uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Anis Zarrad

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Mobile Gnutella-based Network System to Support the Distributed Collaborative Virtual Environment

TITRE DE LA THÈSE / TITLE OF THESIS

Azzedine Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Abudlmotaleb El Saddik

Wail Gueaieb

Walaa Hamouda

Gabriel Wainer

Gary W. Slater

Le Doyen de la Faculte des etudes superieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Mobile Gnutella-based Network System to Support the Distributed Collaborative Virtual Environment

by

Anis Zarrad

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For

Doctor of Philosophy

in

Computer Science

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

University Of Ottawa

Your file Votre référence
ISBN: 978-0-494-73926-6
Our file Notre référence
ISBN: 978-0-494-73926-6

The undersigned hereby recommend to

The faculty of Graduate Studies and Research acceptance of this thesis

A Mobile Gnutella-based Network System to Support the Distributed Collaborative Virtual Environment

Submitted by

Anis Zarrad

In partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

In

Computer Science

Azzedine Boukerche, Ph.D.

(Thesis Supervisor)

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AOI: **A**rea **o**f **I**nterest

CQT: **C**ached **Q**uery **T**able

CVE: **C**ollaborative **V**irtual **E**nvironment

CVE-Profile: **C**ollaborative **V**irtual **E**nvironment profile

EGAN: **E**nhanced **G**nutella for **A**d-hoc **N**etworks

GUID: **G**lobal **U**nique **ID**

GUS: **G**nutella **U**ltrapeer **S**ystem

MANETs: **M**obile **A**d-hoc **Net**works

MCVE: **M**obile **C**ollaborative **V**irtual **E**nvironments

MF: **M**obility **F**actor

MOG-CVE: **MO**bile **G**nutella for **CVE**

MMG: **M**assive **M**ultiplayer **G**ame

MPP: **M**obile **P**eer-to-peer **P**rotocol

NF: **N**etwork **F**actor

NS: **N**etwork **S**imulator

TTL: **T**ime **T**o **L**ive

QID: **Q**uery **ID**

VE: **V**irtual **E**nvironment

Abstract

Collaborative **V**irtual **E**nvironments (CVEs), such as military training simulations, emergency preparedness exercises, and online games have made a considerable impact on both commercial and academic fields over the last few years. Due to the rapidly increasing usage of personal mobile devices and the need of executing CVE applications in environments that have no previous network infrastructure, **M**obile **C**ollaborative **V**irtual **E**nvironments (MCVEs) will become ubiquitous in the future. In such systems, users will share a 3D virtual environment through their mobile devices in an ad-hoc network (MANET) in order to accomplish specific missions. We aspire to develop and deploy CVEs over MANETs using the peer-to-peer model. Both peer-to-peer networks and MANETs exhibit the same features; which include complete decentralization, self-configuration, and self-healing. However, peer-to-peer networks and MANETs operate on different network layers, and may introduce poor performance. Mobile ad-hoc and peer-to-peer networks drive the future of CVEs, and thus form the foundation for this thesis.

Most existing CVE systems are tuned to specific tasks, and their architectures are typically tightly coupled with wired networks and desktop settings, therefore they are not adequate enough for addressing mobile collaborative virtual environments. The evolutionary step in MCVE applications is handling the mobility effect in all its forms, and tackling the poor performance of the ad-hoc network. In this thesis, we explain the main networking issues involved in building scalable MCVEs, and review the most well-known CVE and peer-to-peer systems. We then present a mobility-aware cross-layer approach for MCVE applications; this approach relies on building a dynamic multi-tier overlay network architecture-based Gnutella network. The proposed overlay network will manage mobile devices using a novel rule-based discovery technique, in which users are discovered by their virtual environment interest and proximity. With the help of a caching mechanism and a Gnutella Ultrapeer System to perform a dynamic Ultrapeer selection, we were able to out-perform traditional mobile overlay networks, as shown in our simulation results.

Dedication

This thesis is dedicated to my parents Ali and Najet for all their support and guidance. Thank you.

I wish to thank my wife Mouna and my daughter Dalya for their love. You have a very special place in my heart.

I would like also to extend my thanks to my sisters, Olfa, Leila, and Ines for their encouragement.

Last but not least, many thanks go out to my aunt, Nejla, my father in law, Nejib and my mother in law, Kaouther for their continuous support.

Acknowledgements

I would like to thank my supervisor, Prof. Azzedine Boukerche, for his constant support and encouragement, and the fruitful discussions between us throughout my graduate studies.

I also wish to express my appreciation to Prof. Regina Araujo, and to Dr. Richard Pazzi who made many valuable suggestions and gave constructive advices.

I would also like to extend my thanks to Prof. Nejib Zaguia, to whom I am indebted for his support during my studies.

Finally, I would like to thank all the people who assisted me in completing this work.

Chapter 1

Introduction

1.1. Overview

Collaborative Virtual Environments (CVEs) can be defined as 3D graphic applications that can be shared by many users throughout a network. Over the past few years, a handful of interesting CVE applications have been developed in a variety of domains, ranging from multiplayer games to virtual cities, virtual shopping malls, and various training simulations (i.e. military, industrial, commercial, engineering, medicine, educational classrooms, etc.). A virtual environment typically consists of both, static and dynamic objects; static objects, such as background, terrain, and buildings are prone to unpredictable modifications, while dynamic objects, such as spy planes and vehicles inflict higher demands on the network in terms of bandwidth and latency. Each user is represented as a graphical embodiment called an avatar [85]. According to the avatar concept, dynamic objects can be modified and driven by users; when a user triggers an object action, the action is reflected locally and also at all the other remote users' hosts. This process should occur while incurring an acceptable network delay [23], so that all users will share the same view of the Virtual Environment -VE-, and will therefore sustain the feeling of immersion while interacting in the system. An increase in the number of users logically raises a challenge in updating the world across the network, especially if operating in mobile ad-hoc networks (MANETs) [52].

CVE applications that require no previous network infrastructure (e.g. online games, emergency preparedness scenarios and rescue activities in disaster situations, such as war or earthquakes) constitute the key contributing factor to the dissemination of new generation CVE systems, called Mobile Collaborative Virtual Environments

(MCVEs). All users in a MCVE share the 3D virtual environment on their mobile devices in an ad-hoc network. Due the storage limitation of mobile devices, we adopt the standard partition technique [10]. The virtual environment is segmented into multiple hexagonal cells, also called zones, and where each zone is associated with a unique ID. Activities in zones are grouped into session [7]. A session is an abstract concept; it refers to a group of users interested in the same mission in the VE. Each avatar [85] in the VE has an Area of Interest (AoI) that represents the radius of visibility. Figure 1 shows an overview of a mobile collaborative virtual environment application.

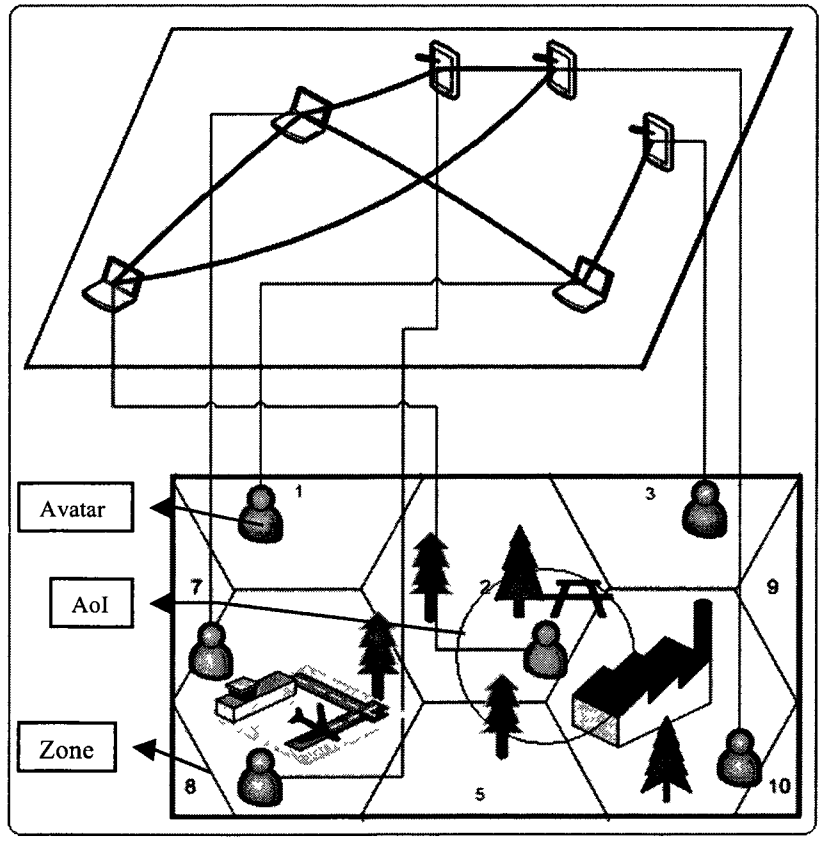

Figure 1: Overview of an MCVE Application

To better understand the impetus of such system, we illustrate a firefighter emergency preparedness scenario [14]. In a fire disaster, many human resources are involved at

all levels of the first-responder organization (firefighters, police officers, and ambulance services). The nature of the problem involves using all available resources efficiently in order to provide responders with a rescue plan that is based on a 3D graphical representation of the accident site while they are heading to their destination. Such a plan is quite advantageous for firefighters because it helps them to better assess the situation, and hence forecast the scenario awaiting them at the site of the accident. The immediacy of the response time, the scarcity of services and resources for most mobile devices, and the lack of any pre-requisite network infrastructure are all reasons for the high interest in such scenario.

In this thesis, we focus primarily on the protocols involved in the establishment of on-the-fly ad-hoc networks between avatars [85] in the virtual environment as a simple and cost effective option to support the collaboration issue. Clearly, the creation of a MCVE system with an independent infrastructure, and which can run anytime and gives rise to hard challenges. Although mobile devices and ad-hoc networks provide users with the convenience of accessing the Internet from any location, they do pose various limitations. For mobile devices, these limitations include constraints on screen size, network bandwidth, CPU, and memory. Ad-hoc networks are error-prone since nodes may disconnect suddenly. Moreover, as more participants join a mobile environment, network traffic increase, causing augmenting network latency. From our point of view, we believe that it will be possible to overcome the aforementioned limitations by introducing a peer-to-peer application running over MANETs (Mobile Ad-hoc NETworks) [52] to serve MCVE applications. Peer-to-peer network based applications [61], [63], and [67] and MANETs [52] have several characteristics and functionalities in common. In essence, both are self-organizing and self-healing with a dynamic topology, thus, peer-to-peer network appears to be a promising tool for creating an efficient MCVE system, albeit such solution introduces more complexity for each node participating in the virtual environment, and generates increased network overhead due to the decentralized nature of the network.

1.2. Requirements of MCVEs

The collaborative virtual environment requirements have been well researched and studied for many numbers of years [61], [62], and [66]. Recent advancements in mobile technologies have put a mobility requirement on traditional CVE requirements. *Mobility* does offer more magnitude to the system, specifically the ability to work autonomously in a mobile environment without losing interest in the CVE application. Classical requirements are outlined in different ways in many studies [73], [81]; here, we present the most important ones. *Efficient communication mechanisms* are necessary to reduce the traffic generated by user interaction messages, so that *large scalability* can be supported without affecting the overall performance of the system or losing interest in the application. In typical online game applications, network latency tolerance [23] usually varies between 10ms to 1000ms. However, it is argued in [108] that network latency in CVE applications should not exceed 100 ms, but may reach a maximum of 200 ms [84] in some cases. The *extensibility of the system at run time* gives the ability to extend and add new features to the virtual environment without interrupting the system during run time. In current systems, any extensibility causes a halt in the application itself, a 'buzz-killer' for critical applications that provide 24-hour service. In our previous work [15], we designed a modular architecture based on a linear story concept that focuses on the *extensibility* requirement. *Persistency* requires that the state of the virtual environment be maintained and carried forward from one session to another. Persistency can be achieved easily with client/server architecture; however, this can lead to bottlenecks and single points of failure that may cause longer delays, and thus cause the system to become less interactive. Hybrid solutions can be used to overcome the centralization problems but they also introduce more management complexity at each node. *Adaptability* is also important in that, since the system involves a large number of users, it must be well adapted to different devices that have different network and hardware capabilities. In mobile environments, adaptability is considered a fundamental requirement due to the limitations of certain mobile devices. *Composability* is the ability to integrate and reuse components from

different applications and environments. *Operability* signifies that the system use standards and patterns in order to allow interactions between heterogeneous systems. The geometric world data of an extensive region such as a battlefield may need to be divided into small areas called zones that are distributed across the network. The distribution of the data helps to reduce host memory use and processing power since less data needs to be received, processed, and stored. Thus, a *good mechanism for data distributed management* [10] is also a crucial requirement for any CVE application. As mentioned above, it is important to note that, in practice, MCVE applications are usually modeled as low-mobility systems since the participating users require concentration and considerable attention in order to interact and collaborate with the VE, to realize their goals of attaining high scores and/or accomplishing their missions successfully.

Attempting to build CVE systems with all of the aforementioned requirements is practically impossible; most of the existing CVE systems [34], [61], [75], and [96] are designed to handle selected requirements in specific network architecture. To the best of our knowledge, it seems that mobility and CVEs are never placed together in order to develop an MCVE system that is able to run anywhere, and establish communications between users using limited network management and administration. In this thesis, we diminish the CVE boundary requirements so as to consider only the mobility, efficient network communication mechanism, scalability, and adaptability requirements in the efforts of developing MCVE without imposing complexity on the resulting approach. The most relevant questions that we must consider are: "Which strategies should be adopted? "; "Is a straightforward implementation of existing peer-to-peer systems [57], [78], [79], [88], and [110] over MANETs satisfactory from the point of view of communication efficiency and persistency aspect? "; "If not, then what modifications should be applied in order to handle the MCVE requirements? ". Alternatively, we may discard the wide variety of existing peer-to-peer systems, and propose a completely new system to meet our

objectives. Another important question is to determine to what extent we should attempt simplicity and flexibility.

Our proposed system has the potential to be one of the better approaches in the research area of mobile collaborative virtual environments. Our developed system should keep a reduced amount of network traffic and latency among a very large number of users participating in a virtual environment so that these users can maintain the feeling of immersion, which is an essential parameter for determining the success of any MCVE application.

1.3. Research Motivation and Background

This study of MCVE applications is motivated primarily by the fact that mobility raises several challenges, including wireless mediums and device limitations. Many researchers have already devoted efforts in building CVE systems for wired networks, with examples including NPSNET-V [75], DIVE [42], and SCORE [66]. At this point however, and to the best of our knowledge, there is no evaluation that explores how both systems (CVE systems and MANETs) act together. A few studies have highlighted the collaboration aspect in mobile environments, namely CSCW (Computer-Supported Cooperative Work) [105], face-to-face collaboration [62], and mobile file-sharing systems [24], [40], and [100], all of which feature applications that exhibit important differences from MCVEs; however, since collaboration in MCVEs is more complex, emphasis is on the user's interest, rather than on the exchange of information using IP addresses. Mobile collaborative virtual environment applications are still in the early stages of development; there is thus a plethora of open problems that remain unexplored, including the effects of node mobility, frequent node disconnection, network capacity, interaction with underlying ad-hoc routing protocols and consistency. Among many other motivations for studying MCVEs originates from the need to apply CVE applications where users are unable to employ conventional infrastructure-based communication devices in an isolated

area due to war, earthquakes, or other disasters. For obvious reasons, it is imperative that such scenarios must be tried using different tactics and various tests (for example, regarding missiles and weapons) be run in computer graphics environments before applying them scenarios in real life; accidents involving real missiles and lethal weapons can very often be deadly, and thus have devastating impact, both economically, and personally.

1.4. Problem Statements

While we are motivated by CVE applications over MANET, we identify the limitations of mobile devices in wireless networks, especially when employed in collaborative activities.

Mobile environments provide lower bandwidth, higher delay, and less reliability than wired networks. This can be an important problem for the obvious reason: update messages must be delivered with an acceptable network delay [84], and [108]. With higher latencies, the user starts losing the feeling of immersion, decreasing his/her rate of interaction, and eventually causing him/her to lose interest to the application.

While the deployment of MANET in peer-to-peer system seems to be a natural choice, straightforward deployment is not efficient, and may lead to failure. The node discovery does not aim to mirror the virtual network to the physical network. Also, overlay network maintenance causes more than 50% of the total amount of network traffic. Periodic update messages such as "heartbeat" or "PING" require the network interface to be on all the time. Continuous CPU usage is reserved to process the messages received, thus affecting network bandwidth and battery lifetime, and in turn, burdening the user with high costs. Moreover, peer-to-peer network operate in the applications layer, while MANETs operates in the network layer.

The unpredictability of MANETs can cause sudden node crashes, which has a devastated effect on the performance of MCVE system as a whole. Nodes have to re-establish connections with neighboring peers, and the virtual environment data must be relocated according to the new physical position of the user in the VE. Often, nodes are able to leave the network over time; however, it is difficult for the virtual environment to remain persistent if nodes disappear, taking with them data that is necessary for the proper functioning of the virtual environment.

Another problem that has not quite been addressed by the existing proposed architectures [51], [63], and [105] is network heterogeneity; considering a large-scale system, many nodes are quite powerful, meaning that they can handle higher workload charges easily; weak nodes on the other hand will have a negative impact on the system as a whole. Thus, the system would either have to specify a minimum level of performance for all participating nodes, or control the amount of data transferred so that weak nodes do not affect the overall system. Both solutions are impractical, since the first one will exclude some users from joining the VE, while the second may probably see some nodes ignoring some VE resources that are vital for the system to function.

Single points of failure will also make the system less interactive. Peer-to-peer networks do not enjoy the convenience of a unique service provider, but rather see other nodes playing the roles of server and client. However, while peer-to-peer network can address some fault-tolerant issues, it also brings some side effects, such as complex network maintenance and network traffic.

In mobile systems, there is a need to merge an ad-hoc routing protocol with the underlying network layer. So far, the impact of routing protocol on the developed must be well analyzed and studied. Aberer *et al* [1] claim that AODV routing protocol is more suitable for low-mobility applications. According to [22] however, Gnutella performs better when proactive ad-hoc routing protocols are used. Barbosa

et al [2] reveal an interesting finding: each routing protocol performs well in some scenarios for some metrics, but suffers from drawbacks in others. This conclusion underlines the importance of identifying the network characteristics (node workload, network size, mobility) before committing to a particular ad-hoc routing protocol. A new proof, accompanied by simulation results analysis is required in order to select the most well-adapted ad-hoc routing protocol for our developed MCVE applications.

1.5. Research Overview

The developed mobile collaborative virtual environment system surpasses the existing system by proposing a Gnutella Multi-tier overlay using a novel cross-layer approach; it takes into consideration the user's interest in the virtual environment, node capability, and node proximity. The system determines whether or not a message can be passed to the application layer. This behavior guarantees less workload and improved network latency. The resulting overlay is designed to match the virtual network to the physical network. ! " #$%&' " (&) *+ , -(.-.' / 0 -") &1 $%2 . -3 %$4- "$5) .-1 6 *- 7) / $8) -$9) %*$ (5) 5-1 6 &0- : %$/) . . 6 "3-8) . . (3) .-#%$8-$&0) %-"$5) .; - <0) %) #$%) , -&0) -! *&% (:)) %u) &1 $%2-.+ .&) 8-: %$&$/$*-6 .68 : *) 8) "&) 5-6$-:) %#$%8-a dynamic Ultrapeer election based on a score measurement that combines the network characteristics and the user behaviors in the VE; we also include a caching protocol to control the message traffic. The caching protocol decomposes the cached data, and also defines query response rules that can be modified at run time to reflect the new position of the user in the virtual environment. The evaluation of the developed system and the performance analysis are done using NS-2 simulator [80].

1.6. Summary of Contributions

As described above, numerous problems constitute dark waters that remain unexplored in the mobile collaborative virtual environment area. We have looked forward to filling the void in prior research in order to conjure up an efficient solution

to supports MCVE applications. The main contributions of this thesis can be summarized as follows:

First, we model a collaborative virtual environment system over MANETs using a Gnutella network. Since we will use an existing peer-to-peer network protocol, it is imperative that we justify the choice of Gnutella protocol. We conduct separate analyses of the variety of available peer-to-peer systems, and present their advantages and weaknesses with regards to the collaboration issue when employed in mobile environments. The proposed solution is called "MOG-CVE", acronymed from **MO**bile **G**nutella for **CVE**. Current MANET routing is based on IP addresses, in most cases however, peer discovery is based on the collaboration context, presented by Session ID and Zone ID [10]. To solve these problems, MOG-CVE extends MANET routing beyond its standard at the network layer; the application layer must be aware of the physical network in order to minimize the node workload and the network overhead. The system exploits a new multi-tier Gnutella overlay to improve the quality of the resulting network topology. In the resulted overlay, peers position themselves in multiple layers according to their Area of Interest (AOI) [10] via logical links with other peers participating in the same zone ID. By using the users' CVE interests to implement a multi-tier Gnutella network [97], the discovery and search query processes are routed according to the user interests.

With regards to the scalability and adaptability requirements, we re-designed the Gnutella Ultrapeer System (GUS) [38] to shield nodes having limited resources—called leaf nodes—from network traffic and heavy workloads. Large number of unqualified Ultrapeers may deteriorate the system performance. GUS involves a policy to select a subset of Ultrapeer nodes to serve leaf nodes based on the network behaviors and the user interest in the virtual environment. The Gnutella Ultrapeer System's node members must be well-dispersed to offer equivalent accessibility to leaf nodes.

Finally, we identify the constraints that significantly impact the network performance of the MCVE system. We study the interaction between the underlying ad-hoc routing protocol and the system itself. We develop an application using NS2 [80] simulator that takes into consideration node mobility in the network and the avatar mobility in a game scenario. We evaluate the benefits of the enhancements reported for the chosen peer-to-peer protocol to serve MCVE applications. We implement our system under a set of well known existing routing ad-hoc protocols in the literature: reactive (DSR [55] and AODV [50]), proactive (DSDV [59]), hybrid (ZRP [44]) and hierarchical (HSR [54]). The focus of this is not set on comparing a set of ad-hoc routing protocols against each other, but rather determining the MCVE network performance under each routing protocol. As a complementary result, we also compare our proposed protocol with two most well known mobile applications called MPP [40], and EGAN [21].

1.7. Performance Evaluation

Testing our architecture as a whole against existing systems will be very costly, and will require a qualitative endeavor; thus, we conduct a separate comparison for each implemented protocol with other similarly well-known existing protocols; we may stamp a success mark on each protocol if we produce an improved network performance.

We simulated our system using Network Simulator (NS2 v2.33) [80] software as a simulator tool. To evaluate the MOG-CVE level performance, we simulated two other common similar systems MPP [40], and EGAN [21]. The implemented protocols were tested on various network scenarios using different metrics, such as network size, node mobility speed, and pause time. All tests were conducted on Linux machines in PARADISE Laboratory at the University of Ottawa.

1.8. Thesis Structure

The remainder of this thesis is organized as follows. In Chapter 2, we provide a review of the recent literature on networked CVEs. Chapter 3 provides an overall description of the entire system. Chapters 4, 5, and 6 present and analyze solutions for each implemented protocol: Chapter 4 describes related works on peer-to-peer over MANETs, followed by the proposed overlay network formation protocols. Chapter 5 details the Ultrapeer selection concept as a solution for scalability and fault-tolerance problems. Chapter 6 solves the broadcast problem by introducing a novel caching protocol for the MCVE application. Chapter 7 summarizes the simulation results for the implemented protocols, followed by associated analysis and discussions. Finally, Chapter 8 presents our conclusions, and identifies a number of open problems that would worthwhile to invest in the future.

Chapter 2

Background and Related Work

Over the past few years, several CVE systems of wide applicability have been developed; ranging from military training combat (land, earth, sea) [34], [75] to commercial application [32], [99] to multiplayer games [77], [103], and virtual shopping mall [23], etc. Such systems are typically designed either on top of the client/server or peer-to-peer architecture. Most of the current CVE applications utilize a centralized server model. It is well known that the client/server architecture can lead to bottlenecks and single point of failures problems, which may cause longer network delays, and render the system less interactive; peer-to-peer networked CVEs were developed as an alternative solution to the client/server model, to circumvent the mentioned problems. However, peer-to-peer network solutions introduce more complexity to every node of the VE application, and can generate more network traffic since every host may send data to every other host participating in the VE. In this chapter, we study previous works based on peer-to-peer networked CVE, in order to identify their main advantages and drawbacks when deployed in MANET [52].

2.1. Collaborative Virtual Environments (CVE) Systems

A number of standards and prototypes have been proposed in the literature such as [35], [64], [51], and [76] to address the issue of collaborative virtual environment. In such systems, when a user interacts with the VE, an update message is generated. This message should be sent to all other nodes participating in the virtual environment. The manner of sending the update message differs from one application to another, depending on the communication architecture. We classify our studied CVEs applications into two main categories: *peer-to-peer* and *client server* models. CVEs designers can choose one model, or they can 'mix' both models, using peer-to-

peer for some aspects of their design, and client-server for others. Figure 2 depicts the main communication architecture solutions used in the literature to develop CVEs applications.

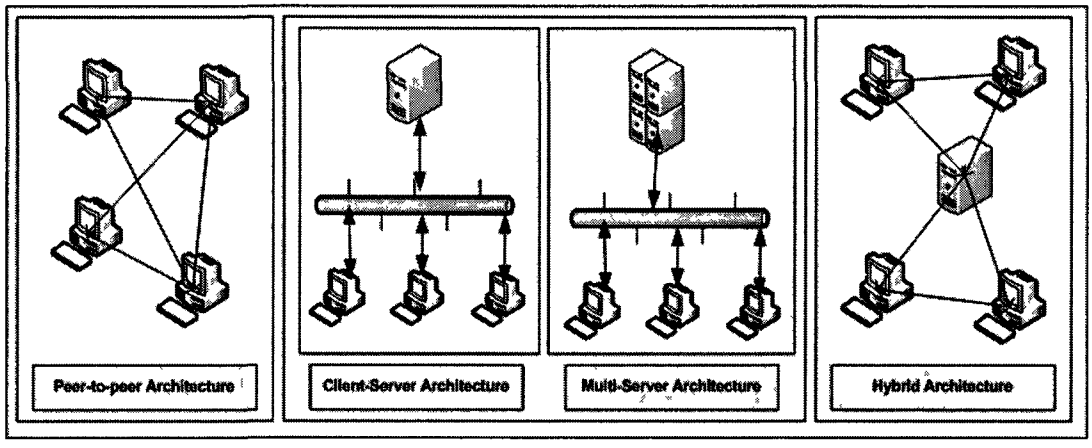

Figure 2: CVE Networked Topology

In the next section, we present a detailed survey of several peer-to-peer CVEs systems that are considered the most prominent in this area. We also examine the client-server architecture in this survey, since it represents the foundation of CVE applications. Figure 3 summarizes and classifies all the studied systems in this thesis.

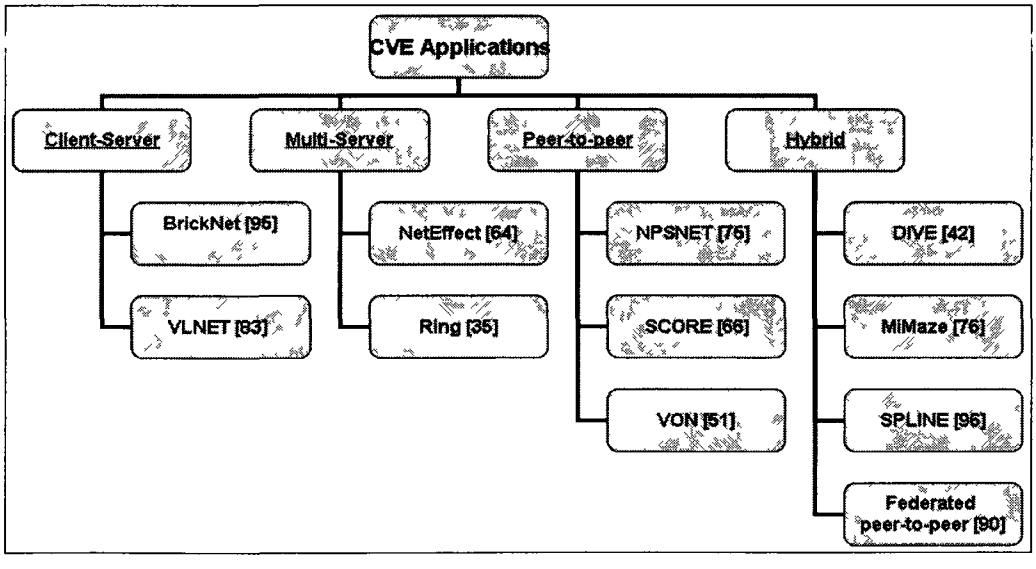

Figure 3: CVEs Classification

Most of the studied CVEs systems presented in Figure 3 were developed either for academic or commercial applications. Since our main focus is on MCVE, we also present the most important works in the literature that deal with peer-to-peer over MANET [52] for collaborative purpose [5], [26], and [48].

2.1.1. Client-Server-based CVE

In this section we present the main CVE systems that fall in the client-server and multi-server categories. Each avatar has to send an update message to the server, no matter how many users are participating in the CVE, and receive an update message from other nodes through the server. The major users of client server model are commercials multiplayer games; SecondLife [95], EverQuest [32], and UltimaOnline [103] are such games worthy of mention. Usually, game designers choose to exhaust less effort on network performance in favor of enhancing graphics representation and sounds. Many other works have been proposed in the CVE literature [83], [64], [35], and [95] for academic and commercial use. BrickeNet [95] provides a filtering technique to minimize the number of messages to be handled by each user. Compared to other existing architectures, BrickeNet introduces an interesting strategy of controlling shared objects. Instead of sharing the virtual environment, each node manages a local copy of the VE, and selects a set of objects to be shared with other users. This strategy offers the user full privacy, but only at the expense of collaboration activities, since users cannot access all the VE resources. VLNET [83] (Virtual Life NETwork) was originally distinguished by its sophisticated 3D graphics and communication protocols. However, the major limitation of the proposed system is that as the numbers of users increase, the server suffers bottlenecks due to the vast volume of messages generated for sending; multi-server architecture has been developed as an alternative to overcome the server bottleneck. Systems adopting such architecture, like NetEffect [64] and Ring [35], devote special attention to minimizing network traffic, particularly traffic that must go through servers. NetEffect [64] was designed to simplify the development of

network-based virtual worlds. A server master guarantees the load balancing among the set of available servers by migrating clients from one overloaded server to another one with less workload. In Ring system [35], authors exploit a Potentially Visible Set (PVS) data structure to handle the scalability requirement. Unlike NetEffect, Ring uses a static virtual environment partition, where each user is represented by an entity rendered on every other computer participating in the network. Based on its current position, the user enjoys the leisure of selecting statistically which server to connect to. The key feature of the Ring system is that the server-based visibility algorithms compute potential visual interactions between users in order to reduce the number of messages required to maintain consistency. However, multi-server architecture suffers from a side effect: the migration between servers may incur a higher waiting time when a user requests a new server; a peer-to-peer architecture has been developed as an alternative solution to the Client-Server; next section describes existing peer-to-peer CVE, and highlights their main drawbacks.

