Faculty of Electrical Engineering, Mathematics and Computer Science
Network Architectures and Services

Analysis and deployment of the BitTorrent protocol for Community Ad-hoc Networks

E.D. Ayele
(4033507)

Committee members:
Supervisor: Dr.ir. F.A. Kuipers
Mentor: Dr.ir. R.C.J. Smets
Member:Dr.ir. E. Onur

Date August 11, 2011
M.Sc. Thesis No: PVM 2011-069

Table of Contents

List of Tables

List of figures

Abstract

The fast developments in smart devices and wireless data communication (3/4G, WiFi) create opportunities for new services for telecom operators and mobile software developers. This thesis investigates an example of such a novel service. It researchs the possiblity of file sharing in an ad hoc community based network (CAhN) created by smart devices. This thesis describes a typical CAhN use case taking both technical characteristics and community behaviour into account. The analysis of the existing protocols and the challenges and issues faced in the CAhN use case leads to recommend the use of the BitTorrent file sharing protocol for P2P file sharing. Subsequently the research presents an experimental setup and simulation to asses the performance of file sharing. Peer-to-peer concepts provide new possibilities but also challenges for distributed applications. The BitTorrent protocol establishes a virtual overlay network that addresses peer heterogeneity and dynamics for each local community ad hoc network. Thus, the system overcomes the topological restrictions and provides a scalable and robust high performance infrastructure for group communication. In this thesis work, a concept of community ad hoc network and the P2P overlay network is described. With the help of NS-2 simulations, the BitTorrent protocol is implemented and it is performance is tested in accordance with the defined parameters.

Chapter One

Introduction

1.1 Motivation

The wireless fidelity of today's small and affordable devices has risen the massive need for instant and effortless networking to a whole new level where user demand cannot be satisfied with the usual mobile Internet configurations. For example, sharing of large files using the internet can be a problem. What is expected from wireless devices today is close to the very definition of ad hoc networking, i.e. the ability to form a wireless mobile network anytime, anywhere without the use of any centralized administration or existing infrastructure [1].

Although they are owned and operated by individuals, smart devices are "social" devices by nature; we can talk, text, email, browse the web, run applications and access our favorite social networks. Even though smart devices become more capable every day, the conventional technology continues to rely on infrastructure-based communication. In this work the proposed concept is driven by the vision of facilitating and promoting anonymous social interaction by electronic personal devices. Examples of social mobile applications include the establishment of groups or communities based on shared interests (music, art, software, etc.) or activities (work, hobbies, etc).

The current existing communication systems do not fully support anonymous networking. Anonymity is considered very important when a person desires to protect one's identity from others. Anonymity is desired by anyone who fears retaliation, discrediting, or simply desires privacy. Anonymity finds uses amongst others in corporations for whistle blower cases; in law enforcement for anonymous crime tip lines; in universities for course and faculty evaluation; in government for political discussion and voting; and in business areas for customer feedback.

The following examples describe the current issues related to anonymity.

1. File sharing and anonymity

Because of the ability to intercept and manipulate network traffic by, e.g. government bodies and service operators, the Internet has lost most of its anonymity feature.

- Deep Packet Inspection (DPI), enables Internet data mining, eavesdropping, and censorship by the Internet Service Providers (ISPs). Although DPI technology has been ambiguously used for Internet management for many years, the technology has been used anti-competitively and has reduced the openness of the Internet.
- Control of the Internet by governments limits freedom of speech. People in a country with a repressive political regime often rely on anonymity to avoid persecution for advocating deviant political opinions. Even in democratic countries, some people claim, rightly or wrongly, that certain political opinions are scrutinized when identities are exposed. Exemplary scenarios where anonymity is useful to the end users are: employees who are dependent on an organization and afraid of revenge, may expose serious misuse. Anonymous tips can be used as an information source by newspapers, as well as by police departments when soliciting for tips aimed at catching criminals. When their identity is not revealed, users are more equal in anonymous discussions and factors like status, gender, etc. will not influence the evaluation of what they say.
- More often applications are invading the end-user's privacy. They deliberately spy on personal data details and transfer or sell them to third parties.
- School teachers and students can share class related material like documents, notes and rich media because teachers are always looking for ways to engage with students.

2. Congestion in radio access networks

Sharing large files using mobile networks can be an issue. For example, recording a video from a phone video camera and trying to send it by email or upload it to the web is difficult if it´s more than a minute long. This is mainly due to limited resources in the radio access part of the mobile network. If direct social interaction exits between the individual sending the video and a recipient, transfer of data could have happened directly from device to device. In this case messages and data would never have to travel all over the Internet. By separating the data path from the signaling path, the congestion imposed on radio access networks can be decreased.

In summary community ad hoc network (CAhN) can be used for:

- Sharing content
- Spreading of knowledge without concern for retaliation
- Off-loading of operator-owned wireless networks by restraining the growth of mobile traffic load in Radio Access Network
- Anonymous contribution

1.2 Goal and Scope of the thesis

The very first goal of the research is to find alternative communication methods for data and information sharing in an anonymous way. There are three major research issues for this thesis work, Figure 1:

1. How a typical CAhN use case looks like?
2. What are the technical challenges and design issues in such a CAhN use case?
3. What are the existing technologies that enable a file sharing experience?
4. How well does the preferred technology perform for file sharing in the typical CAhN use case?

The main research question is how the CAhN enhances the filesharing expereince among the commmunity in a typical use case? Item 1 to item 3 will reserach the issues concerning the typical use case developed. Item 4 is investigated using the Network Simulator Version 2.34 (NS2), where the different parameters determining the data transfer performance related to CAhN are analyzed. How to use these parameters given the loose participation of users and their temporal and spatial uncertainty in participating in local and perhaps very small P2P networks to arrive at instruction for the user that optimizes the transfer of files. The general scope of the CAhN project is shown in Table 1.

Table 1: Scope of the project

Scope of this thesis work	• CAhN concept description • Assessment of different file transfer mechanisms and selection of most promising one based on selected criteria • NS2 performance analysis of the selected transfer mechanism • Embedding of the mechanism in the system and impact on the use-case.
Future work (out of scope)	• Anonymity in data exchange in CAhN • Core application development for smart phones • Tracking System development

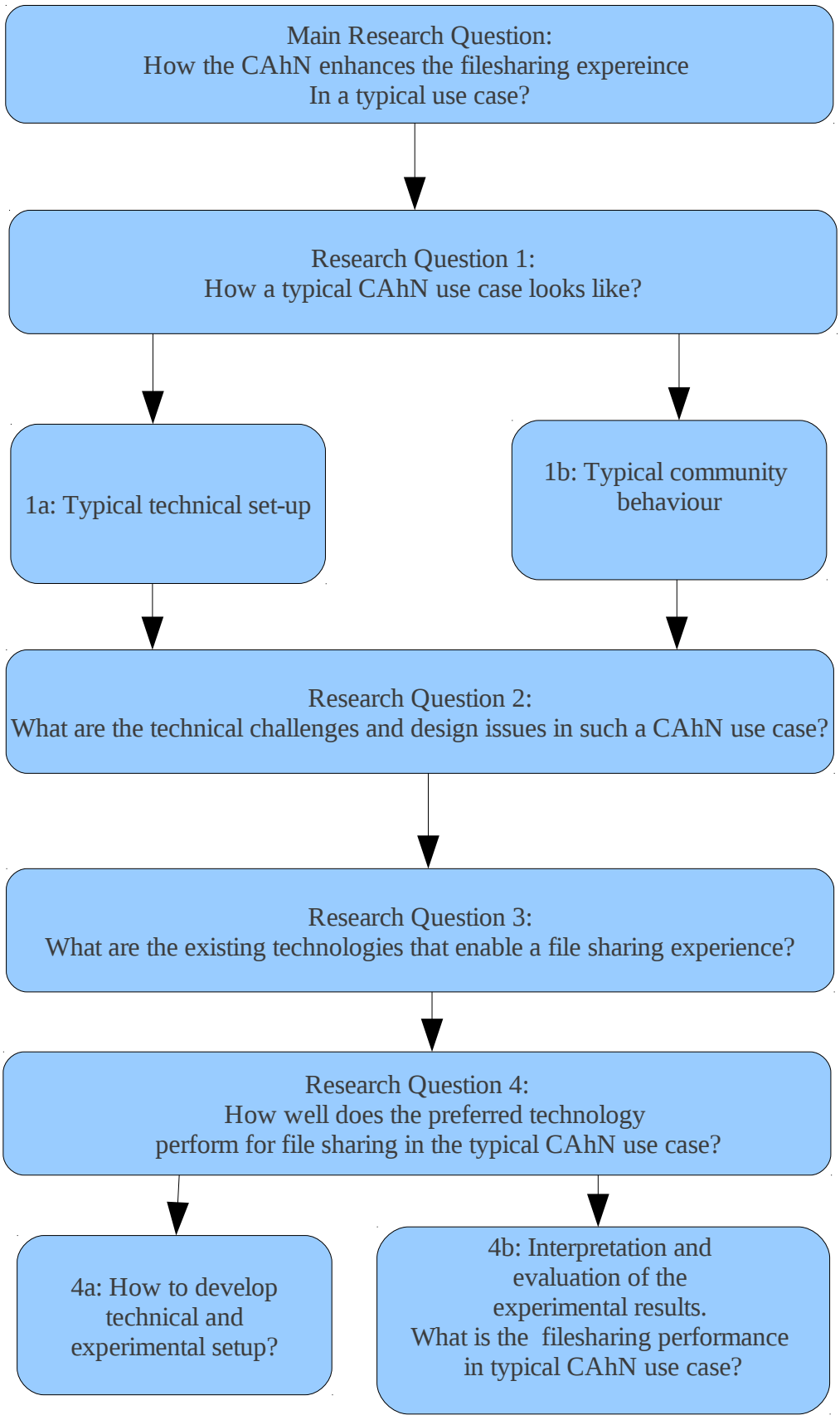

Figure 1: Research Issues for the thesis work

1.3 Structure of the report

The remainder of the report will focus on technologies that is used to locally transfer the files that need to be shared between mobile users. In Chapter 2 the typical use case is presented. In chapter 3 the concept of CAhN is described. Amongst this is the concept behind the community ad hoc Network. Existing research findings are reviewed. Consequently, specific shortcomings are identified and new research issues are established that are of importance to make this technology of interest. Chapter 4 discusses the proposed solution for the deployment of CAhN and the different anonymous file sharing techniques are presented. Chapter 4 describes the implementation of a local CAhN and the explanation of the NS-2 simulator. Chapter 5 presents the purpose and methodology of the experimentation for the NS-2. The NS-2 simulation result is shown in chapter 6, which will be helpful for assessing the performance of the selected P2P ad hoc local network. Chapter 7 presents the analysis of the results in light of existing literature and allows Chapter 8 to draw conclusions and propose further work and suggestions on this research topic.

Chapter Two

Introduction to CAhN, a typical high-level use-case

2.1 Introduction

In mobile ad hoc networks, mobile devices are usually carried by humans, so the movement of such devices is based on human decisions and social behavior. It is important to consider the behavior of individuals moving in groups and between groups because the mobile devices are carried by individuals in the community. In order to capture this type of behavior, group mobility must be modeled. Such a model is heavily dependent on the structure of the relationships among the people carrying the devices. As a basic characteristic of such a model the following assumption is made. In social networks, clustering appears to be far greater than in networks based on random stochastic models. This is related to the fact that humans usually organize themselves into communities. The effective utilization of localized and time bound networks as means for sharing information is impacted by both, the networking protocols and the mobility and communication patterns.

This means that in this localized and time bound framework the need for the network establishment dictates how ad hoc routing and P2P protocols are used. This need is dependent on the need of people to share certain information which shapes the topology of the network as well as the traffic on the application level in the network. The people who want to use the ad hoc network as a means for information sharing are part of an underlying social network that shows the way people are interconnected based on specific content or descriptors of content they require and offer. Figure 2 illustrates the concept for the use-case.

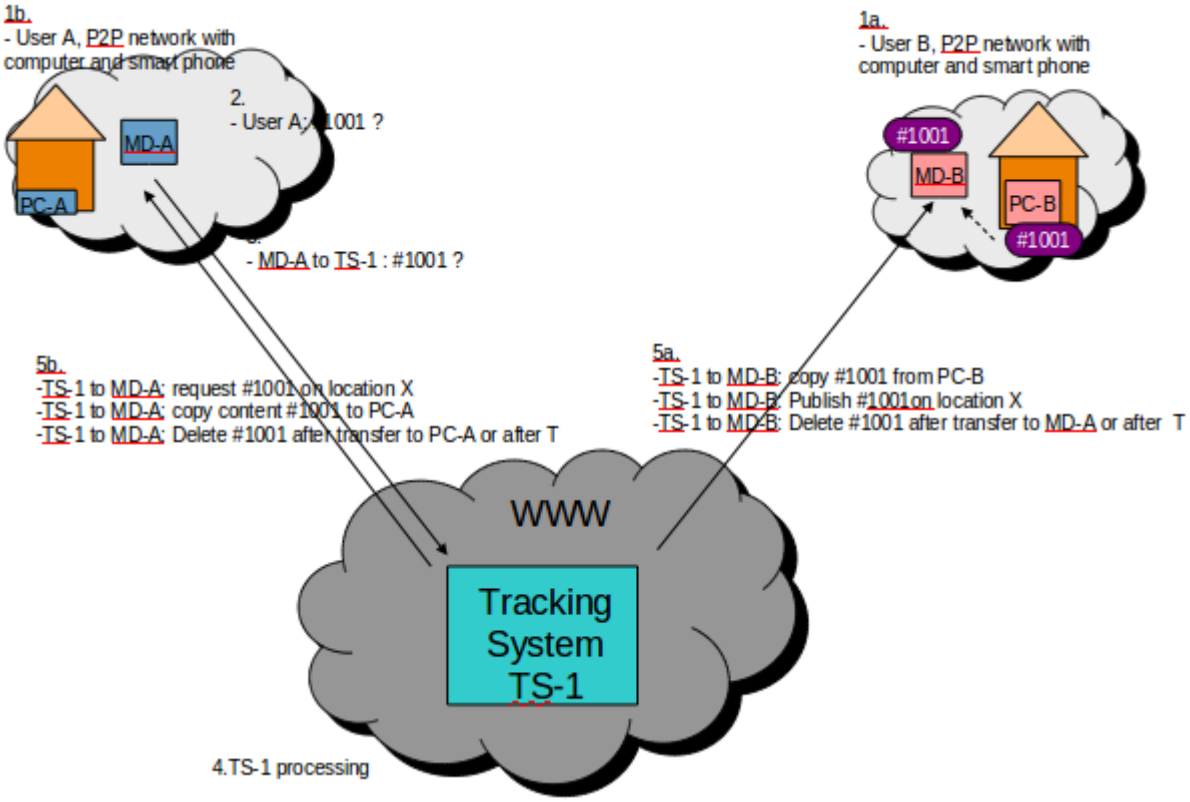

Figure 2: Community Ad hoc Networks use case for step 1-step 5

The different components which make up the CAhN concept are shown in Figure 2. This use-case is build around two users (A and B), each with a mobile device (MD-A and MD-B) and a home PC (PC-A and PC-B) and a central tracking system (TS-1) located on the Internet. The home PCs function as a storage facility for the mobile devices because MDs have limited capabilities compared to the storage available on a PC.

The Tracking System (TS-1), which is part of the Internet, receives requests from the mobile nodes and in return provides routing and topology information by means of an optimization function (not displayed). A node collects topology information of the community network and reports it to the TS. CAhN is designed to use P2P technology to transport contents directly between users. And to use the Internet for signaling purposes, i.e. for node and content information sharing in the P2P network and optimize routing and P2P file transfer in the local network. TS will be used as the main tracking and scheduling system for the CAhN. Its responsibility will be to calculate how the requests from one node can be transferred to another using intermediate node.

