# Securing Communications in an Anonymous Overlay Network

By

Simon Wilkinson

B.C.S.

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Computer Science
At

Ottawa-Carleton Institute for Computer Science

Carleton University
Ottawa, Ontario
April 2007

*Your file Votre référence*
*ISBN: 978-0-494-27011-0*
*Our file Notre référence*
*ISBN: 978-0-494-27011-0*

## Abstract

Anonymous message sending techniques define a rapidly evolving area in privacy research. Most current systems address only specific aspects of security and privacy. This thesis presents a protocol that addresses all applicable aspects within an anonymous network, and remains secure even in the face of multiple colluding adversaries. The use of techniques such as secret sharing and adaptive routing across disjoint paths through the use of swarm inspired techniques provide multiple tiers of security for all aspects, creating a system with no single point of failure. Simulation results show the effectiveness of these techniques, while also demonstrating that the overhead incurred through the introduction of these anonymity and security measures is within an acceptable range. The concepts and algorithms developed within this thesis have applications beyond those presented herein, and ideas for future directions and improvements are examined.

## Acknowledgements

I would like to thank my supervisors, Tony White and Evangelos Kranakis, for their guidance, patience and support. Their assistance from formulating the base concepts presented in this thesis, up to the fine tuning of the final document was greatly appreciated. Having the opportunity to utilize their knowledge and expertise has been a tremendous benefit, and played a crucial role in making this thesis what it is.

To my fiancé Amber, my parents John and Regina, my sister Antje, and my soon to be in-laws Mitch and Vanessa, thank you for your love and support.

# Table of Contents

# List of Tables

# List of Figures

# List of Equations

## List of Acronyms

| | |
|---|---|
| ACO | - Ant Colony Optimization |
| B.S. | - Beyond Suspicion |
| DATA | - Data Message |
| DES | - Data Encryption Standard |
| DoS | - Denial of Service |
| IDA | - Information Dispersal Algorithm |
| IP | - Internet Protocol |
| MAC | - Media Access Control |
| MANET | - Mobile Ad-Hoc Network |
| OP | - Onion Proxy |
| OR | - Onion Router |
| P2P | - Peer-to-Peer |
| Poss. I. | - Possible Innocence |
| Prob. I. | - Probable Innocence |
| RDM | - Request Download Message |
| RRM | - Resource Reply Message |
| RSM | - Resource Search Message |
| RTT | - Round Trip Time |
| SIDA | - Secret Information Dispersal Algorithm |
| SSMP | - Secret-Sharing-Based Mutual Anonymity Protocol |
| TCP | - Transmission Control Protocol |
| TLS | - Transport Layer Security |
| TTL | - Time-to-live |
| UC | - Utility Counters |

# Chapter 1

# Introduction

## *1.1 Problem and Motivation*

Anonymous message sending techniques define a rapidly evolving area in privacy research. Such sending methods are required for many applications ranging from simple untraceable e-mail communication or distribution of documents, to anonymous electronic voting and payment systems [THV04].

Currently implemented systems in the field of anonymous overlay networks attempt to address specific issues regarding privacy and security, but do not consider all aspects related to these ideas. As will be shown in Chapter 2, there are three main aspects of network security to consider in an anonymous network: identity protection, data confidentiality and data protection.

In order to help attain a strong notion of identity protection, the concept of anonymity will be employed. This anonymity will be provided through the use of adaptive routing techniques. In adaptive routing protocols, there are no static routing tables, thus making it difficult to determine the path a packet has taken or will take. The application of techniques from the field of swarm intelligence, specifically with

respect to ant systems, have shown excellent results when applied to routing problems, and are especially applicable to networks with changing topologies, such as a Peer-to-Peer overlay network.

With these concepts in mind, the problem statement addressed in this document can be defined as "the creation of an adaptive routing protocol to achieve identity protection, data confidentiality and data integrity in an anonymous overlay network."

## 1.2 Contributions

This document presents an adaptive routing protocol that provides secure communications in the context of an anonymous overlay network. The development of this protocol includes the creation of several algorithms, including node initialization and neighbour selection algorithms, along with algorithms to help ensure sender and receiver anonymity. The adaptive nature of the protocol is provided through the development of a swarm-based algorithm that is built upon the unique combination of several existing techniques in a manner to discover disjoint paths in an on-demand fashion. The application of swarm-based routing (discussed in detail in Section 2.4) to overlay networks is an area of research which has previously been explored, but the idea of using competing chemical signals to route across disjoint paths has had little exposure, particularly with respect to communications networks [Whi00].

This document also presents an alteration to the standard notion of secret sharing and threshold schemes (discussed in Section 2.2). Instead of dividing the

shares of a secret among a group of users, the shares are divided among a set of disjoint paths and routed to a single end user. In this manner, one can achieve secret sharing between only two users, while still offering a level of security comparable to a standard application involving many more users.

This document also presents experimental and qualitative analysis of these algorithms. The inclusion of these security measures should come at a minimal performance cost to nodes within the network. The average time taken to reconstruct a message should not exceed the average end-to-end delay by more than 20%, and the average routing load should not vary by more than 15% as a result of the incorporated anonymity measures. These performance objectives are put in place as an end user of the system would not want the cost of anonymity to cause him to experience considerable lag compared to other, non-anonymous systems.

## *1.3 Outline*

The remainder of the thesis is organized as follows. The next chapter will provide background information on the main concepts used within the design of the protocol presented in this document, including definitions of secure communications, threshold schemes, and ant-based routing. Chapter 3 will then review some relevant state of the art systems that aim to achieve similar goals as the protocol proposed in this document, and look at their shortcomings in relation to the problem statement. Chapter 4 will then present the proposed protocol in detail, and Chapter 5 will present

simulation results and analysis of the protocol. Finally, Chapter 6 will provide conclusions and ideas for future work.

# Chapter 2

# Background

This chapter will present a brief review of some of the pertinent ideas used to construct the protocol presented in Chapter 4. It will begin with a review of the components of Network Security, and more specifically, which of these aspects pertain to anonymous networks. It will then move to the idea of secret sharing through the use of threshold schemes in Section 2.2. Section 2.3 will discuss peer-to-peer networks, and in particular, anonymous peer-to-peer networks. The subsequent section will discuss the notion of anonymity, and introduce concepts that define views and degrees of anonymity. Section 2.5 will present background on the idea of swarm-based routing, highlighting many of its desirable properties. Finally, Section 2.6 will conclude this chapter, and reiterate the main components presented within it.

Section 2.1 below will begin with a definition of network security, upon which the strength of the protocol presented in this document will be built.

## *2.1 Network Security*

As the volume of data being exchanged on the internet increases, network security becomes more and more crucial. Security is a broad topic; but in its simplest

form it is concerned with making sure that unintended people cannot read or worse yet, secretly modify messages intended for other recipients. Security also deals with the problems of legitimate messages being captured and replayed, and with people trying to deny that they sent certain messages. Most security problems are intentionally caused by malicious people trying to gain some benefit, get attention, or harm someone [Tan03].

In the context of network security, four different aspects have been defined ([Tan03], [For01]): privacy (secrecy), authentication, message integrity, and non-repudiation (see Figure 2.1).

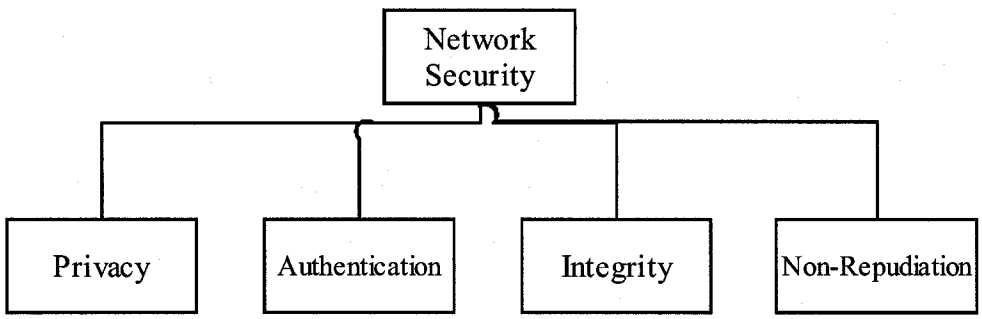


**Figure 2.1 - Aspects of Network Security**

In the context of message sending, privacy implies that the sender and receiver expect confidentiality. The transmitted message should make sense only to the intended receiver. To all others, the message should be unintelligible. Authentication indicates that the receiver is sure of the sender's identity and that an imposter has not sent the message. Data integrity means that the data must arrive at the receiver exactly as it was sent. There must be no undetectable changes made during the transmission,

either accidental or malicious. Non-repudiation means that a receiver must be able to prove that a received message came from a specific sender. Thus, the sender must not be able to deny sending a message that he, in fact, did send [For01].

In the context of an anonymous network both authentication and non-repudiation are impossible. As participants within the network perform anonymous transactions, there is neither a way to ensure that the receiver is aware of the sender's identity, nor is there a way to prove that a received message came from a specific sender. More importantly, however, is that within an anonymous network these are not desirable properties. Thus, of the four aspects to network security, only two are applicable to anonymous networks: privacy and data integrity. These notions will be assessed in detail in the following sections.

### 2.1.1 Privacy

In terms of secure communication, privacy can be broadly categorized under the following headings:

- Hiding the content or nature of a communication
- Hiding the parties engaged in a communication (prevention of identification, or anonymity)
- Hiding the fact that a communication takes place

To further simplify, these three headings could be collapsed to two, namely identity protection and data confidentiality. By hiding the content or nature of a communication, we are providing data confidentiality. Through hiding the parties involved in communications, along with hiding the fact that a communication takes place, identity protection can be provided.

Hiding the content or nature of a communication (data confidentiality) is commonly achieved through the use of encryption. Here, encryption refers to algorithmic schemes that encode plain text into a non-readable form, or ciphertext, preventing any non-authorized party from reading or changing the data. The receiver of the encrypted text uses a "key" to decrypt the message, returning it to its original plain text form. Another method to hide the content of a communication is known as Secret Sharing, and will be introduced in Section 2.2 below. Its application to providing data confidentiality will be discussed in Chapter 3.

Hiding the parties involved in a communication (identity protection) is most often achieved by hiding the true identity of the participants. Anonymity typically refers to a person, and often means that the personal identity or personally identifiable information of that person is not known. The basic intuition behind anonymity is that actions should be divorced from the agents who perform them, for some set of observers. With respect to a basic information-hiding framework, the information that needs to be hidden is the identity of an agent (or set of agents) who perform a particular action. In terms of network communications, anonymity is often provided through the

use of anonyms or pseudonyms. Networks that employ these types of mechanisms are referred to as "anonymous networks." From whom the information needs to be hidden, that is, from which observers, depends on the situation [WON05]. Within communications networks, the adversary is most often an individual or group of individuals that are attempting to subvert communication. The adversarial model that affects the protocol presented in this document will be presented in Section 2.4. Another method of achieving anonymity is through hard to trace routing methods, such as authorized third party systems or through the use of relays. Some of these anonymizing techniques, such as MIX networks and Onion Routing, are discussed in Chapter 3.

Hiding the fact that a communication is taking place (the other portion of identity protection) can be achieved by creating a random data flow, such that the presence of a genuine communication is harder to detect and traffic analysis is less reliable. This type of approach could be termed "security by obscurity," and is similar to the notion of finding a needle in a haystack. Another similar method makes it difficult to tell where a packet originated from and where its intended destination is. Chapter 4 will introduce such a notion to aid in the obfuscation of the true sender and recipient of a message.

### 2.1.2 Data Integrity

Data integrity is most often achieved through the use of a cryptographic hash function. A hash function takes a long string (or message) of any length as input and produces a fixed length string as output, sometimes termed a message digest or a digital fingerprint. This is one of the approaches which are used in the protocol presented in this thesis, and will be discussed in further detail in Chapter 4.

The use of threshold schemes and secret sharing will also add to the model of data integrity presented in this document. Threshold schemes and their properties will be examined in detail in the following section.

## *2.2 Threshold Schemes*

Threshold (or secret) sharing schemes are multi-party protocols related to key establishment. The idea of secret sharing is to start with a secret, and divide it into pieces called shares which are distributed amongst users such that the pooled shares of subsets of users allow reconstruction of the original secret. This may be viewed as a key pre-distribution technique, facilitating one-time key establishment, wherein the recovered key is pre-determined (static), and, in the basic case, the same for all groups [MOV97].

A (k, n)-threshold scheme ($k \leq n$) is a method by which a trusted party computes secret shares $S_i$, $1 \leq i \leq n$, from an initial secret S. Each share, $S_i$, is distributed to user $P_i$.

Any k or more shares can be pooled to reconstruct S, but any group of k-1 or fewer shares can not.

Shamir's Threshold Scheme [S] is based on polynomial interpolation, and the fact that a univariate polynomial $y = f(x)$ of degree t-1 is uniquely defined by t points $(x_i,y_i)$ with distinct $x_i$ (these define t linearly independent equations in t unknowns). Thus, each group member may compute S as a linear combination of t shares. The algorithm for this scheme is outlined below (Mechanism 12.71 in [MOV97] :

**Shamir's (t; n) threshold scheme**

SUMMARY: a trusted party distributes shares of a secret S to n users.

RESULT: any group of t users which pool their shares can recover S.

1. Setup. The trusted party T begins with a secret integer $S \geq 0$ it wishes to distribute among n users.

   (a) T chooses a prime $p > \max(S, n)$, and defines $a_0 = S$.

   (b) T selects t−1 random, independent coefficients $a_1, \ldots, a_{t-1}$, $0 \leq a_j \leq p-1$, defining the random polynomial over $Z_p$, $f(x) = \sum_{j=0}^{t-1} a_j \bullet x^j$.

   (c) T computes $S_i = f(i) \bmod p$, $1 \leq i \leq n$ (or for any n distinct points i, $1 \leq i \leq p - 1$), and securely transfers the share $S_i$ to user $P_i$, along with public index i.

2. Pooling of shares. Any group of t or more users pool their shares. Their shares provide t distinct points (x; y) = (i, $S_i$) allowing computation of the coefficients $a_j$ , $1 \leq j \leq t - 1$ of f(x) by Lagrange interpolation. The secret is recovered by noting $f(0) = a_0 = S$.

This scheme has several desirable properties [MOV97]:

1. **Perfect:** given knowledge of any t-1 or fewer shares, all values $0 \leq S \leq p\text{-}1$ of the shared secret remain equally probable.

2. **Ideal:** the size of one share is the size of the secret.

3. **Extendable for new users:** new shares (for new users) may be computed and distributed without affecting shares of existing users.

4. **Varying levels of control possible:** providing a single user with multiple shares bestows more control upon that individual (this corresponds to changing the access structure).

5. **No unproven assumptions:** unlike many cryptographic schemes, its security does not rely on any unproven assumptions (e.g., about the difficulty of number-theoretic problems).

## *2.3 Peer-to-Peer Networks*

There has been much interest in peer-to-peer data sharing and content distribution applications. They are used by millions of users and they represent a large

fraction of the traffic in the Internet [SW02]. These applications are built on top of large-scale network overlays that provide mechanisms to discover data stored by overlay nodes. Section 2.3.1 will look at different classifications of these overlays. Following this, Section 2.3.2 will introduce the notion of anonymous peer-to-peer networks.

### 2.3.1 Overlay Classifications

A common feature of all routing algorithms is the presence in every network node of a data structure, called a routing table, holding all the information used by the algorithm to make the local forwarding decisions. The routing table is both a local database and a local model of the global network status. The type of information it contains and the way this information is used and updated strongly depends on the algorithm's characteristics. A broad classification of routing algorithms is the following:

- centralized vs. decentralized
- static vs. dynamic
- structured vs. unstructured

The main difference between centralized and decentralized systems is that centralized, or more specifically centrally coordinated systems, such as instant messaging networks and Napster, make use of some form of central server that acts as

a broker between peers, while in decentralized systems, no such intermediary exists. This is illustrated in Figure 2.2. The primary advantage of centralized systems is their simplicity. Given that all data is concentrated in one place, centralized systems are easily managed and have no questions of data consistency or coherence. A primary virtue of decentralized systems is their extensibility. Decentralized systems also tend to be fault-tolerant: the failure or shutdown of any particular node does not impact the rest of the system.

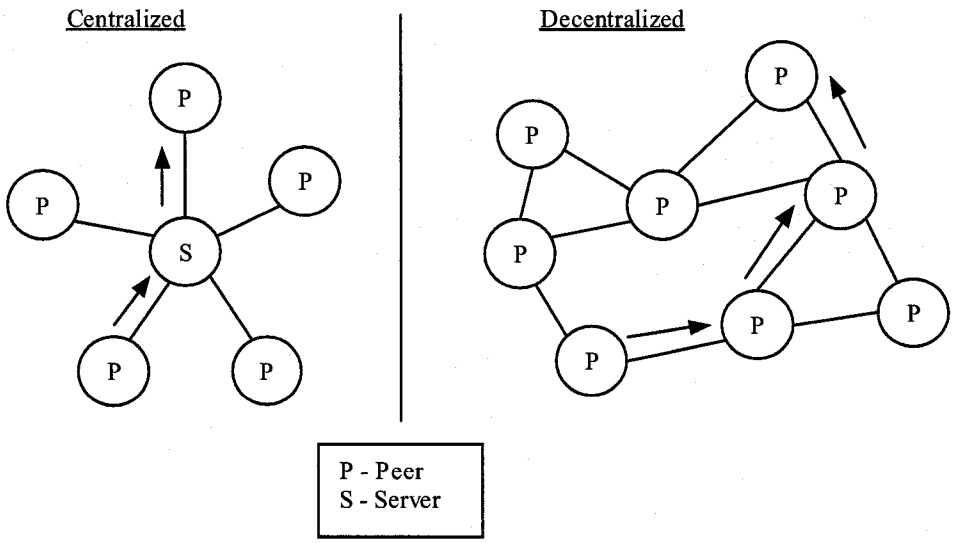


**Figure 2.2 - Differences in structure and message flow for centralized and decentralized networks**

In static (or oblivious) routing systems, the path taken by a packet is determined only on the basis of its source and destination, without regard to the current network state. This path is usually chosen as the shortest one according to some cost criterion, and it can be changed only to account for faulty links or nodes. Dynamic routing systems are, in principle, more attractive, because they can adapt the routing policy to

time and spatially varying traffic conditions. As a drawback, they can cause oscillations in selected paths. This fact can cause circular paths, as well as large fluctuations in measured performance. [DD98].

Unstructured overlays, Gnutella [Gnu06] for example, organize nodes into an arbitrary graph and use floods or random walks to discover data stored by overlay nodes. Each node visited during a flood or random walk evaluates a query locally on the data items that it stores. This approach supports arbitrarily complex queries and it does not impose any constraints on the node, graph or on data placement, for example, each node can choose any other node to be its neighbour in the overlay and it can store the data it owns. But unstructured overlays cannot find rare data items efficiently because this requires visiting a large fraction of overlay nodes. Structured overlays, for example, [SMK+01, RP01, ZKJ01], were developed to improve the performance of data discovery. They impose constraints both on the node graph and on data placement to enable efficient discovery of data. Each data item is identified by a key and nodes are organized into a structured graph that maps each key to a responsible node. The data or a pointer to the data is stored at the node responsible for its key.

### 2.3.2 Anonymous Peer-to-Peer Networks

Recently, anonymous networking has been used to secure communications. In principle, a large number of users running the same system can have communications

routed between them in such a way that it is very hard to detect what any complete message is, which user sent it, and where it is ultimately going from or to.

An anonymous peer-to-peer (P2P) computer network is a particular type of peer-to-peer network in which the users and their nodes are identified by pseudonymous by default. The primary difference between regular and anonymous P2P networks is in the routing method of their respective network architectures. Traditional peer-to-peer networks operate as a web or mesh of neighboring node connections. Each node connects to a few other nodes in the network, and those nodes connect to a few other nodes, which in turn connect to a few other nodes, and so on. A node can then flood a search request to its neighbours, and these neighbours forward the request to their neighbours, and so on. Any node that has the requested resource responds to the request, and within the response includes its Internet Protocol (IP) address. When the requesting node chooses another node to download the resource from, it creates a direct connection to this node using the provided IP address. In an anonymous network, however, a node would respond to a resource request with its pseudonym, and when the resource is downloaded, there is no direct connection established. Instead, the download is routed through the mesh of connected nodes. In this manner, neither the sender nor the receiver is certain who is at the other endpoint of communication, thus providing anonymity between the two. These networks allow for unfettered free flow of information, legal or otherwise.

Because IP addresses can uniquely identify users and their machines on a network, the goal of anonymous routing protocols is to disassociate a user's IP address from the traffic the user initiates (or receives) on the network [SLS01]. When receiving data on any network it must come from somewhere and data must have been requested by someone. The anonymity comes from the idea that no one knows who requested the information as it is difficult - if not impossible - to determine if a user requested the data for himself or simply requested the data on behalf of another user.

The primary cost of providing anonymity is the additional latency of data delivery to responders and the additional bandwidth consumed forwarding traffic for other members. There is also an additional latency cost from the querying process [SLS01].

## *2.4 Anonymity*

This section will present the terms and ideas associated with anonymity that will be used throughout the document. It will present different degrees and views of anonymity, but will begin by defining the adversary who is attacking anonymity.

### 2.4.1 The Adversary

What we would like to realize is absolute anonymity against every possible adversary. But an adversary can control all network stations, all lines, and even the communication partner and so absolute anonymity is theoretically impossible. Therefore we need reasonable models of a possible adversary. The goal of the

adversary considered in this thesis is to determine the identities of the sender and receiver, and to compromise communication between the two.

It is important to note that the burden of proof is put on the adversary. The adversary must identify a node and prove that a communication originated from (or terminated) there. Holding just the restriction that only evidence obtained from protocol-related interactions can be used to prove that a particular node performed a particular action, the adversarial model allows the adversary to do almost anything except breaking cryptographic primitives. Thus, the adversary is assumed to see all encrypted and unencrypted traffic between nodes at all times, but cannot decrypt encrypted traffic between two nodes providing the adversary does not control either node. The adversary controls an arbitrary number of nodes in the network, and the nodes are free to collaborate out-of-band. Adversarial nodes can violate the protocol and/or be well-behaved participants within the network. Communication between arbitrary nodes may be disrupted by the adversary.