2.1.2. Peer-to-peer Network based CVEs

This section constitutes an essential reference to the existing systems that are closely related to our work. In peer-to-peer systems, each avatar directly communicates its update information to all other participants on the network. In such a model, there is no central server and no single point of failure, since all nodes play client and server roles. A number of well-known applications, namely DIVE [42], MiMaze [76], NPSNET [75], SPLINE [96], SCORE [66], and VON [51] are extensively described; other, less common applications like MASSIVE [4], Federated peer-to-peer [90], SimMud [61], and Zoned federation [53] are briefly reviewed.
Federated peer-to-peer [90] uses hybrid architecture, where nodes are organized into various groups, and managed by a *multicast reflectors* scheme. SimMud [61] proposes a solution based on DHT Pastry [91] structured network to support massive online multiplayer games. The VE is divided into regions, with each region managed by a super node, and playing the root role for the multicast region tree. Any user

action is received by the root and delivered to all multicast members. Network latency is affected, since communication between nodes operates under DHT [25]. To overcome this problem, Zoned Federation [53] model uses DHT only for topology connectivity, while uni-cast communication is used between regular nodes and the region's super-node. MASSIVE [4] (**M**odel, **A**rchitecture, and **S**ystem for **S**patial **I**nteraction in **V**irtual **E**nvironments) proposes a technique to limit the number of connections as the number of users increases, in order to reduce network traffic. Each object is associated with an *Aura* that defines the area in the VE in which the object can publish. Objects can communicate only when their *Auras* overlap. MASSIVE was developed to handle mainly scalability and network heterogeneity.

To the best of our knowledge, the first well-known peer-to-peer CVE prototype is **NPSNET** [75] (**N**aval **P**ostgraduate **S**chool NPS)**.** It was developed in 1990 for large-scale military simulations. The designed prototype follows the Model-View-Controller (MVC) pattern to offer reusability and simplicity. NPSNET uses SIMNET and DIS [75] as networking technique solutions to interoperate with other simulation systems. The virtual environment is divided into a well-defined hexagonal cells structure, with each cell having its own multicast group so as to save network bandwidth. To achieve scalability and reduce network traffic, NPSNET incorporates the dead reckoning [75] algorithm, however, NPSNET has been designed with special focus on military simulation application; and it does not take into consideration the network heterogeneity issue.

DIVE [34] (***D**istributed **I**nteractive **V**irtual **E**nvironment*) was developed at the Swedish Institute of Computer Science. This approach seems to be an improvement of NPSNET system. The virtual environment is divided into flexible sub-regions. DIVE uses RTP (Real-time Transport Protocol) for stream data and SRM (Saleable Reliable Multicast) for non-stream-based data communication. The network traffic is reduced by means of the SRM protocol, since there is no need to send keep-alive message. The SRM protocol is able to detect missing data packets on

reception. We classify DIVE under the hybrid architecture category since there is a central node called *DIVESERVER*, which is needed to supply the initial connection; however, Frécon et al classified it as a peer-to-peer system. This architecture seems more appropriate for scalability requirement because it features a multicast technique, in addition to the virtual environment partition.

SPLINE [96] (***S**calable **P**latform for **L**arge **I**nteractive **N**etworked **E**nvironment),* was developed in order to provide a solution that facilitates interoperability between system parts. SPLINE enables a hybrid topology based on the client-server and multicast approach in order to cope with low bandwidth networks. In the proposed architecture, users do not communicate with each other, instead they only communicate with the world model. In contrast to the regular partitioning scheme used for NPSNET [75], SPLINE [96] divides the virtual world into arbitrary shapes called *LOCALES* where each *LOCALE* has its own multicast. Users are permitted to be "present" in more than one *LOCALE* through the use of *spObserver* objects. *LOCALES* are considered to be the key feature for improving scalability in SPLINE scalability. The major limitation of this approach is that it does not handle persistency; objects exist only as long as the application that owns the object runs.

SCORE [66] is a scalable multicast-based communication protocol developed around 2000. It was designed to support multiple multicast groups and multiple agents. In SCORE, the virtual environment is dynamically divided into several areas, or cells, in order to offer better world state management. Each cell has its own multicast group, and an avatar subscribes to a set of multicast groups, which fall, at least partially, within its Area of Interest (AoI). To control network traffic, SCORE allows for different size cells based on the population of the area.

MiMaze [76] originated from **M**ulticast **I**nternet **Maze**. It is a real-time interactive game built on top of a hybrid architecture, where a central server is used to

get the state of the game when new players join existing sessions, and also to perform session management. The communication system is based on RTP over UPD/IP multicast. A bucket synchronization mechanism is used to guarantee the consistency of the game. MiMaze also employs the dead reckoning [75] protocol to compensate for messages that do not arrive before the global state is calculated. To reduce the network traffic, MiMaze [29] employs only one type of message.

VON [51] (**V**oronoi-based **O**verlay **N**etwork) was developed essentially to address the scalability requirement in CVE applications. VON gives a simple solution based on the geometrical construct Voronoi diagram to solve the neighbor discovery problem. To minimize network traffic, a dynamic adjustment of the AoI is implemented at each node. Compared to other existing scalable architectures, VON is able to constrain the node resource-use.

Continuing advancements in mobile technologies may enable MCVE to become the major applications platform in the research community for the upcoming years. In the next section, we present the most important collaborative mobile applications presented in the literature.

2.1.3. Mobile Peer to-peer based Collaborative Applications

All the above described systems in Section 2.1.2 were designed with wired network and desktops in mind. Some efforts have been recently undertaken to design collaborative applications on a mobile peer-to-peer network basis. At the present time, most research community focus their efforts on mobile file sharing applications as mobile collaborative applications. Authors in [48], [21], and [26] demonstrate how peer-to-peer protocols can be successfully implemented in mobile ad-hoc network in order to enable information sharing. Seneviratne *et al* [48] addresses the problem of mobile file sharing by introducing the mobile agent. The architecture uses mobile agents to participate in Gnutella network on behalf of mobile devices, in order to reduce the volume of communications messages and the power consumption of the

mobile devices. MobiGrid [26] address the problem of information sharing using an interplay scheme between structured peer-to-peer system and MANET. This architecture is unique in that it allows nodes to negotiate the key space in order to build a sophisticated binary search tree. Choi *et al* [21] outline an enhanced Gnutella protocol for ad-hoc networks that addresses the PONG message flow problem. A peer-to-peer metric value is used to select the Ultrapeer node candidate. The proposed system provides better performance than Gnutella in terms of query hits and network overhead. ORION system [100] explores the concept of cross-layer over MANET as a scheme combining the discovery process routing at the network layer and the query process at the application layer. ORION provides unnecessary network overhead when application layer routing is used. MPP [40] (Mobile Peer-to-peer Protocol) is another example of cross layer approach, and is based on DSR [55] ad-hoc routing protocol and an inter-layer communication channel. Each node announces itself to the routing protocol, and a search request is broadcasted to all nodes. Any receiving node will process the search request even when it does not satisfy it. It is clear to see that MPP does not scale when the network size and the network query rates increases. Also, searching and transferring data requires several acknowledge messages between the network and the application layer, which may cause extra energy consumption and increase the arrival time of the corresponding replay message; many other common approaches that deal with mobile file sharing are described in Chapter 3. As some authors in [61], and [63], and [66] state, there is a notable difference between a file-sharing application and a CVE application. In the latter, the content search and the computing are much more complex, as the user interest may vary over the time depending on its avatars' position in the virtual environment. Also the network performance has an important impact on collaborative application compared to file sharing application. Contrary to mobile file sharing research area, mobile collaborative applications have received less attention in the literature. There are only a small number of works [5], [45], and [102] that study the concept of VE in a mobile environment. Bejan *et al* [5] define a mobile platform for online games. The game is managed by a session node; the discovery process relies on a Resolver Service in

order to send queries and process search services. The Resolver Service is an interesting alternative, but introduces high network overhead, since each device must be queried about whether or not it will host the service. PnPAP [45] (Plug and Play Application Platform) is a hybrid system solution through which applications can access different types of networks over various protocols (SIP [89], JXME [57], and Jabber); PnPAP uses a holistic connectivity management layer [46]. PnPAP offers free opportunity and simplicity since it allows users to employ many network technologies simultaneously in the network layer. Alf *et al.* [105] designed a framework to support Computer-Supported Cooperative Work (CSCW) on mobile phones using J2ME and Personnel Area Network (PAN). The developed scheme is not suitable for CVEs because collaboration tasks in CVEs are more complicated than the cooperative work scenarios featured in CSCW. For instance, CVE systems are expected to enable user interaction, and collaboration with the environment and other participants regardless of their proximity. CSCW is still under progress; simulation scenarios are very limited (only three users), which puts the applicability of the protocol into question. A promising social application called MobiTrip [23] was developed to allow many users to express their opinion in the environment. User options are made available when user devices connect on the fly, or when users approach connection hotspots. Bluetooth technology is used in order to form a social space of nearby devices. A typical example of such an application is the shopping mall, where users can express their opinion about new products. iCoulds [102] is a simple peer-to-peer information-sharing system for mobile networks. It was developed mainly for J2ME (Java 2 Micro Edition). The communication layer in iClouds architecture is based on a simple one-hop exchange message. Proximity limitation and *information sprinklers* are the limitations of this project. A pure peer-to-peer (i.e. without super-peers) network over MANET [52] was used by the PROEM [63] project. It employs XML technology for message representation. The main limitation of the PROEM [63] project is that each node is required to implement an XML parser; it also does not address the problem of adapting the virtual overlay network to the physical network. Many other applications include mobile peer-to-

peer, such as PDA access to video stream, remote processing, remote controllers [41], intelligent transportation [3] systems, inter-vehicle communication, and vehicle-to-road communication [19] have been reported in the literature.

2.1.4. Limitation of Existing Systems

While most of the existing proposed systems in the literature are greatly beneficial (and applicable) in the academic and commercial sectors, they falter suffer from being limited to specific tasks, and their architectures are typically tightly coupled to the characteristics of their networks, and any extension or modification to adopt a new class of application is almost impossible. Systems like [53], [29], [96], and [76] are promising platforms, as long as they stay in a wired network. Mobility cannot be added since they are designed with a wire-line network in mind, and require the use of network infrastructure to offer certain network services like multicasting. Mobile applications like [102], [105], [48], and [63] do not match the physical layer to the application layer, which may (and in most cases will) increase the network overhead. Another common limitation that is not quite addressed by many systems is the network heterogeneity. In any group of participants, it is highly possible to have nodes that are more powerful than others, and it will be optimal to have control over the node load in order to assign extra work to nodes with more hardware capacities. Some solutions will exclude some weak nodes from joining the VE, while others see weak nodes must ignore some VE resources that are vital for the system to function. Table 1 summarizes the most well know mobile peer-to-peer approaches.

	MPP [40]	ORION [100]	EGAN [21]	iCould [102]	PROEM [63]	PnPAP [45]
Peer-to-peer overlay	Gnutella Like	Optimized Routing Independent Overlay Network	Gnutella	-	-	-
Routing algorithm	Uses EDRS (extended version of DSR)	AODV	AODV / DSR /DSDV	UDP based ping/pong	The *Proem* Transport Protocol.	SIP/ JXME/Jabber
Scalability	Good/Avreage	Average	Good	Poor	Poor	Good/Average
Network heterogeneity	No	No	Yes	No	No	No
Main Design principal	Cross-layer design, using a vertical interlayer communication protocol.	Implemented at the application layer. Used specially for file sharing applications	Uses a cross layer approach to negotiate the Ultrapeer selection scheme	Based on one hop communication. Uses Personal Area Networks (PAN).	Based on the concept of *peerlets*. Employs XML message representation.	Provides an abstraction layer for networked applications, to hide the complexity of underlying protocols.
Performance evaluation	Comparison with the ORION system	No Comparison with other systems	No Comparison with other systems	No Performance evaluations are discussed	No Performance evaluations are discussed	No Performance evaluations are discussed
Prototype Implementation	No	No	No	Yes	No	No

Table 1: Comparison of Mobile peer-to-peer approaches

Systems implemented in a mobile environment may not scale well to handle many simultaneous clients when interacting with a shared world due to the saturation of network bandwidth in handling broadcast, or when multicasting from clients. Thus, there is a need to have control over the network traffic, and also a need for scalability and robustness. Looking at the collaboration activities used in the existing systems, and the collaboration in the virtual environment, we observe that they have different requirements and purposes. In this thesis, since our first concern is mobile CVE, it is imperative that we follow a different approach. We introduce a cross-layer approach to build a multi-tier networked architecture based on Gnutella protocol with a smart data-caching scheme to increase the performance level of MCVE applications.

Chapter 3

Overview of our Proposed MCVE System

3.1. Introduction

The internet has accommodated many peer-to-peer systems designed to handle large scale CVEs applications in a wired network. The popularity of mobile ad-hoc networks and the advances of technologies for mobile devices have pushed the CVE to yet to be explored areas. Mobile networks lack what wire-line networks enjoy in bandwidth and hardware capabilities (especially memory). The main concerns when designing MCVE applications are: first, efficient network performance, since network traffic is considered to be expensive in mobile networks; and second, acceptable network latency, so that all participating users can share the same view of the virtual environment, and interact in the system by maintaining the feeling of immersion.

The peer-to-peer paradigms exhibit significant particularities, such as the absence of a central node; the fact that every node plays equal role (client / server), the ability to self configure and self heal, which makes them function seamlessly in a heterogeneous manner in wired and wireless networks. Thus, it is natural to develop and deploy CVE applications over MANET using peer-to-peer model. However, as far as authors are concerned, the impact of ad-hoc on the performance of peer-to-peer networked CVE has not been carefully evaluated yet. Moreover, there are several problems when implementing peer-to-peer systems alone, including synchronization among multiple hosts in the network, flooding process, and data persistency. To tackle these problems, we have designed a cross-layer approach to build a dynamic multi-tier overlay network based on Gnutella protocol. Users position themselves in

multiple layers, depending on their state and physical position in the virtual environment. By organizing the network, we are able to reduce network traffic and network latency. An inter-communication procedure is placed between the application layer and network layer in order to increase MANET's routing performance, and to have better control on the routing path quality.

In this chapter, we first present a brief literature review of existing works on peer-to-peer protocols, and subsequently we justify our choice of using the Gnutella network as a peer-to-peer platform in this work. Finally, we describe a design overview of the entire system and its main protocols components.

3.2. Peer-to-Peer Network Background

Peer-to-peer has become a popular paradigm for accessing and searching information on the internet; many peer-to-peer network protocols are comfortably settled in the literature. According to the network topology and query routing, we classify the peer-to-peer networks into two main categories [16], *Structured* and *Unstructured.* We tend to focus on unstructured networks, since our work exhibits an approach that falls in that category.

3.2.1. Structured Networks

In a structured network, the lookup service is based on a distributed hash-table (DHT) [25]. New nodes are required to changes the DHT infrastructure, to which the routing information needs to adapted. A number of approaches, namely CAN [88], Pastry [91] and Tapestry [110], have been successfully implemented to offer better performance. The most common problem of the structured network is that it does not address the area of interest [51]. Moreover, when DHT is deployed in a CVE application, a node may be rapidly overloaded and as a result, may cause a bottleneck, especially in highly dynamic CVEs, because roles are assigned to nodes

based on a hashing key function, instead of the device hardware and network capability. Another problem with DHTs is that they cause higher latencies than the underlying network path [10].

The major application area of such approaches is still a fixed network. Unfortunately, these schemes may be vulnerable to network mobility because they do not provide any methodologies to mapping the virtual network to the physical network. Also, in DHT, the hash function decides which node hosts the index; therefore, there is no control over the number of peers holding DHTs. In mobile environments where the node's mobility increases or network density decreases, DHT suffers more overhead in the maintenance process; an unstructured network may therefore perform better.

3.2.2. Unstructured Networks

Unlike structured networks, the topology in unstructured networks is created arbitrarily. According to [45], the unstructured peer-to-peer model comprises three generations. The first (Centralized) is a mixture of client-server and peer-to-peer networks; the second (Decentralized), is a pure peer-to-peer network where nodes can play both roles (*client or/and server*); and the third, a hybrid approach of the first and second generations. Table 2 provides an overview classification of the studied unstructured peer-to-peer network architectures. Many of them are widely used in this area, namely Napster [79], JXTA [5], and Gnutella [38]. Others are less common like Kazaa [58], Freenet [33] and Morpheus [78].

	Centralized	Decentralized	Hybrid
Kazaa			✔
JXTA			✔
Napster	✔		
Gnutella		✔	
Morpheus			✔

Table 2: Unstructured Peer-to-peer System Classification

In centralized architecture, a server is dedicated to maintain a peer's index and resources. The server processes all resource queries, and returns a matching list to the requesting node. Clearly, this architecture could not be used within an ad-hoc environment because of the single point of failure and the mobility issue of nodes. Contrary to this, no dedicated servers are required in a decentralized approach. A broadcast transmission is required for both, peer discovery and query requests. The broadcast is controlled by TTL (Time-To-Live) field so that the broadcast can specify the number of hops. Gnutella [56] is taken as leader of decentralized approaches. However, the newly-released Gnutella protocol V0.6 [38] is extended to cover the hybrid architecture. JXTA [57] is a well-known example of a hybrid approach where only selected nodes, called super-peers, are used for resource discovery and query process. Following, we will discuss the main common unstructured peer-to-peer networks.

Napster [79]: Napster was the first widely-applied peer-to-peer file sharing application. It was based on a centralized architecture where a central server maintains the database of song files. Unfortunately, each user had to direct their search query to the central server to identify the location of the requested file. Each participating node had to provide its information to the central server in order to be available for other users. When a peer wished to download a specific file, it requested the server. Upon receiving a positive replay, the peer established a direct connection with its counterpart in the transfer process (the peer holding the file), then downloaded it from there. This architecture obviously suffered from a single point of failure and higher network latency, which may hamper the overall performance of the system.

Gnutella [38]**:** Gnutella is a peer-to-peer file-sharing protocol provided by the Nullsoft 2000 Company. Initially, it was developed with a desktop environment in mind. In 2004, a new version Gnutella V0.6 [38] was released; it moved Gnutella towards the third generation where peers are classified as leaves and Ultrapeers.

Leaves are weak nodes with hardware limitations (storage, computing performance, and network bandwidth), while Ultrapeers are powerful nodes. Gnutella peers discover other peers in the network using PING and PONG messages. The peer forwards PING messages to its neighbors' nodes; neighbors forward the message to their neighbors; and one of the latter nodes is expected to send a PONG message. This recursive broadcast will continue until the Time-to-live (TTL) field of the PING message reaches zero, or a node replies with a PONG message. Message response will follow the reverse path of the message request. In order to download a file, the peer establishes a direct connection with the node holding the requested file. The actual file transfer of resources is accomplished outside of the Gnutella protocol using HTTP.

FastTrack [98] is hybrid architecture based on Napster and Gnutella protocol. Any peer can update itself to become an Ultrapeer. The system offers the possibility to control the node degree and network bandwidth. The concept of Ultrapeer is still a work in progress since the change of node is performed manually. Unlike Napster, once a user logs onto the network, he/she is directed to a Super-Node by the FastTrack servers. This Super-Node contains a list of user files that can be searched and downloaded. The Super-Nodes form a cluster of several machines that intelligently handle file search requests and route them.

JXTA [57] is perhaps the closest work to the Gnutella network. It was developed by Sun Microsystems Inc. JXTA's was developed with the goal of addressing the shortcomings of current peer-to-peer topology. Messages sent to peers were deployed into XML format in order to establish a virtual network. "Peer group" is a coined concept upon which all protocol functionalities are based. Like Gnutella, the JTXA network consists of *edge-peers* and *super-peers* (rendezvous and relay peers). When a peer joins the JXTA network, it finds the nearest rendezvous peer; the edge peer then uses the rendezvous peer as a gateway to the JXTA network. The rendezvous peer uses DHT for optimizing the service discovery process. Any edge

peer sends a query to its rendezvous peer. The rendezvous node replies to the query if it is able to supply the requested resource; otherwise, it forwards the query to another rendezvous peer.

Mobile peer-to-peer system architectures have changed over time due to advances in technology, and the exploitation of new possibilities. Traditional overlay developed for wired networks cannot be a feasible solution for mobile applications due to message flooding, network heterogeneity, and wireless link reliability problems. As a result, the JXTA community developed a light version in order to support the mobility requirement. Two different platforms were designed: JXME Proxied and JXME Proxyless . Proxyless is still under construction, and thus, a number of design and implementation issues need revision. The JXME Proxied platform is based on a hybrid peer-to-peer mode, it relies on central entities in order to play a proxy role between nodes belonging to the same JXME virtual network, and all mobile devices must be in proximity to the server (JXTA relay) so that they can communicate through this server. This approach, however, considers only cellular network and does not aim to match the virtual network to its physical counterpart. Another problem with JXME is interoperability; JXME is made exclusively for J2ME (Java 2 Micro Edition). Current Gnutella [38] architecture is mainly designed to work in fixed and wired infrastructure. Gregori *et al* [24] present an interesting result: a straightforward implementation of Gnutella protocol in the ad-hoc network is not satisfactory. Many efforts were put forward to investing Thus, the system would either have to specify a minimum level of performance for all participating nodes, or control the amount of data transferred so that weak nodes do not affect the overall system. In [21], and [24] authors outline a cross layer technique to enhance the Gnutella network in mobile environment. Anna *et al.* [46] create a combined approach between the Gnutella network and Bluetooth technology to implement the application-sharing information. Bluetooth devices use SDP protocol (Service Discovery Protocol) as opposed to the PING and PONG scheme in order to advertise their services and discover other peers. This approach creates direct communication between users in close proximity. The

major limitations of using Bluetooth lie in the fact that service discovery produces high network overhead, in addition to constraints of proximity; communication between neighbor peers might be missed.

All in all, the approaches mentioned above do not address the question of routing and do not provide any mechanism to adapt virtual topology to the physical realm, which will probably cause unnecessary network overhead. Since a node must process all received queries, it may easily become overloaded; our developed system avoids this shortcoming by employing a cross-layer approach-based Gnutella with a Session Context Awareness in the network layer. To the best of our knowledge, the interaction between the application layer and the underlying ad-hoc routing protocol has never been clearly studied in the literatures. Barbosa *et al* [2] describe an interesting evaluation of ad-hoc routing protocols under Gnutella peer-to-peer network. Their simulation concludes that none of the ad-hoc routing protocols performed well under all circumstances in Gnutella network.

3.3. Why Gnutella for MCVE Applications

To re-cap, previous existing applications seem to be often used for applications in order to share, retrieve, and access information, and are typically designed for static content (MP3, MPEG). According to [93], and due to node mobility, the structured peer-to-peer network is not expected to perform well in mobile peer-to-peer networks. A node does not possess the ability to decide its appropriate neighbors, and hide the complexity of the system. Data that can satisfy the query (in some manner) should be accessible by the requestor node. Presently, it seems that most systems have been initially designed with a specific network technology for a particular application, and as such, any employment of such systems in different context appears to be unfeasible. MCVE applications faced the fact that not only do they have to locate the information using an efficient search query procedure, but they

also need to route the update messages with an acceptable network delay, and thus preserve the immersion feeling in the virtual environment.

The question remains on whether to reuse an existing protocol in order to achieve the previously described objectives, or to simply develop a new peer-to-peer protocol suitable for MCVEs; to minimize the effort and time, and to reap the advantages of former developments, our designed system will reuse the existing Gnutella network. The advantages of using an unstructured decentralized system in MCVE are: the ability to operate on a heterogeneous network, enable node capability to self-organize, and offer seamless adaptability to the mobility requirement. We employ the latest version of Gnutella V0.6 [38] in order to take advantage of node classification, and overcome the network homogeneity and scalability problems.

Gnutella was chosen for this study based on a number of considerations. First, the simplicity of its communication model provides sufficient freedom to the developer to modify any components. Second, the Gnutella protocol possess various advantages, such as robustness, control over node selection probability, scalability, and the ability to cooperate and share resources with other protocols; all of which satisfy MCVE application requirements. Third, Gnutella provides the means to enable content based routing. Forth, it is an open source [56] and widely used system [111] so that anyone can create a Gnutella client. This guarantees a variety of methodologies and approaches. Lastly, Gnutella is well-regarded for being able to operate in a very dynamic network topology.

Unfortunately (as described in [24]), the current version of Gnutella has certain limitations when deployed in mobile environment. The fact that Gnutella was initially designed with a desktop and wired network environment in mind means that a number of issues, including network performance, selection independence and fault-tolerance mechanism, become un-wanted guests in the mobile environment. For that,

we could not use a straightforward implementation of the Gnutella client to handle mobile CVE applications, as shown in the simulation results in Chapter 7.

3.4. Initial Considerations and Terminologies

Similarities between peer-to-peer and MANET [52] encourage the development of an efficient MCVE application. However, both networks lack on-managing and overlay formation, since in both architectures, the network is established as soon as the node opts to interact with another participant. The decision to connect to the network is taken randomly, so inefficient connections may be created between nodes in the environment, causing poor performance. In this thesis, we have identified sufficient commodities to implement MCVE applications using the Gnutella network, where users can collaborate in a fully ad-hoc fashion with anyone and from anywhere. In a Virtual Environment (VE), each user is represented by a graphical embodiment called an avatar [85]. Through the avatar concept, users are able to progress in the VE and see other users' actions. Our entire virtual world is divided into multiple hexagonal zones in order to keep the amount of data that the client handles small enough to fit into the memory of the mobile device. Activities in zones are grouped into session. A session refers to a group of users interested in the same VE mission. A zone owner node is dynamically assigned to act as an authoritative server for each zone. Figure 4 shows an example of a virtual environment. Zones are presented by a regular hexagon with length side equal to t, the maximal diameter $2*t$ and minimal diameter $t\sqrt{3}$. Using an Area of Interest (AoI) diameter equal to $t\sqrt{3}$, we allow the user to subscribe to four zones at most. The virtual environment flat space is divided into ten zones with t equal to 100 meters, as described in Figure 4.

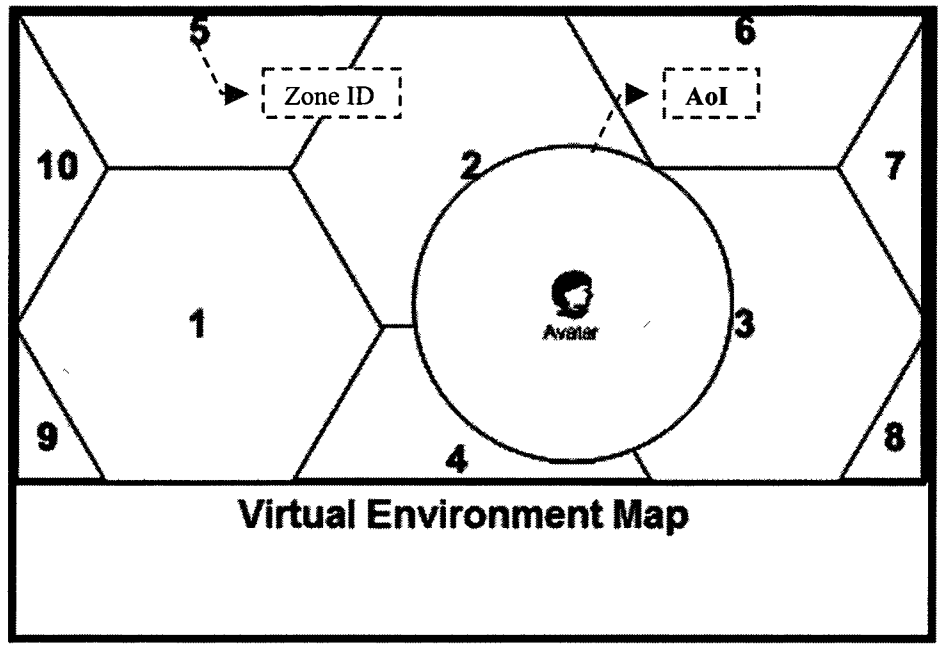

Figure 4: Virtual Environment Zoning

In a CVE, it is likely that there will be many sessions; therefore there is a need for a session manager mechanism to control session access and guarantee the existence of the VE, even when many users leave the session. The system initially assigns a unique ID to each session and zone; when a session becomes active (there is at least one user participating in the session), a session manager node is immediately assigned to the session. When a user joins an inactive session, the node changes its role and becomes a session manager by default; the session has now been activated. Unlike other existing systems [42], [75], and [96], only active zones and sessions are controlled by the node manager in order to minimize the messages traffic. By using the dynamic activation, the multi-tier overlay formation process causes the number of layers in the network to be reduced, which may have a direct impact on the system performance since users are prevented from transmitting unnecessary messages to other layers. The zone owner node is responsible for coordinating the zone state; again, the concepts of 'inactive' and 'active' are applicable to the zone; a peer may be the sole owner of the zones in which it is publishing / subscribing. Nodes may have multiple roles at the same time. We consider the amount of time spent in the session

and node hardware capabilities as important factors in order to select zone and/or session manager.

In this design, node classification is the key feature to handle the scalability issue. We employ a hierarchical node classification defined as follows: session manager, zone owner, and zone member. A zone member can make a request to the zone owner to modify a shared object. The zone owner should block access to the requested object until the player releases the access. During the play, each user often changes his/her physical position in the VE. We referenced this user position using a novel concept called CVE-profile, which is defined as a triple <User ID, Session ID, Zones IDs>. User ID is a unique ID assigned by the system to the node when it first joins, the Session ID is used to identify the session in which the user is currently participating, Zone IDs represent the set of publishing/subscribing zones of users, based on their respective AoI radius. The CVE-profile helps a node to locate other neighboring nodes in the network; we later demonstrate the benefit of this concept and how it affects the performance of the system.

Before describing the system overview, we summarize some terminologies and assumptions that will be used in this work:

- *Zone*: a subdivision of the virtual world.
- *Session*: an abstract concept used to enfold user activities with the same interest.
- *Avatars*: an object that represents a user in the virtual environment.
- *Session manager node*: a node that acts as an authoritative server for a session.
- *Zone owner node:* responsible for coordinating the zone state and users' member activities in the zone.
- *Zone member:* a node that participates in a zone.
- *CVE-Profile*: used to reference an avatar in the virtual environment.

- *Area of Interest*: denoted as AoI, and represents the avatar's radius where it is able to see and interact with other users and objects.

The implementation of the system requires that each user knows the zone map representation of the VE in order to ensure proper functioning. The system architecture description of our system is presented in the next section.