2.2 Use-case Specification and description

The use-case describes an example of how the mobile users can use the community ad hoc network to share a file. The use-case typically considers a case where a mobile peer joins a local community from a different local community. In this case, a specific file (content) has to be transferred form one person's PC to it is MD, and from that MD to a different MD which eventually stores it to the PC

belonging to the recipient peer. The two users are assumed to be not connected to each other before they meet at certain location.

1- The main components of the use-case are
- Mobile Devices (MD)
- Tracking System (TS)
- Personal Computers (PC)

2- Conditions
- There is an active Internet connection for the mobile device and community members.
- There is a file ready to be uploaded to the MD from the home PC, later to be shared with the community.

3- Flow of events in the use-case
- 1b and 1a: The use-case begins with two users, MD-A and MD-B, who have a smart phone and PC. Files that users share are known by TS by means of an ID.
- 2 and 3: User A needs a specific file with an ID of #1001; it then sends a file request to TS with the specific ID of the file, Figure 2.
- 4: TS will start processing the request sent from MD-A. Next, TS will realize that the requested file is situated in MD-B's PC, in a different local CAhN. Therefore, the TS will arrange a suitable condition to exchange the requested file.
- 5a: The TS will send a detailed instruction to user-B, to download the requested data from PC-B to MD-B. And to upload the requested file to the community using a time and location constraint. An example could be to publish between 10.00am and 12.00am (time constraint), publish at major train stations (location constraint), or publish during a concert (time and location constraint). In this use case the constraint is location-X.
- 5b: At the same time TS will inform user-A to request the file from the community also using time and location constraints. And then to transfer the data to its PC-A.
- 6: On location-X, Figure 3, MD-B becomes part of a new peer to peer network and shares the requested file with the community (It must be noted that the community may contain other users besides the user of MD-A, it may also be possible that MD-A is not present).
- 7: Based on expectations, MD-A will join the community formed at location-X. Then it will start downloading the required file. In this way it will get the file it wants.
- 8: Latter, user-A will transfer the file from its MD-A to its PC-A.
- 9a and 9b: Both user-A and user-B will report their status back to the TS (Figure 4). In this report information on successful transfer is provided, but also information on locations where the MD has been as well as the other nodes the user has seen. Obviously it is the user who determines the information he wants to share.
- The use-case ends.

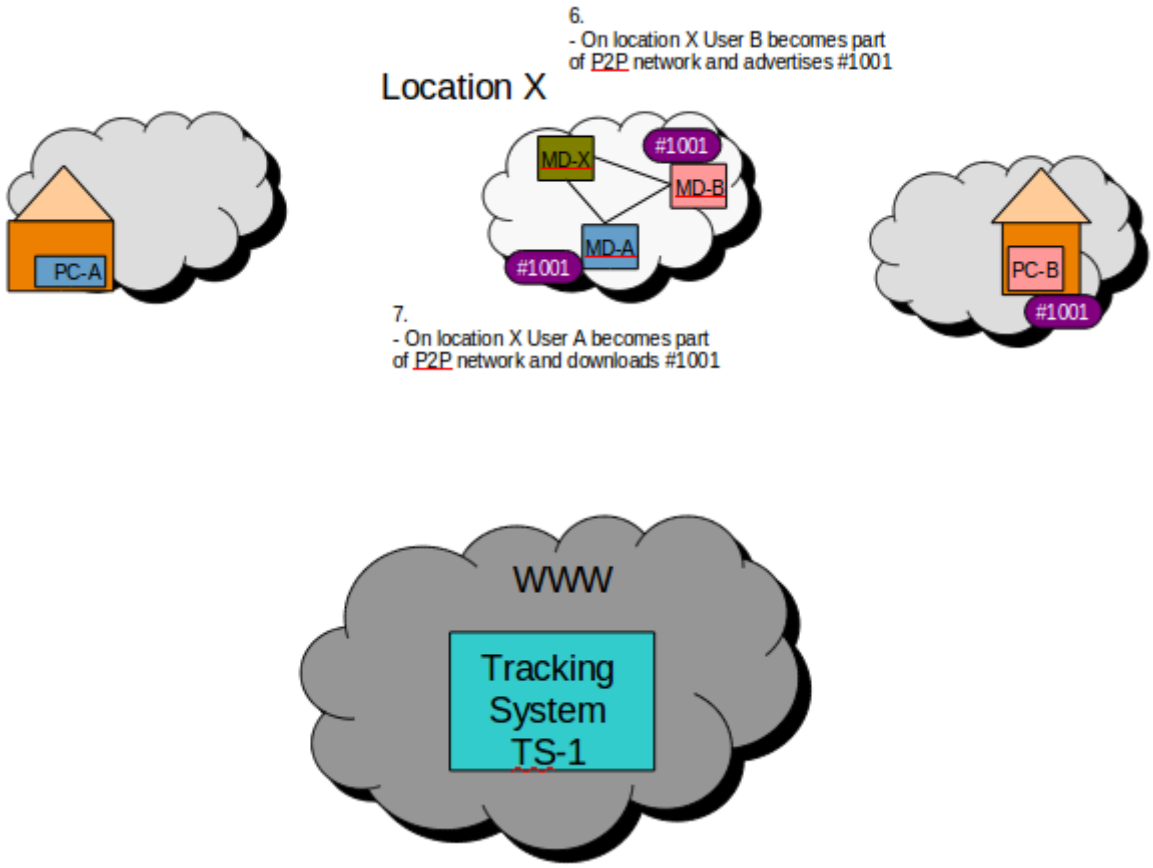

Figure 3: CAhN use-case for step 6 and step 7

The main components of CAhN are:

- *local CAhN*- it is a community created by the P2P network of devices where the trackerless protocol is implemented. A P2P technology is used here. The P2P protocol used could be a BitTorrent like system and the necessary instructions that helps to download a file is in the torrent by the TS.
- *Individual Peers*- communication devices in the community network
- *Tracking System (TS)*- where the signaling including peer content information is stored and processed for further instruction to be sent to the corresponding peers. All MDs in the community need to report to the TS with their peer specific information like:

  UP-LINK COMMUNICATION -> PEERS to TS:
  - Report the hash codes of the content that is available for sharing on their PC or MD
  - Peer ID
  - The location of the peers, so that the TS can track the location of every peer involved in the community.
  - The status of the peers- i.e. STARTED, STOPPED or COMPLETED the DOWNLOADING or UPLOADING of a file.
  - The UPLOADED DATA and DOWNLOADED DATA SIZE.

  DOWN-LINK COMMUNICATION -> TS to PEERS:

- The TS will analyze the statistical data received from all the peers and calculate the necessary performance metrics and accordingly will feedback instructive parameters to the community members to enable properly organized data sharing.

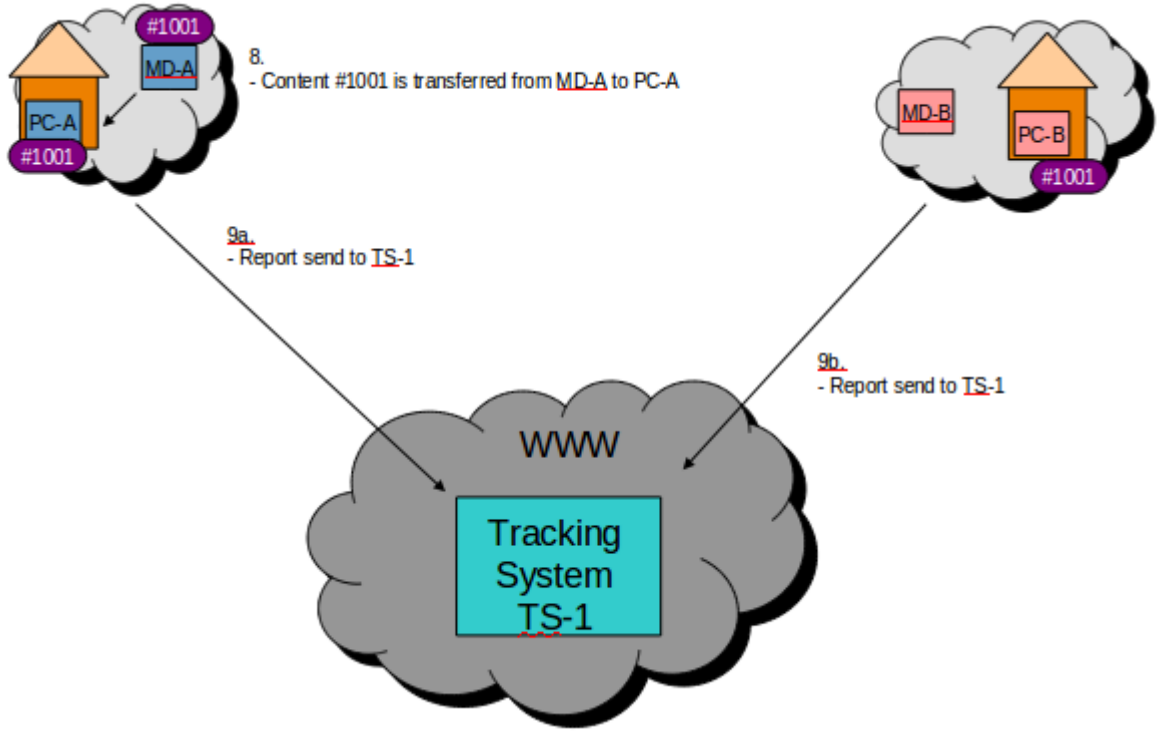

Figure 4: CAhN use-case for step 8 and step 9

Chapter Three

Technical issues and design challenges faced in a typical CAhN use-case

In this chapter the different design issues specific to Community ad hoc Networks are described. The technical issues related to a CAhN are also presented in accordance with the goal of the community ad hoc network, i.e. the provisioning of an alternative way for anonymous data sharing.

3.1 Technical issues related to CAhN development

3.1.1 Technical Issues

Community networks have somewhat complex structures since their application and context depend on the type of community from which it is formed. For instance communities of soccer fans are quite different from communities of formal company employees. Therefore, looking into the community the following technical issues can be identified:

- How can social network characteristics be used to design the CAhN data sharing algorithm?
- How to identify communities and groups in the network and how are communities associated to a geographical area; i.e. are people with strong social links likely to be geographically collocated?
- Where is most of the communication intensity or traffic flow? Is it in between nodes of the same cluster or between nodes of different clusters?
- The social interconnection between the network users has a significant influence on the network performance. When people that are part of a community meet, information present in such a community is likely to be of interest to all other members.
- What are social network characteristics and their impacts on the application layer? The groups of people that use a communication network are part of some kind of social network. In order to be able to incorporate the real life usage of communication networks for human information sharing, an application layer based on social network properties will be implemented.

For the community ad hoc network, special modifications have to be made to the application layer [1], and in the middle layers. This is due to the clustering or grouping structure of the network [1]. Even though there is a lot of work done related to ad hoc networks, most of the already developed technologies for ad hoc networks cannot be applied without adaptation [3].

Community ad hoc networks have special characteristics [3]:

- Essentially a CAhN consists of small clusters of communities, which themselves contain a distributed ad hoc network, interconnected to each other in ad hoc fashion.
- The clustering/grouping in a CAhN is high
- The nodes in a CAhN are not randomly distributed through the entire service area, rather their distribution depends on the social behaviors of the communities considered. In our case the communities are formed by the social interaction between humans.
- They are community-centric networks connecting different communication devices like smart phones.

3.1.2 Requirements and conditions of a CAhN

Some of the requirements and conditions related to CAhN [1]:

- The very dynamic and unpredictable topology.

  An ad hoc network is a dynamically reconfigurable wireless network with no fixed infrastructure. Each host can act as a router and may move in an arbitrary manner. Every node in the network is assumed to be mobile and dynamic in nature. The whole network topology can be affected by the movement of nodes in/out of the network. The movement of the nodes is monitored by the routing protocols employed in the networks. As far as the conventional ad hoc is concerned, there is no known established relation between nodes' content and the routing topology. But when in a CAhN, the community member aims at achieving correlation among the community members with respect to content or interests.

  Mobility of end-users determines the topology as well as the number of Local CAhNs that contain a particular “community color” or content. In Table 2 it is shown that there is only limited correlation between mobility patterns of individual users and the community they belong to [2]. Hence, if efficient transfer of information is to be realized, knowledge is needed on these mobility patterns and correlation between “content color” and “community color” is required.

Table 2: Example of how users belonging to the same or different community can exchange content with the same or different "color" as the community they belong to.

[Cross-relation between the column and row]	between two end nodes within a local CAhN	between two end-nodes across intra-community Local CAhNs
If content matches	Final delivery of information takes place (high-trust)	Final delivery is possible by a user of a different community, as a transit.
If content does not match	No final delivery, but because users have the same content other user may decide to help transport content between Local CAhNs	No exchange of information allowed

- The mobility of a local CAhNs and its individual components. [4]
  This means that the mobility of individual nodes inside the Local CAhN can ultimately make the design of community networks challenging. Unlike conventional ad hoc networks, node movement does not follow random patterns. Rather, it follows typical human movement patterns [4], which require either modeling or prediction using e.g. Markov chains.

- The need for self-organization, self-healing, self-forming etc.
  In this case the system has to deal with topology changes when nodes enter and leave the network. This is an important feature and generic to ad hoc networks. Because the topology of the network is changed by entering and leaving nodes.

- The heterogeneity of the underlying link technologies.
  With increasing diversity of wireless devices, heterogeneity in link transmission power is common in ad hoc networks. Since the routing protocols are sensitive to the node power level, the design has to address this issue for the community network. This relates to the way a local ad hoc network is formed to exchange content. In the case of using a BitTorrent transport protocol, all nodes need to see each other in the network. For this they must all use the same link-technology or act as routers.

- The variety of power, processing and storage constraints of the involved devices.
  The device's memory can also affect the performance of the designed community network. For example, if the nodes in the network are memory constrained, they will be forced to drop packets when their storage limit is exceeded. This will impact the performance of the network negatively.

- The need for fast localization of devices and services.
  Users may only be part of a local CAhN for a limited number of minutes where they can exchange information. A mechanism must be put in place to optimize the transfer of content to global level, including the other local CahNs, to achieve efficiency as high as possible. This requires global information of the local communities that might only be available on the

coordinating server.

- The geographical spreading of the communities.
  Community members may spread across different geographical locations due to their occupation or interests. Knowing the geographical property of communities can simplify file routing to the respective destination. Alternatively keeping track of other members in their vicinity as a function of time provides equivalent information.

3.1.3 Issues related to the different communication layers

Issues concerning the middleware, including MAC and radio access interface layers are [1]:

1-Architecture
The architecture should:

- Expose the structure of the underlying system but hide the implementation details.
- Realize the use case and scenario and possible variants.
- Allow the specification of the requirements of various components.
- Meet both functional and quality requirements.

In terms of creating an architectural framework the following generic issues need to be resolved [2]:

- Which co-operating entities need to be defined to provide a particular functionality?
- How is the mapping of functional entities to physical devices in the network?
- How the model for co-operation between the functional entities is, e.g., is it a tightly coupled co-operation; is it in an asymmetric client server mode or a symmetric peer-to-peer mode?
- What underlying communication services and protocols are required between these functional entities?

2-Resource Discovery
Resource discovery encompasses locating and retrieving information in large, complex networked environments, including the community wide network. A network must be able to discover what devices, networks and services are around that it has the opportunity to link up with. This is called resource discovery [2].
For resource discovery, specific questions that need to be answered are:

• How should resources be characterized in terms of their functionality and their quality, such that their identities and capabilities can be communicated?
• How can a CAhN find out which resources are around and available either locally or remotely?

3-Context Discovery
According to [3] context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered to be relevant to the interaction between a user and an application, including the user and application themselves. Examples for context are identity, spatial information (e.g. location, orientation, speed), temporal information (e.g. time of the day, date), environmental information (e.g. temperature, air quality, and light) or noise level, social situation (e.g. who you are with, and people that are nearby), resources that are nearby (e.g. accessible devices, and hosts), availability of resources (e.g. battery, display, network, and bandwidth), schedules and agendas.

Discovering what context a CAhN or its components are operating in is in general important since it may determine what applications are allowed to run and what entities are needed.