While the adversary described above is extremely powerful, its power must be limited slightly. This limitation stems from the observation that if an adversary is able to control (or at least decrypt) all traffic that a particular node sends or receives, the node cannot engage in any anonymous communication. This is true for all anonymous networks. Therefore, in order to provide the user with any degree of anonymity, any anonymous protocol requires that a node must be able to communicate with at least one other node that is not controlled by a malicious adversary [BG03]. In a peer-to-peer

network, this implies that at least one neighbour of any node must be uncorrupted for there to be any notion of anonymity. The following section gives a detailed analysis of the model of colluding nodes.

#### 2.4.1.1 Group of Colluding Nodes

When considering a group of colluding nodes, the adversary is represented by this group. It is assumed that the colluding nodes have no idea of the overall network structure, and that there is perfect information transmission between the colluding nodes. The colluding nodes have a non-uniform distribution within the network. This is based on the observation that in a real life attack, collaborating nodes would not simply scatter throughout the network, but would most likely try to target a specific node, or in the case of an anonymous network, target a specific region of the network.

In this model, the nodes at highest risk of having communications compromised would be the nodes within this region, while the nodes at least risk would be those separated from this region by some distance. As the distance to this region increases, the threat of having security compromised decreases. One can view the group of colluding nodes as a single entity within the network, as it is assumed that there is perfect information transmission between the nodes. This creates a reduced view of the network, and in essence, constructs a scale-free model. This notion is illustrated in Figure 2.3 below.

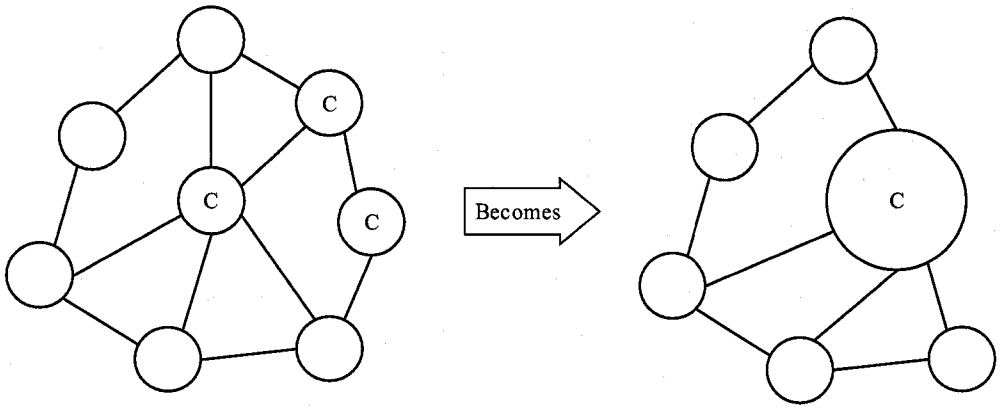


**Figure 2.3 - Reduced Graph of Colluding Nodes**

As can be seen from the figure above, when the graph is reduced in this fashion, it greatly simplifies the topology. An assumption in the threat model is that the group of colluding nodes can not have compromised more paths than the threshold value needed to reconstruct the message. If they did, then they would be able to reconstruct the message simply by collecting enough shares off the first hop from the sending node. If this were the case, then there could be no notion of anonymity of any type of security guarantee.

In the simulations, nodes were given an independent probability of being deemed a colluding node, with a higher probability being given to nodes neighbouring a node already deemed to be colluding. This is presented in detail in Appendix A.

Given a model of the adversary we have to define what we want to keep hidden from him. A strong possibility is to keep the sender and the recipient of a message secret. A weaker possibility is to keep only their relationship secret. This means that while sending and receiving of physical messages is observable, it is infeasible for an

adversary to link the physical message sent by the sender and the physical message received by the recipient. The different views of anonymity considered in this thesis are presented in the following section.

### 2.4.2 Views of Anonymity

When talking about anonymous systems it is vital to be precise about what is anonymous, from whom, under what conditions, and what degree of anonymity is provided. Node-to-node message passing provides anonymity to the originator and final receiver of a message because they can plausibly claim to be nodes in the chain, forwarding the message for someone else.

The main agents involved in file-sharing are the sender, who initiates a search for a file, and the responder or receiver who answers the search query and provides the file. In peer-to-peer networks these agents are communicating through a number of nodes which forward the request and possibly the search data. The adversary in a peer-to-peer system may be the sender, the responder, any node in the system or an outsider. The usual goal is to find out who the sender or responder is, or what they are transferring. Alternatively, we can consider a global adversary who can see the whole network, with the additional goal of linking the sender and receiver. These definitions lead to the following views of anonymity [CC05]:

- Sender anonymity to any node, the responder or a global adversary.
- Responder anonymity to any node, the sender or a global adversary.

- Sender-responder unlinkability to any node or a global adversary.

When discussing the degree of anonymity provided by a system, each of the views above must be considered. It may also be useful to consider an adversary that is a combination of a global adversary, sender, receiver and any number of nodes inside the system. In addition to these views, there must be some manner in which to quantify the anonymity provided by a system. The next section will present such a mechanism.

### 2.4.3 Degrees of Anonymity

Reiter and Rubin [RR98] provide the following useful classification for the degree of anonymity a system provides: These classifications will be used in the analysis of the degree of anonymity provided by current state of the art systems, as well as the proposed protocol presented in this thesis.

- Beyond suspicion (B.S.) - From the adversary's point of view, the detected user appears no more likely to have originated the action than any other node.
- Probable innocence (Prob. I.) - From the adversary's point of view, the detected user appears no more likely to have originated the action than to not to have.
- Possible innocence (Poss. I.) - From the adversary's point of view, there is a nontrivial probability that the detected user did not originate the action.

In [HON05] Halpern and O'Neill formalized the above notions. In the following definitions, the formula $\theta(i,a)$ is used to represent "agent $i$ has performed action $a$, or will perform $a$ in the future." Both probable and possible innocence can be defined using the notion of $\alpha$ -anonymity, which is defined in [HON05] as follows. Action $a$, performed by agent $i$, is $\alpha$ -anonymous with respect to agent $j$ if,

**Equation 2:1 - Alpha-Anonymous**

$$\Pr_j[\theta(i,a)] < \alpha, where \sum_{j=0}^{N} \Pr_j = 1$$

In other words, an action $a$ performed by agent $i$ is $\alpha$ -anonymous to any observer $j$ if from $j$'s perspective the probability that $i$ performed the action is less than $\alpha$. In addition, the sum of all these probabilities over the $N$ nodes under consideration must equal 1. Using this definition, possible innocence corresponds to $\varepsilon$ -anonymity, for some nontrivial value of $\varepsilon$. Probable innocence can be viewed as illustrated in Equation 2.2. Here the alpha-anonymity is a special case of less than or equal to, as opposed to strictly less than. As the highest probability of him performing the action is 50%, indicating that he is precisely as likely to have originated the action as to have to have, this definition covers Probable Innocence from the worst case scenario to the best case scenario, where alpha would be 0%.

**Equation 2:2 - Probable Innocence**

$$\Pr_j[\theta(i,a)] \leq 0.5$$

Using this terminology, Equation 2.3 illustrates the notion of Beyond Suspicion for some node $i$, where $i$ is beyond suspicion with respect to $j$ for each $i' \in I_A$.

**Equation 2:3 - Beyond Suspicion**

$$\Pr_j[\theta(i,a)] \leq \Pr_j[\theta(i',a)].$$

Here, $I_A$ is the set of all agents under consideration by $j$ to have performed the action $a$. The three degrees of anonymity presented in this section will be used in Chapter 3, where current state of the art research is analyzed, along with Chapter 5 where the protocol presented in this document is analyzed.

### 2.4.4 Attacks on Anonymity

This section will present a review of common types of attacks. The adversary may be the sender, the receiver, any node in the system, or a global adversary. Attacks can become much more effective if there are a number of adversaries working together. If adversaries can mostly surround a node, in any system, they can usually degrade that node's anonymity. Knowledge of the network topology is also useful for an adversary. While many implemented systems combine a number of the basic methods for anonymity, real attacks on these systems may combine a number of the attack methods outlined below [CC05].

- *Time-to-Live Attacks*: Time-to-live (TTL) counters determine the maximum number of hops for a message and are used in most peer-to-peer networks to avoid flooding. If an adversary can send a request to a node with such a low time-to-live counter that the packet will not be forwarded, any response reveals that node as the responder.

- *Eavesdropping Attack*: An adversary may simply eavesdrop on communication, leaving the content of the communication unchanged. This examination of message contents can occur either on a link between two nodes involved in the communication, or at an actual node, which is acting in a malicious manner itself.
- *Multiple Adversaries or Identities*: If adversaries can make repeated connections to a single node with different identities, the anonymity of that node is usually lost. Douceur [Dou02] points out that it is almost always possible for a single adversary to assume a number of different identities, which he terms a "Sybil" attack.
- *Statistical Attacks*: Any adversary can gather statistical data over time. Systems that are provably safe for a single run may reveal information about the identities of their participants when all the observable messages of a longer run are analysed for patterns.
- *Time-based attack*: The time a responder takes to respond to a request may indicate how far away the responder is from the adversary, or how many steps the system took. Especially when combined with a statistical attack, this can provide some information on the responder and help to compromise their anonymity.
- *Denial of Service Attacks*: Denial of service (DoS) attacks can be particularly awkward when nodes can act anonymously, as this could mean that the node

performing a DoS attack can not be identified and removed from the system. While anonymous systems cannot stop all DoS attacks, care should be taken to ensure that their design does not make DoS attacks particularly easy.

- *Malleability Attacks*: An adversary modifies the contents of a message. If the message is plaintext, then the adversary simply modifies the plaintext, while if the contents are encrypted, then the adversary attempts to modify the encrypted content to create predictable changes to the decrypted plaintext.

## 2.4 Swarm-Based Routing

Swarm-based routing is based upon the properties of ant-like agents. These agents use a chemical signal analogy called a pheromone to construct routing tables in networks. The fundamental idea behind using swarm intelligence for routing in ad hoc networks is to utilize the interaction of many packets to generate these routing tables while minimizing the use of explicit routing packets. Protocols that employ these ideas are distributed algorithms based on local decisions that inherently adapt to the dynamic nature of the environment. Within the protocol presented in this thesis the terms ants, agents, and messages all refer to the same notion.

Ant algorithms were inspired by the observation of real ant colonies. Ants are social insects that live in colonies and whose behaviour is directed more to the survival of the colony as a whole than to that of a single individual component of the colony. Social insects have captured the attention of many scientists because of the high level

of structure their colonies can achieve, especially when compared to the relative simplicity of the colony's individuals. An important and interesting behaviour of ant colonies is their foraging behaviour, and, in particular, how ants can find shortest paths between food sources and their nest. While walking from food sources to the nest and vice versa, ants deposit on the ground a substance called pheromone, forming in this way a pheromone trail. Ants can smell pheromone and, when choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest). It can also be used by other ants to find the location of the food sources found by their nest-mates. It has been shown experimentally that this pheromone trail following behaviour can give rise, once employed by a colony of ants, to the emergence of shortest paths. That is, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back.

Most applications of swarm-based routing techniques applied to ad hoc networks are in the area of Mobile Ad Hoc Networks (MANETs). These networks share many similar characteristics with Peer-to-Peer (P2P) networks, such as nodes constantly joining and leaving the network, creating a constantly changing topology.

The main component to these routing algorithms is the use of the routing table. The routing tables contain, for each destination, a vector of real-valued entries, one for each known neighbour node. These entries are a measure of the goodness of going

over that neighbour on the way toward the destination. The entries are termed pheromone variables, and are continually updated according to path quality values calculated by the agents. The repeated and concurrent generation of path-sampling agents results in the availability at each node of a bundle of paths, each with an estimated measure of quality. A detailed explanation of these updates can be found in [Whi05].

This pheromone information is used for the routing of both agents and data packets within the network. Thus, all packets are routed stochastically, choosing with a higher probability those links associated with higher pheromone values. In this way data for the same destination is spread over multiple paths, resulting in load balancing.

For data packets, mechanisms are usually adopted to avoid low quality paths, while agents are more explorative, so that less attractive paths are occasionally sampled and maintained as backup paths in case of failure or sudden congestion. In this way path exploration is kept separate from the use of paths by data. If enough agents are sent to the different destinations, nodes can keep up-to-date information about the best paths, and automatically adapt their data load spreading to this. Within the protocol presented in this thesis, there is no difference between a data packet and an agent. All protocol packets behave as agents, updating the pheromone table at each node they are routed across.

Swarm based routing provides many benefits over more traditional routing methods. These advantages are detailed below, by considering important properties of ad-hoc networks [GKB03].

- *Dynamic topology*: This property is responsible for the poor performance of many 'classical' routing algorithms in multi-hop ad-hoc networks. Swarm algorithms are based on autonomous agent systems imitating individual agents. This allows a high adaptation to the current topology of the network.
- *Local work*: In contrast to other routing approaches, swarm algorithms are based only on local information. Therefore, no routing tables or other information blocks have to be transmitted to other nodes of the network to provide a global view.
- *Link quality*: It is possible to integrate the connection/link quality into the computation of the pheromone concentration, especially into the evaporation process. This will improve the decision process with respect to the link quality. It is important to note that the approach can be modified so that nodes can also manipulate the pheromone concentration independent of the agents. This could occur, for example, if a node itself detects a change of the link quality.
- *Multi-path support*: Each node has a routing table with entries for all its neighbours which also contain the pheromone concentrations. The decision rule for selection of the next node is based on the pheromone concentration at the current node which is provided for each possible link. Thus, the approach

supports multi-path routing as the probabilistic routing will send packets all over the network.

In addition to the properties of ad hoc networks that are discussed above, swarm-based algorithms also exhibit the following desirable characteristics [Rot03]:

- *Fast route recovery*: Packets can easily be sent to other neighbours by re-computing next-hop probabilities.
- *Low Complexity*: Little special purpose information must be maintained aside from pheromone/probability information.
- *Scalability:* As with any colonies numbering in the millions, swarm-based algorithms can potentially scale across several orders of magnitude.

## 2.5 Conclusion

This chapter has presented a classification of the components involved in network security, and more importantly, those which have relevance within an anonymous network, namely privacy (identity protection and data confidentiality) and data integrity. It then outlined methods to achieve these goals, including the use of secret sharing through threshold schemes, and the concept of anonymous networking. Section 2.4 looked at some useful definitions of anonymity, and introduced the adversarial model and some possible attacks on anonymity. In Section 2.5, the idea of

swarm-based routing and its many useful traits when applied to anonymous routing in a dynamic network were introduced.

The next chapter will look at current state of the art implementations in anonymous and adaptive routing, and will analyze them with respect to the problem statement presented in Chapter 1, using some of the definitions and concepts highlighted in this chapter.

# Chapter 3

## State of the Art in Anonymous and Adaptive Routing

This chapter will present some of the current state of the art work on providing various aspects of anonymity in a variety of networks, including peer-to-peer systems. It will examine various protocols that claim to provide facets of anonymity and secure communication. It will begin by presenting various anonymity building blocks and will then examine protocols that employ these different techniques to develop anonymous systems. The protocols will be briefly discussed in terms of goals and implementations and will then be analyzed and critiqued, outlining where they fail in the context of the main problem statement of this document.

### *3.1 Various Anonymity Building Blocks*

Many efforts have been made to introduce anonymity in P2P systems. Earlier work falls into two main categories: anonymous publishing and anonymous communication. Recently there have been two main approaches to providing anonymous publishing [HLX+05]. One employs hash functions to mark the key information of documents, such as with Freenet [CSWH01], while the other approach, used in protocols such as Free Haven [DFM00],[Ser02], use MIX networks (described

below) in conjunction with threshold schemes to achieve the goal of anonymous file sharing [HLX+05]. Anonymous communication is commonly accomplished through the use of one of the several methods introduced below.

MIX networks, introduced by David Chaum [Cha81], provide anonymity by forwarding messages from node to node, but instead of forwarding each message as it arrives, the nodes wait until they have received a number of messages and then forwards them mixed up (See Figure 3.1). When done correctly this can hide the sender and the receiver of the message, as well as hiding sender-receiver linkablity from a global adversary. This can be done without requiring all of the nodes to consistently broadcast packets. One drawback is that each node has to hold a message until it has enough messages to properly mix them up, which might be difficult in a file-sharing system due to the asymmetrical nature of downloading files.

Onion routing is a general-purpose protocol [SGR97] that allows anonymous connections over public networks on the condition that the sender knows the public keys of all the other nodes. It is based on David Chaum's MIX networks, though it includes a number of advances and modifications. Among these modifications is the concept of "routing onions", which encode routing information in a set of encrypted layers. The primary innovation in Onion Routing is this concept of the Routing Onion. To create an Onion, the Proxy at the head of a transmission selects a number of Onion Routers at random and generates a message for each one, providing it with symmetric keys for decrypting messages, and instructing it which Router will be next in the path

(Figure 3.2). Each of these messages, and the messages intended for subsequent routers, is encrypted with the corresponding Router's public key. This provides a layered structure, in which it is necessary to decrypt all outer layers of the onion in order to reach an inner layer.

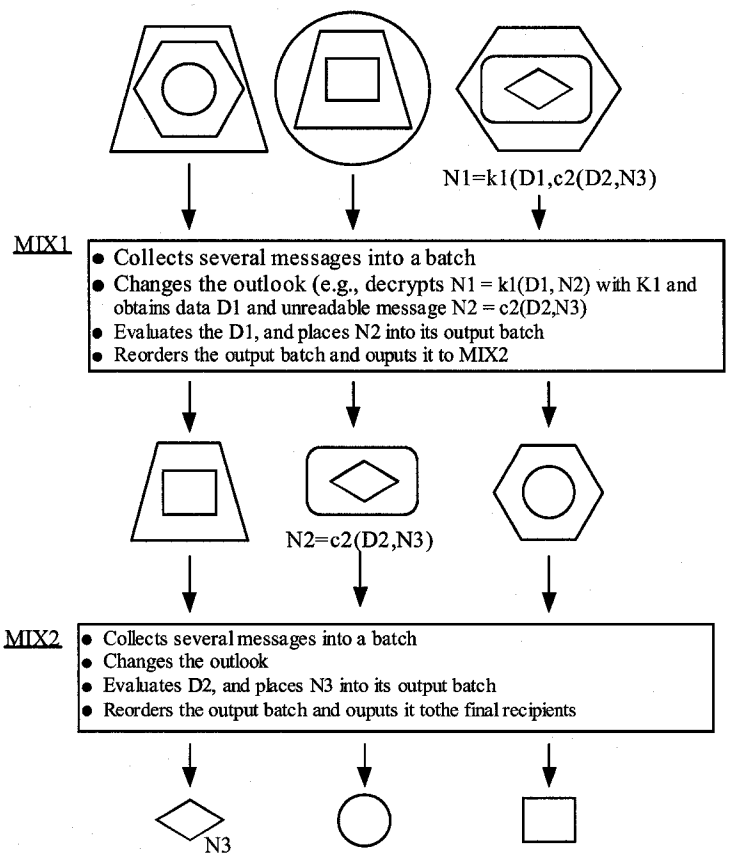


**Figure 3.1 - Example of a MIX scheme**

The onion metaphor describes the concept of such a data structure. As each router receives the message, it "peels" a layer off of the onion by decrypting with its private key, thus revealing the routing instructions meant for that router, along with the encrypted instructions for all of the routers located farther down the path. Due to this

arrangement, the full content of an Onion can only be revealed if it is transmitted to every router in the path in the order specified by the layering.

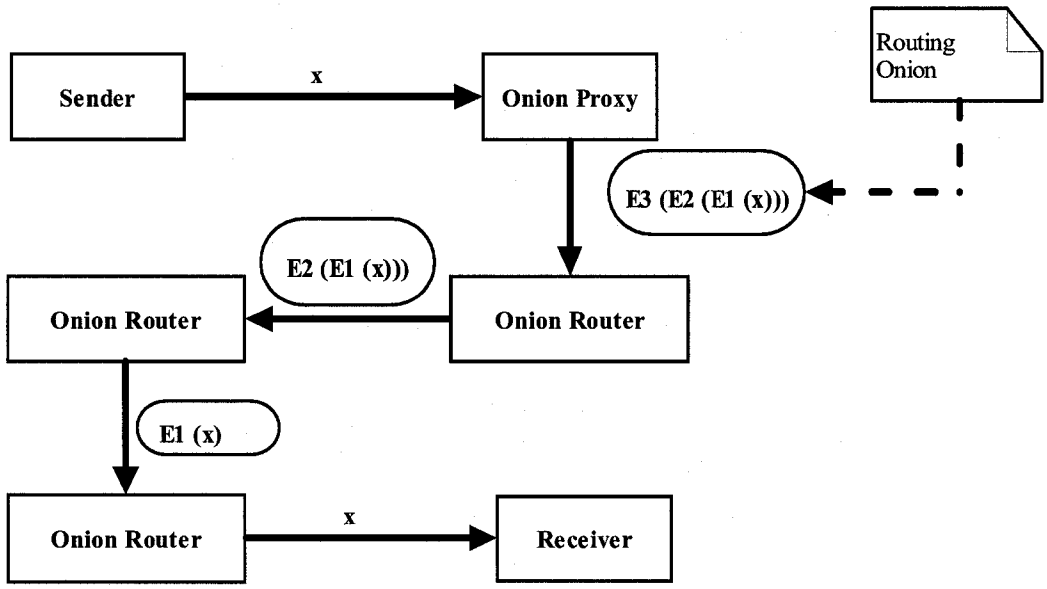


**Figure 3.2 - Onion Routing**

Ant-based protocols such as [Mute06] and [BASM04] have also been developed to provide anonymity measures. This type of approach utilizes the concepts presented in Section 2.5, Swarm Based Routing, in the previous chapter.

The following sections will examine implemented anonymous publishing systems. The first will examine a protocol based on Freenet, a hash-based system, and the second will present Free Haven, a protocol based on the concept of MIX networks. These publishing systems are designed for anonymous storage and retrieval, and the anonymity issues for such a system are different than a system that provides anonymity for communicating parties that are on-line. However, with respect to the problem

statement of this thesis, these systems, along with all others presented herein, will be evaluated with respect to Identity Protection, Data Confidentiality and Data Integrity in order to compare with the protocol and algorithms presented in the subsequent chapter. The final two sections of this chapter will focus on systems that aim to achieve anonymous communication. Section 3.3 will focus on Tor and SSMP, protocols which utilize the Onion Routing model, and finally MUTE, a protocol developed based on swarm-based routing techniques, will be examined.