3.5. System Architecture

In order to achieve our objectives (as specified in Thesis Contributions), we developed a MOG-CVE protocol to form a mobile multi-tier overlay network. MOG-CVE is a complete cross-layer system that allows different mobile nodes to collaborate and share data in order to satisfy the MCVE needs. The key feature is to extend the MANET routing beyond its strands to handle a collaboration routing context. A new session and path control component is added in the network layer in order to allow participating users to discover and communicate with other nodes that offer the same CVE interest described in the CVE-profile, rather than finding other peers using IP address. To improve system performance, and to overcome the unpredictability problem in mobile environment, we integrate a Gnutella Ultrapeer System, GUS, to select a set of Ultrapeer. Selected Ultrapeers will play critical role in the VE like session manager, zone manager etc. Also, a novel caching mechanism is introduced, so as to improve the performance of the data retrieval mechanism, and at the same time, alleviate network overhead. In order to achieve our aforementioned objectives, we redesign the network stack architecture used by our proposed MCVE system as shown in Figure 5.

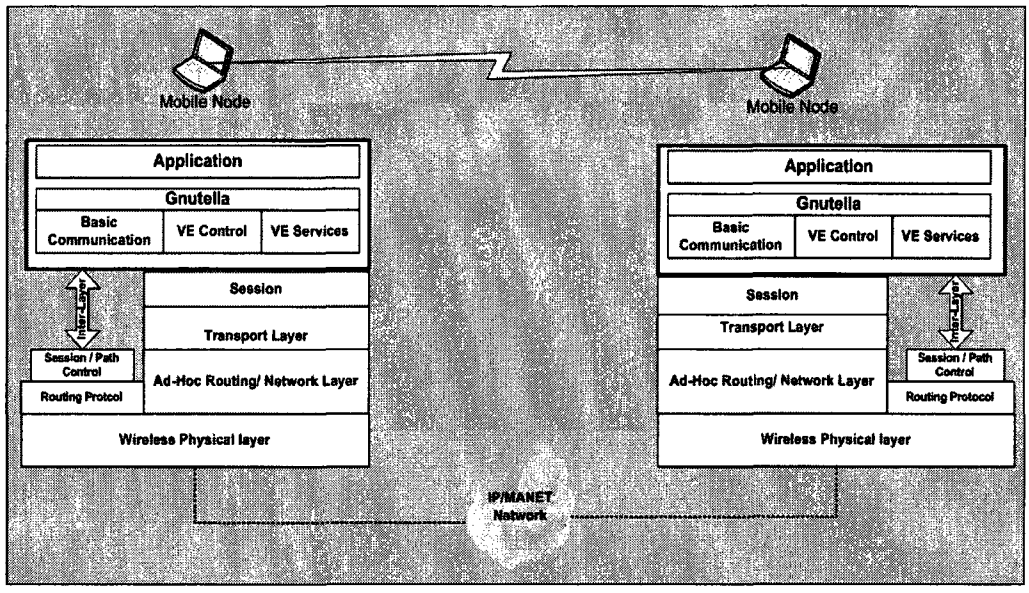

Figure 5: The Communication Protocol Stack

In Figure 5, we focus only on the main layers; namely, the application peer-to-peer Gnutella, network, transport, and wireless physical layer. We apply 'zoom into' the application Gnutella layer in order to explore the implemented protocols in this layer. The message class in the ordinary Gnutella protocol is extended to handle mobility, network management, and collaboration issues. Many modules are incorporated at the application layer:

- The peer-to-peer basic communication module: represents the basic message of Gnutella PING, PONG, QUERY, QUERY Hit, GET, and PUSH.
- The Peer-to-peer VE control module: responsible for controlling the 'join and leave' traffic, for both the session and zone. Also this module also handles messages related to the caching protocol.
- The peer-to-peer VE service module: responsible for maintaining the state of the virtual environment. The update messages related to user actions are also implemented in this module. This module should be later extended to be able to deal with consistency problem by implementing a dead reckoning technique.
- Session and Path Control component: implemented in the network layer to handle the communication session routing in MANET. Network layer should

be conscious about the CVE-profile of its user. The application layer must be aware of the quality of the routing path in terms of number of hops for the requesting node. Algorithm 1 shown below outlines the pseudo code implemented in the session/path control component.

Definitions:

sAdd: represent the source address of the requesting node
session_ID: represent the session ID of the requesting node
hop_Count: represent the number of hop in the routing path
msg_Data: contain the message description
mySession_ID: represent the session ID of current node
DriveToApplicationLayer: procedure used to forward the received message to the upward level
BroadcastMSG: procedure used to broadcast the received message

.....

Algorithm:

```
RegisterMySessionID(mySessionID) // Register the session ID in the network layer

SendRegisterConf() //Send a register confirmation from the network layer to the application
layer


For each RecvMSG(sAdd, session_ID, hop_Count, msg_Data)
    If (mySessionID == session_ID)

        DriveMsgToApplicationLayer(sAdd, session_ID, hop_Count, msg_Data)

    Else

        BroadcastMSG(sAdd, session_ID, hop_Count+1, msg_Data)

    Endif

Endfor
```

Algorithm 1: Pseudo-code for Extended MANET Routing

During the performance evaluation process, we evaluate the impact of the routing on our MCVE system. We consider five well known ad-hoc routing protocols; namely AODV [50], DSDV [11], ZRP [44], HSR [54], and DSR [55] in the network layer. To assess the impact of routing protocols on MOG-CVE, it is important to select network overhead and network latency as quantitative measures, since, they are significant for

MCVE performance. In Chapter 7, we provide a comparative study of MANET routing protocols under MOG-CVE in order to decide about the appropriate ad-hoc routing protocol for our implemented MCVE system.

3.6. The System's Protocols

The key aspect of any MCVE system centers on how the update messages are delivered, and how the network topology is created and maintained. We identify protocols that significantly impact the MCVE system performance. The following subsections present the network topology used and the three main protocols that provide !"#$%&'()$*+,)!(-,&.(!/$-*$-+0$1 234$/'@#5

3.6.1. Network Topology

With regards to flooding and scalability problems, the situation has improved with the new version of Gnutella V0.6 [38]. Gnutella V0.6 was designed based on a hybrid architecture that uses both peer-to-peer and client-server strategies. It introduces the concepts of "Ultrapeer", also called "Super-node", and leaf nodes. Ultrapeer is a node with more resources, while leaf nodes represent weaker entities with limited resources. Our proposed architecture is based on a multi-tier overlay network topology, in which nodes position themselves in layers depending on their CVE-profile and their role. Each layer acts as a cluster network that enfolds all users in the same zone. The cluster head represents the zone owner node. Nodes communicate with other nodes in the same layer via logical links. Communication between layers is facilitated by nodes participating in multiple layers. To guarantee connectivity between multiple layers, we create a session manager and zone owner layer to enclose all nodes with session manager and zone owner roles. Each zone is associated with a list of zone members that contain all zones' user IDs, and is maintained by a zone owner node. This list resembles to the multicast group concept; in Figure 6, we show the network topology architecture.

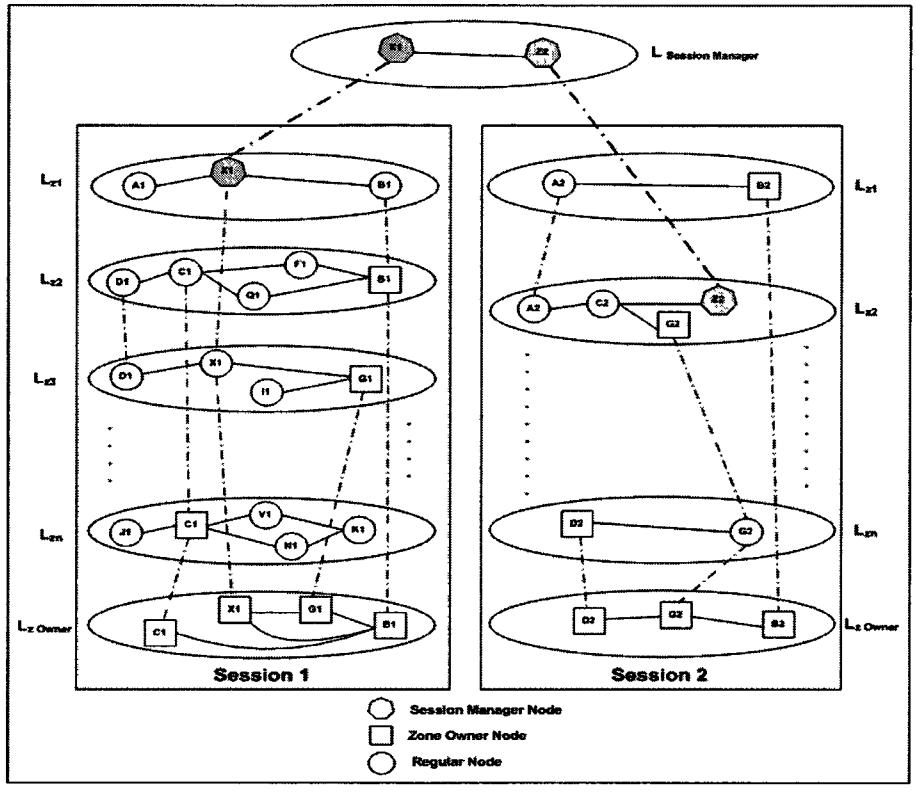

Figure 6: Network Topology of the MCVE System.

To fulfill the communication's purpose, each node will maintain data structure depending on its role. A zone list, member list, and candidate list are maintained in each session manager. The zone list is used to keep information about all zones in the session. The candidate and member lists are stored in each zone owner node, while a list of subscribing/publishing zones is stored in all nodes. In the candidate list, the top element represents the current zone owner ID, and the remaining elements identify nodes that are candidates for a zone owner when the current one leaves the network. The member list identifies nodes that are currently participating in the zone. When a node plays a session role, it maintains a list of all zones and all users in the session. All entries in the aforementioned lists are stored with a time stamp to specify the moment at which each client has subscribed in zones and sessions.

The complex, dynamic world makes it impossible for mobile environments to maintain all world states for CVE applications that run continuously; a main server may be used to tackle this issue. Initially, when there are no players in the VE, the main server maintains the current state of the virtual environment with an original 3D geometric representation of the world. The ever presence of an original VE copy is very useful when a node requests to create a new session. After a first node joins the world, the main server role is shifted to this node and the main server retires. As the number of users' increases and sessions become active, the network will follow the peer-to-peer fashion. The session manager node releases certain pieces of the whole environment to the newly-joined node. In this work, we do not consider the main server entity. We assume that a mobile session manager node that holds all the VE data is already present in the network. This assumption will put a correlation between the system life and the presence of nodes. The system disappears when the last node leaves the world.

In any MCVE application, we believe that there is a need to control the access to the world. Access control in a peer-to-peer network model is a challenging task because of it inherent distributed nature. Each session manager is used as a server in order to store data related to the node registration issue. Nodes that intend to join the world must be registered; in other words, nodes should have a user ID and password. The access control and registration processes are out of the scope of this thesis. Clearly in our design, the session manager represents a single point of failure; therefore it is important to develop a fault tolerance mechanism. One of the main tasks of the designed replication protocol is to maintain a session manager that is failure-less. In addition, our system uses a log file associated with each participating user in the virtual environment to record the last critical actions. During intermitting connections, the log file resumes actions that have not yet been executed or rendered. For instance, a resume download action should take place whenever the connection is disrupted. The log file is executed immediately when a node rejoins the session to prevent previous update actions from starting again from their initial state.

In the next upcoming sub-sections, we provide an overall description of all the developed protocols in this thesis. Note that each protocol will be presented in more details in a separate Chapter.

3.6.2. The Mobile Overlay Network Formation Protocol

A brief description of the MOG-CVE protocol is proposed in this section; the detailed implementation is discussed in Chapter 4. The architecture initially creates a logical network [13] based on Gnutella network in which nodes are dynamically adjusted according to their physical avatars' physical position in the virtual environment, referenced by the CVE-profile. The overlay is created by nodes forming links with others peers sharing common zones in the same session. It is important to keep in mind that the logical connection is knowledge of the address of the reachable node. MOG-CVE is based on a sender-driven CVE-profile mechanism where a user sends a discovery message containing its CVE-profile to discover neighbor nodes in the network. To increase the robustness of the resulted overlay network, the application layer should be aware of the network path quality of neighboring nodes. Also, the application layer should register its session ID in the network layer. By doing this, we overcome the overlay's inefficient connectivity between nodes, and lessen the node workload (by inefficient connection, we mean control over the number of hops, VE interest similarity, and control over node degree logical connections). Information carried over a PING/PONG message cannot act as a gauge to measure the performance of the responding node. Thus, we introduce a new measurement value called the utility value, which is defined as a combination of routing and peer-to-peer metric to differentiate node performance in the system. To improve our system's interest and to reduce user stress during a play session, we provide a simple scheme to handle the replication and the data persistency in MCVE applications. Two kinds of sessions and zones (active / inactive) are supported by this scheme. When at least one user is participating in the session, the session is active; no participating users signify an inactive sessions. The virtual environment must

continue functioning in a proper way, even when users suddenly leave the world. Until now, only few studies have considered both persistency and data replication in mobile environments. It is essential to handle a persistent world since a player may spend a lot of time and effort to interact with the virtual world. In ad-hoc environments, mobile devices are subject to disconnections much more frequently than in wired environments. This feature makes it impossible to ensure persistency and data availability between sessions when all users leave the VE and sessions become inactive. The system is built in a manner that makes failure unlikely, nodes holding critical data are required to select a replica node and play as back-up. The replica protocol investigates how and when nodes should select a replica. For this purpose, we developed a replica selection policy in order to negotiate the quality and the ability of the replica node. On the other hand, due to the limitations of mobile devices, we particularly concentrate on the data analysis of the virtual environment in order to determine which data should be maintained for persistency issues. Processing and storing large amounts of data may result in poor performance due to the resource constraints of mobile devices. Thus, providing a VE data classification system to identify the data that needs to be saved from one session to another will have an impact on the performance of the overall system. We believe that only negligible data needs to be persistent in such a way to be always online and available to other players. Virtual environment Data classification symbolizes the novelty of our persistency protocol compared to other existing protocols [42], and [75].

3.6.3. The Gnutella Ultrapeer System -GUS-

We make use of node classification provided in Gnutella V0.6 [38] in order to build our Gnutella Ultrapeer system (GUS) to address the potential problem of super-node selection and network homogeneity. Current GUS [38] is a promising model as long as it stays in wired network. In this section, we provide a brief description of the protocol, the implementation details are given in Chapter 5. GUS uses an adaptive selection advisor algorithm that enables a leaf node to join the most

qualified Ultrapeer by considering user interest similarity and a combination of network characteristics and user behaviors in the virtual environment. Any GUS members should meet the Ultrapeer criteria, namely stronger computing power, higher storage capacity, and higher access link bandwidth. Ultrapeer criteria can be adjusted dynamically in response to the VE's characteristics. Node roles such as session manager, zone owner, replica, etc... require increased resources in order to store and maintain all the required data (i.e. a list of all zones, users in the session, offline users, etc...). It is important to have these nodes as GUS members to ensure better performance. Nodes with Ultrapeer criteria must donate a part of their bandwidth, storage capacity, and processing power in order to fulfill the role's activities in addition to their regular playing role. Gnutella Ultrapeer System must guarantee at least one super-node connection for each leaf node under all circumstances. Due to node mobility, the system should have the ability to re-configure the network topology with a minimal amount of communication control messages.

3.6.4. The Caching Protocol

In a Gnutella network, nodes can join or leave the network randomly. Flooding mechanisms entail large amount of network overhead and hence necessitates increased computational power. To cope up with such limitations, we implement a novel caching mechanism. Based on Gnutella traffic analysis [111], we identified the necessity of a caching scheme, since more than 55% of Gnutella traffic is caused by PING/PONG messages. The classical pong caching mechanism [67] retrieves the IP address of the sending node by decomposing the Global Unique ID (GUID) identifier and saving it in the local cache memory; thus, we are able to reduce the overhead traffic. In this section, we provide a brief description of the protocol; however, the implementation details are given in Chapter 6.

We discuss the feasibility of designing a caching protocol as a Gnutella optimization technique to reduce the amount of network traffic and the time response query. In this work, we provide a common caching mechanism that could be implemented in any mobile Gnutella client to support MCVE applications. The designed protocol can be easily extended to be used in any other mobile applications, such as file sharing. The novelty in this protocol is that nodes maintain a separate data structure for each logical connection in order to hold the caching entries. Unlike existing schemes, our protocol is designed so that the global cached data structure is maintained separately for each logical connection; this ensures balanced data distribution among all connections and provides the user with a 'spread view' of the network topology. Our developed protocol runs successfully in a mobile environment and is able to achieve improved query response time by selecting the least used node that owns the resource.

Chapter 4

The Mobile CVE Overlay Network Protocol

The proposed protocol, which we refer to MOG-CVE (**MO**bile **G**nutella for **CVE**), uses a smart overlay formation method that dynamically adjusts to accommodate the user's CVE-profile. The developed protocol attempts to offer certain criteria including simplicity of operation, support for node heterogeneity, scalability, and the ability to handle the CVE network requirements. In Mobile Collaborative Virtual Environment (MCVE) applications, communication model and the data download process are usually based on the user position in the virtual environment. To start, the download process requires the intervention of the node to initiate the process, and then waits for it to finish. Due to mobility, the network path quality is never the same during a transfer. Therefore, it is important for such an application to build an efficient overlay that uses a network link selection scheme. This scheme must determine which neighbor node and which network path the node should use during a data transfer.

The main problem in the existing approaches [46], [60], and [63] is that because of the node mobility, the network path will not adapt the underlying physical network over time. Also, due to the fact that users can move forward in the VE, it is always the case that the list of logical neighbors gained by the discovery process should be updated to reflect the changes in the CVE-profile. To achieve better performance, our protocol differs from other protocols mainly in its overlay architecture and its communication model. The network link selection scheme allows the application layer and the network layer to share information about the network path and the collaboration issues in order to overcome the frequent topology changes, and avoid unnecessary intermediate nodes participating in the flooding process. Nodes monitor their logical quality of connections based on their VE interest and routing metrics. As

a result, the overlay becomes healthier since weak connections (characterized by the number of hops and node connectivity degree capacity) are dropped if better ones are available. In standard MANET routing, messages are usually processed at the application layer. Once a message is received at the network layer; it is automatically forwarded to the application layer in order to be answered. The network layer does not have any ability to determine query matching; such a technique may aggravate the workload node and the response query time. Thus, we have extended the functionality of the MANET routing to handle the session ID concept, so that the network layer can handle a partial query matching, and as a result, the application layer will only process queries having similar session IDs to the requesting node. Because of the resemblance between the proactive routing behavior and the probe messages used to maintain a logical connection in the overlay, we have decided to use DSDV [86] as an ad-hoc routing protocol for probe messages. The neighboring resource table for each node remains small since there is no need to store information pertaining to nodes belonging to other sessions.

4.1. Detailed Description of the MOG-CVE Protocol

The MOG-CVE protocol [11], [13] consists of two phases. During the first phase, we identify the scheme for joining and leaving the world. Any new node should request an initial connection to an existing virtual environment member and join the session. The contacted node exchanges control data related to a selected session ID and the initial default set of zone IDs needed by the user avatars. At the end of this phase, the user will have a full CVE-profile. In the second phase, the user is ready to interact with the virtual environment. Now, most of the message traffic is based on session communication group. Users should receive messages that are only part of their interest, as described in the CVE-profile. A multi-tier Gnutella network overlay formation mechanism is proposed to support the interaction in MCVE applications. Network bandwidth preservation and reduced computing power are expected because discover and search messages are routed through specific layers according to the

query context. To increase the user interest in the MCVE applications, an implementation detail about the replication scheme will also be discussed in order to select a replica node for users playing critical roles, such as session manager, zone owner etc... in the virtual environment. Before describing the proposed protocols, we assume the following:

- Each user has a unique ID.
- Avatars are assigned to default positions in the virtual environment, which is defined by x, y, and z (according to world system coordination).
- Every active session should have a session manager node.
- Each session manager handles a current copy of the world state with a list of registered users in the session and an original VE copy.
- Similar to an active session, every active zone should have a zone owner node.
- Every node participating in the world must maintain current information about its session manager and zone owner nodes.
- After joining the world, a node is considered to be trusted and can process any request related to the VE.
- Each user must maintain the virtual environment zones map.

4.1.1. Phase 1: Initial Communication Protocol to Access the Virtual Environment

Any new node that wants to join the world needs to know at least one existing virtual environment member in order to establish the initial connection; in this phase, the protocol displays two levels of communication: the *Session Level* and *Zone Level*. The *Session Level* describes an overview of how the new user joins the world and defines how the interaction between the user node and the session manager node is established. Once a user sets its CVE-profile, the *Zone Level* illustrates the 'zones-join actions' of the set of default zones returned in the CVE-profile during the session level.

4.1.1.1. Session Level

This level represents the initial point to access the virtual environment. Any new node, denoted *Kn*, that wishes to join the world needs to know at least one already existing virtual environment member. The use of a rendezvous node at a well-known location is not practical in the mobile environment. New joining nodes can place undue load on the rendezvous node [104]. An approach to overcome this problem is to use a limited broadcast scheme. A welcome request message *"welcome-request"* that carries a default Time-To-Leave (TTL) is sent, so that *Kn* can discover other nodes participating in the virtual environment (world members). Broadcasts are scheduled to occur periodically with an incremented TTL until responses are received from world members. Between each trial, the node waits for a time interval in order to avoid traffic overload in the network. The broadcast mechanism *"welcome-request"* message works as follows:

- *No "welcome-response" message is received.* This may occur for two reasons: (i) there are no nodes that can be reached by a specified TTL. In this case, a new *"welcome-request"* with an extended TTL must be resent; (ii) the virtual environment is inactive when no users are participating.
- *A "welcome-response" message is received from a VE member node denoted Mn.* In this case the node *Kn* establishes an initial connection with *Mn* using a similar approach to the Gnutella handshaking process. The responding node *Mn* can act as a proxy authority for the *Kn* node in order to join the world.

Upon receiving a *"welcome-response"* message from a world member node *Mn*, the node *Kn* can send to *Mn* either a resume play message to recover its last visited world state if possible, or a regular play message, if *Kn* wants to reset its last state and start a new session. The *Mn* node should forward the play message received from *Kn* to its session manager node. The contacted session manager communicates with other neighbouring session managers and nodes informing them about the changes. When *Kn* requests a resume play message, a resume confirmation play message is sent to it.

This message will also include the last visited world state and user data. On the other hand, when *Kn* requests regular play, a regular confirmation message that contains a list of available active sessions and their usability ratio will be sent. A list of available avatars, with their default positions, is also provided to the new node *Kn*. Initially, each avatar has a set of default zone IDs depending on its physical position in the virtual environment and its Area of Interest (AoI) radius. The user can then decide to join any session based on certain criteria, such as popular session, quiet session, type of preferred avatars available, etc... The protocol has the capability to offer the user the option to create a new session. In this case, *Kn* will receive all the required data with an original copy of the 3D world from the first contacted session manager node. The session manager role for the newly created session is assigned immediately to *Kn*. Node *Kn* should discover and create direct communications with other session managers in the virtual environment.

To minimize the user effort in selecting a suitable session within session level, we propose a natural session choice. For instance, in order to limit message traffic when the user progresses in the virtual environment, it may be useful to select a session where all the proximity neighbours participate. By using the natural session choices, we reduce the number of out-interest (dissimilar CVE-profile) intermediate nodes during a message transfer. New member node *Kn* should broadcast a session discovery message to collect information about sessions IDs in the surrounding area. Before making any decision about the chosen session, *Kn* should wait for a predetermined amount of time to accumulate more responses. The session ID with a highest hit is preferred as the best choice for the new user. However, when operating in a mobile environment, it is difficult to reason on the returned session ID. Nodes may leave and join sessions at any time, thus we may have session IDs of neighbour nodes that change frequently. To overcome this difficulty, a new node should only analyze incoming requests from nodes in *stable* or *full* state, since we expect nodes in such states to be online for an extended period within the same session ID. Node states control mechanism is described in Section 4.1.2. Also, mobility will not highly

affect the overall performance of the session level due to low mobile characteristics of the MCVE applications as stated in Chapter 1.

At the end of the session level, the new joining node *Kn* is required to register its session ID in the network layer through the inter-layer communication procedure to enable the session context-based routing at the network layer and extend the standard MANET routing. This approach works very well to solve the problem of increased messages' arrival time and node workload since nodes are required to drop the messages at the network layer when no session ID matching is found.

Once a node has a full CVE-profile, it must join the default set of zones delivered in the CVE-profile using the zone level, before starting the execution of the Gnutella overlay formation network protocol, which is described in the next section. The CVE-profile is used as a unique reference to identify a node in the network. It contains the user ID assigned by the system, the session ID, and the set of zone IDs to which the user is subscribing/publishing. Based on the CVE-profile, the system will create the desired multi-tier Gnutella overlay network.

During the execution of this level, a problem may arise when a node leaves the session. We have two departure categories; namely, normal and abnormal. In order to handle an ungraceful leave, we have designed a replication protocol for nodes that play critical roles (Session manger, zone owner, replica node etc...) in the virtual environment. Such node must a have a back-up node in order to take over the control when it becomes unavailable. In a graceful departure, two situations may occur. First, if the user is a regular session member, he will simply send a disconnect message to the session manager. The current world state for the disconnected player is then saved in the session manager. Second, if the leaving player is a session manager or zone owner, the node will select a new node from its candidate list to take over the new role. The old node informs the other session managers of its disconnection and gives

the identity of the new node as session manager. The new node should re-establish a connection with the other session managers and zones owners.

The major obstacle that all MCVE applications must overcome is the intermittent node disconnections. For this, we create a log file in each node to record the latest user actions in the virtual environment. When a node joins the network again within a specific amount of time, the log file is immediately executed. By doing this, we allow the user to recover their last actions that have not yet been executed during the short period of disconnection time; for instance, interrupted zone download may be resumed easily. It is important to notice that in the log file, we restrict stored data to that does not allow the user to cheat or play unfairly. The usability of the log files fail when a node disconnect occurs for a longer period of time. The log file should be initialized when the user disconnects for a longer period of time.

Once the user joins the world using the *Session Level*, *Zone Levels* is required in order to join the default zones IDs received in CVE-profile, and then start building in the virtual environment.

4.1.1.2. Zone Level

The *Zone Level* is applied to perform the necessary tasks to join the default zones. The set of default zones is created based on the application specifications; each avatar has a default starting position and predefined numbers of zones that overlap the avatars' AoI. Once a new node has established its first connection, there is a need to join a set of zones retrieved from its CVE-profile. The pseudo-code is shown in Algorithm 2:

Definitions :

SendJoinZoneRequest (zoneIDs, TTL) : Send an initiate zone request
SendJoinZoneResponse(nodeID, zoneIDs, TTL) : Send an initiate zone response request

.....

Algorithm :

```
HandleJoinZoneRequest(zoneIDs, TTL)
{
        IF (Node serve the requested zone)
                SendJoinZoneResponse(nodeID, zoneIDs, TTL)

        ElseIf (TTL > 0)
                BroadcastJoinZoneRequest(zoneIDs, TTL-1)
}

Handle JoinZoneResponse (nodeID, zoneIDs, TTL)
{
        EstablishConnection (mynodeID, nodeID)
        SendZoneSubscription (myNodeID, zoneIDs)// Subscribe in the zone
 }

Handle ZoneSubscriptionResponse (NodeID, zoneIDs)
{
        ReceiveZoneInformation (zoneData)
        IF necessary
                DownloadZoneData()
}
```

Algorithm 2: The Join Zone Procedure.

When a node broadcasts a join zone message, any node serving the requested zone (query matching) should reply to the new node member *Kn*. If no match is found, the query is propagated until the TTL expires. The responding node communicates the information to the zone owner so that the latter can update the zone member and candidate lists. The zone owner then sends update message to all other zone members informing them about the new member node. An additional update message is required to inform the session manager node about the new user. It may be the case that *Kn* dos not receive any response due to inactive zones or limited TTL value, thus, after a specific time-out, the node should communicate with the session manager node to request the zone data. The session manager will send the zone state data to the requesting node, and if the zone is inactive, *Kn* becomes immediately a zone

owner of this zone. To control the node workload and to respect the maximum node degree connectivity, nodes may discard a join zone request even if a match has been found.

In the proposed protocol, we favour a broadcast message to request a join for a specific zone instead of directly communicating with the session manager. This approach has two major advantages:

1. Decentralizing our system as much as possible.
2. Minimizing the bottleneck in the session manager node.

4.1.2. Phase 2: Multi-tier Gnutella Overlay Formation Protocol for MCVE

As described earlier, when building a mobile overlay network for MCVE applications, the resulting network should have the ability to counteract the effect of the limited capabilities of mobile network devices and the flooding problems. In a virtual environment, we characterize the users by their CVE-profile. Nodes must have knowledge about their neighbors having the same interest in order to overcome the settled problems. By "same interest", we mean both nodes should have same session ID and maximum number of common publishing / subscribing zones. The innovative idea here is to extend the IP routing so as to create an efficient multi-tier overlay network that reflects the physical virtual environment position of the user. Nodes place themselves in multiple levels depending on their set of zones, retrieved from their CVE-profile via logical links, to other nodes sharing the same zone. Each node is responsible to maintain a resource table called *NhTab*. Figure 7 shows the structure of our network overlay.

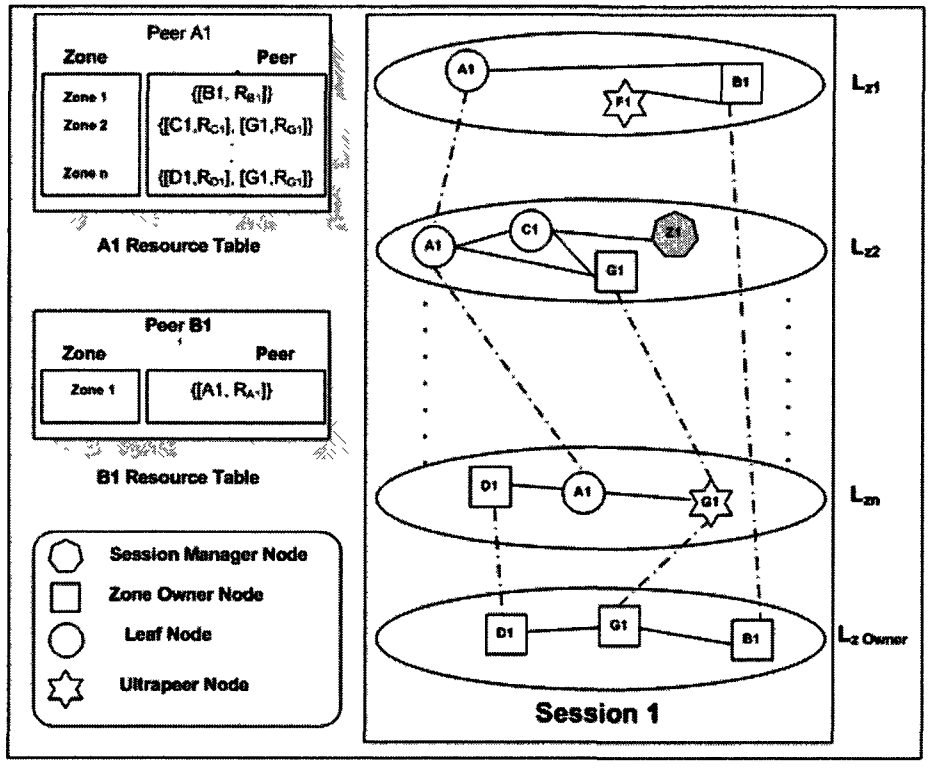

Figure 7: Structure of Multi–tier Overlay Network

Each node *n* should maintain the resource neighboring table. Each entry in *NhTab* has the following information:

- *Node ID*: represents the node ID of the neighbor node.
- *AoI* radius, denoted R_n: represents the Area of Interest radius of the avatars. This information aids the node in determining the location of the node in other zones during a query request.
- *Node class*: describe the node class of the neighbor node, i.e. leaf, Ultrapeer, GUS member, etc. The node class categorization is described in Chapter 5.
- *Time stamp*: the time stamp of the last PING message.
- *The P value*: also referred to as a utility value, it represents a combination of VE user interests and routing metrics of the responding neighbor node.

