known in advance and not to change.

In the proposed replica placement methods, the distribution of one replica (data item) depends on all data items that exist in the network and on their access frequencies and not only on the replicated item. The three proposed methods are:

- Static Access Frequency (SAF): in this method, each node individually allocates replicas in descending order of their access frequencies. Only the access frequencies to data items are taken into account; therefore, mobile hosts with the same access characteristics allocate the same replicas. However, the next two methods consider other characteristics along with the access frequency to avoid the replica duplication.

- Dynamic Access Frequency and Neighborhood (DAFN): in this method, each node considers during the replica allocation both the access frequencies and the network topology. The first allocation is done as SAF. Then, if there is a replica duplication between two neighboring nodes (i.e. at one hop connection), the node with the lowest access frequency selects another replica. However, DAFN does not eliminate completely the duplication because it scans the network once to search neighbors while nodes move and the topology changes.

- Dynamic Connectivity based Grouping (DCG): it is executed to find bi-connected groups in the network (i.e. group of nodes that can connect to each other with different paths so that if an arbitrary node in the group is disconnected, the group is not partitioned). An access frequency of the group to each data item is calculated as the summation of the access frequencies of all the nodes in the group to the item. In each group, the cache spaces of all nodes are treated as a big single cache. Then, data replicas are allocated into this big cache according to their access frequencies.

SAF and DAFN apply a decentralized architecture where DCG applies a decentralized group-based architecture. Later, Hara improved these methods several times; each time, he considers in addition to the access frequency, one of the environment constraints in the replica allocation process. The improvements methods are explained in the corresponding category below.

Discussion: the three methods suffer from several weaknesses and are not actually adapted to mobile networks. They assume that existing data items and their access frequencies are known for all nodes and they do not change. We argue that this assumption is difficult to be maintained in MANETs. It is like having a centralized table containing needed information that each node consults to make its decision about which data item it should replicate. In addition, these methods do not consider any of the resources' constraints nor the network constraints. DCG might be the best method as it optimizes the network traffic and the allocation effort, but it is still a primitive method that can be applied only to scenarios where limited and small number of nodes participate and all nodes know each other in advance.

3.2.1.2. *Skip Copy "SC"* [82]

Skip Copy (SC) addresses decentralized MANETs. It assumes that each mobile node provides its location using a geographical positioning system (e.g. GPS) and that the probability of accessing data from a node is related to the distance of that node from the location where the data is generated. This assumption is used to justify the fact that replicas are distributed only around the node that creates the data item.

The replication process is started after each creation of a new data item. The owner of the new data propagates it to its neighbors using a propagation request. The propagation request includes in addition to the data item (1) the current location of the owner obtained by positioning systems (2) the time of generating the data, and (3) a parameter **C** initialized to 0 and incremented by 1 after each replica placement. The SC method uses a parameter **S** called "Skip Parameter" to ensure that replicas are not placed too close to each other. Thus, upon receiving the request, the node places the replica if (C mod S = 0) and if the distance between the host and the original data (at location P) is smaller than a preset parameter called the "replication range", it forwards the request after incrementing **C**.

After each data access request, the replica relocation process is started. The relocation maintains the condition that a minimum distance of S hops exists between two replicas. When a data request arrives to a node which has the data, it sends back a copy of the data item, with the variable C=0. Then, the replica propagation process with the same conditions is re-executed on each node that receives the request. In this work, the authors also define methods to propagate update replicas and to access data.

Discussion: the SC method ensures a good distribution of the replicas. It controls the propagation of the replicas using the Replication range parameter. The main weakness of this method is the necessity of a positioning system used to determine the location of the host that creates the data item and to control the replication range where the replicas are created. Without the location of the nodes, SC does not work. Moreover, the assumption that the data item is requested only from nodes near the location where the data is created limits the applications that can use the SC. Finally, none of the environment constraints is considered.

3.2.2. Resources-Limitations-Aware Techniques

3.2.2.1. EA, WEA, WEA-B [78]

This work targets fully decentralized architectures. It aims to balance mobile nodes' power consumption as much as possible and prolong their lifetimes. From the authors'

point of view, the main reason to consume power is answering data requests. Thus, to balance the power consumption, the solution covers two aspects: selecting the best path to get the answer (selecting the node among the nodes having the answer) and effective replica allocation so that data requests can be fairly distributed later among the nodes carrying the replicas). Below, we explain the proposed algorithms:

- Selecting the effective path:

If several nodes hold the requested data item, the requestor selects the node carrying the item based on three methods [79]: (1) Minimum Hop (MH) where the node selects the shortest path. If multiple minimal paths exist, it selects the path that has the maximum remaining battery power where the remaining battery power is defined as the minimum remaining battery for all nodes on the path (2) Maximum Battery (MB) where the node selects the path with maximum remaining battery. If multiple paths exist, it selects the shortest path. (3) Battery-Hop (B-H) where the shortest path is selected among paths whose remaining battery power is larger than a threshold.

- Effective replica allocation:

Each node replicates the data items accessed by itself and its nearby neighbors but are held by a small number of them. Thus, for each data item, the node calculates AF+SAF where AF is its access frequency and SAF is the sum of the access frequencies of its neighbors. Then it replicates the item that has the highest AF+SAF. This method is called **Expected Access (EA)**; it balances the power consumption as data with high access frequencies are replicated on a large number of nodes. The EA is extended with multiple versions:

(1) **Weighted EA (WEA)**: the node gives more weight to the items accessed by itself (i.e. to the AF part in the equation). WEA avoids that a node discards (de-allocate) data item frequently accessed by itself.

(2) **WEA-Battery (WEA-B)**: the node changes dynamically the weight of the SAF part based on its remaining battery to avoid replicating data items accessed by others when its power is low.

Discussion: These methods balance the power consumption among nodes because data items with high access frequencies are replicated on a large number of nodes. Furthermore, WEA-B prevents nodes with little remaining battery power from being too frequently accessed from other nodes and so exhausting their batteries in a short time. However, the replication is triggered after each new data request which is impose an overhead; the node uses its access frequencies and the access frequencies of its neighbors (that are assumed not to change) to decide which data item to replicate. Finally, this method leads to heavy replica duplication in the network because nodes in a small area replicate all their frequently accessed data while the data availability of other data item will not be increased.

3.2.2.2. CLEAR [56], [57]

CLEAR (**C**ontext and **L**ocation-based **E**fficient **A**llocation of **R**eplicas) is a location-dependent replica allocation scheme. It deploys a super-peer architecture to facilitate the replication process; it divides the network into regions where each region has one super peer and a group consists of nodes in that region. The super peer is a peer that does not move outside its region and has powerful resources (e.g. battery, and processing capacity). It is responsible for executing the whole replication process.

Each node sends periodically to its super peer the necessary information for the replication process; this information includes a log of the access operations of all data items, the available memory, and the load status of the node. The load status is calculated using the formula $\sum_{i=1}^{N_d}(\eta_{d_i}/s_{d_i})/\eta_i$ where N_d is the total number of data items on the node, $\eta_{d_i}$is the access frequency of d_i, s_{d_i} is the size of d_i and η_i is used to normalize the load with respect to the bandwidth (it is equal to (B_M/B_{min}) where B_M is the bandwidth of the node and B_{min} is the minimum bandwidth.

The super peer starts the replication process when the access frequency of data item(s) exceeds a threshold $\Psi = T_{Acc} / T_{num}$ where T_{Acc} is the sum of the access frequencies of all data items and T_{num} is the total number of data items. It selects the nodes where the data can be replicated depending on their load status, their available memory and the

number of time they accessed the replicated data. The super peer avoids selecting overloaded nodes or nodes with low memory.

CLEAR supposes also that each node has a schedule known by the super peer so that this later does not select a node to be a replica holder in case this node will soon leave the region or will disconnect.

Discussion: The centralized nature of the solution allows synchronized replication decisions between the nodes in the region, thus it avoids replica duplications and non-useful replica creation. However, it is a costly solution; it is based on one super node per region (that is fixed or does not move outside the region); all nodes in the region communicate their information (load status, memory, AF, etc…) periodically to the super node (very high network traffic), that it uses them to make the local decisions and to apply all operations (gathering information of the nodes in the region, distributing replicas, and processing requests).

3.2.2.3. EcoRep [59]

EcoRep proposes an economic model for dynamic replica allocation in mobile P2P networks. It deploys a centralized super peer architecture where groups are formed based on the nodes location. Each group is managed by a service provider that does not leave its region and that collects information to make the replication decisions.

EcoRep is considered as the first economic replication scheme; it evaluates each data item with a price (in terms of virtual currency) that reflects its importance for the network. The price ρ of a data item d depends on many factors: it increases when the number of nodes which accessed it increases, when the access frequency (η) increases, when the consistency is high (the original data item has consistency equal to 1 where the replicas has equal or smaller consistency), when the number of its existing replicas (N_R) decreases, and when the average response time τ required for accessing d decreases (the response time is equal to ($T_W + T_D + T_{delay}$) where T_W is the waiting time spent before sending d to requestor node, T_D is the download time for d and T_{delay} is the path delay). EcoRep quantifies the importance of d by the equation:

$$\rho = \sum_{i}^{N_{MH}} (\omega_i * \eta_i * \varsigma_i) / (N_R * \tau_i)$$

Where: N_{MH} is the total number of the mobile nodes, ω_i is the weight coefficient for node M_i and it is equal to i/N_{MH}, η_i is the number of access requests for d originating from M_i, ς_i is the average consistency with which M_i answered d, N_R is the number of existing replicas of d, and finally τ_i is the response time from M_i.

A node requesting a data item should pay its price to the owner. Hence, a mobile node has to provide services to the network to earn enough currency to be able to issue its own queries. These services include answering queries, hosting replicas, routing queries to other nodes (replication queries or data queries). The revenue of a node M is the difference between the amount that it earns and the amount that it spends. In this way, EcoRep encourages nodes to store replicas so that they can increase their revenues, and thus get better services.

EcoRep is actually based on the CLEAR method presented above [56]; it uses in addition to the criteria used by CLEAR (the load status of the nodes and the available energy) the revenues of nodes to select the suitable replica holders. To select the candidate data item(s), it considers in addition to the access frequency, the origin of the queries to determine their relative importance for the network as a whole. The solution's originality is that it discourages the free-riding and provides incentive for users to become service-providers.

Discussion: EcoRep is a novel replication model; it is the first work that proposed an economical model to avoid the problem of free riding. On the other hand, regarding the replication decisions, EcoRep proposes a centralized solution with a client/server architecture; this architecture does not suite MANETs and it limits the scenarios where the solution can be used. In addition, this centralized nature imposes a considerable overhead in the network. Servers are responsible for collecting the information from all other nodes in this group; this allows making more accurate decisions (as decisions are based on a complete view of the vicinity) but at a high communication cost.

3.2.3. Network-Limitations-Aware Techniques

3.2.3.1. DAFN-S1, DAFN-S2, DCG-S1[35]

In [34], Hara proposed three replication methods SAF, DAFN and DCG (discussed in section 3.2.1.1). SAF replicates most accessed data items, where DAFN and DCG eliminate the replica duplication created by SAF if it exists. Further, in this work, Hara et al. enhances DAFN and DCG to consider the stability of the radio connection between the nodes. When the connection is not stable, the replicated data cannot be shared once they are disconnected; thus, DAFN-S1, DAFN-S2 and DCG-S1 eliminate duplicated replicas only when the connection between the nodes is stable.

DAFN-S1 and DAFN-S2 are a variation of DAFN. In both methods, each node M_i calculates the stability of the associated link B_{ij} with each neighbor M_j. DAFN-S1 eliminates redundant replicas if the connection stability is greater than a threshold value. DAFN-S2 uses a different strategy. When a redundancy is detected, the node calculates using the access frequency p, the probability that it accesses the same data after disconnection $(1-B_{ij})*p$. The replica is eliminated on the current node, if its probability is less than a threshold value.

DCG-S1 adopts the algorithm of DCG in eliminating the replica duplication. However, it forms bi-connected groups of nodes where it considers two nodes connected when the stability of their link is more than a threshold.

Discussion: The three methods consider the topology changes and the network partitioning. The replication is done after each period of time considering with fixed access frequencies. The main weakness of the three methods is the calculation of the radio link stability; it assumes a prior knowledge of the speed and direction of the movement of all mobile nodes which is not realistic in dynamic environments. DCG-S1 is an expensive method as it causes high traffic and needs a long execution time to allocate the replicas. Finally, the relocation period is constant; thus if the relocation period is too long, the methods cannot adapt to the topology changes, and if it is too short, the traffic increases and unnecessary replicas are diffused. So tuning this period is crucial but complex.

3.2.3.2. REDMAN[16], [15], [14], [13]

REDMAN is the abbreviation of **RE**plication in **D**ense **MAN**ets. It addresses centralized group based architectures. It is a solution to disseminate, manage and retrieve replicas among cooperative nodes in a dense MANET. Actually, "dense MANET" indicates that each node is connected to n neighbors and this number remains constant for a long period of time. Considering only dense MANETs excludes the possibility of network partitioning.

This work proposes a centralized solution, it based on the election of a replica manager for each region. This replica manager is responsible for deciding the number of replicas (called degree) needed of each resource (data item) and coordinating the replica degree maintaining process. To this end, it receives from all nodes a descriptor containing the shared resources. The replica degree is determined based on the criticality of the resources included in the meta-data (sent by the owner) and external factors like the number of the nodes in the region. Two protocols are proposed in addition to the main tasks of the replication: (1) Participant Identification: to verify periodically the validity of the density conditions and (2) Replica Manager Election: to elect periodically the replica managers.

REDMAN middleware implements three tasks to manage replicas: (1) the replica distribution process [15], (2) the replicas access [13], and (3) the replica degree maintenance.

The replica distribution process starts from the owner after receiving a notification from the replica manager. It forwards replication requests to randomly chosen neighbors. If the receiver is K-hop far from the owner (where K is a predefined constant), and if it accepts to host a copy it notifies the manager; otherwise it forwards the request to another neighbor.

The manager applies a protocol to maintain the replica degree without guaranteeing an absolute consistency. Each owner replicates its data on the manager (or a neighbor if a connection with the manager is not possible) before leaving the dense network.

Later the manager elects a node to be the new owner for the resource that in its turn executes the replica distribution process.

Actually, the authors propose also propagating information about available replicas to facilitate later the replica access and decrease the load imposed by this process. The mechanism details are omitted here because it is not related directly to the replication.

Discussion: the processes proposed by REDMAN are trivial. First, the replica distribution process is done by selecting nodes from the direct neighbors to hold the replicas without considering any other factors (power constraint, user interest, etc…). Furthermore, the manager election process is expensive. The main role of the manager is to hold information about the criticality of the resources to compute the replica degree which the system can be tolerant with its consistency (e.g. the system can creates less replicas). However, in all their articles, the mechanism of determining the required number of replicas that must be maintained in not explained. Finally, the solution requires a degree of network stability which might be stringent in the context of a mobile network.

3.2.3.3. DRAM[42], [41]

Indeed, if the network partitioning produces multiple partitions such that the replicas of the same data item remain in the same partition, the replication will not enhance the data availability. Thus, DRAM uses the group mobility to implement successful replica placement. “Group mobility” refers to the movements of nodes together; thus DRAM proposes a decentralized group based algorithm to cluster nodes with similar motion into disjoint mobility groups called allocation unit. Nodes in one group share their storage capacities and do not store the same replicas.

The replication in DRAM is done periodically. Each mobile node performs two phases during each reallocation period. The unit construction phase and the replica allocation phase. Allocation units are formed in the first phase according to the reference point group mobility model (RPGM) [39]. Each group has a master node responsible for clustering nodes having the same motion behavior, and removing

nodes that do not send status messages. During the replica allocation phase, data items are allocated to different mobile nodes based on their access frequencies and on the allocation units that were formed in the previous phase. Each unit stores replica for data items that have the highest number of accesses from the nodes in the unit. The replica is placed on the node of the group that has the maximum available storage.

Discussion: DRAM maintains mobility groups in an adaptive manner which keeps the number of broadcasted information as small as possible. The solution assumes the similarity of the nodes movements (obtained by a GPS which is not always available on mobile devices). However, if the nodes move in random patterns, the formed group will be small in size and the operation of joining and leaving an allocation unit will be frequently repeated and thus impose an overhead on the system.

3.2.4. Data-Aware Techniques

3.2.4.1. CADRE [58]

CADRE has been done to complete CLEAR [56]. It is the acronym of Collaborative replica Allocation and Deallocation of Replicas with Efficiency. CADRE deploys the same super peer architecture as CLEAR. In CADRE, the supper peer is called gateway node (GN). Those gateways are responsible for both the replication and the search operations.

The replication is performed based on the nodes characteristics (e.g. load and energy) as well as the nodes schedules. GNs have all necessary information due to periodical messages sent by all nodes. The added values of CADRE are:

- It considers large-size files (images). It proposes keeping replicas based on the available memory size of neighbor nodes. To this end, when a node requests an image it includes its available memory in the request. The node which receives the request, keeps a log of the neighbor's available memory and therefore, based on this log it decides the replica size that it should keep (it corresponds to the maximum number of queries requesting that size). Note that

the solution considers 4 ranges of granularity of the replica size (low, medium, high, original); it uses a compression algorithm to obtain four version of the image with different size.

- It performs the replicas deallocation in a collaborative way. If each node individually decides which replica will be deallocated, it might delete a hot data item that will be requested again and thus re-replicated later. This problem is known by 'thrashing'. Hence, gateway nodes decide based on the regional information which replica should be deallocated when necessary.

- It addresses the issue of a fair replica allocation across mobile nodes. To this end, CADRE considers in addition to the access frequency of a data item DI, the origin of the requests. So the item will not be considered as 'hot' if only one node issues a large number of queries asking for it. To ensure fairness, each node assigns a score to each data item. This score quantifies the importance of DI for the network as a whole. It increases as the data item is demanded from more nodes; hence, given two data items with equal access frequencies, the score of the data item that is demanded from a larger number of nodes will be higher. Then, CADRE replicates the data item with the highest score.

Discussion: CADRE is an original replication model. It addresses aspects that are rarely addressed by other models: the size of the data items (images), the thrashing problem and the replication fairness. However as for CLEAR, it is a centralized model, all the decisions are made by the super peer which makes the cost of the replication system very high (all nodes send their access log, load status, available memory, etc..) to the super node that makes all the operations. Finally, the client/server architecture does not suite the mobile networks.

3.2.4.2. CSAF, CDAFN, CDNG [36]

SAF, DAFN and DCG presented in section 3.2.1.1 are extended again in this work to consider the correlation among data items. The correlation of two data items is the

probability that a user access both data items at the same time. The authors assume that this measure is available for each pair of data items stored by a node and does not change. In order to choose the data item to replicate, the priority of each data item is calculated based on the correlation measure; finally the replication is done periodically by descending priorities. The priority of a data item DI_j is calculated on node M_i as follows:

- The access frequency is calculated as the summation of the correlation of DI_j with all the other data items ($\sum_{k=1}^{n} p_{i-jk}$ where p_{i-jk} represents the strength of the correlation between D_j and D_k on node M_i). Among the possible combination of X data items with the highest access frequency, two data items with the strongest correlation are chosen.

- Among non-selected data, the data items are arranged based on the sum of the correlations of each data with the original data of node M_i or the data items already selected in step one.

- Data items are replicated based on their priority: the data items created by node M_i have the highest priority, then, the data selected in the first step, and finally the remaining data sorted in the second step.

Based on this new calculated priority and access frequency, the three methods SAF, DAFN and DCG become:

- Correlated SAF (CSAF): each node replicates data items by descending priority.

- Correlated Dynamic Access frequency and Neighborhood (CDAFN): when two neighbors have the same replica, the node with lower access frequency changes this replica by another one.

- Correlated Dynamic Connectivity based Grouping (CDCG): according to the priorities in the group, data items are replicated until the memory space of all nodes in the group becomes full.

Discussion: The novel idea is to consider the correlation between data items. But the correlation is determined in a static way and it does not base on the semantic of the documents. Furthermore, this solution suffers from multiple limitations. As for SAF, DAFN and DCG presented above, it does not consider the resources availability; all nodes must know all data items and their access frequencies that do not change. Finally, these techniques yield the best results only when most of the data are accessed in correlation with each other which is not always true.

3.2.5. Resources-Network-Limitations-Aware Techniques

3.2.5.1. Data Lookup and Replication scheme [23]

This work addresses decentralized group based architectures. It proposes distributed data lookup and predictive data replication algorithms. First, the data lookup schema is a local table that improves the data searching. It is stored on each node and it contains the data items stored on all group nodes. This table is derived from periodic advertising messages broadcasted by each node to the members of the groups. It helps in filling the local table to let the node looking for data and eliminating the redundant data items in the groups. The advertising message contains a sequence number, the sender's address, the resources of the nodes (CPU, storage and energy) and the data items available at the sender. The authors propose a protocol to exchange these messages and to decrease the overhead imposed by them. Each node which receives an advertising message updates its lookup table, deletes redundant replicas and deletes old versions of data using the timestamp associated to each data item.

In addition, the authors propose a proactive replication algorithm to remedy to the problem of decreased data availability after network partitioning. They assume that each node follows a predictable movement pattern that allows in predicting its future locations. Using this prediction and the wireless transmission range, the group connectivity and the network partitioning can be predicted. Once the occurrence of a network partition is predicted, data on those nodes that may be rendered inaccessible due to network partitioning are replicated so that the resulting partitions will still be able to access data belonging to nodes in other partitions. The selection of the

candidate node for the replica is based on the node capacities. In case of hosting space limitation, the most frequently accessed data replaces the least frequently accessed ones.

Discussion:

This approach is similar to that of DCG [34], except that the latter defines bi-connected groups of servers for the purpose of data replication. In addition, data access in DCG is not group-based. The proactive approach in advertising data uses broadcasting messages that contains all data items available on the nodes. Furthermore, the assumption that the movement patterns are known is not realistic. This approach is suitable for small networks (as the scenario presented in the article: battle-field communication) but it is not scalable. However, the proposition considers the network and the resources limitations which is a positive point.

3.2.5.2. DREAM [66]

DREAM is a replication technique for real time mobile databases. It is composed of three main parts: replica placement, replica synchronization in case of data update, and finally data access transaction.

DREAM addresses the problems of power limitation and communication strength. It classifies nodes into two categories: nodes with limited storage size, limited battery, and limited communication capacity are “clients” while more powerful nodes are “servers”. Clients store only the query processing module necessary to submit transactions (in term of databases) and receive results. Servers store complete DBMS and provide data to clients.

In DREAM, the replication insures the availability of data items needed to complete the transactions. The replication considers the power limitation by periodically replicating the data items with high access frequencies on servers with the maximum remaining power. The access frequency in DREAM is equal to the number of times a data item is accessed. It is weighted using the type of the transaction that accesses most the data item: contrarily to soft transactions, firm transactions are aborted if they

miss their deadlines; thus the access frequency of data items accessed by firm transactions are given high weight to increase the availability of these data and therefore reduce the number of aborted firm transactions.

If replica redundancy exists among neighbors, they are eliminated depending on the stability of the links connecting two servers. When the reliability ratio between two servers is high, duplicate replicas are eliminated otherwise DREAM keeps the duplications to ensure the availability of the data in case of network partitioning. Finally, DREAM addresses the replica synchronization issue by maintaining two timestamps that indicate when a particular data item is updated.

Discussion: DREAM considers both the power limitations and the network limitation which is an advantage of the system. It can be considered as a data-aware technique as it selects the data item for replication based on the type of the transaction; but we do not present it in the data-aware category since it concerns only database systems. A strong limitation of this approach is that it implements a client server architecture which is not well adapted to MANETs. In addition, it is dedicated to databases systems and communication costs are not considered. Finally, all servers must be aware of all available data on other servers and of their access frequencies which is not adapted to most of MANET applications.

3.2.5.3. REALM[85]

The main goal of REALM (Replication of data for A Logical group based MANET database) is to increase the percentage of successful transactions in a centralized group based architecture. It attempts to reduce the power consumed by the nodes (by placing replica near them), and to balance the power consumed by servers in the network (by distributing replicas). REALM groups nodes with respect to the data items they will need to access. Group membership of nodes helps in identifying the data items that any node in the network will need to access, as well as to identify the most frequently accessed data items on every server.

Replication is triggered on a server when one of the following conditions comes true: the percentage of successfully executed transactions falls below a preset percentage, (2) the occurrence of network partitioning is predicted (3) the transactions executed on the server access data on other servers, (4) finally, the energy level of the server falls below the average energy level of the other servers.

REALM replicates data (only on servers) based on their access frequencies and on the interests of the nodes within two hops from the server. To this end, each server (1) arranges its data items in descending order of their access frequencies (calculated only using accesses from one and two hops neighbors), (2) adds data items which interest the nodes (the interest is predefined according to the group membership) to the ordered list (3) eliminates the data items available on a server within the communication range. (4) finally, replicates the remaining data items until it runs out of storage space.

Discussion: REALM uses a client/server architecture which is unsuitable for most MANET applications. More powerful servers carry all replicas. They have information about the interests of each group and the access frequencies to all data items, which makes the solution not adequate for MANETs. In addition the main objective of REALM is to use the application semantics to distribute replicas. However, nodes with the same interest are divided into groups that are known and do not change and their interests do not change also. There is no consideration for new coming nodes.

Table 3-1 : Replication systems for MANETs

	Constraint-based Classification			Architecture Classification	Other characteristics	Target scenario	Technique	Validation	Weakness & Complexity
	Res	Net	data						
SAF, DAFN, DCG [34]	-	-	-	Dec. G.		Disaster site, users form a team work	Replicate data with high AF	Simulation	AF do not change, any node knows all other nodes, their data and their AF
SC [82]	-	-	-	F.Dec.	Location dependent	Emergency services	Replication after DI creation. The distance between 2 replicas is equal to a parameter	Simulation	Costly: based on flooding replication requests up to a specific range
EA, WEA, WEA-B [78]	X	-	-	F. dec.	Effective Data access	Rescue effort at a disaster site where energy infrastructure has broken down	Replicate highly accessed data by the node and its neighbors to decrease the path of the data requests	Simulation	Costly message exchange and Replica duplication
CLEAR [56, 57]	X	-	-	Cen. G.	Location dependent	Slow moving nodes in shopping mall, tourists in a bus or in museum	Super node collects load status, memory, AF. Replica is placed on non-overloaded nodes in the region	Simulation	High communication cost. Client server architecture
EcoRep[59]	X	-	-	Cen. G.	Economic model, free riding, Location dependent	Slow moving nodes in shopping mall, tourists in a bus or in museum	Each node participates by carrying replica to get the price of DI that it requests.	Simulation	Client/server architecture Communication cost

Method name	Constraint-based Classification			Architecture Classification	Other characteristics	Target scenario	Technique	Validation	Weakness & Complexity
	Resources	Network	Data						
DAFN-S1, DAFN-S2, DCG-S1 [35]	-	x	-	Cen. G.		Disaster management; users form a team work	Replicas are placed on nodes that are likely to remain connected for a while	Simulation	AFs are known and do not change computation of the radio link
REDMAN [13, 14, 15, 16]	-	x	-	Cen. G	Dense MANETs	Commercial centers	Each node asks for n replicas where n is determined by a central replication manager	Simulation NS2	Central point Costly manager election
DRAM [41, 42]	-	x	-	Dec. G.	Group mobility	Scenario with groups (museum, military groups)	Forming mobility unit that share their storage and replicate data w.r.t. AF of all nodes in the unit	Simulation	Assume the similarity in nodes' movements (large number of groups impose communication overhead)
CADRE [58]	x	-	x	Cen. G.	Thrashing & Replica fairness & Image size	Slow moving nodes in shopping mall, tourists in a bus or in museum	Replication based on the load status, memory, AF and the location	Simulation	High communication cost. Client server architecture
CSAF, CDAFN, CDNG [36]	-	-	x	Dec. G.		Rescue workers in disaster area, Members of research project	Place replicas for 2 data that are correlated near each other	Simulation	Correlation is based only on the access frequency

Method name	Constraint-based Classification			Architecture Classification	Other characteristics	Target scenario	Technique	Validation	Weakness & complexity
	Resource	Network	Data						
Replication scheme [23]	x	x	-	Dec. G.	Group mobility	Battlefield	Proactive replication before predicted network partitioning. Advertising data availability to implement local information lookup	Simulation	Broadcast advertising messages (scalability problem)
DREAM [66]	x	x	-	F. Dec.		Address traditional databases	Servers replicate hot data on nodes with max remaining power	Prototype	Client/server Communication cost is not considered
REALM [85]	x	x	-	Cen. G.	Application semantics	Group of nodes that uses the same application (have the interests)	Replicate interesting and requested data on server within two hops.	Prototype	Client/Server Static nodes interests

Cen. G: Centralized group-based Dec. G: Decentralized group-based F. Dec.: Fully decentralized

3.3. Features Comparison

In this section, we present the features used to compare the existing replication techniques presented in this chapter. We divide these features into two categories: the first category (Table 3-2) contains general features that a replication system considers; e.g. the size of replicated data, the architecture of the MANET, etc. The second category (Table 3-3) includes the features that determine how the replication decisions are made e.g., which data item is selected for the replication process, which factors are considered to place the replicas.

3.3.1. The General Features of the Replication System:

- MANET Architecture that it addresses. Note that the existing works are classified in tables 3-2 and 3-3 with respect to the MANETs architecture which facilitates the comparison between techniques that targets the same architecture.

- Client-server classification: does the system distinguish nodes with respect to specific criteria (e.g. functionalities, resources) where some nodes are considered as servers and other are only clients?

- Density: does the system consider the dense networks? it can be scale to dense MANETs?

- Replication level: full or partial replication. Usually, replication systems apply partial replication to adapt to the storage, and the battery constraints.

- Data size consideration: data size influences significantly the replication model. Thus, does the system consider only numerical and small data size, or is it adapted to replicate large size data like multimedia files?

- Routing protocol dependency: some replication solutions are adapted to work along with a specific routing protocol or with a specific category of routing protocols.

- Use of a positioning method (GPS): does the replication system assume a complete knowledge of the nodes positions? In other words, does the replication system use a positioning service like the GPS?

- Communication cost consideration: does the technique consider the communication cost which should be as minimal as possible (distance between the communicating nodes, number and size of exchanged messages to implement the replication process, etc…).

3.3.2. Parameters for the Replication Decisions:

- Partition awareness and radio link stability: does the system consider the network partitioning and the radio link stability in making the decisions (e.g. triggering the replication, selecting the best node to place the replica, selecting the data to replicate in order to avoid data loses).

- Resource awareness: does the replication solution consider the resources that are available on the mobile nodes (i.e. the battery, the storage space, the CPU)?

- Access frequency: does the replication system consider the number of times that a data item was accessed by mobile hosts to select the data item for replication?

- Relocation period: how often the replication process is executed? Is the replication executed periodically or based on specific events (i.e. data creation, network partitioning)?

- Semantic decisions: does the semantics of the data play a role in selecting the replicas? Do the profiles of connected nodes influence the replication process?

- User satisfaction: does the replication process consider the user activities in making the replication decisions in order to keep sufficient resources for the user?

- Location dependency: does the location of the nodes, affect the replication process?

Table 3-2: General features comparison

	MANET architecture	Client-Server Classification	Density	Replication Level	Data size consideration	Routing protocol dependency	GPS utilization	Communication cost consideration
CLEAR [56]	Cen. G.	Yes	No	Partial	No	No	No	No
CADRE [58]	Cen. G.	Yes	No	Partial	Yes	No	No	No
REDMAN [13,14,15,16]	Cen. G	Yes	Yes	Full	No	No	No	No
REALM [85]	Cen. G.	Yes	No	Partial	No	Yes (OLSR)	No	No
EcoRep[59]	Cen. G.	Yes	No	Partial	No	No	No	No
SAF, DAFN, DCG [34]	Dec. G.	No	No	Partial	No	No	No	No
DAFN-S1, DAFN-S2, DCG-S1 [35]	Dec. G	No	No	Partial	No	No	No	No
DRAM [41, 42]	Dec. G.	No	No	Partial	No	No	YES	Yes
CSAF, CDAFN, CDNG [36]	Dec. G.	No	No	Partial	No	No	No	No
Rep scheme [23]	Dec. G.	No	No	Partial	No	Location based RP	Yes	No
EA, WEA, WEA-B [78]	F. Dec.	No	no	Partial	No	Based on the path	No	No
DREAM [66]	F. Dec.	Yes	No	Full	No	No	No	No
SC [82]	F. Dec.	No	No	Partial	No	No	Yes	No
CReaM	***F. Dec.***	***No***	***Yes***	***Partial***	***No***	***No***	***No***	***Yes***

Cen. G: Centralized group-based Dec. G: Decentralized group-based F. Dec.: Fully decentralized

Table 3-3: Parameters for the replication decisions

	Partition awareness	**Power awareness**	**(CPU, storage)**	**Access frequency**	**Relocation period**	**Semantic Decision**	**User Satisfaction**	**Location Dependency**
CLEAR [56]	No	Yes	Yes	Yes	No (AF exceeds threshold)	No	No	Yes
CADRE [58]	No	Yes	Yes	Yes	No (AF exceeds threshold)	No	No	Yes
REDMAN [13, 14, 15, 16]	No	No	No	No (all shared resources)	No (on node connection or disconnection)	No	No	Yes
REALM [85]	Yes	Yes	No	Yes	No (AF, Power, Disconnection)	Yes (G. based on application	No	No
EcoRep [59]	No	Yes	Yes	Yes	No	No	No	Yes
SAF, DAFN, DCG [34]	No	No	No	Fix do not change	Yes	No	No	No
DAFN-S1/S2, DCG-S1 [35]	Yes	No	No	Fix do not change	Yes	No	No	No
DRAM [41, 42]	Yes	No	NO	Yes	Yes	No	No	No
CSAF, CDAFN, CDNG [36]	No	No	No	Yes	Yes	YES (AF based)	No	No
Replication scheme [23]	Yes	Yes	Yes	Yes	No (before network partitioning)	No	No	No
EA, WEA, WEA-B [78]	No	Yes	No	Fixed do not change	No (After each data request)	No	No	No
DREAM [66]	Yes	Yes	Yes	Yes	Yes	No	No	No
SC [82]	No	No	No	No (all shared resources)	No (after data creation & data requests)	No	No	Yes
CReaM	***No***	***Yes***	***Yes***	***Yes***	***No (resource based)***	***Yes***	***Yes***	***No***

3.4. Positioning of the Thesis and Contributions

As we can conclude from the survey, the replication in MANETs has been widely studied and it has taken the attention of a lot of researchers. However, some important points still not handled as explained below. Our model targets **fully decentralized** architectures; it considers the resources availability; and it distributes replicas considering the user interests and the semantic of the data item content. To position our work with respect to the existing techniques, we compare the methods that are proposed to answer the three main questions concerning the replication scheme: (1) when the replication is triggered, (2) which data item(s) are selected for replication and (3) where the replica is placed.

3.4.1. Triggering Replication

Due to the node mobility, one single replica distribution process does not increase data availability. Thus, replication technique must integrate a way to trigger the replication process multiple times to adapt the replica distribution according to the topology changes. In the literature, two ways can be distinguished, the periodical replication and the reactive replication.

The periodical replication is much more common. However, periodical techniques have a basic drawback. If the relocation period is set to a long time, the data availability may decrease because network partitioning occurs and the replication at that time becomes ineffective. Oppositely, if the time period is short, unnecessary overhead is generated to implement the replication. For example, in DRAM [42] (see section 3.2.3.3), for each replication period the algorithm constructs the allocation clusters with nodes having the same motion behavior. Other example is [23] techniques that predict the network partitioning to place the replicas; if the period is short partition prediction algorithms are executed despite the topology remains unchanged.

The reactive replication is adaptive to a specific action(s) (e.g., topology changes, data access patterns, data creation). It is more effective than periodic replication. In the

literature, CADRE [58], CLEAR [56], and EcoRep [59] trigger the replication when the number of data access for a data item exceeds a threshold. In SC [82] replication is related to the creation of new data item in order to propagate it in the network.

Our proposed replication method is a reactive method based on the resources consumption. The idea behind our proposal is that when a node participates to the replication system, it carries replicas and answers data requests concerning them. Consequently, resources necessary to accomplish user's personal tasks are consumed. Hence, a node can only participate in the applications deployed on the network (including the replication system) within the limit of its possibilities so that the nodes with scarce resources participate less than other more powerful nodes.

We implement replication decisions considering the availability of three types of resources (the battery, the storage space and the CPU). The replication is triggered when a high level of resource consumption is detected, with the goal of sharing with other nodes the load of data requests processing.

3.4.2. Selecting data item

In the literature, the decision of selecting data item for replication is based principally on the access frequency. In [34], [78], [35], [59] and [41], the access frequency for a data item is defined as the number of times the node accesses the data item Thus, each node calculates its own access frequencies and replicates locally the data item with the highest access frequency. Actually, using the access frequency as the only factor to select replicas means that the data item that was highly requested before will be replicated again. In addition, the placement of this data item is done on a node that has already requested it. We argue that these solutions head toward increasing the data availability of data items already propagated in the network. However, in real scenarios, a node that requested a data item, will usually not requested again.

Other solutions use additional criteria along with the access frequency. For instance, in REALM [85] a server replicates the data items that interest the nodes at two hops. The interests of the nodes are assumed not to change and are known in advance by the

server. In [36], the data items are selected with respect to a priority. The priority is calculated using the correlation strength between data items; this correlation is assumed not to change and also known in advance by all the nodes.

The model proposed in this thesis considers for the data item selection process two criteria: the data accesses and the semantics of the content. The data access for a data item on a node is defined in our model as the total number of requests received by the node to the data item. The semantic of the content and the previous data accesses are used to predict future requests.

3.4.3. Replica Placement

The replica placement is usually the process in which the constraints are mainly considered. In the previous works, many criteria have been considered to select the best node to place the replicas. The basic factor is the access frequency. This is the trivial solution, where the data item is replicated on the node that has the highest access frequency (i.e. the node that has the most highly requested the data item). As explained above, for the data selection process, this solution does not suite most types of applications since the node that has requested the data will usually not request it again.

The second factor is the network partitioning. Since the partitioning is the main factor that affects the data availability, most of the replication techniques have focused on obtaining a complete image of the network status and node mobility to predict the network partitioning behavior. In general all of these techniques have a main feature, they query the lower network layer e.g. the routing components to obtain the needed information. Besides being expensive and time consuming these querying processes make the proposed protocols and algorithms highly architecture-dependent.

The third factor is the resources availability. In general the battery is the resource that is considered for the replication placement. Trivial solution as in DREAM [66] places replicas on nodes with the maximum remaining power. In [78] the replication technique places replicas on nodes that have high access frequencies for the replica to

decrease the access cost and consequently decrease the power consumption. However, the same problem of access frequency stands here (i.e. the node replicates the data items that highly accessed by itself). In [56] and [58] the load status of the node is studied in order to avoid placing replicas on overloaded and scare-power nodes. However, both solutions are centralized solutions; they use servers to manage the replication and to hold all information concerning the load status of the nodes.