4-Self-Organization
Self-organization is an important component for a successful ability to establish networking whenever needed. Since the Community network will encompass nodes which might dynamically leave or join the network, therefore, the network should be able to self-organize itself after every reconfiguration of the communication system.

5-Addressing and Routing
Addressing is related to the actual transport of messages to specific network elements. What is specific for a CAhN regarding addressing is that

- They have a composite and heterogeneous structure, consisting of devices and cooperating in ad hoc clusters of devices.
- The devices within the community are moving with respect to each other, and the different communities move with respect to each other.

Routing takes place at different levels, i.e. routing within the community (intra-community), and routing among different communities (inter-community routing).

6-Co-operation with infrastructure-based networks and other interconnection structures
The types of infrastructure-based networks a CAhN might need to co-operate with are:

- Public cellular networks, e.g., GPRS and UMTS.
- Public or private fixed networks, e.g., one of the temporary nodes of a CAhN may be a stationary device with both radio interfaces and a fixed connection to the Internet, an organization's intranet or extranet, or a private home network.
- Public or private WLAN infrastructures, e.g., wireless campuses, WLANs in airport lounges, cities, train stations and inside trains.

7-Privacy, Security and Accounting
As far as Community Networks are concerned Privacy, Security and Accounting are the most important. The network should be able to provide some sort of privacy and security for the users in the community. User information systems might contain confidential and private information that can become compromised if left unprotected. Unauthorized use of an accounting system can be disastrous, risking loss of information, bad data input and misuse of confidential information. Therefore, security of the systems is a priority.

3.2 Summary

In this chapter the issues concerning a CAhN is introduced. A CAhN implements the concept of distributive computing, where peers interact with their members in the community. The design issues, requirements and issues related to different layers in the ad hoc communication framework are discussed. It is shown that this concept introduces the challenges: the heterogeneity of the involved technology, the need for self-organization, and the dynamics of the system composition. The impact of these issues on the network design is implicated. Many of the concepts shown here could be applied to the CAhN domain. Research activities focused on this specific area are needed to make the concept of CAhN effective.

Chapter Four

Survey of file sharing protocols

One of the main aims of CAhN is to provide an alternative way to exchange information. As information can be related to all kinds of applications, a choice has been made to focus on the exchange of files. This Chapter will start by first describing the different available anonymous file sharing techniques and show the relation between MANETs and P2P, concurrently, will continue to discuss why BitTorrent is chosen for this thesis work.

4.1 Anonymous file sharing techniques

An anonymity application is an application which can be used to access a network, such as the Internet, anonymously. The application can use either private or public anonymous proxy servers, encryption or diversion and to ensure an anonymous and/or difficultly traceable access to a resource. Examples: Anonymous P2P, I2P (anonymous network), Tor (anonymity network). Figure 5 shows some of the existing methods for anonymous file sharing.

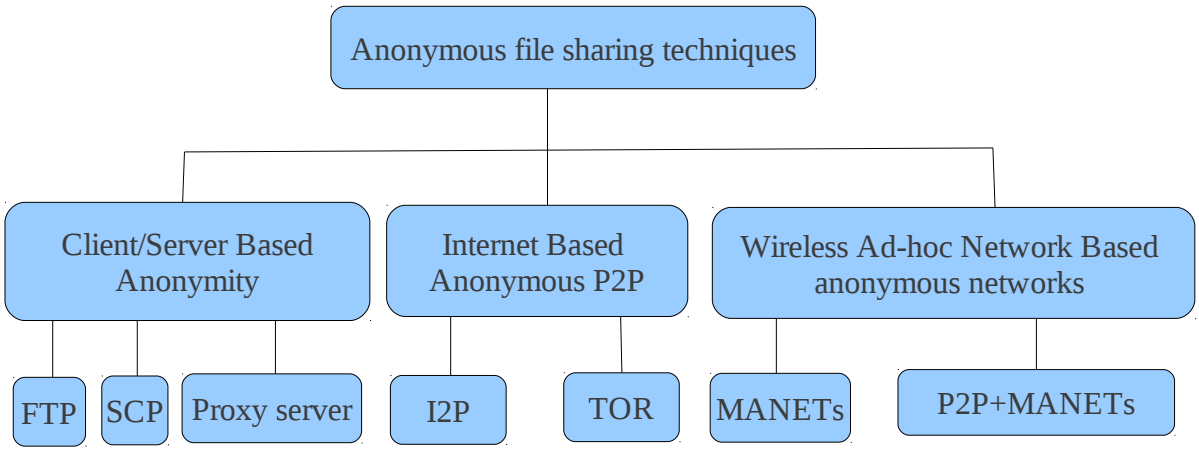

Figure 5: Anonymous file sharing techniques

4.1.1 Client/Server based

File Transfer Protocol (**FTP**) is a standard network protocol used to transfer files from one host to another over a TCP-based network, such as the Internet [12]. FTP is built on a client-server architecture and utilizes separate control and data connections between the client and server. FTP users may authenticate themselves using a clear-text sign-in protocol but can connect anonymously if the server is configured to allow it. A host that provides an FTP service may additionally provide anonymous FTP access. Users typically log into the service with an 'anonymous' account when prompted for user name. Although users are commonly asked to send their email address with a password, no verification is actually performed on the supplied data. Many FTP hosts whose purpose is to provide software updates will provide anonymous logins.

Secure Copy or **SCP** is a means of securely transferring computer files between a local and a remote host or between two remote hosts. It is based on the Secure Shell (SSH) protocol [13]. For upload, the client feeds the server with files to be uploaded, optionally including their basic attributes (permissions, timestamps). This is an advantage over the common FTP protocol, which does not have provision for uploads to include the original date/timestamp attribute. For downloads, the client sends a request for files or directories to be downloaded. When downloading a directory, the server feeds the client with its subdirectories and files. Thus the download is server-driven, which imposes a security risk when connected to a malicious server.

Proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers [12]. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server evaluates the request according to its filtering rules. For example, it may filter traffic by IP address or protocol. If the request is validated by the filter, the proxy provides the resource by connecting to the relevant server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it 'caches' responses from the remote server, and returns subsequent requests for the same content directly. A proxy server has a large

variety of potential purposes, including:

- To keep machines behind it anonymous (mainly for security).
- To speed up access to resources (using caching).

4.1.2 Internet-based anonymous P2P Overlay Networks

The **Tor** anonymity network ('Tor' for short) is a system aiming at online anonymity [14]. Tor is an implementation of onion routing. It works by relaying communications through a network of systems run by volunteers in various locations. By keeping some of the network entry points hidden, Tor is also able to evade Internet censorship [15]. Tor is intended to protect users' personal freedom, privacy, and ability to conduct confidential business. But like all current low anonymous networks, Tor cannot and does not attempt to protect against monitoring of traffic at the boundaries of the Tor network, i.e. the traffic entering and exiting the network. While Tor does provide protection against traffic analysis, it cannot prevent traffic confirmation (also called end-to-end correlation).

The **I2P** anonymous network [16] is a proxy network aiming at online anonymity. It implements garlic routing, a variant of onion routing that encrypts multiple messages together to make it more difficult for attackers to perform traffic analysis. This is an enhancement of Tor's onion routing. I2P is fully distributed and works by encrypting all communications in various layers and relaying them through a network of routers run by volunteers in various locations. By keeping the source of the information hidden, I2P offers censorship resistance. The goals of I2P are to protect users' personal freedom, privacy, and ability to conduct confidential business. Each user of I2P runs an I2P router on their computer (node). The I2P router takes care of finding other peers and building anonymizing tunnels through them. I2P is a low anonymity network, and there are limits to the anonymity offered by such a system. Instead it tries to make attacks more and more expensive to mount.

4.1.3 Wireless ad hoc network based anonymous networks

Wireless stations in ad hoc networks communicate with each other by broadcasting signals to whomever listens to the wireless channel [12]. It is easy for attackers to eavesdrop on the messages. Moreover, a MANET is a non-infrastructure network. Every node within the network can be both an end-user and a router and it may move frequently. If the attackers could insert or compromise a node in the network, they obtain the same rights as all the other legitimate nodes. In this case, anonymity requirements are obviously tougher than usual. In hostile environments, communication anonymity has to be preserved. Network members should not reveal their real identities to others (identity-free). Their locations also need to be concealed to protect themselves (location anonymity). The source and destination of a flow should be indistinguishable among all the nodes (source/destination anonymity) and their end-to-end relation should be hidden (end-to-end communication relation anonymity). All these strict requirements become the tasks of anonymous routing protocols. Fortunately, a variety of techniques have been proposed and utilized. Onion structure [17] is a frequently used technique in MANET anonymous routing solutions. It provides anonymous connections that are strongly resistant to both eavesdropping and traffic analysis. Previously proposed anonymous routing schemes like ARM [18], ASR [19] and ANODR [20] have provided anonymity for MANET communications on certain levels, but they all suffer from different defects. Their most common drawback is that they

sacrifice the networking performance. Normally, a routing protocol has the route discovery phase and the message transfer phase. During the second phase, most of the prior schemes require the packets to be encrypted and decrypted at each node in the path. These cryptographic operations incur large cost. To cope with this problem, a solution is proposed which implements P2P protocols together with the existing ad hoc routing protocols. Deploying an architecture that facilitates both the lookup and sharing of content over a wireless ad hoc network during the route discovery phase, will considerably increase the performances of these routing protocols with their default configuration. Therefore, the difficulties will be both in the lookup of data and in their sharing once localization is achieved.

The research into mobile ad hoc networks, relevant to this project, addresses the issue of how to enable peer-to-peer like applications on smart devices to share files. Most of those approaches were mainly for enabling information exchange triggered manually by users or make information dissemination completely independent of the user. In contrast, CAhN will focus on ad hoc network establishment of community constrained interactions by use of MANET. AdSocial, an experimental software platform supporting social network applications in ad hoc networks is described in [11]. AdSocial targets small-scale scenarios such as friends playing a game on the train or co-workers sharing calendar information. To demonstrate the importance of social networks and related applications AdSocial used ad hoc network technology to interconnect smart mobile-devices.

AdSocial integrates a diverse set of functionalities typically found in social network sites. It is designed to support spontaneous interactions and provide the basis for running collaborative applications. AdSocial is implemented as a local web application running in a regular web browser. Due to these features, it has the following strengths and weaknesses.

Strength:

- Their method of ad hoc network set-up is quite feasible and easy to implement. Since the developed application uses the already installed web browser, to run the whole java script, this increases its advantage over other applications.
- AdSocial operates on peer-to-peer ad hoc WLAN, and it is independent of online servers and infrastructure connections during use.
- AdSocial runs in automatic or interactive modes. AdSocial can stay idle, automatically receive and distribute files when other users are within range.
- is portable to other Linux platforms, it is written entirely in Java

Weakness:

- AdSocial cannot alert the users if something interesting is happening, like someone wanting to have a chat.
- It does not include the interactive mode, i.e. users can search others and contents from the ad hoc network around them and have interactive community chats or private messages.
- No battery-optimization operation
- The framework is fixed, new application plug-in cannot be added as needed.
- Scope is limited to geographic area covered by the ad hoc network. Therefore AdSocial does not support peer-to-peer exchange of information beyond its geographic area. Moreover, signaling complexity in a peer-to-peer network increases with increasing size of the ad hoc network.

4.1.4 P2P Protocols over MANETs

The motivation for supporting P2P overlay abstractions in MANETs is that MANETs and P2P overlays share many essential characteristics such as decentralization that make P2P applications developed for the Internet potential candidates for deployment in MANETs. For example, P2P file-sharing applications such as Gnutella are designed with a server-less architecture [21], which makes them potentially well suited to the infrastructure-less MANET environment. Structured P2P overlays developed for the Internet have been shown to provide a general substrate for building a variety of scalable and robust distributed applications for the Internet, such as distributed storage systems, application-level multicast, and content-based full-text search [26]. A Distributed Hash Table (DHT) abstraction implemented by these structured P2P overlays shields many difficult issues, including fault tolerance, locating objects, scalability, availability, load balancing, and incremental deployment from the distributed application designers [26]. Due to its support for many properties common to distributed applications, a DHT abstraction, if deployed in MANETs, could similarly shield many complexities in constructing distributed applications from the application designers. For example, applications such as file sharing and resource discovery could benefit from the distributed insert/lookup convergence provided by DHTs [27].

The various design approaches for applying peer-to-peer protocols to MANETs can be classified based on the nature of the P2P overlays (structured versus unstructured). In addition, because nodes in MANETs are end hosts as well as routers and all nodes in MANETs are effectively involved in supporting P2P overlay abstractions, the P2P overlay abstraction in MANETs has the option of being implemented either at the network layer or at the application layer. If a protocol is layered over an existing routing protocol for MANETs, it is classified as a *layered design.* If a protocol is integrated with a MANET routing protocol at the network layer, we classify it as an *integrated design (Cross-layered)*. The layered design allows P2P application developed in the Internet to be easily ported to MANETs. Furthermore, it decouples functionalities of the application layer (P2P application) and the network layer (routing), which enables independent development of protocols at the two layers. However, MANETs are a limited resource environment where the performance can be more important than portability and separation of functionalities. In fact, a multitude of cross-layer techniques have been proposed for MANETs for this reason. Thus, for supporting P2P overlay abstractions in MANETs, it is important to study the potential benefits of cross-layering, for example, in an integrated design.

Both structured and non-structured overlays have been implemented in MANET. A cross-layered approach is implemented for the designing of the P2P protocols. P2P applications can perform both at the network layer and at the application layer because they essentially act as clients and servers at the same time. P2P designs can be classified into four designs:

1. *Non-structured and layered design.* In [21] Gnutella is tested for its performance while deploying it on top of ad hoc routing protocols DSR [22], AODV [23] and DSDV [24]. The research showed that the ratio of delivered packets is lower than those of unicast applications deployed over MANET. This is due to the fact that Gnutella chooses neighbors independently of their locations. The overlay construction is inferred to be topology independent.
2. *Non-structured and cross-layer design.* The research done in [25] clearly depicts the results from the integration of non-structured P2P protocol Gnutella in the network

layer. They propose a system that establishes connections on demand through the routing mechanism, performs well in case of data transfer.

3. *Structured and layered design.* A proximity-conscious DHT (Pastry) has been deployed over the DSR routing protocol in [26]. As it is a layered design, there is no interaction between the DHT and the routing protocol. This leads to an overhead in maintaining routes for both the application layer and the network routing layer.
4. *Structured and cross-layer design.* In [27] the research group developed a P2P protocol called Ekta. They designed the protocol by implementing the cross-layered approach together with the underlying routing protocol. They used the concepts of Pastry DHT in the design criteria. Their results show that Ekta is better than the layered design in terms of number of successfully delivered packets.

In [29] the implementation of a typical BitTorrent protocol over the existing network is done. The most relevant work done on BitTorrent is by [30], the research aims at the design of an architecture that allows both the lookup and sharing of digital content over a wireless Ad Hoc network. The architecture will provide our project with a set of primitive concepts that can be used as a basic platform and can be extended with features that would support the building of an efficient Social P2P application where content can be exchanged anonymously. They focus on the problem of file sharing, as a solution for this problem. They proposed the two following main application enhancements to the current BitTorrent Protocol:

- BitHoc Tracker: It is an implementation of a BitTorrent tracker-less solution (membership management solution) for MANET nodes using the Windows mobile 6 operating system.
- BitHoc Client: It is an Implementation of an adapted version of the BitTorrent protocol for mobile ad hoc Networks. They developed it specifically for Windows Mobile 6 operating system. To ensure content sharing, this service decides in distributed manner of the structure of the overlay of data exchanges and the scheduling of the sending of pieces of data among devices. This service is based on the classical BitTorrent file sharing protocol which they adapted to the constraints of the wireless ad hoc network.

Their work is very important for the implementation of P2P in our CAhN project.