For the sake of simplicity, in Figure 7 we show only the Node ID and the AoI radius in the resource table. The resulting approach should offer better resource management, and meets the heavy demands of network capabilities, including latency and bandwidth. MCVE is a network delay and bandwidth-sensitive application; update messages should be received within acceptable delay [49], [43] and [108] to maintain the feeling of immersion. Once a node joins the world, it immediately starts discovering VE members that are sharing the same interest, in order to create logical connection links and build the network overlay, described in Figure 2. Our proposed protocol defines two message types *(ping/pong* and *link_request/link_response)* and four node states *(full, stable, connecting, and initial)* that are related to the size of the neighboring resource table maintained in each node. The use of the node state technique is inspired from, albeit being quite different, the approach outlined in [47]. The basic idea in [47] uses three states to control the physical distance between mobile nodes. Indeed, in MCVE, the collaboration aspect is more significant than the physical distance, since nodes that share the same interest have more opportunities to exchange information than nodes in proximity. The node state approach is important for several reasons. First, it makes the protocol more appropriate when deployed in a mobile environment. Most existing protocols run the discovery process permanently without realizing any new discoveries. Thus, there is a certain energy cost and network bandwidth consumption to process the periodic message being exchanged in the network. Second, there is a need to control the node degree connectivity, which should be proportional to the node capacity. The developed scheme imposes limits on the number of required open connections in the neighboring table. The maximum size of the neighboring table is denoted by nb_{Max}. To avoid overdoing the discovery process, we allow the node to maintain a minimum number of connections, denoted by nb_{Min}. The choice of these points depends on node performance and network size. Node state is assigned based on the current size of the neighboring table. Once a node has a neighboring table size equal to nb_{Max}, it reaches the *full* state; no acceptance of incoming connections and/or discovery processes is necessary. The node in this state would only maintain its current logical connections. When the neighbor table size

reaches the value nb_{Min}, the node state is set to *stable* and no peer discovery is required. However, the node would accept new connections from other nodes with the same interest with respect to the nb_{Max} value. When a node has a neighbor table size of less than nb_{Min}, its state is set to *Connecting*. In this state the node would periodically run the discovery peer process, while accepting any new incoming connections to fill up its neighbor resource table and try to reach the *stable* state and then the *full* state as quickly as possible if possible. In each of the above states, the node would ping all active connections in the neighboring table in order to maintain updated routing information. An important situation that we should consider in this scheme is connectivity. Since we apply a link selection policy in the discovery process, it is possible for a node to have an empty neighboring table. The *initial* state is therefore introduced. A node in such a state would means that no available nodes with the same CVE interest are reachable within a given TTL. The system would have to react immediately, because nodes in such a state deteriorate the overall performance of the overlay network. In this later state, after a timeout, the node sends a burning PING message with restricted CVE-profile <*UserID, Session ID, null*> to discover at least a few nodes in the same session without enforcing the presence of publishing/subscribing zones in the discovery. When, again, no response was received, a second burning message <*UserID, null, null*> is sent to discover nodes in the network without any restriction. In our system, we should not apply the node state to nodes having the session manager or zone owner role, because these nodes should have an overall view of the network topology. Messages like *link_request/link_response* are used by peers to establish logical connections with other members in the world; users must be aware of the mobility of their neighbors. As aforementioned, a probe message is periodically sent to active connections in order to check the connection conditions and detect any connection failure within a very short delay after its occurrence. PING and PONG messages are used in a manner similar to Gnutella in order to maintain the active connections for each node, and also to discover other nodes with the same CVE-profile so as to create the MCVE overlay network. Most existing protocols [5], [38], [40], and [100] do not consider the logical

connection performance. During a discovery process, a node may receive many discovery response messages. Therefore there is a need to classify these messages according to a measurement metric called ***Utility Value***, denoted as !"in order to overcome the inefficient connections problem in the overlay network formation process. This value represents a combination measure between the VE user interest metric and a routing metric. The *P* value is intended to bring the system to a state where each node can negotiate its outgoing connection with other nodes in the network, in order to sufficiently reflect two primary parameters (higher matching of the attached CVE-profile, and the number of hops for the underlying physical network path). This allows an intelligent rule-based neighbor selection so that the neighbor with highest *P* value is always retained by the originator node. The P_{on} value between a requesting node *o* and a node *n* is calculated using the following expression:

$$P_{on} = g_n * \left(w_1 * Mp_{on} + w_2 * 1/Nhops_{on}\right)$$

$$g_n = \begin{cases} 1 & If\ node\ n\ is\ in\ full\ or\ stable\ state \\ 0 & Otherwise \end{cases} \qquad 789$$

In expression 1, we introduce a Boolean function #$ to indicate the state of the node *n.*, Nodes in a stable or full state generally offer more spreading information, and guarantee a certain degree of stability to the topology since the user has already spend a significant amount of time constructing the virtual environment. Node in such state eliminates the risk to select a neighbor that just joins the VE or traveling a zone. The CVE-profile matching, denoted as %&'$"is defined as a percentage ratio between the number of zones common in both nodes *o* and *n*, and the zones set size of the requesting node *o*. The ()'&*$ parameter represents the number of physical hops in the network path between nodes *o* and *n*. This parameter is propagated to the application layer via the inter-layer communication channel. Both *Mp* and ()'&*

parameters are associated respectively with weight factors + ,$and + -. The weight factors allow us to specify a desired balance between the zone matching ratio and the network path. Practically, nodes with higher degree of connectivity offer more spreading information. The *P* value is related to the link change for a particular mobility model. Our results demonstrated that reduced mobility implies more static nodes and slower rates of path changes. Therefore, low mobility models tend to offer better performance for MCVE applications. For both peer-to-peer systems and ad-hoc networks, the lack of out-interest intermediate node during a communication process could easily cause network performance degradation. For this purpose, we extend the PING and PONG message packet format to carry out the extra information related to the CVE-profile and the utility value. We run am extensive series of simulations in NS2 [80] environment in Chapter 7 in order to find out the appropriate values of + ,$ and + -.

Originator node uses the proposed link selection policy to choose their neighbor connections. Depending on the node state, three rules are provided:

a) When the requesting node is in *full* state, and if it receives a connection request from a node having a higher *P* value than any of the already neighboring nodes in the resource table, the node attempts to connect to requesting node. When a connection is established, the node should drop the worst connection with the lowest *P* value in order to avoid violating the maximum number of required connections nb_{Max}.

b) When the requesting node is in *stable* state, the node will favor a connection with a *P* value above the threshold value denoted as *Tp*. *Tp* typifies the average *P* values of already available connections in the node. Each node independently updates its threshold value during the run time. By using such a scheme, nodes continually improve their logical links, so as to guarantee an effective overlay network over the time during the play.

c) When the requesting node is in *connecting* or *initial* state, it should accept any connections from nodes without imposing any conditions on the *P* value, in order to reach the *stable* state as quickly as possible.

Nodes may receive messages with equal ! .values. In this case, to break the tie the connection with the lowest *NHops* number is selected first. The criterion of the lowest node ID is used if the tie persists. To operate properly and to reduce the network traffic, nodes with *Mp* value equal to zero should not respond to any PING message. However, when a node receives a burning PING message, it should respond with a PONG message. The proposed selection link policy is meaningful for MCVE applications since it addresses the concerns of inefficient connection in the overlay network, and at the same time ensures a better network performance compared to other existing scheme [21], and [40]. In Algorithm 3, we briefly describe the pseudo code of the node state assignation.

Definitions:

C_degree: represent the current node degree
P_connection: represent the ! value for the received connection
LP_connection: represent the lowest *P* value in the neighbouring node resource table
TP_connection: represent the threshold ! value to dropping a connection
nb_{Min}: represent the number of connections for switching from connecting to stable state
nb_{Max}: represent the number of connections for switching from stable to full state
State_of_peer: represent the current state of the node

.....

Algorithm:

```
IF(State_of_peer=Initial)
        While (C_degree>=1){
                State_of_peer= Connecting
                Send_Discovery_Msg()
                Accept_Incoming_Connection()
        }
Else IF (State_of_peer=Connecting)
        While (C_degree>=nbMin) {
                State_of_peer= Stable
                Accept_Incoming_Connection()
        }
        While (C_degree=0){
                State_of_peer= Initial
                Send_Discovery_Msg()
```

```
                Accept_Incoming_Connection()
        }
Else IF (State_of_peer=Stable)
        IF (P_connection>TP_connection){

                Accept_Incoming_Connection()
        }
        While (C_degree= nb_Max){
                State_of_peer= Full
                IF (P_connection>LP_connection){
                    Drop_Weak_Connection()
                    Accept_Incoming_Connection()
                }
        }
        While (C_degree< nb_Min){
                State_of_peer= Connecting
                Send_Discovery_Msg()
                Accept_Incoming_Connection()
        }
Else IF (State_of_peer=Full)
        While (C_degree= nb_Max){
                State-of_peer= Full
                IF (P_connection>LP_connection){
                    Drop_Weak_Connection()
                    Accept_Incoming_Connection()
                }
        }
        While (C_degree< nb_Min){
                State_of_peer= Connecting
                Send_Discovery_Msg()
                Accept_IncomingConnection()
        }
        While (C_degree>= nb_Min) and (C_degree<nb_Max){
                State_of_peer= Stable
                Accept_Incoming_Connection()
        }
End IF
```

Algorithm 3: Node State Assignation Protocol

In this protocol, the node is required to run the following main procedures.

- **Discovery procedure**: Used to discover other neighboring nodes in the virtual environment based on the already defined *P* connection value.
- **Smart Selective Forwarding procedure**: Used to forward queries in the overlay network.
- **Search procedure**: Used to request any data such as zone data and shared objects and so on.

- **Maintenance procedure**: Used to maintain connections in the logical neighboring resource table; any outdated connection should be removed and replaced by a new one.
- **Download procedure**: used to download zone date when a QUERY HIT is received.
- **Zone Management Procedure**: used to manage the zone when avatars move forward in the virtual environment.

a) Discovery Procedure

Once the first phase is established as shown in Section 4.1.1, the node should discover other peers with the same interest; it sends a discovery PING message containing the current time and its CVE-profile. By using extended MANET routing, we allow the network layer to decide on the kind of message that it should send in response. Prior to that though, the node must register its session ID through the inter-layer communication in the network layer. Discovery messages with different session IDs are discarded at the network layer. There is no need to have the application layer process such messages since they are out of the scope of their interest. Except for control queries, all other queries context are based on the user interest in the virtual environment. For instance, there is no significance for a user participating in session (ID=1) to handle or process query related to session (ID=2). The Virtual Environment and the user behavior in each session may be completely different. Therefore, the session ID filtering in the network layer brings many benefits to the system performance. Figure 8 shows a comparative scenario between our MANET routing and the typical MANET routing used in [24], [40], and [100]. We use the standard OSI network stack -7 layers-; however in MANET some layers may not exist.

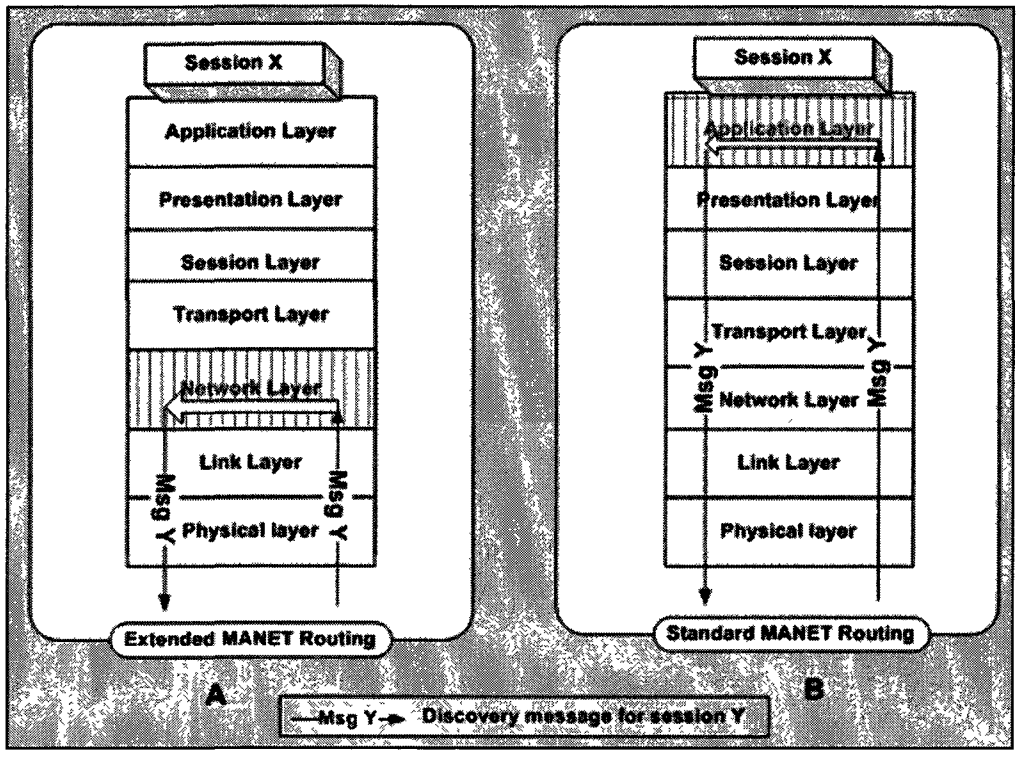

Figure 8: Proposed routing Approach Compared to Standard Approach.

Considering Figure 8A, the average process time (***proc_T_A***) from the moment a user receives a message to the moment the user broadcasts the message is calculated as follows:

$$proc\_T_A = 2*proc\_phl + 2*proc\_lnk + 2*proc\_nt \quad (7.8.9)$$

Where *proc_phl*, *proc_lnk*, and *proc_nt*, represent: the average process time taken by the physical layer, link, and network layer, respectively. If we assume an equal time process (***proc***) in each layer, *E1* becomes:

$$proc\_T_A = 6*proc$$

While in Figure 8B, the average process time (***proc_T_B***) is calculated as follow:

$$proc\_T_B = 14*proc \quad (7.8.10)$$

[1] Describes the amount of time that it takes a message to pass through all the network stack's layer

We may rewrite E2 in a more general format in order to take into consideration the number of layer being used in the layered architecture.

$$Proc\_T_B = 2*numL\ *\&/'\ 0$$

Here, ***numL*** represents the number of layers in the network stack. The upper boundary of ***Proc_T_B*** is ***14****!"#$ the lower boundary is ***8****!"#$ Clearly, the extended routing approach has an impact on the message process time, and therefore, on the network delay, which is considered to be an important performance metric for the MCVE system. Extended routing should not be applied on the burning discovery messages, so any received message must be forwarded to the application layer. The discovery procedure begins when the node starts periodically sending a PING discovery message containing its CVE-profile. Two different situations may then arise.

1. ***No PONG reply is received***. This may occur for two reasons: (i) there are no nodes with the same interest that can be reached by the specified TTL; or (ii) all reached nodes are in a full state. A periodic PING message with an incremented TTL should be transmitted until the node discovers a few neighbors. If no answer is received after a Max_TTL is reached, the node must send a burning ping message.
2. ***A PONG reply is received***. This may also occur for two reasons: (i) a node is discovered by another node with the same interest; or (ii) a node is discovered by another node but with a burning ping message. In either case, the node must make a logical connection with the responding node and should add it in its neighboring table. However, when a node reaches the stable state, it may send a drop request to neighbor nodes having connections made via burning messages. The drop process is not natural; nodes in *connecting* state have the right to ignore this drop request, if its state will go down to *initial*. In both cases (i and ii), the message should shifted to the application layer node in order to be responded properly when query matching is found.

Once a message is received in the application layer, the node should consult its resource table for a query matching.

- When matching zones are found, the receiving node will reply with a PONG message, and forward the query to other neighbor nodes in order to increase the query hit matching. However, when complete matching is detected, there is no need to forward the query. For example, when the CVE-profile requesting node is <NODE1, 1, {2.5}> and the CVE-profile of the contacted node is <NODE2, 1, {2, 3.5}>, a complete matching is detected since the set of zones for the requesting node is included in the set of zones of the contacted node. The PONG message will contain the current time, time of the corresponding received PING, number of common zones, and the current connection degree value. Once the received incoming *P* value corresponds to an ideal scale (*P*>=*Tp*), the originator node sends a "*link-request*" message in order to establish a connection with the responding node. The responding node will reply with a "*link-response*" message. Both nodes begin the Gnutella handshaking process and establish a direct logical connection with the responder node. It is important to note that the node in a *connecting* or *initial* state should accept any reply even when the *P* value is higher than the expected average, in order to reach the *stable* state as soon as possible. We may expect to receive queries more than once; in this case, queries should be discarded by the received node.

- When no zone matching is found, and instead of applying a standard forward scheme used by many systems in the literature [21], [24], and [38], the node should use a smart partial forwarding mechanism, described in the next Section B, in order to reduce the network traffic without affecting the query hits results.

Nodes contacted during the discover process decide whether to reply to the query or not, even when a matching is found based on their workload and connectivity degree.

The system must consider the CVE-profile changes when a user move progresses in the virtual environment. Nodes should be aware of their CVE-profile updates and should update their neighboring table to reflect the CVE-profile. This process is handled in the maintenance procedure described in Section C.

b) Partial Query Forwarding Mechanism

Based on the AoI radius of each peer and the peer's actual zone in the virtual environment, we are able to predict the set of neighbor nodes that should receive the query when no matching is found. Unlike other existing scheme [24], [38], and [48], instead of forwarding the query to all neighbor nodes in the resource table, we select only nodes that have a higher chance for this query matching. The proposed partial query matching mechanism works as follow:

When a node receives a query and no matching is detected, it consults its resource table in order to build a candidate list that contains nodes that may offer this query matching. A node is chosen to be a member of this list if its AoI might overlap with at least one zone in the query. Knowing the node's current zone and its avatars' AoI radius, we can rule out all the other possible zones where the node could be subscribed. For this, the user needs to have the virtual environment map in order to know the zone plan and their sizes. Each zone is completely and uniquely identified. When the requested zone ID is adjacent to the zones in which the user is currently participating (and may be covered by the AOI), the node is selected as a member for this list. For instance, using a zone with a side length t and an AOI diameter equal to $t\sqrt{3}$, nodes subscribed in Z_5 will receive queries about all adjacent zones -Z_1, Z_2, Z_3, Z_6, Z_8, and Z_9- as shown in Figure 9. However, it may be the case (worst case) when a node is subscribed only in Z_5 (it is exactly positioned in the middle of the zone).

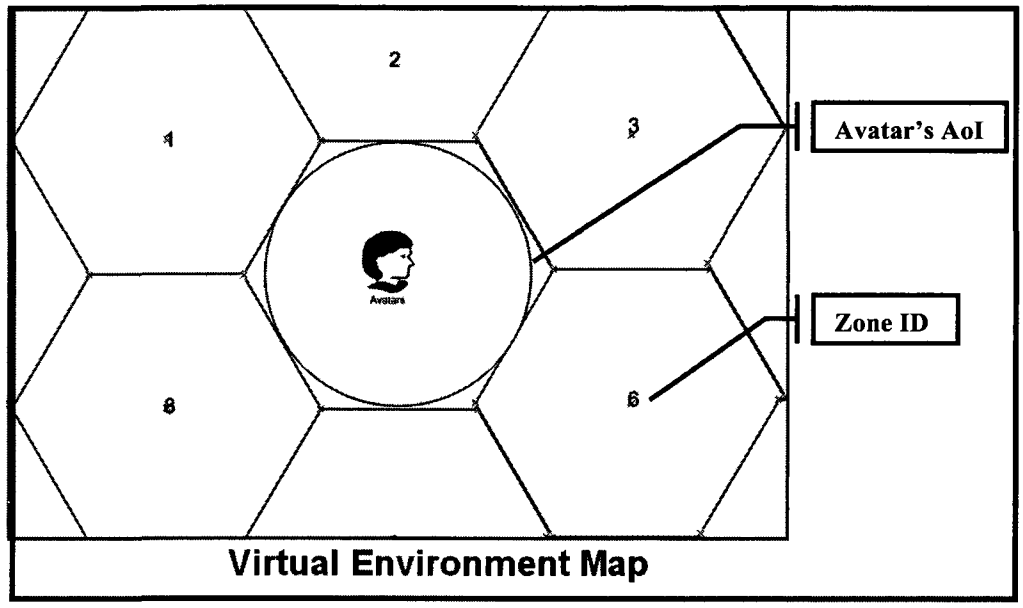

Figure 9: Partial Forwarding Mechanism

This mechanism brings many benefits to the overall system: it first reduces the network traffic, since messages are forwarded only to selected nodes, second it does not affect the query hits results.

c) Search Procedure

In CVE applications, most of search queries are about zones data and shared objects in a specific session. This procedure works in a similar way to Gnutella. A user sends out a search request (Query) about a shared object or data zone; this Query is then propagated through the network, and each node that has a match sends its result back in a Query-Hit message. A node does not have to respond to a query if it has no matching items. However it should forward the query to its neighboring nodes. The matching process in the search procedure begins with an attempt to match the session ID in the received query message with the session ID of the requestor node at the network layer. If matching occurs, the query is pushed to the application layer for more matching criteria. If the query session ID cannot be matched, the query message is then forwarded to all its neighbors. To save more on network bandwidth, we use a selective forwarding scheme as described above. However, we may not receive a

query response due to inactive zone or limited TTL. Thus, after a predefined time out, the originator node must re-send the query to the zone owner layer. The zone owner layer communicates with the session manager node and requests the query on behalf of the originator node. Again here, in this process, we favour a broadcast message request to search for a specific zone, instead of directly communicating with the zone owner or session manager node because of two major advantages: decentralizing the system as much as possible to eliminate the single point of failure and reduce the network traffic in the session manager and zone owner nodes. Usually, a search zone query is followed by a download action. In order to control the network bandwidth and the node workload, overloaded nodes will not respond to the query, even when a matching is detected; the download task requires extra processing power and network bandwidth availability.

d) Connection Maintenance Procedure

Since we operate in a dynamic environment, a maintenance procedure is needed to maintain the current logical connections for each node. Originator node must periodically probe all active connections in the resource neighboring table using a PING and PONG messages, with TTL equal to one; and remove any dead connections. The connection maintenance works as follows: a PING message is periodically sent to the active connection in order to check the connection health and detect a connection failure within a very short delay after its occurrence. PING messages can expire when no response is received from the remote nodes after a certain amount of time has elapsed. In this case, the node removes this connection from its neighboring table. Based on the simulation results, the system performance proved promising when probe messages used a DSDV as an ad-hoc routing protocol, since such messages follow the same proactive behaviors.

The CVE-profile must be adjusted when a user moves forward in the virtual environment; therefore, the ! value may vary over time. Nodes should monitor the ! value for each connection when exchanging probe messages, and a connection with a

null *P* value should be removed. However, nodes in *stable* of *full* states can drop a connection with the lowest *P* value and replace it with a better one having a *P* value higher than *Tp*.

e) Zone Download Procedure

A user navigates the virtual environment via the avatar [85], and the 3D zone data may not be present on his local machine. Thus, there is a need to download the required zones, either from the server or from other users serving this zone, within acceptable delay. The download action requires a timely delay in order not to stress the user during his/her interaction with the virtual environment. In Gnutella network, after receiving a query hit, the originator node sends a HTTP message to the node serving the zone. Both nodes negotiate a direct communication and start transferring the data. The HTTP protocol is able to resume file transfers in case of link error. Indeed, in MCVE applications, it is not enough to build a transfer protocol to simply download 3D zone and multimedia data. Users may request multiple zones depending on their avatars' AoI radius during a move action from many nodes. Many existing protocols [8], and [9] applied for image based rendering can be used in order to predict next upcoming zones that need to be downloaded. Priorities index will help the HTTP protocol to impose more resources for zones that need to be available as soon as possible when the user changes his/her position in the virtual environment.

f) Zone Management

As the user progresses in the VE, there is a need to perform the necessary tasks to join and leave a set of required zones. According to our design approach, every node should join and leave an equal number of zones during a move action, depending on its avatars' AoI radius. The join and leave zones procedures are given below:

-Join a zone: Once the zone data transfer is completed, the originator node sends a join zone message to the responding node. The responding node communicates the information to the zone owner so that the latter can update the zone

member and candidate lists if required. The zone owner then sends updates to all other zone owner members to be aware of the new user. An additional update message is required in order to inform the session manager node of the changes. When a join zone confirmation message is received by to the originator node, it becomes officially a zone member. The user can now start a new discover process since his/her CVE-profile has been changed.

-Leave a zone: a join of a new zone action should be followed immediately by a leave of an old zone in order to respect the number of zones handled by mobile devices. When a node leaves a zone, two cases are possible:

- If the leaving user is a regular zone member, then he/she will simply send a leave message to the zone owner node. The zone owner will remove the user from the member list and send a message to all users informing them about the changes in the member list as well as in the candidate list if required. A leave confirmation message is sent to the leaving user. The user can then safely leave the zone.
- If the leaving user is a zone owner, we have two cases. If the user is the last member in the zone, the user will send all of his/her zone state data with all user data to the session manager node in order to be saved for next login, and the zone will become inactive. If the candidate list (list of potential nodes that can take the zone owner role) is not empty, the node selects the next element in this list to become the new zone owner. When the contacted node accepts the new role, the new zone owner will inform all the zone members of the changes and other zone owners in the same session. Before leaving, the old zone owner will send all the data required to the new zone owner. The old zone should then be retired and removed from the candidate list. However, when the candidate list is empty the node selects randomly a node currently participating in the zone in order to take the role until a potential node joins the specific zone.

When modeling the zone management procedure, we focus only on the graceful nodes departure. The abnormal departure of nodes is covered in the replication protocol handling section.

4.1.3. Persistency and Replication Scheme for MCVE

In this section, we discuss the feasibility of using a mobile Gnutella network to design a persistent protocol for MCVE applications. In order to significantly improve our system's performance, the persistency mechanism is complemented by a dynamic replication protocol to ensure that neither the network, nor the nodes will fail. The replication protocol is implemented to enhance the data availability and improve system recovery when a node leaves the network with persistent data. However, efficient replica management remains a challenging problem due to the inherent, unreliable, and unstable nature of MANET. Mobile ad-hoc network can lead to frequent network changes, which brings in lower data availability than that in conventional networks. Existing schemes [34], [70], and [83] in wired networks may not be applicable to MANET, as they do not consider underlying topology changes that contain the information. In our proposed protocol, participating nodes update the copy residing in the owner node to guarantee the latest data state. Using this replication scheme, we can expect performance improvement by analyzing the data that composes the VE in order to determine what data should be retained for the persistency issue.

In this thesis, we classify the virtual environment data into three main categories:

1. The *scene description* file, which contains the background, terrain, and other data such as static objects.
2. The *avatars and object data representation* files. Used to describe the avatars and object representations.
3. The *user session data* file, which is used to record the state of the VE and the player characteristics data. Each user is required to maintain the user data file for

each session in which he/she is participating. Intuitively, only user session data needs to be replicated; other virtual environment data can be retrieved or rendered from nodes already participating in the VE, or from the session manager node. Due to the limitation of the mobile device, we analyze in more detail the user session data characteristics in order to reduce the amount of data that should be carried out from one session to another. Based on our knowledge, we divide user data into three sub-categories:

- *Permanent user data*, which is relevant to all users, and should remain in the system even when the user leaves the network or logs off from the session. Example of permanent user data includes player modification reported to shared objects, or player objects created during run time, such as a store in a shopping mall. Data used for communication and for controlling the sessions and zones are also listed under this category.
- *Online user data*, also called user life play, is defined as data that should persist only when a user is online, and which must be carried over to the user's next login session. This includes user score, and play credits.
- *Home user data* enfold all data not belonging to permanent and online user data. This kind of data does not require any special care since it can be saved locally in the user machine. An example of such data is avatar's characteristics.

After exploiting virtual environment data classification and its characteristics, we can conclude that the persistency and replication protocols need to be applied for permanent user data, while only persistency is applied to online user data. For home user data, there is no need for any special care; it can be saved locally without affecting the proper functioning of the MCVE application

a) Persistency and Replication Background

Persistency is perhaps the most important requirement that should be taken into consideration when designing any collaborative virtual environment system. The

world state will be retained from one play session to another, even when a player is offline. DIVE [34], EverQuest[32], UltimaOnline [103], and SecondLife [92] are typical applications of persistent worlds. Other online games such as Quake [87] and StarCraft [99] have no persistent ongoing nature: the world state exists only when a user is actively participating in the virtual environment. In DIVE [34], the persistency is handled by running an impassive process called *PERSISTENT,* which periodically saves the world state in a file in order to achieve failure-proof restarts. An *AUTOPERSISTENT* process controls the *PERSISTENT*s' running process. Like DIVE, NPSNET [75] uses active replication to confront the persistency issue. All object modifications are done locally before there are distributed to all peers. However, there is no detailed description of how to maintain persistency among different users, and no know-how of the way they act during the logoff session. Neither DIVE nor NPSNET provide any CVE data classification; thus, all state data is recorded and saved. This may lead to the consumption of the peer storage capacity, especially for systems that run for life and integrates mobile users. In SPLINE [96] project, an object exists only as long as the application that owns it runs. Persistency of the world is maintained only when users are online. Multiplayer online games like EverQuest [32], SecondLife [92], and Ultima Online [103] are built on client server architecture. Many other online games can be found in [28]. The implementation of a "persistent world" in a centralized manner is easier than in a decentralized architecture. All data is stored in the main server where it can be retrieved and rendered at any time. Each user is required only to communicate with the server. Despite these advantages, the client server architecture has some drawbacks as well. Fist the server represents a single point of failure. Second, the maintenance of the server database requires more effort. Finally, the system may cause a bottleneck for both networking and processing data.