The last factor is the nodes' interests. Only few works have addressed the semantic information and the nodes interests to distribute replicas. In REALM [85], the replicas are distributed on server close to interested nodes. However, the nodes interests are known by servers and do not change which is a strong limitation.

In this thesis, we propose a community centric replica distribution. Nodes interested in the same topic form a community of interests. Interests are defined dynamically using the nodes profile. We propose a lightweight protocol to collect information about the community distribution as it varies with time according to the network evolving topology. Then replica is placed near the community that is interested in the content of the replica. In addition, the replica placement in our model considers the resources availability so that replicas are not placed on overloaded nodes.

3.5. Conclusion

In this chapter, we have presented a survey of the main replication techniques proposed in the literature. We have used two classifications: first, a classification based on the operating constraints and second, a classification based on the architecture of the network. We have discussed the replication techniques with respect to the first classification, as it shows the big differences between the existing works. Each technique has been followed by a discussion in order to show its advantages and its drawbacks. Furthermore, the main features of existing replication techniques have been extracted to compare the different techniques. Finally, a comparison study with respect to the three main process involved in the replication of data has been done in order to position the work proposed in this thesis.

II

COMMUNITY-CENTRIC AND RESOURCE-AWARE REPLICATION APPROACH

In this part of the thesis, we present our main contribution, the **C**ommunity-centric and **R**esource-aware R**e**plic**a**tion model. This model is a decentralized model adapted to mobile environments. It uses the replication to pursue two goals simultaneously: increase the data availability and preserve the device resources.

CReaM is divided into three main processes: replication initiation, data selection process and replica placement process (Figure II-1): The replication initiation process is based on resource monitoring. In the remainder of the thesis, we call "**user satisfaction**" the fact of maintaining available resources on the user devices at a reasonable level so that the user activity is not too significantly impacted by the replication system. The data selection process and the replica placement process base on the semantic content of data items and on the interests of the nodes. Thus, we assume that data items are tagged by a list of topics (so-called 'subjects') that determine the semantic content. Users interested in the same subject form a community of interest. In the following, each of the three processes is explained separately:

Figure II-1: The replication processes

1. Replication Initiation Process

To preserve the device resources for the user, we developed a monitoring engine that observes the availability of three types of resources on the user's devices: CPU, storage space and battery on each node. When the consumption of at least one resource becomes high and exceeds a predefined threshold, CReaM triggers suitable actions to remedy to excessive resource consumption. Several types of actions are considered depending on the cause for the intervention. In the general case, the action consists in replicating; in more critical cases, other actions may also be taken: stop responding to access requests, stop placing replicas, etc. In this context, replication is

initiated to balance the load caused by requests over several nodes of the network. Collaterally, the availability of the data item increases, and the resources of the original data item holder are less taxed as its load diminishes.

To make the model dynamic, the thresholds are adapted depending on the current user activity on the node.

2. Data selection Process

In case the replication is the suitable action to remedy the resource problem, the best data item to replicate must be selected. Replicating a randomly chooses data item would probably not lead to achieve load balancing nor to increase the availability of data that interests the nodes.

CReaM selects the data item depending on the type of resource that triggered the replication process. We distinguish the following cases:

- In case of CPU or/and battery consumption: the best data item to replicate is the one that can better alleviate the load of the node. The adversaries replication systems, select the highly requested data item. We argue that this data item is widely spread on the network and there is no gain expected by replicating it. Indeed, our model relies in making decision on the semantics of the content. It analyses incoming data requests to predict hot subjects i.e. interesting subjects for connected nodes. Hence, the best data item to replicate is a data item tagged by a hot subject and that becomes requested.

- In case of low battery, the probability of a disconnection in the immediate future becomes high. The risk is that "interesting" data items that are only available on this loaded node disappear from the network. Hence, the best data item to replicate is a rare data item; i.e. a data item tagged by a hot subject but that has not been disseminated yet to other nodes.

Furthermore, when the consumption problem strikes more than one resource simultaneously, the decision becomes more complicated. To this end, we define the possible node statuses with respect to the resources situation, and we introduce a data

item classification based on multiple criteria e.g., data rarity, number of requests, semantics of the content. Then, a rule-based inference engine is used to determine the node status and select the corresponding category from which to select a data item for replication.

3. Replica Placement Process

A very important point with respect to replication is the choice of the replica holder (i.e., the node that will receive a data item replica), as it strongly affects the load balancing obtained from the creation of the replica. The optimal allocation cannot be realistically determined in a very dynamic environment with varying topology. Therefore, CReaM integrates a prediction engine that based on the information at its disposal, estimates which location should provide the best load balancing and the best data usability. In the same time, the selected replica holder must have sufficient resources to place the replica and to assume the load resulting from the placement.

We propose a lightweight solution to collect information about the interests of participating users. The idea is to place the replica close to users interested in the content of the replicas. In addition to the load balancing, this approach decreases the cost of data requests because nodes can find the answer directly near to them. This solution is summarized by the following steps:

- Each user defines its profile; it selects from a predefined list a limited number of interesting subjects. This profile is disseminated using the routing protocol messages to one-hop neighbors only, in a way that does not impose additional overhead. Users interested in the same subject form a "community of interest".

- Each node collects the incoming profiles to calculate its potential to be in the center of a community of interest e.g., it has a high connectivity with the members of the community. We called "$P_S(N)$" the potential of node N for a subject S; it represents the number of neighbors of N interested in the subject S.

- In case a node finds a community of interest around it (i.e., a high potential value for a subject S), it diffuses its potential value, $P_S(N)$, to neighbors to be used later when a replica holder must be selected.

During the replica placement process, the candidate data item for replication is assigned to one or more community of interest through a tags analysis. Then, the 'best' replica holder is identified as the node with the highest connectivity with the members of the community (e.g. the node with the highest potential value for the replica subject(s)).

Then, the node prepares a replication request and sends it to the selected replica holder. Actually, the node delegates the replication to the replica holder; thus, if this later cannot place the replica because of a resource consumption problem, or because it already has the data item, it corrects the placement by sending the replication request to another node.

Autonomous operation of CReaM

In fact, CReaM is designed to cause as little additional network traffic as possible. It works in autonomous mode. Contrarily to other replication systems, replication decisions are made at the node level without global synchronization. This property is maintained during the three processes: For the replication initiation, each node replicates data when it detects a necessity for replication based on its own resources consumption levels. The 'best' data item(s) to replicate is selected based on the hot subjects that are detected by analyzing the data requests received by the node, without any periodical information exchange. The best replica holder is selected based on the information available on the node about the potential values.

This approach avoids a costly periodical exchange of negotiation messages mandatory to have a complete view of the neighborhood and therefore to make a decentralized replication decisions. Actually, this concept forms the basis of our contribution.

Structure of part II

CReaM is detailed in the next three chapters. Chapter 4 describes the first two processes; it explains the resources monitoring and the behavior of CReaM to cope with issues related to resources preservation. Chapter 5 is dedicated to the management of the communities of interests and the replica placement process. It contains the calculation and the diffusion of potential values in addition to all details concerning the replication request, the replication delegation and the replication correction. Finally, chapter 6 presents the series of simulation that was carried out to validate the key functionalities of our proposed model and to evaluate its performance.

CREAM:

TRIGGERING REPLICATION AND SELECTING DATA

This chapter describes the resources monitoring process based on which our proposed model CReaM determines whether it should take actions. It further specifies which action(s) is (are) to take depending on the case. In numerous cases, the action is replication. For the case of replication, we also explain how the data item(s) to be replicated is (are) selected.

The chapter is organized as follows. Section 4.1 presents fundamental assumptions under which our work was concluded. Section 4.2 provides details on the resource monitoring process. Section 4.3 describes the methods applied by CReaM to select its actions and to select the candidate data items for replication when needed. Finally, section 4.4 concludes this chapter.

4.1. Basic Assumptions and Definitions

In this section, we state basic assumptions and formalize several notions, which will be used in the remainder of this part of the thesis to present our replication model.

- **MANET formalization**

First of all, CReaM operates in a MANET. To formalize the notion of MANET, we adopt the model of evolving graphs proposed by Ferreira [30]. This model bases on classical models representing a network as a graph formed of nodes and links between these nodes representing one-hop connections. However, it extends them in order to capture the dynamicity of MANETs by associating a presence schedule to each node and link, which indicates the time instants in which they are present.

Definition 4-1: Evolving graph (Ferreira [30])

Given a graph G(V,E), an ordered sequence of its subgraphs $S_G = \{G_1, G_2, \ldots, G_n\}$: n ϵ $\mathbb{N}$ such that $\bigcup_{i=1}^{n} G_i = G$ and $T = \{t_0, t_1, t_2 \ldots, t_n\}$ an ordered sequence of time instants, the system $\Gamma = (G, S_G, T)$ where each G_i is the subgraph in place during time interval $[t_{i-1}, t_i]$ is called an evolving graph.

This evolving graph represents the set of states of the MANET over time. Evolving graphs, and the whole part of the thesis presenting CReaM, make use of a discrete-time model. Thus, when considering the MANET in general, we systematically consider its state at a specific time point defined as follows:

Definition 4-2: MANET state

The state of the MANET at a time point t_i is a graph $G_{t_i}(V_{t_i}, E_{t_i})$ such that $V_{t_i}$and E_{t_i} respectively represent the set of nodes and of links present in the MANET during the time interval $[t_{i-1}, t_i]$.

To distinguish between nodes, a unique identifier is assigned to each one. The set of all nodes present in the MANET over its lifecycle is denoted by V = {N_1, N_2… N_j}: j ϵ $\mathbb{N}$ where j is the total number of nodes and N_m ($1 \leq m \leq j$) is the node identifier. The subset of V denoted V_{t_i}then corresponds to the identifiers of the nodes present in the MANET during the time interval $[t_{i-1}, t_i]$.

- **Decentralized model**

Moreover, CReaM is a decentralized model in which the nodes make autonomous decisions. Thus, all the algorithms presented in these two chapters are executed locally on each node. The input parameters to these algorithms are computed or measured locally using the information at the disposal of the node at a specific time point $t_i \epsilon T$.

- **Peer-to-peer data exchange application**

CReaM is a replication model designed to support a peer-to-peer data exchange application deployed over the MANET. In this context, the nodes store data items, which are the target of data requests and replication requests. We assume that a unique identifier is assigned to each data item[4]. The set of all data items over the lifecycle of the MANET is denoted by D = {DI_1, DI_2… DI_k}: k ϵ $\mathbb{N}$, where k is the

[4] Many works have been done in this field for the peer to peer system (i.e. using hashing functions [108]

total number of data items and DI_i ($1 \leq i \leq k$) is the data identifier. The subset of D denoted D_{t_i} then corresponds to the identifiers of the data items present in the MANET during the time interval $[t_{i-1}, t_i]$[5].

Each node is assumed to participate fairly[6] to the data exchange and to CReaM. Thus, during a time interval $[t_{i-1}, t_i]$, a node may execute the following types of actions:

- Send a data request
- Receive a data request
- Send a data replication request
- Receive a data replication request

We also consider that users may use the devices participating in the MANET to execute personal tasks that are neither related to the data exchange application nor to replication.

4.2. Resources Monitoring Process

The goal of the resources monitoring process is to detect problems of excessive resource usage. This is implemented using predefined tolerance thresholds, which are compared to values computed from the monitored values.

[5] A data item and a replica of that data item have different identifiers even if they have the same content.

[6] We consider that all the nodes are cooperative, they participate in the system and there are no free-riders. The problem of free-rider nodes in the context of peer to peer system is devoted by a PhD student in our laboratory

4.2.1. Tolerance Thresholds

Definition 4-3:

A tolerance threshold is a numeric value that represents the allowed usage for a resource. Three tolerance thresholds are defined, one for each type of resource, as follows:

- ***The threshold β***: represents the CPU level used to process data requests.
- ***The threshold γ***: represents the battery level.
- ***The threshold δ***: represents the level of storage space used for storing replicas locally.

The usage of these three types of resources is monitored constantly and compared to the thresholds at regular time intervals. If one of the thresholds is breached, it is interpreted as a problem of lack of resources; CReaM then takes action(s) in order to fix it. CReaM can execute several types of actions, which are detailed in next section 4.3.

To make more accurate decisions, we distinguish between soft and hard values for each threshold. Soft thresholds represent less critical situations. As a result, the actions taken when reaching both types of thresholds are different. When reaching a soft threshold, preventive measures are taken with the goal of avoiding or delaying reaching the hard threshold. When reaching a hard threshold, emergency actions are taken.

Furthermore, an additional value is defined for the threshold γ. It corresponds to the gradient of the battery level; it is used to detect excessive consumption of the battery. Indeed, excessive battery consumption is a hint that the thresholds for remaining battery are probably going to be reached soon; thus considering the consumption enables to anticipate the problem by taking action in a timelier manner. Thus, if the consumption of the battery during the last time interval has exceeded this additional threshold, CReaM also launches an action.

Table 4-1 presents a summary of the tolerance thresholds and their relations.

Table 4-1: Tolerance threshold and their relations

Resource	Soft Threshold	Hard Threshold	Relation
The load on the CPU	β_1	β_2	$\beta_1 < \beta_2$
The remaining battery	γ_1	γ_2	$\gamma_1 > \gamma_2$
The battery consumption	γ_3		
The used storage space	δ_1	δ_2	$\delta_1 < \delta_2$

The tolerance thresholds can be adjusted dynamically by the system based on the current load caused by the user actions. When the user needs more resources to complete his personal tasks, CReaM adapts the thresholds to reserve more resources for him, i.e. it decreases the thresholds β and δ and increases γ. On the contrary, when the user is inactive or less active, the thresholds can be adapted again to let the node participate more in the system[7].

4.2.2. When to act?

In order to precisely define when CReaM launches actions, we define functions used by the node to formalize the consumption of CPU, storage space, and battery. As noted previously in 4.1, these functions are all computed with respect to the previous time interval or to the current time point. The functions are the following:

- NoR(t_{i-1},t_i): returns the number of requests processed by the node during the time interval [t_{i-1}, t_i]. We assume that the CPU used by the device for participating in the data exchange application is linearly proportional to the number of processed requests. Thus, NoR corresponds to the CPU load and is compared with the thresholds β_1 and β_2.

[7] This dynamic adjustment process of the tolerance thresholds is ongoing work (see the future work in chapter 8).

- SS(t_i): returns the amount of storage space occupied by replicas at time t_i. It is compared with the thresholds δ_1 and δ_2.

- BL(t_i): returns the battery level at time t_i. It is compared with the thresholds γ_1 and γ_2.

- BC(t_i): returns the battery consumption during the time interval [t_{i-1}, t_i]. It is calculated based on the function BL as follows: $BC(t_i) = BL(t_i) - BL(t_{i-1})$[8], and it is compared with the threshold γ_3.

Note that while the CPU and storage space related functions take only into account the resource usage related to the node's participation to the data sharing applications, the battery-related functions are global. Indeed, there is no simple way to distinguish between battery consumed to execute personal user tasks and to process data requests or manage replicas. However, this allows taking into consideration the user's usage of the device (which is considered using the thresholds γ_2, γ_3).

Using these functions and the tolerance thresholds, we define a set of conditions (named CONDITIONS). When at least one of them becomes true, an action is triggered.

$$Let\ t_i \epsilon\ T:\ CONDITIONS = \{NoR(t_{i-1}, t_i) \geq \beta_1,$$

$$NoR(t_{i-1}, t_i) \geq \beta_2, SS(t_i) \geq \delta_1, SS(t_i) \geq \delta_2,$$

$$BL(t_i) \leq \gamma_1, BL(t_i) \leq \gamma_2, BC(t_i) \geq \gamma_3\}$$

4.3. Which action(s) to take?

In this section, we fully specify the actions that are taken by CReaM when facing a resource usage problem. We first list all possible actions. For replication actions, the data items that are considered as good candidates differ according to the resource usage problem (e.g. the triggered condition), thus we define a classification of data

[8] Note that we have equal time intervals: $t_i - t_{i-1} = constate$

items to ease the selection process. Then, we present the mapping between conditions and actions, which form a system of rules determining CReaM's behavior.

Several conditions can be true simultaneously; this can be a problem as some actions are contradictory with each other. To resolve this problem, we introduce the concept of node status and define a conflict resolution mechanism, which enables us to propose a general algorithm that determines the action taken by CReaM in all cases.

4.3.1. CReaM's Actions

Actions are taken in order to alleviate resource consumption. In most cases, CReaM reacts by replicating data item(s); nevertheless in critical situations, other actions are possible. The actions of CReaM are classified into the following categories, where the triggering process will be presented in section 4.3.3:

- Replication: CReaM replicates a data item in order to increase its availability and simultaneously distribute the load of processing queries over other nodes, leading in turn to a decrease of the battery and CPU consumption.

- Selectively processing data requests: CReaM acts by being more selective in processing the data requests with the goal of decreasing the battery and CPU consumption.

- Selectively processing incoming replication requests: CReaM starts to only accept specific replication requests while rejecting others. This action affects directly the used storage space, but also the battery and the CPU usage, because of the cost of further processing data requests concerning replicas locally created.

- Delete replica(s): the goal in this case is to free storage space.

4.3.2. Classification of Data Items

When CReaM detects a need for replication, it must select which data item(s) to replicate. Depending on the case, a certain set (class) of data items is more appropriate for replication. Classifying the data items of the node into sets facilitates the selection process so that it turns it mainly to selecting the suitable set. In the following, we define the different sets of data items that are used by the model.

4.3.2.1. Set of Requested Data *Items* ("SDI_{rq}")

To define this set, the following definitions are necessary:

Definition 4-4: Access Counter

The Access Counter of a data item DI at time t_i *"AC(DI, t_i)" is* the number of requests received by the node for DI since t_j where tj corresponds to the connection of the node to the MANET. It is an accumulation starting from zero and increased by 1 each time a request for DI is received. The access counter is initiated for each data item upon its creation on the node.

According to this definition, the relation below hols true:

$$AC(DI, t_i) = \begin{Bmatrix} 0 : i = j \\ AC(DI, t_{i-1}) + NoReq(DI, t_i) : \ i > j \end{Bmatrix}$$

where NoReq(DI, t_i) is the number of requests for DI received between t_{i-1} and t_i

Definition 4-5: Temperature Degree

The Temperature Degree of a data item DI at time t_i, noted TD (DI, t_i) is a numeric value that indicates its "importance" for other nodes of the MANET based on the

number of recent requests for the DI received by the node that computes the TD. The TD starts from 0 and it is updated periodically based on the access counter AC.[9]

At time t_i the temperature degree changes (increases/decreases) based on the following rules (where AC (DI, t_i) and TD (DI, t_i) are represented for the sake of better clarity as AC_i, TD_i respectively):

- The temperature degree increases by x if the access counter has increased by x during the interval $[t_{i-1}, t_i]$.

$$\forall x > 0\ , AC_i - AC_{i-1} = x, \Rightarrow TD_i = TD_{i-1} + x$$

- The temperature degree remains unchanged at t_i if the access counter does not increase during one time interval. This rule reflects the fact that the data item DI is still considered important even if it has not been requested during the last period of time[10].

$$AC_i = AC_{i-1}\ \&\ AC_{i-1} > AC_{i-2} \neq 0\ \Rightarrow\ TD_i = TD_{i-1}$$

- The temperature degree decreases by a predefined constant y if the access counter has remained unchanged during more than one consecutive time intervals.

$$AC_i = AC_{i-1}\ \&\ AC_{i-1} = AC_{i-2} \Rightarrow\ TD_i = TD_{i-1} - y :\ \forall y > 0$$

Note that by definition $\forall i: TD_i \leq AC_i$. In the example of Figure 4-1, AC remains at 7 between t_4 and t_6, whereas TD starts decreasing by y=1 at t_5 until t_7 when TD and AC start increasing again.

[9] The temperature degree is local for the node, it concerns the local data item (an original data item or a replica for a data) and not the content of the data item.

[10] Note that AC is an accumulation: $\forall i: AC_i \geq AC_{i-1}$

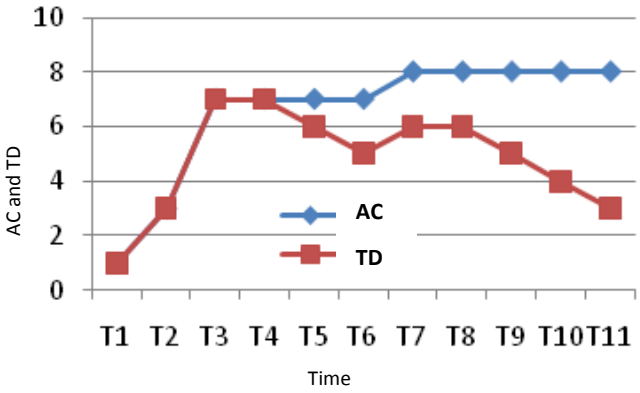


Figure 4-1: Access counter and temperature degree

Compared to AC, which is cumulative, TD represents an estimation of the current popularity of DI. Keeping the value unchanged when DI is not requested during one time interval introduces a certain amount of "inertia" in the computation, which prevents TD from varying too abruptly. The constant **y** enables to tune TD so that previous accesses are "forgotten" more or less quickly when DI starts to be less popular; i.e., **y** allows controlling the inertia of the system which actually depends on the characteristics of the application.

Definition 4-6: Set of Requested Data Items, $SDI_{rq}(t_i)$

We define the set $\mathbf{SDI_{rq}(t_i)}$ as the set of data items DI(s) requested at least once during the time interval $[t_{i-1}, t_i]$.

$$SDI_{rq}(t_i) = \{DI: AC(DI, t_i) > AC(DI, t_{i-1})\}$$

4.3.2.2. *Set of Hot Data Items ("SDI_α")*

This set includes the data items that are considered "important" because they have been recently highly requested by other nodes. These data items are called hot data items.

Definition 4-7: Set of Hot Data Items, $SDI_{\alpha}(t_i)$

We define the set of hot data items at time t_i, noted $SDI_{\alpha}(t_i)$, as the set of the set of data items whose temperature degree is equal to or greater than a threshold called the Hotness Threshold, noted α.

$$SDI_{\alpha}(t_i) = \{DI: TD(DI, t_i) \geq \alpha\}$$

At time t_i, a data item joins this category if its TD becomes equal or greater than threshold α and leaves it when it is no longer hot.

4.3.2.3. *Set of Rare Data Items of Interest ("SDI$_r$")*

This set contains the data items considered as rare in the network while being also potentially interesting for other nodes. More precisely, a data item belongs to this set if its content corresponds to the interests of several neighbor nodes and at the same time it has not yet been disseminated in the network. In order to formalize the first notion, let us introduce the concept of "hot subject" (or topic).

To this end, we assume that all data items are indexed and tagged by one or more subjects of interest. We suppose that this process can be applied to all data items, and that it has been conducted offline before the nodes joined the MANET. A large core of research has been devoted to automatically indexing and classifying documents including the research of our team in LIRIS laboratory [37], [72]. Multiple indexing methods have been proposed (i.e., using statistical and linguistic methods [6], [25], machine learning methods [50], neural networks, self-organizing maps [43]). The data items indexing process is beyond the scope of this thesis. We will further assume in this thesis that the subjects used by all nodes to index their data items belong to the same finite set of keywords (controlled vocabulary) as proposed in [31] where the authors propose a topics selection process to describe the contents of a given document based on a dictionary. They consider the list of topics to be large but static, i.e., pre-defined.

These assumptions enable us to define the concepts of "hot subject" and "rare data item of interest".

Definition 4-8: Hot Subject

At time t_i, a subject S is considered as hot if the ratio of the number of requests for the data items tagged by S during k time intervals $[t_{i-k}, t_i]$ to the total number of incoming requests during the same time $[t_{i-k}, t_i]$ is equal or greater than a threshold (μ)

$$\frac{\mathrm{NoReq_S}(t_{i-k}, t_i)}{\mathrm{NoR}(t_{i-k}, t_i)} \geq \mu \Leftrightarrow S\ is\ Hot$$

Where $\mathrm{NoReq_S}(t_{i-k}, t_i)$ is the number of incoming data requests considering S during k time intervals $[t_{i-k}, t_i]$, and $\mathrm{NoR}(t_{i-k}, t_i)$ is the total number of requests during $[t_{i-k}, t_i]$.

This concept of hot subject aims to determine which subjects have been popular over a relatively long period of time (hence the constant k).

Definition 4-9: Set of Rare Data Items of Interest, $SDI_r(t_i)$

A data item DI is considered as a rare data item of interest, if the following two conditions are met:

1. DI has not been requested during k time intervals.
2. DI is tagged by at least one hot subject.

As a consequence, the set of rare data items at time t_i, noted $SDI_r(t_i)$, is defined as:

$$SDI_r(t_i) = \{DI: AC(DI, t_i) - AC\ (DI, t_{i-k}) = 0 \wedge \exists\ S \in LoS(DI):$$

$$S\ is\ Hot\}$$

Where LoS(DI) is the list of subjects that tags DI.

4.3.2.4. Set of Non-*important Data Items* ("SDI_o")

Definition 4-10: Set of Non-important Data Items, $SDI_o(t_i)$

A data item DI is considered as non-important if its TD is zero and it is not tagged by a hot subject.

As a consequence, the set of non-important data items at time t_i, noted $SDI_o(ti)$, is defined as:

$$SDI_o(t_i) = \{DI: TD\ (DI, t_i) = 0 \wedge\ \forall\ S \in LoS(DI): S\ is\ not\ Hot\}$$

By definition, if the TD of a non-important DI starts increasing or if one of its subjects becomes hot, DI is removed from the set.

4.3.2.5. Relations between the sets of Data Items

Table 4-2 shows an example of data items available on a node, the subject that tags each data item and the evolution of the AC and TD during three time intervals. We assume that the constants are set as follows: the hotness threshold is set to α=9, the number of time interval used to identify the rare data items of interest is set to k=2, and the threshold used to identify the hot subjects is set to μ=0.5.

With this setup, at time t_{i+1}, the set of hot subjects is composed of 'Sport' since the number of requests concerning sport (5)[11] / total number of requests (9)[12] ≈ 0.56, whereas the ratio is of 0 for 'Music' and 4/9[13] ≈ 0.44 for 'Science'. At time t_{i+1}, the sets are the following:

- $SDI_{rq}(t_{i+1}) = \{DI_1, DI_3, DI_4\}$ as their AC has increased between t_i and t_{i+1};
- $SDI_{\alpha}(t_{i+1}) = \{DI_1, DI_4\}$ as their TD is equal to the threshold α=9;

[11] Since k=2: during $[t_{i-1}, t_{i+1}]$, DI_1 was requested 1 + DI_4 was requested 4 + DI_5 was requested 0

[12] During $[t_{i-1}, t_{i+1}]$: the number of requests= 1 (for DI_1) + 0 (for DI_2) + 4 (for DI_3) + 4 (for DI_4) + 0 (for DI_3) = 9

[13] During $[t_{i-1}, t_{i+1}]$: the number of requests for DI_3= 4

- $SDI_r(t_{i+1}) = \{DI_5\}$ as it has not been requested since at least t_{i-1} and it is tagged by the hot subject 'Sport';

- $SDI_o(t_{i+1}) = \{ DI_2\}$ as 'Music' is not a hot subject and $TD(DI_2, t_{i+1})= 0$.

Table 4-2: Example of data items sets

	DI_1		DI_2		DI_3		DI_4		DI_5	
	Sport		Music		Science		Sport		Sport	
	AC	TD	AC	TD	AC	TD	AC	TD	AC	TD
t_{i-1}	15	8	10	1	1	1	15	5	9	5
t_i	15	8	10	0	2	2	16	6	9	4[14]
t_{i+1}	16	9	10	0	5	5	19	9	9	3

Note that a DI may belong to more than one set at the same time. In addition, the DI(s) belonging to the same category can be tagged or not by a hot subject, except the DI(s) belonging to SDI_r and to SDI_o (see definitions above). Figure 4-2 shows the relations between the defined categories, and the distribution of data items that are tagged by hot subjects.

- SDI_r on the one hand and SDI_o and SDI_{rq} on the other hand, are mutually exclusive.

- SDI_o on the one hand and SDI_{rq} and SDI_α on the other hand, are mutually exclusive.

- SDI_{rq} and SDI_α may intersect, and finally SDI_r and SDI_α may intersect.

Note that it is possible that a data item present on the node does not belong to any of these sets, e.g., a data item with $0 < TD < \alpha$, which has not been requested since at least k time intervals and it is not tagged by hot subjects. Such a data item has no role to play in the replication process.

[14] We assume t_{i-2}=9

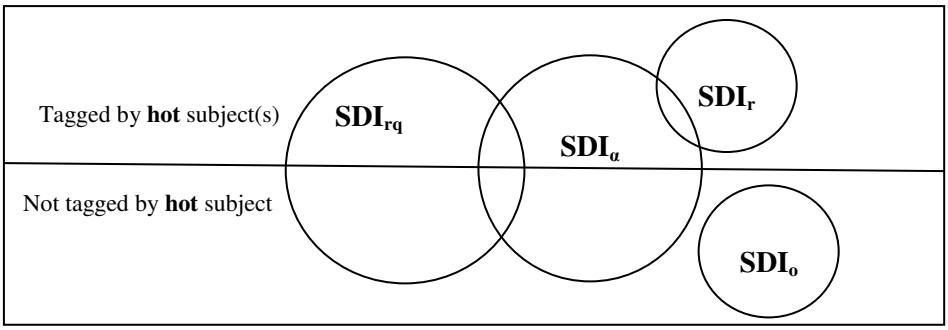


Figure 4-2: Relation between data items' sets

4.3.3. Mapping Conditions and Actions

We now discuss the actions taken by CReaM for each condition that triggers the process (cf. 4.2.2).

- **NoR(t_{i-1},t_i) ≥ β_1:**

When the number of processed data requests exceeds the soft value of the CPU threshold β, the node reacts by initiating a replication action to share the load of the requests with other nodes. Consequently, the NoR is expected to diminish. The question is which data item should be selected for replication?

The goal here is to find a data item for which we can expect many data requests in the next time periods. We may first think about SDI_{α}, the set of hot data items. One problem with this intuition is that these data items that have been highly requested should now exist in many exemplars in the network. Thus, even if more requests for them are generated, the other nodes can now act as additional data providers for these data items. Then, the current node will not receive these requests. On the other hand, the nodes that have previously requested the data items may have already left the MANET (if they are highly mobile, they can leave/disconnect frequently from the network); or they might be refusing to process data requests due to a lack of resources.

To determine to which case the current members of SDI_{α} belong, we consider the set $SDI_{\alpha} \cap SDI_{rq}$. If this set is not empty, then the node contains data items that are hot while still being regularly requested recently. The candidate data item should be selected from this set.

On the other hand, if the set is empty, requests for the hot data items are probably now being served by other nodes, and another set of candidate data items is to be considered.

Thus, in this case, the model uses other criteria to identify the most promising candidate data items for replication. To this end, CReaM tries to predict which data items that have not yet been heavily requested are more likely to be requested soon. To this end we make use of the notion of hot subject defined above (section 4.3.2.3). The assumption here is that hot subjects represent the subjects that currently interest the most users of the MANET, and that data items tagged by them have a higher probability to be requested soon. Furthermore, we argue that among these data items, those that have requested at least once recently are more interesting as this(these) request(s) are a concrete sign of existing interest for these data items in the network. The final criteria for selecting the data items are therefore the following:

1. It is tagged by a hot subject.

2. It was requested during the last period ($DI \in SDI_{rq}$): this condition ensures that the candidate data item has started to be requested and has actually interested users.

The process described above may yield to an empty set of candidates. In this case, a data item from the set SDI_{rq} is selected. Note that SDI_{rq} cannot be empty since the action is triggered when NoR exceeds the threshold β_1.

This process may also output more than one data item; as stated above in section 4.3.1, we aim to select a single data item for each replication action in order to save bandwidth. Thus, in such a case, the set of candidates is filtered until only a single candidate is left using the following criteria in order of priority:

- DIs tagged by the hottest subject.
- DIs with the highest temperature degree TD.
- Random selection.

Similarly to the previous analysis, to avoid replicating a data item that was already largely disseminated, we give the priority to a DI with a hot subject over a DI with a higher TD.

We present below action A_1 which is a generic action; it will be executed in case multiple conditions hold true. The $NoR(t_{i-1}, t_i) \geq \beta_1$ is one of these actions. In this case, the set SDI_{rq} in line 7 cannot be empty since the number of requests exceeds the threshold β_1 so the replication is triggered and line 25 cannot be reached.

Action 1 (A_1)
//Executed when $NoR(t_{i-1}, t_i) \geq \beta_1$

Variables
SDI: set containing the candidate data items
CDI: candidate data item

1. **If** $SDI_\alpha \cap SDI_{rq} \neq \{\emptyset\}$ **Then**
2. $SDI = SDI_\alpha \cap SDI_{rq}$
3. **Else**
4. $SDI = \{DI \in D: DI \in SDI_{rq} \wedge \exists S \in LoS(DI): Hot(S) = true\}$
5. //where LoS(DI) is the set of subjects that tags DI and Hot(S) returns true if S is hot
6. **If** $SDI = \{\emptyset\}$ **Then**
7. $SDI = \{DI \in D: DI \in SDI_{rq}\}$
8. **End If**
9. **End if**
10.
11. **If** (card(SDI) > 1) **Then**
12. $SDI = \{DI: DI \in SDI \wedge \exists S \in LoS(DI): Hottest(S) = true\}$
13. //where hottest(S) returns true if S is the hottest subject
14. **If** (card(SDI) > 1) **Then**
15. $SDI = \{DI: DI \in SDI \wedge \forall DI_j \in SDI: TD(DI_j, t_i) < TD(DI, t_i)\}$
16. **If** (card(SDI) > 1) **Then**
17. SDI = select randomly one DI from SDI
18. **End if**
19. **End if**
20.
21. **If** $SDI \neq \{\emptyset\}$ **then**
22. CDI is the only data item in the SDI
23. Replicate CDI

24. Else
25. **Return** //the replication is not initiated
26. End if

- **$NoR(t_{i-1},t_i) \geq \beta_2$:**

If the load on the CPU keeps increasing and reaches the hard threshold, the node reacts also by initiating a replication action aiming to share the load of the requests and to reduce the NoR (action A_1). In addition, another radical action must be taken in order to immediately preserve the CPU for the execution of the user's tasks. The chosen action is to stop responding to any data request (action A_2).

Action A_2 could have the unwanted side effect of decreasing the performance of the peer-to-peer data exchange application as the node might be the single provider for requested data items. However, if the node stops processing data requests at t_i, the evaluation of the parameters will output $NoR(t_i, t_{i+1}) = 0$ at t_{i+1}. Consequently, NoR will be below both thresholds at t_{i+1} and the node can start processing data requests again. Thus, this represents a good compromise between preserving the CPU and maintaining the performance of the data exchange application.

Action 2 (A_2)

1. Stop processing any data request

- **$SS(t_i) \geq \delta_1$ or $SS(t_i) \geq \delta_2$:**

In case of excessive use of the storage space, replicating does not really make sense, it is the local placement of replica that is impacted. Consequently, the node becomes more selective in the incoming replicas that it accepts: it starts only accepting urgent replica placement requests (action A_3). A replication request is considered as urgent if the replica is identified as a rare data item of interest (cf. 4.3.2.3). Indeed, these

requests concern data items that are potentially interesting to other users and are also at risk of disappearing from the network.

Furthermore, if the hard threshold is exceeded, the node needs to free some storage space allocated to replicas. Therefore, it removes a data item from the set SDI_o by applying a cache replacement algorithm[15] (action A_4).

Action 3 (A_3)

1. Refuse placing all non-urgent replicas

Action 4 (A_4)

1. Delete replica from SDI_o

- **$BL(t_i) \leq \gamma_1$ or $BC(t_{i-1},t_i) \geq \gamma_3$**

When the battery is consumed rapidly or when the soft threshold for remaining battery is reached, the action is to trigger a replication process in order to share the load with other nodes. In these two cases, the conditions and goals of the replication process are exactly the same as detailed above for the soft CPU threshold. Consequently, the same process is applied to select a data item to replicate: action (A_1) is triggered. The only difference is that here on line 7 of the algorithm the set SDI_{rq} can be empty. In this case, CReaM estimates that there is no data item that seems to have the potential of decreasing the load on the node by replicating it. Therefore the replication is not initiated to avoid consuming more battery.

[15] The choice of the cache management algorithms is out of the scope of this thesis. Good options would be either a classical replacement algorithm such as LRU or an algorithm taking advantage of semantic properties of the document [27].

- **$BL(t_i) \leq \gamma_2$:**

If the hard threshold is exceeded, the probability of a disconnection in the immediate future becomes high. The risk is that "interesting" data items that are only available from the node disappear from the network, which will globally decrease the data availability. Consequently, in this case more than one action is necessary to (1) avoid the data loss and simultaneously (2) avoid accelerating the battery consumption:

- A replication process targeting one or more rare data items of interest from the set SDI_r is initiated (cf. action A_5). If the set is empty, the replication process is not initiated to avoid consuming unnecessary battery with no expected benefit. In case several data items belong to SDI_r, the data item tagged with the hottest subject is selected. Due to the risk of disconnection, this action is considered "urgent"; the node adds this information in its replication request.

- Stop processing data requests unless the request concerns rare data item of interest (action A_6). This action is necessary to preserve the battery and so try to avoid or at least delay the disconnection.

- Stop accepting incoming replication requests (action A_7). Indeed, creating new replicas on a node that is expected to disconnect soon does neither enhance data availability nor achieve the load balancing. Moreover, it leads to further battery consumption on the node, worsening the current situation of the node.

Action 5 (A_5)

If ($SDI_r \neq \{\emptyset\}$) **then**
    If card(SDI_r) >1 **then**
        $SDI = \{DI: DI \in SDI_r \wedge \exists S \in LoS(DI): Hottest(S) = true\}$
        If card(SDI) >1 **then**
            SDI = random data item from SDI
    Else
        $SDI = SDI_r$
    End If
    Replicate CDI ∈ SDI
    Return
Else
    Return
End if

Action 6 (A_6)

1. Process only request concerning rare data items of interest

Action 7 (A_7)

1. Refuse placing replicas

Table 4-3 summarizes all actions. It contains the conditions and the corresponding action(s) when a condition holds.