The proposed solution for CAhN, will incorporate the principles of the BitTorrent in creation and maintaining of the P2P network to help facilitate the content look-up and discovery in the network. Basically BitTorrent is implemented currently on the Internet. It’s one of the fastest P2P protocols designed for the Internet. Most of the sharing methods available currently are Direct Connect types. Direct Connect means each person has to be connected directly and solely to one another in order to share files. The drawback of this is that the person receiving the data is limited by the upload connection speed of the person sending it. What BitTorrent does is to remove this limitation by allowing virtually unlimited number of people to connect to one another and share the same file at the same time.

4.2 Summary

All of these technologies have in common that anonymity is not guaranteed as these protocols rely on IP connectivity and it is the IP address that reveals the end-user's identity and links the user to content that is being exchanged over the Internet. In addition, client/server and P2P based anonymity require a trusted man in the middle as a broker or a well-known trusted supplier of a required functionality or content (TOR, I2P or proxy server). Ad hoc networking technology may not require exchange of ones identity, but it has the inherent issues that content exchange can be slow as ad hoc networks are not global networks. This leads to long convergence times and requires delay tolerant transport mechanisms to achieve transfer of content.

P2P allows the development of protocols to address some of the challenges of peer networks such as security, anonymity, scalability, fairness and robustness in the networks. A major drawback for the gnutella protocol is that searching is quite expensive, with the exception of BitTorrent network that use a hybrid structure together with DHT, the search mechanisms provided in the Gnutella networks are not efficient for CAhN because network resources, like bandwidth, are wasted through the flooding of query message in the network. Since one of the major scarce resources in ad hoc environment is bandwidth, deploying Gnuteella in CAhN will throttle the whole network performance. Also the protocols of Gnutella define no standard way for new nodes to join the network. This is an issue especially for CAhN because the current methods used to identify nodes to connect to are not secure and go against the principle of preserving peer anonymity.

Chapter Five

Deployment of Local CAhN Simulator

This chapter describes how the different CAhN components that are used for the local infrastructure-less network are simulated. The basic features of the simulation tool are also explained along with the deployment and adaptations made to the BitTorrent protocol in the local CAhN.

5.1 Network Simulator 2 (NS2)

Network Simulator (Version 2), widely known as NS2, is an event driven simulation tool that has been proven useful in studying the dynamic nature of communication networks [31]. NS2 is open-source software and free to use for all users. Due to its open source nature, a user can modify parameters at different layers, create his or her own applications, and develop new protocol. The freeware nature of NS2 makes it very attractive to students and network researchers and ideal for study and research purpose. NS2 provides substantial support for simulation of transport, routing, and multicast protocols over wired and wireless networks.

Figure 6 shows the basic architecture of NS2. NS2 provides users with an executable command, which take input arguments. One of these arguments is the name of a Tcl simulation scripting file. In most cases, a simulation trace file is created, and is used to extract results from the simulation in the form of values, graphs and/or animations. NS2 consists of two key languages: C/C++ and Object-oriented Tool Command Language (OTcl). C/C++ is used for high-fidelity protocol simulations which require high-speed system programming for byte manipulation, packet processing, and algorithm implementation.
OTcl is used for:

- Describing the simulation setup.
- Defining and setting values as input parameters for the simulation.
- Define the output parameters that are needed for the determination of key performance indicators.
- Program the simulator to execute, e.g. multiple simulation runs.

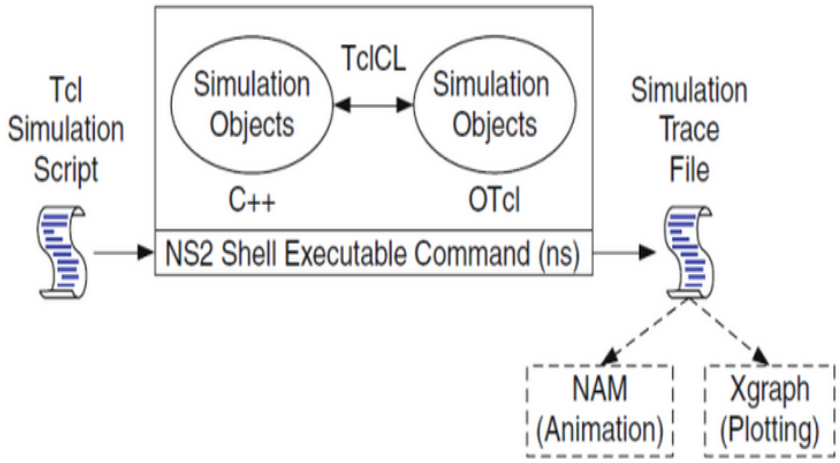

Figure 6: NS2 Architecture [26]

Figure 6 shows that the C++ and OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, we can extract a relevant subset of text-based data and transform it into a more conceivable presentation.

5.2 BitTorrent

The protocol created in 2002 by Bram Cohen and was intended as an efficient and scalable alternative to current methods of file distribution in today's P2P networks. One of the main problems in P2P systems like Gnutella [21] is that they do not scale well when downloading large files. A typical connection has high download and significantly lower upload rate, so in the traditional method when downloading a file the source peer provides the download bandwidth among the requester of the file.

This approach is neither scalable nor efficient and ensuring fairness in the system would result in poor download rates for all down-loaders. The BitTorrent protocol solution to ensure fairness in P2P was to split the files into little pieces of same length which are then traded between peers in a *tit-for-tat* manner [10]. File pieces are downloaded in an unordered fashion from the source so that peers can exploit the use of peer connection to other down-loaders to obtain file pieces they do not have. This novel approach in the BitTorrent protocol achieves high efficiency by redistributing the cost of file download among the peers in the network and helps to prevent parasitic behavior in the network [10]; using *tit-for-tat* method for trading pieces, peers with high up-load to other peers likely will download faster.

5.2.1 Terminology

A P2P system is a system that uses P2P technology to provide a set of services; this group of services forms together an application such as file sharing. A peer is the participant in a P2P system that contributes to or uses the system's resources and services. A peer is completely disconnected until it joins the system, and is active until it leaves the system. A swarm is a group of peers, from all the peers in a P2P system, that interact with each other for a specific goal, such as transferring a file. A swarm starts being active when the first peer joins that swarm, and ends its activity when its last peer leaves. The lifetime of a swarm is the period between the start and the end of the swarm. A community is the group of peers who are or can easily become aware of the existence of each other's swarms. The view on P2P systems considers three levels of operation. A P2P system includes at least a peer level, but may also include any of the community and swarm levels.

All the peers may get stuck with an incomplete file, since no one in the swarm has the missing pieces. When this happens, a peer with a complete file (a seed) must connect to the swarm so that those missing pieces can be transferred. This is called reseeding. Usually a request for a reseed comes with an implicit promise that the requester will leave the client open for some time period after finishing (to add longevity to the torrent) in return for the reseeding the file. Choked, is a term used in the description of the BitTorrent protocol. It refers to the state of an uploader, i.e. the thread that sends data to another peer. When a connection is choked, it means that the transmitter doesn't currently want to send anything on that link. A BitTorrent client signals that it's choked to other clients for a number of reasons, but the most common reason is that the client has reached the maximum of active simultaneous uploads allowed. Upload requests not served are marked as choked.

5.2.2 Concept

Peers download files by participating in a file swarm which is an ad hoc network of peers that are temporarily assembled to distribute a particular resource. BitTorrent relies heavily on global components to allow nodes to locate resources and also find each in the network. The centralized components in a BitTorrent P2P network include torrent servers, web search engines and the tracker server [10]. Communication between the peers and global component of BitTorrent P2P are passed through simple HTTP request/response messages. Trackers in the BitTorrent system play a central role in the downloading of files in the swarm. They are responsible to help downloaders locate one another in the system by keeping track of the seeders and down-loaders of a file. Peers wishing to participate in the swarm for a file must first obtain the '.torrent' file to find out which tracker holds tracking information about the file and other important information about the file. Participants in a file swarm use the tracker as an intermediary to gather download statistics; peers report to the tracker information about what pieces it has, its download & upload rate, its IP address and the port the BitTorrent client application is running on.

Each peer polls the tracker for information about other peers in the file swarm to find suitable peers to trade the pieces. Trackers respond to each request with a random list of peers that are participating in the swarm. This property allows for robustness in the file swarm by ensuring that peers with fast and slow connections have equal opportunity to maximize their download rate. For file downloads to be successful in BitTorrent, initially at least one peer (seeder) with the complete file must participate in the swarm until the complete set of file pieces has been fully disseminated in the network. As a result

this allows BitTorrent to efficiently handle requests for large files. In the case of a large file, the seeder must stay online long enough until at least one peer has successfully downloaded the file in the network. Eventually, as more peers in the system complete their download and remain as seeders in the network, request for file start to complete quickly since each new peer participating in a file swarm adds additional upload bandwidth.

5.2.3 BitTorrent Deployment in NS-2 for the local CAhN

The concept for the implementation of the BitTorrent is illustrated in Figure 7. The figure shows the block diagram adopted for CAhN from [30]. The BitTorrent used is modified to be a trackerless P2P protocol, which makes it ideal to implement it in the local infrastructure-less network that is part of the CAhN concept. In this implementation the tracker system is separated from the actual data transmission. The tracker system in the CAhN will be used to processes the signaling received form all the peers in local CAhN and process the control signals to provide feedback to the peers. The feedback will comprise the system statistics and instruction to maintain optimal system performance. The community members that use the P2P protocol can be managed and monitored in a flexible way, while at the same time it will be easier to include the anonymity aspect of the CAhN. For the purpose of simulation the trackerless BitTorrent protocol is deployed on top of the DSDV routing protocol. The routing table stores the routing information for the peers. The *"Plug-ins"* is any code extension that can be plugged to the main core code to enable other services.

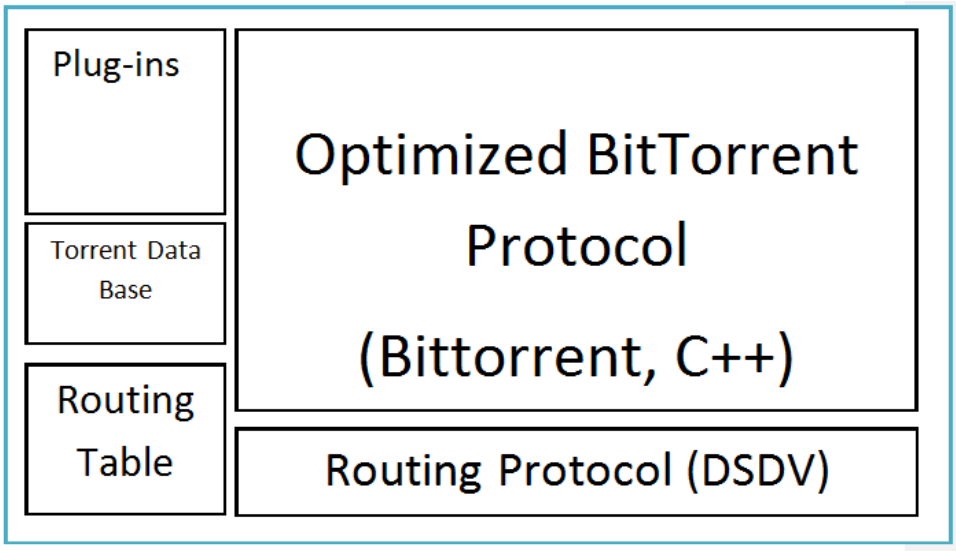

Figure 7: Block Diagram for BitTorrent Implementation

The different BitTorrent functionality is implemented with C++ in NS2 modules. The BitTorrent model has many applications in remote collaboration and communication networks but for the purpose of this work only the BitTorrent protocol that allows peers to share and access resources is considered. For each system examined the thesis will focus on the underlying principles and examine how sharing of resources is achieved using this protocol. The deployment will include four modules. The list of the C++ modules which implements the optimized BitTorrent functionalities are described as follows:

- ***Bitagent.*** This module contains the implementation of the BitTorrent agent class and structure. The Agent class will transmit the control message by using UDP protocol.
- **Bittable.** Thise module contains the implementation of several data structure classes (BitPeer, BitNeighbor, PeerTable, NeighborTable, Piece, PiecePool). BitPeer represents the peer object of the BitTorrent node. BitNeighbor represents the neighbor peers of the BitTorrent node. PeerTable contains the peer lists of the BitTorrent node by using Tcl_HashTable, whose key is the id_ of the peer object. NeighborTable contains the neighbor lists of the BitTorrent node by using Tcl_HashTable, whose key is the id_ of the neighbor object. PeerTable/NeighborTable should be initiated with the maximum number of object which it can contain. Piece represents the piece object of the file which will be shared by BitTorrent nodes. A file consists of several pieces and a piece in turn consists of several blocks. PiecePool contains all the pieces of the file by using Tcl_HashTable whose key is the id_ of the piece object, and maintains the status of the piece (i.e. COMPLETE, DOWNLOADING, or INCOMPLETE).
- **Bittcpapp**. This module contains the implementation of BitTorrent TCP functionality class, BitTorrent Control class, and BitTorrent Payload class. It handles the TCP connection establishement between peers. BitTorrent TCP class is inherited from the TCP application module which is the famous application to use full TCP protocol in NS-2. In order to establish the TCP connection between two BitTorrent nodes, BitTorrent TCP class should be attached to each of the wireless nodes. Because the TCP connections will be established dynamically between every node in the community, this TCP connection is maintained by BitTorrent Connection object. BitTorrent Control class maintains the control packet of BitTorrent functionality implemented. BitTorrentPayload represents the packet header and the payload of

data transmission. Because NS-2 doesn't support the transmission of application payload, the CBuf and CBufList of TCP applications in NS-2 is used to capsulize the data transmission to BitTorrent TCP class or BitTorrent object [31].

- **Bittorrent**. This module implements the main functionality of the trackerless BitTorrent protocol. This module contains the implementation of the main BitTorrent content sharing algorithm, BitTorrent Connection, Global BitTorrent List, and Global BitTorrent Connection List. BiTorrent Connection maintains the dynamic TCP connection between two BitTorrent (BitTorrent TCP Applications) nodes. Global BitTorrent List maintains the global list of BitTorrent nodes in the simulation. Global BitTorrent Connection List maintains the global list of BitTorrent Connection peers in the simulation. It is only for implementation details, never related with the BitTorrent protocol.

5.3 Routing Protocol

As described in section 3.2, there are different existing routing protocols that can be used together with the BitTorrent protocol. The available ad hoc routing protocols can be classified as proactive, on-demand (reactive) and location-based routing protocols depending on their application. For the simulation purpose it is proposed to use a pro-active routing protocol, specifically DSDV routing protocol. It is reasoned that since the ad hoc network in the simulation will have relatively small number of peers to capture the behaviour of the local CAhN and the DSDV performs best in networks with moderate mobility scheme, few nodes and many data sessions involved [24]. The route informations are maintained through periodic and event triggered routing table exchanges. DSDV also implements incremental dumps and settling time, which are used to reduce route request latency.

DSDV is not efficient for ad hoc networks involving larger number of peers. Since nodes need to maintain a complete list of routes which overloads the individual peer's performance. For practical cases it better to implement on-demand routing protocols. In [33], the performance of multi-hop wireless ad hoc network routing protocols were measured with respect to metrics like Packet Delivery Fraction, End to End Delay and Routing Overhead in three different scenarios: pause time, number of nodes and node speed. The results indicated that the performance of on-demand is superior to regular DSDV. It is also observed that the performance is better especially when the number of nodes in the network is higher. When the number of nodes is increased, the performance of proactive (DSDV) degenerated due to the fact that a lot of control packets are generated. The reason for the performance to drop is due to varying source and destination nodes and placement barrier in network topology. The simulation also revealed that DSDV consumes more computation overhead rather than on-demand routing protocols in the presence of high mobility, yielding inferior performance.