## *3.2 Freenet: An Anonymous Storage and Retrieval System*

Developed by Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, Freenet [CSWHO1] is an adaptive peer-to-peer network application that permits the publication, replication, and retrieval of data while protecting the anonymity of both authors and readers. Freenet operates as a network of identical nodes that collectively pool their storage space to store data files and cooperate to route requests to the most likely physical location of data. No broadcast search or centralized location index is employed. Files are referred to in a location-independent manner, and are dynamically replicated in locations near requestors and deleted from locations where there is no interest. It is infeasible to discover the true origin or destination of a file passing through the network and difficult for a node operator to determine or be held responsible for the actual physical contents of her own node [CSWH01]

Freenet has five main design goals [CSWH01]:

- Anonymity for both producers and consumers of information
- Deniability for possessors of information
- Resistance to attempts by third parties to deny access to information
- Efficient dynamic storage and routing of information
- Decentralization of all network functions

While Freenet does not aim to hide the provider of a particular file it does aim to make it impossible for an adversary to find all copies of a particular file. A key feature of the Freenet system is that each node will store all the files that pass across it, deleting the least used if necessary. A binary file key, obtained by applying a hash function, of the title (and other key words) identifies the files. Each node maintains a list of the keys corresponding to the files on immediately surrounding nodes.

To retrieve a file, a user must first obtain or calculate its binary file key. He then sends a request message to his own node specifying that key and a hops-to-live value. When a node receives a request, it first checks its own store for the data and returns it if found, together with a note saying it was the source of the data. If not found, it looks up the nearest key in its routing table to the key requested and forwards the request to the corresponding node. If that request is ultimately successful and returns with the data, the node will pass the data back to the upstream requestor, cache the file in its own data store, and create a new entry in its routing table associating the actual data source with the requested key. A subsequent request for the same key will

be immediately satisfied from the local cache; a request for a "similar" key (determined by lexicographic distance) will be forwarded to the previously successful data source.

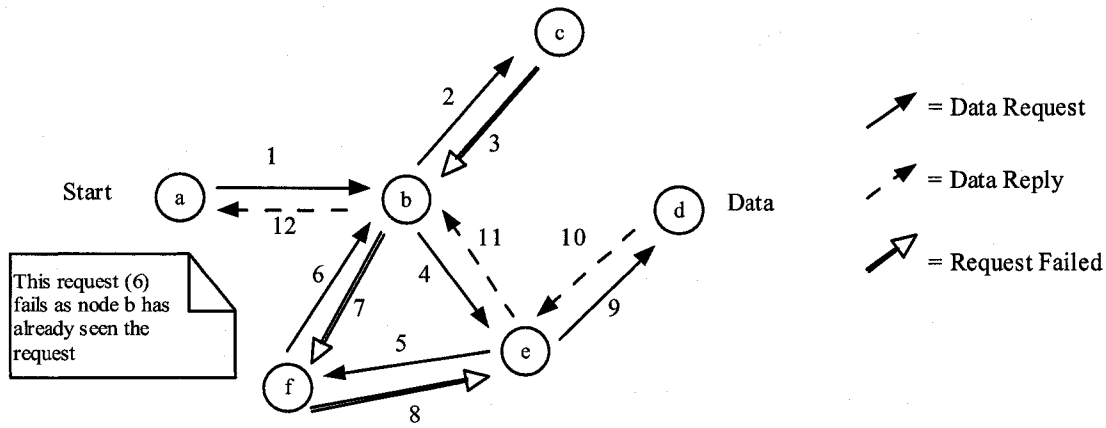


**Figure 3.3 - A typical request sequence in Freenet**

Figure 3.3 depicts a typical sequence of request messages. The user initiates a request at node $a$. Node $a$ forwards the request to node $b$, which forwards it to node $c$. Node $c$ is unable to contact any other nodes and returns a backtracking "request failed" message to $b$. Node $b$ then tries its second choice, $e$, which forwards the request to $f$. Node $f$ forwards the request to $b$, which detects the loop and returns a backtracking failure message. Node $f$ is unable to contact any other nodes and backtracks one step further back to $e$. Node $e$ forwards the request to its second choice, $d$, which has the data. The data is returned from $d$ via $e$ and $b$ back to $a$, which sends it back to the user. The data is also cached on $e$, $b$ and $a$.

The primary security goals for Freenet are protecting the anonymity of requestors and inserters of files. The creators of the system also indicate the importance

of protecting the identity of storers of files. Although trivially anyone can turn a node into a storer by requesting a file through it, thus "identifying" it as a storer, what is important is that there remain other, unidentified, holders of the file so that an adversary cannot remove a file by attacking all of the nodes that hold it. Files must also be protected against malicious modification, and finally, the system must be resistant to denial-of service attacks [CSWH01].

#### 3.2.1.1 Identity Protection

The following table gives a general idea of the anonymity provided by Freenet [HLX+05], through the use of the views and degrees of anonymity provided in Chapter 2.

**Table 3:1 - Anonymity Provided By Freenet**

| | Freenet |
|---|---|
| Sender anonymous to Global Adversary | No |
| Responder anonymous to Global Adversary | No |
| Sender anonymous to Responder | Probable Innocence |
| Sender anonymous to Node | Probable Innocence |
| Responder anonymous to Sender | No |
| Responder anonymous to Node | No |
| Sender-Responder unlinkable to Node | Probable Innocence |
| Sender-Responder unlinkable to Global Adversary | No |

Protection for data sources (sender anonymity) is provided by the occasional resetting of the data source field in replies. The fact that a node is listed as the data source for a particular key does not necessarily imply that it actually supplied that data, or was even contacted in the course of the request. It is not possible to tell whether the downstream node provided the file or was merely forwarding a reply sent by someone else. In fact, the very act of successfully requesting a file places it on the downstream node if it was not already there, so a subsequent examination of that node on suspicion reveals nothing about the prior state of affairs, and provides a plausible legal ground that the data was not there until the act of investigation placed it there. Requesting a particular file with a hops-to-live of 1 does not directly reveal whether or not the node was previously storing the file in question, since nodes continue to forward messages having hops-to-live of 1 with finite probability. The success of a large number of requests for related files, however, may provide grounds for suspicion that those files were being stored there previously.

In the TTL forwarding scheme mentioned above, each node that receives a chain-mode message essentially flips a weighted coin. When a node flips heads, it picks one neighbour to send the message to and continues the chain. If a node flips tails, it drops the message. Increasing the probability of heads increases the expected length of the chain and also increases responder anonymity. An adversary has no way to force its neighbour to be the only node that sends back results, since it has no way to control how many nodes beyond its neighbour will process the message.

This scheme can fall victim to a statistical analysis attack. If the tail lengths are determined by flipping independent coins at each hop, we expect more length-1 tails than length-2 tails. An adversary can send searches repeatedly into a tail and measure the frequency of various search results. The most frequent results must be coming from the length-1 tails, so they must be from the adversary's neighbour.

Another attack that could occur involves multiple colluding nodes working in conjunction with each other. During the execution of this scheme, nodes only send the request on to at most one of their neighbours. An adversary could compose a fake search request that instantly enters this scheme (by falsely setting the TTL value to 1) and send it to a selected node. This node would send back results and send the tail-chain message on to one of its neighbours. If the chosen neighbour is also controlled by the adversary, the two attacking nodes have effectively cornered the selected node. Figure 3.4 below illustrates this attack: Adversary A has sent a message with a TTL value of 1 to node B, and receives results back from it, while adversary C has received node B's forwarded message. If adversary C drops the message, then node A can be sure that the results it sees are coming from node B.

A node is always sending a message in this scheme to only one neighbour. Malicious nodes can exploit this fact about the scheme, and can corner a selected node. If this node happens to send message in this state onward to an adversary, the adversary knows that the message was not sent on to any other neighbours. Thus, whatever

results they receive must be coming from the selected node. When cornered in this way, a node has no room to claim plausible deniability.

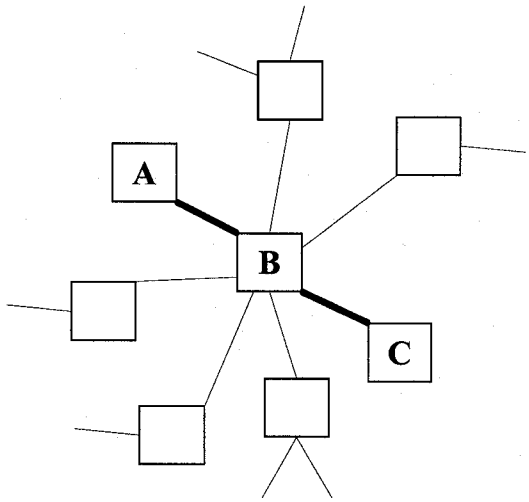


**Figure 3.4 - Nodes A and C effectively surround node B**

As Freenet communication is not directed towards specific receivers, receiver anonymity is more accurately viewed as key anonymity, the act of hiding the key which is being requested or inserted. Unfortunately, since routing depends on knowledge of the key, key anonymity is not possible in the basic Freenet scheme. Therefore, a specific request can be linked to a specific node, thereby partially hindering aspects of pseudonymity.

Within Freenet, the concept of sender-receiver unlinkability is again not fully applicable. Freenet is designed for anonymous storage and retrieval, and the anonymity issues for such a system are different than a system that provides anonymity when communicating parties are on-line. Thus, since Freenet is concerned with anonymous publishing and retrieval of documents, and not direct file transfers between peers, this

sender-receiver relationship does not exist directly, and consequently is not addressed within the algorithm.

### 3.2.1.2 Data Confidentiality

In Freenet, data confidentiality is provided through the concept of deniability. Within the system, nodes cache copies of the data sent across them. So, after a node forwards a file to its neighbour, the neighbour will know that the node is offering that file for download [CC05]. Thus a node can deny having had said data before the request came across it.

Against a local eavesdropper there is no protection on messages between the user and the first node contacted. Since the first node contacted can act as a local eavesdropper, it is recommended that the user only use a node on his own machine as the first point of entry into the Freenet network. Messages between nodes are encrypted against local eavesdropping, although traffic analysis may still be performed (e.g. an eavesdropper may observe a message going out without a previous message coming in and conclude that the target originated it).

### 3.2.1.3 Data Integrity

Modification of requested files by an adversarial node in a request chain is an important threat to data integrity, and not only because of the corruption of the files themselves. Since routing tables in Freenet are based on replies to requests, a node might attempt to steer traffic towards itself by pretending to have files when it does not

and simply returning fabricated data. Even without steering traffic towards itself, a node may apply a malleability attack when handling a requested file.

## *3.3 MIX Networks*

### 3.3.1 Free Haven: Distributed Anonymous Storage Service

The Free Haven Project [DFM00] aims to design, implement, and deploy a functioning distributed anonymous storage service. Storage is distinguished from publication in that storage services focus less on accessibility and more on persistence of data. The project is based on the idea of both secret sharing and MIX networks [CC05].

The Free Haven design is based on a community of servers called the servnet. Each server, or servnet node, holds pieces of some documents. These pieces are called shares. In addition, each servnet node has a persistent identification or pseudonym which allows it to be identified by other servnet nodes or potential Free Haven users. Each server has a public key and one (or more) reply blocks, which together can be used to provide secure, authenticated, pseudonymous communication with that server. Every machine in the servnet has a database which contains the public keys and reply blocks of the other servers on the network.

To introduce a file $F$ into the servnet, the publishing server first uses Rabin's information dispersal algorithm (IDA) [Rab89] to break the file into $n$ shares $f_1,...,f_n$ where any $k$ shares are sufficient to recreate $F$. The server then generates a key

pair $(PK_{doc}, SK_{doc})$, constructs and signs a data segment for each share $f_i$, and inserts those segments as new data into its local server space. Attributes in each share include a timestamp, expiration information, the public key which was used to sign it (for integrity verification), information about share numbering, and the signature itself.

Documents in Free Haven are indexed by $H(PK_{doc})$, the hash of the public key from the key pair which was used to sign the shares of the document. Readers must locate (or be running) a server that performs the document request. The reader generates a key pair $(PK_{client}, SK_{client})$ for this transaction, as well as a one-time remailer reply block. The server broadcasts a request $H(PK_{doc})$, along with the client's public key, $PK_{client}$, and the reply block. This request goes to all the other servers that the initial server knows about. These broadcasts can be queued and then sent out in bulk to conserve bandwidth.

Each server that receives the query checks to see if it has any shares with the requested hash of $PK_{doc}$. If it does, it encrypts each share using the public key $PK_{client}$ enclosed in the request, and then sends the encrypted share through the remailer to the enclosed address. Once k or more shares arrive at the destination, the client can reconstruct the file and the retrieval of the document is complete.

### 3.3.1.1 Identity Protection

In Free Haven, the issue of identity protection is addressed through the use of an external MIX-based communications layer. This prevents most or all adversaries from being able to determine the source or destination of a given message, or establish linkability between each endpoint of a set of messages. Even if server administrators are subpoenaed or otherwise pressured to release information about these entities, they can openly disclaim any knowledge. Therefore, as with most other publishing systems, where they allege to provide anonymity, this is most often done through the use of a separate protocol/system that the publishing system is layered on top of. The level of anonymity provided by MIX networks is outlined in the table below.

**Table 3:2 - Anonymity Provided By MIX Networks**

| | MIX networks |
|---|---|
| Sender anonymous to Global Adversary | No |
| Responder anonymous to Global Adversary | No |
| Sender anonymous to Responder | Beyond Suspicion |
| Sender anonymous to Node | No |
| Responder anonymous to Sender | No |
| Responder anonymous to Node | No |
| Sender-Responder unlinkable to Node | Beyond Suspicion |
| Sender-Responder unlinkable to Global Adversary | Beyond Suspicion |

### 3.3.1.2 Data Confidentiality

As is the case with Freenet, nodes cache copies of the data sent across them in Free Haven. So, after a node forwards a file to its neighbour, the neighbour will know that the node is offering that file for download. Therefore, Free Haven provides a notion of data confidentiality in the same manner as discussed with Freenet in the previous section.

The encryption of messages between agents in the Free Haven system is again fully reliant on the choice of anonymous communication channel, such as Onion Routing. Thus, Free Haven itself provides no notion of data confidentiality in this sense, as any eavesdropping attack would illuminate the contents of a message without the use of this separate anonymous protocol.

### 3.3.1.3 Data Integrity

Data integrity is provided through the mechanism of secret sharing in Free Haven. As files are broken into shares modified shares will not reconstruct properly, thus an adversary could not insert any tailored data in a manner to cause predictable damage or modifications, such as with a malleability attack.

## 3.4 Onion Routing

This section will examine protocols that are based on the notion of Onion Routing. Section 3.4.1 will first look at Tor, and then section 3.4.2 will look at Secret-Sharing-based Mutual anonymity Protocol (SSMP).

As both of these systems are based upon Onion Routing, the following table [HLX+05] gives an analysis of the views and degrees of anonymity provided by these designs, according to the previously introduced terminology. The issue of identity protection will be examined for each protocol in detail in the subsequent sections.

**Table 3:3 - Anonymity Provided By Onion Routing**

| | Onion Routing |
|---|---|
| Sender anonymous to Global Adversary | No. |
| Responder anonymous to Global Adversary | No |
| Sender anonymous to Responder | Beyond Suspicion |
| Sender anonymous to Node | No. |
| Responder anonymous to Sender | No |
| Responder anonymous to Node | No |
| Sender-Responder unlinkable to Node | Beyond Suspicion |
| Sender-Responder unlinkable to Global Adversary | Beyond Suspicion |

### 3.4.1 Tor: The Second-Generation Onion Router

Tor [DMS04] is a second-generation Onion Routing system that addresses limitations in the original design. Tor seeks to frustrate adversaries from linking

communication partners, or from linking multiple communications to or from a single user. It provides the following improvements over the original Onion Routing design [DMS04]:

- Perfect forward secrecy
- Separation of "protocol cleaning" from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control
- Directory servers
- Variable exit policies
- End-to-end integrity checking
- Rendezvous points and hidden services

Each onion router (OR) maintains a Transport Layer Security (TLS) [DA99] connection to every other onion router. Each user runs local software called an onion proxy (OP) to fetch directories, establish circuits across the network, and handle connections from user applications. These onion proxies accept Transmission Control Protocol (TCP) streams and multiplex them across the circuits. The onion router on the other side of the circuit connects to the requested destinations and relays data.

Each onion router maintains a long-term identity key and a short-term onion key. The identity key is used to sign TLS certificates, to sign the OR's router descriptor (a summary of its keys, address, bandwidth, exit policy, and so on), and (by directory servers) to sign directories. The onion key is used to decrypt requests from users, to set up a circuit and negotiate ephemeral keys.

A user's OP constructs circuits incrementally, negotiating a symmetric key with each OR on the circuit, one hop at a time. To begin creating a new circuit, the OP (call her Alice) sends a create cell to the first node in her chosen path (call him Bob). (She chooses a new circuit identifier $C_{AB}$ not currently used on the connection from her to Bob.) The create cell's payload contains the first half of the Diffie-Hellman handshake ($g^x$), encrypted to the onion key of the OR (call him Bob). Bob responds with a created cell containing $g^y$ along with a hash of the negotiated key $K = g^{xy}$.

Once the circuit has been established, Alice and Bob can send one another relay cells encrypted with the negotiated key.

To extend the circuit further, Alice sends a relay extend cell to Bob, specifying the address of the next OR (call her Carol), and an encrypted $g^{x_2}$ for her. Bob copies the half-handshake into a create cell, and passes it to Carol to extend the circuit. (Bob chooses a new circuit identifier $C_{BC}$ not currently used on the connection between him and Carol. Alice never needs to know this circuit identifier; only Bob associates $C_{AB}$ on his connection with Alice to $C_{BC}$ on his connection with Carol.) When Carol

responds with a created cell, Bob wraps the payload into a relay extended cell and passes it back to Alice. Now the circuit is extended to Carol, and Alice and Carol share a common key $K_2 = g^{x_2 y_2}$. This is illustrated in Figure 3.5 below.

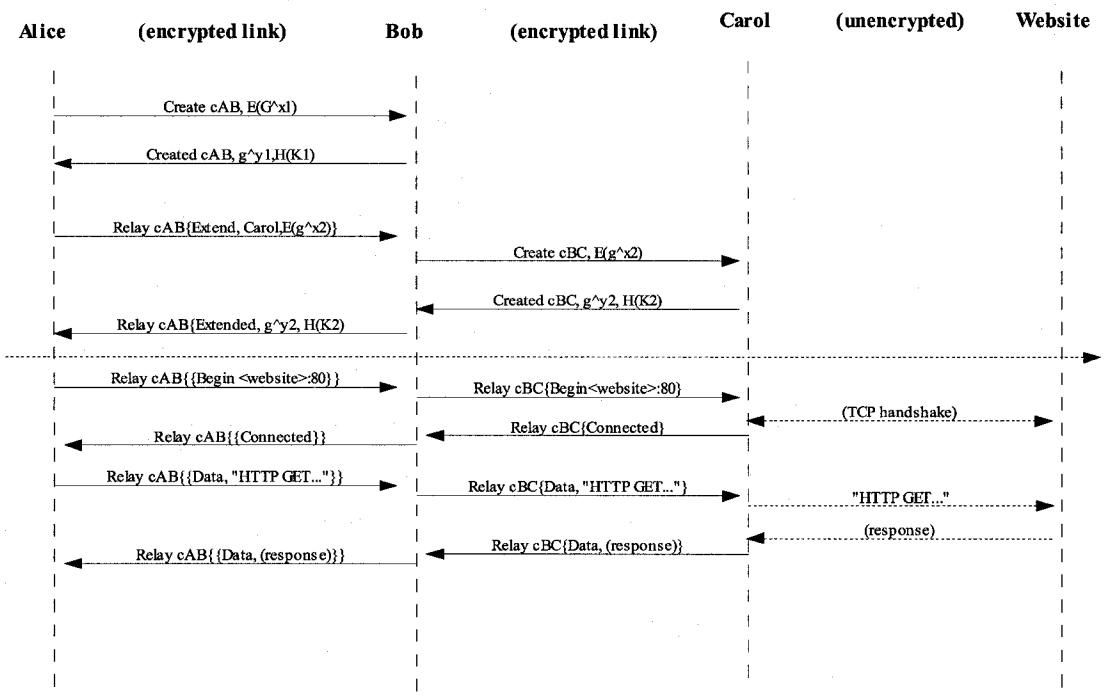


**Figure 3.5 - Alice builds a two-hop circuit and begins fetching**

To extend the circuit to a third node or beyond, Alice proceeds as above, always telling the last node in the circuit to extend one hop further.

### 3.4.1.1 Identity Protection

Tor is a network of virtual tunnels that allows people and groups to improve their privacy and security on the Internet. Tor helps to reduce the risks of both simple

and sophisticated traffic analysis by distributing transactions over several places on the Internet, so no single point can link a specific sender node to its intended destination.

Tor focuses mainly on protecting the transport of data [Tor06]. It does, however, still provide a degree of anonymity. Tor allows clients to choose between various configuration options that can help increase their anonymity. For example, clients concerned about request linkability can rotate circuits more often than those concerned about traceability. Allowing choice may attract users with different needs; but clients who are in the minority may lose more anonymity by appearing distinct than they gain by optimizing their behaviour [ADS03].

As Tor incrementally creates a private pathway or circuit of encrypted connections one node at a time, and each node along the way knows only the immediately previous and following nodes in the circuit, no individual Tor node knows the complete path that each fixed-sized packet (or cell) will take. Thus, neither an eavesdropper nor a compromised node can see both the connection's source and destination. Later requests use a new circuit, to complicate long-term linkability between different actions by a single user [Din04].

Tor only minimally prevents time-based attacks, such as end-to-end timing correlations. An adversary watching patterns of traffic at the sender and the receiver will be able to confirm the correspondence with high probability. The greatest protection currently available against such confirmation is to hide the connection between the onion proxy and the first Tor node, by running the OP on the Tor node or

behind a firewall. This approach requires an observer to separate traffic originating at the onion router from traffic passing through it: a global adversary can still do this, but it might be beyond a limited adversary's capabilities.

None of the systems based on onion routing provide receiver anonymity, however, the receiver cannot resolve the sender of a particular message since messages take different, potentially randomly chosen, routes through the network [SBS02]. Thus, while there is no receiver anonymity provided, there is the notion of sender-receiver unlinkability.

#### 3.4.1.2 Data Confidentiality

Data confidentiality is provided by the layered encryption architecture of onion routing. This does leave the contents of a message exposed on last hop, when the message is forwarded to the destination in plaintext. Thus an adversary would be able to link a specific node (the receiver) to specific content within the message by observing this final hop through an eavesdropping attack. This final hop is also a mounting point for an attack on data integrity, as the message is visible in plain text.