Table 4-3: Summary of the monitoring conditions and actions

Condition		Action and Short Description	
C_1	$NoR(t_{i-1},t_i) \geq \beta_1$	A_1	Replicate: requested hot DI or requested DI related to hot S
C_2	$NoR(t_{i-1},t_i) \geq \beta_2$	A_1	Replicate: requested hot DI or requested DI related to hot S
		A_2	Stop processing any data request
C_3	$BL(t_i) \leq \gamma_1$	A_1	Replicate: requested hot DI or requested DI related to hot S
C_4	$BL(t_i) \leq \gamma_2$	A_5	Replicate rare data item of interest
		A_6	Process only requests of rare DI of interest
		A_7	Refuse placing replicas
C_5	$BC(t_{i-1},t_i) \geq \gamma_3$	A_1	Replicate: requested hot DI or requested DI related to hot S
C_6	$SS(t_i) \geq \delta_1$	A_3	Refuse placing all non-urgent replicas
C_7	$SS(t_i) \geq \delta_2$	A_4	Delete non-important replica

The mappings defined above may be written in the form of Event-Condition-Action (ECA) rules. ECA rules have been mainly used in active database systems [67]. An ECA rule R is a triple (Event, Condition, Actions), with the following semantics:

- The Event, denoted R.event, determines when the rule should be evaluated;
- The Condition, denoted R.condition, is a logical test; if it is true, the Action(s) is (are) triggered;
- The Actions, denoted R.actions, correspond to one or more processes to be executed if need be[16].

In our case, the rules are all evaluated periodically, thus we only have an event TimeUp such that the difference between the current system time and the time of the last evaluation equals the predefined time window.

Therefore, we can define CReaM's set of rules R as follows: R = {R_1, R_2, R_3, R_4, R_5, R_6, R_7}, with:

- R_1 = (TimeUp, C_1, A_1)
- R_2 = (TimeUp, C_2, {A_1, A_2})
- R_3 = (TimeUp, C_3, A_1)
- R_4 = (TimeUp, C_4, {A_5, A_6, A_7})
- R_5 = (TimeUp, C_5, A_1)
- R_6 = (TimeUp, C_6, A_3)
- R_7 = (TimeUp, C_7, A_4)

CReaM operates by periodically evaluating the conditions of these ECA rules. A condition that holds true triggers the actions of the corresponding rule. However, it is

[16] We have slightly extended the classical ECA rules by enabling a rule to trigger a set of actions rather than a single action.

possible that several conditions are verified simultaneously. This can be a problem as some actions associated with different rules are incompatible with each other. We deal with this problem in the next section, which enables us to fully specify CReaM's behavior.

4.3.4. CReaM's Behavior

To deal with the problem of contradictory actions, we first introduce in this section the notion of node status. This enables us to further introduce the set of actions that should be executed. We then describe the action conflict resolution process that is applied if this set contains incompatible actions. Finally, we detail the general algorithm that determines CReaM's actions for all possible statuses.

4.3.4.1. Node Status

A node is considered to be stable when the user is satisfied from the consumption of the three resources (CPU, battery, and storage space), i.e. when all the conditions of the set CONDITIONS are false. When the consumption of at least one resource exceeds a predefined limit, the node enters into another status. A status could be seen as a sub-set of CONDITIONS representing all conditions that are currently true. The set of possible node status is the power set of CONDITIONS. However, in practice a number of observations can be made in order to eliminate redundant and/or impossible status:

1. There is a fixed order for the hard and the soft thresholds for all resources. If we note C_h and C_s the conditions for the hard and soft thresholds respectively on a resource, at a given moment, either both conditions are false, or C_s is true and C_h is false or both conditions are true. In the latter case, only C_h is considered, i.e. only the actions corresponding to C_h should be triggered since they are designed for the most critical resource situations. Thus, in practice, the conditions corresponding to the soft and the hard threshold for the same resource are considered as mutually exclusive.

2. For the battery, the threshold of the rapid consumption γ_3 can be reached at the same time as either the soft threshold γ_1 or the hard threshold γ_2.

Based on these observations, we define the node status as follows:

Definition 4-11: Node Status

Let

$\boldsymbol{NoR} = \{NoR(t_{i-1}, t_i) \leq \beta_1,\ NoR(t_{i-1}, t_i) \geq \beta_1,\ NoR(t_{i-1}, t_i) \geq \beta_2\}$: be the set representing the CPU usage status

$\boldsymbol{BL} = \{BL(t_i) \geq \gamma_1, BL(t_i) \leq \gamma_1, BL(t_i) \leq \gamma_2\}$ be the set representing the battery level status,

$\boldsymbol{BC} = \{BC(t_{i-1}, t_i) \leq \gamma_3, BC(t_{i-1}, t_i) \geq \gamma_3\}$ be the set representing the battery consumption status,

$\boldsymbol{SS} = \{SS(t_i) \leq \delta_1, SS(t_i) \geq \delta_1, SS(t_i) \geq \delta_2\}$ be the set representing the storage space status,

The node status S is defined as the quartet $S = (nor, bl, bc, ss)$ where $nor \in NoR, bl \in BL, bc \in BC, and\ ss \in SS$. The set of all possible status denoted as Ω *is* the Cartesian product of the four sets defined above: $\Omega = NoR \times BL \times BC \times SS$.

Status notation:

- When there is no problem in the consumption of a resource, the function name is replaced by a dash.
- When the consumption of a resource exceeds a threshold, the corresponding function name in the quartet is replaced by the threshold.

For example, if the battery level becomes lower than the soft threshold ($BL(t_i) \leq \gamma_1$), BL takes the value γ_1 and the node status is represented by (-, γ_1, -, -). When the soft threshold of the battery and the soft threshold of the CPU are exceeded the node status is represented by (β_1, γ_1, -, -). The stable status is represented by (-, -, -, -).

Table 4-4 shows the list of the conditions and the used notations.

Table 4-4: Node status notation

	Condition	Notation in the status
NoR	$NoR \leq \beta_1$	-
	C_1: $NoR \geq \beta_1$	β_1
	C_2: $NoR \geq \beta_2$	β_2
BL	$BL \geq \gamma_1$	-
	C_3: $BL \leq \gamma_1$	γ_1
	C_4: $BL \leq \gamma_2$	γ_2
BC	$BC \leq \gamma_3$	-
	C_5: $BC \geq \gamma_3$	γ_3
SS	$SS \leq \delta_1$	-
	C_6: $SS \geq \delta_1$	δ_1
	C_7: $SS \geq \delta_2$	δ_2

Defining CReaM's behavior amounts to determine which action(s) must be executed for each possible status. To this end, we can define the set InitialActions(S), which lists the set of actions associated to a status through the ECA rules triggered by the status' conditions:

$$\forall S \in \Omega: InitialActions(S) = \bigcup_{i=1}^{7} R_i\,.actions \mid R_i.condition \in S$$

For example, InitialActions(S) = {A_1, A_4} is the list of actions associated to the status S = (β_1, -, - , δ_2). When the node enters this status, CReaM reacts by executing the actions belonging to this set.

The problem is that in a number of cases, the actions contained in InitialActions(S) cannot all be executed simultaneously due to incompatibility. For example, action A_3 (refuse placing all non-urgent replicas) cannot be executed at the same time as action A_7 (refuse placing all replicas). In such cases, a conflict resolution process must be applied before the actions are executed.

4.3.4.2. Action Conflict Resolution Process

In the following, we determine which actions are incompatible. First, we note that several actions trigger independent processes (e.g. replicating, processing data requests, processing replica placement) that can be executed simultaneously without causing a problem. These actions are said to be compatible for example, A_5: replicate rare data item of interest is compatible with A_2: stop processing any data requests.

Two actions can actually be **incompatible** if they concern the same process with different targets or parameters. The example mentioned above concerns two incompatible action A_3 and A_7 such that both concern the same process (processing replica placement) but with different targets (A_3: refuse placing all non-urgent replica while A_7: refuse placing replica).

By considering the seven rules presented in section 4.3.3, the actions below appear as non compatible:

- (A_3, A_7): both actions concern the replica placement process. A_3 only accepts placing replicas considered as rare while A_7 refuses placing requests regardless of their content.

- (A_2, A_6): both actions concern the processing of data requests. A_2 refuses processing any requests while A_6 is more selective and processes only the requests concerning rare data items of interest.

- (A_1, A_5): both actions concern the replication process. A_1 replicates a requested hot DI or requested DI tagged by a hot subject, while A_5 replicates a rare data item of interest[17].

We denote INCOMPACT the set of pairs of incompatible actions: INCOMPACT = {(A_3, A_7), (A_2, A_6), (A_1, A_5)}. The action conflict resolution process is required if

[17] As stated in section 4.3.3, CReaM is designed to replicate at most a single data item in each time interval, this is why these two actions are incompatible.

$InitialActions(S)$ includes at least one of the elements of incompact (where S is the current status).

The conflict resolution process consists in applying for each pair in INCOMPACT a rule from a set of rules called the action reduction rules. A reduction rule is defined as a triple ($Action_i$, $Action_j$, $Action_k$) with $i \neq j$ and ($Action_i$, $Action_j$) $\in$ INCOMPAT. The semantics of this type of rule is that if incompatible $Action_i$ and $Action_j$ are supposed to be executed simultaneously, $Action_k$ is executed instead. To design these rules, we have simply considered each case separately and determined which action should benefit the system the most. The produced rules belong to two categories:

- **Domination rules** of the form ($Action_i$, $Action_j$, $Action_i$): $Action_i$ is said to be dominating $Action_j$ and is executed in cases in which both actions would have been simultaneously triggered.

- **Derivation rules** of the form ($Action_i$, $Action_j$, $Action_k$) with $i \neq j \neq k$: when $Action_i$ and $Action_j$ would have been simultaneously triggered, a third action, which is neither $Action_i$ nor $Action_j$ is triggered.

The details of the analysis are provided below. For each case, we discuss the type of the reduction rule used to solve the conflict between the pair in INCOMPAT.

- (A_3, A_7): A_7 is triggered when the condition C_4: BL $\leq \gamma_2$ holds, while A_3 corresponds to the condition C_6: SS $\geq \delta_1$. Since exceeding the hard threshold of the battery risks a disconnection in the immediate future, placing replicas does lead neither to increase the data availability nor to share the load. (i.e. applying A_3 is not useful). Thus, to solve the conflict, a domination rule from the form (A_3, A_7, A_7) is applied.

- (A_2, A_6): Action A_2 is triggered by the condition C_2: NoR $\geq \beta_2$ corresponding to high CPU usage. Action A_6 is triggered by the condition C_4: BL $\leq \gamma_2$ which means low battery level and high risk of disconnection. Placing rare data items of interest can have a direct effect of data availability, which CReaM aims to preserve. Thus, we consider that processing data requests concerning this set of data items should

have a higher priority. Consequently, the conflict is solved by a domination rule of the form (A_2, A_6, A_6).

- (A_1, A_5): these actions trigger the replication process but select their candidate data item from different sets. A_1 selects a hot data item requested in the previous time interval, or if there is none, it selects a data item that is related to a hot subject and that was requested in the previous time interval. Oppositely, A_5 replicates a rare data item of interest.

  As in the previous conflicts, we must first give priority to the rare data items of interest because the data availability is CReaM's primary concern. Now the question is whether we should try to replicate another data item if there are no rare data items of interest. To resolve this issue, we must consider the conditions that can lead to A_1 (e.g. C_1, C_2, and C_5):

  If the condition is C_1 or C_2, it means that the number of incoming data requests was large in the last time interval. Thus, we replicate a hot data item if it exists to ensure its availability. On the other hand, if the condition leading to A_1 is C_5, we have at the same time a low battery level and high battery consumption. Clearly, disconnection is imminent and the only action is to replicate the rare data items. Thus, if there are no rare data items of interest and the cause of A_1 is C_5, no further action is required. This reasoning also holds if C_5 and C_1 or C_2 are simultaneously true.

  This process defines a new type of replication action, which we call A_8. Therefore, the conflict between A_1 and A_5 is resolved by the derivation rule (A_1, A_5, A_8). A8's algorithm is given below.

Action 8 (A_8)

1. SDI = {∅}
2. **If** ($SDI_r \neq \{\emptyset\}$) **Then**
3. SDI = SDI_r //replicate rare data item of interest
4. **Else**
5. **If** (! C_5) **then**

```
6.                  IF SDI_α ∩ SDI_rq ≠ {∅} Then
7.                      SDI = SDI_α ∩ SDI_rq   //replicate requested hot data item
8.                  End If
9.          End If
10. End if
11.
12. If (card(SDI) > 1) Then
13.         SDI= {DI: DI ∈ SDI ∧ ∃S ∈ LoS(DI): Hottest(S) = true}
14.         //where hottest(S) returns true if S is the hottest subject
15.         If (card(SDI) > 1) Then
16.             SDI = {DI: DI ∈ SDI ∧ ∀DI_j ∈ SDI: TD(DI_j, t_i) < TD(DI, t_i)}
17.             If (card(SDI) > 1) Then
18.                 SDI = select randomly one DI from SDI
19.             End if
20.         End if
21. End If
22.
23. If SDI ≠ {∅} then
24.         CDI is the only data item in the SDI
25.         Replicate CDI
26.         Return
27. Else
28.         Return //the replication is not initiated
29. End if
30.
```

To summarize, to determine which action(s) must be executed for each node status, the node proceeds as follows (algorithm statusActions). After determining the status S, the node prepares (1) InitialActions(S): the set of actions associated to the status S through the ECA rules triggered by its conditions, and considers (2) INCOMPAT: the set of all pairs of actions that are incompatible with each other. For each pair of actions that is both in INCOMPAT and InitialActions(S), the corresponding action reduction rule is applied. Once all conflicts have been resolved, the remaining actions can be executed.

Note that the three elements of INCOMPAT: (A_3, A_7), (A_2, A_6) and (A_1, A_5) are all mutually disjoint and that each reduction rule replaces them by an action that does not appear in any other reduction rule. Therefore, for a given status the process will always produce the same set of actions regardless of the order in which the reduction rules are applied and the reduction is not a recursive process.

Algorithm statusActions

1. Determine the node status S = (nor, bl, bc, ss)
2. InitialActions ← actions corresponding to each condition in S
3. FinalActions = InitialActions
4. Prepare INCOMPAT
5. **For** each pair $(a_i,a_j) \in$ INCOMPAT in InitialActions **do**
6. Apply reduction rule (a_i,a_j,a_k)
7. Delete a_i, a_j from FinalActions
8. Add a_k to FinalActions
9. **End for**
10. Execute FinalActions

Table 4-5 shows three examples applying the algorithm statusActions

Table 4-5: Example of action conflict resolution

Status S	InitialActions/INCOMPAT	Explanation
$(\beta_2, \gamma_2, -, \delta_1)$	$\beta_2 \rightarrow A_1 + A_2$ $\gamma_2 \rightarrow A_5+A_6+A_7$ $\delta_1 \rightarrow A_3$ InitialActions(S)= $\{A_1,A_2,A_5,A_6,A_7,A_3\}$ INCOMPAT=$\{(A_1,A_5)$, (A_3,A_7), $(A_2,A_6)\}$	1. Domination rule (A_3,A_7,A_7) $\rightarrow \{A_1,A_2,A_5,A_6,A_7\}$ 2. Domination rule (A_6,A_2,A_6) $\rightarrow \{A_1,A_5,A_6,A_7\}$ 3. Derivation rule (A_1, A_5,A_8) $\rightarrow$ **$\{A_8,A_6,A_7\}$**
$(\beta_1, \gamma_2, -, \delta_1)$	$\beta_1 \rightarrow A_1$ $\gamma_2 \rightarrow A_5+A_6+A_7$ $\delta_1 \rightarrow A_3$ InitialActions(S)= $\{A_1,A_5,A_6,A_7,A_3\}$ INCOMPAT= $\{(A_1,A_5)$, $(A_3,A_7)\}$	1. Domination rule (A_7,A_3,A_7) $\rightarrow \{A_1,A_5,A_6,A_7\}$ 2. Derivation rule $(A_1,A_5,A_8) \rightarrow$ **$\{A_8,A_6,A_7\}$**
$(\beta_1,-, \gamma_3, \delta_2)$	$\beta_1 \rightarrow A_1$ $\gamma_3 \rightarrow A_1$ $\delta_2 \rightarrow A_4$ InitialActions(S)= $\{A_1,A_4\}$ INCOMPAT= $\{\}$	No conflict **$\{A_1, A_4\}$**

4.4. Conclusion

In this chapter, we have presented the first part of our replication model. CReaM is a fully-decentralized replication model; it provides each node with a method to answer the main issues that define a replication model: **when** should the replication process be triggered, **what** data item should be replicated, and **where** should the replicas be placed. In this chapter, we have presented the answers of the first two questions.

The idea of using the resource usage to trigger the replication is due to the resources-scarce devices used in MANETs. By triggering replication when unsatisfying resource levels are detected (answering the "**when**" question), we aim to simultaneously increase the data availability while preserving resources. To this end, we have defined a set of **tolerance thresholds** for the main resources of a mobile node, namely the CPU, battery and storage space. Each threshold is associated to an action(s) triggered when a resource problem is detected. When actions correspond to replications, the data item to replicate (answering the "**what**" question) is selected based on the type of resource that triggers the replication process. To this end, the data items on the node are classified into categories according to the semantics of the document and the number of processed requests. The mapping between conditions (defined as an exceeded threshold) and an action defines an Event-Condition-Action rule. To generalize the model and deal with cases in which several conditions hold true at the same time, we have defined an action **Conflict Resolution engine** and introduce reduction rules used to solve conflicts and determine CReaM's actions in all possible cases.

CREAM:

REPLICA PLACEMENT

This chapter describes the replica placement process of CReaM. This process is an important point since it affects the load balancing obtained from creating replicas. We propose a placement process that predicts the adequate node to place the replica considering the content of the candidate data item and the interests of participating users. The replica placement is divided into two processes. First the information gathering process where a light weight solution is proposed to disseminate the interests of participating users (i.e., a list of subjects that is included in the profile) and to calculate and disseminate the estimation for each node to be the adequate node to carry replicas. Second, the placement process itself in which the candidate node to carry the replica is selected based on the information collected during the first process.

The chapter is organized as follows. Section 5.1 presents the requirements imposed by the replica placement process. Section 5.2 provides an overview of the contribution; the tackled issues and of the proposed solutions. Basic concepts and definitions are given in section 5.3. The replica placement process is divided into two processes: the placement process is described in section 5.4, while the information gathering process is described in section 5.5. The PInfo message proposed by CReaM to exchange this information is detailed in section 5.6. Finally, we conclude the chapter in section 5.7.

5.1. Requirements

The replica placement process in a mobile environment must satisfy some requirements, in order to be viable despite the constraints imposed by such dynamic networks like ad-hoc networks:

- **Exchange the minimum number of messages**: exchanging messages is energy consuming and imposes an important overhead. Hence, the number of exchanged messages needed to complete the replication process must be minimized.

  In this consideration, CReaM makes autonomous decisions in the replication initiation process in order to avoid periodical negotiation messages. However,

the replica placement is part of the replication process that cannot be done without exchanging messages. Indeed, two types of messages are needed for implementing the placement process: (1) a first type of messages is needed to collect information about the surrounding environment; this information will be later used to select the replica holder(s); (2) the second type of messages is needed during the replica placement process to exchange the data item to be replicated, and to transmit acknowledgments in case of a successful replica placement. Hence, the proposed system must strive to limit both types of messages.

- **Load balancing**: one of the main reasons for replication is to increase the data availability, so that any node belonging to the network can obtain rapidly the responses corresponding to its requests. A second reason is to balance the load of the data requests; the requests can actually distributed on multiple nodes instead of loading one single node. Hence, the selection of the replica holder(s) has a significant impact on the replication effectiveness. The replication system should select a node that is capable of sharing the load.

- **Resources Consumption**: another core idea of our contribution is to preserve the resources for the user. Therefore, the portion of the resources dedicated to the replication process must be adapted depending on the device's actual situation. Concerning the replica placement process itself, placing a replica on a device should be related to the dedicated portion of the storage space available for the replication, and to the battery level (see 4.3.3).

5.2. Contribution Overview

In order to achieve a good balance of the requests processing load, we propose to place the replica on a node that is connected to a maximum number of nodes interested in the content of the replica. The selected node might not be interested in the corresponding replica, but since it is surrounded by nodes that share the same interest in the content of the replica, we expect that it will help these nodes accessing the data item.

Thus we first identify the different communities that exist in the network. Here, a **community** is loosely defined as a set of nodes interested in a specific subject. When we want to place a replica of a data item, CReaM searches as candidate replica holder a node that is (1) with the highest one-hop connectivity with the corresponding community and (2) with enough resources. We expect here that the nodes which are interested in the content of the data item, will request it in the near future and that the selected replica holder will be close enough to answer the requests. For example, in our first scenario, presented in the introduction of this thesis (section 1.2.1), replicating a comment concerning a rock show on a device that is connected to a group of fans, should help them to find it rapidly.

To conclude, this solution aims at sharing the load of data requests, and at the same time, at decreasing the communication cost caused by the data retrieval since the replica is located geographically close to requestor nodes.

Issues and Proposed Solutions

Indeed, the calculation of the centers of interest requires exchanging messages to collect information about the interests of each connected user. These exchanges impose additional overhead. The main challenges here is to make a compromise between the number of exchanged messages needed for the calculation and the benefits of interests' calculation on the replication process effectiveness. In order to deal with this issue, CReaM proposes:

- An efficient protocol to exchange the necessary information using small messages called PInfo messages.

- Adopting a compromise by not maintaining a complete view of the surrounding vicinity. Oppositely, CReaM selects the replica holder based on a partial view concerning the communities of interest. That means, each node makes its decisions based on the information that it disposes at the moment of making the decision even if this information is not complete. That is, the selected replica holder might not be the optimal one if we consider the whole network, but according to the decider node that makes the decision based on its

partial view of the network, the selected node is considered as the most adequate. Our expectation, in this strategy, is to dramatically reduce the communication overhead while not significantly affecting the quality of the replica placement.

Conclusion

To accomplish the proposed solution, the replica placement process is divided into two separate processes:

(1) The information gathering process: It is responsible for managing the necessary information for the second process. It collects information concerning the distribution of the communities of interests in the network. To this end, each node calculates and disseminates the information concerning the communities of interests around it. Since the network is mobile, the distribution of the communities changes. Therefore, this process is done periodically to keep the information up to date.

(2) The placement process: responsible for selecting the replica holders and distributing the replicas. Contrary to the information gathering process, the placement process is not periodical; it starts once a need for a new replica has been detected. The decisions made during this process are based on the information collected by the first process.

The basic concepts and definitions for both processes are given in the next section. Then, we start explaining the placement process while the information gathering process is detailed later.

5.3. Basic Definitions

Below, we present the basic definitions required to describe CReaM's replica placement process. All these definitions are relative to the state of the MANET at a given time t_i. To simplify the notations, in the following sections we omit the time point t_i from the definitions. Therefore when we mention the MANET G(V,E), it is to

be understood as the graph representing the state of the MANET at a given time point. Similarly, all the definitions that make use of the graph properties (set V and/or E) are also to be understood as relative to a given state of the MANET.

Definition 5-1: Direct Neighbor

Let $G(V,E)$ be a MANET, and let $N_k, N_j \in V$. N_k is called **direct neighbor** to N_j if they are connected at one hop. We define DirNeig(N_j) as the set of Direct Neighbors of N_j:

$$DirNeig(N_j) = \{N_k \in V : \exists\ (N_k, N_j) \in E, k \neq j\}$$

For example, in Figure 5-1, DirNeig(N_2) = {N_1, N_3, N_6, N_7}.

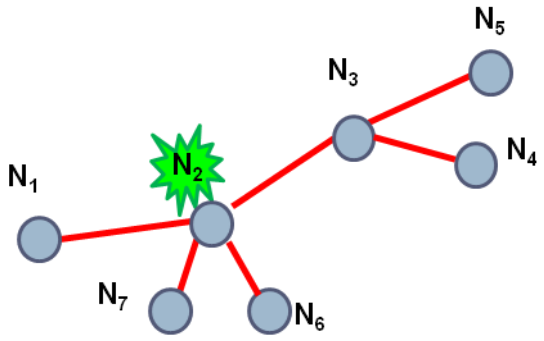


Figure 5-1: Example "Direct neighbors"

Definition 5-2: Node Profile

Let B be a set of keywords forming a controlled vocabulary and called subjects. We define the **profile of a node** N by the set $Prof_N \subset B$ that has the following properties:

1- $\forall S_i, S_j \in Prof_N \ \wedge i \neq j : S_i \neq S_j$

2- $1 \leq Card(Prof_N) \leq maxS$ where maxS is the maximum allowed number of subjects for N.

Let $S \in B$ be a subject and $N \in V$ be a node. We say that **node N is interested in subject S** or similarly that **S interests N** if S belongs to the profile of N:

$$S \in Prof_N \Leftrightarrow S\ interest\ N$$

We assume here that the subjects used by all nodes to describe the user interests belong to the same finite set of keywords (controlled vocabulary). Furthermore, maxS limits the number of the subjects that can be included in the profile so that the user only selects subjects that constitute a real interest for him.

Definition 5-3: Community of subject

We define C(S, N) as the **community of subject** S around node N by the set of direct neighbors of N interested in S. We denote this community by C(S, N) where C(S, N) $\subset DirNeig(N)$

$$C(S,N) = \{N_k : N_k \in DirNeig(N)\ \wedge S \in Prof_{N_k}\}$$

Definition 5-4: Potential of node for a subject

Let $S \in B$ be a subject and $N \in V$ a node. We define $P_S(N)$ as the **Potential** of node N for subject S by the cardinality of the community of S around N.

$$P_S(N) = card(\ C(S\ ,N)\ )$$

The potential value is a basic concept for the replica placement process. It is the estimation for node N to be the adequate node to carry replicas tagged by S.

For example, in Figure 5-1 above, if we assume that all nodes in the figure are interested in subject S, then $P_S(N_2) = 4$.

Each node locally calculates the potential of all possible subjects belonging to the set B. These values will help other nodes to decide whenever to pick this node as a candidate to carry a replica. To this end, the node diffuses the calculated potential values to the other nodes, and it maintains the list of the potential values of the other

nodes. Thus, two sets of potential values are to be distinguished on each node: "Personal Potentials" and "External Potentials".

Definition 5-5: Personal Potentials

Let $N \in V$ be a node, and B the set of subjects. We call **Personal Potentials** the set of couples (Subject S, Potential Value of N for the subject S) for all the subjects in B. These Potentials are calculated locally on the node N and noted PrsP(N)

$$PrsP(N) = \left\{ \left(S_j, P_{S_j}(N) \right) \in B \times \mathbb{N} \mid j \in \{1, .., |B|\} \right\}$$

Definition 5-6: External Potentials

Let $N \in V$ be a node, ε be a predefined threshold and B be the set of subjects. We call External Potentials on node N the set of triplets (Subject S, Node N, Potential value of node N for Subject S) that are known by N. We denote the External Potentials by ExtP(N).

ExtP(N) contains only:

- The significant External Potentials : $P_S(N_i) \geq \varepsilon : N_i \neq N$
- The best Potential values for each subject in B. More precisely:

Let $V_n \subseteq V$ be the set of nodes such that their potential values are known to N. We define the set of TopX nodes for subject S on node N as follow:

$$TopX(N,S) = \{N_1, \dots N_X \in V_N | \forall N' \in V_N \setminus \{N_1, \dots, N_X\} : P_S(N_i) \geq P_S(N') \}$$

$$where\ i = 1..X$$

Note that this definition is relative to a node N as the TopX nodes are determined based on the potential values that are known to N at the time of construction of the set; as the potentials change over time due to the dynamicity of the network, they must be periodically updated.

The set of **External Potentials** on N is then formally defined as:

$$ExtP(N) = \left\{\left(S_j, N_i, P_{S_j}(N_i)\right) \in B \times V \times \mathbb{N} \mid N_i \in TopX(N, S_j) \wedge P_{S_j}(N_i) \geq \varepsilon\right\}$$

Note that ExtP(N) can contain the Potential values of nodes that are not direct neighbors.

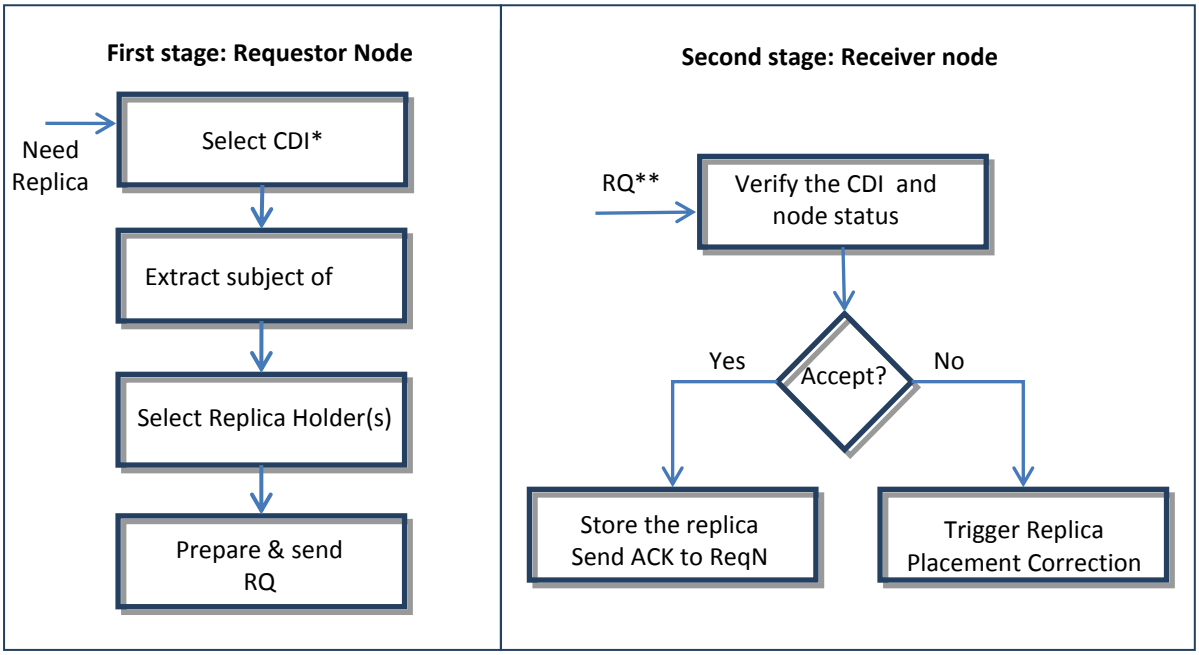
5.4. Placement Process

This process is executed when a rule of the ECA engine triggers a replication action as explained in chapter 4. It consists in selecting the replica holder(s) based on the External Potentials values collected during the information gathering process.

This process is divided into two stages (Figure. 5-2). The first stage is executed on the node that decides to create a replica. It includes the selection of the candidate data item (cf. chapter 4) and of the candidate replica holder(s), and terminates by sending the replication request on the network.

The second stage is executed on the candidate replica holder(s); it includes the creation of the new replica if the node status allows it (i.e., if the resources levels do not exceed the tolerance thresholds (cf. section 4.2.1). In case the node could not create the replica, this stage includes also the placement correction process that forwards the replication request to another node to move the replica to a suitable place. The next sections are dedicated to explain both stages, and their related topics.

In the following, we call the node which sends the replication request and asks for creating a new replica the Requestor Node (**ReqN**); while the selected replica holder is called the Receiver Node (**RecN**).



* CDI: Candidate Data Item ** RQ: Replication Request

Figure 5-2: The stages of the placement process

5.4.1. First Stage Executed on the Requestor Node ("ReqN")

Each connected node keeps track of the External Potentials; it receives from its neighbors their (Personal and External) potential values, and stores locally the highest X values for each subject (see definition 5-6). During the decision making process, the requestor node applies the following steps:

1. Extract the set of subjects S that tags the Candidate Data Item (CDI) that was selected for the replication process (see section 4.3.3). As explained in the previous chapter, many classical indexing methods have been proposed for; we actually based on a hierarchical solution that considers the subjects as a finite set of keywords (controlled vocabulary) [43] [31], [54].

2. Consult the External Potentials to select the q nodes with the highest Potential Value for the subjects selected in step 1, where q is equal to the number of replicas needed to be created. Some questions should be considered here:

- In case the External Potentials are not available or not sufficient, which node should be selected? For example, if the node wants to create 3 replicas for a data item related to "sport", while only 2 External Potentials are available, what is to be decided?

- In case several nodes have the same (high) Potential Value for the subject, which one should be selected?

- In case the candidate data item has more than one subject, which node(s) should be selected among the available candidate. For example, if the node wants to create one replica for a data item related to "sport" and "news"; and if the available External Potentials contain 2 values for "sport" and 2 for news. Which candidate replica holder should be selected?

3. Prepare the Replication Request (RQ) and send it to the replica holder(s) selected in step 2. For each RQ the requestor node ReqN expects to receive an acknowledgement message indicating the success of the replication placement process.

In the following, we detail the algorithm of the selection the candidate replica holders and the preparation of the replication requests. The complete algorithm of the first stage executed on requestor node is presented just after completing the description of both algorithms.

5.4.1.1. Selecting the Candidate Replica Holder(s)

In order to present the algorithm of selection of the Candidate Replica Holders, we first need to present our assumptions on the document indexing process. We assume that:

- All data items possess in their meta-data a **L**ist **o**f **S**ubjects denoted LoS. The subjects belonging to LoS describe the content of the document with the same

strength (i.e. we do not consider the case of different weights in the indexing relation between the document and the subjects).

$$LoS(DI) = \{S \in B : S\ tags\ DI\}$$

- The indexation process always identifies the most specific subjects corresponding to the document. Thus, if the document is related to more than one subject, where some subjects constitute specialization of others, only the most specific subjects are included to the list LoS. For example, if a document is related to "Sport" and "Football", only the subject "Football" is added to the list LoS as "Football" is a specialization of "Sport". We call the "Sport" a parent subject for the subject "Football"[18]. Details about the structure used to represent the subjects are presented in 5.5.1.

Indeed, to select the candidate replica holders, our algorithm selects the nodes that have the highest potential value for at least one subject from the list LoS. The number of replica holders that the algorithm should select is equal to number of the replicas[19] (NoR) that must be created. The following cases are distinguished:

- The External Potentials on the node contain information about the subjects in the list LoS: thus, the nodes which have the highest Potential Values are selected (regardless the subject which these values are related to as long as it belongs to LoS). Let us consider the following example: the requestor node N_1 should select NoR=2 replica holders for the candidate data item CDI that is tagged by the following list of subjects: $\{S_1, S_2\}$. N_1 has the External Potentials shown in Figure 5-3:A[20]. We note that the table contains 3 Potential Values for subjects S_1 and S_2, so the selected replica holders are trivially nodes (N_3, N_4) that have the highest values.

[18] A subject can have more than one parent subjects (cf. 5.1.1).

[19] As explained in chapter 4, CReaM replicates only one DI each time the replication process is triggered; however, it can send multiple replication requests to create multiple replicas for a single DI.

[20] For the sake of simplicity, the triplet of the External Potential (S, N, V) is represented as $P_S(N)$.

In addition, if the External Potentials contains for the same node Potential values for more than one subject listed in LoS, the sum of these values is considered. Let us consider the following example: NoR=1, LoS={S_1, S_2} and the External Potentials are those shown in Figure 5-3:B. We can note that the highest potential value is $P_{S_1}(N_4)$=17. However, two potential values of N_3 exist and the sum of the two values equals to 27; thus N_3 is selected as replica holder since it is connected to the maximum number of users interested in the content of the replica.

A

$P_{Si}(\dots)$	Value
$P_{S1}(N_3)$	15
$P_{S1}(N_4)$	17
$P_{S2}(N_2)$	12
$P_{S3}(N_6)$	11

B

$P_{Si}(\dots)$	Value
$P_{S1}(N_3)$	15
$P_{S1}(N_4)$	17
$P_{S2}(N_3)$	12
$P_{S3}(N_6)$	11

C

$P_{Si}(\dots)$	Value
$P_{S1}(N_3)$	15
$P_{S3}(N_4)$	18
$P_{S3}(N_5)$	17
$P_{S4}(N_6)$	12

Figure 5-3: Example of External Potentials on node N_1

- The External Potentials table does not contain enough Potential Values related to the subjects in the list LoS (i.e. if the number of values is less than the number of requested replica NoR): in this case, we search if Potential values for the parent subjects are available in the External Potential table and we select the highest values.

  Let us consider the example: Node N_1 should select 2 replica holders for the candidate data item CDI that is indexed by the subjects: {S_1, S_2} where S_1 is "Football" and S_2 is "Shopping". We note that the External Potentials table (see Figure 5-3:C) contains only one potential value for S_1 and no value for S_2. So the node N_3 is considered as a replica holder.

However, another node should be selected. In fact, the table contains information about the subject S_3 "Sport" that is considered as parent for S_1 "Football". Two Potential Values for S_3 are available (potential on node N_4 and N_5). Thus the corresponding node with the highest value (N_4) is selected to be candidate replica holder along with N_3 selected before.

It is noteworthy that, even if the potential values of parent subjects are higher than the potential values of their children, our algorithm gives priority to the children subjects. Back to example Figure 5-3:C we call that N_1 should ask for 2 replicas; even if the two potential values for S_3='Sport" are higher than the available potential for $S_{1=}$"Football", the algorithm selects N_3 because S_1 belongs to the list LoS that indexes the document CDI so that it reflects more precisely the content of the candidate data item.

- Finally, if the External Potentials table does not contain enough information about neither the subjects in the list LoS nor their parents, the requestor node completes the list of candidate replica holders by randomly selecting nodes.

In the following, the algorithm **selectReplicaHolders** that implements this strategy is presented. It makes use of the following data types:

- **Subject:** subject of the indexing hierarchy (see 5.5.1). Provides the methods (1) getParentSubjects() which returns the list of direct parent subjects. (2) getID () which returns the unique identifier of the subject.

- **Potential**: object containing four attributes (subject, nodeID, value, timestamp). The timestamp is used by the node to indicate the time instant at which the potential is locally registered.

- **PotOrderedList**: ordered list of Potential objects. The operations applied to an object of this type provide the following guarantees: (1) the list cannot contain two Potential objects with the same (subject, nodeID); (2) the list is maintained sorted by decreasing order of potential value. If an operation (from the list of operations presented below) must insert a Potential object with a (subject, nodeID) that already exists in the list, the attribute 'value' of the existing

object is updated by adding the new 'value' of the Potential parameter of the operation. Then the object is moved to the proper rank in the ordered list.

The following methods[21] are defined for this data type:

1. **addListOfElements**(ListOfPotentials): adds a list of Potential objects to the PotOrderedList while respecting the constraints defined above.

2. **addElement**(Potential): adds a Potential at its proper place in the list, or updates it as defined above if its nodeID with the same subject is already in the list.

3. Integer: **getNumberOfElements**( ): returns the number of elements in the list.

4. PotOrderedList: **getFirstElements**(num): returns the top "num" elements of the list as a PotOrderedList. If the list contains less than num elements, it is returned without modification.