For fair conditions there are some previous studies (such as [34], [35]) that analyze routing protocols in mobile environments. In this study, they have compared proactive (DSDV) and reactive (DSR) routing protocols for fair conditions and also evaluated DSDV and DSR behavior in unfair conditions. Because mobility is an important issue for comparison of routing protocols, they have designed the topology with different pause times that shows the degree of mobility in the network. It is concluded that DSR shows better drop rate and delay characteristics than DSDV, because network changes are updated dynamically in DSR. Whereas DSR routing table is updated when there is any change in a network, DSDV routing table is updated in a period of time. If there is any change in the network between periods, nodes recognize any change just in periodic update time. It is shown that DSR

achieve better performance for drop rate and delay when network is characterized by data rates compare to DSDV.

For ad hoc routing in the local CAhN, DSDV proactive protocol that provides each node with the list of shortest path routes to all other nodes in the network, is used in the NS-2 simulation. This is because in the simulation involving a small number of peers, a DSDV routing protocol is easy and efficient. All the peers will have routing information of the whole community. Each peer periodically transmits updates including its own sequence number and routing table updates. Peers also send routing table updates for important link changes.

Chapter Six

Simulation Set-up and Procedure

6.1 Simulation Structure

This chapter describes the simulation structure followed to generate the intended statistics. Tcl scripts and C++ modules have been used for this research to generate the simulation results. This section of the report concentrates on explaining these processes involved. Figure 8 shows the experimental procedure used to perform the simulation.

6.1.1 Expectations from the simulation

Based on the implementation of the simulation the following questions are expected to be answered:

- If the scope of sharing is limited, is the downloading opportunity between peers limited?
- Is there fair sharing among the peers in this case?
- Is there piece size diversity for the file in the community? What are its impacts on the performance of the BitTorrent protocol?

The simulation will indicate the most important control parameters that can be fedback to the peers from the Tracking System, in order to implement controlling mechanism in the community.

6.1.2 Evaluation Methodology

The evaluation methodology will be based on simulation with NS2 result. The scalability of the BitTorrent protocol under different settings will be considered. For instance,

- Effect of peer neighbour degree or neighborhood scope
- Effect of piece-block size in the BitTorrent protocol

6.1.3 Experimental Procedure Involved in the simulation

Experimental process steps involved in simulation are shown in Figure 8. The steps are described as follows:

- *Problem definition.* It is to clearly pin point the problem at hand. The aim of this simulation is to see how BitTorrent perform in filesharing in the local community with respect to the defined performance metrics.
- *Simulation Model.* The model contains the ad hoc routing and BitTorrent protocols implemented with C++ modules in NS-2. Sources, sinks and communication agent interfaces defined via OTcl which controls C++ through Tcl linkage established at the Tcl script.
- *Extend Simulator.* It is where necessary changes and manipulations are made to meet the simulation specific purpose, i.e. performance analysis of the Local CAhN.
- *Execute simulation.* It involves the invocation of the discrete event simulator routines. The simulator maintains an event list (packet list), by providing the required input parameters when triggered by the simulation processes.
- *Post-process results.* It outputs a *trace.tr* file after the simulation is finished. This file contains information concerning the performed simulation. An additional task is needed to extract the required result and describe it in a graphical or statistical fashion.

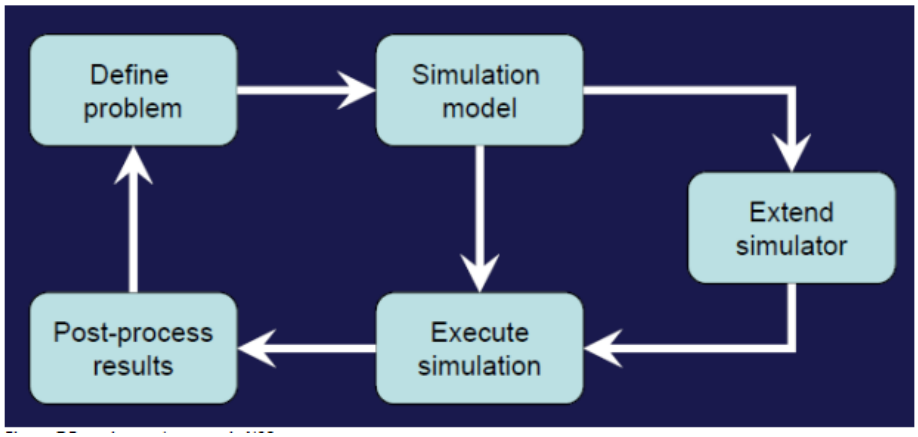

Figure 8: Experimental procedures in NS2

6.2 Definition of Performance metrics

To evaluate the performance of the BitTorrent protocol, three metrics are used for analyzing our simulation results.

- Throughput- is the average rate of successful file delivery over the communication channel. Measures only data packets received at the application layer. It is given by the ratio of the total data received to total data sent in the peer-to-peer system.

$$\text{Throughput} = \frac{Total\ Bytes\ Downloaded}{End\_time - Start\_time} \quad [\text{bps}]$$

- Downloading Ratio DR(*i)* is a function of the peer ID, and equals the ratio of total number of application layer packets received by each peer to the sum of application layer packets received by all the peers. It is interesting to investigate how the peers in the community contribute to the file sharing processes. DR(*i*) captures this aspect of the community network by measuring the ratio of data downloaded among peers.

$$DR(i) = \frac{r(i)}{\sum_{i=0}^{N-1} r(i)}$$

Where: *i= i*th *the peer's id*
N= the total peers in the community
r[i]= the data received by peer-i

- Download finish time- indicates how soon it will take for each peer to download the required files from the community members. When sharing a data in the community, it is intersting to measure the time it takes to completely download the data from the respective peers.

6.3 Simulation parameters

Configurable parameters of Tcl script are set as indicated in Table 3. Concerning the underlying layers Table 4, the nodes connect to each other using the IEEE 802.11 MAC Layer with the RTS/CTS-Data/ACK mechanism enabled. Table 5, for ad hoc routing the DSDV proactive protocol that provides each node with the list of shortest path routes to all other nodes in the network is used. The parameter for the routing agent is the ttl, which will be used for the implementation of the neighborhood control. Table 3, shows the specific parameters set to test the simulation performance for the neighborhood degree, and for the piece size distribution.

For the simulation to produce the intended result mentioned in section 6.1.1, the necessary parameters should be set up. Identifying the correct simulation parameters is a key for a successful and nearly realistic analysis of the study. The general parameters specific to BitTorrent protocol are described here. To run the simulation in NS-2, all data structures and treatments have been added to simulate the functionalities of BitTorrent in C++. In Tcl script all the BitTorrent protocol parameters are defined and initialized [30]. Some of the parameters are:

- *max_upload_num_:* The maximum number of active uploads connections per peer. It is also the number of parallel choking slots.
- *choking_period_*: The period of the choking algorithm. It is the interval of time during which a peer tries to offer pieces to a set of effective neighbors.
- *choking_best_neighbors_num_*: The number of neighbors chosen as effective neighbors during a choking period because they are best uploaders.

- *received_bytes_reset_interval_*: In the choking algorithm, each peer chooses its best neighbors with whom to reciprocate data based on how many bytes they send to it in some specific interval.
- *flooding_ttl_*: The maximum number of hops that will be used while flooding the network with HELLO messages.
- *piece_num_*: The number of pieces in the file to be shared.
- *block_num_*: The number of blocks in one piece.
- *block_size_*: The size of a block in bytes. The size of the shared file is equal to piece_num_ * block_num_* block_size_.

Table 3: Simulation parameters for Configurable parameters of Tcl script

Parameter	Comments	Value	Default value
max_upload_num	*The maximum number of upload connections that a peer can have*	Default [1, 2, 3, 4]	4
piece_num	*The number of pieces in the file which will be shared*	Depends on simulation [1, 10, 100...s]	10
block_num	*The number of blocks in one piece*	Depends on simulation [1, 10, 100...r]	100
block_size	*The size of a block. The unit is bytes.*	20*100 bytes	20*100 bytes
number_of_nodes_in_simulation_	The number of peers in the community	Depends on simulation	Depends on simulation
try_upload_interval_	*The interval to try to offer the piece to its neighbors*	Default	*40 Seconds*
max_neighbor_num_	*The maximum number of neighbors that node can have*	Depends on simulation [1, 2...10]	10
flooding_ttl_	*The time-to-live value which will be used in flooding of HELLO message*	Depends on simulation [1, 2, 3...the max Number of hops]	3
received_bytes_reset_interval_	*Interval to reset the history about how many bytes a peer receives from a specific node*	Default [60, 80...] bytes	60 bytes
choking_best_slot_num_	*The number of best slot for choking algorithm*	Default	3

Table 4: Radio power or energy consumption

Parameters	**Comments**	**Value**
rxPower	Receive Power	100mW
txPower	Transmision Power	100mW
CSThresh_	Carrier Sensing Range	70m
RXThresh_	Transmission Range	50m

Table 5: Protocols or schemes used for implementation

Setting	**Comment**	**Scheme**
chan	Radio channel	WirelessChannel
prop	Propagation	TwoRayGround
mac	Mac layer	Mac/802_11
ant	Antenna	OmniAntenna
rp	Routing protocol	DSDV

6.4 Network topology set-up

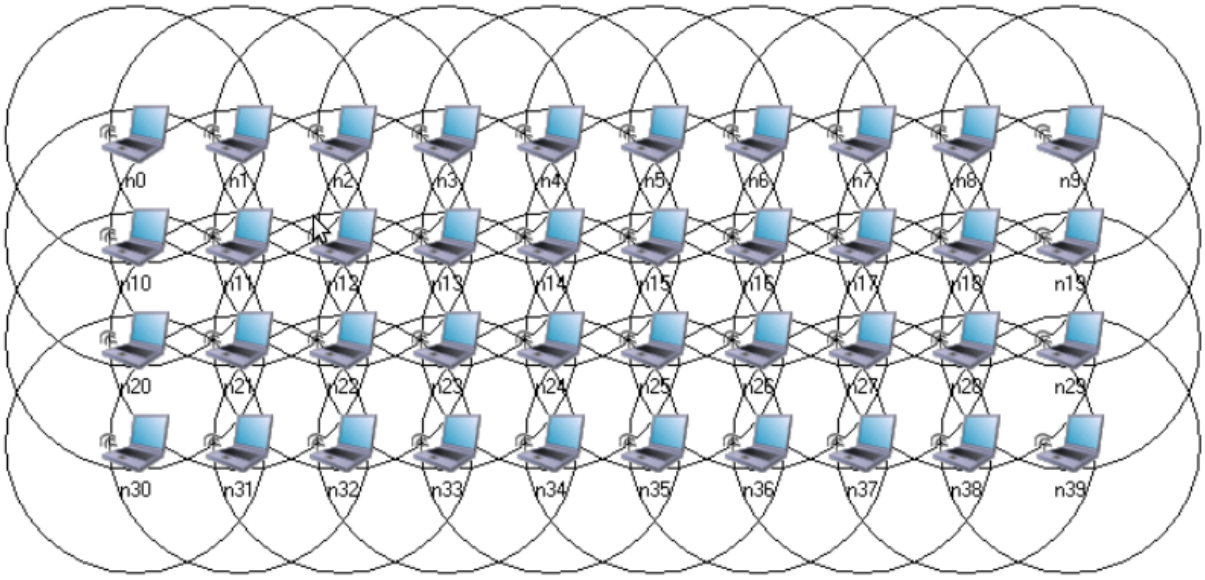

Figure 9: Network Topology [30]

A network of nn-nodes, depending on the simulation, distributed on a plane following a grid topology as shown in Figure 9 is used. The distance between two physical neighbors is set to 40 m for a range of wireless transmissions equal to 50m, Table 4. This ensures connectivity while minimizing interference in the community for the simulation. In the simulation node-0 initially holds all the data to be shared, where all the other peers will be leechers by default. At the beginning of the simulation, node-0 located at the top left is the seed and the other nodes are leechers. The file size will be set depending on the simulation parametrs for piece size and block size. All peers start downloading the file at the same time t=1500s by first looking for each other then sharing the pieces of the file according to BitTorrent protocol. This interval at the beginning gives the network enough time to prepare the routing table by flooding the network with “hello” messages unitl it reaches equilibruim state. The equilibruim state is the time it takes to stablize the network interms of peer look-up and peer discovery, before the actual data sharing starts.

Chapter Seven

Simulation Results and Analysis

7.1 Simulation results

The effect of neighborhood and piece size diversity in the community is simulated. Peers cooperating to download a file using BitTorrent form a sharing session. In order to allow a fast replication in the network, the file is split into several pieces and pieces are split into several blocks, Figure 10. The transmission unit is the block and the exchange unit is the piece: a peer cannot start uploading a piece to another peer until it holds all its blocks. A peer that owns a complete copy of the file is called seed and peers that are still downloading it are called leechers. Peers consult periodically their routing table to get an up-to-date set of neighbor peers that may include both leechers and seeds.

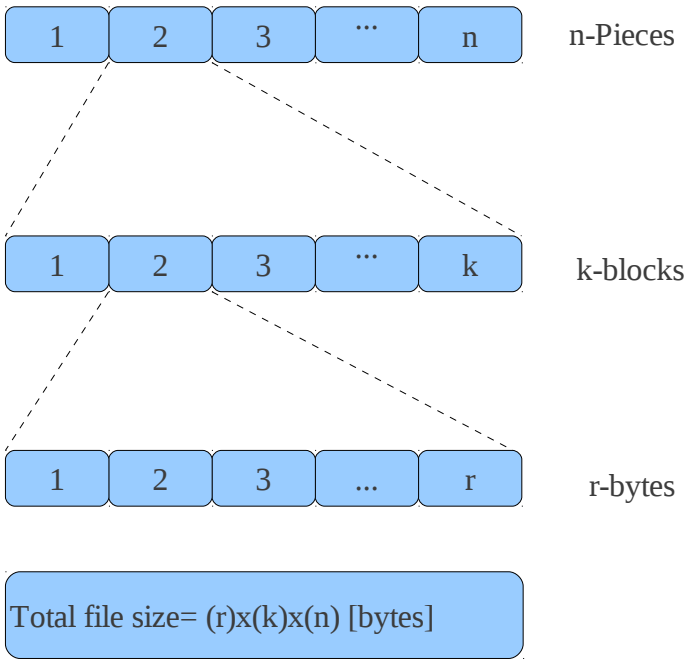

Figure 10: A file structure in BitTorrent protocol

7.1.1 Effect of neighbourhood scope

One of the factors in regular topology-unaware BitTorrent is the scope of the neighborhood. In this section, the impact of the neighbourhood scope on both the download finish time, throughput and downloading ratio is studied. Several simulations have been run on the topology changing each time the flooding scope (TTL) of HELLO messages destined to peer discovery. Figure 11 compares the finish time for TTL=5, TLL=3 and TTL=1.

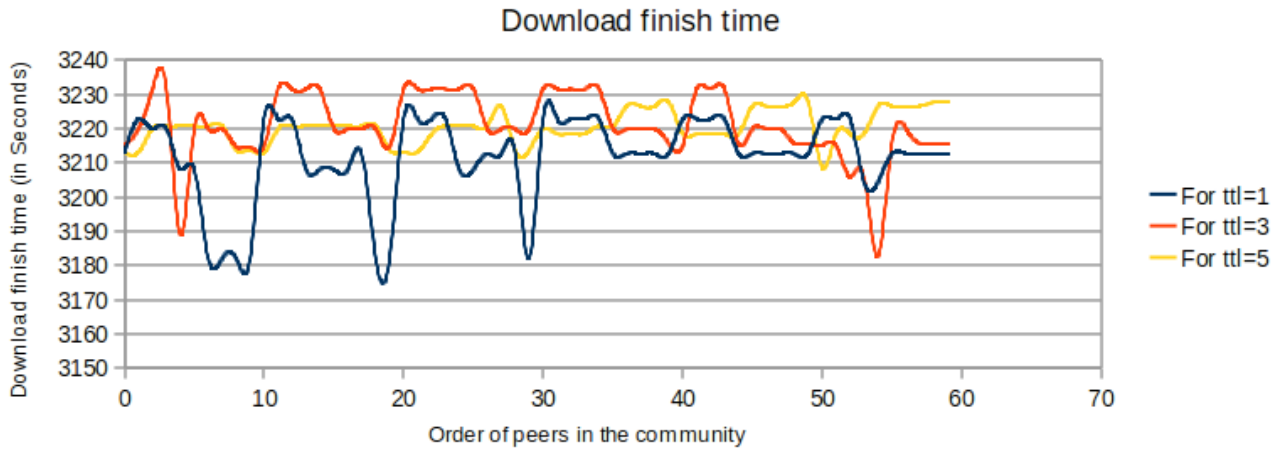

Figure 11: Download finish time for each peer in the community

Figure 11 shows the behaviour of the experiment for the peer with respect to their finish time, the three plots of the X- and Y-axis scales denoting peer orders and finish times respectively for different ttl values. As it is possible to see in figure 11, in the first few peers where the download finish time is lower for the default ttl value (ttl=5). In the next nearby peers the finish time is much lower for small ttl. The figure shows the download finish time when all the peers have finished their downloads. Each curve represents the download finish time for each peers in the local community. For ttl=1 the blue curve, the download finish time is seen to be small for most of the peers in the community. For ttl=3 the red curve, the download finish time is relatively large for most of the peers.