#### 3.4.1.3 Data Integrity

In terms of data integrity, because the original Onion Routing design used a stream cipher without integrity checking, traffic was vulnerable to a malleability attack: though the adversary could not decrypt cells, any changes to encrypted data would create corresponding changes to the data leaving the network. This weakness allowed

an adversary who could guess the encrypted content to change a padding cell to a destroy cell or change the destination address of a cell. Because Tor uses TLS on its links, eavesdropping adversaries cannot modify data through malleability attacks.

### 3.4.2 SSMP: Secret-Sharing-based Mutual anonymity Protocol

Secret-sharing-based Mutual Anonymity protocol (SSMP) [HLX+05] provides both sender and receiver anonymity, in addition to communication security. In SSMP, the idea of a secret sharing scheme is employed to guarantee anonymous query issue, and the introduction of the information dispersal algorithm (IDA) [Rab89] together with the onion routing method is used to achieve a complete reply-confirm interaction between initiators and responders. The main advantages of this protocol include a high degree of anonymity and low cryptographic overheads compared to those used in previous mutual anonymity protocols [HLX+05].

Shamir's secret sharing scheme is used in SSMP to distribute the Data Encryption Standard (DES) key. For the purpose of space efficiency, an information dispersal algorithm is used to split the requested files. First introduced by Rabin [Rab98] the Information Dispersal Algorithm (IDA) is similar to Shamir's scheme, as an $(m,n)$ IDA intends to distribute information $s$ among $n$ processors. The recovery of the information is available if the collection of shares is up to $m$ $(1 < m < n)$. The difference between IDA and Shamir's scheme is that the length of each distributed

fragment of IDA is not $|F|$, but $\frac{|F|}{m}$, where $|F|$ is the file size. Hence, IDA is more space efficient than Shamir's scheme. As the basic IDA does not provide protection from adversaries or security for the data, it is replaced with a secret information dispersal algorithm SIDA [Kra94], which operates in a similar fashion.

SSMP is conducted in the following steps. First, the initiator $O$ randomly selects a list of neighbouring peers, $r_1, r_2, ..., r_n$. It generates a random DES key $T$, which is used to encrypt the file, $f$. The encrypted file is then partitioned into $n$ fragments using SIDA, and the DES key is partitioned into shares using Shamir's $(k,n)$ Secret Sharing scheme. The triples (of a fragment of the file, a fragment of the key, and $k$) are distributed to $n$ pieces $h_i$. Peer $O$ then randomly chooses neighbouring peers $r_i$, and send $h_i, i = 1..n$ to them, respectively. Secondly, when O's neighbours receive a new query share, they flood these shares with a certain probability. If not flooding, they simply forward query shares to one of their neighbouring peers except O.

Once a peer receives a query share, it compares it with its locally received share query, $sq$, in the switch table. If the share belongs to an existing query group, it stores this share information to the same label subset; otherwise it establishes a new subset with the new query share and puts this share in.

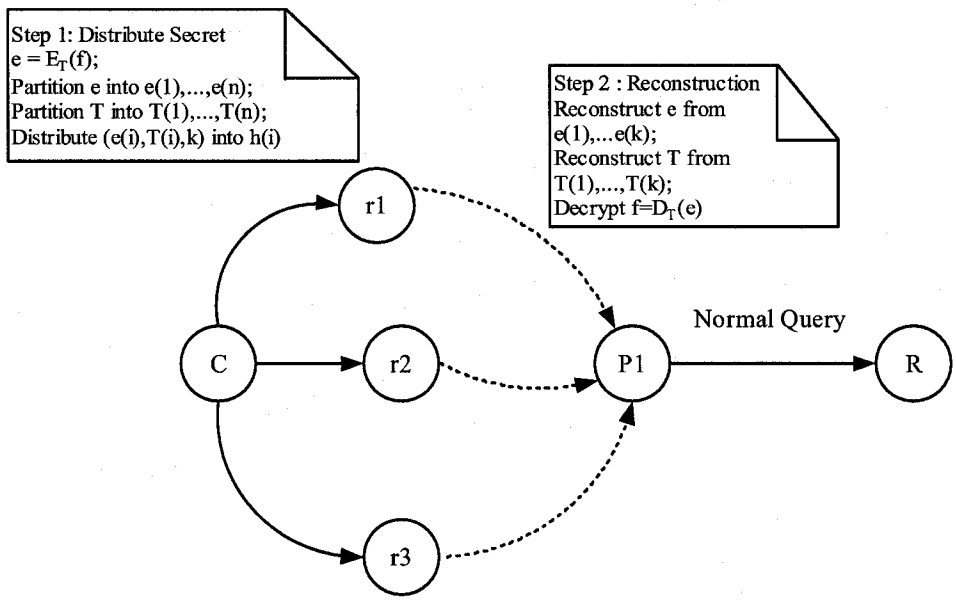


**Figure 3.6 - Share flooding**

Let $P_1$ be one of the peers that collects a sufficient number of shares of the query to recover the file, $f$. Given k or more shares, the peer can reconstruct the encrypted file and the encryption key. Used these pieces in conjunction, the peer can now decrypt the file. These two steps are illustrated in Figure 3.6 above.

When a peer $R$ is able to provide the requested file $f$, it builds two anonymous paths in advance based on the bidirectional onion protocol, a forwarding path and a returning path. The reply message includes an index of desired files. When the reply packet reaches $P_1$, $P_1$ first decrypts the cipher and gets the share query and file index. It then checks the share query in its local switch table. Before sending it back to peer $O$, $P_1$ should organize all received replies to a single packet and use the public key of peer

$O$ to encrypt it. Then $P_1$ randomly chooses a return peer from its switch label according to the share query. In next hop, the receiver checks the share query in its local switch table. If the reply packet does not belong to it, it forwards this packet again to the next randomly chosen peer marked with the share query, until the packet reaches peer $O$.

There could be more than one peer who can recover the file, $f$. At the same time, peer $O$ may receive more than one reply from certain peers delivered by those $P_i$. Peer $O$ picks one as a responder, $R$.

$R$ gets the request-confirm packet from the last return path onion peer. It makes use of SIDA by selecting a random symmetric key and encrypting the file with it. It then uses the public key of peer $O$ to encrypt this symmetric key. The encrypted file is then partitioned into $m$ fragments using a $(k,m)$ information dispersal algorithm. The encrypted symmetric key is then partitioned into $m$ fragments using Shamir's (k, m) scheme. These tuples are then packaged into packets and $R$ randomly chooses $m$ of its neighbours (if $R$ 's neighbours are less than $m$, $R$ can send multiple packets to the same neighbour), and sends the above packets to these neighbours. The packet is then forwarded onward in the same manner described previously.

When peer $O$ receives k or more shares, it can reconstruct the key and encrypted file, and use these pieces to decrypt the file into something comprehensible.

### 3.4.2.1 Identity Protection

When the initiator is ready to issue a query, it first distributes the requested file ID $f$ into $n$ pieces of secret shares using Shamir's secret sharing scheme. It then sends out the split secret shares to some random neighbours. Its neighbouring peers flood the fragments with a certain probability. When any peer, $P_i$, $1 < i < n$, collects $k$ or more shares, the plain query information is recovered, and $P_i$ floods the query message into the P2P system. Thus, determining the initiator of a request is difficult, but choosing random neighbours to whom the shares are sent may be vulnerable to a statistical attack, as the number of shares and their distribution would vary. As the size of the network grows, the probability of determining the initiator decreases.

Once a peer is able to provide the requested file, it creates two onion paths: one is for sending the reply information back to the initiator through a peer node; and the other one is built as an anonymous path for the initiator to return the confirmation packets. Eventually the chosen responder peer that receives the confirmation will split the file into $m$ fragments using SIDA scheme and deliver the requested data back to the initiator.

The probability of an initiator to correctly guess the responders identity is low within SSMP. The reply of the query is forwarded by intermediary peers. If the responder is not the final intermediary peer, its identity is hidden by the onion path. A

responder also has a low probability of correctly identifying the initiator's identity, thus giving unlinkability between the two.

### 3.4.2.2 Data Confidentiality

Data confidentiality is also partially achieved through the use of secret sharing. As a request is split into shares, peers must collect enough of these shares to reconstruct the request. As Onion paths are used to reply to these requests, data confidentiality is provided through the layered encryption of onion routing, protecting against simple eavesdropping attacks on the links between Onion Routers. As with other systems analyzed in this chapter, the use of Onion Routing does leave the contents of the messages exposed in plaintext on the final hop of the onion path, exposing the receiver. The requested file is sent to the originator using a SIDA scheme. So as with the request, a minimum of k shares need to be collected in order to reconstruct the file.

### 3.4.2.3 Data Integrity

Within SSMP, data integrity is also achieved through the use of secret sharing schemes, as the individual shares need to be intact and unmodified in order to properly reconstruct the original secret. Any modification to the shares would make the original secret incomprehensible once reconstructed, making it resilient to malleability attacks.

Therefore, SSMP is able to protect file information from malicious entities. It does, however, require that k shares of a request are collected before the request is

handled by any peer. This may take a long period of time, which would be undesirable to users of the system. In addition, there is no effort to try and separate the paths or locations of the shares, which may make it easier for an adversary to collect the required number of shares to reconstruct the original secret.

## *3.5 Ants-Based*

### 3.5.1 MUTE: Simple, Anonymous File Sharing

Developed by Jason Rohrer in 2003, MUTE [Mute06] is a secure, anonymous, and distributed communications framework. Mute routes all messages through a network of neighbour connections, using an ant-inspired algorithm. Nodes within the network use virtual addresses and encrypt all traffic between them.

MUTE File Sharing uses distributed search based on controlled flooding to locate files by name based on free-form query strings. While many other P2P networks control flooding using a TTL scheme, MUTE introduces a more effective and scalable mechanism called utility counters (UC). Utility counters (UCs) are a simple mechanism that limits how far a query travels based on both the branching factor at each query hop and the number of results generated by the query.

Each search query is tagged with a UC, and the UC starts out at 0. Search queries are forwarded with a "send to all neighbours" scheme just like they are in a TTL network, and the UC on a query is modified before forwarding in two different ways. First, if a node generates results for a given query, the node adds the number of

results generated to the UC before forwarding the query. Second, the number of neighbours that a node plans to forward the query to is added to the UC. For example, if a node receives a query with a UC of 25, and it generates 10 results, then it first increases the UC to 35. If it plans to forward the query to 6 of its neighbours, it increases the UC again to 41, and then it forwards a query to its neighbours.

Nodes in a UC-based network each enforce a UC limit: if a search query's UC hits this limit, it is dropped without being forwarded to neighbours. For example, if the UC limit is 100, a node that drops a query can surmise one of the following: that the query has generated at least 100 results, that the query has passed through hops with a total branching factor of 100, or that the query has reached the limit through a combination of both factors (for example, that it has generated 75 results and has passed through hops with a total branching factor of 25). Thus, the UC limit approximates an accurate control over how much utility a search sender can glean from the network with a single query [Mute06].

Since MUTE users are anonymous, none of the nodes in the network know exactly where to find a particular recipient (or, more precisely, which computer a particular recipient is using). Like ants that are unaware of the overall environment layout, MUTE messages must be directed through the network using only local clues.

One modification to the generic Ant Colony Optimization (ACO) [DC99] meta-heuristic made in MUTE is the use of two types of pheromone, instead of just one. When ants leave the nest in search of food, they walk randomly, leaving trails of home-

finding pheromone as they go. When an ant finds food, it picks up a piece and follows its home-finding trail back to the nest, leaving a trail of food-finding pheromone as it goes. If a wandering ant ever encounters a food-finding trail, it follows that trail to the food source, leaving more home-finding pheromone as it goes.

MUTE's ant-routing also discovers "short" paths between a sender and receiver. However, the fast roundtrip time is what makes a path attractive to MUTE messages. In low-traffic situations, the fastest path will often be the shortest one with the fewest hops. As traffic increases, however, a short path may become slower if it is overloaded, and a longer path with less traffic may be faster. Thus, using only local routing clues, MUTE can automatically balance load throughout the network and avoid congested routes.

MUTE uses a probabilistic algorithm for back-route selection. For each recently seen sender address, MUTE keeps a queue of pointers to recent neighbour connections on which messages from that sender have been received. If three sender addresses (Alice, Bob, and Eve) and five neighbours (numbered 1 to 5) are being tracked, the routing table would be as shown in Table 3.4 below.

Looking at the pattern in this table, it can be seen that most messages from Eve came through neighbour 4. Thus, the locally optimal choice when routing messages back to Eve would be to send them through neighbour 4. Briefly connecting this table with the ant motivation, the connection with neighbour 4 has the strongest "from Eve" pheromone scent.

If a message addressed to Alice is received, Alice's column from the table is selected, then one of the neighbour entries is selected uniformly at random, and the message is routed through that neighbour. This scheme prefers to route messages through the most popular paths, but occasionally routes messages through less popular paths [Mute06].

**Table 3:4 - MUTE Routing Table**

| From: | Alice | Bob | Eve |
|---|---|---|---|
| **1** | 1 | 5 | 4 |
| **2** | 2 | 1 | 4 |
| **3** | 2 | 2 | 4 |
| **4** | 1 | 1 | 5 |
| **5** | 4 | 1 | 4 |

A queue of the last $N$ unique seen from-addresses is maintained. Seeing a "from" address again moves that address to the head of the queue. When the queue fills up, addresses are dropped from the tail of the queue.

For each address, a queue of the last $M$ channels on which messages from that address have arrived is stored. This list may contain duplicates or old channels that no longer exist. Each time a message from the address is received, a reference to the receiving channel is added to the head of the queue. As the queue fills up, channel references are dropped from the tail of the queue.

There is thus a two-level queue, with a maximum of $N$ entries in the first level, and $M$ entries in each second level queue, for a total maximum of $N \cdot M$ entries. As these are user definable values, the user may therefore in part determine the amount of storage overhead they are willing or able to maintain.

To route a message to a particular address, a node simply picks a channel at random from the corresponding from-address queue and send the message through that channel. With a small probability, a selection from all channels at random with equal probability (uniformly) may be picked.

If the message's to-address is not in the from-address queue, the message is broadcasted out on all of the known channels. No queue entry is added for an address until a message is processed with the address as a from-address. This algorithm has the following properties [Roh03]:

- Picking from the address's channel queue at random (with a queue that contains repeats) creates a probability distribution over the channels. Channels that have recently seen a lot of messages from the address have a larger weight in this distribution.
- Switching to a uniform distribution periodically makes the system robust against all message traffic for an address being routed through the same node (which may maliciously start dropping these messages after routing them correctly for a while). If a "tried and true" route goes bad, a new one will be discovered when a uniform routing distribution is used.

- Defaulting to broadcast for unfamiliar addresses effectively explores the network for routes, since response messages will be routed back through this node and populate the routing table.

### 3.5.1.1 Identity Protection

The main way that MUTE protects a users' privacy is by avoiding direct connections between senders and receivers. Recall that other P2P networks deliver a particular request to many nodes without making direct connections to any of them but, when it comes time to transfer a file, these networks use direct connections. An analysis of the level of anonymity provided by MUTE can be seen in Table 3.5 below [HLX+05}.

**Table 3:5 - Anonymity Provided By Ant-Based Routing**

| | Ant-Based |
|---|---|
| Sender anonymous to Global Adversary | No |
| Responder anonymous to Global Adversary | No |
| Sender anonymous to Responder | Probable Innocence |
| Sender anonymous to Node | Probable Innocence |
| Responder anonymous to Sender | Probable Innocence |
| Responder anonymous to Node | Probable Innocence |
| Sender-Responder unlinkable to Node | Probable Innocence |
| Sender-Responder unlinkable to Global Adversary | No |

MUTE routes all messages, including search requests, search results, and file transfers, through the network of neighbour connections. Thus, while a node is aware of the IP addresses of its neighbours, it is not aware of the IP address of the node that it is downloading from. It is in this manner that the notion of sender-receiver unlinkability is established.

To ensure both sender and receiver anonymity, MUTE employs a 3-stage probabilistic approach to manipulating its version of TTLs, utility counters (UC), in order to prevent Time-To-Live attacks. We will first look at how sender anonymity is protected.

Receiving a query with a UC of 0 from your neighbour implies that your neighbour sent that query. For this reason, MUTE uses a hybrid approach for search flooding. The idea is to pass a random number generator along with each message and have nodes manipulate the state of the random number generator as they decide to forward or drop a message. A node will forward the same generator with the same state on to each of its neighbours, so the neighbours will all make the same decision about whether to forward or drop the message. Thus, we have all nodes at a given tree level making the same probability computation, ensuring that the message will be dropped eventually instead of traveling forever.

The hybrid approach involves starting each search as a random-depth flood, as described above, with each node manipulating a random number generator that is attached to the search query. During the random-depth flood portion of a query's life,

the UC is left at the starting value of 0. Once a particular result comes out of the random number generator (a trigger value), the message changes modes into a UC-limited flood. Thus, a message travels for a random number of hops with no UC limits, and then travels further with UC limits.

If each search from a given node uses a different random depth flood, then searches generated by an immediate neighbour will have different statistical properties than searches generated by more distant nodes: if a series of searches is generated by a neighbour, a given will always be part of the random flood, but if the searches are generated by a more distant node, the random flood will sometimes be over by the time the search reaches this node. To prevent this kind of statistical analysis from being performed by an adversary across multiple searches, a node must use the same starting generator state repeatedly for each search request that a given node sends. Thus, a given node always starts a search with the same a flood depth, though this depth cannot be determined by other nodes. Since the same depth flood is used for each search, the given node will always be part of the flood or always not be part of the flood, independent of whether the sender is the node's neighbour.

The above scheme ensures that an adversary cannot determine the origin of a search request, but it does nothing to guarantee the anonymity of nodes that are responding with search results.

No deterministic dropping scheme can be used to protect responder anonymity. An adversary can simulate a deterministic scheme ahead of time and find a mechanism

that would force its neighbour to process and drop a message. Thus, a truly probabilistic scheme in which each node makes an independent probability calculation is needed, as an adversary could not manipulate such calculations.

Each node, upon start-up, decides randomly whether it will drop messages or pass them on after the UC limit has been hit (termed tail chain messages), and it behaves the same way for every tail chain message that it receives. In order to introduce more doubt, this tail chain behaviour is extended to introduce branching. Thus, each node decides on its behaviour at start up in a probabilistic fashion to ensure that there are enough nodes within the network that will drop messages at this stage (so that a message will not be forwarded indefinitely).

Figure 3.7 shows a distribution tree that uses both of the discussed anonymizing schemes. The circular nodes are passing the message in random-flood mode, which continues for two hops in the tree. The triangular nodes are passing the message in UC mode, adding to the UC at each hop depending on the number of results generated and the branching factor. Once the UC limit is hit, the message switches into chain mode. Each square node either passes or drops all chain-mode messages that it receives, so each chain eventually reaches a drop-node and ends.

This scheme falls victim to the same kind of statistical analysis that we mentioned in the previous section. If the tail lengths are determined by flipping independent coins at each hop, we expect more length-1 tails than length-2 tails. An adversary can send searches repeatedly into a tail and measure the frequency of various

search results. The most frequent results must be coming from the length-1 tails, so they must be from the adversary's neighbour. To thwart this kind of analysis, we need to ensure that the each particular tail is always the same length, though we must prevent the adversary from determining or influencing that length. Therefore each node also picks one neighbour at start up and, if it passes tail chain messages, always passes them through that neighbour, as long as that neighbour remains connected. Tail lengths will change slowly over time as the network topology evolves, but gathering meaningful statistical information from the resulting tail lengths will be difficult, especially since an adversary has no information about how the topology beyond a given neighbour node is changing.

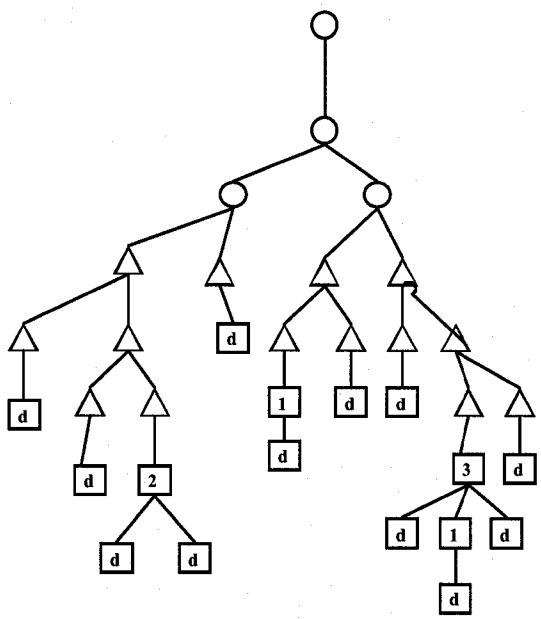


**Figure 3.7: Distribution Tree**

### 3.5.1.2 Data Confidentiality

MUTE provides only limited means of data confidentiality. MUTE nodes use encryption, but only to secure the direct connections to their neighbours, which is known as link-encryption. For a given MUTE route, each leg of the route is encrypted separately. When a node receives a message from one of its neighbours, it essentially decrypts the message, reads it, and then re-encrypts it to route it onward through another neighbour. Therefore each MUTE node that routes a message has full access to the unencrypted contents of the message. Thus, it would be possible for these intermediary nodes to record the contents of messages that pass through them. Worse yet, it is even possible for intermediaries to modify the contents of messages before passing them on.

End-to-end encryption would solve this problem of intermediate nodes having access to the contents of a message, but the creator of MUTE, Jason Rohrer, claims that it is impossible to fix this problem in an anonymous network. He claims that "end-to-end encryption actually would work if senders had a secure way to obtain receiver keys. Of course, in the MUTE network, the sender and receiver are anonymous. How is the sender going to obtain a key from an anonymous receiver before commencing communications with that receiver? To preserve their anonymity, the sender and receiver cannot connect to each other directly. Thus, the receiver's key must somehow be transferred through the MUTE network itself." [Roh04]

#### 3.5.1.3 Data Integrity

MUTE does not provide any sort of mechanism to ensure data integrity. Thus any data that is modified between a source and destination node would not be detected by any means within the system.

## *3.6 Summary*

This chapter, dealt with the state of the art in anonymous networks and adaptive routing. It introduced basic building blocks for anonymity, including Onion Routing, and swarm based routing, along with implemented systems that built upon these ideas. While these systems do provide various aspects of secure communication in anonymous settings, none address them all. Where some are focused on the protection of the identity of participants, others are more concerned with maintaining the integrity or confidentiality of the data within the system.