5. Boolean: **notEmpty**(): returns true if the list is not empty and false otherwise.

6. Potential: **removeElement**(): returns the first Potential object of the list and removes it from the list.

7. **deleteElement (** Potential): deletes from the list the object with the attribute nodeID equal to the nodeID of the Potential parameter.

8. **randomComplete**(nb): randomly selects nb nodes (from the routing table) not in the list and inserts them in the list.

9. **replaceTheSmallest**(Potential): replaces the Potential object with the smallest attribute 'value' with the parameter Potential.

10. Potential: **getElement**(i): returns the i^{th} Potential object in the list without removing it.

[21] Some of these methods are used in the selectReplicaHolders algorithm, while the others are used later in the chapter.

The main parameters and variables used by the selectReplicaHolders algorithm are the following:

1. **ExtPotTable**: array of PotOrderedList in which each element stores the External Potentials of one subject (see Figure 5-4). The PotOrderedList of subject S is accessed by the subject's ID (i.e. ExtPotTable[S.getID()] is the list of the Potential for subject S).

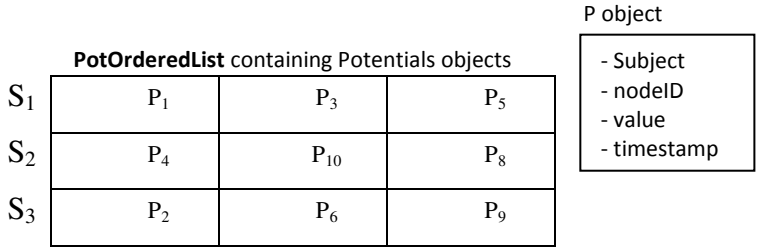


Figure 5-4: ExtPotTable and PotOrderedList

2. **ExtPotSet:** object of type PotOrderedList used to keep the Potential objects containing the replica holders selected by the algorithm (i.e. the output of this algorithm is a list of nodeIDs where each nodeID is the id attribute of a Potential object listed in the ExtPotSet.

3. **LoS, tmpLoS:** list of subjects. LoS is initialized with the list of subjects of the candidate data item. If these subjects correspond to less than NoR replicas, it is used together with tmpLoS to recursively explore the parents of the subjects until enough replica holders have been identified.

Algorithm 5-1

Algorithm selectReplicaHolders

Input

LoS: list of subjects of the candidate data item (type: array of Subjects)
ExtPoTable: table containing the External Potentials of the node (type: array of PotOrderedList i.e., list of Potentials for one subject sorted in decreasing order)
NoR: requested number of replicas (type: Integer > 0)

Output

CRHs: list of candidate replica holders (type: list of Integer corresponding to nodeIDs)

Variables

ExtPotSet, **ExtPotSet'**: variables used to construct the list of selected replica holders (type: PotOrderedList)
NoRH: number of replica holders that remain to be selected (type: Integer > 0)
tmpList: temporary ordered list of Potential used for readability (type: PotOrderedList)
parentSubjects: list of Subjects used to recursively explore the parents of LoS (type: array of Subjects)
potential: temporary object (type: Potential)
tmpLoS: temporary list of subjects used to keep the parent subjects that were visited by the algorithm to avoid visiting them more than once (type: array of Subjects)

Begin

```
1.  For (each S in LoS) do
2.      If (ExtPoTable[S.getID()] != Null) then //External potentials for S exist
3.          tmpList = ExtPoTable[S.getID()].getFirstElements (NoR) /*get the
4.                                          Potential objects with the highest values */
5.          ExtPoSet.addListOfElements( tmpList ) )
6.          /*The addListOfElements() takes an ordered list of Potential objects
7.          and adds them while maintaining a descending order of value. If the nodeID
8.          exist, the algorithm updates the nodes by adding the sum of both values and
9.        moving it to the proper rank*/
10.     End if
11. End for
12.
13. ExtPoSet = ExtPoSet.getFirstElements(NoR) /* updates the list of replica holders so that it
14.                                          does not contain more than NoR elements*/
15. //to calculate how much nodes still missing
16. NoRH = NoR – ExtPoSet.getNumberOfElements()
17. tmpLoS = ϕ
18. While ( NoRH > 0 and (exist subjects in LoS)) do
19.     /* The number of external potentials is smaller than the number of the replica needed
20.     → recursively search for the Potentials of parent subjects. If the last list of processed
21.       parent subjects only contains the Root, it means that all subjects have been
22.      investigated thus we must exit the loop*/
23.
24.     For (each S in LoS) do
25.         parentSubjects = S.getParentSubjects() //returns the list of parent subjects of S
26.         For (each S' in parentSubjects) do
27.             /*test if S' was not processed before in case two subjects has the same
28.              parent*/
29.             If (S' does not belong to tmpLoS ) then
30.                 If ( ExtPoTable[S'.getID()] != NULL ) then
31.                     //get the highest potentials and add them to temporary set ExtPoSet'
32.                     tmpList = ExtPoTable[S'.getID()].getFirstElements(NoRH)
33.                     ExtPoSet'.addListOfElements(tmpList)
34.                 End if
35.                 /* keep the manipulated parent subjects in a temporary list. In case not
36.                   enough nodes are identified after manipulating the parents of LoS, the
37.                 parent subject of the subjects in this list is recursively considered*/
```

```
                        tmpLoS.addElement(S')
                    End if
                End for
            End for

        //get first NoRH element in case the loop lead to a set with  more than NoRH potentials
            ExtPoSet.addListOfElements(ExtPoSet'.getFirstElements(NoRH))
            LoS = tmpLoS //in case we need to explore an additional level of parent subjects
            NoRH = NoR – ExtPoSet.getNumberOfElements()
End while

If (NoRH > 0) then
        //the process still has not identified enough nodes, we randomly complete the list
        ExtPoSet.randomComplete(NoRH)
End if

//Build the final list by extracting the nodeID from the Potential objects
While ( ExtPoSet.notEmpty() ) do
        potential = ExtPoSet.removeElement()  //return one potential object
        CRHs.addElement( potential.nodeID )  //add the nodeID to the list
End while

Return CRHs
End
```

5.4.1.2. Replication Request (RQ)

The replication request is prepared in two cases:

- The node is a requestor node: the RQ is prepared after determining the data item that the node has decided to replicate and selecting the candidate replica holders.

- The node is a receiver node: Indeed, a receiver node may be unable to create a replica as requested; in this case it propagates the RQ to another node. This process is explained in details in section 5.4.2,

Below, we present the list of fields contained in RQ (Figure 5-5)

- CDI: candidate data item

- SenderID: id of the requestor node.

- DestinationID: id of the destination node.
- OrgSenderID: id of the original sender of the request. Actually, if the receiver node decides to resend a replication request it keeps this field unchanged. The interest of OrgSenderID is to identify to which node the acknowledgement message should be sent when the replica is created.
- CRHs: list of candidate replica holders. This field contains all node IDs that the requestor node has selected. This field informs the receiver nodes about the other nodes that have received the replication request so that they do not select them again if they have to further propagate the request.
- TTL: Time-To-Live used to prevent endless propagation of the replication request in the network. A receiver node that decides to forward the replication request decreases the TTL by one in the new request. When the TTL equals 0, the receiver node is not allowed to create a new request.
- Rare: Boolean field to determine if the replica is a rare data item of interest. The replication request that concerns a rare data item is considered **Urgent**; thus, the node which receives the request, accepts it even if the soft threshold of the storage space is exceeded (cf. 4.3.2.3).

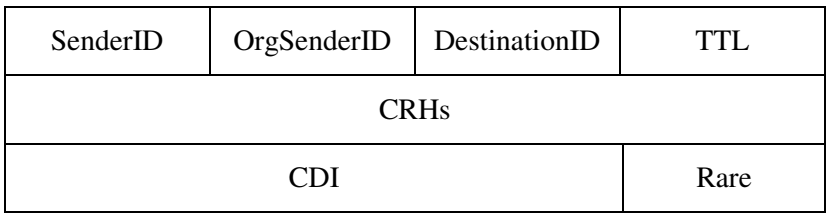


Figure 5-5: The Replication Request

We present in the following the algorithm used to create the replication request. It makes use of the following data type:

- **DataItem:** object containing the attributes: candidate data item CDI and rare. Provides the method getListOfSubject(), which returns the list of subjects that indexes the data item and the method isRare(), which returns true if the data item is considered as a rare data item of interest.

Algorithm 5-2

Algorithm prepareReplicationRequest

/* The algorithm takes as input the replication request and modifies it. It is called by a requestor or a receiver node.
The requestor node affects to the senderID and OrgSenderID its own ID and to the TTL the suitable value. Finally, it adds to CDI, destinationID and CRHs the parameters of the algorithm CDI, destinationID and CRHs respectively.
The receiver node affects to the senderID its own ID, to destinationID the parameter of the algorithm, decreases the TTL by one, and adds the new destinationID to the CRHs
Note that in case of receiver node, the parameters CDI and CRHs are null */

Input

RQ: the replication request: null if the node is the requestor
CDI: the data item object containing the data and the attribute rare (type: DataItem)
destinationID: the destination nodeID (type: Integer > 0)
CRHs: list of candidate replica holders selected by the algorithm selectReplicaHolders

Output

RQ: the replication request after being modified

```
Begin
1.  If (RQ is Null) then
2.          //the case of requestor node
3.          RQ.setCDI (CDI) //CDI contains the data and the attribute rare
4.          RQ.setTTL( fixed value )
5.          RQ.setOrgSenderID ( nodeID )
6.          RQ.CRHs( CRHs )
7.  Else
8.          //the case of receiver node
9.          RQ.decreaseTTL();
10.         RQ.addToCRHs( destinationID )
11. End if
12.
13. RQ.setDestinationID ( destinationID )
14. RQ.setSenderId( nodeID )
15.
16. Return RQ
End
```

5.4.1.3. Algorithm of the Placement Process executed on the Requestor Node

We can now present the complete algorithm of the first stage executed on the requestor node **replicaPlacementOnReqNode**. It calls the previous two algorithms **selectReplicaHolders** and **prepareReplicationRequest**.

Algorithm 5-3

Algorithm replicaPlacementOnReqNode

Input

CDI: candidate Data Item (type: DataItem)
NoR: number of requested replicas (type: Integer)
ExtPoTable: table containing the External Potentials of the node (type: array of PotOrderedList (one list of Potentials for one subject sorted in decreasing order)

Output

Send RQ to each candidate replica holders

Variables

LoS: list containing the subjects that tag the candidate data item (type: array of Subjects)
CRHs: list of candidate replica holders (type: list of Integer corresponding to nodeIDs)
RQ: replication request

Begin

1. LoS = CDI.getListOfSubject() //extract the list of subjects
2. CRHs = selectReplicaHolders[22] (LoS, ExtPoTable, NoR)
3. **For** (each CRH in CRHs) **do** //prepare and send RQ for each candidate replica holder
4. RQ = null /* RQ is initiated to null to distinguish in prepareReplicationRequest the
5. case of requestor/receiver node */
6. RQ = prepareReplicationRequest[23] (RQ, CDI, CRH, CRHs)
7. send(RQ, CRH); //send the replication request to candidate replica holder
8. **End for**

End

[22] See Algorithm 5-1

[23] See Algorithm 5-2

5.4.2. Second Stage Executed on the Receiver Node ("RecN")

The job of the requestor node finishes at the moment of sending the replication request to the candidate replica holders; it then **delegates** the replication process to the receiver nodes (i.e., to the candidate replica holders).

Hence, if the receiver node cannot accept the replication request for any reason, it should complete the replication process by selecting another replica holder and sending a new replication request. Thus, the nodes collaborate to complete the replication rather than sending back a rejection message to the requestor node and starting the operation again. This delegation has a Time-To-Live to avoid overloading the network with unsuccessful replication requests.

When a node N receives a replication request RQ, it may refuse placing the replica depending on the following criteria:

- The candidate data item CDI already exists on the receiver node.
- The node status does not allow the replica placement. As explained in the previous chapter (cf. section 4.3.3), the node acts by refusing placing replicas in two cases: (1) the used storage space by the replicas exceeds the threshold δ_1, therefore the node processes only replication requests concerning rare data items. (2) The battery level is below the threshold γ_2.

In both cases, the node applies the Replica Placement Correction Process explained in the next section.

5.4.2.1. Replica Placement Correction

In our model, the node makes its decisions based on its partial and incomplete view of the network. For example, it does neither know the distribution of the data items on the connected nodes, nor all External Potentials, nor the resources status of the nodes. Thus, the requestor node might send a replication request to a node that cannot carry

the candidate data item. In this case, the receiver node refuses the replica it applies the replica placement correction.

The correction process answers the following question: should the replication request be forwarded to another node? The answer bases on the content of the replication request. When the replication request concerns a **rare** data item of interest and the receiver node **already has** a copy of it, the replication request should not be forwarded to another node. Indeed, this RQ is sent because the requestor node has estimated that the CDI does not exist on the neighbor nodes. Since it exists on the receiver node, the later corrects the requestor decision by stop forwarding the RQ. However, in case the node does not have the data item, it forwards the request to another node regardless the type of the replica.

5.4.2.2. Algorithm of the Placement Process Executed on the Receiver Node

The algorithm of the replica placement on RecN uses a modified version of the candidate replica holder selection algorithm (see Algorithm 5-1), in order to avoid selecting a node that has already received a replication request. This version takes an additional parameter, an exclusion list (initialized to the CRHs field from the received request). Calls to functions addListOfElements() and completeRandom() are replaced by calls to modified functions addListOfElementsNotIn() and completeRandomNotIn(), which take the exclusion list as an additional parameter, only adding a Potential if its nodeID attribute does not belong to the exclusion list.

Algorithm 5-4

Algorithm replicaPlacementOnRecNode

Input

RQ: replication request
ExtPoTable: table containing the External Potentials (type: array of PotOrderedList)

Output

Verify if the node can accept the replica placement, and apply the replica placement correction process if necessary.

Variables

CDI: candidate data item (type: DataItem)
reqN: id of the requestor node (type: Integer)
CRHs: list of candidate replica holders (type: list of Integer)
newCand: new candidate replica holder (type: Integer)
newRQ: new replication request to be sent (type: ReplicationRequest)
sendNewRequest: Boolean

Begin

```
1.  CDI = RQ.getCDI ( )
2.  reqN = RQ.getOrgSenderID ( )
3.
4.  sendNewRequest = false
5.
6.  If (CDI exists on the node) then //in this case verifying the node status is not necessary
7.                                    since the CDI exists and the node will not place it again
8.
9.          /*If it is rare the node does not send another RQ otherwise, the node send RQ  */
10.         If ( RQ.isRare() = false) then
11.                 sendNewRequest = true
12.         End if
13. Else
14.         /* Test the node status  (δ1 and γ2):
15.            The replica is placed if (1) the battery level does not reach the threshold γ2 and the
16.            used storage space does not exceed the threshold δ1.If it reaches the threshold  δ2
17.            the placement is done only for a request concerning a rare DI of interest[24].
18.         */
19.         If ( node status allows processing the request  ) then
20.                 //(resources are available & CDI does not exist) → place replica
21.                 create the replica
22.                 send ACK message to reqN
23.         Else
24.                 sendNewRequest = true
25.         End if
26. End if
27.
28. //send a newReplicationRequest if needed
29. If (sendNewRequest = true and RQ.getTTL() != 0) then
30.         CRHs = RQ.getCRHs()  //return the nodes that received the same RQ
31.         /*select new candidate replica holder that does not belong to the list CRHs
32.           the API selectReplicaHoldersNotIn takes as parameters: 1- list of subjects indexing
33.           CDI. 2- the external potentials. 3- number of replica holder needed. 4-list of nodes
34.           already received RQ */
35.         newCand = selectReplicaHoldersNotIn(
36.                                 CDI.getListOfSubject() ,
37.                                 ExtPoTable,
38.                                 1,
39.                                 CHRs)
```

[24] In case the remaining storage space is not sufficient to store the new rare replica, a memory management algorithm should be used to replace the new replica with a non-important data item.

40. newRQ = prepareReplicationRequest[25](RQ, CDI , newCand , CHRs)
41. newRQ.send(newCand) //send the newRQ to the new candidate newCand
42. End if
End

After presenting the replica placement process, we can now dedicate the next section to the second process involved in the replica placement: the information gathering process.

5.5. Information Gathering Process of the Replica Placement

The information gathering process keeps the information necessary for the placement process updated. It is responsible for calculating and disseminating the Personal and the External Potentials. During this process, each node executes the following steps:

- It collects information about the existing communities around it. To this end, it receives the profiles of its direct neighbors and extracts their subjects of interest. Periodically, it uses the collected information to calculate its Personal Potentials (e.g. its potential for all subjects). Then, it disseminates these values to the other nodes in order to be used by them in the replica holder selection process.

- When receiving the potential values from another node, it participates in the dissemination process.

In order to explain the algorithms of calculation and dissemination of the potential values, we need to first present the data structure used to represent the subjects of interest, and our strategy to disseminate the profiles of nodes containing the subjects that interest the node.

[25] See Algorithm 5-2

5.5.1. Data Structure Representing the Subject Hierarchy

In CReaM, we have made the assumption that the documents are tagged by subjects representing user interests (cf. section 4.3.2.3). These subjects belong to a finite set of keywords organized in an n-ary hierarchy with a single root. The internal and leaf nodes of this data structure correspond to the user interests, while the root is a virtual node that does not represent any user interest.

The nodes in the hierarchy are connected by links; these links represents an **Is-A** relation between a father and a child node, e.g. between a subject and a sub-subject. Figure 5-6 shows a part of the hierarchy, in which football is a sub-subject of sport and the relation between them is presented by a directed link from sport to football.

Definition 5-7: the Subjects Hierarchy

We define the Subjects Hierarchy as an oriented graph H (B, L) where B is the set of subjects and L is the set of the links between the subjects. We note LF(H) the set of H leaves.

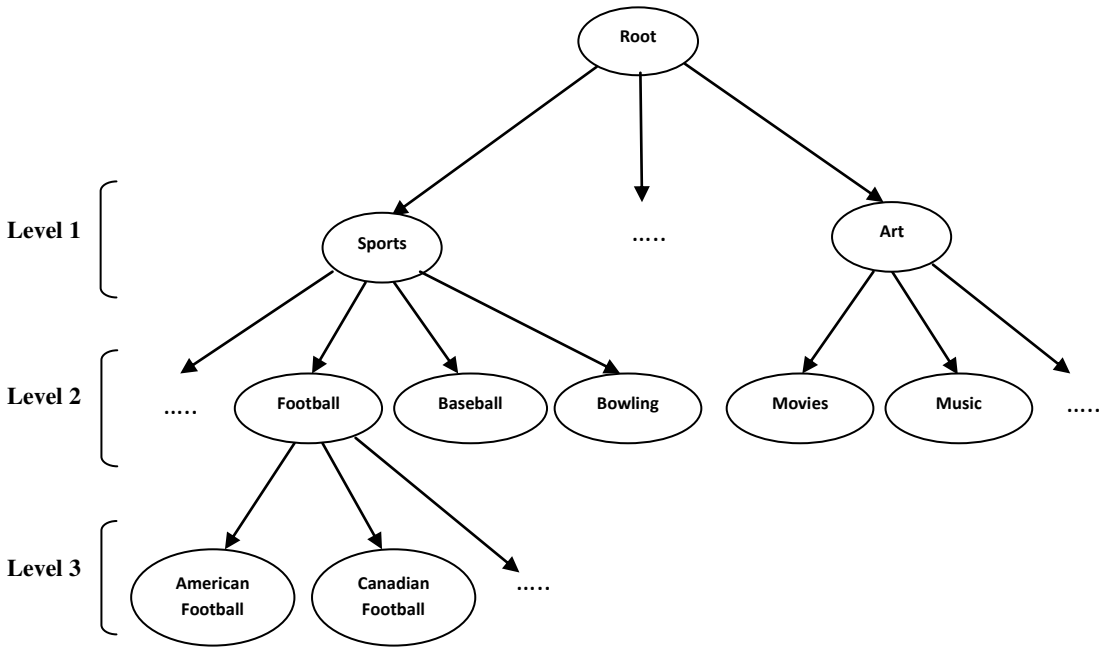


Figure 5-6: Part of the subjects hierarchy

An example of such a hierarchy is the "Open Directory Project DMOZ[26]". In this thesis, we have limited ourselves to the first three levels of the DMOZ hierarchy (without considering the root). This provides a satisfying level of specialization so that the user can precisely determine from the available subjects his/her exact interests. Indeed, the first level contains 16 subjects, the second level contains nearly 600 subjects, and the third level contains nearly 5000 subjects. The hierarchy in Figure 5-6 is only a small portion of the total hierarchy used in the DMOZ project.

Each subject (a node in the hierarchy) has a unique identifier. This identifier is used later when the node wants to disseminate a Potential values (section 5.5.4). It should be noted that some subject nodes in the DMOZ hierarchy have more than one parent; that means the hierarchy is a graph and not a tree[27]. Figure 5-7 shows some examples of subjects that have more than one ancestor. In "A" "Healthcare" is a child for the "Health" and the "Business" subjects. In "B" the "Antiques" subject has both the "Art" and the "Recreation" subjects as parents.

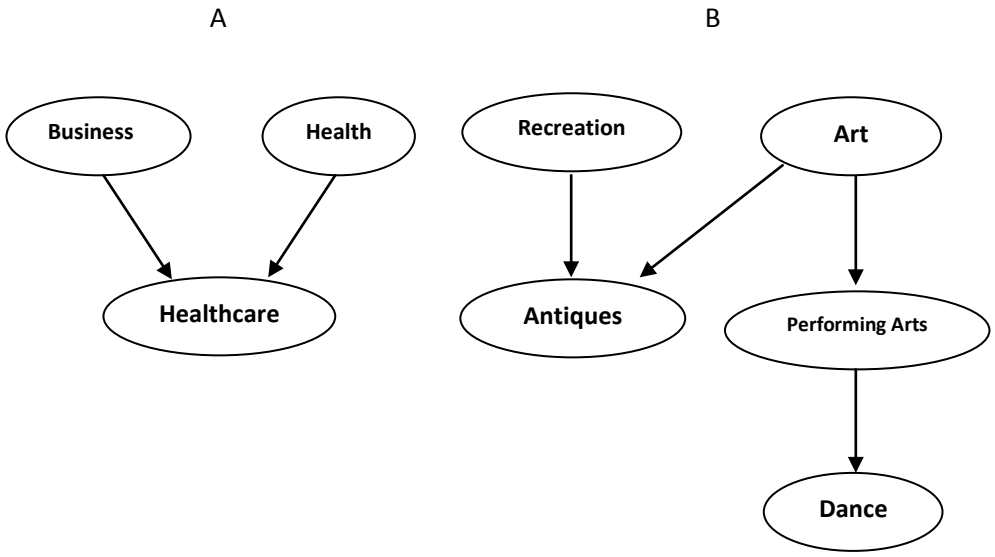


Figure 5-7: Examples of subjects with multiple ancestors

[26] www.dmoz.org: The Open Directory project is a human-edited directory, constructed and maintained by a vast and global community of volunteer editors. It is used to classify Web content. Its name is DMOZ directory.

[27] Indeed, the notion parent/child is used here instead of descendant/ascendant usually used in graph theory, because the subjects are related with a "is-a" relation

The hierarchy is an input for our system; we assume that all the participating devices in the network use the same hierarchy to describe the user's interests. It is supposed to be fixed i.e. it cannot change during the execution of CReaM.

We have chosen to represent a subject S using three fields (Figure 5-8). Some of them have been added for the purpose of calculating the potential values of each subject as we will explain later in this chapter.

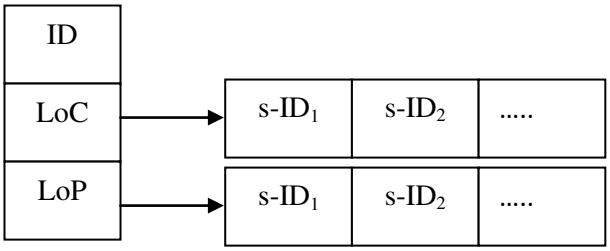


Figure 5-8: Representation of a subject

- ID: unique identifier for the subject S.
- LoC: list of the subject's children. This field is useful to calculate the potential value for the subject S based on the potentials of his children. This field is empty for all leaf nodes.
- LoP: list of the subject's parents (ancestors). As for the LoC, this field is used to traverse the hierarchy from bottom to up when selecting the candidate replica holders (see Algorithm 5-1). This field is empty for all subjects in the first level from the top of the hierarchy.

5.5.2. Node Profile Propagation

Nodes need to exchange their profile (see definition 5-2) in order to calculate the Personal Potentials locally on each node. This exchange must be periodical so that the node keeps an up-to-date list of the interests of its direct neighbors. Since our model aims at limiting the overhead imposed by the replication, we need to avoid creating too many messages to propagate the profiles. Thus, we propose to exploit the

messages that circulate in the network for other purposes (like the messages sent by the routing protocols) to diffuse the profiles.

Actually, the routing protocols use, for the neighborhood detection, periodic exchange of small messages called Hello messages. Hello message is diffused to the one-hop neighbors only; when a node receives a Hello from a neighbor, it creates or refreshes the routing table entry of the corresponding neighbor. Failure to receive any hello from a node for a predefined time interval indicates that this node is no longer within the transmission range, and that the connectivity has been lost.

Since the exchange of the Hello messages is periodic, our system can use it to diffuse the profiles.

In fact, some points must be considered to lead this solution to success: Hello messages are sent periodically to maintain the connections between the nodes, and to maintain the requests paths. They are sent with very small time slot between two successive messages (the recommended interval value between two messages is one second [21]); they must be small enough not to impose overhead on the network. For example, the size of the Hello message in the AODV protocol is 20 bytes only (Figure 5-9). So if CReaM's information to be included, the total size must be kept small.

To limit the size of the new field added to the Hello message, the subjects that need to

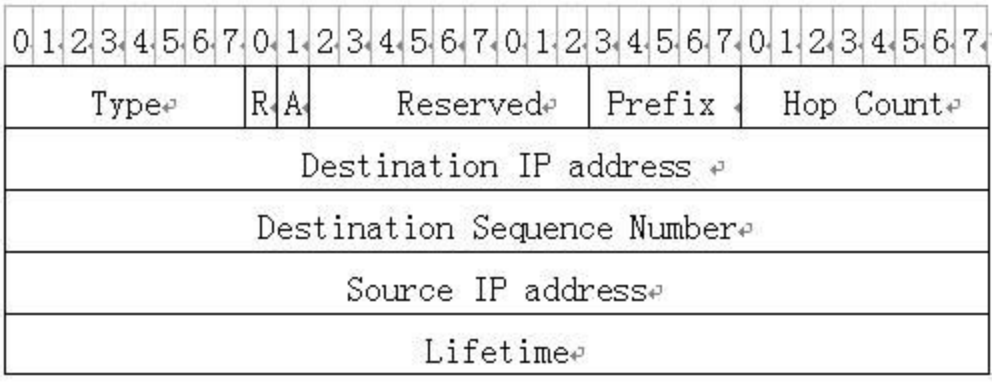


Figure 5-9: Hello message used by AODV

be propagated are divided and included in consecutive messages so that the size limit remains respected. Since the total number of subjects in the first three levels is almost 6000 subjects, 2 bytes are needed to represent the subject identifier. Consequently, we propose to add 2 subjects (4 bytes) only in each Hello message which is equivalent to 1/5 of the original size of a AODV Hello message.

5.5.3. Personal Potentials Calculation

The calculation of the Potential for a subject S depends on the type of the subject. If it corresponds to a leaf node in the hierarchy H(B, L), the calculation of the potential value depends on the number of direct neighbors interested in this subject. However, if the subject corresponds to an internal node in H(B, L) then the potential depends on the number of direct neighbors interested in that subject **or** in any child subject.

To this purpose, we define the notion of Global Community of a subject:

Definition 5-8: Global Community of a Subject

Let $S \in B$ be a subject and $N \in V$ be a node. We define the **Global Community** of S around N, noted GC(S, N) as follows:

- If S is a leaf node in H(B, L): GC(S, N) is equal to the Community of S around N (see 5.3).

$$\forall S \in LF(H), \forall N \in V: \; GC(S,N) = C(S,N)$$

- If S is an internal node of H(B, L): GC(S, N) is recursively defined as the set resulting from the union of the Community of S with the sets of the Global *Community of S's children.* Formally, l*et Ch(S) denote the set of S's children:*

  $\forall S \notin LF(H), \forall N \in V$:

$$GC(S,N) \; = \; C(S,N) \cup \bigcup_{S_k \in Ch(S)} GC(S_k,N)$$

Definition 5-4 of the potential of a node for a subject given in section 5.3, does not consider the hierarchy of the subjects. Thus, based on the definition of the Global Community, we propose to now identify the potential of the node N for the subject S as the cardinality of the set GC(S, N)

$$P_S(N) = Card(GC(S,N))$$

5.5.3.1. Example of Calculation of Personal Potentials

Figure 5-10 outlines an example of a network. The profile of each node is shown next to the node's name.

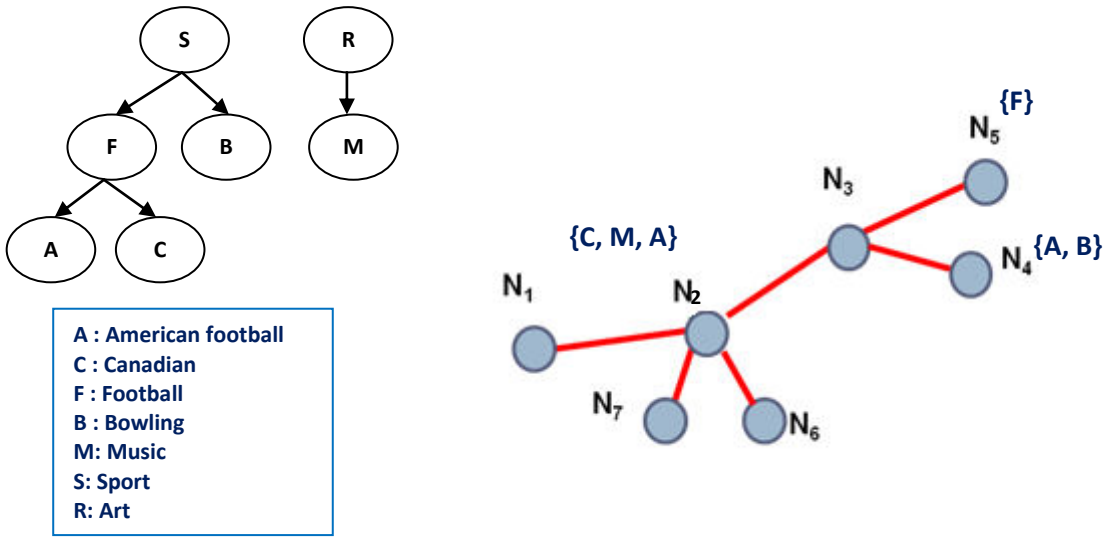


Figure 5-10: Example of Personal Potentials calculation

We want to compute the potential of N_3 for all subjects:

1- Calculate the Potential values of the leaf subjects: LF(H) = {A, C, B, M}

S_j	$GC(S_j, N_3)$		$P_{Sj}(N_3)$
A	$GC(A, N_3) = C(A, N_3)$	$\{N_4, N_2\}$	$P_A(N_3)=2$
C	$GC(C, N_3) = C(C, N_3)$	$\{N_2\}$	$P_C(N_3)=1$
B	$GC(B, N_3) = C(B, N_3)$	$\{N_4\}$	$P_B(N_3)=1$
M	$GC(M, N_3) = C(M, N_3)$	$\{N_2\}$	$P_M(N_3)=1$

2- Calculate the Potential values of the internal subjects:

S_j	$GC(S_j, N_3)$		$P_{Sj}(N_3)$
F	$GC(A,N_3) \cup GC(C,N_3) \cup C(F, N_3)$	$\{N_4, N_2\} \cup \{N_2\} \cup \{N_5\}$	$P_F(N_3)=3$
S	$GC(F,N_3) \cup GC(B,N_3) \cup C(S,N_3)$	$\{N_4, N_2, N_5\} \cup \{N_4\} \cup \{\}$	$P_S(N_3)=3$
R	$GC(M, N_3) \cup C(R, N_3)$	$\{N_2\} \cup \{\}$	$P_R(N_3)=1$

Periodical Calculation: Note that, in a mobile environment, the nodes are mobile entities with different velocities, which means that the connections between them change frequently and, therefore, the Potential Values change too. Hence, the Potential are calculated periodically, i.e., for each period T, the node calculates its potential for each subject and updates its value if necessary. The Personal Potentials list maintains the potentials of the node for the different subjects: PrsP(N)=$\{(S_1, P_{S_1}(N) = V_1), (S_2, P_{S_2}(N) = V_2), \dots, (S_k, P_{S_k}(N) = V_k)\}$: where k is the number of subjects. (cf. definition of PrsP(N) in section 5.3).

5.5.3.2. Algorithms

To calculate the Personal Potentials, we must construct the community of all subjects around the node, and then periodically calculate the potentials. In order to present the algorithm personalPotentialsCalulation, we first detail two algorithms used by this former: interestsExtractiont that extracts the interests of neighbor nodes and calculates the community of each subject around the node; potentialCalculation that calculates the potential for each subject according to the community sets.

a. (Algorithm 5-5) **interestsExtraction**

The algorithm makes use of the following data types:

- **HelloMessage**: object containing all the information included in the Hello message diffused between direct neighbors. The method getUserProfile() is defined to return the subjects that interest the message owner.
- **Cmm**: is a set containing the community of a subject around the node (definition 5-3) (type: list of Integer (nodeIDs)). The following methods are defined for this type:
    1. **addElement**(nodeID): adds a nodeID to the set Cmm in case nodeID has not been added before.
    2. Cmm: **union**(listOfNodeIDs): adds the ids listed in ListOfNodeIDs to the set Cmm.
    3. Integer: **getNumberOfElements():** returns the number of elements of the set Cmm.
    4. **deleteElements()**: delete all the elements of the set (re-initialization).

This algorithm is executed each time the node receives a Hello message from a direct neighbor. The algorithm takes as input an object helloMsg of type HelloMessage. It extracts the subjects that interest the sender by calling getUserProfile() (line 4). For each subject S_j (line 7-10), the algorithm adds the sender ID to the table CmmList[j] using addElement().

Algorithm 5-5

Algorithm interestsExtraction

Input:
helloMsg: the received Hello message containing a profile (type: HelloMessage)

Input/Output

CmmList: array that stores the Community for all subjects (type: array of Cmm)
//ex: for subject S_1 CmmList[1]={N_1, N_2, N_4, N_7}

Variables
LoS: list of user interests (type: list of Subject)
nodeID: node identifier (type: Integer)

Begin
1. nodeID = helloMsg.getSender( ) //get the identifier of sender node
2.
3. //get the list of subjects that interest the user from his profile
4. LoS = helloMsg.getUserProfile( )
5.
6. //add the user to the community of each subject *in "LoS"*
7. **For** (each subject $S \in LoS$) **do**
8. //the function addElement insert the nodeID in case it does not appear in the list
9. CmmList[S.getID()].addElement(nodeID) //CmmList[i] is the community of S_i
10. **End for**
11. **Return** CmmList

End

b. (Algorithm 5-6) **potentialCalculation**

The calculation of the potential for an internal subject uses the potentials of its children. Thus, to compute the potentials, we need to go through the hierarchy of subjects in an inverse breadth-first traversal order[28], i.e., level by level starting from the last level. To simplify the process, we can apply a simple (not an inverse) breadth-first traversal with a complexity never more than O(|V|) where |V| is the cardinality of the set of vertices. Then the inverse order of the visited subjects gives the needed order[29]. Since the hierarchy is static, this process can be done only once and the traversal order can be saved in a global variable.

We note inverseBreadthFirstNext(S) the function which returns the next subject to S in inverse breadth-first order. To initialize the traversal, we also define the function lastSubject() which returns the "last subject", i.e. the first one in inverse breadth-first order.

[28] For more details see [80] [47].

[29] That is, the order of the inverse breadth first traversal

Algorithm 5-6

Algorithm potentialCalculation

/* This algorithm is called periodically (after each time interval t) by calculatePersonalPotentials. It returns two variables PrsP and prvPrsP.
PrsP: is an array containing the Personal Potentials of the node (at current time t_j), while prvPrsP is an array used to keep the previous Personal Potentials (calculated at time t_{j-1}). Actually, it is necessary to keep the Personal Potentials of the previous time interval prvPrsP because they are used in the dissemination process. (cf. section 5.6.3)*/

Input

CmmList: an array of the Community of all subjects (type: array of Cmm)
//CmmList[i] = community of S_i around the node

Input/Output

PrsP: array representing Personal Potentials of the node at the moment of executing the algorithm (type: array of Integer) //PrsP[i]=(P_{si}(N)) at time t_j

prvPrsP: array to keep the previous Personal Potentials of the node (type: array of Integer) //prvPrsP[i]=(Ps_i(N)) at time t_{j-1}

Variables

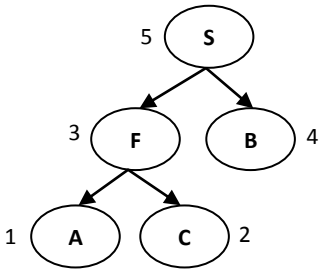
S: temporary subject (type: Subject)
H: subjects hierarchy
GCList: array representing the Global Community of all subjects (type: array of Cmm) //GCList[i] =global community of S_i around the node

Begin

```
1.  initialize GCList to an empty array
2.  S = H.lastSubject() //Initialization of S to the last subject in the hierarchy
3.  GCList [S.getID()] = CmmList[S.getID()] //the last subject is a leaf by construction of H
4.  While ( (S = H.inverseBreadthFirstNext(S)) != Root ) do
5.          GCList [S.getID()] = CmmList[S.getID()]
6.          //if S is not a leave, we add the set of nodes interesting in S's children
7.          For (each S' in Ch(S)) do //Ch(S) is the set of S's children
8.                  GCList[S.getID()].union( GCList[S'.getID()] )
9.          End for
10. End while
11.
12. //calculate the Personal Potentials at the moment as the cardinality of GC, after saving the
13. Potentials of the last period in prvPrsP */
14. prvPrsP = PrsP
15. For (each S ∈ B) do //B is the set of subjects
16.         PrsP[S.getID()] = GCList[S.getID()].getNumberOfElements()
17. End for
End
```

To illustrate the result of the execution of Algorithm 5-6 on the previous example (section 5.5.2.1), at the end of the algorithm we will get the following arrays (see Figure 5.2.2 below): CmmList (output of Algorithm 5-5), GCList and PrsP. Note that

the numbers next to the subjects names in the hierarchy indicate the traversal order of the hierarchy.