Figure 12 shows the throughput, in Kbps, simulation result when the performance of the community is impacted by neighborhood scope for different ttl values (ttl=1, 3, 5). The curves are plotted as Y-axis scale representing the throughput measured in Kbps and the X-axis scale represents the number of peers in the community. The throughput simulation result shows that the performance of the community can be impacted by neighborhood scope. The throughput is higher for larger scopes (ttl=3 and ttl=5) than for smaller scopes (ttl=1). On average, for larger number of community members the ttl=3 out performes the other scope ranges. 100% of the peers showed to have a throughput of less than 6Kbps for ttl=1. While for ttl=3 and ttl=5, all peers have a throughput of above 6Kbps.

The normalized Downloading Ratio for each peer in the community is compared on Figure 13. The Y-axis indicates the percentage of downloading ratio in the community for all the peers. The X-axis shows the order of the peers in the community, ordered according to their increasing value of the number of hops from the seed. The simulation is made for different values of the ttl. The first few peers show that they have less downloading ratio compared to that of the far peers from the seed. For smaller scope, i.e. ttl=1, the downloading ratio is lower than the ttl=3 and ttl=5. The downloading ratio is higher for ttl=5 for the peers near to the seed. The plot shows that as the peer's distance increases from seed the downloading ratio will be much higher for a community using ttl=5.

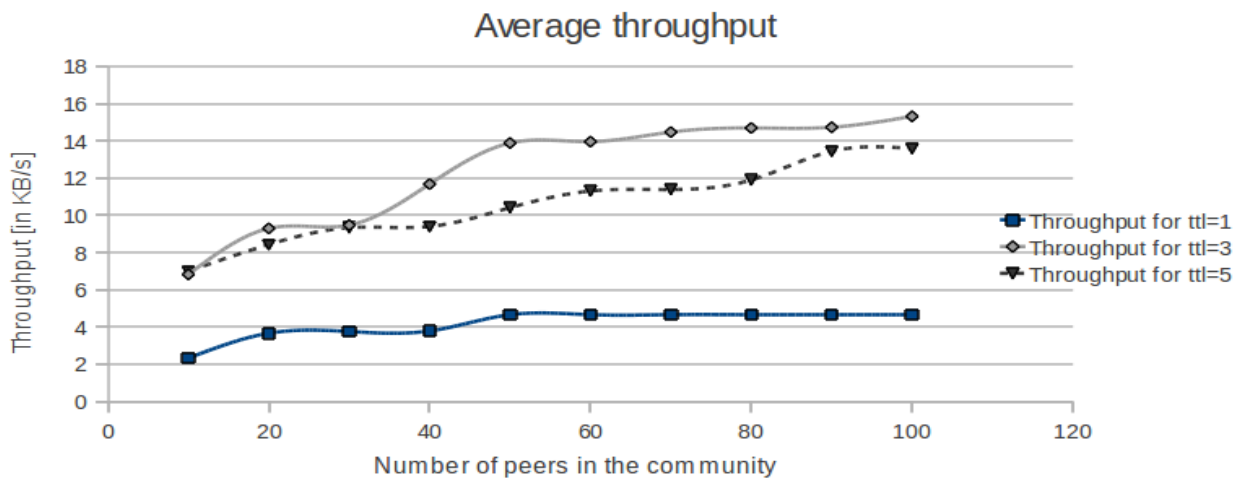

Figure 12: Average Throughput for different ttl cases

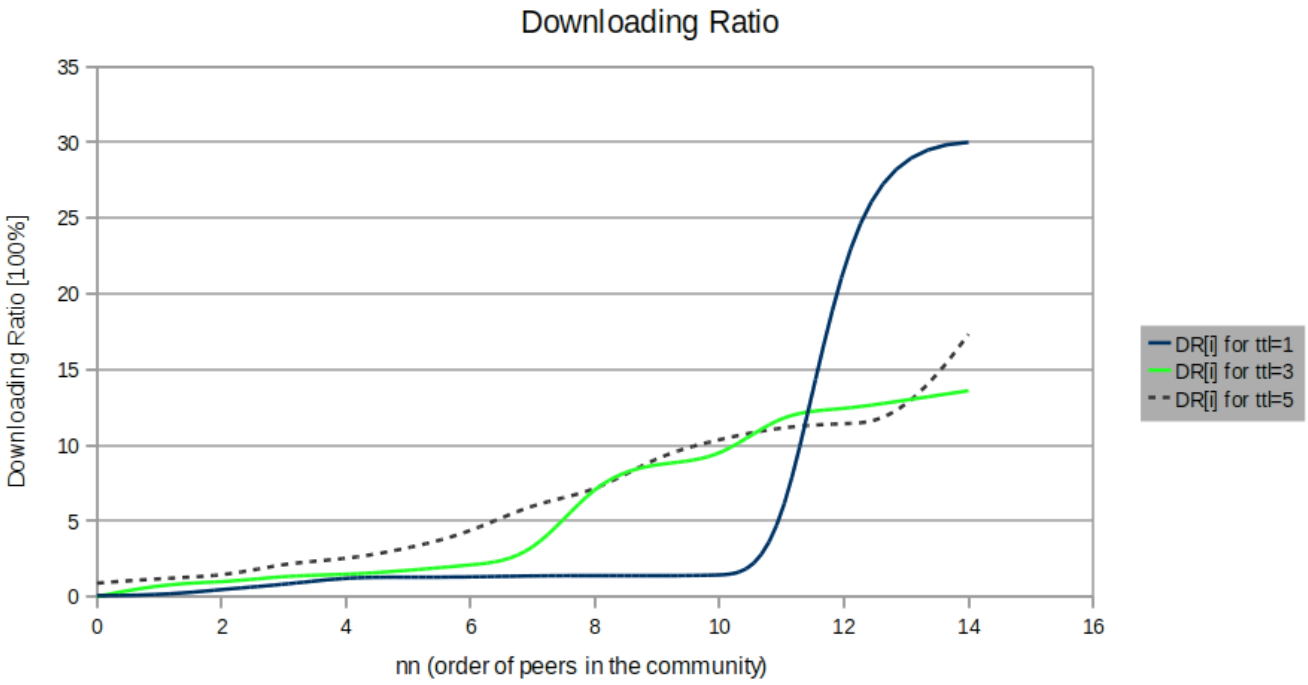

Figure 13: Downloading Ratio for the peers in the community

7.1.2 Effect of file size

In this section the piece diversity in the community is considered and its impact on the finish time of peers, throughput, and downloading ratio is presented. The TTL of HELLO messages is set to its maximum value so that all peers are neighbors of each others. Three sizes of pieces are used whereas the size of the file is left constant, see Figure 10 for the file structure.

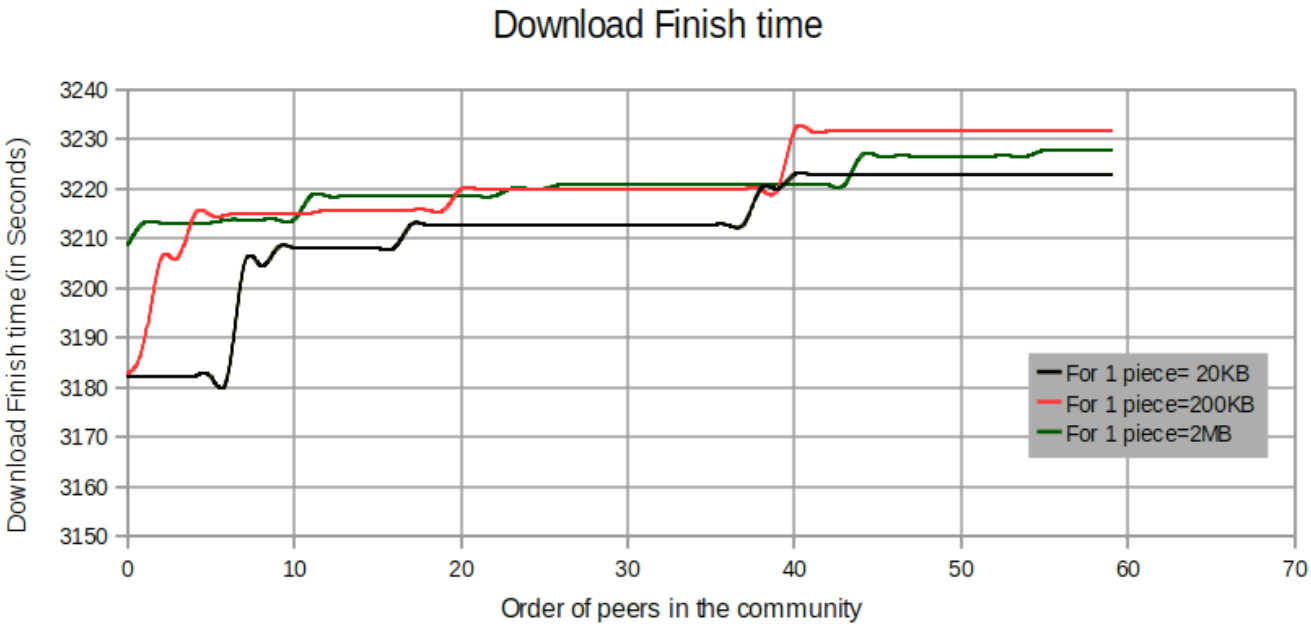

Figure 14: Download finish time for different piece sizes

Figure 14 plots the download finish time as a function of the number of peers in the community for small and large pieces. The Y-axis scale indicates the download finish time for each peer in the community. While the X-axis scale in this figure represents each peers located at certain number of hops from the seed by setting the TTL to default value. In this experiment, peers are started with the same time delays. The total amount of peers started is 60; the size of the downloaded content is shown with different piece sizes. The first set consists of 20KB piece, and it is increased to 2MB piece size. It is expected in this case to have all the peers downloading from the seed, since only seed has the data at the beginning of the sharing session. Since the peers start sharing data after a small delay i.e. until they reach equilibrium state, the peers near to the seed will have a better finish time than the farthest peers. The figure shows that in all the three cases the download finish time is lower for files comprising piece sizes of 20KB. With the exception that for the farthest nodes the peers showed to have a lower finish time for a larger piece size, i.e. for 2MB, than the 200KB size.

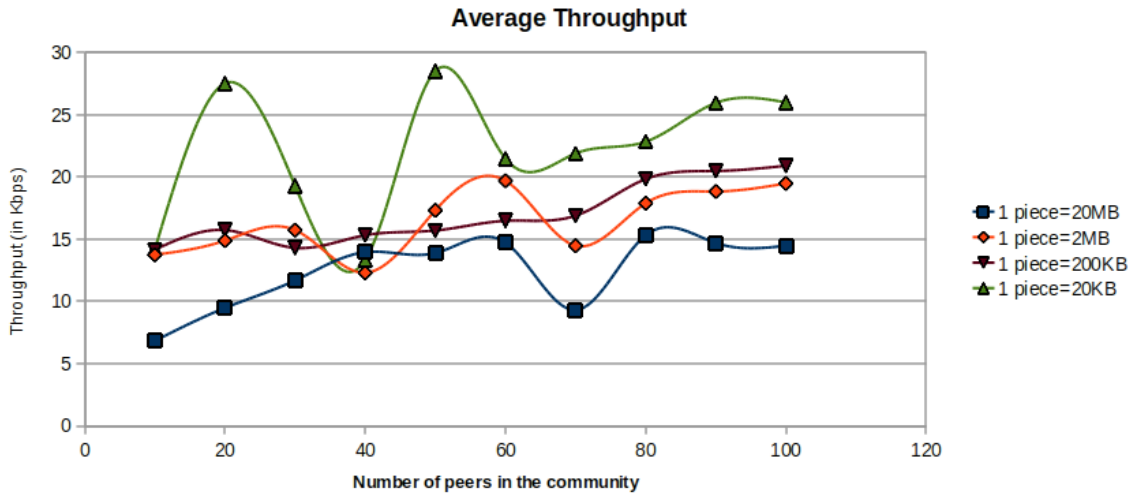

Figure 15: Average throughput

Figure 15, shows the impact of the piece diversity on the average throughput of the community. The throughput is measured in Kbps. The Y-axis scale is the throughput and the X-axis scale is the order of the peers in the community. It can be seen from the figure that for small piece size the community performs well, by indicating high throughput. As piece size increases the throughput is observed to decrease proportionately. Especially, for piece size 20MB the throughput is gets lower as the peer's distance from the seed increases. For almost 100% of the nodes in the community the throughput is large when the piece size is set to 20KB. Specially, if a node is far away from the seed the throughput will be higher if the file size is set to be low.

Figure 16 presents the downloading ratio for all nodes in the network for small and large pieces. Nodes are ranked and numbered as a function of their distance from the seed. Figure 16 displays the results concerning the swarm used in the community. The red curve, representing the downloading ratio of peers using 200KB piece size in the swarm, plunges over for the peers moving far away from the seed. Figure 16 displays the results of the experiment concerning the number of pieces used for sharing data and its downloading ratio. The Y-axis scale is the downloading ratio in percentage while the X-axis scale is the peers ordered in their increasing number of hops from the seed. It can be seen from the figure that the curves for the three piece sizes followed the same trend. The curves show a constant decrease in downloading ratio as the peer's number of hops increases.

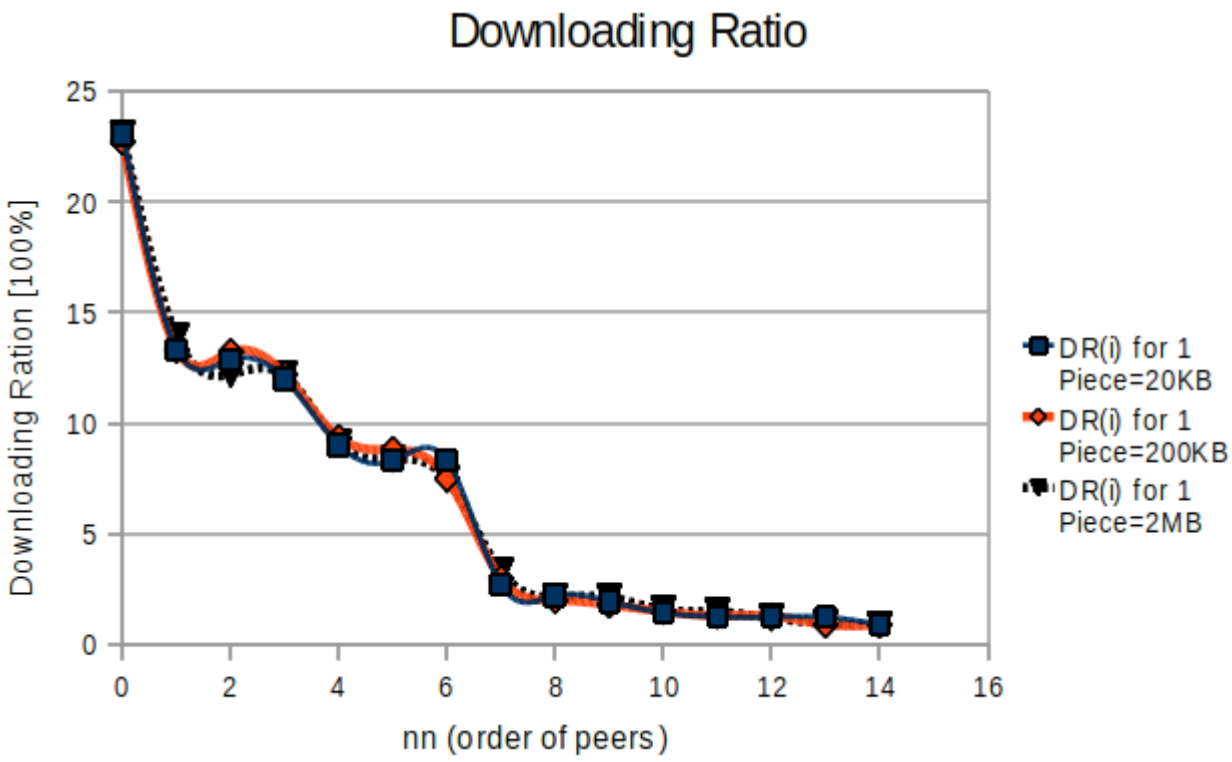

Figure 16: Downloading Ratio for different piece size

7.2 Simulation result analyses

The simulation result analysis starts by first considering the effect of neighborhood scope and continues to investigate piece size diversity in the community. The routing overhead is expected to be maximum since on one hand the volume of exchanged data is large [29]. On the other hand any packet sent over multiple hops will unfairly consume bandwidth from intermediate nodes which are also part of the community.