In the context of the fault-tolerance and data availability solutions, mobile CVEs require data replication in order to enrich the persistency requirement. Coordinator nodes that play a major role in the virtual environment must be replicated. For

instance, session manager node, zone owner node, or node owing shared object must have a replica node to play as a backup. In [70], the authors study the replication issue on structured [110] peer-to-peer networks in order to support Massive Multiplayer Games (MMGs). Each participating node and application object is mapped to a unique ID. Objects are mapped to nodes where their IDs are numerically close to the object ID. Regions, also known as zones, are mapped in the same manner and are assigned to a node participant. Each node can easily elect a replica node to take over the work when the node is down. The replication mechanism takes advantage of the structured network to control and manage the replica node. However, such a mechanism may cause many difficulties when a higher portion of replication is requested; in DHT, a hash function decides in which node the replication must reside, and so in certain cases, nodes may become overloaded or unable to be handled by the closest node because of the enormous amount of data that the node must store. To the best of our knowledge and experiences, the replication issue has never been seriously studied in MCVE research area. Usually, in mobile ad-hoc data replication schemes such as [27], [37], and [43] do not allow updates of data. Since updates are permitted by different hosts without constraints, data consistency may be affected.

By using the peer-to-peer paradigm to develop mobile collaborative virtual environment, we permit multiple and separate points of communication. Every node that joins the system can bring more resources to the virtual environment, so that with more users, the system has more resources to preserve the state of the world and hence provide the sense of continuity. The Node classification scheme provided in Gnutella V0.6 [38] allows us to assign the replica role only to a node with Ultrapeer criteria.

b) Description of the Persistency Scheme

The persistency protocol runs a periodic process on the user side in order to store the persistent data in the session manager and zone owner nodes, as well as record the last current user actions in the log file. The process begins automatically when a user

joins a session. The user sends the user data to the zone coordinator, which will then forward it to the session manager where it should be saved so that it will be available to the user when he/she logs in to the session again. If a user wants the latest version of a zone, he/she requests from the zone owner, otherwise the version can be accessed from any neighbor node serving the requested zone. Other new updates can be rendered during the play time. Each zone owner maintains an update count, which incases when a new update is performed on the zone by a participating used. This count plays an important role to maintain consistency between primary copy and its replica copy.

Definition

Z_n: represent the zone of the requesting query
setZone: represent the current set of zones of the user's CVE-profile
uCout: represent the update count
SendUpdate(): procedure to inform the zone coordinator about zone update.
RecordAction(): procedure used to record update action in the log file.
BrodcastUpdate(): procedure used by coordinator node to inform all participating node about an update.

.....

Algorithm

```
ZoneUpdate()
{
        RecordAction()
        SendUpdate()
}
HandleSendUpdate()
{
        uCount++
        BrodcastUpdate()
}
HandleBrodcastUpdate()
{
        If( Zn ϵ setZone)
        {       UpdateLocalCopy()
                ForwardUpdate()
        }
        Else
                ForwardUpdate()

}
```

Algorithm 4: Zone Update Protocol.

The proposed protocol has to efficiently perform when the session manager and/or the zone owner node fails. Therefore, there is a need for fault detection and recovery mechanism. This mechanism is described in the replication protocol in Section **C**. Saving user data in the session manager works to our benefit, since the possibility of cheating is reduced. In order to fulfill the limited mobile device storage requirement, other data that is not likely to be altered for the purposes of persistency, i.e. avatars characteristics data, should be saved in the log file located in the user machine. The log file may be very helpful when a connection interruption occurs for a short delay. For instance, when a node is in a tunnel or under a bridge, the system can resume the last previous actions when the connection is re-established. One of the most practical actions offered by the system is that of zone download resume. There is no need to restart a download from the beginning when an intermittent connection occurs. Many advantages can be obtained through the integration of the log file idea. First, the connection failure-less restarts are minimized. Second, network traffic is reduced, since no broadcast is necessary to relocate the data.

In a normal disconnection scenario, before the user disconnects, the node communicates with the session manager to request disconnection, and transfers its current VE state. The session manager saves the user's state data with the user key reference, and then sends a leave confirmation message to the user. The session manager requires increased storage capacity and network connection in order to store a list of all zones and clients in the session with their join time stamp, and the online users. An update process runs to delete data that is no longer valid for the user, this includes old scores for which only the last value should be maintained. When an abnormal disconnection occurs, the periodic process, with the help of the log file, arranges to reduce the virtual environment state lost. The proposed persistency protocol is given as a simple solution to handle the MCVE application. An ongoing future work is needed to enhance its performance.

c) Description of the Replication Scheme

The primarily concern of the replication protocol is to elect a replica node for nodes playing a coordinator role in virtual environment. By the coordinator node, we mean a node with a key role in the system, such as session manager, zone owner, objects owner, etc. The session manager node may begin to run out of control when the number of users increases exponentially in the session. Any failure in coordinator nodes will deteriorate the overall system performance, and the user will lose interest in the application. Because Gnutella operates under an unstructured network characteristic, it is difficult to adapt any replication scheme developed for a structured network. The closest node (Key ID) to a coordinator node has to be its replica. However, there can be situations where the node closest to the coordinator is unable to handle the replication job or represent a bottleneck; thus, system performance is affected. The requirements for being a replica node there must be no cycle of roles when two neighbour nodes act as both, coordinator and replica at the same time, as shown in Figure 10. For instance, when node *Y* plays a coordinator role for node *X,* thus, node *X* can not play a coordinator role for node *Y.* The *Replica Request* message number three shown in Figure 10 is discarded.

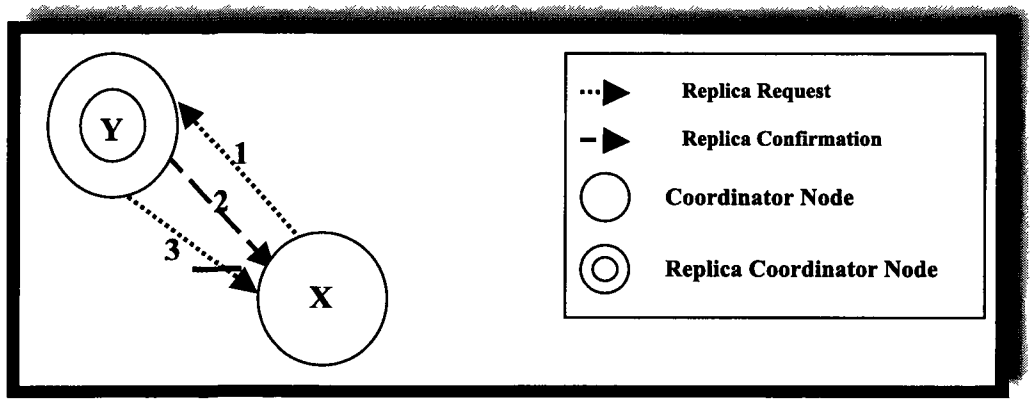

Figure 10: Cycle Avoidance of Replica Coordinator Peer's Role

The protocol is based on an on-demand single replica, but it can be extended, and can handle multiple replicas depending on the join/leave rate and the importance of the replicated data for the whole system. The replica looks up procedure works as

follows: The coordinator node first selects a node with specific criteria (Gnutella Ultrapeer System -GUS- member described in Chapter 5) located in its neighboring table and places them in increasing order (based on their *P* value) into separate list called replica candidate. If there is more than one element in the list, the node with highest *P* value is contacted first. Otherwise, the coordinator node has no choice but to contact the only element in the list. The use of the *P* value to sort the replica candidate list ensures better performance in terms of network bandwidth and message spreading. The node coordinator sends a *REPRequest* message to the top element in the replica candidate list, and the receiving node sends a *REPReply* if it fulfills the replica requirements. This replica selection scenario is shown in Figure 11. Darker node represents an Ultrapeer with a highest *P* value. Therefore, it is contacted by coordinator node neighbour using *Replica Request* and *Replica Confirmation* messages.

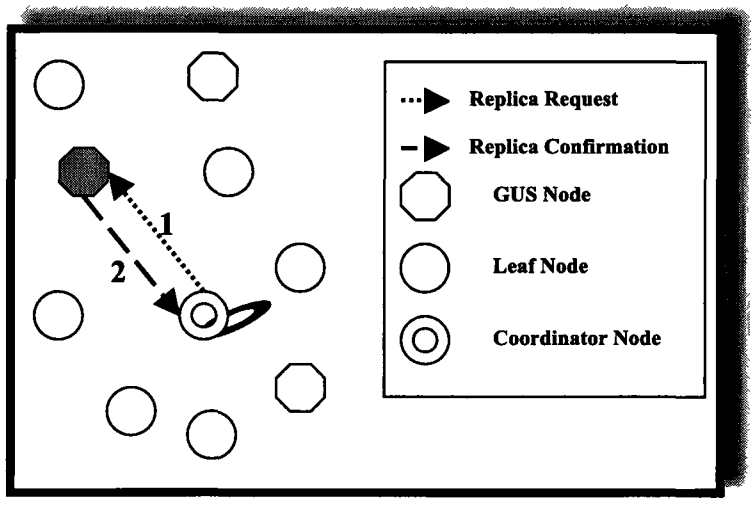

Figure 11: Scenario for Selecting a Replica

The coordinator node waits for a time interval -TIMER- for a response to arrive. If no response is received, the coordinator node then contacts the next element until a response is received. Any contacted node that fails to respond should be removed from the list. When a node receives a *REPReply*, it immediately transfers all of the necessary data that should be replicated. At the end, both nodes should be aware of each other's status by exchanging a PING/PONG message. Once the coordinator

node has found its replica node, also known as a standby node, it should continue controlling the status of the replica node. The coordinator node should ping its current replica node periodically, and vice versa. When a coordinator node sends a PING to its replica node, it expects a PONG message from its replica within a specific amount of time. Here, two cases may arise:

1. The replica node does not answer with a PONG message within the time limit. In this case, the coordinator node will choose a new replica node and send a *REPDisactive* message to the old replica to tell it that "you are no longer my replica." The old replica will reply with a DisactiveConfirmation message, and can then destroy all the replicated data for this coordinator node.
2. A PONG message is received, but with a lower P value than expected, which means that the replica node is no longer suitable for the coordinator node. The performance between both nodes worsens. The coordinator node should look for a new replica with a better P value by running the look-up replication procedure. The new replica node will receive all of the necessary data. Until the transfer is complete, the old replica continues to behave as a replica node. Once the new replica takes over, the old replica will receive a deactivate message from the coordinator node and destroy all the replicated data.

During the execution of the step 1, a problem may arise when the coordinator node assumes that its replica has disappeared from the network (no DisactiveConfirmation was received) or vice-versa, and it selects a new one. As a result, the coordinator may find itself with two replica nodes at the same time because the old replica is not reachable due to mobility or the DisactiveConfirmation message has been lost. To solve this problem, the coordinator node communicates with the session manager after a specified time has elapsed without having received a PING message, in order to deactivate its current replica. Similarly, the replica node communicates with the session manager when no PING has been received from the coordinator node. In the end, both nodes will become aware of each other through the session manager node,

and the session manager takes control of the situation and sends a deactivate message to the replica node if it is still in the network. The coordinator node will be informed to look for a new replica. This solution stipulates that only one replica for each coordinator node should be active under all circumstances. However, this solution increases the replica time recovery, which is the time needed to select a replica node and transfer data. The time recovery may be longer than expected since the coordinator node, when detecting the failure of its current replica, will contact the session manager to select a new replica, this is in addition to transfer time. Time recovery depends on the network characteristics and the size of the transferred data. The advantage of the replica protocol is that it does not produce an extra network overhead. The tradeoff between the coordinator node and the replica node is integrated with the Gnutella maintenance connections, and nodes should probe their neighbors anyway to maintain connections.

Chapter 5

The Gnutella Ultrapeer System (GUS) Protocol for MCVE

The current Gnutella Ultrapeer system proposed in [38] is a promising model to handle scalability and heterogeneity. Data may in fact become unavailable where loose connections among leaf nodes are created. Yet, despite its application in many domains, Ultrapeer selection still requires a great deal of research to adapt to mobile environments. Applications running on mobile devices may need services or resources that are not locally available. Therefore, they trigger a search on other mobile nodes. In Gnutella [38] network, we introduce the concept of the Gnutella Ultrapeer System (GUS), where message flooding is only conducted among Ultrapeers and where all leaf nodes are represented by corresponding Ultrapeers. The Ultrapeer, also called "superpeer", is responsible for processing and relaying queries that come from the leaf-peers and other superpeer neighbors. The proposed GUS paradigm allows a decentralized scheme to run more efficiently, since it provides multiple separate points of failure and thus increases the efficiency of our MCVE application. We exploit the necessity to design GUS from the Gnutella traffic analysis described in [111]. The study summarizes an interesting problem related to the network performance: more than 85% of TCP connections are refused by leaf nodes, and about 13% of the total nodes connected are Ultrapeers. Since *leaf* peers constitute a large portion, almost 87%, of the total population, and suffer from a higher error connection ratio, their behavior has a significant impact on Gnutella networks performance. The situation worsens when a mobile network is applied, due to the fluctuating reliability of wireless links and mobile devices hardware limitation. We perform a dynamic Ultrapeer election based on a score measurement that combines the network behaviors and the user behaviors in the virtual environment. Thus, not

every change in network topology will entail an immediate system re-configuration. The GUS protocol should fulfill the *&+.!$- .#0&,) #$ -,) #0, $&'$ # $We believe that the proposed selection scheme can be considered as a key feature to improve network heterogeneity and scalability requirements.

5.1. Related Work

In this section, we focus on previous techniques used to enhance the unstructured network in order to overcome network heterogeneity and scalability. Super-node selection is a well known approach that imparts organization unto the network. Networks like Gnutella [38] and KaZaa [58] have started to integrate more structure by introducing the concept of super-peer, also called Ultrapeer, to improve system scalability. Gnutella has the most popular super-peer selection method [111]. Ultrapeer nodes must meet predefined criteria: high network bandwidth, large processing power, and storage capacity, and a predefined time must have elapsed before they can become Ultrapeers. Despite all of its advantages, the scheme brings forward several problems that must be addressed. After a certain period of time, if the newest joining nodes fill the Ultrapeer criteria, the system will soon have too many Ultrapeers; or if most of peers are weak, then the system will behave as a centralized peer-to peer system. Compared to pure peer-to-peer network, Ultrapeer systems have higher search efficiency and better scalability. In current Gnutella networks, the selection scheme fails to provide low access from the leaf nodes to the Ultrapeers, and currently there is no way to control the ratio of Ultrapeers to leaf nodes. In the case of KaZaa, little is known about the selection protocol description. However, some researches identify some similarities with Gnutella. H_2O (Hierarchical 2-level Overlay) [69] describe a new selection scheme based on advertisement messages: where each superpeer advertises its information, and leaf nodes cache the advertisement. H_2O provides some flexibility, since nodes can choose the best super-node using the local cache information. Frequent leaving and joining will increase the number of messages being transmitted during the super-node search step. Also, there

is no guarantee that a leaf node will find a super-peer that corresponds to its criteria. Melamed *et all* [74] describe a new selection approach called Araneola to handle flexibility and provide load balancing. However it does not provide any mechanism to deal with the network heterogeneity issue. Li *et all* [109] examines the workload in each node in order to estimate an appropriate Ultrapeers to leaves ratio. According to this technique, each node can decide locally on the ratio by exchanging control message. A simple mathematical model is used to estimate the ratio. Many file sharing applications implement a random walk [18] as opposite to the super-node selection scheme to overcome the flooding problem and to make the file searching more efficient. In the context of mobile peer to peer network, the deployment of Ultrapeer selection is not straightforward. System performance depends on the Ultrapeer capability; weak nodes jeopardize the whole system. Consequently, the super node should be selected based on rigorous criteria and measures. A well-known algorithm [36] that uses the node ID as a selection criterion has been developed. The selection criterion to becoming an Ultrapeer node is having the lowest ID number in the set, including all neighbor nodes and the node itself. Since the mobile network is never stable, a re-configuration process runs frequently.

The main problems with the existing protocols [36], [69], [74], and [111] are that, due to node mobility, the quality of the acquired paths changes over time and certain paths can become broken or contain a weak connection. Additionally, when users move forward in the virtual environment, they must switch their neighbors and select new nodes with the same interest in order to increase the system performance. To overcome the settled problems in previous works, we take a different perspective that not only focuses on the peers' capabilities but also considers both node activity in the virtual environment and network behavior as part of the metric selection. The leaf node selection determines which Ultrapeer is optimal in the surrounding area in order to request a service and become a GUS member. We also aspire to isolate as much as possible the GUS members' selection criteria from fluctuating factors in the network in order to reduce the frequent re-organizations of the Ultrapeer system. GUS members act like the proxies of their leaf-peers and keep an index of their leaf peers'

shared data. However, and as far as the authors are concerned, there has been no report on super-node selection for mobile collaborative virtual environment applications. In the next section, we describe our Gnutella Ultrapeer System -GUS-.

5.2. Description of GUS Protocol

The deployment of an Ultrapeer selection scheme problem is highly challenging because, in mobile environments, a large number of unqualified super nodes may be selected and this can deteriorate the system's performance. Moreover, in MANETs, neither the node characteristics nor the network topology are known a priori. Most of the proposed solutions pay little attention to the dynamic nature of the network and use fixed rules to negotiate the Ultrapeers' capabilities, such as random selection [38], or fixed rule values selection [36]. Consequently, such schemes may not allow the best MCVE application to adapt to the virtual environment's user interest and the dynamic changes in mobile networks. Therefore, when selecting a GUS member, we must consider a combination of the dynamic conditions in the network in the surrounding node area and the user's behaviors in the virtual environment [13]. Before describing the Gnutella Ultrapeer System, we present the node classes categorization used by this protocol:

- ***Leaf***: a node with weak hardware capabilities; if necessary, this node can be updated to become a *mandatory GUS member*.
- ***Ultrapeer***: a node with the following criteria: higher computational power, storage capacity, and network bandwidth availability.
- ***GUS candidate***: a node with Ultrapeer criteria and a specific time session time spent defined by the system during the runtime. The time spent in the session should be long enough to increase the chance of this node not leaving the virtual environment too soon. Such a node can be updated by a leaf neighbor node to become a regular *GUS member*.

- ***Regular GUS member***: a node with GUS candidate criteria currently participating in the Gnutella Ultrapeer System.
- ***Mandatory GUS member***: a node that receives an upgrade message from its direct neighbor to become a GUS member without meeting the requirements of a GUS candidate.

Each node can appoint itself to be an Ultrapeer using an on/off Ultrapeer flag. The switching process between node classes is triggered whenever an improvement in the GUS is required. In the remaining text, we refer to regular GUS members and mandatory GUS members as GUS nodes. When the node status is a GUS member, extra information is carried in the *NhTab* as shown in Table 3 in order to let a leaf node decide about its associated GUS member.

Ndegree	**Degree of connected node**
NF	**Network Factor**
MF_{cve}	**User Mobility Factor in collaborative environment**

Table 3: GUS Node Information

In the Gnutella Ultrapeer System design, when an Ultrapeer node detects a change in some values defined in the Table 3, it sends an update message to inform all of its neighboring nodes about the changes. There is no need for periodic messages, and thus bandwidth and network overhead are saved. The main purpose of the *NF* parameter is to obtain feedback about the network behaviors in the area surrounding an Ultrapeer. The MF_{CVE} represents the motion of the node in the virtual environment to indicate the rate of zone change during a period of time. In other words, it represents the number of times a node changes its CVE-profile during a specific time slot. The *NF* value is defined as:

$$NF = \left[\frac{Al - Dl}{Al + Dl}\right] \qquad (2)$$

The following values are used:

- *Al*: represents the cumulative sum of added links during a discovery process.
- *Dl*: represents the cumulative sum of dropped links during a maintenance process.

After each network event (ne), the *Ultrapeer* will perform the following actions independently:

If (network_event = drop) **then** *Dl*++

If (network_event) = add **then** *Al*++

An update message is sent to the neighboring nodes to inform them of any changes. It is important to note that the dropped links related to the CVE-profile are discarded, since they do not provide any feedback on the network. In (2), we consider also the difference between *Al* and *Dl* to limit the effect of unqualified Ultrapeers. For instance, if Ultrapeers u_1 and u_2 have respectively: Al_1= 4, Dl_1 =1 and Al_2= 3, Dl_2 =0 then both nodes have the same behaviors. However, u_2 is clearly a more viable option as it does not drop any links. We thus enhance the *NF* metric by integrating the sum of these values. In our proposed selection scheme, an Ultrapeer with the highest *NF* and lowest MF_{cve} values reveals the maximum benefits to a leaf neighbor. In addition, we allow a fixed margin for the Ultrapeers to stabilize. When it changes its CVE-profile, the Ultrapeer must wait for a specific time before it can be selected as a GUS member, even when all the required criteria are reached. By using a waiting rule, we limit the effect of the burden on the CVE-profile and on session ID changes. The waiting time should be long enough to prevent the node from selecting GUS members that are only traveling through a zone. To operate properly, there is a need to initialize the activity value to 1 when a node joins a session for the first time. The pseudo-code of the Ultrapeer selection protocol is shown in Table 4.

Definition:

List$_{nf}$: List to store the received NF's values from the neighboring nodes
List$_{MFcve}$: List the received MF_{cve}'s values from the neighboring nodes.
GetIndex(element in* List$_{nf}$*): Procedure returns the index position of a given $List_{nf}$ element in the List$_{MFcve}$.
index: Used to the store index of the selected node in List$_{MFcve}$. Initially set to an infinite number.
GetNodeID(index): Procedure to return the node ID of the corresponding index.

Algorithm:

1. Sort the $List_{nf}$ elements in decreasing order.
2. Sort $List_{MFcve}$ elements in increasing order.
3. Place the first *n* highest elements in $List_{nf}$ in a candidate list called List$_{cd}$.
4. **For** each element in List$_{cd}$, i=0, ... List$_{cd}$.size **do**
   tmpIndex= *GetIndex(*List$_{cd}$ *[i])*
   If index > tmpIndex
   index= tmpIndex
   end if
5. **End for**
6. nodeID=*GetNodeID(index)*
7. **Return** nodeID

Table 4: The GUS member selection algorithm.

Based on the returned node ID, the leaf node contacts the *Ultrapeer*; however, the contacted *Ultrapeer* may reject the request due to its workload or availability. In this case, the leaf node removes the *Ultrapeer* from the $List_{cd}$ and selects the next candidate. When none of the elements in the $List_{cd}$ list can accept the request, the leaf node uses the remaining elements in $List_{nf}$ as $List_{cd}$ to retrieve the appropriate GUS node ID. If no GUS members can accept the request, the leaf node uses the GUS candidate as $List_{nf}$ and runs again the procedure described in Table 4. If the problem persists, the leaf node randomly selects an Ultrapeer neighbors as a GUS mandatory member. However, in the event that all contacted nodes reject the request; the leaf node declares itself a mandatory GUS member. In terms of reliability issues, if a GUS member leaves the network ungracefully, data may be lost and the VE may be affected. Therefore, any leaf node must run the whole process of selecting a GUS member replica in order to limit the effects of the unexpected departure. All consequent messages must be broadcasted to inform other neighbors. As mentioned earlier, unfortunately, the selection algorithm is not always applicable and some exceptions may occur. The system should have the ability to force a node to be a

GUS member if the zone population or the overall Ultrapeer-to- leaf ratio becomes very low.

The resulting approach should react quickly in substituting a mandatory GUS member for an available regular GUS member. Therefore, each leaf node neighbor should monitor its resource table and then exchange its mandatory GUS member for a regular GUS member. This scheme seems more reasonable since a leaf neighbor can independently discover a regular GUS member in the surrounding area instead of letting the mandatory GUS member decide on its behalf. To allow the GUS member to control such a situation can be too costly and time consuming because it would be difficult to find a GUS member that accepts many leaf node connections at the same time.

To handle adaptability and to maintain GUS quality, a leaf node can switch to another GUS member neighbor when one with better performance can be found: this occurs when the node's current Ultrapeer values are compared to the newly received Ultrapeer values during the runtime. For this, we enable a self downgrade and upgrade process of the node status. All nodes compute their Network Factor –*NF*- and Mobile Factor -MF_{CVE}- values and broadcast them to their neighborhoods with the values of their other direct neighbors. Thus, each node becomes familiar with the topology information in its two-hop neighborhood. After processing the local collected information, node *n* must perform the following actions.

- If node *n* receives an advertisement *PING* from a GUS member candidate *k* with enhanced or higher *NF* and MF_{cve} values than its current GUS member *l*, then *n* sets its primary GUS member to *k* and its secondary GUS members to *l*. Otherwise, *l* continues offering the service to the node *n*.
- If GUS member *n* receives an advertisement message from a GUS candidate or another GUS member *k* with enhanced *NF* and MF_{cve} values, then *n* should ensure that it will downgrade itself to GUS candidate and set *k* as its primary GUS member; otherwise it remains a GUS member. The downgrade operation is not

natural; *n* should first communicate with its direct leaf nodes by broadcasting a *PING* request message. If one of the contacted leaf nodes replies with a negative answer, indicating that no other GUS member node is available in the surrounding area, then the node *n* should stay and continue to behave as a GUS member and to ignore the downgrade process.

In this way, it is possible to reduce the number of GUS members one after another in a fully distributed manner without affecting the overall GUS performance. Also, GUS guarantees adaptability, flexibility, and reliability for applications. It is important to note maintaining GUS does not generate a lot of traffic since we employ the *PING* and *PONG* messages already used in the maintenance procedure to exchange measurement values.

Chapter 6

A Novel Caching Protocol for MCVE

In this chapter, we discuss the design and the implementation issues of the caching protocol used by our system. This protocol can be seen as an appropriate technique to reduce the amount of network traffic and network latency. We propose two different techniques to handle the PONG and QUERY caching separately. In doing so, a review of previous caching efforts is also described in this Chapter.

In [30], [67], and [93], cached data is usually maintained in a global hash table, with refresh action running as a background process to update the data. Because logical connections in peer-to-peer applications are meant to be independent, it is important for such applications to separate the cached data for each logical connection. As a result, unnecessary traffic messages are avoided since control message are only sent through a designated connection. Note that only Gnutella Ultrapeer System members can implement the caching protocol, because such nodes are more powerful and provide more network bandwidth than any other nodes in the network.

6.1. Related Work

In an unstructured peer-to-peer network, flooding is the heaviest mechanism [94] in terms of overloading the network. Message processing entails an expensive bandwidth network and node computation power. Data caching can be used as an alternative technique to overcoming the flooding problem [94]. Recent LimeWire Gnutella client software implements a pong caching [67] mechanism in order to reduce the pure overhead of PING and PONG messages. During a "pong" forward, a receiving node can save the information as cached data. When a node receives a

PING message, it sends a selection of recently cached data back to the requested user without having to forward the discovery PING message. To refresh the pong data, a node periodically pings all of its logical neighbors' connections, thus consuming extra bandwidth and processing power. Emixode Communications [30] exploits a scheme to overcome the refresh process problem by running a filtering process to update the pong cache data. When a node holds sufficient fresh cache data, it replies immediately with a PONG, otherwise it broadcasts the PING. The main disadvantage with the proposed approaches is that the cached data may take a longer time to be filled, especially for nodes receiving few PING messages, or when the network size is very small. In PReCinCt [93], the caching policy decides whether to cache data or not based on the popularity of the data, the distance between both nodes, and the size of the data. Such an approach requires a routing geographical protocol like GPRS, in order to localize peers. The refresh process represents the main drawback when data caching is employed. Periodic updates of cached data increases the network overhead and aggravates network latency. Therefore, the classical periodic refresh process in a given dynamic environment seems to be impractical. Existing protocols do not currently have sufficient flexibility [93], and [67] to update the cached data, even when a node does hold sufficient valid cached data; it must continually run the refresh process.

Query caching protocols were developed to complement the PONG caching. The query cached results are valid only up to a time out period. When the cached data expires, it is removed from the cache, and the query is forwarded out to its neighbors. Blundell *et al* [6] address the query caching and the response time problems by proposing three schemes with different emphases. Expanding ring is the first scheme, and is based on progressively increasing TTL until a query response is received. This may be an interesting solution for finding the lower bound of TTL. However, in the worst case, the expanding ring may generate more traffic when a resource is far away, thus causing longer response network delay. The second scheme uses keyword hashing to improve query responses. Each node is requested to exchange its shared

resources table with other nodes in the network; most likely, a query is forwarded to peers holding the resource. The last solution selects a random walker peer that in turn forwards the query to one of its peers. This approach may alleviate network overhead, since the traffic is directly proportional to the number of walkers per search. However, a connecter walkers' node leads to an increased query response time.

Unfortunately, all the proposed schemes do not consider the rare queries and the node workload. Usually after a query hit, the node is required to download data from the responding node; the download task depreciates the node's performance. Our caching approach, while being simple, provides a separate control for cached data retrieval mechanism in order to alleviate the network overhead, and accelerate the query response time. Node workload is ensured by a pooling method to alternate the download process between nodes serving the required data. Also, we develop a refresh component to keep the rare query caching results valid for a longer time than that for popular ones.

6.2. Description of the Pong Caching Protocol

The proposed caching protocol [12] and [14] is developed initially to support MCVE applications. However, it can be extended to handle many other mobile applications, such as file sharing and market work. Each node maintains a separate hash table for each logical connection in order to hold the pong caching entries. Each entry in the hash table stores the following information:

- *IP address*: IP address of the responding node.
- *Port number:* Port number of the responding node.
- *Time stamp:* Time indicating when the last PONG message was received. This field is used to clean outdated pong data.
- *Nhops value:* Indicates how far the responding node is on the network.

- *IDs of subscriber/publishing zones:* Set of zone IDs in which the responding node is publishing/subscribing.
- *Total size of shared subscriber/publishing zones:* Total size of shared zones held by the responding node.
- *IDs of shared objects:* The IDs of shared objects controlled by the responding node.

Only Gnutella Ultrapeer System -GUS- members can cache the data. The overall idea of the protocol is shown in Figure 12. We assume each node has four logical connections both approaches are implemented and compared in Chapter 7. When using a separate data table for each connection, the refresh process is also controlled separately. Whenever the data table size falls below a threshold value, the refresh process is started only via the connections having a small data table size.