CmmList

1-A	N_4, N_2
2-C	N_2
3-F	N_5
4-B	N_4
5-S	

GCList

A	N_4, N_2
C	N_2
F	N_4, N_2, N_5
B	N_4
S	N_4,N_2, N_5

PrsP

A	2
C	1
F	3
B	1
S	3

Figure 5-11: Illustrative example of the *potentialCalculation* algorithm

c. (Algorithm 5-7) **calculatePersonalPotentials**

This algorithm starts when the node connects to the network, and stops after its disconnection. It periodically calculates the Personal Potentials of the node by calling the function potentialCalculation (line 12). Between two computations, the algorithm monitors the incoming Hello messages and extracts from them the set of subjects that interest the direct neighbors by executing the function interestsExtraction (lines 6→10).

Algorithm 5-7

Algorithm: calculatePersonalPotentials

Intput

timePeriod: time between two calculations of the Personal Potentials

Input/Output

PrsP: array of Personal Potentials of the node at the time of execution of the algorithm (type: array of Integer) //PrsP[i]=potential of the node for the subject S_i (P_{si}(N))
prvPrsP: array to keep the previous Personal Potentials of the node (type: array of Integer)

Variable
CmmList: array of Cmm, where each element represents the Community for one subject.

Begin
1. **Do**
2. //delete the existing nodeIDs to restart the potential calculation for the new period
3. **For** (each S ∈ B) **do**
4. CmmList[S.getID()].deleteElements()
5. **End for**
6. **Do**
7. Wait until receiving a Hello Message helloMsg;
8. //extract *the interests from the hello message and modify the subjects'* hierarchy
9. CmmList = interestsExtraction[30] (helloMsg , CmmList);
10. **Until** (t = timePeriod)
11. //calculate the Personal Potentials
12. potentialCalculation[31](CmmList , PrsP, prvPrsP);
13. t=0;
14. **Until** (false)

End

After presenting the calculation of the Personal Potentials, we present in the next section, how these potential, in additional to the External Potentials, are disseminated to other nodes in the network.

5.5.4. Potential Values Dissemination

During the replica holder selection, the node uses its External Potentials to select the node with the highest potential for the replica's subjects. According to Definition (5-6), the External Potentials on node N are the Personal Potentials of other nodes received by N. Thus each node calculates its Personal Potentials (Algorithm 5-7), and disseminate them. In addition, since the Personal Potentials are re-calculated periodically, the dissemination process is also performed periodically in order to keep the External Potentials up to date.

[30] Algorithm 5-5

[31] Algorithm 5-6

In this dissemination process, each node sends its Personal Potentials to its direct neighbors who in turn propagate them to their own direct neighbors. The diffusion is done with a preset TTL (Time-To-Live) value, such that the propagation stops when the TTL is reached.

Example: in Figure 5-12, node N_3 calculates the potential value for S_1: $P_{S_1}(N_3)$ and disseminates it to N_2, N_4, and N_5. Also, it receives from N_5 the $P_{S_1}(N_5)$, and sends it to N_2 and N_4.

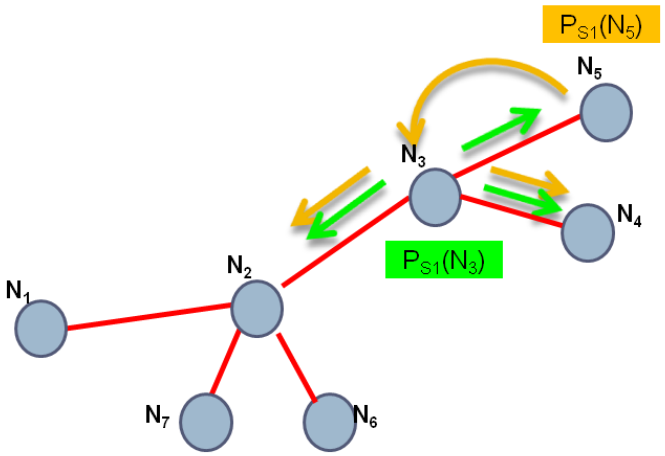


Figure 5-12: Potential dissemination

Definitely, exchanging the different potential values imposes some overhead on the network. For this reason, some optimizations are proposed in the next section to decrease the overhead caused by these periodic exchanges.

5.5.5. Optimizations

As the proposed approach above requires keeping on each connected node the potential values for all possible subjects of all connected nodes, it can be a costly solution in terms of storage and communication; it consumes storage space for saving the Potential Values and it consumes the bandwidth for exchanging those values

between the neighbor nodes. However, some optimizations can be done in order to decrease these overhead.

Indeed, the node keeps information about the potential for all subjects, stored in a table similar to Table 5-1 below. This table may become large if the number of subjects is high, or if the density of the MANET is high, i.e. the number of connected nodes is high.

Table 5-1: External Potentials on a node

	$\mathbf{N_1}$	$\mathbf{N_2}$	$\mathbf{N_3}$	…	$\mathbf{N_{|V|}}$
$\mathbf{S_1}$	$P_{S_1}(N_1)$	$P_{S_1}(N_2)$	$P_{S_1}(N_3)$	…	$P_{S_1}(N_{\|V\|})$
$\mathbf{S_2}$	$P_{S_2}(N_1)$	$P_{S_2}(N_2)$	$P_{S_2}(N_3)$	…	$P_{S_2}(N_{\|V\|})$
…					
$\mathbf{S_{\|B\|}}$	$P_{S_{\|B\|}}(N_1)$	$P_{S_{\|B\|}}(N_2)$	$P_{S_{\|B\|}}(N_3)$		$P_{S_{\|B\|}}(N_{\|V\|})$

In addition, the Potential Values should be disseminated on the network; so each node N_i reacts cooperatively and disseminates to its direct neighbors the potential received from other nodes $(P_{S_1}(N_1), P_{S_1}(N_2), \dots P_{S_{|B|}}(N_{|V|}))$ in addition to its Personal Potentials $(P_{S_1}(N_i), \dots P_{S_{|B|}}(N_i))$.

However, not all of these values are useful for selecting the candidate replica holder; thus, if the node keeps only the values that affect and improve the replica placement process, we can decrease the storage space needed to store the potentials, and consequently decrease the exchanged data needed to communicate the potentials. Below we list some observations that help in improving the proposed solution.

- The node does not need all the External Potentials values in the table to select a replica holder. When the replication process is triggered, the node must select a set of candidate replica holders depending on the number of replicas that it wants to create. However, the number of replicas is limited to X replicas each time where X ≤ number of connected nodes. Hence, the node needs only to keep the X highest External Potentials for each subject.

- Low Potential Values are not useful; indeed, the Potential indicates the probability for the node to be an adequate replica holder for a corresponding subject; so, if the value is low, it means that the node is connected to a small number of nodes interested in the subject. Consequently placing a replica on it would not give any additional improvement for the whole system. For example, selecting a candidate node that has only one direct interested neighbor is similar in terms of load balancing to select a random node. Thus, keeping and diffusing such information is useless to the replica placement process; it costs storage space, network bandwidth, and computing power. Therefore, CReaM only saves potentials if above a threshold (ε).

Example: at the storage level, if we apply both improvements, the node will have a table similar to Table 5-2 instead of the table presented above if the maximum number of Potential values stored for each subject X=3 and the threshold ε of the significant Potential is equal to 4.

Table 5-2: Optimized External Potentials table on node

Subject	Potentials
S_1	$P_{S_1}(N_1) = 9$
	$P_{S_1}(N_3) = 9$
	$P_{S_1}(N_4) = 8$
S_2	No Potentials available
S_3	$P_{S_3}(N_2) = 5$
	$P_{S_3}(N_4) = 4$
…	
$S_{\|B\|}$	$P_{S_{\|B\|}}(N_3) = 6$

Thus the number of External Potentials on a node is less than |B|*X while without the proposed optimisation the number may reach |B|*|V|.

Moreover, applying both improvements significantly decreases the communication cost:

1- The node diffuses a Personal Potential value to its direct neighbors only when this value becomes higher than the threshold ε or this value was higher than ε and it was later modified (**positively** or **negatively**).

   It is important to mention that a Potential Value could be diffused even if it is **not** significant. This case occurs, when a node diffuses to its neighbor a change in its situation. The Potential for a subject could change and become not significant because of the mobility of the node or of the mobility of their neighbors. So the node must propagate such changes also.

2- Furthermore, when it receives a new External Potential $P_S(N)$, a node now applies the following strategy:

- If the table contains already $P_S(N)$; the existing value is either replaced by the new one if the latter is still a significant Potential Value or deleted otherwise. (Note that the new Ps(N) could be equal to the existing value, in this case the value remains in the table and we consider that the value is replaced).

- If the table includes less than X Potential values for the subject S, it changes the table by adding the coming value.

- If the table contains X potential values for S, and the new coming value(s) is higher than at least one value, the node replaces the smallest value in the table by the new one.

- In any other case, the new value is ignored.

In case the table of the External Potentials is changed and the Time-To-Live (TTL) is not yet expired, the node reacts cooperatively and retransmits the new Potential Value to direct neighbors. The algorithm of disseminating Potential Values is presented later in section 5.6.2 after presenting the PInfo message used to communicate the Potentials.

5.6. PInfo Message

In this section, we introduce a new type of message named PInfo defined to disseminate the potentials. PInfo messages are periodically diffused by each node to its **direct neighbors** only. It should be small enough to avoid imposing an important overhead on the network. It consists of the following fields as shown in Figure 5-13:

ID
Personal Potentials **PP**
External Potentials **EP**

Figure 5-13: PInfo message

- ID: identifier of the origin node.
- The **PP** field contains the Personal Potentials of the origin node. It consists of pairs that have the form (V, SL) where V is the potential value, and SL is the list of subjects for which the node has a potential equal to V. For example, a node N with 3 Personal Potentials $P_{S_1}(N) = P_{S_3}(N) = V_1$ and $P_{S_2}(N) = V_2$ will diffuse a PP = $(V_1, \{S_1, S_3\}), (V_2, \{S_2\})$.
- The **EP** field contains the External Potentials. Each value in this part is represented as a pair of the form (PL, TTL) where PL is list of Potential values and TTL is their corresponding Time-To-Live. Each element in the PL list is a triplet of the form (subject S, node N, potential value V). For example, if three External Potentials need to be disseminated: $P_{S_1}(N_2) = 5, P_{S_2}(N_5) = 8$ with TTL=2 and $P_{S_1}(N_3) = 9$ with TTL=1, the PInfo message will contain the following EP: $(\{(S_1, N_2, 5), (S_2, N_5, 8)\}, 2), (\{(S_1, N_3, 9)\}, 1)$.

  We recall that if the table of the External Potentials has not changed between two successive PInfo messages, this part is empty.

In the following section, we present the different formats of the PInfo message that can be used to decrease the communication cost and optimize the use of the bandwidth. Then, we present the algorithms of (1) maintaining the External Potentials table on the node using the received PInfo and (2) preparing the PInfo message.

5.6.1. Formats of the PInfo Message

In fact, several improvements can be applied to reduce the size of the PInfo message; we developed a method to disseminate only the difference from the previous PInfo message. Indeed, the PP is the field that is repeated from a message to another; the message size can be reduced by including in the PP only the Personal Potentials that have been changed (positively or negatively) since the last message.

We therefore introduce two formats of PInfo message: the first format is the Complete Form (PInfo-CF) presented above, and the second format is the Differential Form (PInfo-DF) which contains a compressed PP with only the Potentials that are significant and are not included in the previous PInfo message or that have changed since this message was transmitted.

The node sends a PInfo-CF message when it connects to the network; then it periodically sends a PInfo-DF message during k time intervals, then a PInfo-CF message is sent once again. Figure 5-14-A shows a PInfo-CF for a node N_2 at time t_i that has 3 Personal Potentials higher than the threshold ε=4: ($P_{S_1}(N_2) = P_{S_3}(N_2) = 6, P_{S_2}(N_2) = 5$) and that has in its External Potentials table 3 value changes since t_{i-1}: ($P_{S_2}(N_4) = 4, P_{S_9}(N_{15}) = 8, P_{S_{12}}(N_1) = 7$). At t_{i+1}, N_2 sends the PInfo-DF message shown in Figure 5-14-B that contains in the PP part: a new Potential value for the subject S_4, the change in the Potential for S_2 (it becomes 0). The EP part contains the changes in the External Potential of N_{15} for S_9 and one new External Potentials received since t_i. In case the PInfo-DF message was not used at t_{i+1}, the node would send the PInfo-CF message shown in Figure 5-14:C. (Note: if N_2 receives since t_i the same values $P_{S_2}(N_4) = 4$ or $P_{S_{12}}(N_1) = 7$, the PInfo-DF message (Figure 5-14:B) and the PInfo-CF message (Figure 5-14:C) at t_{i+1} will contain these values in their EP part).

A

Originator IP address
(6, {S_1, S_3}), (5, {S_2})
({(S_2, N_4, 4)}, 1), ({(S_9, N_{15}, 8), (S_{12}, N_1, 7)}, 3)

B

Originator IP address
(7, {S_4}), (0, {S_2})
({(S_9, N_{15}, 5)}, 4), ({(S_2, N_{10}, 5)}, 2)

C

Originator IP address
(6, {S_1, S_3}), (0, {S_2}), (7, {S_4})
({(S_9, N_{15}, 5)}, 4), ({(S_2, N_{10}, 5)}, 2)

Figure 5-14 PInfo-CF and PInfo-DF messages

5.6.2. Algorithm 5-8: maintaining the External Potentials Table

Two algorithms are needed in order to maintain the External Potentials table. A first algorithm is executed each time a PInfo message is received. A second algorithm is executed after each time interval in order to clean the invalid potential values.

- **Integration of a new PInfo message:**

Algorithm 5-8 below is executed after receiving a PInfo message. It takes as input this PInfo message and it returns the modified External Potentials table ExtPoTable and a list of External Potentials to be sent to the direct neighbors in the next PInfo message.

The first section of the algorithm (lines 1→7) transforms the PP field of the PInfo message to take the form of the EP in order to handle all potentials in the same way. The second section (lines 9→42) modifies the External Potentials table if necessary: it replaces the existing potentials with new ones if the table already contains the potential of a node with a different value; it adds new values if the table has less than

X values for the considered subject. Otherwise, it replaces values such that the highest values are kept in the table. The last section of the function (lines 44→46) adds the table updates to a list in order to propagate it in the next PInfo message. This list is called toSendList and it contains the Potentials with a non-expired TTL.

This algorithm makes use of the data types defined for Algorithm 5-1: PotOrderedList, and Potential.

Algorithm 5-8

Algorithm updateExtPotTable
//This algorithm is executed each time the node receives a PInfo message.

Input
PInfo: the received message

Inputs/Outputs
ExtPoTable: table containing the External Potentials of the node (type: array of PotOrderedList = list of Potential objects for the same subject in a decreasing order of Potential Values; it contains maximum X objects)
toSendList: list of External Potentials that must be forwarded to direct neighbors in the next PInfo

Variables
nodeID: temporary variable (type: Integer)
initTTL: the initial value of the TTL
PList: temporary list containing the pairs of the EP part: (PL, TTL) where PL=(S, N, V, T)
modification: Boolean
potential: temporary variable (type: Potential) //contains 4 attributes (S, N, V, timestamp)
timestamp: variable containing the current time.

Begin
1. //make the Personal Potentials part (PP) from the PInfo similar to the EP part
2. nodeID = PInfo.getSenderID( )
3. **For** (each $(V, S) \in PP$) **do**
4. pair = Make a pair from the form ((S, nodeID, V), initTTL)
5. PList.add(pair)
6. **End for**
7. **Add** pairs in the EP part of PInfo to the PList[32]
8.
9. modification = false
10. **For** (each pair ((S,N,V), TTL) in PList) **do**

[32] If the EP contains multiple Potential values with the same TTL (e.g. ({(S_1, N_1, V_1), (S_2, N_2, V_2)}, TTL)), multiple pairs are added to the PList: ((S_1, N_1, V_1),TTL) and ((S_2, N_2, V_2), TTL)

```
11.          potential.setNodeId(N)
12.          potential.setValue(V)
13.          potential.setSubject(S)
14.          potential.setTimestamp ( timestamp )
15.
16.          /* if a Potential with the same nodeID and the same subject exists before (with the
17.             same or with another value) → delete the object if the new value is not significant
18.             otherwise replace the value with the new one */
19.          If ( (S,N) exists in ExtPoTable[S.getID()] ) then
20.                  If  (V < threshold ε) then
21.                          ExtPoTable[S.getID()].deleteElement (potential)
22.                  Else
23.                          ExtPoTable[S.getID()].replace(potential)
24.                  End if
25.                  modification = true
26.          Else
27.                  // if a new potential value is less than ε the algorithm ignores it
28.                  If (V > threshold ε) then[33]
29.                     If (ExtPoTable[S.getID()].getNumberOfElements() < X ) then
30.                             //X is the max number of External Potentials for each subject
31.                             ExtPoTable[S.getID()].addElement( potential )
32.                             modification = true
33.                     Else
34.                             //compare the new value V with existing values for S in the table
35.                        If (V higher than one potential in ExtPoTable[S.getID()] ) then
36.                                //Replace the smallest potential
37.                                ExtPoTable[S.getID()].replaceTheSmallest(potential)
38.                                modification = true
39.                         End if
40.                     End if
41.                  End if
42.          End if
43.          //if a modification occurred, the value is added to a list of modification to be sent later
44.          If ( modification == true ) and (TTL!=0) then
45.                  add or replace ((S,N,V), TTL-1) in the list toSendList
46.          End if
47.  End for
End
```

- **Cleaning invalid values**

Since the nodes are mobile, and they connect and disconnect frequently, the potential values are calculated and disseminated periodically. The previous algorithm maintains

[33] Usually, the node does not receive a Potential value that is less than ε unless it has it in the External Potentials table with a value higher than ε. However, this case may occur because of the mobility of the nodes; in this case, the insignificant value will be ignored.

the table updated based on the new values that are propagated using the PInfo messages. Each node sends a PInfo-CF message periodically even if no change has occurred to its Personal Potentials; the periodical exchange of the complete form PInfo-CF helps other nodes to detect that the sender still exists in the network even if its Personal Potentials have not changed. Note that in the Algorithm 5-8 **updateExtPotTable** when the PInfo message contains a potential value that already exists in the External Potentials table, the algorithm considers that a change has occurred on the table (lines 19-25) and propagates this value in a further PInfo message. So when a PInfo message is received, the timestamp of all the potentials listed in the message are updated.

However, the External Potentials of the disconnected nodes must be deleted from the table to free the place for other significant potentials. To this end, Algorithm 5-9 is responsible for cleaning the External Potentials table periodically. The cleaning process includes deleting the values that have not been modified since **J** time intervals.

Algorithm 5-9

Algorithm cleaningExtPoTable

Input

ExtPoTable: table containing the External Potentials of the node (type: array of PotOrderedList (list of Potentials for one subject in decreasing order)

Output

Delete from the ExtPoTable the potentials that (the current time - their timestamp) ≥ **J**

Variables

potential: temporary variable (type: Potential)
Tt: the current time instant

Begin

```
1.  For ( each subject S in B) do
2.      For ( each potential for S in ExtPoTable[S.getID()]) do
3.          If (Tt - potential.getTimestamp() ≥ J) then
4.              ExtPoTable[S.getID()].deleteElement(potential)
5.          End if
6.      End for
7.  End for
End
```

5.6.3. Algorithm 5-10: Prepare a PInfo Message

According to the time, a PInfo-CF or PInfo-DF message is generated: each k time intervals, the algorithm prepares a PInfo-CF message. Otherwise, a PInfo-DF message is prepared. This algorithm takes four parameters as inputs: the first parameter "toSendList" is the list of External Potentials that have changed since the last PInfo message; the second parameter is the list of Personal Potentials "*PrsP[]"* at the time of sending the PInfo message (recall: PrsP[] is the output of the algorithm "potentialCalculation"[34]); the third parameter is the prvPrsP[] i.e., the Personal Potentials of the node at the moment of sending the previous PInfo message. Finally, the last parameter is a boolean parameter completeForm to indicate that the complete form is to be sent. The output of the function is a new PInfo message that will be sent to the direct neighbors of the node.

The algorithm prepares first the PP part then the EP part. For the PP part, if the complete form is prepared, the node adds all significant Potential values (i.e., value > ε), and those that became insignificant during the last period. If the PInfo-DF is prepared, it puts into the PP part: (1) new significant potentials (2) significant potentials changed but still significant (3) significant potentials that became insignificant. To determine the PP part, the algorithm compares the values of the arrays PrsP and prvPrsP to detect any change that has occurred during last period.

For the EP part, the algorithm includes the changes that have occurred on the External Potentials table during the last period which are listed in the toSendList parameter.

Algorithm 5-10

Algorithm preparePInfo

Input

toSendList: this list contains the External Potentials that must be disseminated. It is the output of updateExtPoTable (Algorithm 5-8).

PrsP: array containing the Personal Potentials of the node at the time of preparing the PInfo with PrsP[i] = $P_{Si}(N)$. It is modified by potentialCalculation

[34] See Algorithm 5-6.

(Algorithm 5-6).
prvPrsP: array containing the Personal Potentials of the node at the time of sending the previous PInfo. It is modified by potentialCalculation (Algorithm 5-6)
completeForm: Boolean, takes true if the PInfo that is to be sent is a PInfo-CF message

Output
msg: message of PInfo-CF or PInfo-DF form

Begin

```
1.  msg = new PInfo message
2.  msg.setNodeID( this.getNodeID());    //the current node ID is the owner of the message
3.
4.  //The PP part
5.  While ( i < |B| ) do //|B| is the number of Subjects
6.        /* if the potential value changed since the last PInfo message, it must be sent in this
7.          PInfo message (regardless the form of the PInfo) with only one exception: the value
8.          is less than the threshold in prvPrsP and in PrsP (i.e. it changed but it was less than
9.          the threshold and it still less than the threshold */
10.       If (prvPrsP[i] != PrsP[i]) then
11.             If not(prvPrsP[i] <ε and PrsP[i] <ε) then
12.                   msg.addToPPPart( PrsP[i] )
13.             End if
14.       Else  //no change during last period
15.             //but if it is a PInfo-CF message → send the value if it is significant
16.             If (completeForm and PrsP[i] > ε ) then
17.                   msg.addToPPPart( PrsP[i] )
18.             End if
19.       End if
20. i = i +1
21. End while
22.
23. //The EP part
24. /*add to the PInfo the list of External Potentials that must be diffused. These external
25.  potentials are in the list toSendList and have the form (EP, TTL)  where EP is an external
26.  potential */
27.
28. Assemble the EPs of the with the same TTL //(i.e. with the  form ({EP1, EP2}, TTL))
29. PInfoMsg.addToEPPart (toSendList)
30.
31. Return msg
```

End

To illustrate the behavior of Algorithm 5-10, we consider the example in Figure 5-15. If the threshold ε is equal to 6, the algorithm prepares the following PP part:

- If the message is a PInfo-DF message: The PP part includes: S_2 (has changed and it still higher than ε), S_3 (has become higher than ε), and S_4 (has become less than

ε). The PP part does not include: S_1(has not changed) nor S_6 (has changed but still less than ε) nor S_5 (has not changed and is less than ε).

- If the message is PInfo-CF: the only difference from the PInfo-DF form is that S_1 is now included into the PP part as it is higher than ε.

	prvPrsP
S_1	10
S_2	15
S_3	5
S_4	9
S_5	4
S_6	3

	PrsP
S_1	10
S_2	12
S_3	8
S_4	4
S_5	4
S_6	5

Figure 5-15: Illustrative example of preparation of the PP part of a PInfo message

5.7. Conclusion

The replica placement process of CReaM has been presented in this chapter. It uses the notion of **user interests** and **document topical information** in order to find the best node to place the replica. The idea is that a data item should best be placed on a node with very good connectivity with other nodes that are interested in the data item's topic. The set of nodes that are interested in a topics form a **Community of interest**; the replica placement problem then amounts to finding the "center" of this community.

To resolve this issue, we have introduced the concept of "**Potential**", which represents the number of one-hop neighbors of a node that are interested in a given topic. User interests are defined based on self-declared profiles that the nodes exchange with their direct neighbors using a modified version of the Hello message. Using Potential values, a node can estimate the probability of another node to be in the center of a community of interest.

The replica placement process is divided into two processes: first, the **information gathering process**. We have defined in this process a protocol to enable the information about the most significant potential values to be disseminated in the network with minimal overhead. Second, the **placement process**, where the information collected is then used to select the most promising replica holder. However, as the nodes are autonomous, they can send a replication request to a node that refuses the incoming request. To deal with this problem, we have proposed a replica placement correction process that ensures that a suitable placement is found for each replication request.

SIMULATION AND EXPERIMENTATION RESULTS

In order to experimentally assess the performance of the CReaM replication model described in the previous two chapters, we have implemented it in a prototype middleware. We have then simulated a MANET of nodes running this middleware using the OMNet++ framework. This enabled us to study the behavior of the model along different metrics under varying conditions. This chapter presents the results of this experimental evaluation. In section 6.1, we detail the design of our middleware. In section 6.2, we briefly review existing simulation frameworks and outline the reasons for our choice of OMNet++. The details of the simulation model, the configuration of the environment and the method that we applied to generate a synthetic workload are described in section 6.3. The evaluation of CReaM is covered in section 6.4. Finally, we conclude the chapter in section 6.5

6.1. The Middleware

In this section, we present the middleware that implements the CReaM model. In the remainder of the chapter, we assume that this software component is installed in all devices participating in the MANET (Figure 6-1).

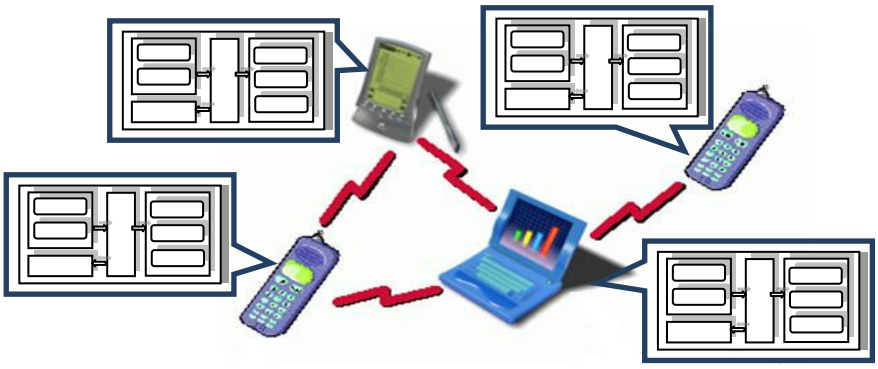


Figure 6-1: The deployment of CReaM

The role of the middleware is to support applications through replication. It allows devices to participate in the replication system defined by CReaM. More precisely, it is in charge of creating replicas on other nodes by preparing and disseminating

replication requests in the network; and of managing the local placement of replicas by deciding whether to accept or reject incoming replication requests. To carry out these tasks, the middleware makes use of services provided by the device's OS.

The architecture of the middleware is outlined in Figure 6-2. It is composed of the following components:

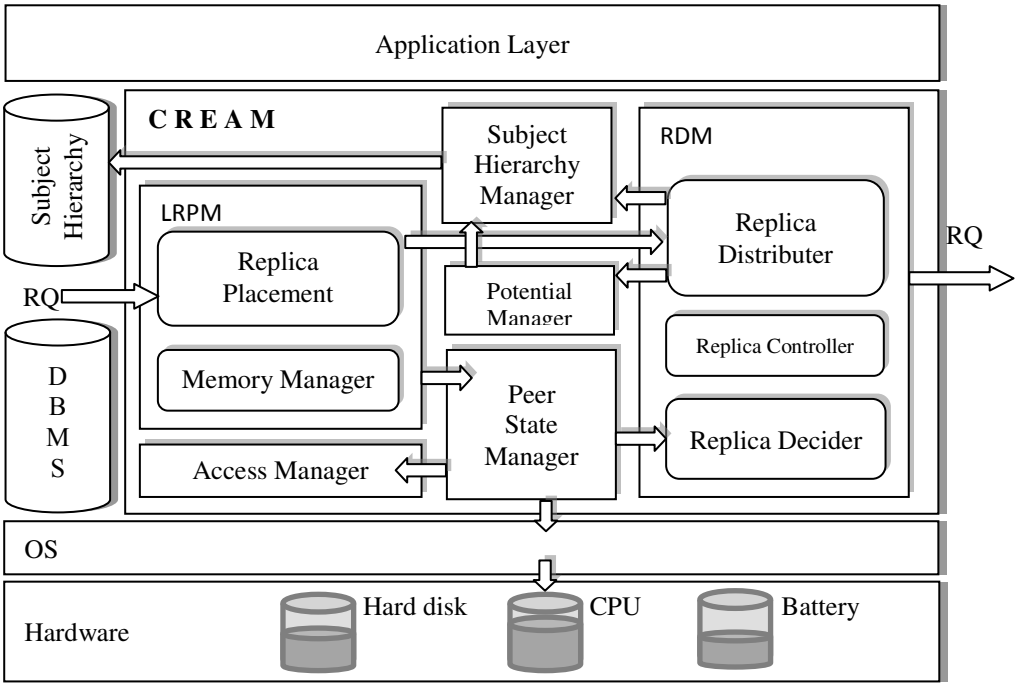


Figure 6-2: The Middleware

- **The Access Manager "AM"**

The AM maintains statistics on the access requests received for each data item present on the node. It keeps track of the values of Access Counter "AC" and Temperature Degree TD (cf. section 4.3.2.1). Note that, the connection between the AM and the request processing engine is not displayed in Figure 6-2.

- **Subject Hierarchy Manager "SHM"**

This component manages and provides access to the hierarchical data structure of subjects introduced in section 5.5.1. In particular, it provides functions to return the

needed subjects, to return the parents/children relations, and to go through the hierarchy (algorithms 5-1, 5-6). By providing an API for accessing the hierarchy, the SHM makes the other components of the architecture independent of the actually used hierarchy, thus improving the generality of the middleware. The SHM also maintains the relations between the data item and the subjects that tag it (Algorithm 5-3).

- **The Potential Manager "PM"**

This component is responsible for calculating the Personal Potentials of the node (algorithm 5-7), managing the External Potentials Table (algorithm 5-8 and 5-9) and sending/receiving the PInfo messages (algorithm 5-10). To calculate the Personal Potentials, it aggregates the information about the interests of the node's neighbors that it receives in the Hello Messages. It regularly calculates the potential value of the node for each subject in the Subject Hierarchy "SH". Furthermore, the PM sends/receives the PInfo messages and keeps the External Potentials table up to date. When replication is required, the component RDis (explained below) communicates with the PM to selects the replica holders based on the Potential values. Note that, the exchange of PInfo messages is not displayed on Figure 6.2.

- **The Peer State Manager "PSM"**

The PSM has a fundamental job in the middleware; it controls the replication process by assigning tasks to other components of the architecture and receiving their outputs. Its first monitors the resource consumption of the device. For this purpose, it communicates with the lower layer (the operating system) and the AM to obtain inputs required to periodically compute the functions presented in section 4.2.2: (1) BL: battery level, (2) BC: battery consumption in the last time interval, (3) SS: used storage space and (4) NoR: number of requests processed during a time interval. Based on the measures returned by these functions and the tolerance thresholds, the PSM determines the status of the node. If the node is not in a stable status, the PSM warns the corresponding component in the middleware to apply the suitable action(s) and to start the replication process if necessary.

- **The Replica Dissemination Manager "RDM"**

Its responsibility is to create replication requests and send them in the network. The RDM consists of 3 subcomponents:

(1). The **Replica Decider "RDec"**: the RDec starts working after being notified by the PSM that the node is not in a stable status. It implements the ECA rules and the action conflict resolution rules (cf. section 4.3.4) to decide which action(s) to take in order to restore the user satisfaction in the consumption of his device's resources. When replication is needed, it prepares the replication request (RQ) (see section 5.4.1.2) and selects the data item (CDI) to be replicated. The RDec then transmits the RQ to the next component, the RC.

(2). The **Replica Controller "RC"**: the role of the RC is to determine the number of replicas that are to be created. Several strategies might be applied here by considering the number of connected nodes, the network bandwidth, the temperature degree of the replica, and many other factors. For the simulation conducted in this thesis, we have opted for a fixed number of replicas, determined experimentally. Once the number of replicas has been determined, the RC adds it to the RQ and transmits it to the next component, the RDis.

(3). The **Replica Distributer "RDis"**: this component is responsible for sending replication requests in the network. It is activated in two cases: when the node needs to replicate a data item and when it receives a replication request that it cannot accept. Indeed, in the latter case, the node has to forward the RQ to another node. The RDis acts slightly differently in each case. In the first case, it first consults the Subject Hierarchy Manager "SHM" to obtain the subjects of the candidate data item in order to search a suitable candidate replica holder(s). It communicates them to the Potential Manager "PM", which returns candidate replica holders selected based on the External Potentials (algorithm 5-1). The number of selected replica holders must match the number of replicas determined by the RC. The RDis then modifies the RQ accordingly and sends it to the network. It follows the dissemination strategy designed for CReaM (cf. section 5.4.2) to ensure that the requested number of replicas is achieved. When forwarding a refused incoming RQ, the RDis applies the same procedure but selects only one candidate replica holder that, to its knowledge, did not

receive the request before. Then, it decreases the Time-to-Live of the replication request by one.

- **The Local Replica Placement Manager "LRPM"**

The LRPM is responsible for placing incoming replicas on the node. It receives replication requests from neighbor nodes and decides with the support of the PSM whether to accept or refuse them. In case of refusal, it launches the replica correction process. It consists of two main components:

(1). **Replica Placement (RP)**: it decides to accept/refuse the RQ. If the replica does not already exist on the node, the RP solicits the PSM to see if the current status of the node allows placing the replica (algorithm 5-4). If the node does not success in placing the replica, it asks the RDis to forward the replication request to another node.

(2). **Memory Management (MM)**: this component manages the storage space reserved for replicas. If the storage space is full, a replacement algorithm is applied in order to decide which replica should be deleted, with the goal of keeping the most "important" ones. This component has not actually been implemented in this thesis, and was left for future work[35].

To evaluate the performance of CReaM, a prototype implementation of the middleware has been developed and integrated into a simulation environment. In the next section, we briefly present available frameworks for simulating mobile ad-hoc networks.

[35] A research work has been done on the cache management in our laboratory [27]

6.2. Simulation Framework

The general purpose of a network simulation is to realistically model a real world network in order to test the behavior of a software component or protocol under specific conditions. After running a simulation instance, the results can be analyzed to deduce how the tested component would perform in a live environment. Simulation has the advantage of enabling tests under many different conditions with different configuration parameters. A network simulation framework provides pre-developed simulation components for many network layers and protocols. Thus relatively complex environments can be realistically simulated without having to develop a very complex simulator from scratch: the developer can focus on developing the simulator for the component that is to be evaluated. This motivates our decision to use this kind of framework in this thesis.

In this context, several network simulation frameworks were serious candidates to develop our simulation. We have conducted a study in cooperation with a master student Johannes Hautz at the University of Passau to evaluate the frameworks with respect to our requirements [38]. All selected frameworks implement discrete event simulations, meaning that the operation of a system is represented as a chronological sequence of events [53]. When running the simulation, the events do not occur in real time, but are specified by a system internal time measurement. The simulation framework guarantees that the events occur in the right sequence with respect to the simulation time. The simulation time is usually implemented by assigning a timestamp to all simulated events. We limit this section to present the description of two simulation frameworks that are the most pertinent with respect to our requirements.

- **Network Simulator "NS-2"**

NS-2 [62] is arguably the most popular network simulation environment; it is highly trusted by most members of the networking research community. NS-2 is a discrete-event simulator organized according to the OSI model. Its core components are written in C++ and released under an open-source license. It was primarily designed

to simulate wired networks. Support for wireless networking has been added to it by several extensions. The Monarch CMU project [2] has published an implementation of the IEEE802.11 layers (Wi-Fi). The BlueHoc [18] and BlueWare [20] projects have provided implementation of Bluetooth layers.

NS-2 is a sound solution for MANET simulation. Unfortunately it suffers from a lack of modularity as well as from its inherent complexity. Indeed, adding components/protocols or modifying existing ones is not as straightforward as it should be. Another well-known weakness of NS-2 is its high consumption of computational resources. A harmful consequence is that NS-2 lacks scalability, which makes the simulation of large networks problematic.

- **OMNet++**

OMNeT++ is an open source, extensible, modular, component-based C++ simulation library and framework [64]. The main objective of OMNeT++ is the simulation of communication networks; however, it is used also to simulate complex IT systems. OMNeT++ enables the design of wired and wireless network simulators that can be easily parameterized.

OMNeT++ provides the basic mechanisms and tools to write simulations, but not "out-of-the-box" simulators. Specific application areas are supported by simulation packages written for OMNeT++, such as INET, INETMANET, MiXiM. These packages are developed independently of OMNeT++.

Unfortunately, similar to NS-2, OMNet++ and the simulation packages targeting mobile networks are still in the development phase. However, OMNet++ has the advantage of providing a powerful GUI that facilitates development of a simulation testbed. In addition, it includes a graphical analyzing tool, which is very useful for performing experimental evaluations. In view of this, we have chosen to use OMNet++ in conjunction with INETMANET to simulate and study the behavior of CReaM in the context of a mobile ad-hoc network. The next section provides additional details about OMNet++ and INETMANET in order to present afterwards the implementation of the simulation of CReaM in these frameworks.

6.3. OMNet++ and INETMANET

In OMNet++, a project is composed of nested components called modules. Each module simulates the behavior of an object that plays a role in the system (e.g. UDP protocol, mobile node, etc). A specific network is thus represented and simulated by a hierarchy of communicating modules.

The communication between the different modules is realized by sending messages via gates, which are defined as the input and output interfaces of each module. Modules that contain sub-modules are called compound modules; the others are called simple modules (Figure 6-3). Hence, in OMNet++, the hierarchy of modules that represents the whole network is itself a compound module.

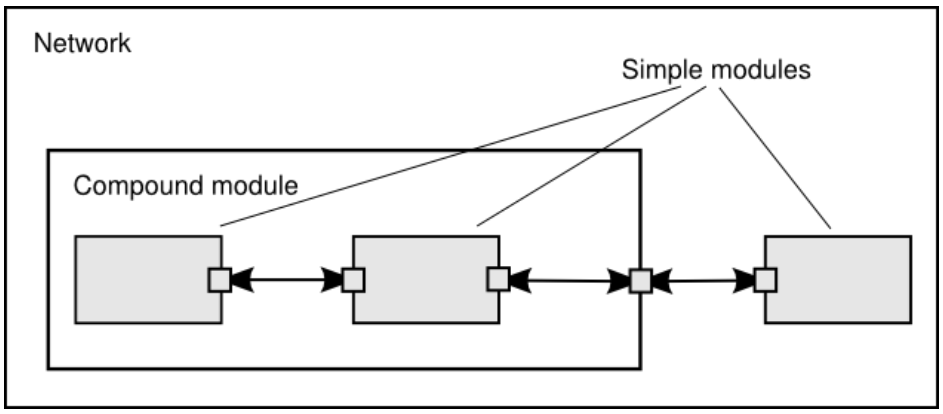


Figure 6-3: Compound and simple modules in OMNet++

The functionalities of a simple module are implemented in C++ in order to describe the behavior of the object. To define the module structure, the NED (NEtwork Descriptor) language is used. .