All the presented protocols have weaknesses, and are often intended to simply solve a particular issue. As can be seen in Table 3.6, no single design provided a complete notion of anonymity For example, systems based on either Onion Routing or MIX networks failed to provide receiver anonymity, while MUTE failed to provide any mechanism for data confidentiality or integrity.

**Table 3:6 - Anonymity Provided By Different Designs**

| Legend<br>Prob. I. = Probable Innocence<br>B.S. = Beyond Suspicion | Freenet | MIX networks | Onion Routing | Ant-Based |
|---|---|---|---|---|
| Sender anonymous to Global Adversary | No | No | No. | No |
| Responder anonymous to Global Adversary | No | No | No | No |
| Sender anonymous to Responder | Prob.I. | B.S. | B.S. | Prob. I. |
| Sender anonymous to Node | Prob. I. | No | No. | Prob. I. |
| Responder anonymous to Sender | No | No | No | Prob. I. |
| Responder anonymous to Node | No | No | No | Prob. I |
| Sender-Responder unlinkable to Node | Prob. I. | B.S. | B.S. | Prob. I. |
| Sender-Responder unlinkable to Global Adversary | No | B.S. | B.S. | No |

The following chapter will present a protocol that addresses the problem statement presented in Chapter 1. It will aim to provide anonymity for both the sender and receiver, while also providing sender-receiver unlinkability. It will be resilient against several attack types, including time-to-live attacks, statistical attacks, eavesdropping attacks and malleability attacks. The protocol will be shown to provide all facets shown in Table 3.6, which none of the systems reviewed in this chapter accomplish. As with some of the systems presented in this chapter, the proposed protocol will also incorporate the use of threshold schemes to help attain both data confidentiality and data integrity.

# Chapter 4

# Securing Communication in an Anonymous Network

This chapter will introduce the solution to the problem statement presented in Chapter 1. It will begin with an overview of the solution, looking at how it addresses the problem statement. It will then move into several sections that will detail the various main aspects of the solution design. The first of these sections will look at what happens when a node joins a network, examining in detail how the node identifiers are generated, how a node chooses its neighbour connections within the system and what happens to these connections when a node leaves the network. After this, the different routing stages and message types used in the system will be identified. This will be followed by an examination of the structure and purpose of the pheromone tables located at each individual node. The use of these pheromone tables in the routing protocol will then be discussed, highlighting the formulas used to make hop by hop routing decisions and how these formulas make use of competing pheromone types to create disjoint paths between source and destination node pairs. This section will also consider the strategy employed to update the entries in the pheromone tables, rewarding paths that are deemed to be of high quality. Finally, the mechanisms used to ensure sender and receiver anonymity will be presented.

## 4.1 Protocol Overview

Recall that the problem statement presented in the previous chapter was defined as "the creation of an adaptive routing protocol to achieve identity protection, data confidentiality and data integrity in an anonymous overlay network." In order to achieve this goal, various anonymity building blocks that were presented in Chapter 2 will be employed, including the use of secret sharing schemes, some of the core concepts of MIX networks and Onion Routing, along with ant-based routing. The protocol does not rely on any type of centralized authority figure, nor does it require any information outside the scope of the local view of a single node.

The protocol aims to provide network security within the context of an anonymous network. It is the notion of anonymity that provides the stated goal of identity protection. Anonymity is provided through the use of unique identifiers assigned to each node. Messages will be routed in a hop by hop manner using these identifiers, as opposed to two nodes creating a direct connection through the use of IP addresses, as is the case in most traditional P2P networks. In this manner, node identifiers will not be able to be linked to a particular IP address. Avoiding direct connections between communicating nodes also provides a measure of anonymity, as the nodes taking part in the communication have no idea who the node they are communicating with is, apart from the node identifier. In addition to these measures, which provide sender-receiver unlinkability, and both sender and receiver anonymity from a global viewpoint, extra measures are included to protect both sender and

receiver anonymity in a local context. As neighbouring nodes could draw conclusions about a node's identity based on the value of the TTL field of a packet by applying time-to-live attacks, extra measures are taken to ensure that these types of attacks can not take place. These measures include additional hops for a packet that are not reflected by the TTL value, and that can not be modified or exploited by an adversarial node through statistical attacks. These techniques are discussed in detail later in this chapter.

Data confidentiality is provided through two separate mechanisms in order to create a multi-tiered notion of confidentiality, while avoiding a single point of failure. The contents of a message are protected from an eavesdropping attack through the use of link encryption. This still exposes the contents of a message at all nodes along a messages path, however. In order to further protect the contents of a message from these intermediary nodes and malleability attacks, the use of Shamir's secret sharing scheme is utilized. The application of the scheme involves splitting apart the message contents, and then routing the shares of the message along disjoint paths to the intended destination, where the message can be reconstructed. Using this technique, intermediary nodes do not see enough shares of the message to glean any meaningful information about the message as a whole, even though they are able to see the contents of the messages they process. In order to achieve this routing through disjoint paths using an ant-inspired routing technique, the notion of multiple pheromone types is used in this protocol. These competing chemical signals attract packets of the same

type, and repel packets of differing types. Thus, each share of a secret is assigned a different pheromone type, leading the shares to travel along differing paths. The standardization of these pheromone types across the network allow the strengthening of trails of all types across the network.

The notion of data integrity is also provided through the use of Shamir's Threshold Scheme. If any of the shares are modified, then the original secret (the message) will not reconstruct properly. Thus, it would be impossible for an adversarial node to modify the contents of a message in any predictable manner, without first intercepting sufficient shares, which routing the shares across disjoint paths prevents.

As can be seen in Figure 4.1 below, when a node first joins the network, it first follows an initialization mechanism. After this initialization phase, a node is active within the network. At this point, the node may search for a resource, receive and process other messages from within the system, or leave the system. While a node is active in the network, it may receive messages concurrently with searching for resources. The following sections will examine these main blocks of the figure in greater detail.

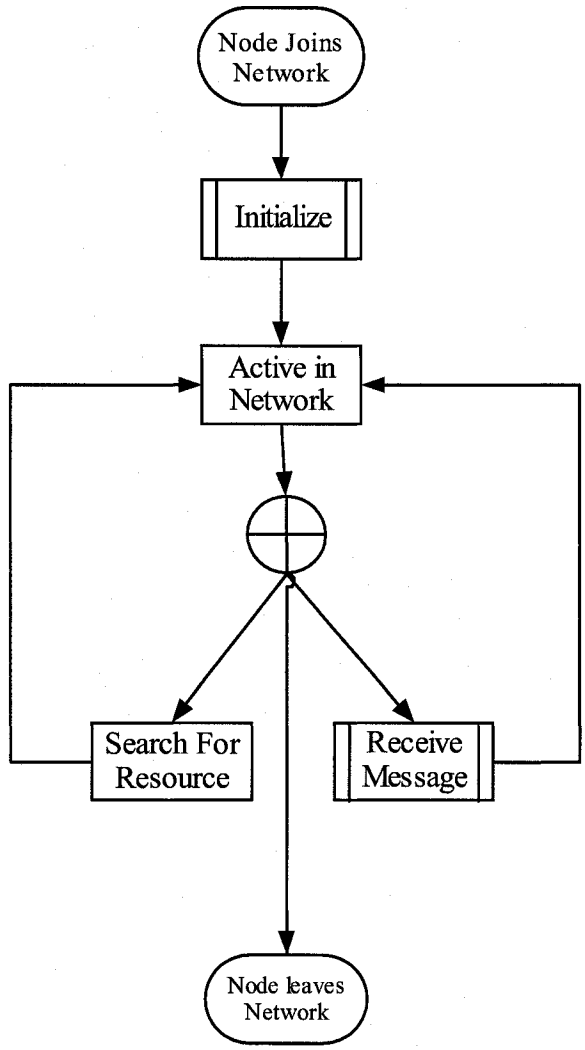


**Figure 4.1 - Network Node Lifecycle**

## *4.2 Initialization*

It is within the initialization phase that a node prepares itself for communication within the system. The figure below outlines the major steps of this phase.

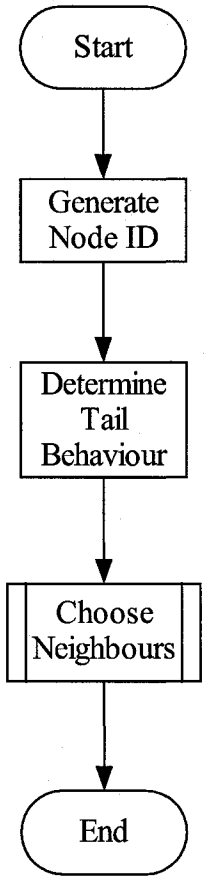


**Figure 4.2 - The Initialization Phase**

A node first generates its node identifier, or the pseudonym that the node uses within the system, which is detailed in Section 4.2.1. It then determines it tail behaviour. This is a concept used to help ensure the notion of receiver anonymity, and will be discussed in detail in Section 4.5.2. The node subsequently starts a process to choose a group of neighbouring nodes. This algorithm is introduced in Section 4.2.2. Once the neighbour set has been chosen, the initialization phase is complete, and the node is free to start searching for resources within the network..

### 4.2.1 Node Identifiers

Each node in the network is identified using a unique identifier generated upon joining the network. This pseudonym will be used to route messages to a particular node. No node within the network should be able to link a node identifier to a specific IP address, with the exception of its own node identifier and IP address.

To create unique identifiers the hashing function SHA-1 is employed, with the input data consisting of the current time and the node's Media Access Control (MAC) address. This will generate an id that is unique in both space and time, under the assumption that only one peer is running on a single physical device. There is no danger, in terms of security and privacy, of incorporating the MAC address, as the hash function is one-way, so an adversary would not be able to learn any information about the network participants. By having a new identifier generated each time a node joins the network, it will be more difficult to link any specific pseudonym across a series of transactions over time.

### 4.2.2 Choosing Neighbours

When a node joins the network, neighbours will be chosen in a method to create a proximity based overlay design. It has been shown that better peer selection can lead to improved performance [CCR04], thus some structure will be added to the creation of the overlay. This proximity based method attempts to better reflect the underlying network topology by choosing peers that are close in the real world. This reduces

average latency between peers, and is likely to boost the average bandwidth, since clients on the same subnet are more likely to be paired [Qur04]. While proximity neighbour selection has been shown to be a particularly efficient structuring mechanism, it has to continuously adapt the virtual network to the physical network to maintain its effectiveness, incurring additional control traffic. Using an ant-based routing mechanism provides continual adaptation without incurring much of the extra control traffic.

The idea is to build the probabilities for node x selecting a node as a neighbour so that if the round trip time (RTT) between x and y is lower than between x and z, than y should have a greater probability of being selected than z as a neighbour of x. Upon joining the network a node will broadcast hello messages, wait a timeout interval, and then compute the probabilities based on the RTT, and then choose up to a limit m neighbours. This will allow a set of neighbours within a given "distance" to respond, and a pseudo-random set to be chosen as peers, following the idea that nodes should preferentially select those other nodes as peers with which it has low observed communication latencies. This is designed to be a probabilistic model, so that an adversary gains no advantage through the knowledge of a deterministic selection algorithm. This behaviour can be seen in more detail in the pseudocode in Figure 4.3 below.

As can be seen from the pseudocode, a node broadcasts hello messages, and then collects replies during a timeout interval. If the minimum number of neighbours is

received, the node computes the probabilities for these responses as described below, and otherwise it clears all responses and sends a new round of hello messages.

```
function send_hello
    for each physical neighbour
        send hello message
    end for
    set hello timeout
    collect replies in <response time>
end send_hello


function hello_timeout
    if min number of responses received
        complete_initialiazation
    else
        clear replies
        send_hello
    end if
end hello_timeout
```

**Figure 4.3 - Hello Message Pseudocode**

If the minimum number of replies has been received, the node first sorts these replies based on the RTT. This aids in the efficiency of the probability calculations. It then calculates the probability $p_i$ of selecting a reply based on the RTT. The node builds a table of normalized, cumulative probabilities. This cumulative probability function (and it components) are illustrated in Equations 4.1 an 4.2 below. In these equations, $rtt_i$ refers to the RTT of the $i^{th}$ response, and $N$ is the number of responses received. It then generates a random number, which will be the number of neighbours

selected using these calculated probabilities. For each of the neighbours selected using this method, the node generates another random number and uses it to index into the structure of cumulative probabilities. The node at this index is selected as a neighbour, and is added to the pheromone table. This node is then removed from the collection of received responses and the probability structure is recalculated, as the values need to be normalized again.

**Equation 4:1 - Calculate Cumulative Probabilities**

$$p_i = \frac{\left(rtt_i\right)^{-1}}{\kappa} + p_{i-1} \;, p_0 = 0$$

**Equation 4:2 - Cumulative Probability Denominator Function**

$$\kappa = \sum_{i=0}^{N} \left(rrt_i\right)^{-1}$$

For example, if a node receives four separate replies with RTTs of 2, 3, 4, and 3 seconds, $\kappa$ would be equal to the sum of the inverses of these values, namely 1.416. To find the probability for each response, Equation 4.1 is employed. Thus, the four probabilities would be:

- $p_1 = \frac{(2)^{-1}}{1.416} + p_0 = 0.3529 + 0 = 35\%$
- $p_2 = \frac{(3)^{-1}}{1.416} + p_1 = 0.2352 + 0.3529 = 59\%$

- $p_3 = \frac{(4)^{-1}}{1.416} + p_2 = 0.1765 + 0.59 = 77\%$
- $p_4 = \frac{(3)^{-1}}{1.416} + p_3 = 0.23 + 0.77 = 100\%$

With this list of cumulative probabilities, if the random number generated is 0.65, then the third response is selected, or if the random number is 0.23, then the first response is selected.

At this point, the node uses a different technique to choose the remaining (up to the maximum) number of neighbours. Having a secondary selection mechanism helps to guard against adversarial nodes, using the knowledge of preferential selection based on proximity to the node in question, from cornering or surrounding this node. In this secondary phase, entries from the response set are simply chosen at random. This behaviour is shown in the pseudocode in Figure 4.4 below.

Departures from the network are taken care of through a timeout value known as a lease which associated with an entry in the pheromone table. If an entry in the table is not modified during this lease period the entry is removed from the tables. This type of approach will also handle a broken link situation, as the node associated with the broken link would be removed automatically after a period of inactivity. This also reduces the number of control messages by eliminating the need for any heartbeat or "I'm alive" type messages between nodes.

```
function complete_initialization
    // Sort by time to aid in the probability calculations
    sort(<response time>);
    // calculate the probabilities for the response set
    calculate_cumulative_probabilities

    generate random number r2 between 0 and 3

    for 1 until minNumNeighbours + r2
        generate a random number between 0 and 1
        choose the response with matching probability
        insert this node into pheromone table
        remove node from <response time>
        recalculate cumulative probabilities
    end for

    for minNumNeighbours + r2 until maxNumNeighbours
        //make sure there are still entries in <response time>
        if size of <response time> == 0
            break
        end if
        generate a number between 1 and size of <response. time>
        choose the entry at this index in <response time> as a neighbour
        insert this node into pheromone table
        remove node from <response time>
    end for
end complete_initialize
```

**Figure 4.4 - Neighbour Selection Pseudocode**

### 4.2.2.1 Hello Message Structure

The structure of both the Hello and Hello_Reply messages can be seen in the figures below.

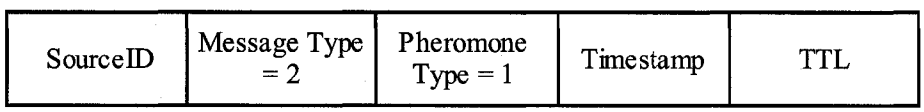


**Figure 4.5 - Hello Message**

| SourceID | Message Type = 2 | Pheromone Type = 1 | Timestamp | Destination ID | TTL |
|---|---|---|---|---|---|

**Figure 4.6 - Hello Reply Message**

Hello messages are flooded from a node to all of its neighbours, who then proceed to flood the node to all of their neighbours (excluding the neighbour they received the message from). This proceeds until the TTL value has expired. Hello replies are sent back to the node identified by the SourceID field in the Hello message, using the Next Hop Probability function introduced in Section 4.4.1.

In comparison to the other packet structures that will be presented later in this chapter, these messages differ in that they have a smaller maximum TTL value. This is done in order to try and localize the responses that a node receives, correlating to the goal of a proximity based neighbour selection method. By having the smaller maximum TTL value, the Hello messages will not propagate as far as other messages in the system, and will thus only reach nodes that are close by. Both of these messages types are assigned a pheromone type of 1, as they are not split into shares, but are still used to update or create entries in neighbouring node's pheromone tables.

#### 4.2.2.2 Neighbour Connections

Neighbour connections in the network are built on top of secure streams that have the following properties:

- RSA public/private keys to exchange secret keys

- AES (Rijndael) 128-bit secret keys used in CFB mode with all-zero initialization vectors to encrypt stream data
- Separate AES keys used for each stream direction
- Fresh AES keys used each time a new stream is established

These connections require that a fresh stream is established for each download.

## *4.3 Generate Search Request*

After completing the initialization phase, one of the options a node has is to generate a search request. If a node does this, the flow of messages within the system can be seen in Figure 4.7 below.

As this figure shows there are four main routing stages that take place: searching for a resource, responding to a search message, requesting a particular resource from a particular node, and sending the requested resource to a particular node. It is important to note that it is possible to respond to incoming messages while searching for a resource, as well as executing multiple searches concurrently. Figure 4.7 only shows the flow between the routing stages for a single search request that has been generated. This section will look at these different routing stages and the packet structures for these messages.

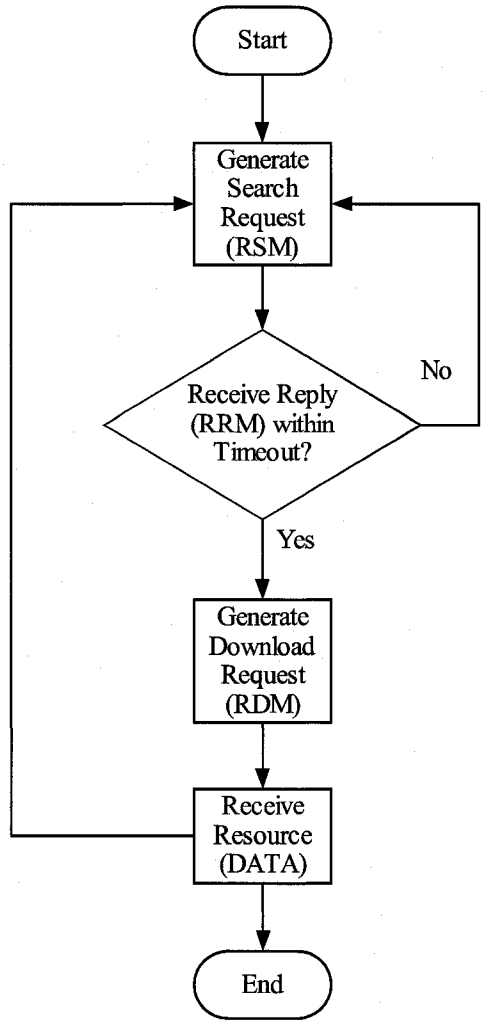


**Figure 4.7 – Resource Search Message Flow**

### 4.3.1 Message Types

Corresponding to the different stages mentioned above, there are four different message types used in these stages: Resource Search Message (RSM), Resource Response Message (RRM), Request Download Message (RDM), and data messages containing the actual resource (DATA). The different packet header structures for

these messages can be seen in Figures 4.8 and 4.9, respectively. The body of these messages varies depending on the type of message. For the RSM, the body contains the name of the resource being searched for, along with an identifier for the particular search request. In the RDM message, the body also contains the name of the requested resource, along with the identifier that was contained in the RSM, and another unique identifier for the response itself. The RRM message body contains this response identifier along with the resource name, and the DATA message bodies contain the actual resource itself along with the response identifier again. The use of these linking identifiers allows nodes to ensure that communication is taking place with a particular node throughout the course of the communication. The identifiers are created by generating a SHA-1 hash of the node's identifier along with the current time. Thus, no other node could generate an identical identifier and intercept the stream of communication. In addition, by carrying this information in the body of the message, an adversarial node would need to collect sufficient shares to reconstruct the message in order to even see what this value is.

| SourceID | Message Type = 3 | Pheromone Type = 1 | Timestamp | TTL |
|---|---|---|---|---|

**Figure 4.8 - Resource Search Message**

| SourceID | Message Type (RDM=4, RRM=5 DATA=6) | Message ID | Pheromone Type | Number of Shares Required | Timestamp | Destination ID | TTL |
|---|---|---|---|---|---|---|---|

**Figure 4.9 - All Other Message Types:**

All of these packets have a higher maximum TTL value than the Hello messages presented previously. In this manner, these message types will propagate further through the network, creating a higher probability of locating a particular resource that a node is searching for. These packets are all routed using the pheromone tables located at each node, choosing a next hop towards the destination in a probabilistic fashion. This technique is discussed in detail in the following section. As with the Hello and Hello_Reply messages, the RSM message is assigned a pheromone type of 1 in order to allow it to be used to update or create entries at neighbouring node's pheromone tables. The RRM, RDM, and DATA messages are broken into shares using Shamir's threshold scheme. These packets have a field indicating the pheromone type of the packet, as the competing pheromone types are used to create disjoint paths. The pheromone types are represented as integer values, and are assigned to shares in the following manner: A node creates as many shares as it has neighbours, and for each of these shares it assigns an incrementing pheromone value. For example, if a node has five neighbours, then it would break the message into five shares, with the first share having a pheromone value of one (1), the second share a value of two (2), and so on up to the fifth share. They also contain a field that indicates how many of these shares need to be collected in order to reconstruct the original message, along with a Message ID field so that shares belonging to the same message can be grouped together for reconstruction. The number required to reconstruct is a percentage of the number of neighbours that a node has. Should this percentage not be a whole number,

the floor of the result is taken as this threshold value. All of these shares have a constant size, as this is a property provided by the threshold scheme. The payload is split into equal sized shares, and the headers all contain identical sized information, creating a packet size that is also identical as a whole. This makes it more difficult for a casual observer to differentiate between packets. Other schemes use message padding to achieve a similar result, but this is not necessary within this protocol as it is an intrinsic property provided by the threshold scheme.