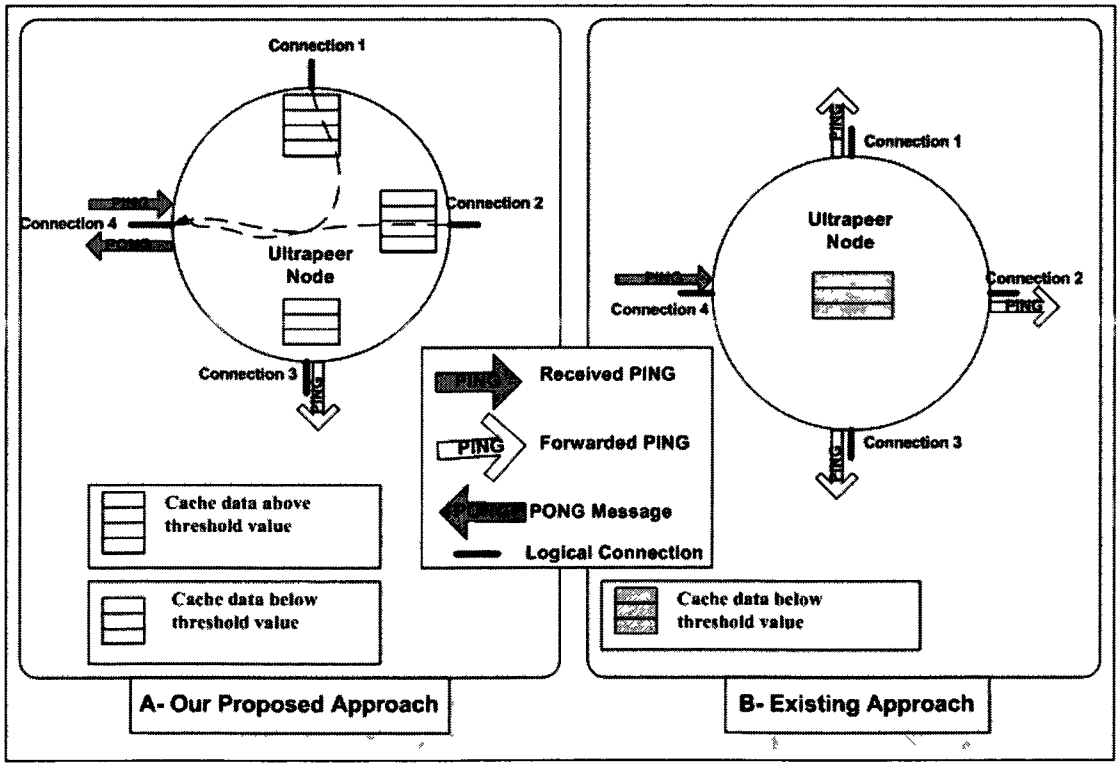

Figure 12: Proposed Pong Caching Protocol versus Standard Approach

Initially, all hash tables contain no pong cache data. A node is required to let the PING message pass. As a PING message generates corresponding PONG message, the PONGs are stored in the appropriate hash table before they are routed to the original sender, according to the Gnutella protocol's specifications.

- If the TTL is not expired (TTL > 0) when a leaf node receives a PING message, the node forwards the message, otherwise it drops it.
- When a GUS members node receives a PING message, two scenarios may commence: 1) The node drops the PING message because the PONG cache hash table contains enough fresh entries to reply with a PONG message; 2) The node forwards the PING message only via connections with a hash table size that is less than a defined threshold value, and at the same time answers the PING from the remaining connections with sufficient cached data, as shown in Figure 13 A. Nodes are required to monitor the hash table size of each connection.

Algorithm 5 describes the pseudo-code of the proposed PONG caching protocol.

Definition:

CT_Size: Current caching table size
Th_Size: The threshold value for letting PING message pass
ReceiveMsg(): Procedure to handle received message

.....

Algorithm:

```
IF (ReceivedMsg().isPing and ReceivedMsg().sameProfile)
        For (each connection i)
                IF (CT_Size_i >= Th_Size_i)
                        ReplyWithPong()
                Else
                        Let the PING message pass only through logical connection i
                End IF
        End For

Else IF (ReceivedMsg().SameProfile = False)
        ForwardPingMsg()
End IF
```

Algorithm 5: Pong Caching Pseudo-code

Intermediate nodes are only required to cache data that is relevant to their CVE-profile; it is not necessary to store data outside their interest. This will obviously reduce network overhead, and conserve the limited storage of mobile devices.

Assuming that all cached data is updated, data that should be returned back to the originator node must meet pre-defined PING conditions. The PING conditions are as follows: Nodes should have a minimum hop value with a higher CVE-profile matching, which is defined as that with same session ID, and a maximum number of common zones that overlap with the zone interest of the originator node. The PONG caching data is dynamically associated with the CVE-profile to handle the collaborative aspect. In a highly dynamic environment, however, the caching scheme might become impractical, as nodes may change their CVE-profile at rapid rate.

Nodes may accumulate older data, so there is a need for a refresh process in order to periodically remove old cached data. Refresh has a number of negative aspects, one being the increased network overhead, another impact of network topology re-configuration. The main issue that we have to consider the frequency of refreshing the cached data; since we use a separate hash-table for each connection, data refreshing is also controlled separately. The refresh process runs separate threads for each active connection, and periodically filters outdated cache data entries at set intervals by sending a probe PING message. One of the advantages of using separate control for each connection is to overcome the network overhead and maintain a balanced topological network view.

6.3. Description of the Query Caching Protocol

The technique of query hits caching has been considered to overcome the inefficient use of network resources, and also to reduce the query response time. Similar to the PONG caching scheme, only GUS members can cache the query hits result they see. We give special care to the limited storage capacity of mobile devices.

For each passing query hit, peers cache the response only when the query description matches its CVE-profile. For instance, if a query is about zone *X*, only nodes participating in zone *X* will cache this response. Figure 13 depicts our query cache scheme. Letters inside the node shape represent the publishing/subscribing zone. For simplicity, only one zone is considered for each node. As shown in Figure 13, intermediate regular nodes will only broadcast any received query hit messages, no caching is required.

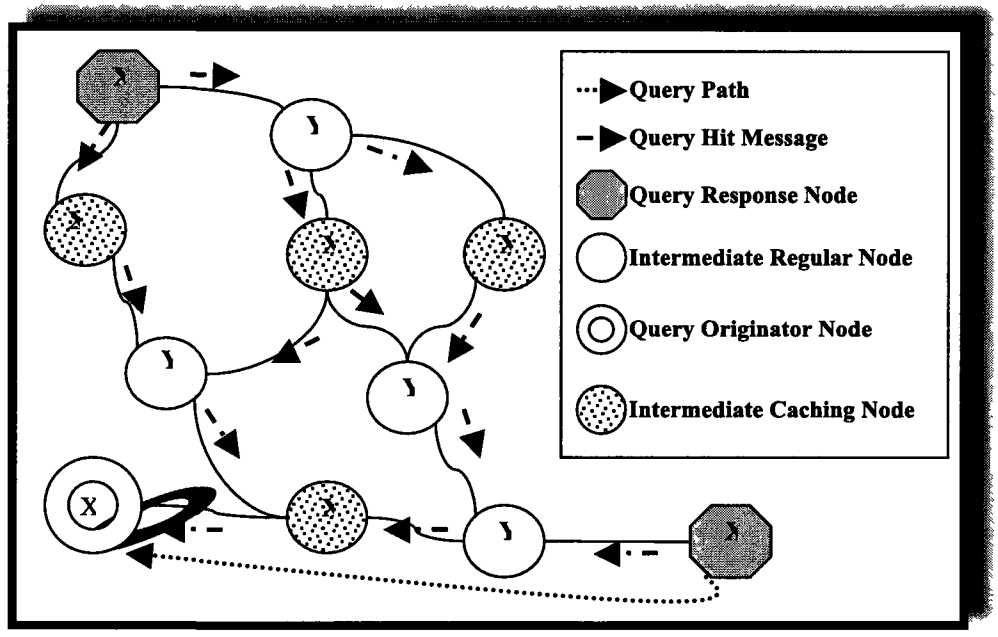

Figure 13: Proposed Query Caching Scheme

Query hits are cached in the cache query table, denoted CQT, according to the tuple of <QID, Query_description, Hits_result> values of the corresponding query. The QID (Query ID) is used as a key index in CQT to determine whether an incoming query hit should be cached so that the duplicated query response among neighboring peers may be minimized; Query_description provides a brief description of the query. Hits_result is a vector used to memorize the set of responses to the corresponding Query ID. Each entry in the Hits_result vector contains the following information:

- *Key IP address*: References the node that owns the resource.

- *Nhops value:* Identifies how far the node that owns the resource is.
- *Time stamp*: Indicates of the last received query hit.
- *Speed*: Expressed (in kb/second) to define the speed of the responding node.

The advantage of storing the replied query hits for the same query in a vector is to have an independent control for each entry. A vector entry can be removed from the cached data only when it expires (not updated for while). However, a CQT entry is removed only when the Hits_results vector becomes empty. Similar to the PONG caching technique, a separate CQT is maintained for each connection. The summation of the vectors' sizes for the same QID among available logical connections justifies the resource popularity. It is well known that Gnutella performs poorly when a rare query is requested. For this reason we allow nodes to keep the cached hits results of rare queries for a longer period of time than the popular queries. We also need to implement a refresh component in order to maintain the validity of the cached result. A periodic process should PING the query hits replying nodes and update their time stamp field in the Hits_result vector. This approach may hurt the overall performance of the system, since network traffic is increased. However, we dramatically reduce the network overhead when a rare query is requested. Here again, the use of an independent data structure for each connection provides good support for minimizing the amount of PING/PONG messages used during the refresh process. In the best case, this allows a node to answer a rare query without needing to forward it, and in the worst case, a rare query is never requested. In addition to the above described mechanism, we implement a pooling query hit search method in each GUS node members in order to select a set of hit results that are useful to the searcher node, and at the same time ensure a balanced selection. The selection criteria are based on the minimum speed requirement and the replied query hit node qualifications. This method uses pointers to remember the last previous query hit entries transferred to searcher nodes in the recent past while. These hit results should not be used for the next upcoming query request. This process controls the query network flow. Alternating between all possible query hits with same QID provides an appropriate

distribution of the cache's results and nodes' workload. To increase the number of query results, a speed regulator can be used to collect more query hit responses with many possible speeds for future use. Intermediate nodes should rewrite the minimum speed field of the original query message. To better understand our query caching protocol, we describe a simplified scenario: Initially, each leaf node transfers its resources to the GUS member nodes located in its neighboring table. When a peer *n* wants to send a new query, it randomly selects a GUS member node *m* from its neighboring resource node table. Receiving node *m* checks whether the query corresponds to a valid entry in the CQT, and if so, the node *m* replies from the cache. Otherwise *m* will not answer the query and rewrites the minimum speed field to a pre-assigned value, and forwards it to its GUS neighboring nodes. The minimum speed pre-assigned value can be determined dynamically based on the network technology being used for the whole system, or it can be initially fixed by the system. Intermediate GUS members with the same interest will cache the information in CQT before forwarding it. Node *m* caches the query results and analyzes them to select a set of appropriate query hits for its requesting leaf node *n*.

The zone download request action could be the most demanding request in our system, since it requires greater resource availability like network bandwidth and CPU processing time. Using a pointer in the pooling method to remember the last used node for a transfer may eliminate the possibility of a successive replied node access for the same query during a short period of time. Nodes can initiate the download without interrupting other nodes.

The simulation results provided in Chapter 7 show that our caching protocol outperform standard caching scheme when the numbers of users increases (so do the GUS member nodes).

Chapter 7

Simulation Results and Performance Evaluation

Overall performance of the mobile collaborative virtual environment application depends mainly on the network performance. This chapter presents performance measurements done on our implemented system. Network overhead and network delay were measured as well as the average packet lost percentage. We also discuss the actual mobile popular systems when integrated into collaborative virtual environment. The results attained in this chapter provide answer to the question about the best suitable ad-hoc routing protocol for our MCVE applications.

7.1. Simulation Environment

To evaluate the performance of our MCVE system, we implement MOG-CVE using NS-2 [80] simulator. The goal of the performance study is to exploit the merits of our MCVE system compared to other existing systems. In this work, our main concern is about collaborative virtual environment under MANET, thus there is a need to analyze the impact of the interaction of our system with many underlying ad-hoc routing protocols from network performance point of view.

Due to the limitations of NS2 [80] software in the evaluation of routing protocols, a workaround is required to be able to simulate MOG-CVE under well known ad-hoc routing protocols proposed in the literature, such as HSR [54], ZRP [44], DSR [55], DSDV [86], and AODV [50]. HSR is selected in order to study the impact of heterogeneous and hierarchical topology on the network overhead. DSDV is selected

to study the impact of periodic update message on the network overhead. AODV is selected as a routing protocol because AODV is one of the most popular on-demand ad-hoc routing protocols. DSR is chosen since it represents the most popular source routing protocol. And finally, ZRP is preferred because it combines two main different routing methods (reactive and proactive). Table 5 summarize the properties of ad-hoc routing protocol used during simulation.

	Route Computation	Network Structure	Network Path	Source Path
DSDV	Proactive	Flat	Single	No
AODV	Reactive	Flat	Multiple	No
DSR	Reactive	Flat	Multiple	Yes
ZRP	Proactive/ Reactive	Flat/ Hierarchical	Multiple	No/Yes
HSR	Proactive/ Reactive	Hierarchical	Single	No

Table 5: Ad-hoc Routing Protocols properties.

It is important to know, that the focus of performance study is not set on comparing a set of ad-hoc routing protocols against each other [11], but rather determining the MCVE network performance under each routing protocol and evaluate the routing effect on our system. The setting parameters for the implemented system are summarized in Table 6.

Description		Default Value
Periodic probe interval to ping the neighbor nodes		7 s
PING Time Out		10s
Periodic interval to refresh the QUERY and the PONG caching table		7s
Periodic interval to send an update message (VE action)		10s
Default Time To Live (TTL)		7
Minimum size of the PONG caching table (PT) that allows a node to answer a PING message.		5
Simulation area		1000*500
Node state connections	Full	8 = nb_{Max}
	Stable	5 = nb_{Min}
	Connecting	Between 1 than nb_{Min}
	Initial	0
Simulation duration		900s
Initial pre-assigned session time spend		100s
Packet size		1024 byte
Low Mobility	Pause time	10s
	Speed	0m/s-10m/s
Fast mobility	Pause time	1s
	Speed	11m/s-15m/s

TABLE 6: SIMULATION SETTING PARAMETERS

We allowed about 100 seconds for the NS-2 application to stabilize before starting collecting information. We considered two mobility models in our simulation.

- The first model uses The Random Waypoint Group Mobility (RWG) [80] to simulate the node mobility in the network. This model extends the classic random waypoint model by applying mobility to a subset of close-by nodes at a time. Such model is useful, where members are divided in different groups and they interact with each other. We create two main scenarios (slow and

fast) as shown in Table 6 to study the effect of the node mobility [17] in the network on the system performance. In this model we illustrate, two groups formed randomly. Each group has a leader that determines the group's motion behavior. All the nodes in the group remain within 250 meters from the leader.

- The second model uses the random trip walking with reflection [82] to simulate the avatar movement in the virtual environment. In this mobility model, at a trip transition instance, a node picks a direction, trip duration, and numeric speed. The node before hilts the boundary of the domain, it is reflected into the domain. The common random way point [80] mobility model is unrealistic because it is unlikely that the avatars spread themselves throughout an area like a building or a city. Also it is unrealistic to let the avatars pause only at the edge of the simulation domain. We generated the desired model using [82]. An example could be <6=0; &.$>5-?#@$A$>,+5%#0$ -*$,-?#'@$>5&BCB@$>5&BC/@$>#,?$!(5#@$>'D##?$5#&,@$>'D##?$?#.!&@$ >D&+'#$!(5#@$>D&+'#$!(5#$?#.!&@$E>!0&F#.$!(5#@$>!0&F#.$!(5#$?#.!&@GIn this model, we generated a 500*260 topology that represents the virtual environment size with 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 avatars. The proposed VE surface area is enough to cover a large commercial building in a city. Avatars in the VE are usually humans, and therefore it is important to consider their speeds carefully. On average the running speed of human is 8m/s [107] with a pause time equal to 10 seconds. The walking average speed is about 2m/s [107]. The resulted generated script file was used as an input to implement simple game scenario described in Section 7.3.

Another import issue that has a straight impact on the system performance is the periodic probe interval for heartbeat messages. Within a small period, the network overhead increased, while the number of dropped connection decreased. However, with larger period, the situation reversed. To look for an appropriate periodic intervals, we ran our system for 10 runs with network size equal 50 nodes, each of

duration 900 seconds, varying the periodic interval between 1 and 10 seconds. The most favorable periodic interval value corresponds to the lowest ratio between the total number of messages and the total number of dropped messages. As shown in Figure 14, a probe interval value equal to 7 seconds provides a lowest ratio. Other settings parameters were assigned as recommended in previous works [24], and [40].

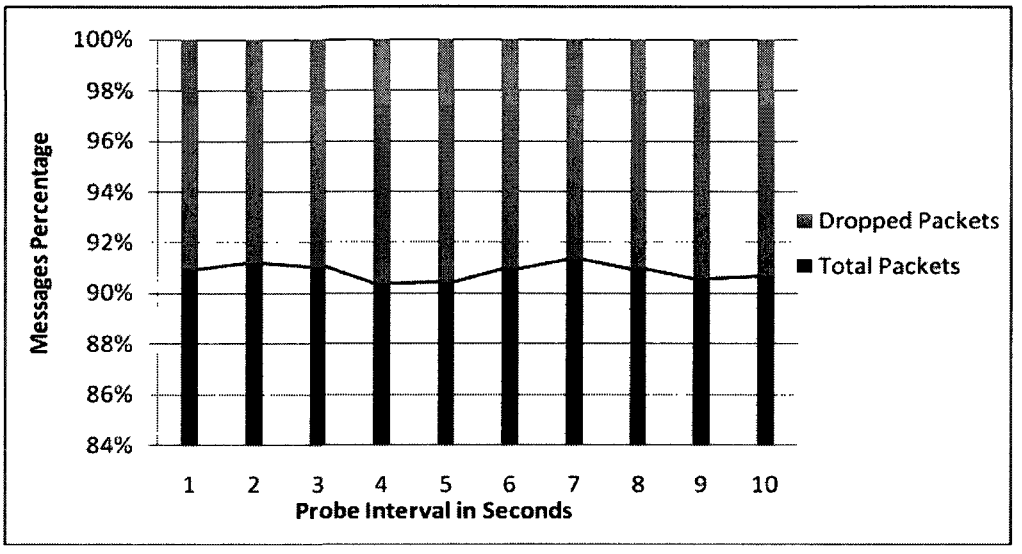

Figure 14: Routing Packets Overhead versus Dropped Packets with Various Probe Intervals

A rectangular simulation space (1000*500) was chosen to force a node to displace more distance than in square simulation space. In the simulation, we used various network sizes (10, 20, 30, 40, 50, 60, 70, and 100 nodes). In NS-2, each node was configured using the 802.11 MAC layer at 2 Mbps, with a 250-meter transmission range. The first observation was that the mobility pattern of nodes in MCVE influenced the route discovery, the route maintenance, and the caching mechanism; a low mobility model tends to dampen the path changes in the topology, nodes are most likely stable. Contrary to a high mobility model, nodes added entropy to the system and caused frequent path churn and packet losses. As mentioned above, the proposed work focused mainly on low mobility model, since the MCVE application is considered a low mobility application, to let user concentrate on their mission. However, simulation results with fast mobility scenarios were given so as to better

understand the behavior of the implemented system, and to validate the decision of the mobility model being used in this thesis.

7.2. Experimental Game Scenario

To put our system through a typical game situation, we integrate a tool called MCVE-AvSim into NS-2 environment. The proposed tool was implemented using C++ language to simulate the avatar's movement pattern in the virtual environment. As aforementioned earlier, we generate a script movement file using the random walking with reflection [82] for each network size. Then, we parse the file using the *Perl* language in order to create a separate movement file for each node participating in the VE. The file format consists of:

```
$ns_ at TIME "$node_(NODE) set X_ X1"
$ns_ at TIME "$node_(NODE) set Y_ Y1"
$ns_ at TIME "$node_(NODE) set Z_ Z1"
$ns_ at TIME "$node_(NODE) set Zone_ ZoneID"
```

The first three lines are used to define the node position in the virtual environment, where the last line is used to define the zone ID for the specific position (`X1`, `Y1`, `Z1`). In the simulation, we assumed that all avatars have the same Area of Interest (AoI) diameter, equal to $t\sqrt{3}$, where t represents the length side of the regular hexagon zone shape. Therefore, we allowed the user to subscribe to four zones at most. The virtual environment flat space was divided into ten zones, with t equal to 100 meters, as shown in Figure 15. A single object is placed randomly in each zone. When an avatar joins a zone it updates this object, then sends a message with a time stamp to all its neighbors. To add more realism to the random walking reflection model, we selected zone 2 to be our mission goal, thus all nodes must move toward this zone. The total simulation time of 900 seconds is enough to let the most distanced avatars to reach the goal; each node manipulates its mobility file independently. A load position procedure is called periodically to update the avatar position in the VE based on the movement file. When an avatar changes its set of zones in the virtual environment, it

broadcasts a message in order to request the new data zone; any node serving the requested zone sends a reply message.

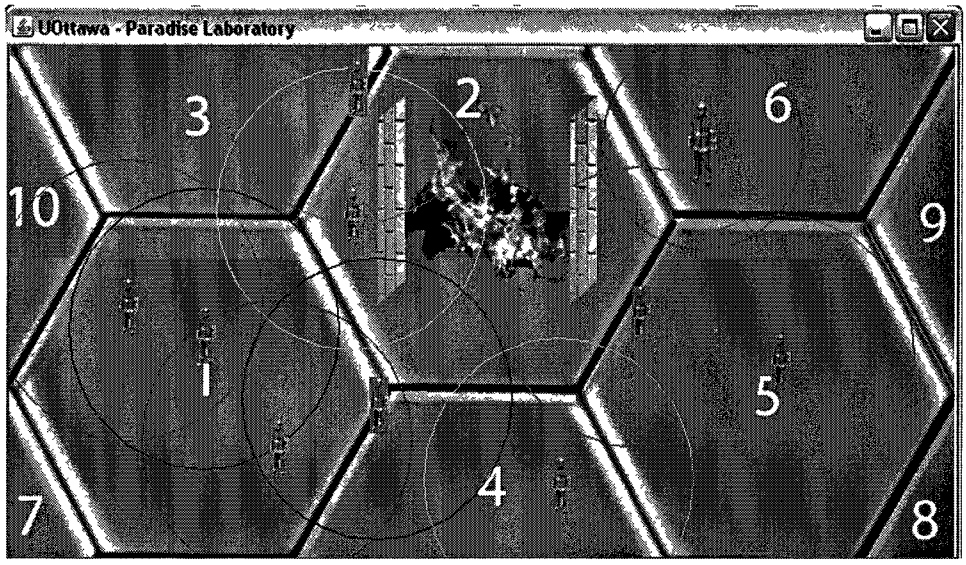

Figure 15: Example of the Zones Representation in Virtual Environment

An example of initial avatars physical positions in the virtual environment are shown in Figure 16 using a random walking with reflection generator [82] with 50 nodes.

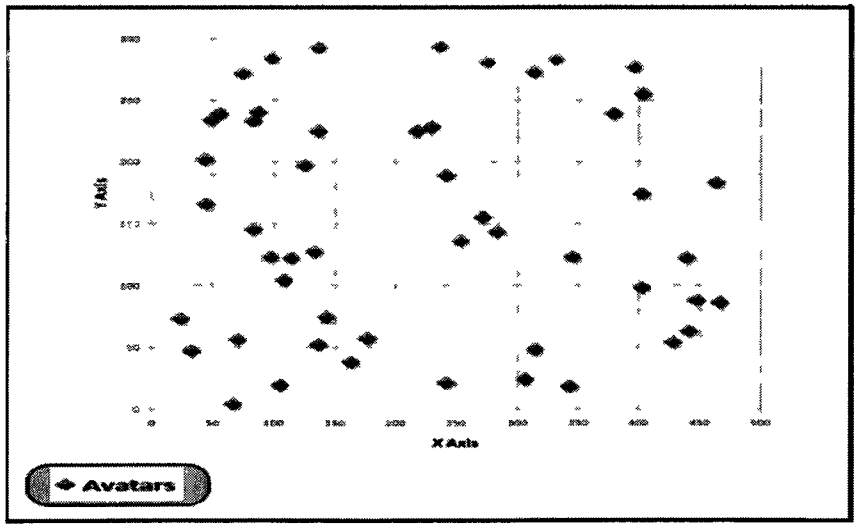

Figure 16: Example of Avatars Distribution using Random Walk with Reflection generator.

Initially, in the simulation scenario, the session manager node holds the whole virtual environment, while other nodes handle only a copy of data zones depending on their initial physical position in the virtual environment (VE) and their Area of Interest (AoI). Nodes with lowest ID in the surrounding area were selected as zone owner node. After a node joins the world during a play when an avatar starts moving in the VE following the movement file, the user must leave its current zones and request new zones according their new physical position and AoI in the VE. For this, a broadcast process must be triggered to look for the requested zone data in the network.

In this simulation, we consider two session IDs. Each node decides about its session based on its node ID (odd or even). Nodes with the lowest ID take the role of session manager. We assumed that only 25 % of the total participating nodes have the hardware capability to be Ultrapeer, and consequently can participate in the Gnutella Ultrapeer System (GUS).

7.3. Performance Evaluation

The performance analyses are based mainly on three primary metrics: network overhead, network latency, and packet delivery ratio. We believe that such metrics are a must for determining the performance quality of any MCVE application. We investigated the impact of the underlying ad-hoc routing protocols, pause time, and network size on the aforementioned metrics. Network overhead and latency, and packet delivery ratio are retrieved directly from the NS-2 trace file, thanks to the "*Perl*" language. The packet lost ratio is calculated as the total number of dropped packets divided by the total number of packets sent. Network latency is the average delay for data transfer from a sender node to a receiver node. The average network overhead is measured in Kb/s and defined as (the total number of packets surfing in the network*packet size)/simulation time.

In order to observe the improved performance of our system, we implemented three prior existing protocols, namely MPP [40], mobile plain Gnutella [38], and Enhanced Gnutella for Ad-Hoc Networks (EGAN) [21]. We also implemented Lowest-ID algorithm [36] under mobile Gnutella to illustrate the improvement reported in the basic GUS. The choice of the Lowest-ID algorithm stems from its popularity to face the cluster head election problem in mobile ad-hoc networks. This research direction is worthwhile because only a single routing protocol is used in the simulation scenarios in other works. The simulation results revealed that many of the conclusions drawn in previous protocol comparison studies are no longer true under the conditions brought about by the traffic generated by the overlay network with a limited number of logical connections per node. Barbosa *et al* [2] make claim to an interesting result: each routing protocol performs well in some scenarios, under some metrics, but displays a weakness in others, while Park *et al* [22] declares that Gnutella performs better when both a proactive ad-hoc routing protocol and hierarchical node classifications are used. In summary, and with respect to the simulation scenarios, none of the previous works share the same opinion on the performance of routing protocols when Gnutella network operates over MANETs.

7.3.1. Simulation Results for our System

In the following sections, we study the performance of our system alone based on the network overhead, network latency, and packet delivery ratio metrics. Probe messages are routed using the proactive routing ad-hoc protocol DSDV [86].

First let us determine the most appropriate values for both weight values ; 8 and ! "# used to define the utility $ values in Chapter 4, in order to negotiate the logical connection performance during a discovery procedure.

$$P_{on} = g_n * (w_1 * Mp_{on} + w_2 * 1/Nhops_{on})$$

$$g_n = \begin{cases} 1 & If\ node\ n\ is\ in\ full\ or\ stable\ state \\ 0 & Otherwise \end{cases}$$

We ran the simulations with {[w_1=1, w_2= 0], [w_1=0.9, w_2= 0.1]... [w_1=0, w_2= 1]} different weight patterns, and we set the simulation with 50 wireless nodes in a low mobility scenario. The appropriate weight pattern represents the lowest network overhead generated by the system. We considered PING and PONG, messages in the network overhead. Figure 17 shows the variation of the network overhead when we varied the weight factors under the AODV, DSR, DSDV, HSR, and ZRP routing protocols. For the ZRP protocol, we used the value 2 hops as radius since it provides a better network performance, as shown in Figure 18.

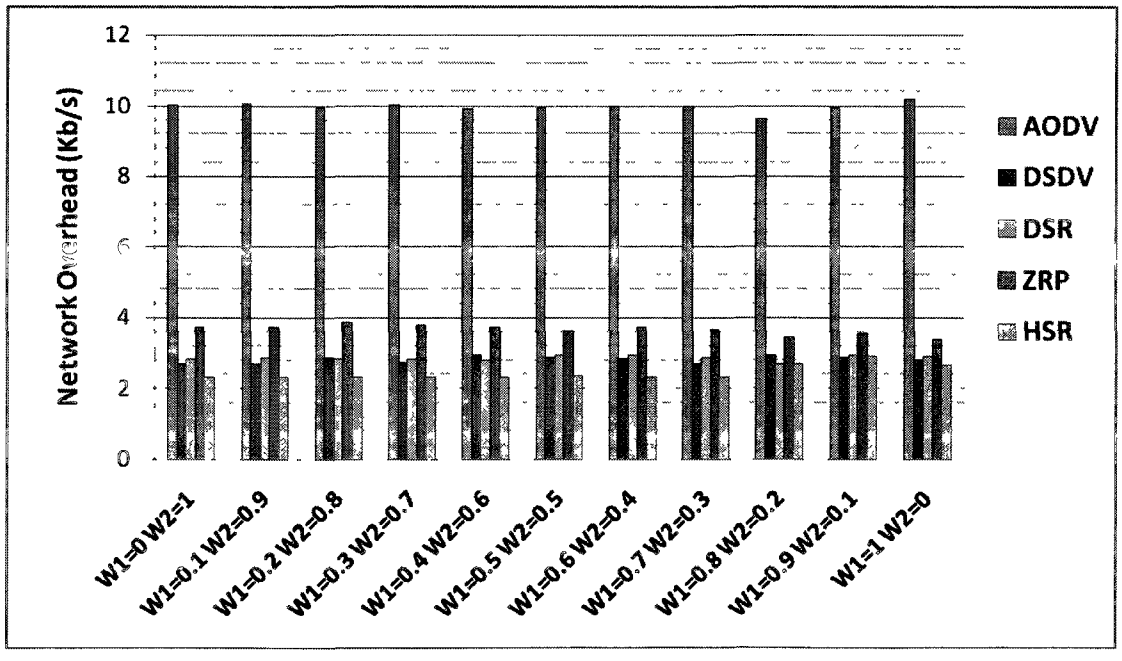

Figure 17: Network Overhead to Optimize the Weight Factors Using 50 Nodes

As can be seen in Figure 17, no matter what ad-hoc routing protocol had been used, the average network overhead of our system was minimal when $w_1 = 0.7$ and $w_2 = 0.3$. The most suitable weight pattern is [$w_1 = 0.7$, $w_2 = 0.3$], the selection policy then favored connections with neighbor nodes having a higher CVE-profile matching ratio degree. Figure 18 justifies our choice of ZRP radius 2. We run our approach under ZRP with several radius -r- values (r =1, r =2, r =3, and r =4).