The performance of peer-to-peer file replication comes from its piece and peer selection strategies [30]. Two such strategies have been introduced by the BitTorrent protocol: the rarest first and choke algorithms. Piece selection strategy is handled by the '*rarest first*' algorithm in the BitTorrent protocol stack. While selecting neighbors to upload data is performed by the 'chocke' algorithm. The aim of a piece selection strategy is to guarantee that each peer is always interested in any other peer. The rational is that each time the peer selection strategy unchokes a peer; this peer must be interested in the unchoking peer. This way, the peer selection strategy can reach the optimal system capacity. In choke algorithm at most 4 remote peers can be unchoked and interested at the same time.

7.2.1 Effect of neighbourhood scope

Multi-hop paths are more subject to failures because of the mobility. Hence, sending blocks to distant peers consumes time and bandwidth from close peers without considerably improving sharing opportunities [30]. Interestingly, Figure 11 shows the finish time decreases when the neighborhood

scope is decreased. This is mainly due to better TCP performance over short paths and to smaller routing overhead. Control packets, namely PIECE UPDATE and HELLO packets, are sent only in the restricted neighborhood. Case TTL=5 is better than the case TTL=1 because of the interference between physical neighbors.

Figure 12, the throughput simulation result shows that the performance of the community can be impacted by neighborhood scope. The throughput is relatively higher for larger scopes (ttl=3 and ttl=5) than for smaller scopes (ttl=1). On average, for larger number of community members the ttl=3 out-performs the other scope ranges. Figure 13 plots the downloading ratio DR(i) as a function of the number of peers for the different values of TTL. Pieces are set to be small in this figure. It can be seen that the improvement in the finish time (i.e. decrease in finish time) when reducing the neighborhood comes at the expense of a lower downloading ratio. The diversity of pieces in the network decreases and the file propagates more or less as a wave in a unique direction from the seed to the farthest nodes. For small TTL, distant peers can not participate in the replication of pieces. They only wait for pieces to arrive to their physical neighbors to obtain them. Clearly, this is bad for cooperation among peers. An optimal solution is one that improves the finish time while preserving large values for the downloading ratio; which ultimately should made by the TS in the CAhN.

The attempt to reduce the network overhead by having low values for the neighborhood scope, Figure 12 and 13 show that, in fixed ad hoc networks, low values of the scope improve the download time, but yield the low values for the downloading ratio Figure 13. In fact, when the value of the neighborhood scope is small, pieces propagate in almost a unique direction from the initial seed to the farthest peers, which limits the reciprocation opportunities. The Figures show indeed that limiting the neighborhood while sending some pieces to faraway nodes results in the best performances in fixed networks in terms of download finish time and downloading ratio.

7.2.2 Effect of file size

Figure 14 plots the download finish time as a function of the number of peers in the community for small and large pieces. Each point in this figure is an average over multiple simulations and over all nodes located at the number of hops from the seed by setting the TTL to default value. As expected, the finish time increases as far as we move away from the source. One can notice in the figure that for small pieces, remote peers have better finish time than for large pieces. This is because the rate of

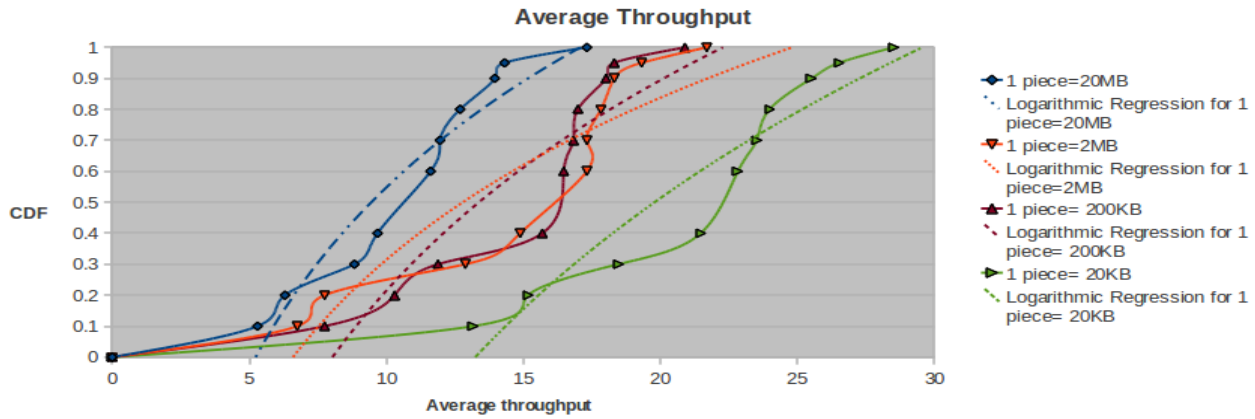

Figure 17: Average Throughput, in Kb/s, for different piece size

transmission of small pieces is larger. A remote peer can then receive more pieces in the choking period and share them with others when pieces are small. The reusability of pieces and network resources improve in this case. This is confirmed in Figure 16 where the average downloading ratio DR(i) is plotted as a function of the number of peers to the seed. Each point in this figure is also an average over multiple simulations.

It is clear that the downloading ratio for small pieces is more important because distant nodes can now quickly get complete pieces and replicate them in their neighborhood. Unfortunately, in the case downloading ratio, piece diversity alone seems to appear less influential on the downloading ratio performance. This is because only one area of the network is in activity at the same time. To better illustrate the idea, Figure 16 shows downloading ratio for all nodes in the network for small and large pieces. Nodes are ranked and numbered as a function of their distance from the seed. It can be seen that, downloading ratio for distant nodes is low and that nodes wait for pieces to arrive to their upstream nodes before obtaining them.

Figure 16 show that the BitTorrent strategy decreases the downloading ratios for distant peers. This is due to the diversity created by sending original pieces to distant nodes. Unlike the case of limited neighborhood, pieces do not propagate in the network as a wave but nodes exchange pieces in all directions. Figure 15 and 17, shows the impact of the piece diversity on the average throughput of the community. It is clear from the figure that the throughput increases for smaller piece size. Figure 17 shows the CDF for the different piece sizes. For piece size 20KB almost 100% peers in the community have good performance compared to peers performing on larger piece sizes. The experiment clearly shows that the throughput of peers in the community depends on the piece size. In real deployment of BitTorrent, peers should dynamically compute the best piece size [30].

7.3 Summary

In this chapter the performance of the BitTorrent protocol is tested for neighbourhood scope and piece size diversity. It is observed that since ad hoc nodes are mobile, it is better to keep the neighbourhood of a peer smaller, to avoid the deterioration of data transmission in the community due to the dynamism of the system. The downloading ratio has high value due to the diversity of pieces introduced by the mobility of nodes and the downloading opportunities it raises. Furthermore, the finish time is better and more equally distributed since far nodes can receive pieces from the beginning of the session and can replicate them in their close neighbourhoods. Pieces are sent in an optimal way and in a reduced number so as to limit the routing overhead. This creates parallel areas of activity in the network. Far nodes do not need to wait for pieces to arrive to their neighbourhoods to download them. Hence, pieces propagate in the network in all directions. The solution of BitTorrent, therefore, encourages a good management of neighbourhood and piece selection that reduces finish time and increases downloading. A final conclusion must be a trade-off between good finish time and good downloading ratio between peers should be made by the community members depending on the information fedback by the TS to increase their performance. This can be a statistical data feedback by the tracking system in terms of the controlling parameters especially, the ttl (for controlling the neighbourhood scope) and the piece_num and block_num (for piece diversity).

Chapter Eight

Conclusion and Future work

In achieving the research goal, "Finding an alternative communication method for data and information sharing in an anonymous way". The P2P filesharing mechanism is implemented together with ad hoc networks that is able to make use of the performance of the BitTorrent protocol while at the same time avoiding the complexities and pitfalls of currently available communication systems like client/server or pure ad hoc based networks. In addition the aim of the research is finding a mechanism that is more scalable and distributed than the currently existing technologies. Finally, it is also aimed at leveraging the work to closely experment with the performance of the proposed communication method. This chapter looks back at the findings and contemplates on how well our research goals have been achieved. This chapter is organized as follows. First the summary of the work and the findings with respect to each of the chapters are explained in Section 8.1. It will then contemplates on the findings and provide concluding remarks with respect to each of the research goals. The conclusions will be the subject of future work and on the implication of this proposed communication mechanism. This chapter ends with recommendations for future work in Section 8.2.

8.1 Conclusions

By re-iterating the research goal in order to see how much of the initial goals have been achieved. The research goal is defined as follows: finding an alternative communication method for data and information sharing in an anonymous way. The main research question is how the CAhN enhances the filesharing experience among the community in a typical use case? The research explains the issues concerning the typical use case developed. The performance of the BitTorrent protocol is investigated using the Network Simulator Version 2.34 (NS2). Where the different parameters determining the data transfer performance related to CAhN are analyzed. How to use these parameters given the loose participation of users in participating in local and perhaps very small over relay networks to arrive at instruction for

the user that optimize the transfer of files. The general scope of the CAhN project is shown in Table 6.

Table 6: Summary of the research findings

Steps	***Descriptions***	***Corresponding Chapter***
1	CAhN concept and use case development	2,3
2	Assessment of different file transfer mechanisms and selection of relevant one	4
3	NS2 deployment and performance analysis of the BitTorrent file-sharing mechanism	5,6,7

(Step 1) This research investigated the community ad hoc network and proposed its deployment, which focused on infrastructureless environments and P2P communication. In particular, peer-to-peer networks were chosen as the appropriate candidate for the implementation of community ad hoc networks. The research work also analysed the different issues related to CAhN. CAhN concept is explained. Community ad hoc networks can be used in different areas like sharing content, spreading of knowledge without concern, off-loading of operator-owned wireless networks. In this research work it is proposed to deploy CAhN as an alternative communication method for data and information sharing. The proposed system adapts to the dynamics of the peer-to-peer networks. Peers may create content based groups at any time, without the need of any centralized management. In short, it gives them the communication tool to collaborate in sharing data among the community peers. A use case describing how a mobile user can use the community ad hoc network to share a file is presented. The use-case typically considers a case where a mobile peer joins a local community from a different local community.

(Step 2) A key survey of the differences between BitTorrent and the different currently available file sharing techniques is made. The difference is that unlike the other filesharing techniques, BitTorrent based ad hoc networking is based on ad hoc peers exchanging pieces of a single file with each other. This difference is largely due to the implementation of a "*tit-for-tat*" mechanism in the BitTorrent protocol which is attributed to be one of its main success factors. It is described that the majority of the currently available techniques are tailored to the monitoring infrastructure networks and government bodies, which makes them difficult for a more guaranteed content sharing.

(Step 3) Performance test of BitTorrent protocol in the local community ad hoc networks and embedding of the mechanism in the system were performed. The NS2 simulation results for local CAhN were presented outlining the simulation methodology and simulation parameters used. The impact of the neighbourhood and piece size diversity on local CAhN were analysed to test the performance of the P2P protocol. The BitTorrent protocol performance is tested for neighbourhood scope and piece size diversity. The test showed that it is better to keep the neighbourhood of a peer smaller, to avoid the deterioration of data transmission in the community due to the dynamism of the system. The downloading ratio has high value due to the diversity of pieces introduced by the mobility of nodes and the downloading opportunities it raises. Furthermore, the finish time is better and more equally distributed since far nodes can receive pieces from the beginning of the session and can replicate them in their close neighbourhoods. Pieces are sent in an optimal way and in a reduced

number so as to limit the routing overhead. This creates parallel areas of activity in the network. Far nodes do not need to wait for pieces to arrive to their neighbourhoods to download them. Hence, pieces propagate in the network in all directions. The solution of BitTorrent, therefore, encourages a good management of neighbour and piece selection that reduces finish time and increases downloading. A final conclusion must be a trade-off between good finish time and good downloading ratio between peers should be made by the community members depending on the information fedback by the TS to increase their performance.

8.*2* *Recommendations for future work*

There were some limitations faced by the research throughout the thesis work in relation to how the simulation was performed. For example in the performance study of the BitTorrent protocol in ad hoc networking, concerning how the test was compiled, the test has been limited to having information on the activity of each peer with only grid network topology. But the actual CAhN network topology usually displays a different topology with clustering behaviour. Furthermore, the test has been limited to studying a file trace in which peers are less mobile. The future work will be on applying the performance test to more mobile ad hoc networks. High dynamicity of such networks will lead to new interesting problems. Even though the results could be generalized to CAhNs, it is of importance to conduct further measurement and research to rule out the possibility of other test scenarios being influenced by chance. Other limitation is the simulation tool that is used to obtain the required results. Even though NS-2 has benefits, it has some limitations. It takes more effort to realize each simulation module and complexity of adding components. It has also limitation in its out-put data refinement.

The main future work will include the extension of the system to have anonymity in data exchange in CAhN, the core application development and tracking System development. The implementation of anonymity will be transferred to the routing layer. Once the BitTorrent algorithm is in place for fast data discovery and allocation, the anonymity will be provided by the routing protocol as specified in this thesis work. The next main task will be the development of the tracking system (TS). Also the future work will include the design and implementation of a more sophisticated simulation tool to study the properties of the communication mechanism under various other scenarios such as simulations involving more number of peers. Questions regarding the feasibility of the practical implementation of the CAhN concept also remain to be answered. Future studies of CAhN need to take all the technical issues of CAhN into account. Ultimately, a completely functional tracking system (TS) enabled CAhN needs to be studied in the real world. Another interesting area of future work is to find the vulnerabilities of CAhN system.

Appendix A

Changes made for the performance testing

Several modifications have been made to the original source code to personalize the code to the project performance metrics. This section highlights the main changes made to the source code to perform the necessary simulation. The flow chart shows how the system simulation is achieved, through step by step development. The main codes modified are the C++ code of the routing protocol, DSDV, i.e. dsdv.cc and dsdv.h, and the background Tcl script to control the peers. All the steps are described as follows.

1. Makefile

In order to simulate the BitTorrent implementation, a define "BITTO" to the Makefile of NS-2 is added. Also, there is an additional define variable called "BIT_DOWNLOADING". If these variables are inserted the BitTorrent node will handle the DOWNLOADING state of a block; If a block is downloading from a neighbor, a node will not request to offer that block to any other BitTorrent nodes only if there exists an incomplete block of a piece which are not in the DOWNLOADING state. This helps to minimize the number of packets in the network, but it can make that node taken a little bit more to finish downloading.