When searching for a resource, a node will flood a RSM to all of its neighbours selected during the initialization phase. This RSM will contain the name of the resource that the node is looking for. When a node receives one of these RSMs, it searches its local drive to see if it has the resource being requested. If it does, it creates an RRM and sends this back to initiating node. The local search process itself is outside the scope of this thesis. If the TTL value on the message dictates that it should not yet be dropped, the node then floods the RSM message to all of its neighbours, except for the neighbour from which it received the message, regardless of whether or not it found the requested resource locally. The RSM is sent as a whole message, in order to facilitate quicker responses to a request for a resource. In SSMP, presented in the previous chapters, sender anonymity was addressed through breaking these search messages into shares, and sending out the shares to neighbouring nodes. Whenever a node collected enough shares to reconstruct the query, it then sent this query into the network as a whole message. In this manner it becomes difficult to tell where a request

actually initiated from. It does, however, introduce a delay between a node requesting a resource, and this request actually being inserted into the system, which is an undesirable property for users of the system. In the protocol presented here, a trade off between responsiveness and the contents of a search request were considered. Recall that every node is assigned a unique node identifier that can not be associated to a particular IP address. Thus, while a resource request message can be linked to a particular node identifier, this node identifier cannot be linked to any particular node. This made the trade off of quicker response time seem appropriate, as there are no linkages created to any particular node. RSMs are assigned a pheromone type of 1.

RRMs are sent using a different technique than simply flooding them to all of the node's neighbours, as in the previous stage, since this message has a specific destination that it is being sent to. Figure 4.10 shows the pseudocode for the algorithm used in this routing stage, termed "share routing". The response is first broken apart into shares using Shamir's threshold scheme, to protect the responding node from adversarial nodes learning what resources it has stored locally. Once the RRM is broken into shares, it is then routed using the values contained within the node's pheromone table as input into a probabilistic function to determine the next hops. This function routes shares across disjoint paths by using the notion of competing pheromone types to deter shares of the same message from using similar hops. The pheromone table structure and contents, along with this probabilistic function will be examined in greater detail in Section 4.4.1. As mentioned previously, the use of a

threshold scheme to break apart the message strengthens the notion of using link-encryption, as even though an intermediary node can see the contents of the packet it receives, this packet contains only one share of the original message, and as thus, does not give the node any information about the actual message.

```
function share_routing
 create message
 break into n shares using Shamir's Threshold Scheme
 for i=1 to n
   assign share(i) pheromone type i
   send share(i) using next hop probabilty function
 end for
end share_routing
```

**Figure 4.10 - Send RRM Pseudocode**

When the original, initiating node receives enough shares to reconstruct a RRM, it determines which node has the requested resource. If the initiating node decides to download the resource from this node, it creates a RDM, and sends this back to the node that created the RRM. This message is routed in the same manner as the RRM. Similarly, when the intended destination receives sufficient shares to reconstruct the RDM, it sends back the DATA message in the same manner, breaking the resource into shares and sending these along disjoint paths selected in a hop-by-hop manner.

As all of these messages are routed in a probabilistic fashion over varying paths, each hop on these paths employs link encryption, and the contents of the messages are concealed (with the exception of the RSM) through the use of the

threshold scheme, it is extremely difficult to create a link between a RSM, RRM, RDM, and DATA message transaction.

## *4.4 Receiving a Message*

As can be seen in Figure 4.11, within all the various routing stages, whenever a node receives a packet, it first checks to see if it has already received this packet, and if so, the packet is dropped. This prevents cycles from occurring in the route. A node maintains a buffer of recently received packets, and checks incoming messages against this buffer to search for duplicate messages. A message is deemed to be a duplicate if it originated from the same source ID, was generated at the same time (based on the timestamp), is of the same message type, and of the same pheromone type as a message contained within the buffer. Messages within the buffer are stored for a period of time and older entries are removed through a garbage collection process. Each time a message is received and stored the garbage collection process is invoked, removing messages which are older than the garbage collection time metric. A node will then update its pheromone table for the received packet (discussed in detail in Section 4.41), and will then handle the packet depending on its type.

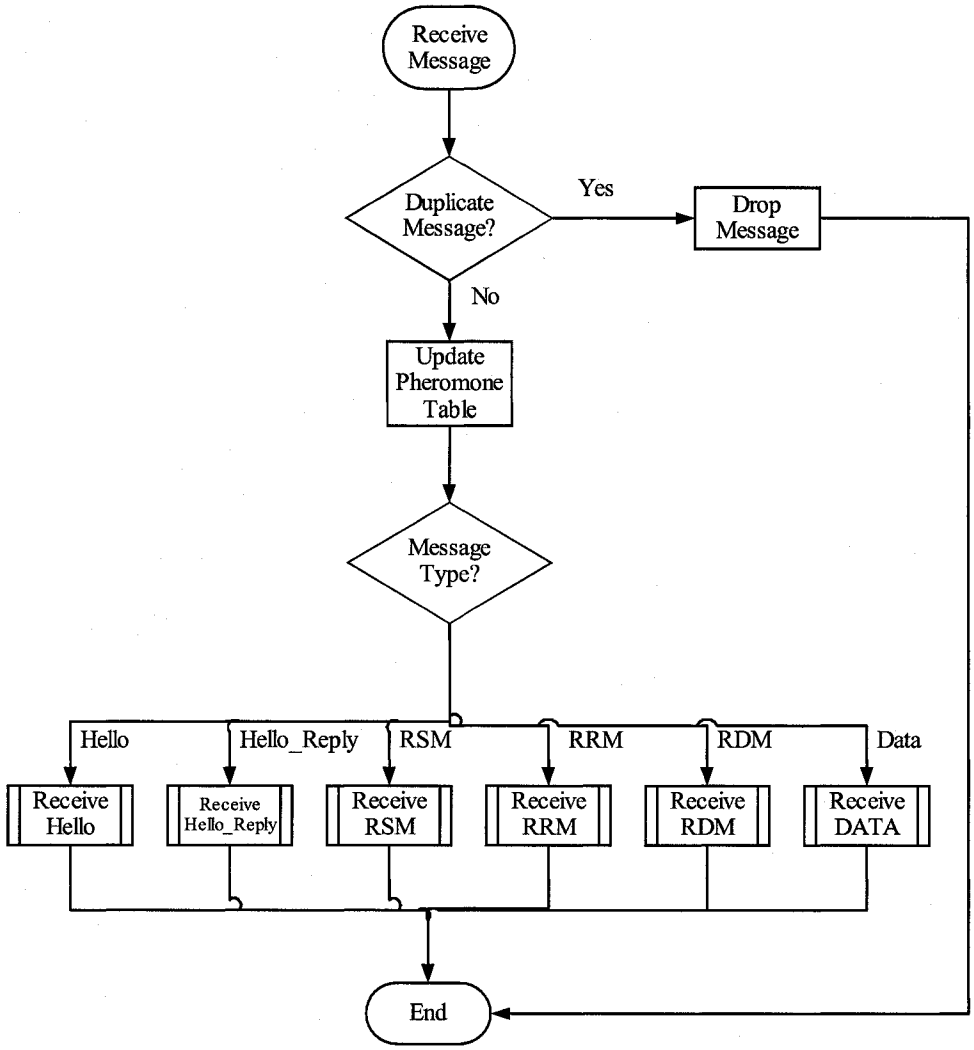


**Figure 4.11 - Receive message flow**

### 4.4.1 The Pheromone Table

Each node in the network contains a structure known as a pheromone table. This pheromone table replaces the idea of the routing tables used in traditional routing techniques. Within the protocol presented in this chapter, there are some modifications made to this basic notion of a pheromone table. As the system has several pheromone

types within it, the pheromone tables must also represent this notion. It must not only contain a measurement of the quality of the path by choosing a particular neighbour to route a message to a particular destination across, but it must do this for several pheromone types. It must also use the interaction of these pheromones to strengthen trails of the same type, while repelling differing types from travelling along the same path. The structure used to achieve this can be seen in Figure 4.12.

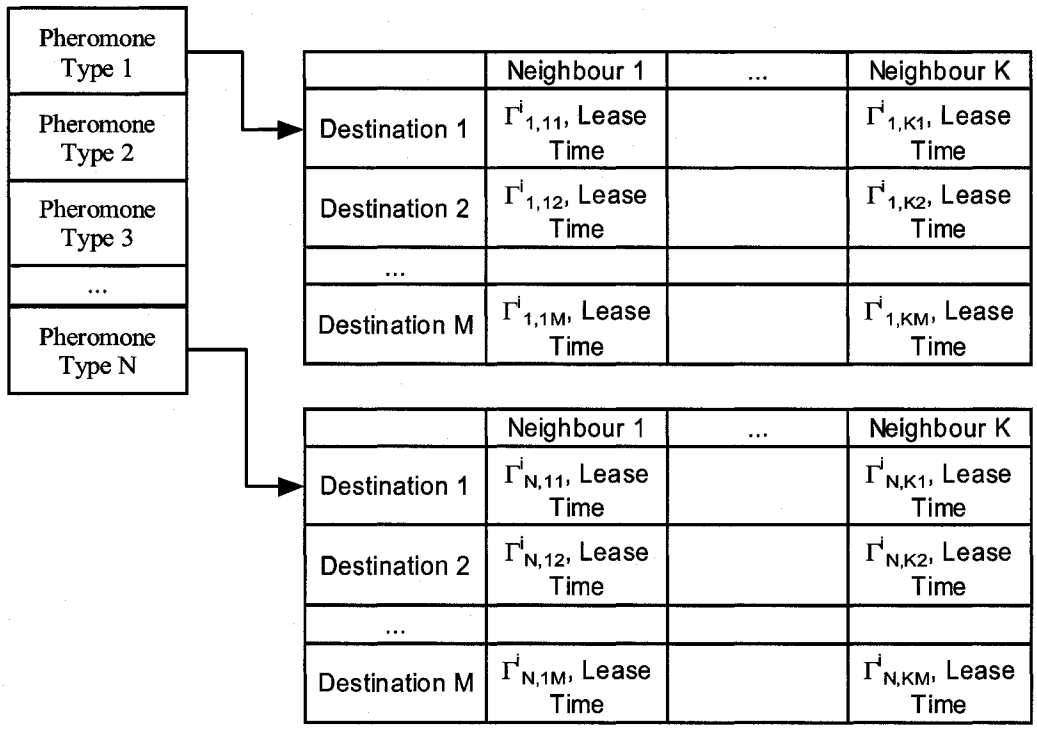


**Figure 4.12 - Pheromone Table Structure For Node i**

When a node receives a Hello message, it creates an entry for the Node ID of the destination carried on the packet in the pheromone table. An entry is also created whenever a packet is processed at the node that carries a Node ID that does not already

exist in the table. In addition, if a message is received containing a pheromone type not currently in the table, an entry for this type is created. This can be seen in detail in the pseudocode supplied in Figure 4.13. As was mentioned previously in this chapter, entries are removed from the table after a period of inactivity. In this manner, "stale" pheromone entries will be removed, leaving only the most recent, and active, links as those under consideration in the next hop calculation.

```
function update_pheromone_table
 insert_new_pheromone_type
 if destination node not in pheromone table
  insert_destination_node()
 end
 update W_Best for destination node
 update pheromone table entry
 update lease time for pheromone table entry
 remove stale entries from pheromone table
end update_pheromone_table

function insert_destination_node
 create new W_Best entry for destination
 for each pheromone type
  add destination entry
  add_destination_helper
 end for
end insert_destination_node

function add_destination_helper
 for each neighbour
  add neighbour for destination with initial pheromone value
 end for
end add_destination_helper

function insert_new_pheromone_type
 if pheromone table does not contain pheromone type
  create new pheromone type entry
  for each known destination
   add destination entry for new pheromone type
   add_destination_helper()
  end for
 end if
end insert_new_pheromone_type
```

**Figure 4.13 - Pheromone Update Pseudocode**

To handle the multiple pheromone types, and to handle accessing the table in an efficient manner, the table is structured using the different pheromone types as the first level index. In order to make pheromone types globally feasible in a network where routing decisions are based on local views is attained by using an integer representation. Each node assigns a pheromone type of 1 to the first share, a pheromone type of 2 to the second share, and so on. As each pheromone type would ideally be routed along a different first hop, the number of shares created is limited to the number of neighbours a particular node has. Thus, as the maximum number of neighbours is limited, so is the maximum number of pheromone types. Contained as an entry at each of the pheromone type indexes are tables containing each destination that the node knows about and every neighbour of the node. These tables contain a measure, $P^{i}_{s,nd}$, indicating a quality metric for choosing a neighbour as the next hop on route to a particular destination. The update mechanism for this value is shown in Equation 4.6.

In order to limit the number of agents needed within the system and to minimize the amount of network traffic (compared to using multiple ant types in systems such as [DDG05] and [SHBR96]) a single ant type updates the pheromone tables of each node along its path on its way to the destination node, viewing its source as a destination. Thus, as an agent moves along its path, it collects trip metrics and uses this to represent the quality of using this path to travel from the current node to the

source of the agent. This does make the same assumption as in [SHBR96] and [Guer97] that traffic congestion is approximately equal in both directions. In this manner, the number of agents and the amount of network traffic generated by the system is effectively halved in comparison to the traditional method.

The metric considered when determining the quality of a route is the round trip time. As quicker times (and therefore likely shorter paths, or paths with more available bandwidth, or paths with minimal end-to-end delays, or all of the above) are favoured, the actual metric is the inverse of the time difference between when a packet was sent from the source and when it is processed at a given node. While other metrics such as hop count may give more precise information about the quality of a path, carrying this type of information with an agent compromises the security of the originating node. If an intermediary node knew how many hops a packet had taken, it would provide crucial information with respect to linking a node identifier with a particular node. It is for this reason that time is the only metric considered.

The routing information of a node *i* is represented in a pheromone table $\Gamma^i$. The entry $\Gamma^i_{s,nd} \in \mathbb{R}$ of this table is the pheromone value for pheromone type *s* indicating the estimated goodness of going from *i* over neighbour *n* to reach destination *d*. If pheromone information is available, the ant chooses its next hop *n* with probability $P^i_{s,nd}$: If no pheromone information is available, the ant is flooded to all of the nodes neighbours.

**Equation 4:3 - Next Hop Probability Function**

$$P^{i}_{s,nd} = \frac{(\Gamma^{i}_{s,nd})^{\alpha} \cdot (1/\phi^{i}_{s,nd})^{\beta}}{\sum_{m \in N^{i}_{d}} (\Gamma^{i}_{s,md})^{\alpha} \cdot (1/\phi^{i}_{s,md})^{\beta}}$$

In the next hop probability function shown in Equation 4.3, $N^{i}_{d}$ is the set of neighbours of *i* over which a path to *d* is known, and $\alpha$ is a parameter value which can control the exploratory behaviour of the ants. This value is always greater than 0 ($\alpha > 0$), otherwise the signal is repulsive, not attractive.

**Equation 4:4 - Foreign Pheromone Trail Computation**

$$\phi^{i}_{s,nd} = \sum_{p_{\min}, p \neq s}^{p_{\max}} \Gamma^{i}_{p,nd}$$

In these equations, $\phi^{i}_{s,nd}$ represents the amount of pheromone trail not belonging to pheromone type *s* between nodes *i* and *n* with respect to reaching node *d*. This is called the amount of foreign pheromone. It is the sum of all pheromone trails left by other ant types. The calculation for this value can be seen in Equation 4.4 above. In this equation $p_{\min}$ and $p_{\max}$ refer to the minimum and maximum values for the pheromone types, respectively. The power $\beta$ indicates an ant's sensitivity to the presence of foreign trails, and has the property of always being greater than 0 ($\beta > 0$). Through the interaction of the competing pheromone types, packets assigned different

pheromone types will choose different paths from one another. The idea of using the sum of all other pheromone types is taken from [NVV04]. This paper showed promising results using this idea, particularly with respect to how using this definition the ant types all converge toward their own path, rather than all trying to converge to one single path. This is necessary for obtaining disjoint solutions. Usually it means one ant type converges on the best path found, while the next converges toward the second best path. There are situations where the shortest path is not in the solution set, because it would result in a suboptimal set of paths.

To simplify and reduce calculations, each ant will have a parameter $q_0$ associated with it. At each node a variable q will be randomly generated with a uniform distribution on the interval [0, 1]. If $q > q_o$, Equation 4.3 will be used to choose a next hop. Otherwise ($q \leq q_o$) the following calculation will be used:

**Equation 4:5 - Simplified Next Hop Function**

$$argMax_{m \in N_d^i} \left( (\Gamma_{s,md}^i)^{\alpha} \cdot (1/\phi_{s,md}^i)^{\beta} \right)$$

The value of the entry $\Gamma_{s,nd}^i$ is updated as follows:

**Equation 4:6 - Pheromone Update Function**

$$\Gamma_{s,nd}^i = \gamma \cdot \Gamma_{s,nd}^i + (1-\gamma) \cdot \tau_{s,d}^i$$

This update occurs whenever a node processes a protocol message (Hello, Hello Reply, RSM, RRM, RDM, and DATA). It is used to reinforce the pheromone entry in

the pheromone table for this particular hop. In the simulations, the pheromone concentrations were unbounded. Here, $\tau^i_{s,d}$ is defined as follows:

**Equation 4:7 - Path Reinforcement Calculation**

$$\tau^i_{s,d} = \left( \frac{W^i_{best:s,d}}{\mathrm{T}^i_{s,d}} \right)^{\theta}$$

At each intermediate node *i*, the ant sets up a path towards the destination (source) *d*, creating or updating the pheromone table entry $\Gamma^i_{s,nd}$ in $\Gamma^i$. The pheromone value in $\Gamma^i_{s,nd}$ represents a running average of the inverse of the cost in terms of estimated time to travel from *i* to *d* through *n*. In the formula above, $\mathrm{T}^i_{s,d}$ is the travelling time as estimated by the ant, and $W^i_{best:s,d}$ is the best time recorded over a window. $\mathrm{T}^i_{s,d}$ is included in the calculation of $W^i_{best:s,d}$, making the update in the interval (0,1). $\theta$ is a parameter used to weight the ratio between the current and best seen time. The $\mathrm{T}^i_{s,d}$ value is calculated by subtracting the timestamp on the message (put there at the source when the message was created) from the current time, as illustrated in Equation 4.8 below. This gives an estimate of the time it took for the message to reach the current node from the source. This idea of computing the quality of a path based on timing with respect to the best seen time is adapted from [CDG04] and [DD98]. Time windows smooth out spikes in performance that would otherwise

destabilize control. This type of approach using time as a measure of path quality has had very promising results in dynamic, ad hoc network environments.

**Equation 4:8 - Estimated Travelling Time Value**

$$T^{i}_{s,d} = currentTime - timestamp_{message}$$

Once the pheromone table has been updated, the actual handling of the message takes place. The following sections will detail how this occurs for each message type.

### 4.4.2 Receive Hello Message

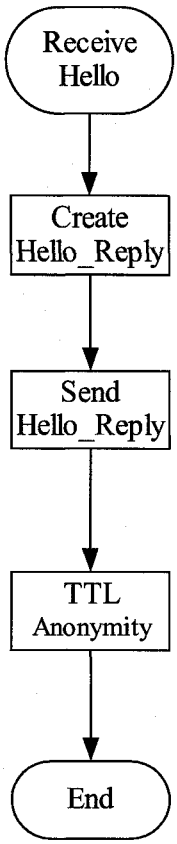


**Figure 4.14 - Receive Hello Message Flow**

The handling of a Hello message is shown in Figure 4.14 above. When a node receives a Hello message, it creates and sends a Hello-Reply message back to the source node of the Hello message. It then follows a subroutine named TTL Anonymity which will be examined in detail in Section 4.5. This routine adds anonymity protection for both the sender and intended receiver of a message through the manipulation of the TTL value, and sends the message along to one or more neighbours, depending on the message type and the result of the TTL manipulations.

### 4.4.3 Receive Hello Reply

The process for handling a Hello_Reply message can be seen in Figure 4.15 below. A node first checks to see if it is the intended destination. If it is, the reply information is added to the array of received Hello Reply messages, so that the responding node will be included in the neighbour selection process that was outlined previously in this chapter. The TTL Anonymity subroutine is then called. If the node is not the intended destination, the TTL Anonymity protocol is called without adding the responding node to a response array.

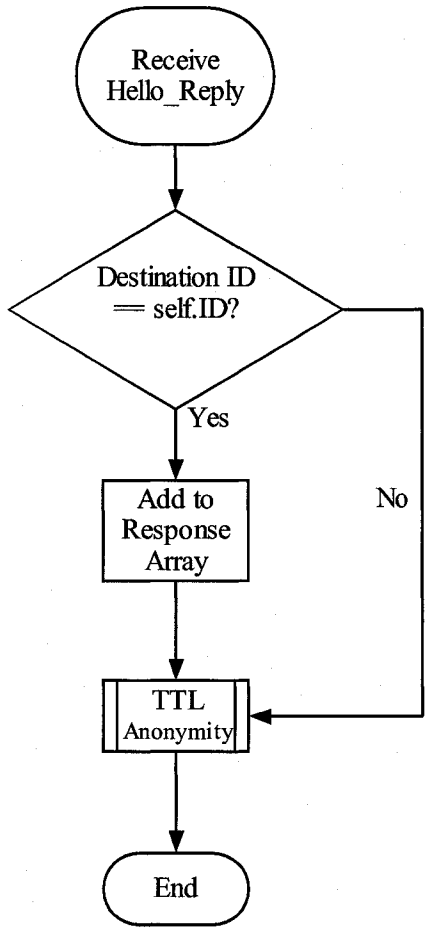


**Figure 4.15 - Receive Hello Reply Message Flow**

### 4.4.4 Receive Resource Search Message (RSM)

When a node receives a RSM, it checks its local drives to see if the resource being requested is stored locally. If it does, the node creates a RRM and sends this to the requesting node. Recall that RRMs are broken apart into shares using Shamir's threshold scheme, and that each share is given a different pheromone type and routed across disjoint paths to the intended destination. The RSM that was received is then

handled using the TTL Anonymity measures. If the resource is not available locally, no RRM is created, and the receiving node simply handles the RSM using the TTL Anonymity routine. This is illustrated in the figure below.