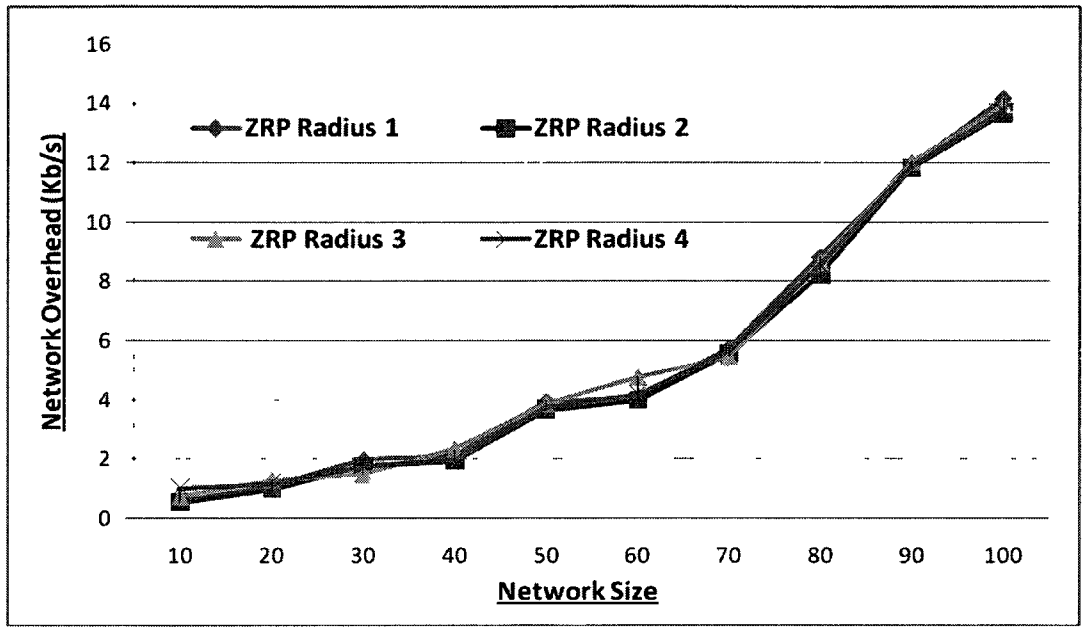

Figure 18: Impact of the ZRP radius on the network overhead

Clearly, the ZRP performance is related to the radius value. The ZRP protcol requires restricted overhead in the zone due to the frequent messages needed in order to update the routing table. As seen in Figure 18, the optimal radius value was 2 when the network performance had been raised by 10%. From this point on, in the rest of the simulation scenarios, we will use $w_1 = 0.7$, $w2 = 0.3$, and radius 2 when the ZRP ad-hoc routing protcol is used.

A) Network Overhead

Network overhead measurement was done to find out how MOG-CVE protocol handles in the searches and content transfers, when user moves forward in the VE. Figure 19 reports the network overhead as a function of network size. Our system was badly affected when implemented under AODV. The network overhead of AODV increases faster than all other ad-hoc routing protocol, because of the higher Gnutella agent routing overhead. When the network traffic is overloaded AODV doesn't scale.

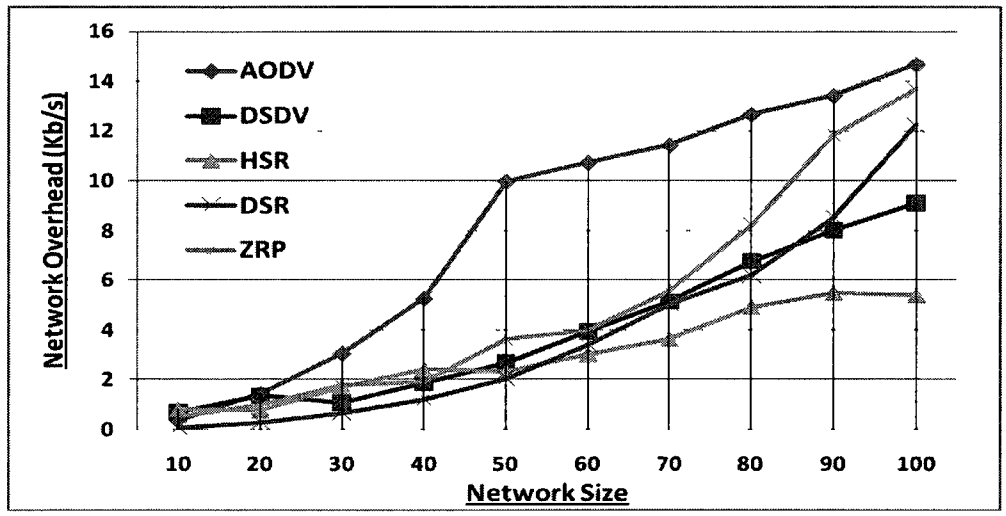

Figure 19: Total Network Overhead in Low Mobility Model

In a small-sized network (20 nodes), no matter what ad-hoc routing protocol was used, our system exhibited almost the same overhead. This is an expected result because the number of available peers was lower than the lower-bound connections (nb_{Min}), and thus the nodes continued to discover indefinitely without reaching a stable state. In addition, due to the usability of link selection policy, some nodes may have rejected the connection request, and therefore resorted to looking for new neighbor nodes. ZRP uses Interzone Routing Protocol (IERP) packets [44], for searching a route; however, these packets may cause a network congestion resulting in the decrease of network performance. Regarding the network overhead metric, HSR appear to be more appropriate for our MCVE application when the network

grow. In hierarchical routing the network in organized into cluster witch help to maintain a relatively stable network. Only communication between session manger nodes and zone owner node will be propagating across long distance. Thus, the traffic control is largely reduced. The continued changes of the hierarchical cluster are covered by the use of logical partition to group user participating in the same session ID and zone ID into separate subnets. We can see that, flat routing is simple and efficient for small network. However, when a network becomes large, the volume of routing messages increase and it will take a large time to arrive at the remote node.

B) Mobility Effect

A node's speed determines how quickly its position changes, which in turn sets the rate of network change. The high mobility model yields the highest network overhead and a corresponding drop in the delivery ratio. The reason for this is that in low mobility model, the network is more stable than in high mobility model, where nodes are forced to move more often. To validate this hypothesis, we conducted an experiment with various network sizes, as shown in Figure 20. In contrast to our earlier experiment (whose results are shown in Figure 19), an increased number of messages were generated since the network topology changes fast, which may have caused extra overhead to re-establish network paths.

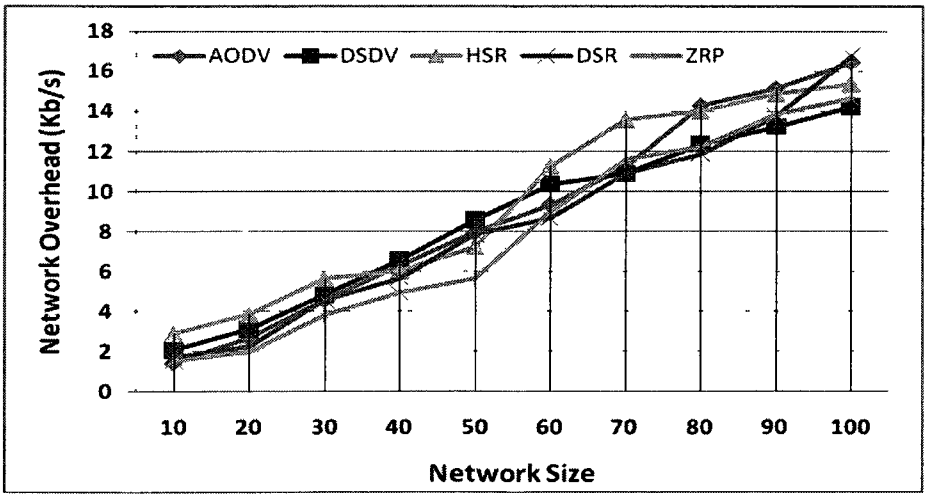

Figure 20: Average Network Overhead in Fast Scenario

Increased mobility leads to weakened network connectivity. ZRP and DSDV appear to be less affected by mobility. For HSR, due to the cost of continuous updates in the physical partition, the network overhead is increased by about 20% compared to the low mobility scenarios in Figure 19. We concentrate our efforts on the low mobility environments since CVE application is considered a low mobility application. However, an investigation into a best ad-hoc routing protocol for fast scenarios is a subject of ongoing and future work. In the rest of the simulation results presented below, we used the low mobility model.

C) Network Latency

Network latency represents the average time duration between the time a message is sent and that of its reception. Figure 21 shows the network latency produced by our approach under the DSDV, HSR, ZRP, DSR, and AODV routing protocols. The different network sizes are shown along the x-axis, while the average network latency, measured in ms, is shown on the y-axis.

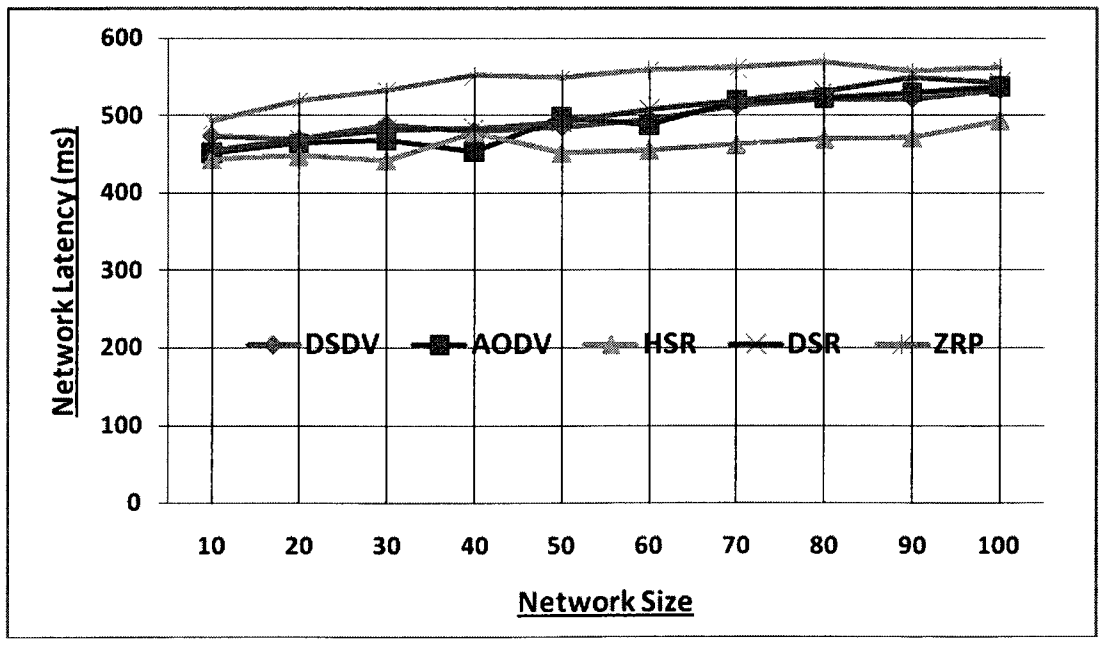

Figure 21: Network Latency in Low Mobility Model

As it can be seen from the results, ZRP was badly affected; the reason for this increased delay is that since ZRP implemented two protocols Routing Protocol (IARP) [44] and Interzone Routing Protocol (IERP) [44], packets were held in the buffer before being transmitted when inter-zone communication was required. The result shows also, in AODV the network delay is increased at the beginning of the transition caused by the route discovery overhead. Figure 21 demonstrate that, HSR exhibited the most network latency, followed by DSDV, and then DSR. This is due to the fact that HSR uses the hierarchical routing. Only communications between zone nodes manager and session nodes manager are propagated across long path. Of course, the drawbacks of HSR compared to flat routing are: the need to maintain the hierarchy and the cost to continuously update the cluster hierarchy. Fortunately, the classification of our system under low mobility model came to help.

Network latency is an improtant issue to determine the MCVE application quality; update messages should be delivred to all users within an acceptable delay to maintain a sense of immersion. The network latency requirement for any CVE application should be less than 400 ms [108] to provide a sense of immersion for participating users. Such requirement was initially stated for CVE in wired network. We believe this requirement should be relaxed to cover CVE in mobile environment. None of the created scenarios can reach the desired latency. Overall, in our simulation the best what we can get is about 453 ms. As a future work we may improve this by integrating a dead reckoning mechanism in the Gnutella Client.

D) Packet Lost Ratio

The packet lost ratio represents the ratio of the total messages dropped over the total number of messages surfing the network. Figure 22 shows the variation in the packet delivery ratio when we increased the network size under the AODV, ZRP, DSDV, HSR, and ZRP ad-hoc routing protocols. Usually, the offered ratio is increased by increasing the network overhead.

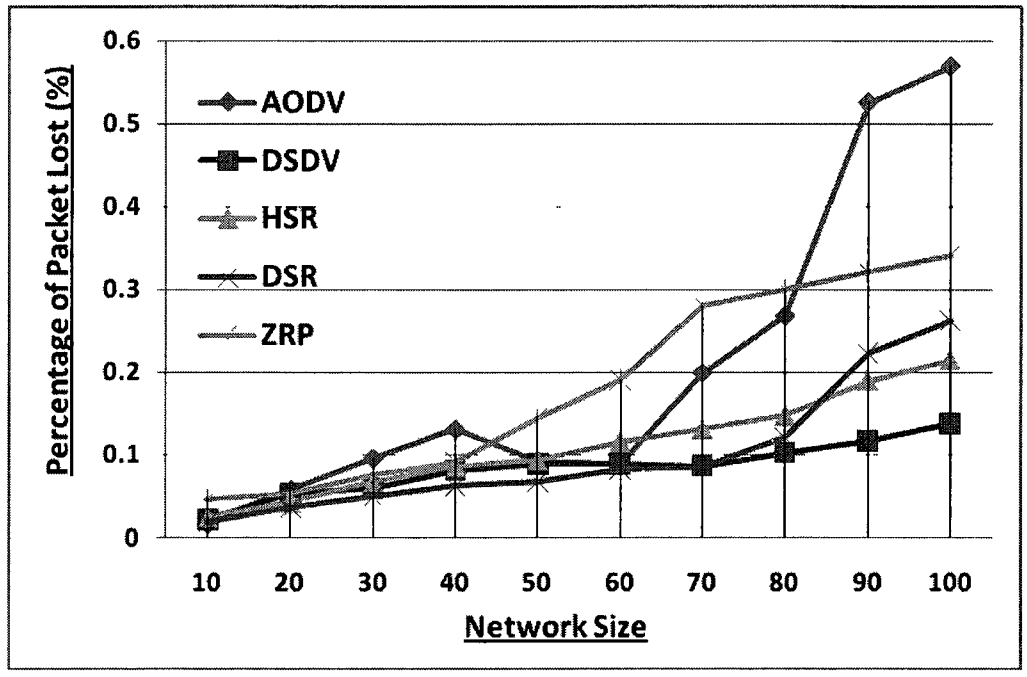

Figure 22: Packet Lost Ratio in Low Mobility Model

The packet lost percentage (across all ad-hoc routing protocol) is low when network size increases is simply due to the fact that when the network is congested packets are more likely to be dropped. HSR and DSDV performed better regarding this metric with a slight improvement in DSDV. From Figure 22, AODV had the worst performance regarding this metric followed by ZRP. ZRP used a caching scheme, and thus many routes became invalid, which may explain the large amount of dropped packets. With DSR we observe a promotional ratio when the network size is relatively small (70 nodes). As the route length increases, many paths become stale due to the aggressive caching. The packet delivery ratio of the HSR and DSDV routing protocols were close to 100%. In agreement with the network overhead results showed in Figure 19, when the network size increased, the packet delivery ratio decreased.

E) Node Connectivity Degree

The integration of node states scheme during the discovery process lead us to study the variation in the node connectivity degree behavior during the simulation. Node stabilization is related to both network size, and the number of zones in the VE. In Figure 23, 24, and 25, we set the network size respectively to 10, 50, and 100 wireless nodes. The different simulation times are shown along the x-axis, while the average node connectivity degree is shown on the y-axis.

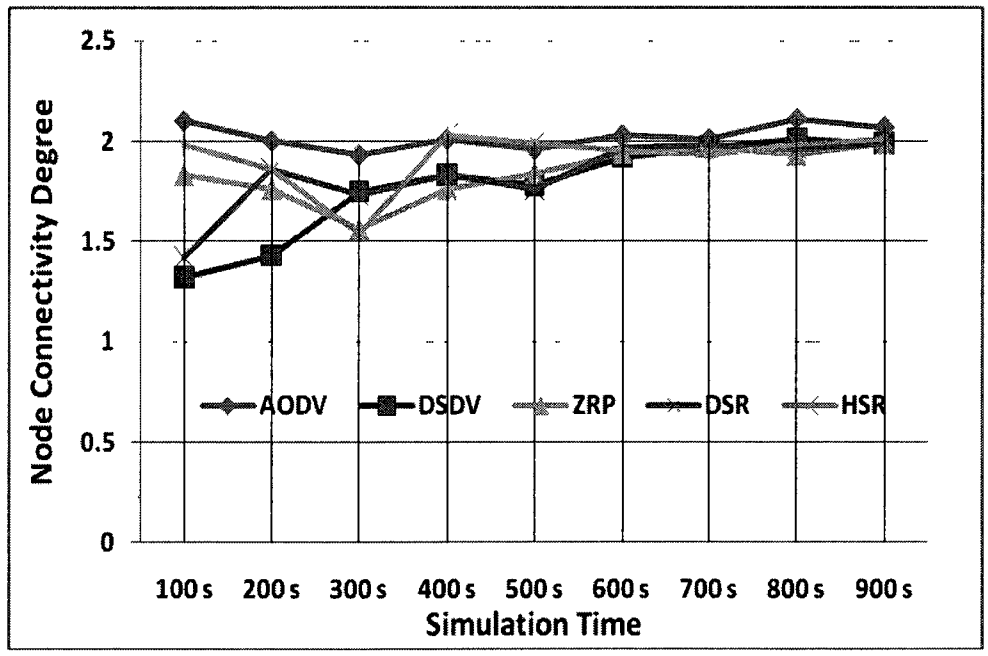

Figure 23: Connectivity Degree when the Network Size is Equal to 10

In agreement with the previous network overhead results, small scenarios provided an expected high network overhead because the number of available nodes was too small to fill up all the slots -8- in the nodes' resource neighboring table. Nodes remained in the connecting or initial state and this generated extra network overhead. In addition to the permanent discovery problem, low connectivity may have had an impact on the network partition, since in small networks; many nodes can find themselves isolated.

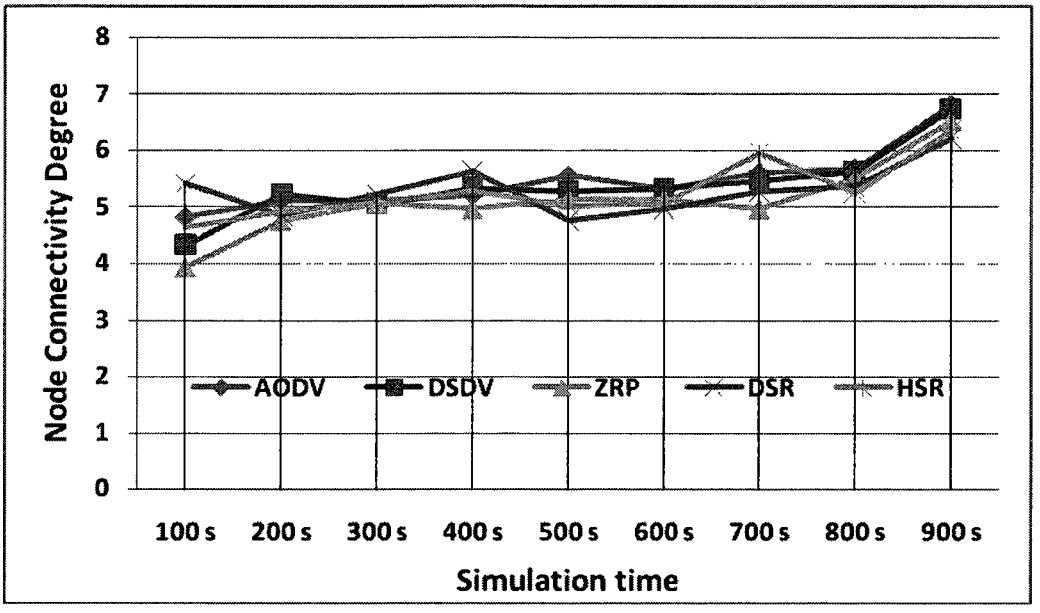

Figure 24: Connectivity Degree when the Network Size is Equal to 50

When the network size increased, the connectivity degree also increased; this is mainly caused by the increased number of available connections. As shown in Figure 24, after 300 seconds, the network became stable, since most of nodes reach their stable state.

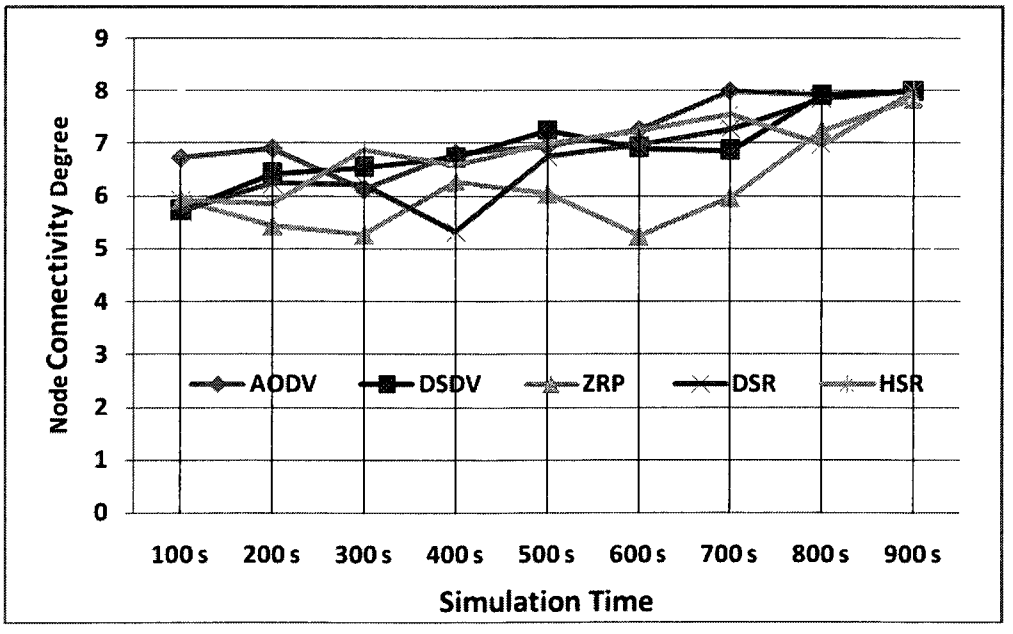

Figure 25: Connectivity Degree when the Network Size is Equal to 100

In a large network size (Figure 25), the network stabilized after around 300 seconds (across all ad-hoc routing protocol). Clearly, when we increased the network size, the simulation time needed for network stabilization decreased. HSR has acceptable degree connectivity about 7.32 when the network size is increased and the simulation time is greater than 500 seconds.

7.3.2. Comparison with other Existing peer-to-peer Protocols

In this section, we implemented three other different Mobile Gnutella models, namely mobile peer-to-peer MPP [40], mobile plain Gnutella [38], and an Enhanced Gnutella for Ad-Hoc Networks (EGAN) [21], in order to compare them to our system. Mobile peer-to-peer [40] is selected because it is one of the most popular mobile applications in the literature. The choice of the EGAN scheme stems particularly from the desire to face the inefficient connection problem in the overlay network, while mobile plain Gnutella was selected to illustrate the enhancement reported by our system and also to prove that a straightforward implementation is not satisfactory in MANETs. In order to ensure a fair comparison with our system, we tune Gnutella V0.6 [38] to support the four node states (*Initial*, *Connecting*, *Stable*, and *Full*). Since CVE is a network delay and bandwidth sensitive application, we selected network overhead, network latency, and packet delivery ratio as the main metrics for our comparison. In accordance with [40], [24], and [86], we set the ad-hoc routing protocol simulation for the implemented systems as follow: EDSR for MPP, and AODV for EGAN and mobile plain Gnutella. All of the experiments results were based on a slow mobility.

A) Network Overhead

Figure 27, reports an evaluation of network overhead a function of network size for each implemented systems. The simulation results show the impact of the caching mechanism and the connection selection scheme on the performance behavior of our system MOG-CVE.

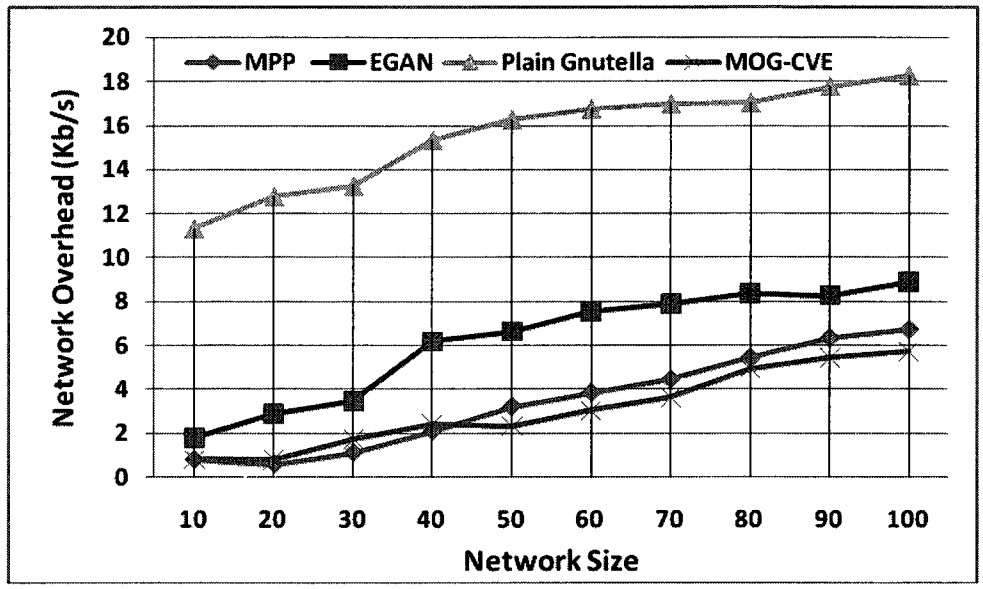

Figure 26: Network Overhead Systems Comparison

Clearly plain Gnutella is the worst affected. This confirms the claim that a straightforward implementation of Gnutella in mobile environment is not satisfactory. Figure 26 shows, the average number of messages in our approach are about 5% less compared to MPP. However, when compared to EGAN, our approach diminishes the routing overhead by about 20 %. This is, caused by the positive impact of the Gnutella Ultrapeer System and the link selection policy. The use of extra Ultrapeer advertisement -UADV- messages to inform other neighbor nodes in EGAN may increase the network overhead. MPP show that for practicably sized network (40 nodes), with limited mobility, the network overhead is quite limited. However, in particular situation, when the network traffic increases too much, scalable MCVE applications are not well supported by MPP. Update messages and search requests will not scale to both growing network and increasing request rate. This is as expected result since MPP was originally developed for mobile file sharing application, thus, all messages are routed in a similar way. We observe that when using HSR, the effect of scalability become less significant, since the routing process follows a hierarchical topology. The proposed network topology formation algorithm

performs better than the MPP algorithm when employed in MCVE applications. In addition, as the number of queries increases, the performance of the MOG-CVE algorithm decreases due the caching algorithm that comes to help.

B) Network Latency

In this section, we study the impact of the network latency performance on the implemented systems. The main factor that influences the performance of this metric is the flooding problem. Figure 27 shows the network latency of the implemented approaches.

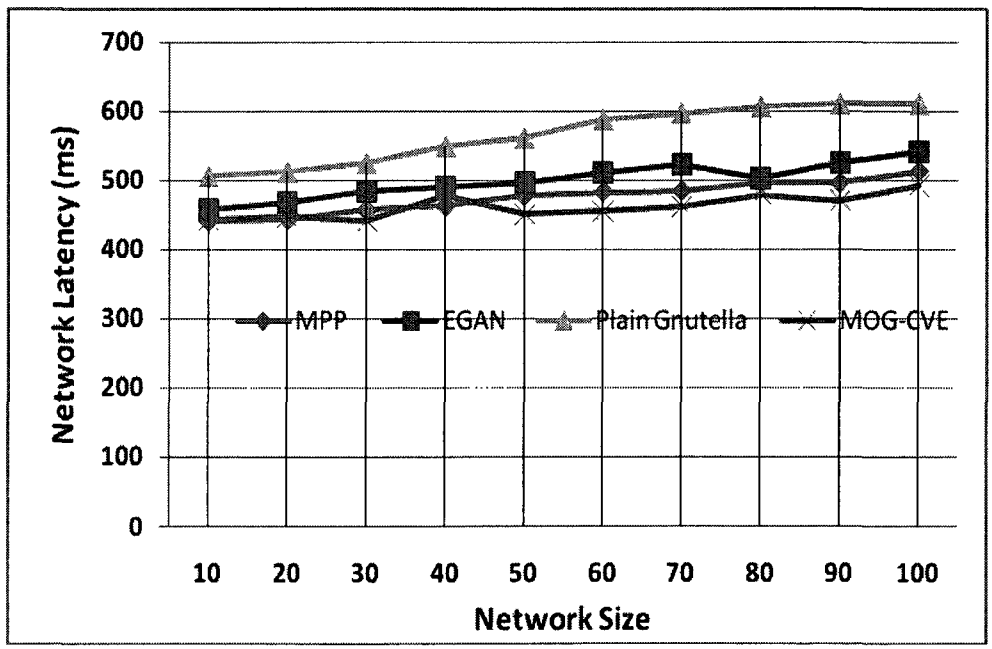

Figure 27: Network Latency Comparison

Concerning this metric, our protocol provided a slight improvement over MPP especially when the network size is small. This improvement is due to the integration of session ID for queries filtering in the network layer. Network latency in plain Gnutella is very high; this is caused by the incremented network overhead generated by Gnutella when implemented in mobile environment. Comparing our system to plain Gnutella, part of the considerable decrease in the network latency metric was

due to the caching protocol implemented in each GUS members. EGAN consider some processing delay during the supper node selection. One interesting observation is that the de network latency agrees with the network overhead shown in Figure 26.

C) Delivery packet ratio

Concerning the packet delivery ratio metric, the reason for a low delivery ratio was usually due to network congestion and higher network overhead. Figure 28 report the packet delivery ratio as a function of network size in various systems. Overall, simulation results indicate that the packet delivery ratio was optimal when the network size had been relatively small.