2. Modified files of the existing NS-2

In order to implement the BitTorrent, some existing ns-2 implementations are modified. Here are the lists of modified files.

- $NS/Makefile
- $NS/tcl/lib/ns-default.tcl
- $NS/common/packet.h
- $NS/ns-process.h
- $NS/dsdv/dsdv.cc

Appendix B

Changes on Tcl Code

These are the parameters which can be configured at tcl scripts. All the default values are inserted at *NS/tcl/lib/ns-default.tcl*.

Application/BitTorrent set max_peer_num_ 10
-> The maximum number of peers that node can have

Application/BitTorrent set max_neighbor_num_ 10
-> The maximum number of neighbors that node can have

Application/BitTorrent set max_upload_num_ 4
-> The maximum number of upload connections that node can have
-> When this value set as 1, the BitTorrent performs a serial transmission.
When this value set over 1, it performs a parallel transmission.

Application/BitTorrent set max_download_num_ 4
-> The maximum number of download connections that node can have

Application/BitTorrent set local_rarest_select_flag_ 1
-> The flag from where the BitTorrent calculate the local rarest piece
-> If is 1, it will calculate from the PeerTable.
-> If is 2, it will calculate from the NeighborTable.

Application/BitTorrent set control_tcp_flag_ 1
-> The flag about how its control message to transmit
-> If 0, use UDP. If 1, use TCP.

Application/BitTorrent set use_bidirection_tcp_flag_ 1
-> The flag about whether to use its TCP connection for uni-direction or bi-direction
-> If 0, it uses uni-direction. If 1, bi-direction.

Application/BitTorrent set peerTable_flag_ 1
-> The flag about whether the node removes its peer which became a Seed when the node itself became a Seed.
-> Recommended to use 1 always.

Application/BitTorrent set selectNodeToUpload_flag_ 2
-> The flag about how to select node to upload
-> If 0, it selects nodes based on a choking algorithm of original BitTorrent (named basic)
-> If 1, it selects only physical 1-hop nodes (named 1HOP)
-> If 2. it selects based the closest node first strategy
-> If 3. a node selects nodes like basic and only the seed selects nodes randomly

Application/BitTorrent set try_upload_interval_ [expr 10 [Application/BitTorrent set max_upload_num_]]*

-> The interval to try to offer the piece to its neighbors

-> The unit is sec (simulation time)

Application/BitTorrent set printCPL_flag_ 1

-> The flag about whether to generate result files of the current having Piece ID list or not

-> If 0, do not generate, and if 1, generate the result files.

Application/BitTorrent set choking_best_slot_num_ 3

-> The number of best slot for choking algorithm

Application/BitTorrent set choking_optimistic_slot_num_ 1

-> The number of optimistic slot for choking algorithm

Application/BitTorrent set peer_update_interval1_ 100

-> The "peer_update_interval" is the interval to flood the HELLO message into the network

-> This value is the peer_update_interval for startUp phase. So, we strongly recommend to set this value smaller than the value of peer_update_interval2 which is the peer_update interval for steady phase.

-> When the timer for peer update is expred, the node starts to flood HELLO message. Whenever it receives the HELLO_REPLY from a node, it adds the sender of HELLO_REPLY to its peerTable.

-> The peerTable is maintained based on update manner not re-creation manner. The remove procedure performed only when the node itself became a seed and at that time it removes only the seed peers from its peerTable.

Application/BitTorrent set peer_update_interval2_ 500

-> The interval to update its peerList by flooding HELLO message at the steady phase

Application/BitTorrent set neighbor_select_interval1_ [expr 2[Application/BitTorrent set try_upload_interval_]]*

-> The interval to select neighbor from the peerTable at startUp phase

-> We also strongly recommend to set this value smaller than the neighbor_select_interval2_

-> Unlike peerTable, neighborTable is maintained based on re-creation manner not update manner

-> In the current version of our BitTorrent implementation, we select neighbors based on its hopcount (Closest one first), but it can be changed based on the performance results.

Application/BitTorrent set neighbor_select_interval2_ [expr 2[Application/BitTorrent set try_upload_interval_]]*

-> The interval to select neighbor from the peerTable at steady phase

Application/BitTorrent set received_bytes_reset_interval_ 60

-> The original BitTorrent adapts the choking algorithm to facilitate the file sharing. In choking algorithm, the BitTorrent node chooses the best unchoking nodes based on how many bytes it provide to me.

-> This value is the interval to reset the history about how many bytes it receives from a

specific node.

Agent/BitAgent set flooding_ttl_ 3
-> The time-to-live value which will be used in flooding of HELLO message

PiecePool *set piece_num_* *10*
-> The number of pieces in the file which will be shared

PiecePool *set block_num_* *10*
-> The number of blocks in one piece

PiecePool *set block_size_* *[expr 10*1000]*
-> The size of a block. The unit is bytes.
-> Therefore, the size of the file which will be shared is piece_num_ * block_num_ * block_size_.

Setting of Tcl script

Other protocols or schemes used for implementation. Below shows configuration that is adapted for all experiments and implementation.

set opt(chan) *Channel/WirelessChannel*
set opt(prop) *Propagation/TwoRayGround*
set opt(netif) *Phy/WirelessPhy*
set opt(mac) *Mac/802_11*
set opt(ifq) *CMUPriQueue*
set opt(ll) *LL*
set opt(ant) *Antenna/OmniAntenna*
set opt(ifqlen) *50*
set opt(seed) *0.0*
set opt(rp) *DSDV*

```
$ns_ node-config -adhocRouting $opt(rp)    \
        -llType $opt(ll) \
        -macType $opt(mac) \
        -ifqType $opt(ifq) \
        -ifqLen $opt(ifqlen) \
```

*-antType $opt(ant) *

*-propType $opt(prop) *

*-phyType $opt(netif) *

*-channel [new $opt(chan)] *

*-topoInstance $topo *

BitTorrent Setting for Tcl script

Belows are about how to create and intialize the BitTorrent node at Tcl scripts.

set node_($i) [$ns_ node $i]

set id [$node_($i) id]

set bitAgent_($i) [new Agent/BitAgent]

set bit_($i) [new Application/BitTorrent $id $node_($i) $opt(nn)]

set addr [$node_($i) node-addr]

$ns_ attach-agent $node_($i) $bitAgent_($i)

$bit_($i) attach-agent $bitAgent_($i)

set piecePool [new PiecePool $file_name_]

$bit_($i) setPiecePool $piecePool

$ns_ at $opt(measureStart) "$bit_($i) start"

The ID of node is passed to the instance of NS-2 which the BitTorrent layer will be attached, and the number of node in the network topology to a BitTorrent node at creation. Also, in order to use a BitTorrent node in the simulation, BitAgent class is binded to every node which will be deployed in the network. After that, a PiecePool (file) is created and attach to a BitTorrent node. At last, the command "start" is called to a BitTorrent node which start the functionalities of the BitTorrent protocol. The "start" command should be called during simulation time.

Appendix C

Routing Protocol Code

Modification on the *'dsdv.cc'* and '*dsdv.h'* files:
Definition of moving node variable
1-In *'dsdv.h'* the moving node variable is defined:

```
bool movingnode;
```

With this variable the node status can be set, as leaving or if it is still inside the community.

2-In *'dsdv.cc'* after

```
DSDV_Agent::DSDV_Agent (): Agent (PT_MESSAGE), ll_queue (0), seqno_ (0),
  myaddr_ (0), subnet_ (0), node_ (0), port_dmux_(0),
  periodic_callback_ (0), be_random_ (1),
  use_mac_ (0), verbose_ (1), trace_wst_ (0), lasttup_ (-10),
  alpha_ (0.875),  wst0_ (6), perup_ (15),
  min_update_periods_ (3) // constants
{
  table_ = new RoutingTable ();
  helper_ = new DSDV_Helper (this);
  trigger_handler = new DSDVTriggerHandler(this);....
```

The above code is needed to initialize, and all nodes are initially assumed to be in the community.

3- Code to catch which node is leaving is added in dsdv.cc after

```
DSDV_Agent::command (int argc, const char *const *argv)
{
  if (argc == 2)
```

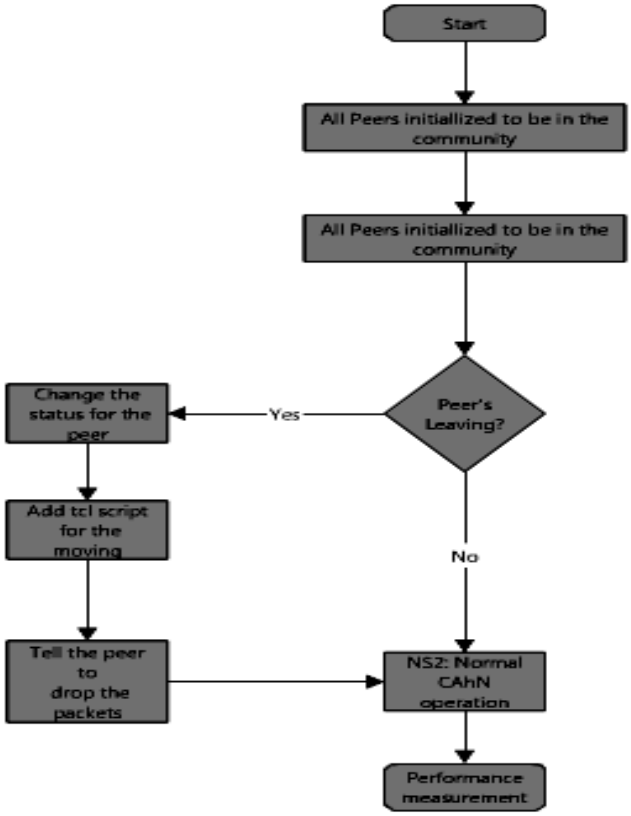

Figure 18: Flow chat showing the simulation procedure for a node

```
{
  Tcl& Tcl = Tcl::instance();
 if (strcmp (argv[1], "start-dsdv") == 0)
       {
        startUp();
        return (Tcl_OK);
       }
```

following line

```
if(strcmp(argv[1], "mnode") == 0)
      {
       movingnode= true;
       Tcl.resultf("%2.2f\n", movingnode);
       return Tcl_OK;
      }
```

4-Setting Tcl script for the moving node. For example, adding the following line to set node 5 as

moving node.

```
$ns_ at 0.0 "[$node_(5) set ragent_] mnode"
```

5- The recv (Packet * p, Handler *) function is used to select next hop node when routing data packets. So it tells, in this case, the movingnode to just drop any packet when it has received already.

```
void DSDV_Agent::recv (Packet * p, Handler *)
{
  hdr_ip *iph = HDR_IP(p);
  hdr_cmn *cmh = HDR_CMN(p);
  int src = Address::instance().get_nodeaddr(iph->saddr())
//    if movingnode
  if ( movingnode == true ) {
    drop(p, DROP_RTR_ROUTE_LOOP);
   // DROP_RTR_ROUTE_LOOP is added for no reason.
 }
```

Bibliography

[1] C. Bisdikian, P. Bhagwat and N. Golmie, "Wireless Personal Area Networks", Guest Editorial, IEEE Network, Vol. 15, No. 5, pp. 10–11, 2001.
[2] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research", Department of Computing
[3] I.G. Niemegeers and S.M. Heemstra de Groot, "From Personal Area Networks to Personal Networks: A User-Oriented Approach", Kluwer International Journal of Wireless and Personal Communications, Vol. 22, No. 2, pp. 175–186, 2002.er Science, Dartmouth Computer Science Technical Report TR2000-381, Dartmouth College, 2000.
[4] M. Musolesi and C. Mascolo. A Community Based Mobility Model for Ad Hoc Network Research . In Proceedings of MSWiM'04, pages 20–24. ACM Press, October 2004.
[5] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In Proc. of ACM SIGCOMM, August 2001.3. K.M. Alzoubi, P.-J.Wan, andO. Frieder.Message- ptimal connected dominating sets in mobile ad hoc networks. In Proc. of ACM MobiHoc, June 2002.
[6] A. Rowstron and P. Druschel. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. of Middleware, November 2001.5. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In Proc. Of ACM SIGCOMM, August 2002.
[7] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. Tapestry: an infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U.C. Berkeley, April 2001roc. of ACM SIGCOMM, August 2001.
[8] Moni Naor and Udi Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach. Proc. SPAA, 2003
[9] http://www.coralcdn.org/
[10] http://www.BitTorrent.org/
[11] Emre Sarigo,Oriana Riva, Patrick Stuedi ,Gustavo Alonso . Enabling social networking in ad hoc networks of mobile phones , Systems Group, Department of Computer Science, ETH Zurich Microsoft Research
[12] Forouzan, B.A. (2000). TCP/IP: Protocol Suite. 1st ed. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
[13] http://blogs.oracle.com/janp/entry/how_the_scp_protocol_works
[14] https://www.torproject.org/
[15] J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In Proceedings of CRYPTO 2005, pages 169–187, 2005.
[16] http://www.i2p2.de/
[17] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private Internet connections. In Communications of the ACM, 1999.
[18] S. Seys and B. Preneel. Arm: Anonymous routing protocol for mobile ad hoc networks. In IEEE AINA, 2006.
[19] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng. Anonymous secure routing in mobile ad-hoc network. In 29th Annual IEEE International
Conference on Local Computer Networks, 2004.
[20] J. Kong and X. Hong. Anodr: Anonymous on demand routing with untraceable routes for mobile ad hoc networks. In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing, 2003.
[21] The Gnutella protocol specification, 2000. http://dss.clip2.com/GnutellaProtocol04.pdf.
[22] D.B.Johnson,D.A. Maltz, Dynamic Source Routing in Ad hoc Wireless Networks, Kluwer

Academic, 1996.
[23] C.E.Perkins, E.M.Royer, Ad hoc on-demand distance vector routing. In Proc. of the IEEE WMCSA, Feb 1999.
[24] C.Perkins, P.Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers in ACM SIGCOMM'94.
[25] S.Ratnasamy, P.Francis, M.Handley, R. Karp and S.shenker. A scalable content-addressable networks. In Proc. of ACM IGCOMM, Aug. 2001.
[26] L.B. Oliviera, I.G.Siqueira, A.A.Loureiro, Evaluation of ad hoc routing protocols under a peer-yto-peer application. In Proc. of IEEE WCNC, Mar 2003.
[27] S.M. Das, H.Pucha, Y.C. Hu. Ekta: an efficient peer-to-peer substrate for distributed applications in mobile ad hoc networks. Technical Report TR-ECE-04-04, Purdue University, Jan 2004.
[28] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips,"The BitTorrent P2P file-sharing system: measurements and analysis". In Proceedings of the 4th International Workshop on Peer-To-Peer, 2005
[29] S. Rajagopalan, C-C. Shen, A cross-Layer Decentralized BitTorrent for Mobile Ad hoc Networks. In Proc. of ACM/IEEE MOBIQUITOUS, San Jose, CA, USA, 2006.
[30] Mohamed Karim SBAI, Chadi BARAKAT, Jaeyoung CHOI, Anwar AL HAMRA and Thierry TURLETTI: BitHoc: BitTorrent for Wireless Ad Hoc networks, Project-Team Plan`ete, INRIA Sophia Antipolis, France
[31] T. Issariyakul, E. Hossain, Introduction to Network Simulator NS2,. DOI: 10.1007/978-0-387-71760-9 1, Springer Science and Business Media, LLC 2009
[32] G. Ding, B. Bhargava, Peer-to-Peer File-Sharing over Mobile Ad hoc Networks. In Proc. of the 2nd IEEE PERCOM-W, Orlando, FL, USA, 2004.
[33] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 85–97, New York, NY, USA, 1998. ACM Press.
[34] D. B. Johnson. Routing in ad hoc networks of mobile hosts. In IEEE Workshop on Mobile Computing Systems and Applications, pages 158 –163, December 1994.
[35] Su Mon Bo, Hannan Xiao, Aderemi Adereti, James A. Malcolm and Bruce Christianson. A Performance Comparison of Wireless Ad Hoc Network Routing Protocols under Security Attack