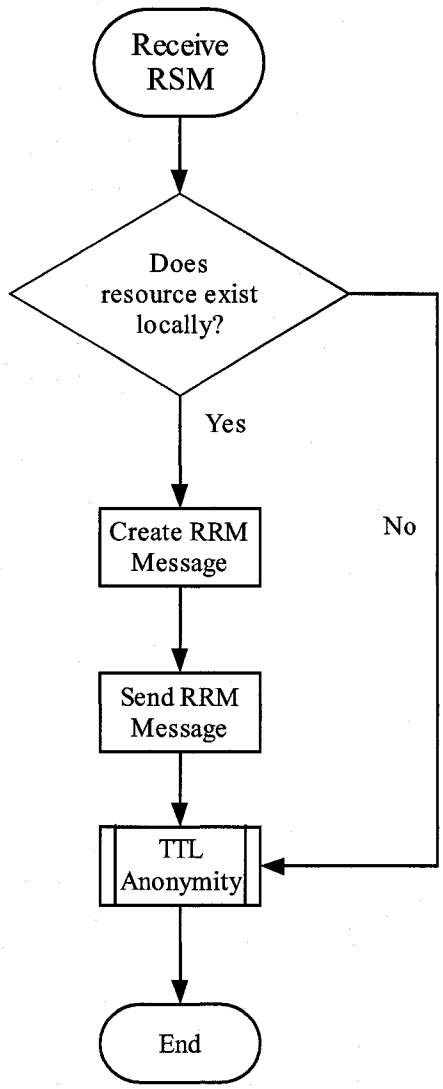


**Figure 4.16 - Receive RSM Flow**

### 4.4.5 Receive Resource Reply Message (RRM)

As with the handling of the RSM described above, when a node receives a RRM it first checks to see if it is the intended destination. If not, then it moves into the TTL Anonymity routine. Otherwise, it stores the share of the RRM that was received, and then checks to see if it has now received enough shares to reconstruct the full message. If the node has not received enough shares to reconstruct the message, it moves to the TTL Anonymity handling of the RRM message. Otherwise, it reconstructs the message to learn which node has the resource it requested. The node then creates and sends a RDM in order to request that the resource be sent to it. Finally the node handles the forwarding of the RRM in the TTL Anonymity routine. Figure 4.17 below illustrates this flow.

In the handling of this message type, along with the other message types that are received as shares, when the initial share of a message is stored, it also has the time it was received stored. Periodically, the storage structures for these shares are checked, and if the timestamp is older than an expiration period value, all shares are discarded. This checking interval along with the expiration period value are presented along with the rest of the simulation parameters in Appendix A.

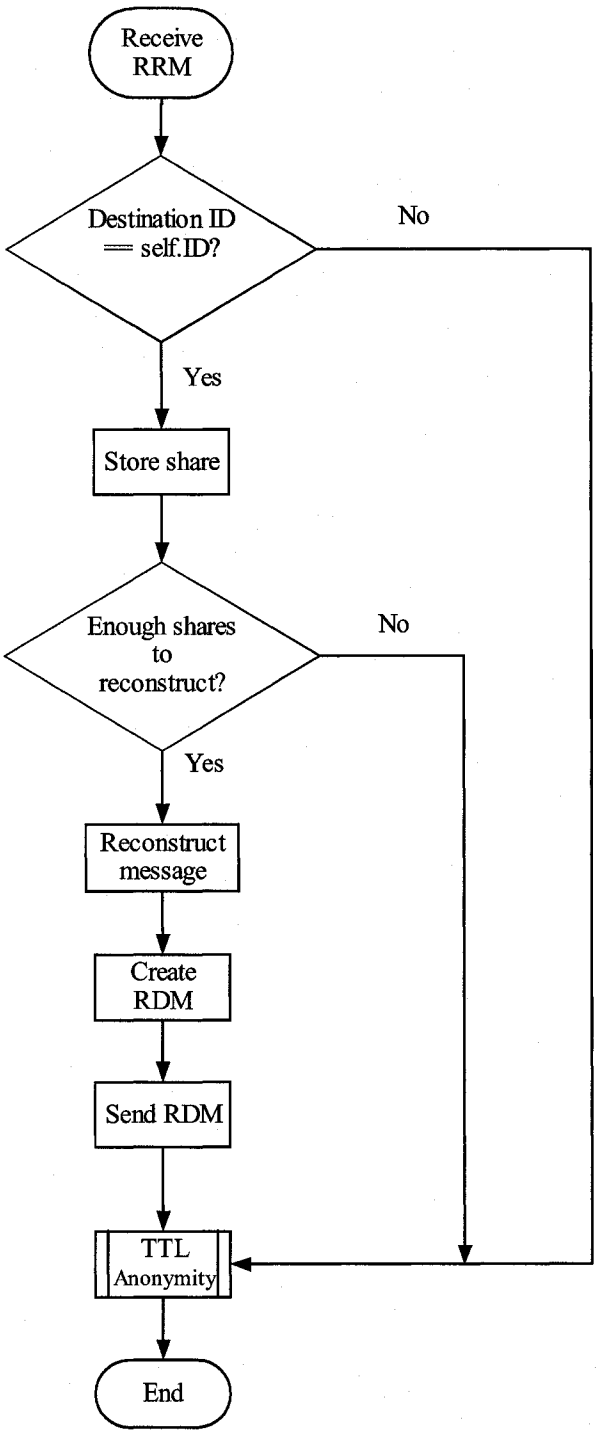


**Figure 4.17 - Receive RRM Flow**

### 4.4.6 Receive Resource Download Message (RDM)

The handling of this message type is identical to the handling of the RRM described in the preceding section. The only difference is that in this phase the node creates a DATA message once it has received enough shares of the RDM to reconstruct the whole message. The DATA message it constructs contains the requested resource. Figure 4.18 illustrates this phase.

### 4.4.7 Receive Data Message (DATA)

As with the previous two stages (RRM, RDM), when a node receives a DATA message it first checks to see if it is the intended destination. If it is, it stores the shares that was received, and then checks to see if it has received sufficient shares to reconstruct the entire resource. If there are sufficient shares, the node reconstructs the resource. It then handles the forwarding of the DATA message through the TTL Anonymity measures. Figure 4.19 illustrates this flow.

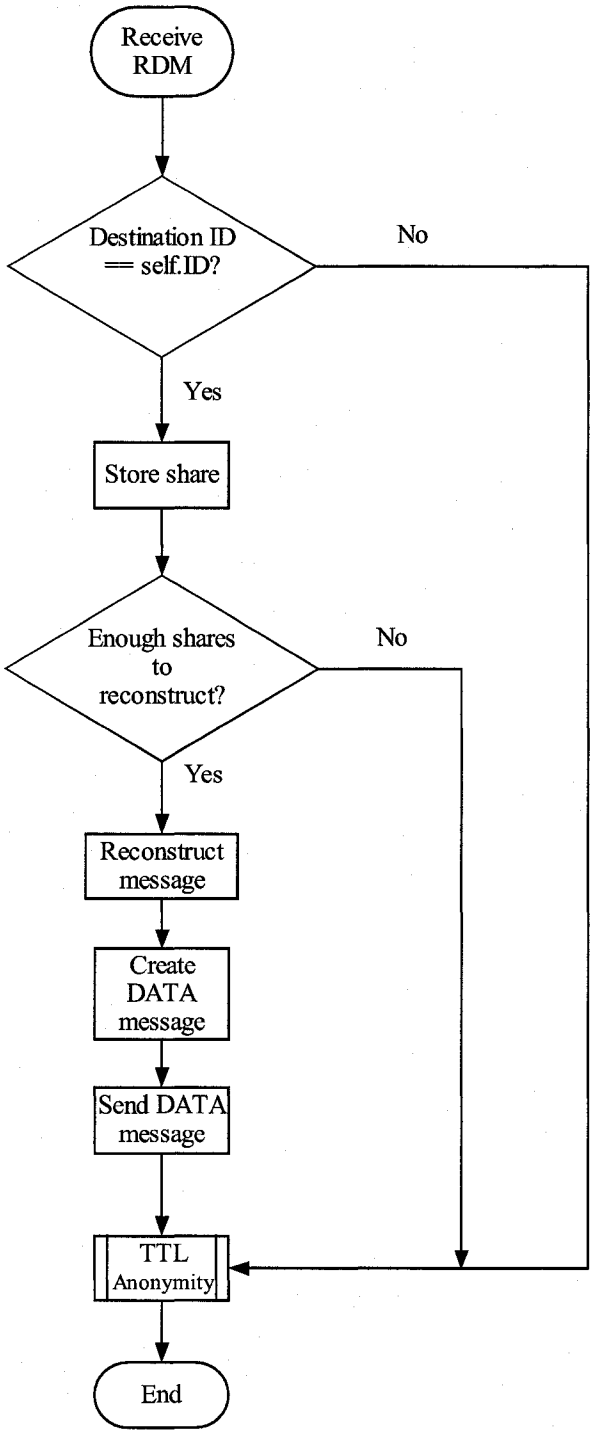


**Figure 4.18 - Receive RDM Flow**

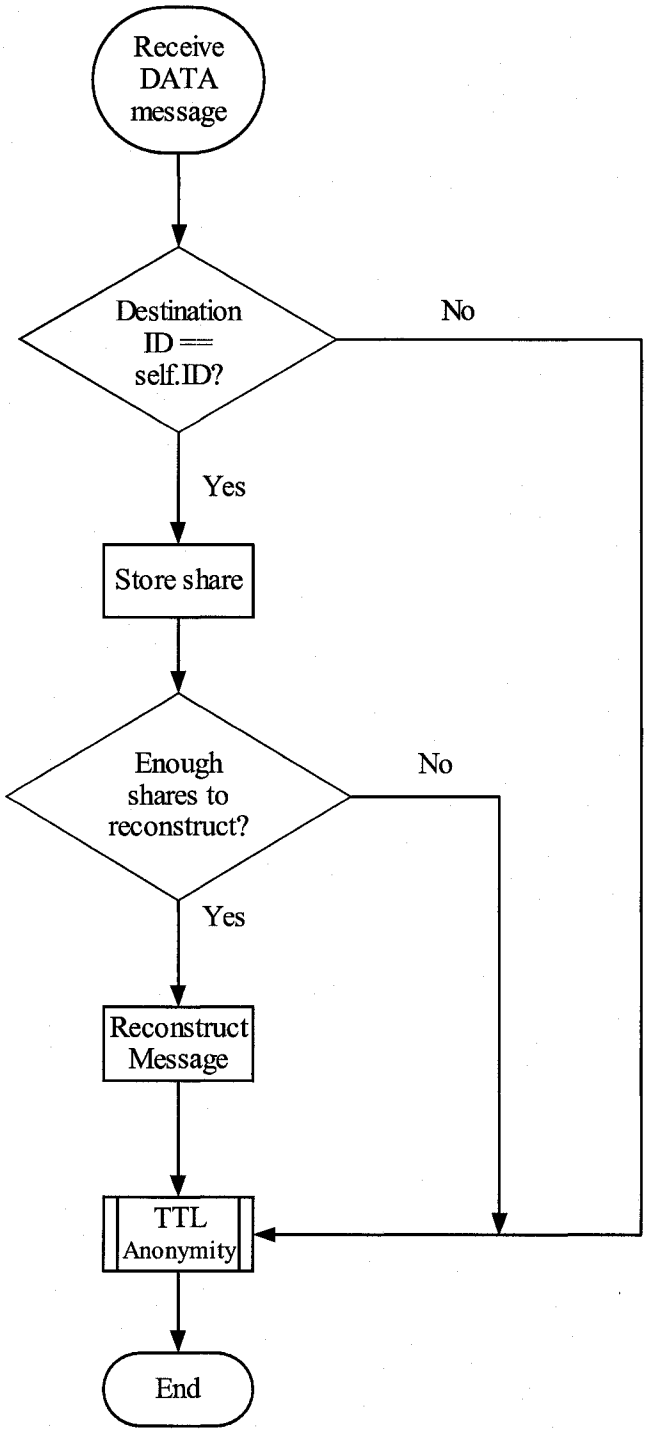


**Figure 4.19 - Receive DATA Flow**

## 4.5 TTL Anonymity

This section will examine the techniques to obfuscate both the sender and the receiver of a message. The notion of sender-receiver unlinkability is provided by avoiding direct connections between the pair, as well as the probabilistic hop-by-hop routing approach that is employed through the use of an ant-based routing approach. The use of link-encryption also strengthens this notion of unlinkability due to its nature of acting like a MIX as discussed in a previous section above. The use of pseudonyms in the form of unique node identifiers provide a measure of anonymity to all the nodes in the network, but attacks on anonymity can still take place through the exploitation of the scope limiting mechanism used in the network, namely TTLs. The overall handling is shown in the Figure 4.20.

In Figure 4.20, everything above the decision point of "TTL == 0?" corresponds to the sender anonymity portion, while everything in the diagram following this decision point corresponds to the receiver anonymity portion.

The following sections will present the techniques that will protect the anonymity of sender and receiver nodes against time-to-live and statistical attacks. First the techniques used to provide sender anonymity will be discussed, and then an analysis on receiver anonymity will follow.

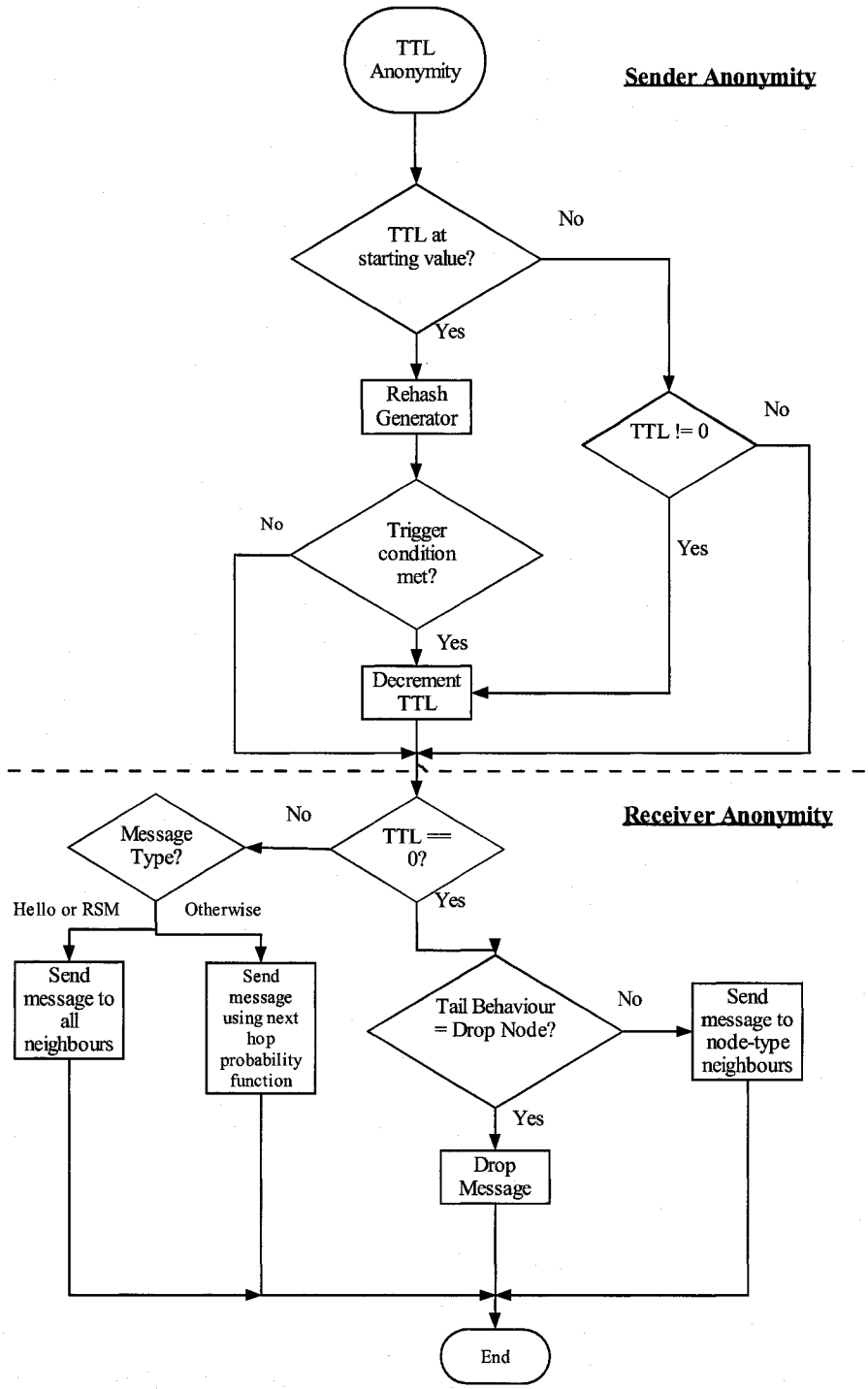


**Figure 4.20 - TTL Anonymity Flow**

### 4.5.1 Sender Anonymity

In order to mask the true origin of a message, the value of the TTL is not automatically modified at the first few nodes of a path. In order to determine whether or not this value should be modified, a dependent probability computation is implemented. In order to achieve this, a random number generator is attached to the message. At the first few nodes, this value would be manipulated, and if it met certain criteria (deemed a trigger value), then the TTL value would be decreased. Otherwise, the TTL value would remain at 0, and passed along to the next node. Thus, a message would travel a random number of hops with no TTL modifications, and then would travel further following a standard TTL mechanism. In this manner a message is flooded a certain number of hops into the network before the TTL value is modified, making it impossible to tell where the message actually originated from.

This random depth flooding mechanism still suffers from a statistical vulnerability. If each search from a given node uses a different random depth flood, then searches generated by an immediate neighbour will have different statistical properties than searches generated by more distant nodes: if a series of searches is generated by a neighbour, a particular node will always be part of the random flood, but if the searches are generated by a more distant node, the random flood will sometimes be over by the time the search reaches this particular node. To prevent this kind of statistical analysis from being performed by an adversary across multiple searches, the same starting generator state must be used repeatedly for each search

request that a given node sends. Thus, a given node always starts a search with the same flood depth. Since the same depth flood is used for each search, a particular node will always be part of a flood or always not be part of a flood, independent of whether the sender is this node's neighbour.

To implement such a random number generator, a 20-byte SHA1 hash value is used as the generator state and attached to each packet. To generate a new random value from the generator, a node simply re-hashes the current hash value, which generates another hash. A trigger value can be taken from the last few digits of the hash. For example, to switch message modes with a 1 in 5 chance, a node would look at the last byte of the newly-generated hash and switch modes if the last byte is less than or equal to 51 (there are 256 possible values for the last byte). SHA1 is a cryptographically secure one-way function, so it is very difficult to obtain previous generator states from the current state, and thus it is difficult to determine how far a message has traveled so far.

### 4.5.2 Receiver Anonymity

The above scheme ensures that an adversary cannot easily determine the origin of a search request, but it does nothing to guarantee the anonymity of nodes that are responding with search results. For example, to determine what files an immediate neighbour is sharing, an adversary could send a search request with an artificially high TTL. With a properly chosen TTL value, the adversary could trick the neighbour into

sending back its own results but not passing the message further. For example, if a node creates a message and sets the TTL value on it to the maximum value, and then forwards this message to its neighbour, it knows that the neighbouring node will drop the message after processing it, due to the TTL value. Thus, if the initiating node receives any responses to its request, then it could ascertain that these results came from the neighbouring node, and thereby, link this neighbouring node to its Node ID, which would be included in the response. This would allow the node to create a linkage between the Node ID and IP address of its neighbouring node, thus compromising its anonymity.

No deterministic dropping scheme can be used to protect responder anonymity. An adversary can simulate a deterministic scheme ahead of time and find a drop mechanism value that would force its neighbour to process and drop a message. Thus a truly probabilistic scheme in which each node makes an independent probability calculation is required - an adversary could not manipulate such calculations.

One scheme to accomplish this is to dictate that the message needs to be forwarded a random number of hops further past the node which exceeds the TTL limit. Simply choosing whether or not to forward the message to someone brings about a new problem however, as explained in the following example. An attacking node could send a particular node a message with a high TTL, as mentioned above. If this node only forwards the message to one neighbour, and if the chosen neighbour is also controlled by the adversary, the two attacking nodes have effectively cornered this

particular node. Therefore, in addition to the uncertainty regarding the number of extra hops, there must also be some uncertainty about the number of neighbours a node forwards the message to. Once the idea of branching of messages is introduced, the further along the message travels, the smaller the probability it ever gets terminated becomes.

To deal with these issues introduced by branching messages, the following mechanism is used in the system. Upon joining the network, every node decides whether to forward a message or not, and how many neighbours to forward it to. If it can be ensured that there are enough nodes in the network that messages will be dropped, then this scheme will work. So how many nodes are needed so that messages will terminate in the worst case? Nodes that do not forward messages are termed drop-nodes, and nodes that forward to one neighbour 1-nodes, to two neighbours 2-nodes, and to n neighbours n-nodes. Thus, in the worst case we need at least n drop-nodes for every n-node in the network.

The number of n-nodes in a network are identified as c(n). This first constraint mentioned above can then be illustrated as in Equation 4.9 below.

**Equation 4:9 - First Constraint**

$$c(drop) \geq c(1) + 2 \bullet c(2) + \ldots + n \bullet c(n) + \ldots$$

This essentially states that there must be enough drop nodes to cover all the forwarded chains in the network. It should also be stated that the sum of all the types of nodes must equal the number of nodes in the network (N):

**Equation 4:10 - Constraint 2**

$$N = c(drop) + c(1) + c(2) + ...c(n) + ...$$

These constraints must now be converted into a probability distribution that each node can use to independently select its tail message behaviour. Let p(n) be the probability that a given node is an n-node. This conversion can be generated by dividing each count, c(n), by the total number of nodes in the network, as shown in the equation below:

**Equation 4:11 - Probability Function**

$$p(n) = \frac{c(n)}{N}$$

Therefore, if this conversion is applied to the second constraint shown in Equation 4.9 above by dividing through by N, the following is obtained:

**Equation 4:12 - Proper Probability Distribution**

$$1 = p(drop) + p(1) + p(2) + ... + p(n) + ...$$

This ensures that the solution has a proper probability distribution. If a similar approach is followed with the first constraint shown in Equation 4.8 , the following is achieved:

**Equation 4:13 - Drop Node Numbers**

$$p(drop) \geq p(1) + 2 \bullet p(2) + ... + n \bullet p(n) + ...$$

This formula dictates that the probability of being a drop-node must be at least as large as the sum of the probabilities of being another node type. This will ensure that there are enough drop nodes to cover every n-node in the network.