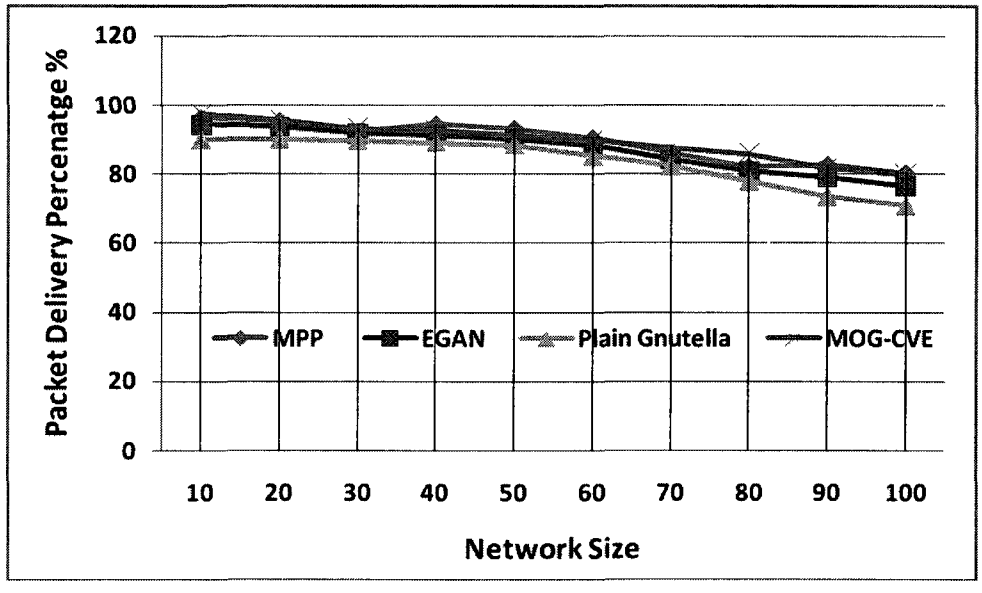

Figure 28: Packet Delivery Ratio Comparison for the implementer Systems

Our system MOG-CVE and MPP are the systems that produce the highest packet delivery ratio. The plain Gnutella network was badly affected. This is concurrent with the previously described network overhead results shown in Figure 26. In EGAN, due to node mobility, the cached information during the bootstrapping process may become outdated, and, thus, many packets are dropped. The shape of the curve in Figure 28 is similar to the network overhead results provided in Figure 26.

D) Impact of the GUS on the System Performance

In this section we analyses the performance of the Gnutella Ultrapeer System. The performance analyses was based on the system's re-configuration number, which reflects how the frequency of leaf nodes switching their GUS member. Since every switch required extra network overhead, therefore it was to important study the impact of re-configuration number on our implemented system and compared with Gnutella V0.6 [38] and Lowest ID algorithm [36]. Figures 29 shows a variation of the re-Configurations number when we increased the network size.

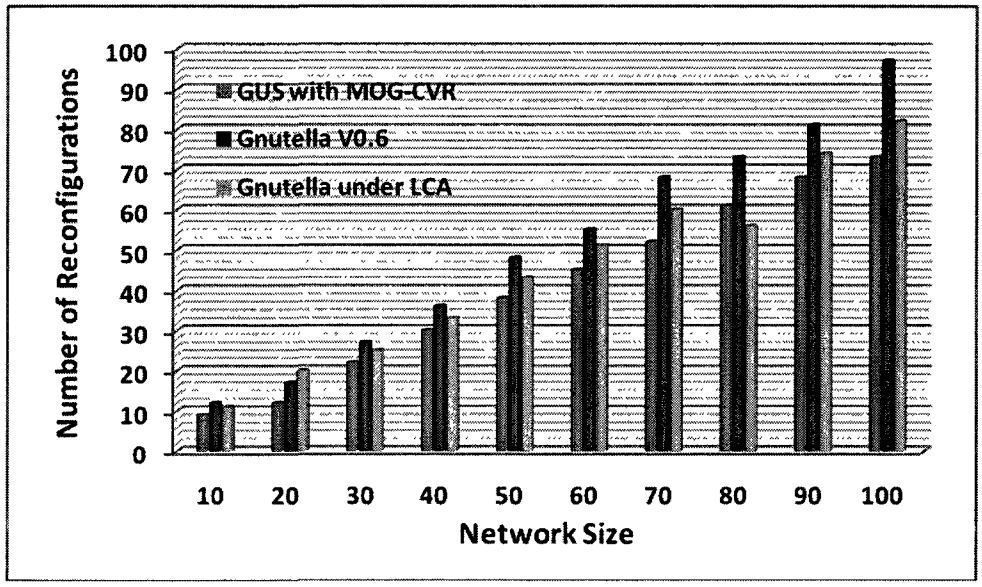

Figure 29: Average Number of Switches under AODV

Compared to Gnutella V0.6 the average network overhead was higher than that for Gnutella LCA. This is an expected result since, in Gnutella V0.6 scheme, when the network availability decreased, it may have failed to be reselected as an Ultrapeer, while the lowest-ID node continues to offer services as Ultrapeer. For this reason, Gnutella LCA network overhead was lower than that of Gnutella V0.6. Our GUS uses the dynamic selection policy to reduce the number of re-configuration. In accordance with the incremented network overhead generated by the implemented protocols shown in Figures 29, when the network was small (around 20 nodes), no matter what

ad-hoc network protocol had been used, and all protocols had a similar behavior, since the number of nodes with Ultrapeer capability was very low. Leaf nodes continually try to change their mandatory GUS members without any results.

Chapter 8

Conclusions and Future Work

8.1. Conclusion

The Mobile Collaborative Virtual Environment (MCVE) application gazes at potential future applications. Its infrastructure-less network, self-administration, and applicability for collaborative aspects render it a prime candidate as a driving force for the advancement of CVE applications. Early proposals focused their attention on the information sharing application [21], [24], and [67], or collaboration work [45], and [62]; and to the best of our knowledge, there are no research efforts that have considered the CVE and mobile environment together as a whole system. This thesis has focused on the communication and networking strategies used to support the CVE application under MANETs, with the goal of achieving better efficiency and performance.

Our investigation began by looking at the problem of building overlays. An initial evaluation of current platforms for structured and unstructured peer-to-peer computing suggested that an unstructured network will be the most suitable topology for MCVEs; such a network is designed to tolerate dynamics such as frequent topological re-configurations, and is robust against abrupt node failure. The network also addresses the question of the avatars' Area of Interest (AoI), prompting us to argue that the best performance and efficiency of the MCVE application may be achieved by having the Gnutella overlay management protocol under HSR as a ad-hoc routing protocol. Usually, each node in a MANET has to participate cooperatively in the routing and forwarding process, and peer discovery cannot be conducted in an appropriate manner at the application layer. These considerations led us to consider the cross-layer as a vital technique for achieving efficiency and

performance. The work presented in this thesis is based on hybrid network architecture, and the distinctive feature of our work lies in the innovative approach of extending the MANET routing beyond its standard uses by integrating the session ID routing context. Previous works [21], [24], and [38] have promoted cross-layer architecture, however due to the high dynamic of virtual environment; this strict layered design has shown not flexible enough, preventing from performance optimization in MCVE applications.

The first protocol addresses the overlay network issue; its goal is not only to build an overlay network, but also to enable logical connection optimization. The Gnutella Ultrapeer System (GUS) is integrated to offer flexibility and scalability to the MCVE in a MANET environment. Significant improvements have found their way into the overlay by introducing a novel caching mechanism, which can be considered a key feature for overcoming the flooding problem in mobile environments (since we implement the caching protocol at each GUS member node). Although this approach delivers good performance however, it was difficult to realize when the size of the GUS is relatively small.

To complete our perspective on MCVE, we provided a simple mechanism integrated with a replica protocol for the Mobile Gnutella-based network in order to support the Collaborative Virtual Environment. Network heterogeneity is also guaranteed through Gnutella Ultrapeer System (GUS) member nodes dispersed throughout the network, as well as the session manager node. A VE data classification is provided in order to target the necessary data that should persist, not all of the description files need to be retained. A replica algorithm was designed between the coordinator node and its replica to ensure failure-less, while maintaining the basic functions of the Gnutella network. Unlike other techniques, the replica algorithm is not directly affected by newly joined nodes, as is the case in other structured networks. The overall system was shown to exhibit more reliability and efficiency because the replica node was selected based on a selection policy that isolated network behaviors.

The proposed mobility models play a critical role in the accurate simulation of collaborative virtual environment performance in MANET. We have evaluated the sensitivity of mobility details on MCVE in a game context, where we proposed a mobility model to maneuver the motion in the network, in popular wireless simulator NS-2. This software allows to model a wireless network using a specific script and by assigning conditional probabilities. One problem is that implementing such system is time consuming and prone to errors. We use a predefined real game scenario to avoid these implementation problems. We implemented an avatars tool using C++ as an independent system in NS-2 that generated avatars movement. The range of performance variation that was displayed across both models highlights ours point regarding the importance of the mobility model being used in the MCVE application. The results indicate that low mobility model yields lowest network overhead and maximum network latency, while high mobility model causes weakened connectivity in the network.

Since our focus is CVE applications under MANET, thus, we implemented our MCVE under various ad-hoc routing protocols in order to understand their impact. We noticed an interesting result; according to the routing strategy in MCVE, HSR outperformed AODV, DSDV, DSR and ZRP in all of the proposed metrics. From our point of view the reason is that some HSR criteria make it eligible for our MOG-CVE. The criteria are:

- The routing protocol architecture follows a multilevel hierarchical design with a zoning repartition.
- Logical partition is based on a logical relation that reflects the user's interest in the virtual environment. Each node knows only the node connectivity within its zone and the zone connectivity of the whole network.
- Routing is established based on the session ID, zone ID and node ID of the destination.

The use of multilevel hierarchical routing protocol was discussed in [20], and [54]. With the hierarchical approach, it is easy to see that many of the scaling problems disappear.

The purpose of our designed system is to cope with unstructured peer-to-peer computation in order to deliver efficient communication for MCVE applications, as well as to provide continuous service with minimum data error recovery when a node disconnects. Another important quality factor of our system is 'the no need' for the creation of new messages, since no extra control messages are required to manage the virtual environment. PING and PONG messages are extended to be used for this convenience. Obtained simulation results has shown that our MCVE proposed system reduce significantly the network overhead by 5% and 20% respectively when compared to MPP [40] and EGAN [21].

8.2. Future Work

A number of open problems must be solved to allow the development of an efficient MCVE system. These problems suggest a variety of research directions that need to be pursued to make such a system feasible.

One such direction would be to investigate the integration of dead reckoning technique into our system in order to improve the system performance. It would also be very valuable to compare the network overhead and the network latency for the system with and without dead reckoning technique.

The current system requires that the user should register only to one session. It would be preferable that an extended version of the cross-layer approach must be suggested so as to best fit the collaboration context in many sessions.

Finally in term of battery energy saving, in this thesis we are unaware of the studies of overlay network on energy constraints devices. A future work must examine how to adapt our MOG-CVE to efficient manage the node battery resource.

Bibliography

[1] K. Aberer, P. Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, and R. Schmidt. "*Advanced p2p Networking: the P-Grid System and its application*". PIK Journal - Praxis der Informationsverarbeitung und Kommunikation, vol 26, num 3, Special Issue on peer-to-peer Systems, 2003, pp: 86-89.

[2] L . Barbosa, I. Guimaraes, and A. Ferreira. "*Evaluation of ad-hoc routing protocols under a peer-to-peer application*". Proceedings Wireless Communications and Networking WCNC, USA 2003, pp: 1143-1148.

[3] A. Bejan, and R. Lawrence. "*Peer-to-peer cooperative driving*". Symposium 7th International on Computer andInformation Sciences, USA 2002, pp: 259-264.

[4] S. Benford, and L. FahlŽn. "*A Spatial Model of Interaction in Large Virtual Environments*". Proceedings of the 3rd conference on European Conference on Computer-Supported Cooperative Work, Italy 1993, pp: 109-124.

[5] M. Bisignano, G. Di Modica, and O. Tomarchio. "*A JXTA compliant framework for mobile handheld devices in ad-hoc networks*". Proceedings of the 10th IEEE Symposium on Computers and Communications, Spain 2005, pp: 582-587.

[6] N. Blundell, and L. Mathy. "*An Overview of Gnutella Optimisation Techniques*". Proceedings of PGNetwork ,UK 2002, pp: 235-243.

[7] A. Boukerche, R. Araujo, and M. Laffranchi. "*Multiuser 3D virtual simulation environments support in the Gnutella peer-to-peer network*". Journal of Parallel and Distributed Computing archive, Vol 65, Issue 11, 2005, pp: 1462-1469.

[8] A. Boukerche, J. Feng, R. Werner. "*A buffer management technique for 3D image-based rendering on mobile devices*". Proceedings of the 4th ACM international workshop on Mobility management and wireless access table of contents. USA 2006, pp: 158-163.

[9] A. Boukerche, and R.Pazzi. "*A Peer-to-Peer Approach for Remote Rendering and Image Streaming in Walkthrough Applications*". Proceedings ICC IEEE International Conference on Comunications, Vol 24, Issue 28. USA 2007, pp: 1692-1697.

[10] A. Boukerche, A. Roy , and N. Thomas. "*Dynamic Grid-Based Multicast Group Assignment in Data Distributed Management*". Proceedings 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications, USA 2000, pp: 47-54.

[11] A. Boukerche "*A Performance Comparison of Routing Protocols for Ad Hoc*". Proceedings of the 15th International Parallel Distributed Symposium, USA 2001, pp: 23-27.

[12] A. Boukerche, A. Zarrad, R. de Araujo. " *A Performance Evaluation of an Optimized Caching Protocol for the Mobile Gnutella based Network to Support Distributed Collaborative Virtual Environments*". Proceedings Local Computer Networks, Spain 2007, pp: 207-209.

[13] A. Boukerche, A. Zarrad, and R. Araujo. "*A Smart Gnutella Overlay Formation for Collaborative Virtual Environments over Mobile Ad Hoc Networks*". Proceedings of the 10th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Spain 2006, pp: 143-156.

[14] A. Boukerche, A. Zarrad, and R. Araujo. "*Optimized PONG Caching Technique for Mobile Gnutella Network to Support Large-Scale Collaborative Virtual Environment*". Proceeding Global Telecommunications Conference CLOBECOM, USA 2007, pp: 5288-5292.

[15] A. Boukerche, A. Zarrad, D. Durate, L, Andrade and R. Araujo. "*A Novel Solution for the Development of Collaborative Virtual Environment Simulations in Large Scale*". Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Canada 2005, pp: 86-96.

[16] M. Castro, M. Costa, and A. Rowstron. "*Peer-to-peer overlays: structured, unstructured, or both?* ". Technical Report MSR-TR-2004-73, Microsoft Research, UK 2004.

[17] I. Chatzigiannakis, E. Kaltsa, and S. Nikoletseas. "*Evaluating the effect of user mobility and user density on the performance of ad-hoc mobile networks*". Journal Wireless Communications and Mobile Computing, Vol 4, Issue 6, USA 2004, pp: 609-621.

[18] Y. Chawathe, S. Ratnasamy, L. Breslau, and S. Shenker. "*Making Gnutella-like P2P Systems Scalable*". Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, Germany 2003, pp: 407-418.

[19] L. Chisalita, and N. Shahmehri. "*A peer-to-peer approach to vehicular communication for the support of traffic safety applications*". The 5th IEEE International Conference on Intelligent Transportation Systems, Singapore 2002, pp: 336-341.

[20] T. Chiu, and S. Hwang. "*Efficient Fisheye state routing protocol using virtual grid in high-density ad-hoc networks*". Proceedings Advanced Communication Technology, Korea 2006, pp: 4-11.

[21] H. Choi, H. Park, and M. Woo. "*An Enhanced Gnutella for Ad-Hoc Networks*". Proceedings of the International Conference on Systems and Networks Communication, French Polynesia 2006, pp: 3-13.

[22] H. Choi, H. Park, and M. Woo. "*Performance Analysis of Peer-to-Peer Application in Ad-hoc Networks*". Proceedings ITS Telecommunications, France 2005, pp: 49-52.

[23] M. Claypool, and K.Claypool. "*Latency and player actions in online games*". Communications of the ACM, Special issue: entertainment networking, Vol 49, Issue 11, USA 2006, pp: 40-45.

[24] M. Conti, E. Gregori, and G. Tur. "*A cross-layer optimization of gnutella for mobile ad hoc networks*". Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, USA 2005, pp: 343-354.

[25] P. Costa, and D. Frey. "*Publish-subscribe tree maintenance over a DHT*". Proceedings of the 4th International Workshop on Distributed Event-Based Systems, USA 2005, pp: 412-420.

[26] A, Datta. "*MobiGrid: P2P Overlay and MANET Rendezvous a Data Management Perspective*". Proceedings of the 15th Conference On Advanced Information Systems Engineering, Austria 2003, pp: 123-131.

[27] S. Davidson, H. Garcia-Molina, and D. Skeen. "*Consistency in Partitioned Networks*". Proceedings ACM Computing Surveys, Vol 17, Issue 3, USA 1985, pp: 341-349.

[28] Digital Media Homepage (2007): http://www.slyck.com/

[29] C. Diot, and L. Gautier. "*A distributed architecture for multiplayer interactive applications on the Internet*". IEEE Network Magazine, Vol 13, Issue 2, 1999, pp: 6-15.

[30] Emixode Home Page (2007): http://rfc-gnutella.sourceforge.net/src/pingreduce.html

[31] A. Ephremides, J. Wieselthier, and D. Baker. "*A design concept for reliable mobile radio network with frequency hopping signaling*". Proceedings of IEEE, Vol 75, Issue 1, 1987, pp: 56-73.

[32] EverQuest Homepage (2008): http://www.everquest2.station.sony.com/

[33] FreeNet Homepage (2008): http://freenetproject.org/

[34] E. Frécon, and M. Stenius. "*DIVE: A scaleable network architecture for distributed virtual environments*". Journal Distributed Systems Engineering (DSEJ), Vol 5, Issue 3, 1998, pp: 91-100.

[35] A. Funkhouser. "*RING: a client-server system for multi-user virtual environments*". Proceedings symposium on Interactive 3D graphics, USA 1995, pp: 85-93.

[36] M.Gerla, and J Tzu-Chieh. "*Multicluster, mobile, multimedia radio network*". Journal Wireless Networks, Vol 1, Issue 3, 1995, pp: 255-265.

[37] V. Gianuzzi. "*Data Replication Effectiveness in Mobile Ad-Hoc Networks*". Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc sensor and ubiquitous networks, Italy 2004, pp: 17-22.

[38] Gnutella V6 Specification (2006): http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[39] F. Greenhalgh. "*Analysing movement and world transitions in virtual reality tele-conferencing*". Proceedings of 5th European Conference on Computer Supported Cooperative Work, UK 1997, pp: 313-328.

[40] I. Gruber, R. Schollmeier, and W. Kellerer. "*Performance Evaluation of the Mobile Peer-to-Peer Service*". Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, USA 2004, pp: 363- 371.

[41] M. Guarnera, M. Villari, A. Zaia, and A. Puliafito. "*MANET: possible applications with PDA in wireless imaging environment*". The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol 5, Portugal 2002, pp: 2394-2398.

[42] O. Hagsand. "*Interactive MUVEs in the DIVE System*". IEEE Computer, Vol 3, Issue 1, 1996, pp: 30-39.

[43] T. Hara. "*Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility*". Proceedings INFOCOM, USA 2001, pp: 1568-1576.

[44] L. Hao-Jun, Q. Fei-Yue, and L. Jun. "*Research on Mechanism Optimization of ZRP Cache Information Processing in Mobile Ad Hoc Network*". Proceedings Wireless Communications, Networking and Mobile Computing, USA 2007, pp: 1593-1596.

[45] E. Harjula, M. Ylianttila, J. Ala-Kurikka, J. Riekki, and J. Sauvola. "*Plug and Play Application Platform: towards mobile peer-to-peer*". Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, , USA 2004, pp: 63-69.

[46] A. Hayes, and D. Wilso. "*Peer-to peer information sharing in a mobile ad hoc environment*". Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Application, UK 2004, pp: 154-162.

[47] D. Howie, M. Ylianttila, E. Harjula, and J. Sauvola. "*State-of-the-Art SIP for Mobile Application Supernetworking*". Nordic Radio Symposium 2004, Including Finnish Wireless Communications Workshop, Finland 2004.

[48] T. Hsin-ting, B. Thai, and A. Seneviratne. "*Supporting mobile devices in Gnutella file sharing network with mobile agents*". Proceedings 8th IEEE International Symposium on Computers and Communication, Turkey 2003, pp: 1035-1040.

[49] S. Hu and G. Liao. "*Scalable Peer-to-Peer Networked Virtual Environment*". Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, USA 2004, pp: 129-133.

[50] A. Huhtonen. "*Comparing AODV and OLSR Routing Protocols*". http://www.tml.tkk.fi/Studies/T-110.551/2004/papers/Huhtonen.pdf, 2004.

[51] S. Hu, J. Chen, and T. Che. "*VON: a scalable peer-to-peer network for virtual environments*". appear in IEEE Network Magasine, Vol 20, Issue 4, 2006, pp: 22-31.

[52] IETF Working Group MANET, Mobile Ad-hoc Networks (MANET) Charter (2005): http://www.ietf.org/html.charters/manet-charter.htm

[53] T. Iimura, H. Hazeyama, and Y. Kadobayashi. "*Zoned federation of game servers: a Peer-to-Peer approach to scalable multi-player online games*". Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, USA 2004, pp: 116-120.

[54] A. Iwata, C. Chiang, G.Pei, M. Gerla, and W. Chen. " *Scalable Routing Strategies for Ad-hoc Wireless Networks*". Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, Vol 17, Issue 8, 1999, pp: 1369-1379.

[55] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva. "*The dynamic source routing protocol for mobile ad hoc networks*". Internet Draft, Internet Engineering Task Force, 2001. http://www.ietf.org/internetdrafts/draft-ietf.

[56] JTella Homepage (2008): http://jtella.sourceforge.net/

[57] JXTA java Micro Edition project (2006): http://jxme.jxta.org

[58] Kazaa Homepage (2007): http://www.kazaa.com/us/index.htm

[59] R. Khalaf, A. El-Haj-Mahmoud, and A. Kayss. "*Performance Comparison Of The AODV and DSDV Routing Protocols In Mobile Ad Hoc Networks*". Proceedings Communication System and Networks, Spain 2002, pp: 17- 23.

[60] A. Klemm, C. Lindemann, and O. Waldhorst. "*Peer-to-peer Computing in Mobile Ad Hoc Networks*". Performance Tools and Applications to Networked Systems, , LNCS 2965, Springer-Verlag, 2004, pp: 187-208.

[61] B. Knutsson, L, Honghui, and B. Hopkins. "*Peer-to-peer support for massively multiplayer games*". Proceedings of the INFOCOM Conference, China 2004, pp:107-115.

[62] G. Kortuem, J. Schneider and D. Preuitt. "*When peer-to-peer comes face-to-face: collaborative peer-to-peer computing in mobile ad-hoc networks*". Proceedings of the First International Conference on Peer-to-Peer Computing, Sweden 2001, pp: 75-82.

[63] G. Kortuem. "*Proem: a middleware platform for mobile peer-to-peer computing*". ACM SIGMOBILE Mobile Computing and Communications Review, Vol 6, Issue 4, USA 2002, pp:62-64.

[64] P. Kumar, G. Singh, A. Mitchell, T. Das and K. McGee, "*NetEffect: A Network Architecture for Large- scale Multi-user Virtual Worlds*". Proceedings ACM Virtual Reality software and tecchnology, Switzerland 1997, pp: 157-163.

[65] R. Lea, Y. Honda and S. Matsube. "*Community Place: architecture and performance*". Proceedings of the second symposium on Virtual reality modeling language, USA 1997, pp: 41-50.

[66] E. Lety, T. Turletti, F, Baccelli. "*SCORE: a scalable communication protocol for large-scale virtual environments*". Journal IEEE/ACM Transactions on Networking, Vol 12, Issue 2, 2004, pp: 247-260.

[67] LimeWire PONG caching Homepage (2008): http://www.limewire.org/wiki/index.php?title=Pong_Caching

[68] Limewire Homepage (2008): http://www.limewire.com

[69] V. Lo, D. Zahou, Y. Liu, G. Dickey and J. Li. "*Scalable Supernode Selection in Peer-to-Peer Overlay Networks*". Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems. USA 2005, pp:18-27.

[70] B. Loo, J. Hellerstein, R. Huebsch, S. Shenker and I. Stoic. "*Enhancing p2p file sharing with an internet scale query processor*". Proceedings of the 30th International Conference on Very Large Data Bases, Canada 2004, pp: 432-443.

[71] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. "*Search and replication in unstructured peer-to-peer networks*". Proceedings of the 16th international conference on Supercomputing, USA 2002, pp: 84-95.

[72] M. Macedonia, and M. Zyda. "*A Taxonomy for Networked Virtual Environments*". IEEE MultiMedia, Vol 4, Issue 1, USA 1997, pp: 48-56.

[73] D, McGregor, A. Kapolka, M. Zyda, and D. Brutzma. "*Requirements for large-scale networked virtual environments*". Proceedings of the 7th International Conference on Telecommunication, Croatia 2003, pp: 353-358.

[74] R. Melamed, and I. Keidar. "*Araneola: A scalable reliable multicast system for dynamic environments*". Proceedings 3rd International Symposium Network Computing and Applications, USA 2004, pp: 5-14.

[75] R. Michael R, M. Macedonia, J. Zyda, R. David and R. Pratt. "*NPSNET: A Network Software Architecture for Large Scale Virtual Environments*". Presence: Teleoperators and VirtualEnvironments, Vol 3, USA 1994, pp: 256-287.

[76] MiMAZE Homepage (2008): http://www-sop.inria.fr/rodeo/MiMaze/

[77] MMORPG Homepage (2008): http://www.mmorpg.com/

[78] Morpheus Homepage (2006): http://www.morpheus.com/

[79] Napster Homepage (2008): http://www.napster.com

[80] Network simulator ns-2 Homepage (2008): http://www.isi.edu/nsnam/ns/

[81] M. Oliveira, J. Crowcroft, and M. Slate. "*An Innovative Design Approach to Build Virtual Environment Systems*". Proceedings of the workshop on Virtual environments, Switzerland 2003, pp: 143-151.

[82] S. PalChaudhuri, J. Le Boudec, and M. Vojnovic. "*Perfect simulations for random trip mobility models* ". Proceedings. 38th Annual Simulation Symposium, USA 2005, pp: 72-79.

[83] S. Pandžić, K. Tolga, L. Elwin, N. Thalmann, and D. Thalmann. "*A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments*". Proceedings Eurographics, Hungary 1997, pp: 177-188.

[84] K. Park and V. Kenyon. "*Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment*". Proceedings of the IEEE Virtual Reality, USA 1999, pp: 104-111.

[85] M. Peterson. "*Learning interaction in an Avatar-based Virtual Environment: a Preliminary Study*". Journal Pacific Association for Computer Assisted Language Learning, Vol 1, Issue. 1, 2005, pp: 29-40.

[86] H. Pucha, M. Das, and Y. Charli. "*The performance impact of traffic patterns on routing protocols in mobile ad hoc networks*". Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, Italy 2004, pp:211-219.

[87] Quak Homepage (2008): http://www.quake4game.com/

[88] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "*A scalable content-addressable network*". Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, USA 2001, pp: 161-172.

[89] A. Roach. "*Session Initiation Protocol (SIP)-Specific Event Notification*", Internet proposed standard RFC 3265, RFC Editor, 2002.

[90] S. Rooney, D. Bauer, and R. Deydier."*A federated peer-to-peer network game architecture*". IEEE Communication Magasine, Vol 42, Issue 5, 2004, pp: 114-122.

[91] A. Rowstron and P. Druschel. "*Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*". Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms, Germany 2001, pp: 329-350.

[92] SecondLife Homepage (2007): http://secondlife.com/

[93] H. Shen, M. Joseph, M. Kumar and S. K. Das. "*PReCinCt: A Scheme for Cooperative Caching in Mobile Peer-to-Peer System*". Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, USA 2005, pp: 57-66.

[94] H. Shen, M. Kumar, and Z. Wong. "*Energy-Efficient Caching and Prefetching with Data Consistency in Mobile Distributed Systems*". Mobile Networks and Applications, Vol 10, Issue 4, 2005, pp: 475-486.

[95] G. Singh, L. Serra, W. Pong, and H. Ng. "*BrickNet: A Software Toolkit for Network-Based Virtual Worlds*". Presence: Teleoperators and Virtual Environments, Vol 3, Issue 1, USA, pp: 19-34.

[96] SPLINE Homepage (2007): http://www.merl.com/projects/spline/

[97] M. Srivatsa, B. Gedik, and Ling Liu. "*Large Scaling Unstructured Peer-to-Peer Networks with Heterogeneity-Aware Topology and Routing*". IEEE Transactions on Parallel and Distributed Systems, Vol 17, Issue 11, 2006, pp: 1277-1293.

[98] I. Stoica, R Morris, D. Karger, M. Kaashoek, and H. Balakrishnam. "*Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*". Proceedings of the ACM SIGCOMM Conference, USA 2001, pp: 149-160.

[99] StraCraft Homepage (2008): http://www.starcraft2.com/

[100] M. Sung; J. Lee; and Y. Je Heo. "*Towards reliable peer-to-peer data sharing over mobile ad hoc networks*". Vehicular Technology Conference, Sweden 2005, pp: 2196-2200

[101] B. Tang, Z. Zhou, A. Kashyap, and T. Chiueh. "*An Integrated Approach for P2P File Sharing on Multi-hop Wireless Networks*". Proceedings of the IEEE Int. Conference on Wireless and Mobile Computing, Networking and Communication, Canada 2005, pp: 268-274.

[102] The IClouds Homepage (2007): http://www.iclouds.tk.informatik.tu-darmstadt.de

[103] UltimaOnline Homepage (2007): http://www.uo.com

[104] V. Vishnumurthy, and P. Francis. "*On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks*". Proceedings 25th IEEE International Conference on Computer Communications, Spain 2006, pp: 1-12.

[105] A. Wang, M. Norum, and C. Lund. "*A peer to peer Framework for Mobile Collaboration*". Proceedings Software Engineering Applications, USA 2006, pp: 108-115.

[106] Wikipedia home page (2008): http://en.wikipedia.org/wiki/Orders_of_magnitude_(speed)

[107] Wikipedia Homepage (2009): http://en.wikipedia.org/wiki/September_11,_2001_attacks.

[108] M. Wloka. "*Lag in Multiprocessor VR*". Presence: Teleoperators and Virtual Environments (MIT Press), Vol 4, Issue 1, 1995, pp: 50-63.

[109] Z. Zhange, Y. Liu, and L. Xiao. "*Dynamic layer management in super-peer architectures*". Proceedings of the International Conference on Parallel Processing, Canada 2004, pp: 29-36

[110] B. Zhao, J. Kubiatowicz, and A. Josep. "*Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*". Tech. report UCB/CSD-01-1141, UC.Berkely, 2001.

[111] S. Zhao, D. Stutzbach, and R. Rejaie. "*Characterizing Files in the modern Gnutella Network: A Measurement Study*". Proceedings of SPIE Multimedia Computing and Networking, USA 2006, pp: 113-126.