In addition, we use the INETMANET framework to simulate a mobile ad-hoc network. INETMANET is an extension of the simulation package INET, designed for OMNet++. It is under active development. The two frameworks INET and

INETMANET provide almost the same functionality, but INETMANET contains additional protocols and components to adequately model wireless communications:

- Mobility-aware routing protocols for the network layer. Several routing protocols are available, such as reactive (AODV), proactive (DSR) or even hybrid (OLSR) routing protocols.
- Mobility models such as Ns2-mobility, restricted_const_speed mobility, and linear mobility.
- Link Layer protocols, such as the standard IEEE802.11 with its amendments "a", "g", and "e", used to ensure radio connectivity.

6.4. Design of the Simulator and Experimental Results

In this section, we present the simulation that was carried out using OMNet++ and INETMANET in order to validate the key functionalities of our proposed model and to evaluate its performance. We start by defining the metrics that were measured during the experimentations; we then present the configuration of the simulation environment, describe the implementation of CReaM and finally provide the results of our experimentations. Two series of experiments were conducted. The first series aimed to evaluate the performance of CReaM, while the second series was a performance comparison between CReaM and other replication systems.

6.4.1. Performance Metrics

Three metrics were defined for the experiments, namely data availability, overhead, and user satisfaction. They were used to evaluate the system's performance under specific settings and to compare our system with existing replication systems. These metrics are explained below:

6.4.1.1. *Data Availability "DA"*

This metric represents the percentage of available requested data during the simulation i.e., the percentage of successful requests. Formally, it is defined by the formula:

$$DA = (NoSR/NoTR)*100$$

where NoSR and NoTR are the number of successful data requests and the total number of data requests processed during the simulation respectively. This metric represents a measure of success of a data replication protocol with respect to its main goal of increasing the availability of data items in the network. Unlike in the case of conventional static networks, in mobile environments achieving 100% data availability is nearly impossible, due to the mobility of the nodes and the frequently changing network topology.

6.4.1.2. *Overhead "OVH"*

This metric represents the number of messages generated by the replication system to create the replicas during the simulation. Formally, it is defined as:

$$OVH = \sum_{i=1}^{n} NoM_i$$

where NoM_i is the number of messages related to replication sent by node i and n is the number of nodes in the system. The aim of any replication system is to reduce this overhead as much as possible.

6.4.1.3. *User Satisfaction "US" and Total User Satisfaction "TUS"*

This metric aims to measure the fraction of the simulation time during which the level of remaining resources and of resource consumption is "satisfying", i.e. during which the node is in the stable status as all measured parameters have satisfying values with respect to the tolerance thresholds. Indeed, the idea of maintaining resource

consumption within predefined boundaries is at the center of the design of our model, as it distinguishes it from other replication models. Thus, the user satisfaction is a critical criterion to evaluate the success of our approach. Formally, this metric is defined for each node as $US_i = (NoTS_i/T)*100$, where $NoTS_i$ is the total time during which the i^{th} node has satisfying resource parameters and T is the duration of the simulation.

In addition to the US, we define the Total User Satisfaction (TUS) to observe the effect on the whole network. It is defined as $TUS=\Sigma_t(NoUS_t)$, where $NoUS_t$ is the number of satisfied users during time period T.

6.4.2. Configuration of the Environment

For the experiments, we simulated a square region of size 1500x1500m. This size was selected because it is large enough to cover the motivating scenarios of the thesis, (e.g. the festival music in the streets, detailed in section 1.2.1) and that this square region allows testing multi-hop connections (the outdoor range of the IEEE802.11n protocol which is used in the simulation is 250m). During a simulation run, a predefined number of nodes move within this region and interact with each other for a predefined period of time. Each node runs CReaM and uses the following models:

- **Mobility Model**: each node moves according to the Random Waypoint mobility model [17]. This model is widely used in MANET simulations as it is considered to be the closest to typical movement patterns of real mobile nodes. Random Waypoint breaks down the movements of a node into two types of phases, pauses and motion periods. In a pause period, a node stays at its current location for a certain time. In a motion period, a node moves toward a new random destination at a speed selected in a predefined interval[36]. The destination, speed and direction for each node are all chosen randomly and independently of other nodes. In our simulation, the moving speed of each node was chosen randomly in the range 0 to 3 m/s, corresponding to the

[36] The nodes move in the entire square that does not contain obstacles.

typical range of walking speed of human beings; pause times were set at 3 seconds. The initial position of each host was set randomly.

- **Data Items Distribution Model:** each node is initialized with a set of data items chosen from a predefined list. The number of data items in the network has been varied between simulation runs in order to study its impact on the results. The data items are distributed on mobile nodes in the beginning of the simulation based on the Zipf distribution [5], which has been frequently used to model non-uniform distribution of items involving conscious human choices.

- **Data requests generation model**: is based on the Poisson model [77] with a mean value of 4 requests each 2s. The packet length of a request message is 128 bytes (including UDP and IP headers). Periodically, the simulator selects randomly a mobile node to send the requests. The selected requestor broadcasts the requests to its neighbors and waits for responses. When a host that receives a request holds a replica of the requested data item, it sends back a response message containing the answer. The response message may be simply forwarded to the requestor by a unicast reply using the reverse route of the request. Each requested host copies the replica to its local storage after it receives the replica.

The communication layer is simulated using the following parameters:

- AODV as routing protocol to route requests and data. AODV is indeed usually considered as one of the most efficient routing protocols for MANETs [73]. Note that it could be easily replaced by another protocol, as neither the simulator nor the replication model considers the overhead due to the routing protocol.

- UDP “User Datagram protocol”: UDP is more appropriate in this environment as it implements neither flow control nor error correction contrarily to TCP. These functionalities are no critical requirements of the type of applications

that we target, and also they would cause additional overhead. Thus, it is better to omit them by selecting UDP.

- IEEE802.11n in the MAC layer
- The bandwidth is set to 3Mbit/s (which is so low comparing to the data rate of IEEE801.11n that can go up to 100Mbit/s "effective rate")

Table 6-1: Simulation parameters

MAC Protocol	IEEE802.11n
Bandwidth	3 Mbit/s
Network protocol	IP
Transport Protocol	UDP
Mobility Model	Random waypoint model
Node Speed	1 to 3m/s
Routing protocol	AODV

An important set of parameters of CReaM are the tolerance thresholds. We vary them from one node to another: for each node and each threshold, the simulator assigns a random value taken from a predefined range. For the tolerance threshold of the CPU, the soft value β_1 varies between 4 and 7 requests per 10 seconds, and the hard threshold β_2 varies between 8 and 10 requests per 10 seconds. Indeed, currently the tolerance thresholds are determined arbitrary; however, we aim in a future series of experimentation to determine these thresholds based on the user activity (cf. future work of CReaM 8.2).

6.4.3. CReaM Performance Evaluation

In this section, we analyze the performance of CReaM in various network conditions. Simulation run lasted 600 seconds and the results presented below correspond to the average of the executed runs (the number of runs changes for each particular setting due to the time needed to complete the run. In average, 10 runs were done for a particular setting). In these series of experiments, we consider the effect of three parameters on the performance, namely the number of replicas, the number of data

items and the number of nodes [84]. The following results of this first series of simulations show the performances of CReaM without the implementation of the Potential to place the replicas; i.e., the candidate replica holders are selected randomly among the nodes that are known from the routing table.

6.4.3.1. The Effect of Increasing the Number of Replicas

To evaluate the effect of the number of replicas, we measured the obtained performance in 5 different settings: (1) CReaM is not applied at all; (2) CReaM is applied; each time the replication process is triggered, exactly one replication request is generated; (3) CReaM is applied and two replication requests are generated; (4) three replication requests are generated and finally (4) four replication requests are generated. In all cases, the other simulation parameters were fixed at 100 nodes and 75 data items.

Figure 6-4 outlines the variation of performance measured in terms of data availability over the time. We see here that CReaM significantly increases the data availability. Moreover, the improvement increases over time. We also note that the number of replicas impacts the data. We conclude that RQ should be set to at least 2, as the performance with RQ=1 is clearly inferior. Moreover, we observe that after a cold start phase, all curves corresponding to RQ≥2 more or less converge. Considering the fact that increasing RQ increases the overhead caused by the replication, we conclude that the best compromise is to set this parameter to 2; all subsequent experiments were conducted with this setting[37].

[37] The study of the precision for a sample of these results with a 95% confidence interval shows that the margin of error is around 2%.

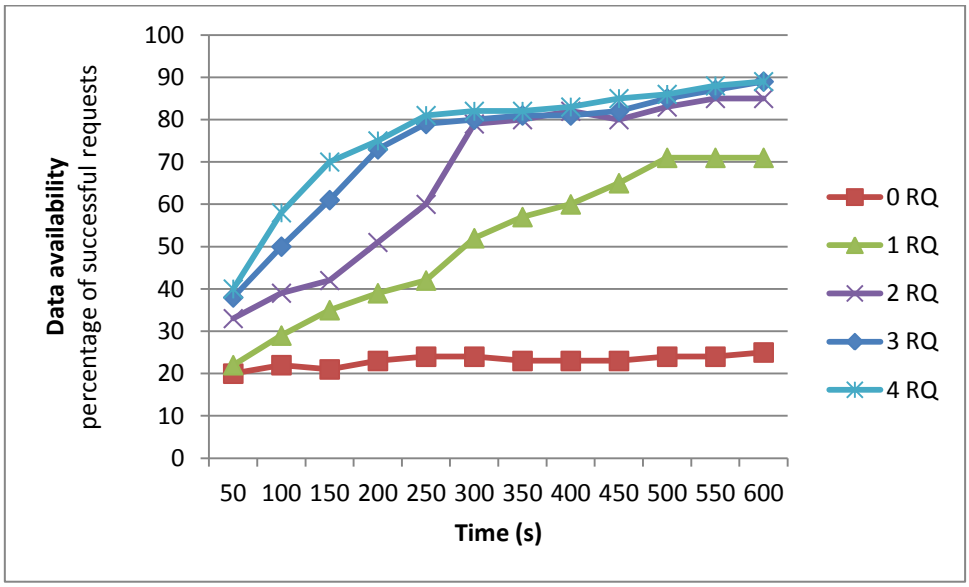


Figure 6-4: The effect of increasing the number of replica on the data availability

Furthermore, a sample of 10 users has been selected to study the effect on the user satisfaction. Figure 6-5 shows the individual results. We can note that CReaM increases the user satisfaction even when only one replica is created in each replication process. Moreover in all 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.

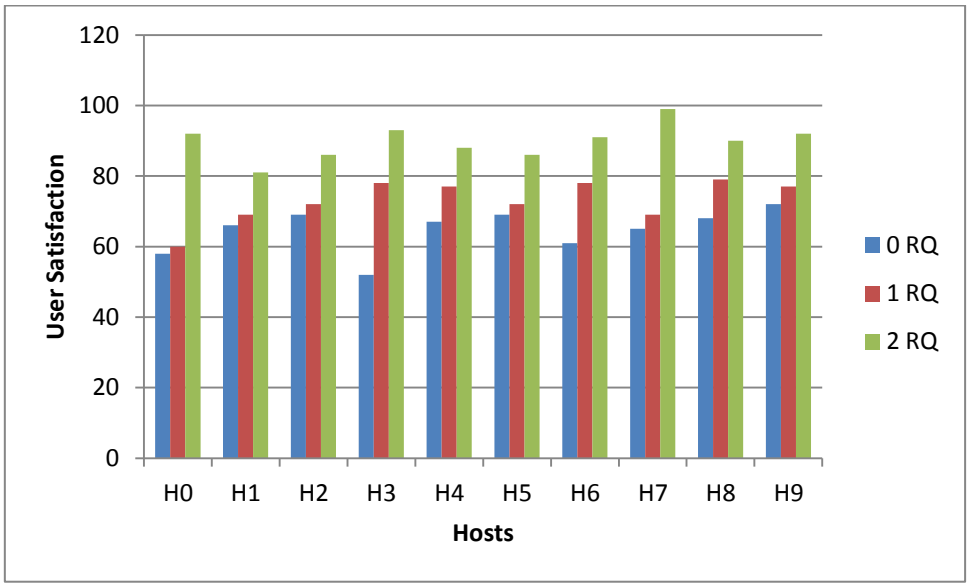


Figure 6-5: The effect on the user satisfaction

6.4.3.2. The Effect of Increasing the Number of Data Items

In this experiment, CReaM is evaluated based on the data availability. Figure 6-6 shows that as the number of data item increases, the positive effect of the replication gets less significant. Indeed, the reason for these results is due to the simulator settings. The data items are randomly distributed on nodes and the number of requests remains constant regardless of the data items increase. Hence, as the number of data items increases, the dispersion of query targets increases as well; each node processes fewer requests and the tolerance thresholds, especially the CPU threshold, are rarely reached. Consequently, the replication is less triggered and the data availability remains low. This interpretation can be illustrated when the number of data items is equal to 150; indeed, we can see that the percentage of satisfied requests is almost the same when CReaM does not create replicas or when it creates 2 replicas. Finally, the choice of the data requested is random in the simulation setting while in the real world, a few data items would actually be requested the most, which should activate more the replication process on the most requested data, increasing consequently the data availability. To get results close to reality, we study in next section the effect of increasing the number of nodes, when the number of requests increases proportionally to the number of nodes.

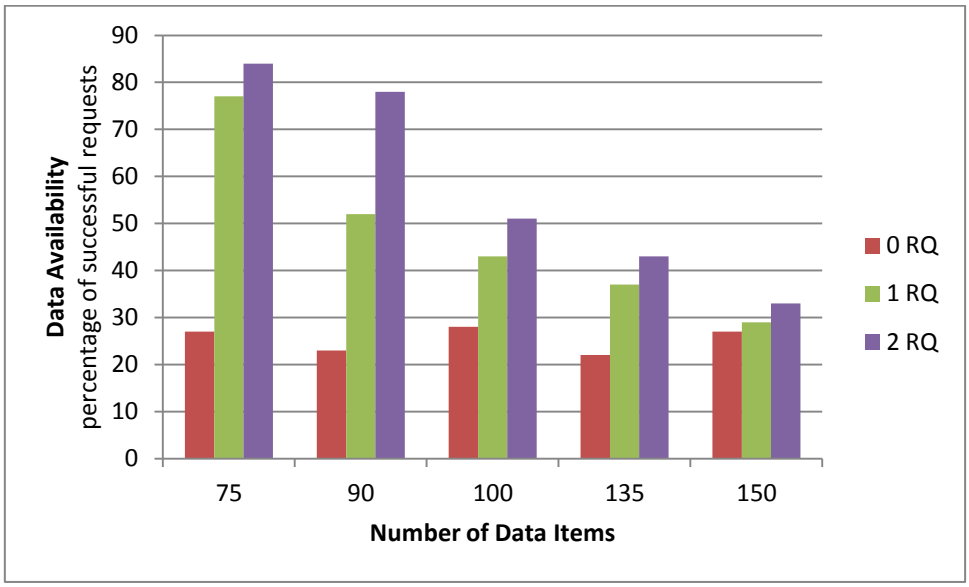


Figure 6-6 : The effect of the number of data items on data availability

6.4.3.3. The Effect of Increasing the Number of Nodes

In this series of experimentations, we study the effect of the network density. To this end, we keep the simulation area size unchangeable, and we increase the number of nodes. To obtain a scenario close to the reality, the increase in the number of nodes is accompanied with a proportional increase in the number of requests and in the number of data items in the network. Figure 6-7 shows that CReaM maintains a high data availability in all cases when one replication request is sent. However, the results show that the percentage of successful requests when the number of nodes equals to 200 nodes is higher with 1RQ than with 2RQ. Indeed, this result is due to the network collisions that occur when the number of packets sent at the same time in the network increases and that lead to discard the packets. Note that, if the number of nodes continues increasing while the simulation area remains fixe, the degradation in the data availability will be obtained in all cases (0RQ, 1RQ, and 2RQ) while that with CReaM the data availability remains much higher.

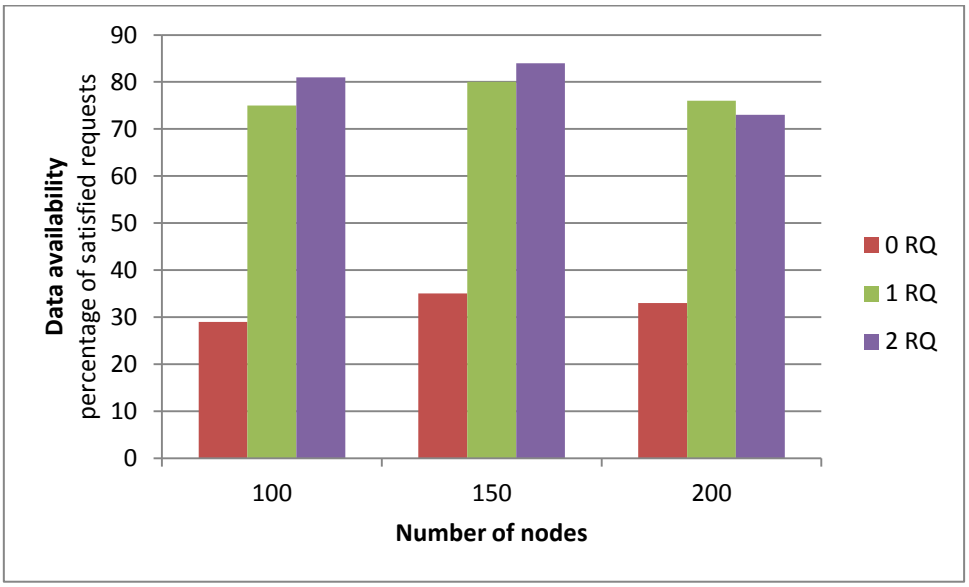


Figure 6-7: The effect of the number of nodes on the data availability

6.4.4. Comparative Study

CReaM is a reactive replication model; it starts the replication process when the resources consumption exceeds acceptable levels. This is one of the main different features from most existing works, in which each node periodically decides whether to replicate some of its frequently accessed data items [34]. Thus, we compared our model to a general periodical replication model using the performance metrics presented above [83].

The parameters of the periodical model are identical to those of CReaM (cf. section 6.4.2); however, the parameter that replaces the tolerance thresholds and the time interval when the consumption levels are verified is the replication period. It is set to 10s (to be equivalent to the time interval of CReaM). For both models, the number of nodes is fixed to 100 nodes and the number of replicas is equals to 2 each time the replication is needed.

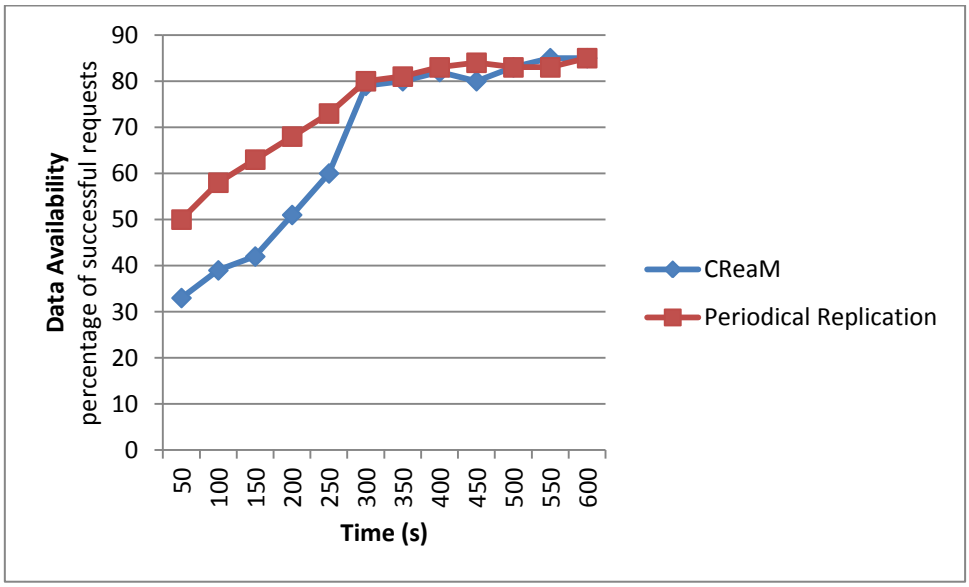


Figure 6-8: Data availability over time

Figure 6-8 shows the obtained results. We note that the periodical replication model increases the data availability better than CReaM in the beginning of the simulation; but with time, CReaM increases the data availability in the same way as the periodical replication does. In other words, CReaM needs some time to reach the same performance, because the replicas on the network are created only when it's needed. However, for the same reason CReaM causes less overhead.

A comparison between the overhead caused by CReaM and by the periodical replication was done. In this experiment, the overhead includes the replication requests and the replication acknowledgements that are sent from all nodes. Figure 6-9 shows that during the simulation, CReaM causes much less overhead than the periodical model; for instance, at time 600 CReaM causes half overhead than the periodical model though the data availability (see Figure 6.8) is the same. This is because CReaM only sends replication requests when the availability of a resource becomes less than the tolerance thresholds.

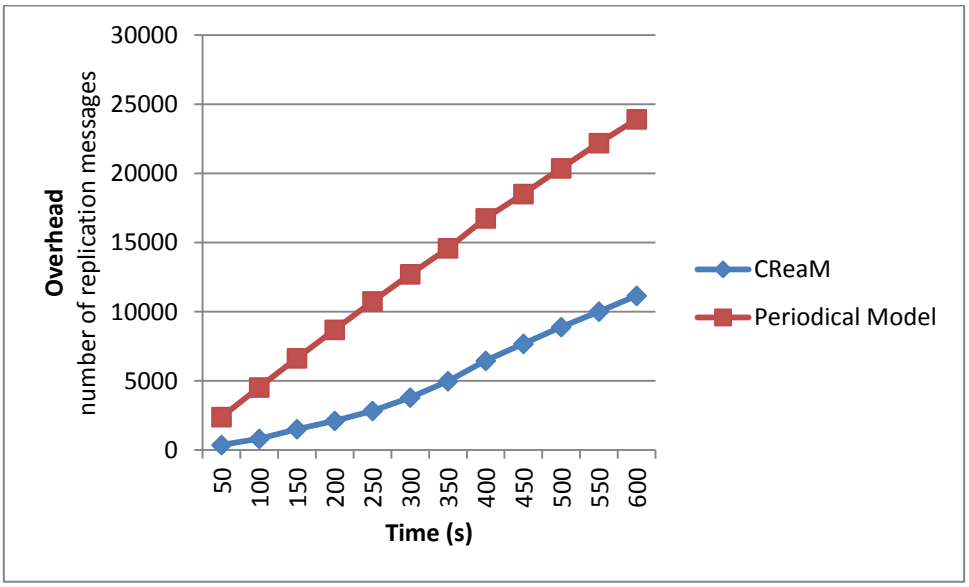


Figure 6-9: Overhead over time

In a second series of experiments, we have studied the influence of the number of data items on the overhead created by CReaM. From Figure 6-10 and Figure 6-11, we can observe very different results between the overhead caused by CReaM and by the periodical model while the global rates of the data availability are close. We can also note that the positive effect of CReaM on the overhead gets more significant as the number of data items increases. As explained in section 6.4.3.2, the number of requests remains constant regardless of the data items increase, hence, the dispersion of query targets increases (which explain the degradation of the data availability whatever is the considered replication model) and the replication is not triggered by CReaM. On the other hand, the overhead of the periodical model remains fixed since the replication is not related to the number of requests diffused in the network. Regarding the data availability, as shown in Figure 6-11 the global rates are close in all cases. For example, when the number of data item is equal to 75, CReaM causes a 50% lower overhead than the other model while they provide the same rate of data availability.

Finally, the random choice of the requested data item affects the obtained global rate of data availability. In most real life applications a few data items would actually be requested which should lead to significantly better data availability.

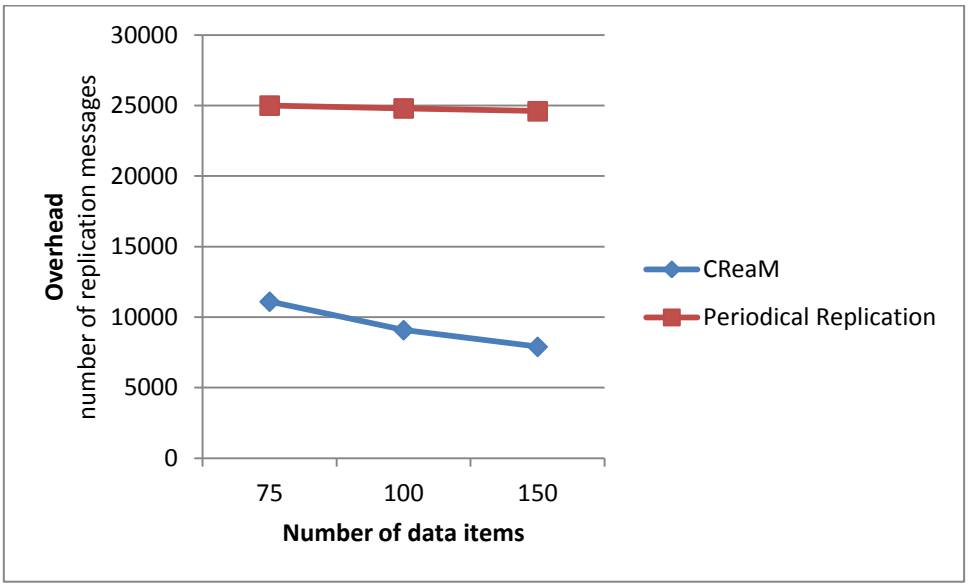


Figure 6-10: The effect of the number of data items on the overhead

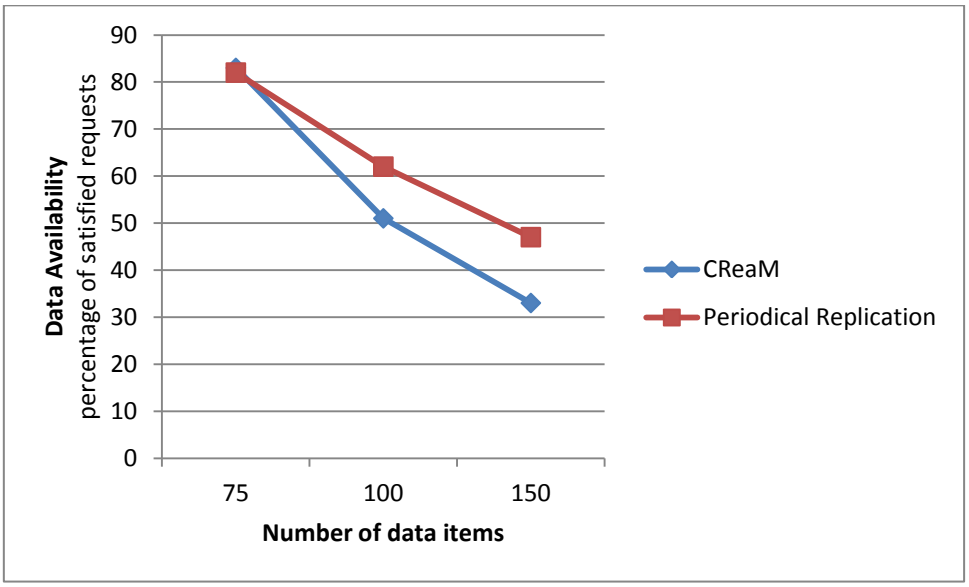


Figure 6-11: The effect of the number of data items on the data availability

A last series of test has been done to compare both models in terms of user satisfaction. Figure 6-12 shows individual results for a sample of 10 users. We can note that CReaM dramatically increases the user satisfaction. This tends to show that

CReaM has the desired effect of better distributing the load of data requests, which contributes to keep the user satisfied.

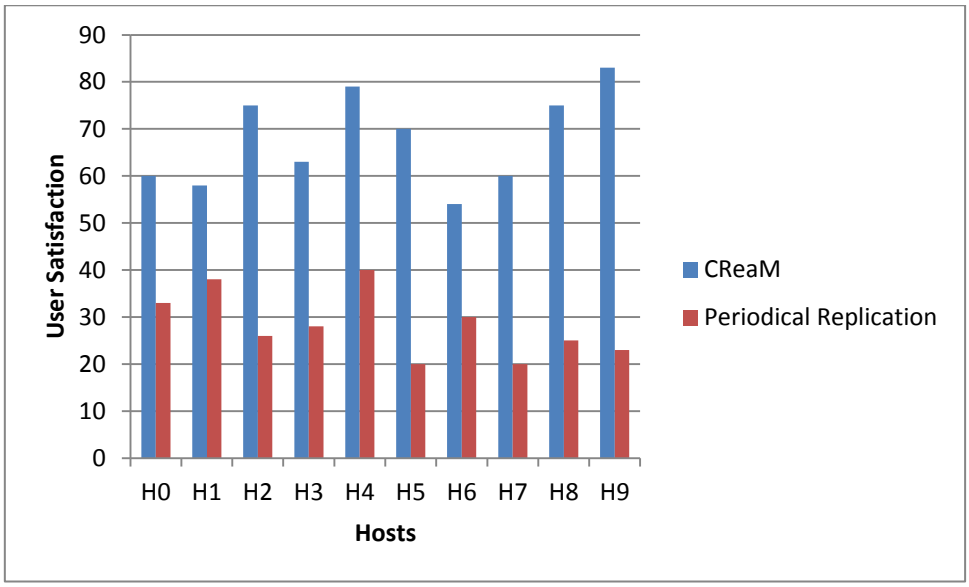


Figure 6-12: The user satisfaction

In addition to the user satisfaction US, we have evaluated the TUS. Figure 6-13 shows that with CReaM the number of satisfied users is significantly higher than with the periodical replication model that does not care about the 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.

Indeed, the periodical replication leads to high replica duplication, since it replicates data after each time interval even if replication is not needed. This replication increases significantly the data availability but on the other hand since a lot of nodes receive the data requests concerning the same replicas all these nodes will process the requests and consume their resources consequently the number of unsatisfied nodes increases. With time this case will be encountered with CReaM (especially when the number of replicas generated by a replication action equals to 2, but CReaM takes this into account in its strategy for controlling the resource usage (e.g., it stops responding

to data requests (for one time interval) when the hard threshold is exceeded) and therefore it increases the number of satisfied users.

Note that the variability of the obtained results (Figure 6-13) over the time (both for CReaM and for the periodical model) is due to the random distribution of requests on the nodes; the simulator selects a number of requests to be sent each period of time based on a Poisson model, and it randomly selects the sender nodes. Consequently, requests might be processed by the same nodes in 2 subsequent time intervals or they might be distributed over the nodes.

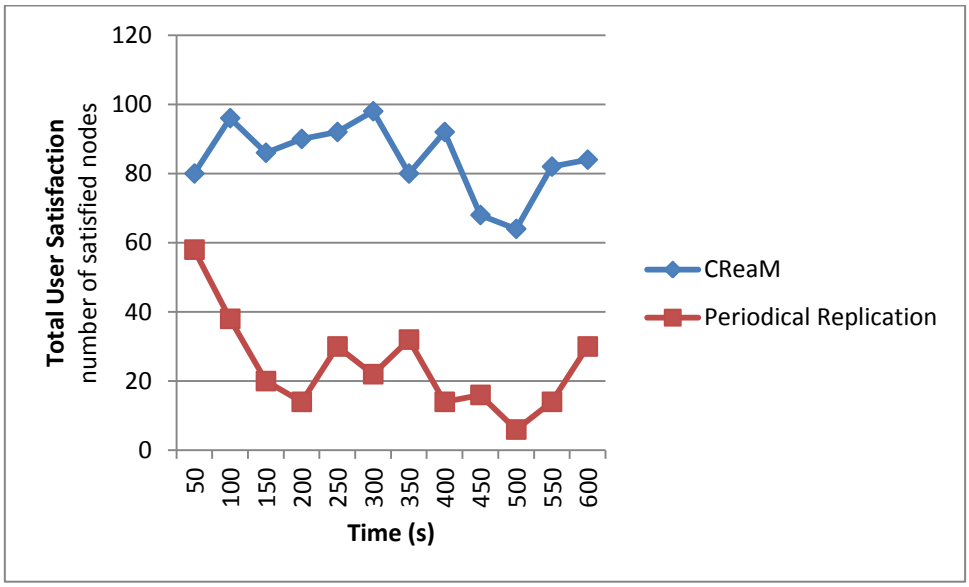


Figure 6-13 Total user satisfaction over time

6.5. Conclusion

To evaluate the CReaM model, we have first designed a middleware that implements CReaM in a component-based prototype. We have then decided to use the OMNET++ simulation framework with INETMANET to evaluate the model using a simulation study. In these experiments, we have measured the performance using three metrics: data availability (percentage of successful data requests), overhead (number of

additional network messages generated by the replication system) and user satisfaction (fraction of time during which resource availability is satisfying – determined using the tolerance thresholds).

This first series of experiments studies the effect of the number of replicas and the number of data items on the performance of CReaM. It shows that CReaM does indeed improve the data availability by increasing the number of satisfied queries while satisfying the user for the availability of his resources. In a second series of experiments, we have compared CReaM with a classical periodical replication model. The results show that over time, both models have similar results with respect to data availability. However, CReaM clearly outperforms its adversary when considering the user satisfaction, which demonstrates its resource-preserving feature. Moreover, the overhead caused by CReaM is significantly lower.

Other series of experiments are still under development to evaluate the effect of using potentials for selecting replica holders on the overhead, on the data availability (especially when the number of nodes increases) and finally on the response time of data requests.

III

CoFFee

CoFFee:

Cooperative and Infrastructure-Free peer-to-peer System

In this chapter, a multimedia content distribution system is proposed for highly dynamic networks, such as Vehicular Ad-hoc Networks (VANETs). This system is named CoFFee, Cooperative and infrastructure-Free peer-to-peer. This research work has been achieved in collaboration with Talar ATECHIAN, a PhD student at LIRIS laboratory. CoFFee being a decentralized P2P system, it mainly performs based on the cooperative behavior of the nodes. Some operations are applied on the multimedia files in order to improve the content distribution in the network. In addition, a replication mechanism is also proposed to contribute in increasing the data availability in the network.

This chapter is organized as follows: in section 7.1, an overview of Vehicular Ad-hoc Networks is presented. DG-CastoR a geo-multicast protocol in explained in section 7.2. Then, in section 7.3, the proposed multimedia file partitioning technique is described. Section 7.4 is the main part of the chapter; it explains in details the functionality of CoFFee. The results of experimentations are presented and discussed in section 7.5. Finally, the integration of CoFFee and CReaM is discussed in section 7.6.

7.1. Introduction

Vehicular ad-hoc networks (VANETs) are a special kind of MANETs, mainly composed of mobile vehicles. In VANETs, each mobile vehicle, named also mobile node, acts as a router. This characteristic allows the mobile nodes to behave as relays connecting the nodes that are out of the communication range to each other. Consequently, a network with a wider range is gradually created. One specific property of VANETs is that rather than moving on a random pattern, vehicles tend to move in an organized manner and are constrained by the road networks' topology. Similarly to MANETs, VANETs are characterized by a dynamic topology because of the high mobility of the nodes. When a mobile node joins or leaves the communication radio range of another node, the network topology changes. Frequent updates are therefore necessary to maintain the connectivity between communicating

nodes. Compared to MANETs, in VANETs this problem is particularly challenging because of the very high mobility of the nodes.

VANETs target two main categories of applications: intelligent transportation systems and infotainment applications [55]. Intelligent transportation systems vary from basic management systems such as car navigation, traffic signal to more advanced applications that integrate live data and feedback from a number of sources such as parking guidance, road safety, real time traffic condition, services and car accident alert, etc… Infotainment applications include Internet access inside vehicles, commercial advertisements, and various peer-to-peer applications.

Peer-to-Peer (P2P) applications are proposed as alternatives to traditional centralized client/server architectures. Decentralized well-known P2P systems include for example Gnutella [28], Chord [44], CAN [81] and Freenet [24]. These P2P systems are usually deployed over wired networks. The development of Peer-to-Peer applications for file sharing in VANET environments introduces additional challenges. Indeed, P2P for VANETs requires stable link connectivity between nodes to perform efficiently. In general, deploying a P2P system over a VANET requires an adaptation to the characteristics of this environment mainly: (1) the dynamic aspect of the network causing frequent breaks of the links; (2) the limited bandwidth allocated to each node and (3) the nodes' limited resources.

The Quality of Service (QoS) in P2P systems over VANETs is based mainly on two key factors: (1) the routing protocol ensuring a reliable delivery of the requests and the data, and (2) the proper use of the shared bandwidth when transmitting data, especially if the transmitted data are large (e.g. multimedia files) and the allocated bandwidth is limited.

In this chapter, a novel multimedia content distribution Peer-to-Peer system, called CoFFee: **C**ooperative and In**f**rastructure-**F**ree p**ee**r-to-p**ee**r [9], is proposed. CoFFee is deployed at the application layer of the OSI model. It is conceived to be handled by DG-CastoR, a well-adapted routing protocol for dynamic ad-hoc networks such as VANETs, since the mobility traces of the nodes are considered as a key element in the routing process [8].

CoFFee has been particularly designed to improve the Quality of Service (QoS) in terms of transmission of multimedia files between mobile nodes. The QoS is ensured mainly by the nodes that cooperatively behave in the infrastructure-free ad-hoc network, in order to intelligently process the disseminated requests. More precisely, in CoFFee, the nodes cooperate to:

- Avoid transmitting redundant data
- Increase the data availability of the data items that are identified as rare[38] in the network,
- Optimize the bandwidth usage.

Concretely, CoFFee partitions multimedia files into fragments to facilitate the transmission process and to optimize the use of the bandwidth. Therefore, when a request is diffused over the network, each provider node (i.e. node that has the requested file) by receiving the request, prepares a set of files (fragments of the file) as response set to the request. Furthermore, provider nodes apply some cooperation actions in order to ensure a complete delivery of the requested data item to the requester node.

Two cooperation actions are applied on the set of retrieved fragments to be sent to the requestor node (also called master node): (1) deleting from the set the file/fragment(s) that have been already transmitted by other node(s) (2) replicating the file/fragments that are considered rare in the network.

Before explaining CoFFee in details, we present in the next two sections an overview of the routing protocol "DG-CastoR" proposed by Talar Atechian (a PhD student of our team at the LIRIS laboratory), and the multimedia file partitioning technique that we propose.

[38] The definition of rare data item in CoFFee differs slightly from the definition given in CReaM (cf. section 4.3.2.3). In CoFFee, a data item is considered rare if it only exists on one node among the nodes that process cooperatively the requests.

7.2. DG-CastoR : A Geo-Multicast Routing Protocol

Disseminating packets in a very dynamic network such as a VANET needs an adapted packet propagation strategy to handle the high mobility of the mobile nodes. Thus, the packet dissemination process in Intelligent Transportation Systems (ITS) is applied in a push manner, i.e. some nodes (e.g. the ones located in a specific geographic area) are selected as potential nodes to receive the propagated packets (in multicast manner) without the need for these nodes to send any prior request. This type of diffusion is called a geo-multicast routing mechanism. Oppositely, in an infotainment application, a file transfer is triggered only when a node sends a request in the network, also called "push" mode.

The main challenge in infotainment applications, comparing to ITS, is this query/response mechanism. A node that sends a request should receive properly the response. In some cases, the request route is reused for the reply. However, in a highly dynamic network such as a VANET and because of its ad-hoc nature (i.e. nodes can leave or join the network at any moment without notification), it is not easy to maintain the same routing path until the response is sent. Such a mechanism indeed necessitates frequent updates of the links and therefore increases the overhead due to the control packets diffused on the network.

DG-CastoR, a geo-multicast routing protocol was proposed by Talar Atechian in [8]. DG-CastoR is a VANET adapted protocol for infotainment applications; it creates an application-layer overlay that aims to distribute properly the queries/responses in the network. The detailed mechanism of GD-CastoR is explained in sections 7.2.1 and 7.2.2.

7.2.1. Query Dissemination Process

The dissemination process of the query is accomplished in a geocast manner; i.e., the dissemination decision making algorithm is aims at finding out the nearest trajectories in terms of spatio-temporal similarities. The proposed method, named TQ (Trajectory Queries), is based on the future trajectories of the mobile nodes. It uses the model of

trajectory adopted in [7]. It is assumed that the future trajectories of the mobile nodes are collected from the GPS navigation system attached to the nodes in KML [63] or GPX [32] format. In general, the traces returned by the navigation systems are in ellipsoidal coordinates. These coordinates are three-dimensional coordinates representing the longitude, the latitude and the elevation. For computation simplicity, only two-dimensional Cartesian coordinates are considered. The TQ method applies transformation methods to transform ellipsoidal coordinates into Cartesian coordinates (x,y). The TQ method belongs to the class of the Range Queries methods [40], [29]. These methods consist in retrieving neighbor nodes that belong to a specific predefined search zone. In TQ, the search zone considers the trajectories of the nodes. Therefore, TQ computes the spatio-temporal similarity between the query trajectory and a set of node trajectories and selects similar trajectories. A detailed description of the TQ spatio-temporal similarity method is presented in [7]. It is important to mention that the output of TQ is a time duration; indeed, the main objective of the similarity measurements is to compute the connectivity duration, (named ECD – **Estimated Connectivity Duration**) between the nodes. Therefore, when the ECD exceeds a minimum threshold duration value (predefined as a constant value), the node is elected to join the overlay that groups the nodes with a close trajectory. It then becomes legitimate to receive the query disseminated by the node initiator of the TQ process.

7.2.2. Overlay Network Construction

Group communication has become important in ad-hoc networks due to the need for collaborative tasks among groups of mobile nodes. However, because of the high mobility of the nodes, it is difficult to maintain a reliable connection during file exchange (multimedia files necessitating an acceptable duration to be transmitted from a source node to a destination). Therefore, in the geo-multicast protocol DG-CastoR, an application-layer **overlay** is constructed for each disseminated query and identified by a unique identifier Overlay_ID. The overlay construction process is initiated by the requestor node named also master node of the overlay. The overlay network member selection is processed by TQ, the nearest neighbor method proposed

in DG-CastoR (see above section 7.2). Thus, the two inputs in the TQ method are the *master node's* trajectory and the trajectory of each neighbor node in the network.

The topology of the application-layer overlay is a star topology centered on the master node. The link between the master node and an overlay member node is initially a virtual link (since the similarity calculation is based on future trajectories), until the node becomes in the transmission range of the master node (physical link), see Figure 7-1.

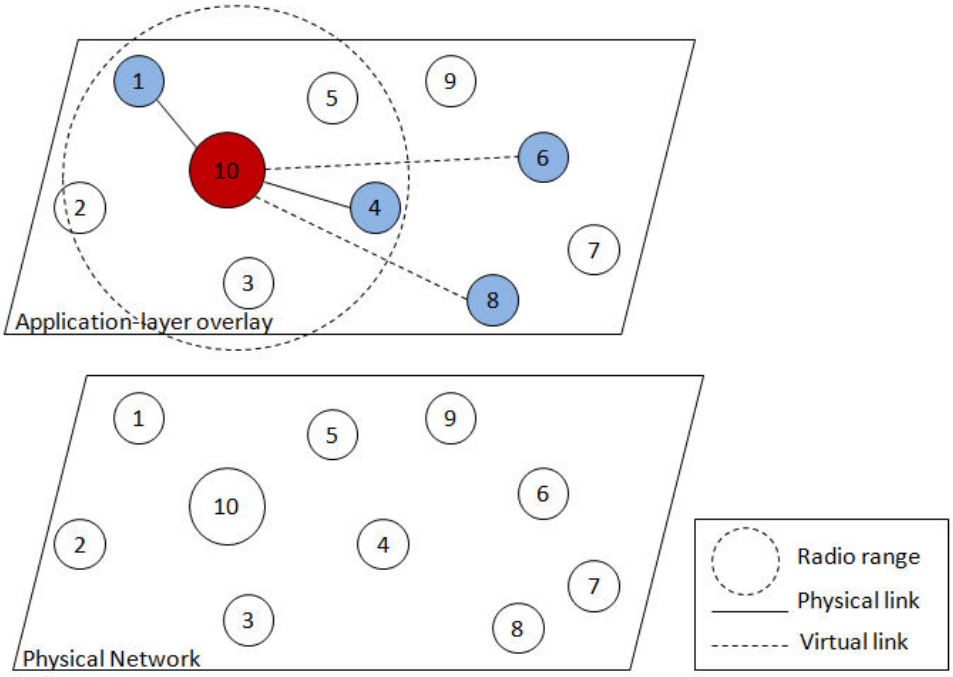


Figure 7-1: Application layer Overlay

Figure 7-1 illustrates an example of application-layer overlay. The node 10 is the master node who initiates the overlay construction (i.e. node 10 is requestor node), and the nodes 1, 4, 6 and 8 are application-layer overlay nodes. The links between the node 10 and the nodes 6 and 8 are virtual links since it is estimated that the node 10 will only meet them in a near future. However, the links between the node 10 and the

nodes 1 and 4 are physical links since they are in the same radio transmission range, so that they can communicate directly.

This concept of overlay presents two main advantages: (1) it guarantees a future reliable communication between any overlay node and the master node (2) It guarantees to track easily the all potential query answer nodes by collecting them in the same overlay. Note that, for each query, a new overlay is created. Consequently, it is possible that a node belongs to multiple overlays at the same time, if it complies with the trajectories similarity conditions.

7.3. Multimedia File Partitioning

Peer-to-Peer multimedia file sharing systems aim to distribute large files (e.g. audio files, videos, images) among nodes using block partitioning. Each provider node distributes the blocks it carries when requested, even if it does not hold the entire requested file. The distribution by block decreases the distribution time, meanwhile, it reduces also the shared bandwidth usage.

Several interesting algorithms have been proposed for partitioning and distributing blocks among the nodes. BitTorrent [68], one of the most popular content distribution protocols, is a file-downloading protocol that relies on external components, such as web pages to search for files. In BitTorrent, files are split up into equal-sized chunks. To find a file in BitTorrent, users access web sites which act as global directories of available files. Thus, to start downloading a chosen file from a retrieved list, the user clicks on a link pointing to .torrent medatada file. The metadata files are distributed among a number of file servers and monitored by a tracker which keeps a global registry of all downloaders and seeds of a corresponding file. The BitTorrent protocol is considered as one of the Internet's most efficient content distribution protocols; indeed, BitTorrent is basically tailored for infrastructure dependent and centralized environments. However, in wireless ad-hoc networks, BitTorrent running with its default configuration does not lead to good performance. Thus, a tracker-less content distribution protocol BitHoc [51] was proposed for ad-hoc networks. The main contribution in BitHoc is the neighbor discovery step; because of the absence of a

centralized tracker, each node, before starting the content distribution step, discovers its neighbors and their contents.

In addition to content sharing protocols, several file fragmentation techniques have also been proposed [75] [86] [60].

In this thesis, a simple partitioning and distributing algorithm is proposed as for a first prototype. It will be interesting to evaluate one of these advanced fragmentation algorithms in a future work. The fragmentation technique we propose in this work is based on a multi-level partitioning of the blocks (named also fragments). Thus, a file F is at the beginning partitioned equally into N fragments (at level 1). Furthermore, each fragment is partitioned into M sub-fragments (see Figure 7.2).

Multi-level partitioning is a technique often adapted to solve transmission problems (short connectivity duration, insufficient bandwidth capacity, etc.). However, in order to restrict the number of fragments/sub-fragments of a particular file, we limit the number of fragments by decreasing the number of sub-fragments by one at each sub-level. The partitioning process stops when the number of fragments at some level reaches the value of 2. This limit is set to facilitate the assembly mechanism of the fragments at the receiver node.

Let consider a scenario where partitioning into sub-fragments is detected as necessary: for instance, assume that a provider node (i.e. node that carries blocks to be sent) detects a congestion of the bandwidth or that it estimates that the communication time will not be sufficient to transfer a large fragment, so that this node decides to divide the fragment into sub-fragments. The number of fragments will be decreased by one at each level: a fragment from level 1 will be divided into N sub-fragments; a fragment from level 2 into N-1 sub-fragments, etc.

In the example of Figure 7-2, the value of N is set to 4 (i.e. 4: number of fragments allowed at level 1). F is therefore first subdivided into 4 fragments (F_1, F_2, F_3, F_4). These fragments are further partitioned into N-1=3 sub-fragments at level 2 ([F_{11}, F_{12}, F_{13}], [F_{21}, F_{22}, F_{23}], [F_{31}, F_{32}, F_{33}], [F_{41}, F_{42}, F_{43}, F_{44}]). At level 3, the value N-2 = 2 is

reached meaning that each fragment of level 2 (example: F_{11}) is subdivided into two sub-fragments ([F_{111}, F_{112}] …). The partitioning thus stops at level 3.

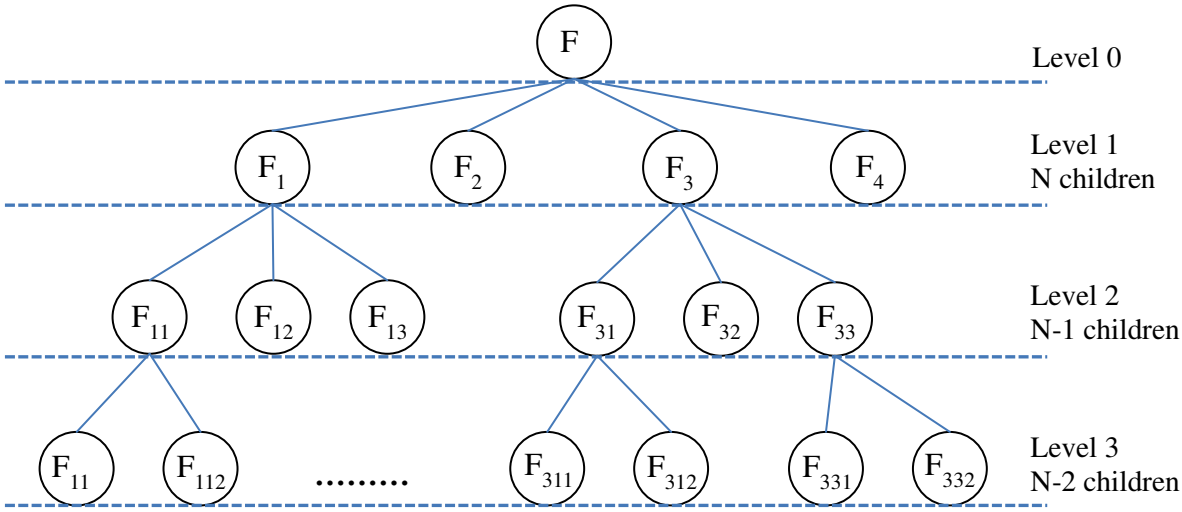


Figure 7-2: File partitioning

Each fragment includes a header containing a unique identifier consisting of the identifier of the file F and the position of the fragment in the tree. During the assembly process, this information is used to reconstruct the whole file. In the example of Figure 7-2 the ancestor fragment of F_{13} is the fragments F_1, the descendant fragments of F_3 are F_{31}, and F_{32}.

7.4. CoFFee: Cooperative and Infrastructure-Free peer-to-peer System for VANETs

We define **data block**, as any response to a query, either the entire file or a fragment of the file. A data block is carried by a **provider node**. In the following, the term **BList** is used to refer to all data blocks carried by a node. Each node is responsible for transferring its BList with the master node when they are in the same communication range (one-hop communication strategy defined in DG-CastoR). Consequently, the

provider node calculates an estimation of the duration of the BList transfer. If this duration exceeds the estimated duration of connectivity with the master node (calculated by the TQ method), the node cooperates with other nodes belonging to the same overlay. The cooperation requires that the nodes have a partial knowledge of existing data blocks in the overlay; for this purpose, when nodes of the overlay are in direct vicinity, they exchange the metadata of their BList. Based on this information, the node can trigger improvement services to reduce the size of its BList.

In order to explain the improvement services proposed by CoFFee, we present in the next sections: (1) the classification of the nodes in the overlay based on their BList content. (2) The calculation of the estimated transmission duration ETD of the BList. (Indeed, improvement services are only applied when the estimated connectivity duration ECD between the provider node and the master node is less than the estimated transmission duration ETD).

7.4.1. Nodes Categorization

CoFFee classifies the nodes in a application-layer overlay into three categories:

- **Root** nodes (R): nodes that contain in the BList one fragment corresponding to the whole requested file.
- **Holder** nodes (H): nodes that contain in their BList some fragments of the requested file (regardless of the partitioning level).
- **Blank** nodes (B): nodes that do not contain data blocks response to the request; (i.e. their BList is empty). These nodes are nevertheless in the overlay because they are expected to be able to communicate with the master node directly with one hop connection (unicast manner) in a near future. This is the criterion that is applied to build the overlay, regardless of whether the node possesses or not blocks of the requested file.

7.4.2. Estimation of the Transmission Duration « ETD »

The transmission duration corresponds to the amount of time required to transfer all the data blocks of the BList from a provider node to the master node (i.e. requestor node). The transfer operation begins when both nodes are in the same communication range so that a physical link is established between them. The estimated transmission duration might be different from the actual transmission duration, since the calculation is done in advance by taking into consideration the theoretical bandwidth allocated to the node and not the effective bandwidth. Note that, the effective bandwidth is the actual speed at which data can be transmitted by a node; the effective bandwidth is concretely determined during the data transmission according to the traffic on the communication links. For a better precision, it would be effective to consider the effective bandwidth instead of the theoretical bandwidth. For this purpose, several methods have been proposed that aim to estimate the effective bandwidth. [88], [10].

Thus, **Estimated Transmission Duration** (**ETD**) for a provider node S is calculated by the following formula:

$$ETD(S, BList) = \frac{\sum_{i=1}^{n} Size(F_i)}{BW}$$

Where Size(F_i) is the size of the data block F_i, n represents the number of blocks in the BList of the node S, and BW is S's theoretical allocated bandwidth.

7.4.3. Nodes Collaboration for Data Sharing improvement

In order to execute data blocks distribution improvement algorithms, some information are exchanged by nodes belonging to a same overlay. Note that, this exchange takes place when overlay nodes become one hop neighbor nodes (i.e. within a same communication range to each other), and each overlay node stores this exchanged information in its neighbor table.

- The BList that respond to the received query (remember that an overlay is built for each query).
- The Estimated Connectivity Duration (ECD) with the master node (i.e. requestor node).
- The estimated start time of the communication, i.e. the moment at which the provider node and the master node will become within the communication range of each other.

Content distribution improvement algorithms are applied when the transmission duration of the blocks in the BList (ETD[39]) exceeds the Estimated Communication Duration (ECD). In such a configuration, the node tries to decrease the size of its BList. Therefore, it consults autonomously and periodically its neighbor table and determines the corresponding improvement algorithm(s) (deletion/replication) to be applied on its own BList. As long as ETD<ECD the node applies improvement algorithms periodically (since the neighbor table is updated when the one-hop neighbors changes) until it meets the master node.

7.4.3.1. Deleting Data Blocks Operations: Prune / DivideAndPrune

The deletion operation consists of deleting from the BList, blocks that can be transmitted by other overlay node(s). The main objective of this operation is to optimize the bandwidth usage by preventing the transmission of redundant data blocks. Note that, the proposed deletion operation includes two actions: Prune and DivideAndPrune.

The Prune action performs as follows: for each data blocks in the BList, the pruning action is applied when one of the following conditions is verified:

[39] For simplicity, ETD, and ECD represent ETD(sender, BList), ECD(sender, receiver) respectively.

- If the data block is transferable by another node (i.e. a node that stores this block and that has an ETD for its BList smaller than ECD with the master node)

- If the ascendant fragment is transferable by another node.

- If all the descendent fragments are transferable by another node.

The DivideAndPrune action is applied when none of the Prune conditions is verified. DivideAndPrune performs as follows: A selected fragment F is partitioned into m sub-fragments (m+1 being the number of fragments at the previous level); of course, this partitioning can only be applied if $m \geq 2$. Once this sub-partitioning is completed, a Prune action is applied on the sub-fragments exactly as described above (cf. section 7.3).

In the following we present the deleting operations algorithm **deletingDataBlocks** Algorithm 7-1. This algorithm is executed on the data blocks (e.g. BList) of the node N. This algorithm uses the following data types:

- **Block**: object corresponding either to the whole requested file or to a fragment of the requested file. It provides the methods (1) getID(): returns the identifier of the block. (2) isDivisible(): a Boolean function returning true if the block can be divided into sub-fragments, otherwise it returns false. (3) divide(): divides the block and returns a list of blocks that contains the sub-fragments.

- **BlockList:** non-ordered list of Block objects. It provides the methods (1) deleteDataBlock(block): removes block from the list if it exists. (2) size(): returns the number of blocks in the list. (3) isEmpty(): Boolean function returning true if the list does not contain any block, otherwise it returns false. (4) addBlocks(listOfBlocks): returns the new list resulting from adding the elements of listOfBlocks. (5) getDataBlock(): returns the first block in the list and removes it from the list, null if the list is empty. Selecting the first block from the list for the pruning operation is trivial; however, ongoing work proposes more intelligent solutions (cf. future work section 8.2.2).

- **Neighbor**: object corresponding to a neighbor node. It provides the methods: (1) getID(): returns the identifier of the node, (2) getBList(): returns the list of the data blocks carried by the node, (3) getOverlayID(): returns the identifier of the overlay the node belongs to. When a node belongs to multiple overlays, multiple objects Neighbor are created in the Neighbor table to represent it.

- **NeighborTable:** array containing Neighbor objects.

Algorithm 7-1

Algorithm: deletingDataBlocks

This algorithm is executed by a provider node when the estimated transmission duration of the data blocks (ETD) exceeds the estimated connectivity duration with the master node (ECD). For each data block in the BList, the node searches in its neighbor table for the nodes that contain the data block (or its sub-blocks) and that can transmit it. In this case, the data block is removed from the BList of the node using the **prune** and/or **divideAndPrune** actions. Then, the estimated transmission duration of the BList is decreased. The algorithm stops when the condition ETD <ECD is true or when no further deletion is possible.

Input:

BList: list of the data blocks of the node (type: BlockList)
neighborTable: table containing the neighbors of the node (type: NeighborTable)
idOvl: identifier of the overlay (type: Integer)
M: master node id, i.e., the source of the request (type: Integer)

Output:

BList: BList of the node after deleting the redundant blocks. It contains the blocks to be sent to the master node (type: BlockList)

Variables:

TBList: temporary list that contains at first the same blocks as BList (type: BlockList).
currentBlock: temporary variable that contains the current data block (type: Block)
subFrags: temporary list of Blocks that contains the sub-blocks of currentBlock

Begin

```
1.  TBList = BList
2.  While ( ETD (this, BList) > ECD (this, M) ) do
3.          /* ECD(this,M) returns the connectivity duration between the current node and
4.           the master node M*/
5.          currentBlock = TBList.getDataBlock() //return a block and remove it from TBlist
6.          If (currentBlock != null) then
7.                  BList = prune[40] (BList, neighborTable, idOvl, currentBlock, M)
8.          Else
9.                  break //no more blocks in the TBList to be pruned so go out of the while loop
```

[40] See Algorithm 7-2

```
10.          End if
11.    End while
12.    //if ETD is still > ECD try the DivideAndPrune action,
13.    //thus initializer the empty TBlist with the blocks remaining in BList
14.    TBlist = BList
15.    While ( ETD (this, BList) > ECD (this,M) ) do
16.          currentBlock = TBList.getDataBlock()
17.          If (currentBlock != null) then
18.               subFrags = divideAndPrune[41](neighborTable, idOvl, currentBlock)
19.               replace (currentBlock, subFrags , BList) /*replace currentBlock in the BList
20.                                                  by the list of its non-pruned sub-Fragments*/
21.               add (subFrags , TBList) /* add subFrags to TBList to continue  dividing the
22.                                          sub-fragement of currentBlock if ETD remains > ECD*/
23.          Else
24.                    Break //no more blocks in the TBList to be divided and pruned
25.          End if
26.    End while
27.    return BList
End
```

Algorithm 7-2

Algorithm: prune

This algorithm is called by the DeletingDataBlocks algorithm. It tests the possibility of pruning the block passed as parameter. If the block can be pruned, the algorithm deletes it from the BList, otherwise it does nothing.

Input

BList: array of the data blocks carried by the node (type: BlockList)
neighborTable: table containing the neighbors of the node (type: NeighborTable)
idOvl: identifier of the overlay (type: Integer)
block: block to be pruned if possible (type: Block)
M: master node id, i.e., source of the request (type: Integer)

Output

BList: BList of the node after pruning the block if it is possible (type: BlockList)

Variables:

N: temporary variable (type: Neighbor)

Begin

```
1.    For (each node N in neighborTable) do
2.          If (N.getOverlayID() = idOvl) then
3.                If (exist block or its ancestor or all its sub-fragments in N.getBList() )
```

[41] See Algorithm 7-3

```
4.                          and
5.                  (ETD(N, N.getBList())<ECD(N, M)) then // block is transferable by N
6.                      BList.deleteDataBlock(block)
7.                      Break //go out the loop For
8.                  End if
9.              End if
10.  End for
11. Return BList
End
```

Algorithm 7-3

Algorithm: divideAndPrune

This algorithm is called by the deletingDataBlocks algorithm. It partitions the currentBlock passed as parameter into sub-fragments and then calls Prune algorithm for each sub-fragment. The algorithm returns the list of sub-fragments of currentBlock that cannot be pruned.

Input

neighborTable: table containing the neighbors of the node (type: NeighborTable)
idOvl: identifier of the overlay (type: Integer)
currentBlock: block that is to be divided and pruned if it is possible (type: Block)

Output

TSubFragments: list containing the remaining sub-fragments after partitioning currentBlock and pruning the possible fragments (type: BlockList)

Variables:

subFragments: temporary list containing the sub-fragments (BlockList)

```
Begin
1.  TSubFragements  = null
2.  If (currentBlock.isDivisible()) then
3.          subFragments = currentBlock.divide()
4.          TSubFragments = subFragments
5.          For each currentBlock in subFragments do
6.              TSubFragements = Prune (TSubFragments, neighborTable, idOvl,
7.                                      currentBlock )
8.          End for
9.  End if
10. Return TSubFragements //return the sub-fragments of currentBlock that cannot be pruned
End
```

7.4.3.2. Replicating Data Blocks Operations: Replicate / DivideAndReplicate

In CoFFee, a node triggers a replication operation when one of the following two conditions is verified:

- If the node detects that its BList contains rare data blocks, i.e. data blocks that are carried only by itself. Such a replication operation contributes considerably in improving the data availability in the network. The rarity of a data block is detected through the lists of fragments identifiers exchanged by the overlay nodes.

- If the estimated transmission duration (ETD) of the BList remains exceeding the estimated connectivity duration (ECD) even after applying the Deleting blocks operations and if some data blocks remaining in the BList are not carried by other nodes capable of completing the transfer to the master node. In this case, the replication of some block(s) from the BList is applied; replicas are placed on node(s) that are able to communicate and to transfer the blocks to the master node. Once the replication is applied, the replicated blocks are removed from the BList of the node in order to decrease its ETD.

The first step in the replication operation consists in searching in the overlay for Blank nodes. Blank nodes are identified from the periodic messages exchanged by the overlay nodes, which means that all the overlay nodes have some knowledge of the BList of other neighbor overlay nodes.

To be selected, a to carry a replica Blank node (named BN) must verify three conditions: (Figure 7-3)

- The estimated connectivity duration between the block provider (i.e. the replication initiator) named N and BN is sufficient to transfer the replica.

- The estimated connectivity duration between BN and the master node is sufficient to transfer the replica.

- BN meets N early enough before meeting the master mode so that they can complete the replica placement process.

Two states of Blank nodes are distinguished: **totally available** nodes and **partially available** nodes. The "totally available" status means that the connectivity duration between the node wishing to replicate and the Blank node is greater than the transmission duration (the first condition is verified). Therefore, the node executes the **Replicate** action; i.e. the replica is fully transferred in unicast manner to the Blank node. The "partially available" status means that the connectivity duration is insufficient for the Blank node to receive the entire replica. In this case, the node executes the action **DivideAndReplicate**: it divides the data block into sub-fragments and **replicates** only as many sub-fragments as allowed by the connectivity duration with the partially available Blank node.

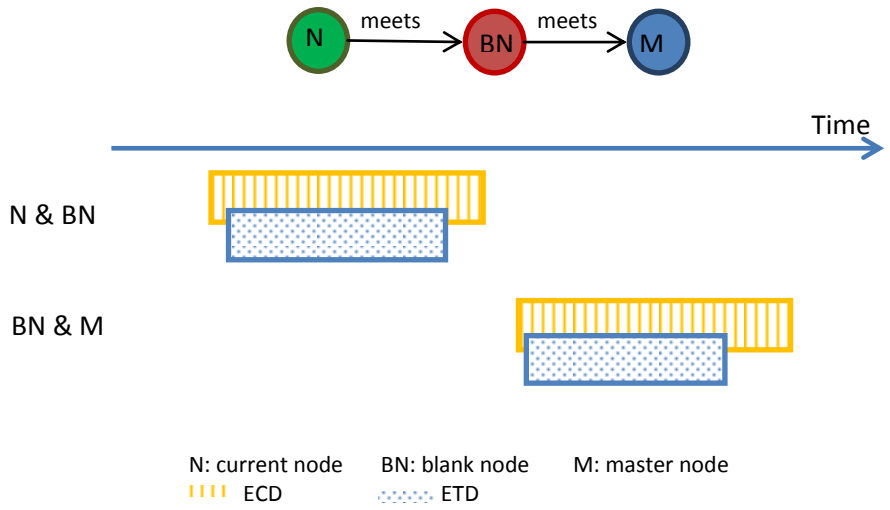


Figure 7-3: Selecting the Blank node for replication

Before presenting the replication algorithm, we present below the recursive algorithm that searches for a Blank node in the overlay.

Algorithm 7-4

Algorithm: searchBlankNode

This algorithm is executed by a node of the overlay to search for a blank node in its neighborhood. If the algorithm finds a blank node to place the block dB, it returns the node, otherwise it returns Null. This algorithm is called by the algorithm **replicate**() for each block that the node wants to replicate.

Input:

neighborTable: list containing the neighbors of the node (type: NeighborTable)
idOvl: identifier of the overlay (type: Integer)
M: master node that sent the request (type: Integer)
dB: data block to be replicated (type: Block)

Output:

BN: the id of blank node if it exists, null otherwise (type: Integer)

Variables:

subFrag: temporary variable that contains the sub-fragments of the current block (type: BlockList)
K: temporary variable that contains a node id.

```
Begin
1.  BN = null
2.  For ( each K in neighborTable ) do
3.          //verify that the neighbor node is in the same overlay
4.      If  ( K.getOverlayID() = idOvl &  //both nodes are in the same overlay
5.          K.getBList().isEmpty() &  //K doesn't have blocks in BList → blank node
6.           STC[42]( K, M) > ETC[43]( K, this)) & //it  meets M after completing replication
7.           ETD(K, {dB})< ECD(K, M) & ETD(this, {dB})<ECD(this, K) then
8.                  BN = K
9.                  Break //blank node is found → end the loop for
10.     End if
11. End for
12. Return BN
End
```

The replication algorithm, named Replicate is presented below.

Algorithm 7-5

Algorithm: replicate

This procedure is called after pruning all possible blocks from the BList. It searches a blank node to host a block. If a satisfying blank node cannot be found, it searches a blank node to

[42] STC(node, Q) returns the start time of the connectivity between node and Q

[43] ETC(node, N) returns the end time of the connectivity between node and N

replicate a sub-fragment of the block. The algorithm stops when the full BList can be transferred by the node during its communication with the master node or when no further replication action is possible.

Inputs

BList: list of the data blocks of the node (type: BlockList)
neighborTable: list containing the neighbors of the node (type: NeighborTable)
idOvl: identifier of the overlay (type: Integer)
M: id of the master node i.e. the node that sent the request (type: Integer)

Outputs

BList: list of the data blocks of the node after replicating and deleting possible blocks; it contains the remaining blocks to send to M (type: BlockList)

Variables

TBList: temporary list that contains at first the same blocks than BList; it will be used to eliminate the processed blocks (type: BlockList)
currentBlock: temporary variable (type: Block)
BN: blank node (type: Integer)
subFragments: temporary variable that contains the sub-fragments of the currentBlock (type: BlockList)

Begin

```
1.  TBList = BList
2.  While ( ETD (this, BList) > ECD (this, M) ) do
3.          /*ECD(this,M) return the connectivity duration between the current node and the
4.          master node M */
5.          currentBlock = TBList.getDataBlock() //return&delete the first block from TBList
6.          If (currentBlock != null) then
7.                  BN = searchBlankNode ( neighborTable, idOvl, M, currentBlock)
8.                  If BN != null then  //a blank node is found to replicate dB
9.                          sendReplicationRequest( BN, currentBlock )
10.                         BList.deleteDataBlock( currentBlock )
11.                 End if
12.         Else
13.                 Break //no more blocks in BList to be replicated, go out of the "while" loop
14.         End if
15. End while
16. //if ETD is still > ECD try the DivideAndReplicate action
17. TBList = BList
18. While ( ETD (this, BList) > ECD (this, M) ) do
19.         currentBlock = TBList.getDataBlock()
20.         If (currentBlock != null) then
21.                 If ( currentBlock.isDivisible() ) then // true if the block can be divided
22.                         subFragments = currentBlock.divide()
23.                         For each block in subFragments do
24.                             BN = searchBlankNode ( neighborTable, idOvl, M, block)
25.                             If BN != null then  //a blank node is found to replicate dB
26.                                     sendReplicationRequest( BN, block )
27.                                     subFragments.deleteDataBlock( block )
28.                                End if
29.                         End for
```

```
30.                         //replace currentBlock with the non-replicated fragments in the BList
31.                          replace (currentBlock, subFragments, BList)
32.
33.                          //remaining fragments (non-replicated) are added to the TBList
34.                          TBList.addBlocks(subFragments)
35.                 End if
36.         Else
37.                 break //no more blocks in the BList to be replicated, thus end the loop while
38.         End if
39. End while
40. Return BList
```

7.4.4. Communication with the Master Node

In this work, only one-hop communication is considered between the nodes; it is also assumed that content distribution systems are delay tolerant systems (this means that the system can tolerate some waiting delay (carry and wait strategy)). However, in real time content distribution systems (e.g. online gaming, real-time streaming), it may be more efficient to create a multi-hop communication in order to speed up the transmission mechanism and consequently reduce end-to-end delays.

The one hop communication framework proposed in this work is established between the overlay provider nodes and the master node. That is, the data transmission process between an overlay provider node and the master node starts when they become within the transmission range of each other.

When a node N receives a request, it joins the overlay of the master node if it can communicate with it in the near future. Before meeting the master node, N prepares its BList and estimates the transmission duration. If ETD is greater than ECD, the node applies the collaboration operations presented above in a periodical manner. When N becomes a direct neighbor of the master node, it applies the following steps (see Algorithm 7-6):

- N sends a transmission request to the master node, M containing its BList

- M sends back to N the **ToSendList** that contains the data blocks existing in the BList after eliminating the blocks already transmitted by other nodes of the overlay.

- N receives the **ToSendList**. If the ETD of the ToSendList is greater than the ECD despite the applied collaboration operations, N selects a list **FinalList** of blocks from the ToSendList such that the ETD of this list is smaller or equal to ECD. Then it sends the blocks to M. The elimination process removes the largest size blocks first, because it seeks to eliminate the minimum number of blocks (i.e. to transfer the largest number of blocks). However, we work on more advanced solutions (e.g. that eliminate the blocks so that the amount of time needed to transfer these blocks is the closest to ECD-ETD).

Algorithm 7-6

Procedure: sendBlocks

Input
neighborTable: list containing the neighbors of the node (type: NeighborTable)
BList: list of the data blocks of the node (type: BlockList)
M: id of master node, i.e. the node that sent the request (type: Integer)

Variables
ToSendList, FinalList: temporary lists that contain the blocks that the node is to send to M (type: BlockList)

Begin
1. **If** (BList.size() != 0) **then**
2. wait until M is in the communication range
3. send BList to M //*send data blocks' identifiers* to the master node M
4. receive ToSendList from M //wait until receiving message from master node
5. FinalList = ToSendList
6. **While** ETD(this, FinalList) **>** ECD(this, M) **do**
7. delete a block from FinalList
8. **End for**
9. **For** each dB in FinalList **do**
10. sendBlock(dB, M) //send data block (file or a fragment) to node M
11. **End for**
12. **End if**

End

7.5. Performance Evaluation

In this section, we evaluate the performance of CoFFee by a series of simulations. As for CReaM, simulations have been developed in OMNet++ and INETMANET frameworks [3].

7.5.1. Configuration of the Environment

In our experimental scenario, it is assumed that the overlay network is already constructed and it contains 50 selected nodes. We limited the number of nodes to 50, based on the DG-CastoR experimental results mentioned in [7] concerning the number of nodes that verify the spatio-temporal similarity measures. For each node a set of parameters are assigned (cf. Table 7-1).

- **Estimated Connectivity Duration "ECD"**: varies between 1 minute and 5 minutes. These values are deduced from the experimental results of DG-CastoR mentioned in [7]. Three travel durations are considered in the simulations; 10 minutes, 20 minutes and 40 minutes. The Estimated Connectivity Durations as tested in [7] are as follows. For a trajectory of 10 minutes route, 1 minute of ECD is attributed to 70% of the nodes; 2 minutes is attributed to 17%, 3 minutes to 9% and finally 4 minutes to 4% of the nodes. However, for 20 minutes and 40 minutes trajectories routes, the values of ECD vary between 1 minute and 5 minutes; so that, for 20 minutes, 2 minutes of ECD is assigned for 60% of the nodes; finally, for 40 minutes, 5 minutes of ECD is assigned for 52% of the nodes. We distribute the estimated connectivity duration randomly among the nodes.

- **Estimated Waiting Time "EWT"**: estimation of the time the node waits before meeting the master node. The waiting time starts from receiving the query that leads the node to join the overlay network constructed for this query. EWT takes values between 1 minute and the trajectory duration (e.g., for a trajectory of 10 minutes, the EWT varies between 1 minute and 10

minutes, for a trajectory of 20 minutes, the EWT varies from 1m to 20m). The values of the EWT are distributed randomly among the nodes.

- **Estimated Transmission Duration "ETD"**: in order to calculate the ETD, the file is partitioned into equal-sized fragments (cf. section 7.3). For the sake of simplicity, we assume that the file is partitioned only to one level, and the size of a fragment is fixed to 25MB (to facilitate the calculation of the complete transfer). Fragments are distributed on the node according to a uniform distribution. Each fragment has the probability of 1/n of being chosen among the n fragments of the entire file, where n depends on total size of the file. In this distribution the probability for a fragment not to be distributed exists.

In the following experimentations, the data transfer is carried out according to the FIFO scheduling algorithm, the first transferred block being the first block in the BList.

Table 7-1: Simulation parameters

Nodes number	50
Transmission range	100 pixels
Region	800x800 pixels
ECD	varying between 1 min and 4 min
Transfer rate	3 Mbps (6 Mbps theoretical)
Fragment size	25MB
Iteration number (simulations)	30

7.5.2. Percentage of Complete Transfer

The rate of the transferred fragments represents the number of received fragments over the entire necessary fragments needed for a complete transfer. An end-to-end transfer for an entire file represents a rate of 100%. In this section, we measure the complete transfer rate for a multimedia file, by varying the size of the file (300MB, 600MB, 700MB and 900MB). We consider that each file is partitioned into equal sized fragments (25MB). The simulation is repeated 30 times, the number of nodes in the network is fixed to 50 and their velocity is fixed to 50km/h by considering that the

vehicles are moving in a city. The travel time is considered as 10 min. Note that, according to the experimentations in [7] and considering a road topology (city map), the average ECD for 10 minutes of travel varies between 1 minute and 4 minutes.

The results of the simulation are illustrated in the Figure 7-4

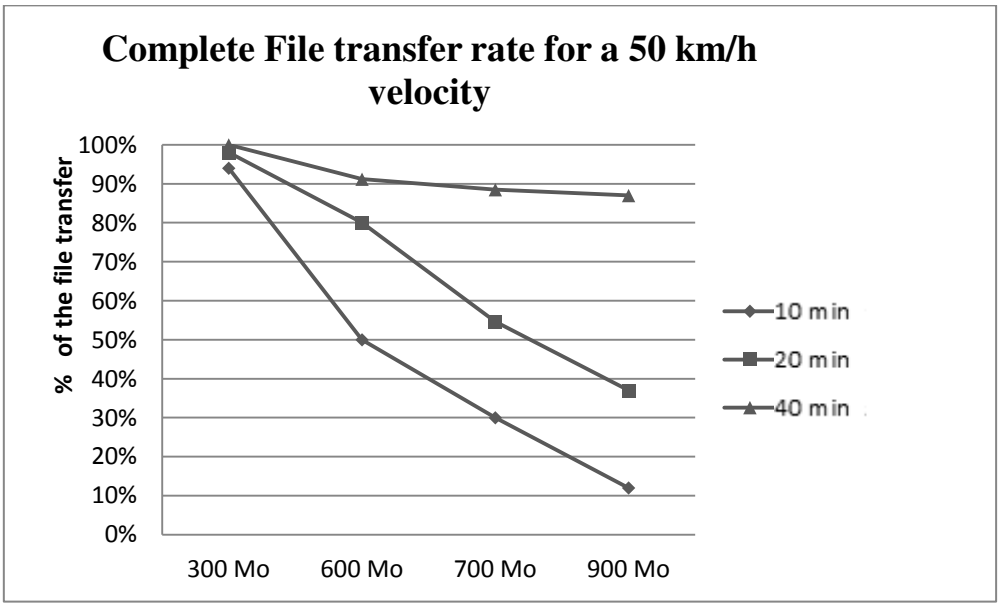


Figure 7-4: Complete File transfer rate for a 50 km/h velocity

The travel duration is an essential parameter in the evaluation of the file transfer rate. Indeed, the ECD between the nodes is distributed between 1 min and 4 min. This distribution is assigned according to the travel duration (10 min, 20 min and 40 min). The simulation result of Figure 7-4 shows that when the nodes travel 10 minutes with a 50km/h velocity and with an ECD varying between 1 min and 4 min, the file transfer rate is high for a file not exceeding 300MB; however, the transfer rate is reduced considerably when the file to be transferred reached 900MB. Note that, for 900 MB (partitioned into equal sized fragments of 25MB), it is not very realistic to consider a travel duration of 10 minutes; it is more realistic to consider 40 minutes route. As illustrated in the Figure 7-4, for a travel duration of 40 min (so ECD equals to 5 min for 52% of the nodes) the transfer rate reaches up to 88% for 900MB file which is very satisfactory.

7.5.3. Effect of the Number of Blank Nodes in the Overlay

These series of experimentations tend to evaluate the effect of the existence of blank nodes in the overlay network. In DG-CastoR, the overlay network is constructed from the nodes that have received the query by selecting those that have an acceptable connectivity duration with the master node. In this section, we considered a travel duration of 20 min and a velocity not exceeding 50km/h. We recall that the blank nodes of an overlay do not contain a response to the disseminated query; however, they are used mainly in the replication process: when a node detects a need to replicate a fragment, it searches for blank nodes that can carry the replica (see Algorithm 7-5).

In these simulations, in addition to a uniform distribution of the fragments, one of the nodes in the overlay is assumed to have the entire file (i.e. it is a "seed node"). This assumption guarantees that all the fragments composing the requested file are available in the network. In addition, in the simulated scenario, the number of blank nodes varies between 10%, 20% and 50% of the total number of nodes. The simulation results are illustrated in the Figure 7-5.

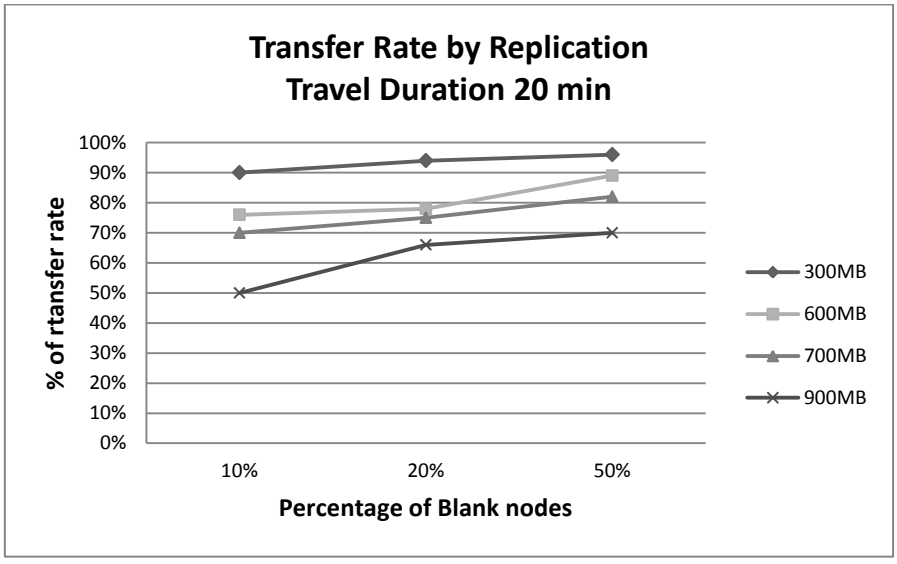


Figure 7-5: The effect of the Blank nodes

The results displayed in Figure 7-5 show that the transfer rate is significantly improved when the overlay contains blank nodes. This could seem surprising as the number of block provider nodes is by definition less when some nodes are Blank. This is because of the replication operation applied on the fragments that places replicas only on blank nodes. So when the number of blank nodes increases in the network, more replication operation can be executed; which tends to improve the fragments transfer to the master node. Note that, for this simulation scenario, it is assumed that all the fragments are available in the overlay and one node is assumed to carry the entire file (seed node). This allows the very effective performance of CoFFee, due in particular to the proposed cooperative improvements, specifically to the replication, that allow adapting the planed transfer of the file blocks to the expected bandwidth.

7.6. CoFFee and CReaM

The replication mechanism used in CoFFee is reactive; it is performed on demand, when the application needs it to answer data requests. Indeed, the replication is initiated when a node is unable to transfer the content to the master node because their connectivity duration is not sufficient. However, in cases where no request is propagated, or when the nodes holding the content (e.g. response file) can transfer it to the master node, the replication is never triggered.

CReaM on the other hand implements a proactive scheme, which is triggered independently from the user requests. The replication is triggered to increase the data availability and to preserve the device's resources. It is not used to answer a specific data request.

Our aim is to use CReaM together with CoFFee to enhance the performance of CoFFee. Each node that deploys CReaM could increase the availability of the data items (file or fragments of files) considered hot and rare in the network, even before sending a specific data request and constructing the overlay used by CoFFee. Thus, the nodes would need to apply much less cooperative improvements, and hence, all

nodes could profit from the communication duration with the master node to transfer blocks. Consequently, the percentage of successful requests is expected to increase.

However, to deploy CReaM on top of a VANET and to make it work in combination with CoFFee, some implementation adaptations must be considered. Below, we give only some proposals that we could not develop due to time constraint, but that we plan to develop as future work:

- Triggering replication: currently the replication is triggered due to a monitoring process of the resources availability. In addition to the three resources (CPU, battery and storage space), the bandwidth could be considered, so that replication starts when CReaM detects available bandwidth.

- Selecting block(s) for replication: now in CReaM the selection of the data item for replication is based on the rarity of a data item where the definition of the rarity is based on the semantics of the content and is related to the nodes' interests. In the context of multimedia content sharing, the rarity of a data item (fragment) could be defined to consider also the availability of the fragments of a file. Hence, a fragment could be considered as rare in case all other fragments are available. For example, assume that two files F_1, F_2 are divided into 4 fragments; all F_1 fragments are available except the first one F_{11}, while only F_{21} is available from F_2. In this case, CReaM should favor the replication of F_{11}.

- Selecting the replica holder: in addition to the resources availability and to the potential value for the replica's subject, CReaM could consider the trajectory of the nodes when selecting the replica holder. Thus, it could place replicas for the blocks that form one file on nodes that have similar trajectories in order to increase later (i.e., when sending a request and constructing the overlay) the number of the nodes in the overlay that can communicate with the master node. This solution could increase the chance to answer the requests and reduces the need for reactive replication triggered by CoFFee.

7.7. Conclusion

In this chapter, CoFFee a multimedia content P2P distribution system for VANETs has been presented. CoFFee is a decentralized system tailored for highly mobile environments. It tends to guarantee an efficient multimedia file distribution among mobile nodes. For this purpose, at the application layer several distribution techniques are implemented; mainly pruning operations that reduce the redundant fragments (file blocks) in the network. In addition, the data availability issue is also considered in CoFFee through replication operations that aim to increase the rare fragments in the network. In the first series of simulations, the results have been shown the efficiency of CoFFee in terms of transfer rate, thanks to its efficient distribution techniques. In addition, in the second series of simulations, the improvement ensured by the replication was evaluated. Thus, in an overlay network containing blank nodes (i.e., nodes that have no data blocks part of the answer to the query), the transfer rate still presents very good results and is even improved because of the replication mechanism applied on the most critical fragments (i.e., the fragments that are hard to transfer) that places these fragments on blank nodes. The content distribution and transfer adaptation techniques proposed in this chapter are mainly based on the cooperation of the nodes within a same overlay. Their effectiveness, illustrated by the results of the experiments, demonstrates the relevance of cooperative mechanisms for implementing applications on top of VANETs.

CONCLUSION AND FUTURE WORKS

Due to the mobility of the nodes and their frequent disconnections in mobile ad-hoc networks (MANETs), the availability of data is inherently lower in MANETs than in fixed networks. Data availability is therefore a critical issue for MANETs since it impacts almost all applications running on top of them. In this thesis, we have dealt with the problem of data availability from two perspectives. In the first part of the thesis, which is the main part of our research work, we have proposed a replication model for MANETs, called CReaM. It bases on the resource usage, user's interests and documents' semantic properties to achieve an efficient replication adapted to mobile networks. In the second part, we have proposed a multimedia content exchange system for VANETs, called CoFFee. It addresses mainly the problem of distributing multimedia contents in highly mobile networks. In addition, it uses a replication mechanism to increase the data availability and so, improve the Quality of Service of the system.

8.1. Summary of Contributions

8.1.1. CReaM

CReaM is a **fully-decentralized replication model**. It is tailored to face the limitations of mobile ad-hoc networks. To this end, it is designed based on two ideas that we deemed promising in the context of MANETs: **efficient resource usage**, and exchange of information about **user interests**.

1. The first idea is straightforward due to the resource-constrained nature of MANETs. Thus the model aims to use replication as a tool to increase the data availability and to preserve enough resources for further user's needs. To this end, we have defined a **monitoring process** to observe the level of the main resources of a mobile node, namely the CPU, the battery and the storage space. This monitoring triggers replication process when the level of a resource becomes insufficient with the goal of sharing the load of data requests with the other nodes of the MANETs. Replicas are placed on nodes where no resources consumption problem exists. In this regard, we have gone beyond a pure replication model by defining actions other than

replication when critical resource problems are identified. Such actions include selectively processing incoming requests when the node is overloaded or selectively processing incoming replication requests. The model has been generalized by mapping the possible actions with the resource problems using **Event-Condition-Actions** rules and an **action conflict resolution** process has been developed to deal with cases in which several resource problems occur simultaneously and their corresponding actions are contradictory.

2. The second idea is originated from the decentralized nature of the model. The absence of a central point makes taking the replication decisions based on a complete view about other nodes of the network not realistic. Thus, the replication decisions are made considering only the knowledge that the node can obtain. However, replication decisions (especially the replica allocation) are critical factors for the benefit provided by the replication system. Thus, we have determined the additional information that could be exchanged to significantly improve the quality of the decisions while causing minimal overhead. To this end, we have proposed to use the **user interests** and the **semantics properties** of the documents.

The first use of this information is done during the selection of a data item to replicate. CReaM favors the replication of data items that are identified as rare in the network and that also correspond to popular subjects that interest the users. Such replication improves the availability of those data items. The second and the main use of this information corresponds to the replica placement. The idea is that a data item should best be placed on a node with a very good connectivity with nodes that are interested in the data item's topics. Indeed, placing replicas near interested users, in addition to increase the availability, decreases the response time for data requests and decreases the resources (bandwidth, power) used to process them.

We defined a protocol to enable the information about the user interests to be disseminated in the network with minimal overhead. This information is then used to select the most promising replica holder. We defined the **community of interest** as the set of nodes that are interested in a topic; the replica placement problem then amounts to finding the "center" of this community.

We evaluated CReaM using the OMNET++ simulation framework along with the INETMANET package. We measured the performance using three metrics: data availability, overhead and user satisfaction. The simulation results show that CReaM significantly improves the data availability and the number of satisfied queries. Compared to a periodical model, the overhead caused by CReaM is dramatically lower. These results tend to show the effectiveness of the proposed solution.

8.1.2. CoFFee

In addition to the data availability, the problem of exchanging large-size data among mobile nodes has been addressed in this thesis. This problem is particularly challenging when the mobility of the nodes is high as in vehicles networks (VANETs). In this context, CoFFee, a multimedia content distribution system for VANETs, has been proposed. Its main objective is to improve the transmission of multimedia content among mobile nodes. For this purpose, first a mechanism has been proposed to allow **partitioning** multimedia file into blocks; therefore, nodes **cooperate** among them to transfer blocks instead of the whole file in cases transferring the whole file is in not possible. Two cooperative distribution techniques have been proposed: (1) **pruning** operations to reduce the redundant blocks during the transfer operation, so that the bandwidth usage is optimized. (2) **replication** operations to increase the availability of rare blocks. The replication mechanism applied in CoFFee aims to involve all available nodes in the virtual network (notably "Blank" nodes that have no content to transfer) to complete the transfer.

The simulation results of CoFFee prove significant improvements in the transfer rate. In addition, in scenarios where a limited number of nodes contain the requested file, the transfer rate is dramatically enhanced by using available Blank nodes to replicate rare blocks of the file. These results illustrate the relevance and the efficiency of the collaborative data distribution mechanisms proposed by CoFFee.

8.2. Future Work

Below are some of the envisaged future works. Some of them are needed to complete the functionalities of our model. The others can be considered as a continuation to this thesis.

8.2.1. CReaM

The short term future work concerning CReaM includes:

- **Dynamic adjustment to the tolerance thresholds**

The tolerance thresholds in CReaM are used to control the allowed usage of the resources. Our aim is to adjust dynamically these tolerance thresholds based on the current load caused by the user actions. When the user needs more resources to complete his personal tasks, the system should adapt the thresholds to reserve more resources for him, e.g. when the user needs more storage space, the threshold δ should be decreased to use less storage for replicas. On the contrary, when the user is inactive or less active, the node should participate more in the system. However, a relation between the consumption of the three resources exists; e.g. when the user needs more CPU, not only the threshold β is adjusted but also the threshold of the battery level γ is concerned.

- **Experimentations**

The simulation study that has been conducted to evaluate the performance of CReaM does not include the calculation of the Potential values. We aim to develop another series of experimentations in order to evaluate the complete model. In addition, since the model uses different thresholds (e.g. the hotness threshold to select the hot data items, the threshold to determine that an External Potential is significant, the number of External Potentials kept for each subject), we want to study the effect of these thresholds and fix the best values.

Furthermore, we think that the performance of the replication system could be affected by the characteristics of the network (e.g. high mobility, network density, obstacles). We aim to evaluate the performance of CReaM in different situations especially in highly mobile networks.

As for the long-term future work, we aim to studding the following issues that are a continuation of this thesis:

- **Optimizing the replica placement process**

Now, the selection of the candidate replica holder is based on the available knowledge concerning the communities of interests around the nodes. However, replication request can be refused if the status of the candidate node does not allow processing the placement request or if it has already the candidate data item. In both cases, the candidate replica holder refuses placing the replica and applies the placement correction process. To enhance the selection process, two propositions can be studied. The first one is to extend the protocol that exchanges the potential values to include minimal information about the resources of the node status. This information should help the node to eliminate nodes that are expected to refuse the placement. The second proposition is a lightweight protocol that uses the information propagated in the network to deduce the distribution of the data items over the nodes. For example, monitoring the different data requests, data responses, and replication requests that reach through the node, should allow the node to infer the location of some data items. Thus, the node could avoid sending a replication request to a node that already has the replica. Both propositions should help in eliminating nodes that are selected by the actual algorithm (i.e. the algorithm that only uses the potential values). This elimination should significantly decrease the risk of refusing the replica placement by the candidate holder and should consequently optimize the replica placement process.

- **Generic replication model**

Our objective is to propose a generic replication model that can be adapted to different environments (e.g. MANETs, VANETs). For each environment, a model could be obtained as an instantiation of the generic model. Based on the context, some

rules would be activated/deactivated, and some parameters tuned, so that the replication decisions are adapted to the context. For example, when the replication model should be deployed on top of a VANET, the "best" place to create a replica can be influenced by the trajectories of the participants.

8.2.2. CoFFee

- **Intelligent algorithms for data blocks elimination**

In the current version of CoFFee, we have adopted a random method to select data blocks from the BList (1) for the improvement algorithms (pruning and replicating) and (2) for the transmission process with the master node. To optimize the content sharing, we propose to select the blocks by order of priority.

Two methods can be investigated: (1) One method is to consider the priority of rare fragments present in the BList, with respect to the other fragments; thus blocks that can be transferable by multiple nodes are eliminated before (2) a second method is to select the minimum number of blocks to eliminate so that the sum of their size leads to solve the problem of non-sufficient communication duration.

- **Calculating the estimated transmission duration (ETD)**

The estimated transmission duration is essential for the implementation of the improvement algorithms since the pruning/replicating methods are triggered when the ETD is smaller than the communication duration. In the current version we have used the theoretical capacity of the bandwidth in the ETD calculation, which might be different from the effective bandwidth used during the content transfer. In addition, we have not considered the case where the node participates simultaneously in multiple overlays (i.e. answering multiple requests in the same time). In this case, the bandwidth is shared to complete the simultaneous transfers. We plan to develop existing works for estimating the effective bandwidth based on two parameters: the vehicles density and the number of requests propagated in the network.

- **Semantic overlay construction**

The construction of the overlay occurs when a data request is disseminated in the network. The overlay contains nodes that have the same trajectory and that have fragment(s) of the requested file. As a long-term future work, we propose to make a semantic overlay constructed based on the trajectory and the user interests; the overlay could constructed even before disseminating data request. The nodes of the overlay are interested in the same topic(s) and they replicate fragments of files of that topic; we expect this semantic approach leads to increase the availability of the most relevant data and consequently to increase the percentage of successful requests.

List of Publications

I. International Conferences

- User Centric Replication Approach to Maintain Data Availability in MANET. Z. Torbey, N. Bennani, D. Coquil, L. Brunie. The 4th International ICST conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications "Mobilware". London, UK. 22-24 Juin 2011. Springer, pp 195-208.

- CoFFee: Cooperative and InFrastructure-Free Peer-To-Peer System for VANET. T. Atechian, Z. Torbey, N. Bennani, L. Brunie. The 9th International IEEE Conference on ITS Telecommunication. Lille, France. IEEE, pp. 510-515. 2009.

II. International Workshops

- CReaM: User-Centric Replication Model for Mobile Environments. Z. Torbey, N. Bennani, D. Coquil, L. Brunie. The International Workshop on "Mobile P2P Data Management, Security and Trust "M-PDMST" in conjuction with the 11th International Conference on Mobile Data Management "MDM 2010". Kansas City, USA. 2010. IEEE, pp 348-353.

- Replica Update Strategy in Mobile Ad Hoc Networks. M.M. Nawaf, Z. Torbey. The International ACM Student Workshop on Management of Emergent Digital EcoSystems "MEDES-SW". Lyon, France. ACM, pp. 474-476. ISBN 978-1-60558-829-2. 2009.

III. National Conference

- Performance Evaluation for CReaM: User-Centric Replication Model. Z. Torbey, N. Bennani, D. Coquil, L. Brunie. The 11th International IEEE Conference on New Technologies of Distributed Systems "NOTERE". Paris, France. 2011. IEEE pp 1-8.

- A Decentralized and Autonomous Replication Model for Mobile Environments. Z. Torbey, N. Bennani, D. Coquil, L. Brunie. Dans le 6émes Journées Francophones Mobilité et Ubiquité "UBIMOB'10". Lyon, France. 2010

BIBLIOGRAPHY

[1] "zigbee Website": http://zigbee.org/ Last visit 2012

[2] "monarch" http://www.monarch.cs.rice.edu/cmu-ns.html Last visit 2012

[3] "OMNet++ website" www.omnetpp.org Last visit 2012

[4] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A Review of Routing Protocols for Mobile Ad Hoc Networks. Journal: Ad hoc networks, 2(1):1–22, 2004.

[5] L.A. Adamic and B.A. Huberman. Zipf's law and the internet. Glottometrics, 3(1):143–150, 2002.

[6] C. Apté, F. Damerau, and S.M. Weiss. Automated Learning of Decision Rules for Text Categorization. ACM Transactions on Information Systems (TOIS), 12(3):233–251, 1994.

[7] T. Atechian. Protocole de Routage Géo-multipoint Hybride et Mécanisme d'Acheminement de Données dans les Réseaux Ad Hoc de Véhicules (VANETs). Thèse de doctorat en informatique, INSA de Lyon, 2010.

[8] T. Atechian and L. Brunie. DG-CastoR for Query Packets Dissemination in VANET. In 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008., pages 547–552. IEEE, 2008.

[9] T. Atechian, Z. Torbey, N. Bennani, and L. Brunie. CoFFee: Cooperative and Infrastructure-Free Peer-To-Peer System for VANET. In 9th International Conference on Intelligent Transport Systems Telecommunications (ITST),, pages 510–515. IEEE, 2009.

[10] K.M.V. Banu. Improving Ad Hoc Network Performances by Estimating Available Bandwidth. International Journal on Computer Science and Engineering (IJCSE), 2(08):2589–2592, 2010.

[11] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 76–84. ACM, 1998.

[12] N. Beijar. Zone Routing Protocol (zrp). In Ad Hoc Networking, Licentiate course on Telecommunications Technology, Finland, 2002.

[13] P. Bellavista, A. Corradi, and E. Magistretti. Comparing and Evaluating Lightweight Solutions for Replica Dissemination and Retrieval in Dense MANETs. In 10th IEEE Symposium on Computers and Communications, 2005. ISCC 2005. Proceedings., pages 43–50. IEEE, 2005.

[14] P. Bellavista, A. Corradi, and E. Magistretti. Lightweight Replication Middleware for Data and Service Components in Dense MANETs. In Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005., pages 142–152. IEEE, 2005.

[15] P. Bellavista, A. Corradi, and E. Magistretti. Redman: A Decentralized Middleware Solution for Cooperative Replication in Dense MANETs. In Third IEEE International Conference on Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops., pages 158–162. IEEE, 2005.

[16] P. Bellavista, A. Corradi, and E. Magistretti. Redman: An Optimistic Replication Middleware for Read-Only Resources in Dense MANETs. Journal of Pervasive and Mobile Computing, 1(3):279–310, 2005.

[17] C. Bettstetter, C. Wagner, et al. The Spatial Node Distribution of the Random Waypoint Mobility Model. In German Workshop on Mobile Ad Hoc Networks (WMAN), pages 41–58, 2002.

[18] BlueHoc Simulator, http://www.scis.ulster.ac.uk/~kevin/top_blue.html last visit 2012.

[19] Bluetooth Website: http://www.bluetooth.org Last visit 2012.

[20] BlueWare Simulator: http://nms.lcs.mit.edu/projects/blueware/software/ Last visit 2012.

[21] I.D. Chakeres and E.M. Belding-Royer. The Utility of Hello Messages for Determining Link Connectivity. In Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on, volume 2, pages 504–508. IEEE, 2002.

[22] I.D. Chakeres and E.M. Belding-Royer. AODV Routing Protocol Implementation Design. In Proceedings. 24th International Conference on Distributed Computing Systems Workshops., pages 698–703. IEEE, 2004.

[23] K. Chen and K. Nahrstedt. Integrated Data Lookup and Replication Scheme in Mobile Ad Hoc Networks. In Proceedings of SPIE: International Symposium

on the Convergence of Information Technologies and Communications, volume 4534, pages 1–8, 2001.

[24] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Designing Privacy Enhancing Technologies, pages 46–66. Springer, 2001.

[25] W.W. Cohen and Y. Singer. Context-sensitive Learning Methods for Text Categorization. ACM Transactions on Information Systems (TOIS), 17(2):141–173, 1999.

[26] Car-To-Car Consortium. http://www.car-2-car.org, 2005. Last visit June 2012.

[27] David Coquil. Conception et Mise en Oeuvre de Proxies Sémantiques et Coopératifs. Thèse de doctorat en informatique, INSA de Lyon, mars 2006.

[28] STUTZBACH Daniel and REJAIE Reza. Characterizing the Two-Tier Gnutella Topology. In ACM SIGMETRICS Performance Evaluation Review, pages 402–403, 2005.

[29] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, and A. El Abbadi. Constrained Nearest Neighbor Queries. Advances in Spatial and Temporal Databases, pages 257–276, 2001.

[30] A. Ferreira. Building a Reference Combinatorial Model for MANETs. IEEE Network, 18(5):24–29, 2004.

[31] A. Gelbukh, G. Sidorov, and A. Guzmán-Arenas. Document Indexing with a Concept Hierarchy. Computación y Sistemas, 8(4):281–292, 2005.

[32] GPX TopoGrafix: http://www.topografix.com/ Last visit 2012.

[33] Hannes Hartenstein, Bernd Bochow, André Ebner, Matthias Lott, Markus Radimirsch, and Dieter Vollmer. Position-Aware Ad Hoc Wireless Networks for Inter-Vehicle Communications: The Fleetnet Project. In the Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 259–262. ACM, 2001.

[34] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2001. Proceedings. IEEE, volume 3, pages 1568–1576. IEEE, 2001.

[35] T. Hara, Y.H. Loh, and S. Nishio. Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks. In Proceedings of the 14th International Workshop on Database and Expert Systems Applications, pages 969–973, 2003.

[36] T. Hara, N. Murakami, and S. Nishio. Replica Allocation for Correlated Data Items in Ad Hoc Sensor Networks. ACM SIGMOD Record, 33(1):38–43, 2004.

[37] Farah Harrathi, Catherine Roussey, Sylvie Calabretto, Loïc Maisonnasse, and Mohamed Gammoudi. Indexation Sémantique Des Documents Multilingues. In INFORSID, editor, Atelier RISE associé au 27ème Congrès INFORSID, pages 31–50, May 2009.

[38] Johannes Hautz and David Coquil. Simulation of Mobile Ad-Hoc Networks for User-Centric Replication. Master's thesis, Passau University, 2009.

[39] X. Hong, M. Gerla, G. Pei, and C.C. Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pages 53–60. ACM, 1999.

[40] H. Hu and D.L. Lee. Range Nearest-Neighbor Query. IEEE Transactions on Knowledge and Data Engineering, 18(1):78–91, 2006.

[41] J.L. Huang and M.S. Chen. On the Effect of Group Mobility to Data Replication in Ad Hoc Networks. Journal: IEEE Transactions on Mobile Computing, 5(5):492–507, 2006.

[42] J.L. Huang, M.S. Chen, and W.C. Peng. Exploring Group Mobility for Replica Data Allocation in a Mobile Environment. In Proceedings of the twelfth international conference on Information and knowledge management, pages 161–168. ACM, 2003.

[43] H. Hyotyniemi. Text Document Classification with Self-Organizing Maps. *STeP'96, Genes, Nets and Symbols*, pages 64–72, 1996.

[44] STOICA Ion, MORRIS Robert, LIBEN-NOWELL David, and et al. Chord: Scalable Peer-To-Peer Lookup Protocol for Internet Applications. In: IEEE/ACM Transactions on Networking (TON), pages 17–32, 2003.

[45] D.B. Johnson, D.A. Maltz, J. Broch, et al. Dsr: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. Ad hoc networking, 5:139–172, 2001.

[46] Khaleel Ur Rahman Khan, Rafi U. Zaman, A. Venugopal Reddy, K. Aditya Reddy, and T. Sri Harsha. An Efficient DSDV Routing Protocol for Wireless Mobile Ad Hoc Networks and its Performance Comparison. European Symposium on Computer Modeling and Simulation, UKSIM, 0:506–511, 2008.

[47] D.E. Knuth. The art of computer programming: Generating all trees: history of combinatorial generation, volume 4. addison-Wesley, 2006.

[48] Y.B. Ko and N.H. Vaidya. Geotora: A Protocol for Geocasting in Mobile Ad Hoc Networks. In Proceedings. 2000 International Conference on Network Protocols., pages 240–250. IEEE, 2000.

[49] Y.B. Ko and N.H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. Wireless Networks, 6(4):307–321, 2000.

[50] D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In Proceedings of the Fourteenth International Conference on Machine Learning, San Francisco, CA, USA. PP 170—178, 1997.

[51] A. Krifa, M.K. Sbai, C. Barakat, and T. Turletti. Bithoc: A Content Sharing Application for Wireless Ad Hoc Networks. In IEEE International Conference on Pervasive Computing and Communications, PerCom2009, pages 1–3. IEEE, 2009.

[52] P. Kuosmanen. Classification of Ad Hoc Routing Protocols. Finnish Defence Forces, Naval Academy, Finland., 2002.

[53] L.M. Leemis and S.K. Park. Discrete-event simulation: A first course. Pearson Prentice Hall, 2006.

[54] L. Limam, D. Coquil, H. Kosch, and L. Brunie. Extracting user interests from search query logs: A clustering approach. In Workshop on Database and Expert Systems Applications (DEXA), 2010 , pages 5–9. IEEE, 2010.

[55] B. LIU. Next Generation Vehicular Traffic Management Enabled By Vehicular Ad Hoc Networks and Cellular Mobile Devices. PhD thesis, University of California, 2011.

[56] A. Mondal, S. Madria, and M. Kitsuregawa. CLEAR: An Efficient Context and Location-Based Dynamic Replication Scheme for Mobile-P2P Networks. In Database and Expert Systems Applications, pages 399–408. Springer, 2006.

[57] A. Mondal, S.K. Madria, and M. Kitsuregawa. QoS-Based Dynamic Replication in Mobile Peer-To-Peer Networks. In Proceedings of Data Engineering Workshop DEWS (ISSN 1347-4413) 2006.

[58] A. Mondal, S.K. Madria, and M. Kitsuregawa. CADRE: A Collaborative Replica Allocation and Deallocation Approach for Mobile-P2P Networks. In 10th International Conference on Database Engineering and Applications *Symposium IDEAS'06.*, pages 21–28. IEEE, 2006.

[59] A. Mondal, S.K. Madria, and M. Kitsuregawa. Ecorep: An Economic Model for Efficient Dynamic Replication in Mobile-P2P Networks. In 13th International Conference on Management of Data COMAD, India. Citeseer, 2006.

[60] A. Mostefaoui and L. Brunie. Sirsale: Un Système d'Indexation et de Recherche de Séquences Audiovisuelles à Large Echelle. Gestion des données multimédias. PP 283-309, Edition Hermes, ISBN : 2-7462-0824-5 2004.

[61] S. Murthy and J.J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. Mobile Networks and Applications, 1(2):183–197, 1996.

[62] NS2 official Website, http://www.isi.edu/nsnam/ns/ Last visit 2012.

[63] OGC KML Website: http://www.opengeospatial.org/standards/kml/ Last visit 2012.

[64] OMNET++ Official Website, www.omnetpp.org Last visit 2012.

[65] One Laptop Per Child, http://one.laptop.org/ Last visit 2012.

[66] Prasanna Pabmanabhan and Le Gruenwald. DREAM: A Data Replication Technique for Real-Time Mobile Ad-Hoc Network Databases. In the proceedings of the 22nd International Conference on Data Engineering *(ICDE'06)*, 2006.

[67] N.W. Paton. Active rules in database systems. Springer Verlag, 1999.

[68] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-Sharing System: Measurements and Analysis. Peer-to-Peer Systems IV, pages 205–216, 2005.

[69] CarTALK Project. The European Project CarTalk 2000. http://www.cartalk2000.net, 2000. Last visit 2012.

[70] The PreVENT Project. The Prevent Project. http://www.prevent-ip.org, 2004. Last visit June 2012.

[71] Rashmi. Rashmi's articles: Mobile Ad-Hoc Network (MANET). http://www.saching.com/Article/MANET-Mobile-Adhoc-NETwork/334, 2008. Last accessed 2011.

[72] Catherine Roussey, Sylvie Calabretto, and Farah Harrathi. Natural Language Processing Method for Multilingual Semantic Indexing. In the proceedings of the 12th International Conference on Applications of Natural Language to Information Systems CNAM. Paris, France. June 2007

[73] E.M. Royer, S.R. Das, E.M. Belding-Royer, C.E. Perkins, et al. Ad Hoc On-Demand Distance vector (AODV) Routing. Mobile Ad Hoc Networking Working Group, Internet Draft. 26 Jan 2004.

[74] E.M. Royer and C.K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. Personal Communications, IEEE, 6(2):46–55, 1999.

[75] S. Saad, J. Tekli, R. Chbeir, and K. Yetongnon. Towards Multimedia Fragmentation. In Advances in Databases and Information Systems, pages 415–429. Springer, 2006.

[76] K. Sarkar, T.G. Basavaraju Subir, and C. Puttamadappa. Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications, chapter Introduction. Auerbach Publications, 2008.

[77] FE Satterthwaite. Generalized Poisson Distribution. The Annals of Mathematical Statistics, 13(4):410–417, 1942.

[78] M. Shinohara, T. Hara, and S. Nishio. Data Replication Considering Power Consumption in Ad Hoc Networks. In International Conference on Mobile Data Management, pages 118–125. IEEE, 2007.

[79] M. Shinohara, H. Hayashi, T. Hara, and S. Nishio. A Data Access Method Considering Power Consumption in Ad Hoc Networks. In 1st International Symposium on Wireless Pervasive Computing, pages PP 193–198, 2006.

[80] J.G. Siek, L.Q. Lee, and A. Lumsdaine. Boost Graph Library: User Guide and Reference Manual. Addison-Wesley Professional, 2001.

[81] RATNASAMY Sylvia, FRANCIS Paul, HANDLEY Mark, and et al. A Scalable Content-Addressable Network. In: International Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, pages 161–172, 2001.

[82] M. Tamori, S. Ishihara, T. Watanabe, and T. Mizuno. A Replica Distribution Method With Consideration of the Positions of Mobile Hosts on Wireless Ad-Hoc Networks. In Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, 2002. ., pages 331–335. IEEE, 2002.

[83] Z. Torbey, N. Bennani, L. Brunie, and D. Coquil. Performance Evaluation for CReaM: User-Centric Replication Model. In 11th Annual International Conference on New Technologies of Distributed Systems (NOTERE), pages 1–8. IEEE, 2011.

[84] Z. Torbey, N. Bennani, D. Coquil, and L. Brunie. User Centric Replication Approach to Maintain Data Availability in MANET. In the processing of the 4th International Conference on Mobile Wireless Middleware, Operating Systems, and Applications, pages 195–208, 2012.

[85] A. Vallur, L. Gruenwald, and N. Hunter. REALM: Replication of Data for a Logical Group Based MANET Database. In Database and Expert Systems Applications, pages 172–185. Springer, 2008.

[86] S.Y. Wang and B. Bhargava. A Fragmentation Scheme for Multimedia Traffic in Active Networks. In Proceedings. Seventeenth IEEE Symposium on Reliable Distributed Systems, pages 437–442. IEEE, 1998.

[87] Wi-Fi website: http://www.wi-fi.org. Last visit 2012.

[88] X. Yu, I.L.J. Thng, and Y. Jiang. Measurement-Based Effective Bandwidth Estimation for Long Eange Dependent Traffic. In Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, TENCON., volume 1, pages 359–365. IEEE, 2001.

[89] T. Atechian and L. Brunie. DG-CastoR: Direction-Based Geocast Routing Protocol for Query Dissemination in VANET. In the Internal Conference

Telecommunications Networks and Systems TNS, Amsterdam, The Netherlands. 2008.

[90] M. Bansal, R. Rajput, and G. Gupta. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501 (Informational), Jan. 1999.

[91] H.T. Cheng, H. Shan, and W. Zhuang. Infotainment and Road Safety Service Support in Vehicular Networking: From a Communication Perspective. Mechanical Systems and Signal Processing, 25(6):2020–2038, 2011.

[92] A. Derhab and N. Badache. Data Replication Protocols for Mobile Ad-Hoc Networks: a Survey and Taxonomy. Communications Surveys & Tutorials, IEEE, 11(2):33–51, 2009.

[93] H.H. Duong and I. Demeure. Proactive Data Replication Using Semantic Information Within Mobility Groups in MANET. Mobile Wireless Middleware, Operating Systems, and Applications, pages 129–143, 2009.

[94] Hoa Dung HA DUONG. Partage de Données en Mode Pair à Pair Sur Réseaux Mobiles Ad Hoc. PhD thesis, l'école Nationale Supérieure des Télécommunications, Paris, Septembre 2010.

[95] M. Hamdy, A. Derhab, and B. König-Ries. A Comparison on MANETs' Service Replication Schemes: Interest Versus Topology Prediction. Recent Trends in Wireless and Mobile Networks, pages 202–216, 2010.

[96] Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. An Overview of Mobile Ad Hoc Networks: Applications and Challenges. In the Journal of the Communications Network, Vol. 3:pp. 60–66, Jully 2004.

[97] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPV4. RFC4728, pages 2–100, 2007.

[98] A. Mondal, S. Madria, and M. Kitsuregawa. LEASE: An Economic Approach to Leasing Data Items in Mobile-P2P Networks to Improve Data Availability. In the proceeding of the Databases in Networked Information Systems, pages 222–231, 2007.

[99] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman. A Survey of Data Replication Techniques for Mobile Ad Hoc Network Databases. The VLDB Journal, 17(5):1143–1164, 2008.

[100] Santin Paolo. Mobility Models for Next Generation Wireless Networks Ad Hoc, Vehicular and Mesh Networks. John Wiley & Sons Ltd, 2012.

[101] R. Prasad and L. Deneire. WPANs to Personal Networks: Technologies and Applications, chapter Mobile Ad Hoc Networks (MANET). Artech House, 2006.

[102] W. Su, S.J. Lee, and M. Gerla. Mobility Prediction and Routing in Ad Hoc Wireless Networks. International Journal of Network Management, 11(1):3–30, 2001.

[103] J.Z. Sun. Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing. In the proceedings of the International Conference on Info-tech and Info-net, ICII Beijing, volume 3, pages 316–321. IEEE, 2001.

[104] Z. Torbey, N. Bennani, L. Brunie, and D. Coquil. CReaM: User-Centric Replication Model for Mobile Environments. In the eleventh International Conference on Mobile Data Management (MDM), pages 348–353. IEEE, 2010.

[105] H. Zhou. "A Survey on Routing Protocols in MANETs". Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, pages 48824–1027, 2003.

[106] G. Cao, L. Yin, and C.R. Das. "Cooperative Cache-Based Data Access in Ad Hoc Networks". Computer Journal, IEEE Computer Society, 37(2):32–39, 2004.

[107] E. Pitoura and P.K. Chrysanthis. "Caching and Replication in Mobile Data Management". Bulletin of the IEEE computer Society Technical Committee on Data Engineering, 30(3):13–20, 2007.

[108] D.T. Ahmed and S. Shirmohammadi. Multi-level Hashing for Peer-to-Peer System in Wireless Ad Hoc Environment. In Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07., pages 126–131. IEEE, 2007.

[109] Wi-Fi Alliance, “Personal, portable Wi-Fi® that goes with you anywhere, anytime” available on: http://www.wi-fi.org/discover-and-learn/wi-fi-direct , Last visit 2012.

List of Figures

List of Tables