There is a solution that satisfies our first constraint with exact equality: $p(drop) = 2/3$ and $p(n) = \frac{1}{3 \bullet 2^i}$. However, if a solution that satisfies the first constraint with loose inequality (with $p(drop)$ being larger than it needs to be) is chosen, then the distance a message travels before being dropped would be reduced. This is a result of there being a larger number of drop nodes within the network. Thus, in the simulations, the following values are used in order to once again try and reduce the number of additional traffic incurred by these anonymity measures:

**Equation 4:14 - Simulation Values**

$$p(drop) = \frac{3}{4}$$

$$p(n) = \frac{1}{2^{(n+2)}}$$

Using these values achieves a proper probability distribution, as shown below:

**Equation 4:15 - Proof of Proper Distribution**

$$\sum_{n=1}^{\infty} p(n) = \frac{1}{4}$$

$$\sum_{n=1}^{\infty} p(n) + p(drop) = 1$$

The use of these values also gives the proper number of each type of node in the network, as shown below:

**Equation 4:16 - Proof of Proper Numbers of Node Types**

$$\sum_{n=1}^{\infty} n \bullet p(n) = \sum_{n=1}^{\infty} n \bullet \frac{1}{2^{(n+2)}} = \frac{1}{2}$$

$$p(drop) \geq \sum_{n=1}^{\infty} n \bullet p(n) = \frac{3}{4} \geq \frac{1}{2}$$

The probabilities for each node type are presented in the following table:

**Table 4:1 - Tail Behaviour Probability Distribution**

| Number of neighbours to forward to | Probability of choosing this behaviour |
|---|---|
| 0 (drop) | 3/4 |
| 1 | 1/8 |
| 2 | 1/16 |
| 3 | 1/32 |
| 4 | 1/64 |
| 5 | 1/128 |
| 6 | 1/256 |
| 7 | 1/512 |
| 8 | 1/1024 |
| 9 | 1/2048 |
| 10 | 1/4096 |

Thus, upon joining the network, each node generates a random number between 0 and 1, and chooses a behaviour based on this generated value and the table shown above. To thwart an (n-1)-neighbour attack, where n is the number of neighbours that a node has, the node decides at initialization, using the above probabilities, how many

neighbours it will pass tail-mode messages to, and it will always pass them to the same set of neighbours as long as those neighbours are connected. If a particular node within this set leaves the network, then a replacement node is chosen at random from the remaining neighbour nodes to replace it. Thus, though an adversary might send a node a search request in tail mode, and though some of the cooperating adversaries may receive some of the forwarded tail-mode messages that we send (since they may be the neighbours that we have selected to receive our tail-mode messages), they cannot be certain that we are not also forwarding the search request on to other neighbours. If a node generates a random number such that the node should forward to more neighbours than it actually has, the node defaults to forwarding to all of its neighbours. As the calculation provided above account for enough drop nodes to cover all other node types in the worst case scenario, if a node forwards to fewer neighbours than intended, a less severe case is generated, and there will still be enough drop nodes within the network to cover all the node types. In fact, this would create more than enough drop nodes.

## *4.6 Summary*

This chapter has presented a detailed review of the protocol designed to satisfy the problem statement. It has outlined both techniques and algorithms used to achieve the protection of identity and data, along with the integrity of said data.

Anonymity is provided through the use of pseudonyms and through the avoidance of creating direct connections between communicating peers. Techniques

were also included to protect against different kinds of search-related attacks that threaten anonymity. First, attacks that are aimed at discovering what a given node is searching for were examined. The random-flood mechanism described above effectively thwarts this attack, even if multiple adversaries are working together. Second, there are attacks that are aimed at discovering what search results a given node is returning. The tail-node mechanism also presented above hinders this attack. The combination effectively prevents time-to-live and statistical attacks, while the use of link encryption prevents eavesdropping attacks.

Data confidentiality is provided through the novel application of threshold schemes, along with the use of competing pheromone types to dynamically and adaptively discover disjoint paths between nodes using only local information. This combination also helps to prevent eavesdropping attacks, in combination with malleability attacks.

The use of threshold schemes also offered data integrity to the system, making any modified data through a malleability attack easy to detect.

The next chapter will analyze in depth how the protocol outlined in this chapter actually addresses the problem statement. Those aspects which are theorized to work will be proven through simulation. The simulation parameters will be presented, along with various metrics that were observed and scrutinized. The results of numerous simulations across varying network sizes will be presented and analyzed in terms of these metrics. Through these simulation results and the numerical analysis of the

remaining properties of the system it will be shown that the protocol does satisfy the problem statement.

.

# Chapter 5

# Protocol Analysis and Simulation Results

This chapter will analyze the protocol presented in the previous chapter with respect to the problem statement presented in Chapter 1. It will examine how the protocol meets the desired qualities; identity protection, data confidentiality, and data integrity. To discuss various aspects of these qualities simulation results will be analyzed and discussed. The first section of this chapter will therefore examine the simulation environment and the rationale behind the decisions in structuring it in this manner. Following this section will be sections discussing how the three desired qualities are met. These discussions will involve both the inclusion of proven techniques and the results of numerous simulations performed in the previously introduced simulation environment. Finally, this chapter will present results including routing load and message reconstruction delay, to demonstrate the impact of the security measures within the protocol on the quality of routing.

## *5.1 The Simulation Environment*

There are two main elements to identify in the discussion of the simulation environment. The first is the underlying network properties upon which the simulation

is run. The second is the model of the P2P network, including resource distribution and how resource search messages (RSMs) are generated. The following two sections will look at each of these elements and present the logic behind the decision making process.

### 5.1.1 Underlying Network Properties

The simulator that the protocol was built upon was NS-2, The Network Simulator [NS06]. NS-2 is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

The design of network protocols is greatly accelerated by the use of simulators, particularly when studying a protocol's behaviour in a large inter-network topology. However, the accuracy of the simulation results is heavily affected by the input network topology. As taking a real map as an input is not always feasible, artificially created topologies are often used [Mag05]. The topology of the networks used in the simulations was generated using The Network Manipulator (Nem) [NEM06]. It is capable of creating realistic Internet-like topologies and can perform thorough topology analysis on them on the fly. An extended scale-free power-law model proposed by Albert and Barabasi [AB00] was used to generate the topology within Nem. The subject of which generator family is most appropriate is the subject of much debate [TGJ+01] [JCJ00] [MMB00]. Since degree-based (power-law) topology generators are

newer, the implication is that they are the most appropriate for modeling internet (and therefore peer-to-peer) networks [Har05]. The Albert-Barabasi model suggests two possible causes for the emergence of a power law in the frequency of outdegrees in network topologies: incremental growth and preferential connectivity. Incremental growth refers to growing networks that are formed by the continual addition of new nodes, and thus the gradual increase in the size of the network. Preferential connectivity refers to the tendency of a new node to connect to existing nodes that are highly connected or popular.

This model interconnects the nodes incrementally. When a node i joins the network, the probability that it connects to a node j already belonging to the network is given by Equation 5.1:

**Equation 5:1- Albert-Barabasi Connection Probability**

$$P(i,j) = \frac{d_j}{\sum_{k \in V} d_k}$$

In Equation 5.1 $d_j$ is the degree of the target node, $V$ is the set of nodes that have joined the network and $\sum_{k \in V} d_k$ is the sum of the outdegrees of all nodes that previously joined the network.

Simulations were run on networks of 10, 20, 50, and 100 nodes. The average degrees for the nodes in these networks are given in the table below.

**Table 5:1 - Mean Simulation Node Degree**

| Network Size | Mean Node Degree |
|---|---|
| 10 nodes | 4.68 |
| 20 Nodes | 6.42 |
| 50 Nodes | 9.53 |
| 100 Nodes | 11.78 |

All tests were run for 12 million time units per run, within which events could take place. There were five different network topologies generated for each size of network, and 20 runs were performed on each of these topologies. Thus, for each network size a total of 100 runs were performed across 5 different topologies.

### 5.1.2 Peer-to-Peer Modelling

As NS-2 is mainly used to test new routing protocols, there is no built in support for modelling P2P networks. As such, simple generation methods were built into the protocol to simulate the properties of a P2P network. The issue of content distribution was addressed by forming a pool of available content, and then randomly distributing it among the peers. This straightforward method was chosen in order to more effectively test the security afforded by the protocol. Details for the content distribution method can be found in Appendix A.

Within an actual P2P network, human users initiate search requests when they are looking for a particular resource. As this is not possible within the simulation

framework, search requests were randomly generated at nodes at certain time intervals. Again, details can be found in Appendix A.

### 5.1.3 Simulation Parameters

The main simulation parameters and their values can be seen in the table below. For a detailed description of the complete simulation environment and all parameters please refer to Appendix A.

**Table 5:2 - Simulation Parameters**

| **Parameter** | **Value** |
|---|---|
| Alpha | 5 |
| Beta | 1 |
| Gamma | 0.1 |
| Theta | 1 |
| $q_0$ | 0.1 |
| Minimum Neighbours | 4 |
| Maximum Neighbours | 10 |
| n from (k,n) Threshold Scheme | Number of Neighbours |
| k from (k,n) Threshold Scheme | The floor of 80% of n |
| $p_{min}$ | 1 |
| $p_{max}$ | 10 |
| MAX_HELLO TTL | 1 |
| MAX_TTL | 3 |

## 5.2 Data Confidentiality

To an adversary eavesdropping on the link between two communicating nodes, data confidentiality is afforded through the use of link encryption. The techniques used here have been proven and are by and large accepted to be sound, preventing the eavesdropping attack. Link-encryption does however allow each node along the path of a packet to "open" the packet and see the contents, as was discussed in the Chapter 3. However, as each message is split apart into shares, the contents of these packets are unintelligible unless enough of the shares have been collected to reconstruct the original message. The use of competing pheromone types to route the shares across disjoint paths should make this very difficult. An adversary would need to be able to collect a sufficient number of shares in order to glean any useful information about the message. This approach also prevents a malleability attack, as modified shares would not properly reconstruct to form the original message. Thus, while using only link-encryption allows a node to view the contents of the package, these contents provide the intermediary node with no information about the actual message.

Simulation results are needed to show that an adversarial intermediary node can not collect a sufficient number of shares, while the intended destination node can. This is achieved through the use of competing pheromone types to route the shares across disjoint paths. The following sections will look at simulation results to show that enough shares reach the intended destination, that no single or group of collaborating

adversarial nodes can collect enough shares to reconstruct the message, and that the shares of the same secret travel across disjoint paths.

### 5.2.1 Shares Reaching the Destination

To track the average percentage of total shares of a message that arrived at the intended destination node, each share that was created for and that was properly received was tracked for every node in the network. This allowed for a per node average percentage of total shares received. The values in Figure 5.1 below represent the average percentage of shares received per node across several network sizes, with varying topologies within each group of similarly sized networks. For all graphs presented, "Network Size" refers to the number of nodes in the network.

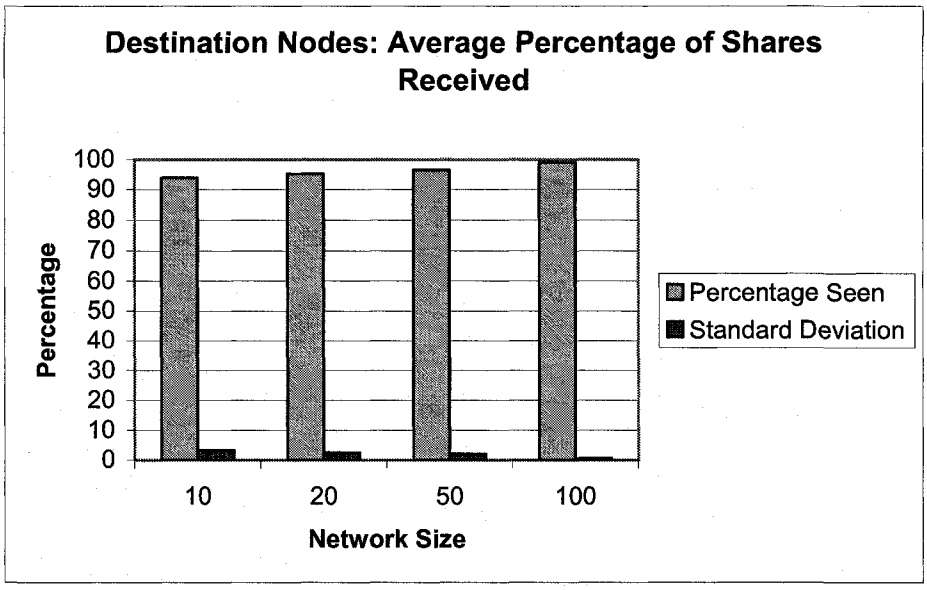


**Figure 5.1 - Destination Nodes: Average Percentage of Shares Received**

As can be seen, a large percentage of shares arrive at the intended destination, and this percentage increases as network size increases. An actual P2P network would normally have a far greater number of participants than the largest network size simulated, which should then have an even higher percentage. Even at a small network size of 10, the percentage of shares received at the intended destination node was 93.9. Within the simulation environment, the percentage of shares required to reconstruct a message was set to 80. For example, if a node were to have 10 neighbours, it would create 10 total shares of the message. Or, in the case of the 10 node networks, where the average node degree was 4.7, a node would, on average, break a message into 4 shares. To have the ability to reconstruct the message, the destination node would need to collect 8 or more shares. This number was selected as it provides a reasonable trade-off with respect to both the number of shares an adversarial node would need to collect in order to reconstruct the secret, while still allowing for a small percentage of shares to be lost due to malicious behaviour or network interruptions such as broken links. Therefore, having a percentage in the high nineties is encouraging, even though these results refer to an optimistic scenario in which no colluding nodes exist to disrupt communications. It indicates that the anticipated number of 80 is slightly low since the average is significantly higher and the standard deviation is quite low, at 3.5 percent in the worst case. This would indicate that a value of closer to 90 would have been more appropriate. Simulation results show that, on average, a higher percentage of shares than this are received, and the standard deviation is low, signifying that not nodes vary

greatly from this number. Therefore a similar level of service could be maintained while reducing an adversary's chance at collecting enough shares to reconstruct, thereby offering a higher degree of security, as the adversarial probability of being able to reconstruct a message would decrease.

### 5.2.2 Intermediary Nodes Intercepting Shares

Now that we have seen that we can receive a percentage of shares by the intended destination node, we need to determine whether or not any single intermediary node can collect a sufficient number of shares to reconstruct a message. Following this, the notion of a group of colluding nodes will be discussed.

#### 5.2.2.1 Single Intermediary Node

These values were computed on a per node basis, as with the intended destination node above. Each node tracked how many shares of a specific message it had seen when the message was not intended for it. The results of this metric can be seen in Figure 5.2 below. This graph shows the average percentage of shares received per node across several network sizes, with varying topologies within each group of similarly sized networks, as with Figure 5.1 above.

As the graph in Figure 5.2 illustrates, as network size increases, the average percentage of shares seen by a single intermediary node decreases, as does the standard deviation of the value. Intuitively this makes sense, as the number of paths between a pair of nodes would increase as network size increases. With the availability of more

paths, the probability of seeing several shares of the same message at an intermediary node would decrease. The standard deviation would also be expected to decrease as network size increases as there would be more paths to spread shares across, resulting in fewer pockets where higher percentages of shares travel through. This greater choice of path selection, along with a higher average node degree (and therefore a higher average number of neighbours per node) results in a greater number of shares being routed across a more diverse set of paths, makes it much more difficult for an intermediary node to collect the (now higher number of) required shares to reconstruct the full message.

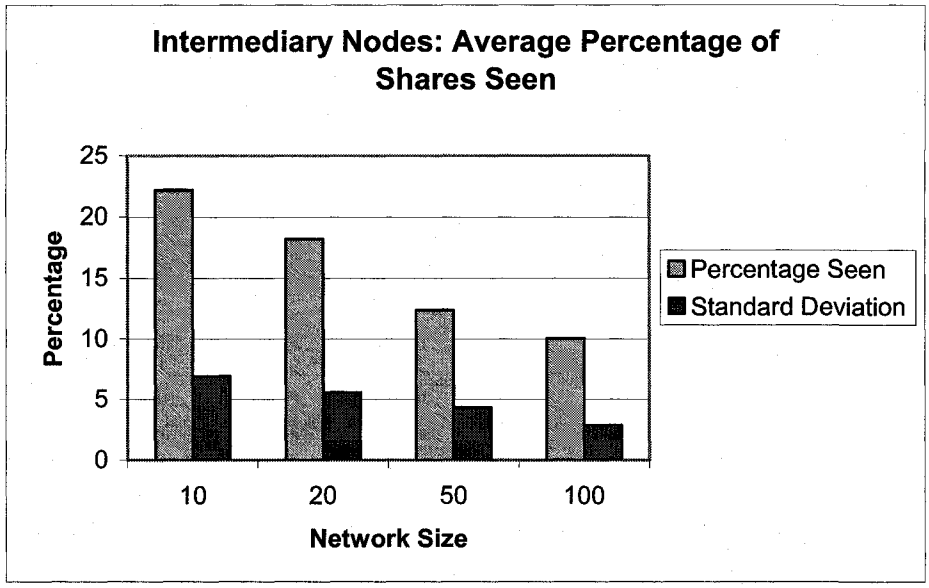


**Figure 5.2 - Intermediary Nodes: Average Percentage of Shares Seen**

In the worst case with a network size of 10 nodes, the average percentage of messages seen was only 22.1. This is well below the required percentage of 80, and a

fairly low standard deviation indicates few nodes differing a great amount from this average. Assuming a normal distribution, the probability that an adversary intercepts 80% of the shares is infinitesimally small, being over 20 standard deviations from the mean in the case of the 100 node network. As was discussed in the previous section, the threshold of 80 percent appeared to be too low according to simulation results. Thus, when comparing these results with a requirement of 90 percent of the total number of shares, the results are very encouraging. At a network size of 100 nodes, the average percentage seen had dropped to 10.0. In an active P2P networks with thousands of participants, this value would be expected to drop even lower, along with the standard deviation.

#### 5.2.2.2 Group of Colluding Nodes

The protocol appears to behave as desired against a single intermediary node, with a low percentage of total shares of a message being seen. However, we would also like to know how secure the protocol is against a group of colluding adversaries. Nodes were given an independent probability of being deemed a colluding node. This percentage was 10% in the simulations. In this manner, on average, 10% of the total number of nodes were colluding, adversarial nodes. If a node was neighbouring a colluding node, this probability rose to 25%. The maximum number of colluding nodes was restricted to 20% of the network size. This value of 20% is an extreme

figure, and is unlikely to occur in an actual implemented system. It was chosen to see how the protocol behaves in extremely unfavourable conditions.

The average percentage of shares of a message that this group of colluding nodes saw during the simulation run were then collected. The results can be seen in Figure 5.3 below.

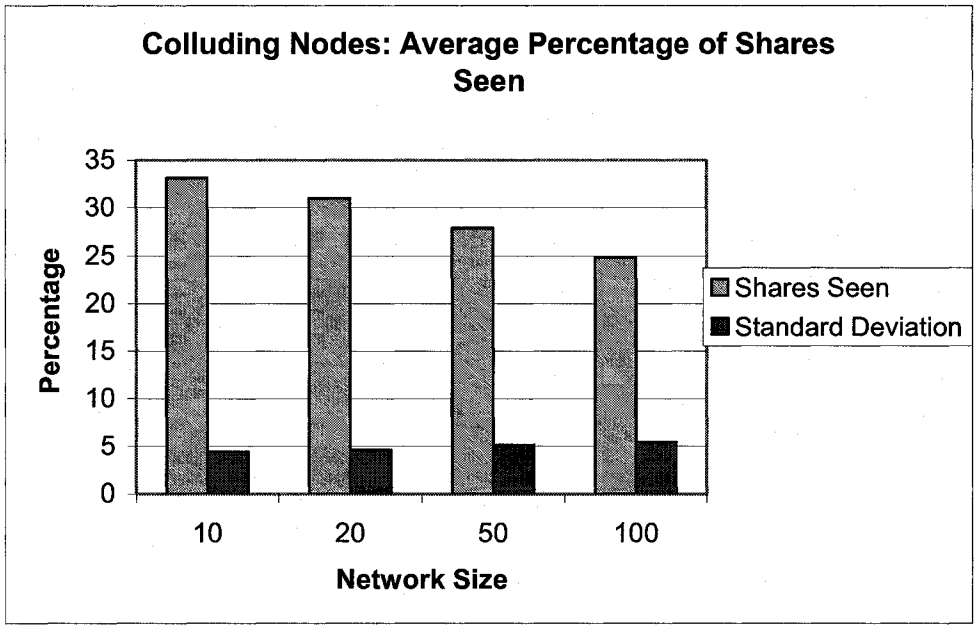


**Figure 5.3 - Colluding Nodes: Average Percentage of Shares Seen**

This figure illustrates the percentage of shares that a group of colluding nodes collects across increasing network size for groups of colluding nodes. As is to be expected, as the network size increases, and thereby the number of available paths increases, the average percentage of shares seen by the colluding group decreases. In this model, the adversarial group does not collect enough shares to reconstruct the entire message. In a non-disjoint solution, where complete messages are routed, these

results would indicate that over 25 percent of messages would be corrupted, wherein this solution, no message is compromised. It is important to note, however, that the disruption of 30% of the shares of a message would be sufficient to interrupt communications, should these nodes not properly forward the shares.

As was the case with the single intermediary node, the probability of a group of colluding nodes being able to collect 80% of the shares and thus bring able to reconstruct a message is extremely small. In the case of the 100 node network (again assuming a normal distribution), the collection of 80% of the shares is near 10 standard deviations from the mean, a much smaller value than in the case of a single intermediary node. However, this still results in a miniscule probability of the colluding nodes being able to collect sufficient shares to reconstruct the message.

### 5.2.3 Path Disjointness

In order to further substantiate the results in the previous sections, which indicate that the shares are being routed across disjoint paths, a measure of this disjointness was required. Nguyen and Zakhor [NZ03] propose a notion to quantify the number of shared links between redundant and default paths. Within the protocol presented in this document, the set of disjoint paths are viewed as the redundant and default paths. They "define the jointness percentage between the default and redundant paths as the number of shared links between them over the number of links of the default path [NZ03]." For example, if there are two paths under consideration, the

default path length is 6 hops, and the redundant path and the default path share 2 links, then the jointness percentage would be 33%.

Figure 5.4 shows the jointness percentage between the disjoint paths for all topologies as a function of network size. As expected, the jointness percentage decreases as the network size increases for across all topologies. This phenomenon is intuitively plausible since larger numbers of network nodes allows more choices of disjoint paths, thus producing more disjoint paths than a network with a smaller number of nodes.

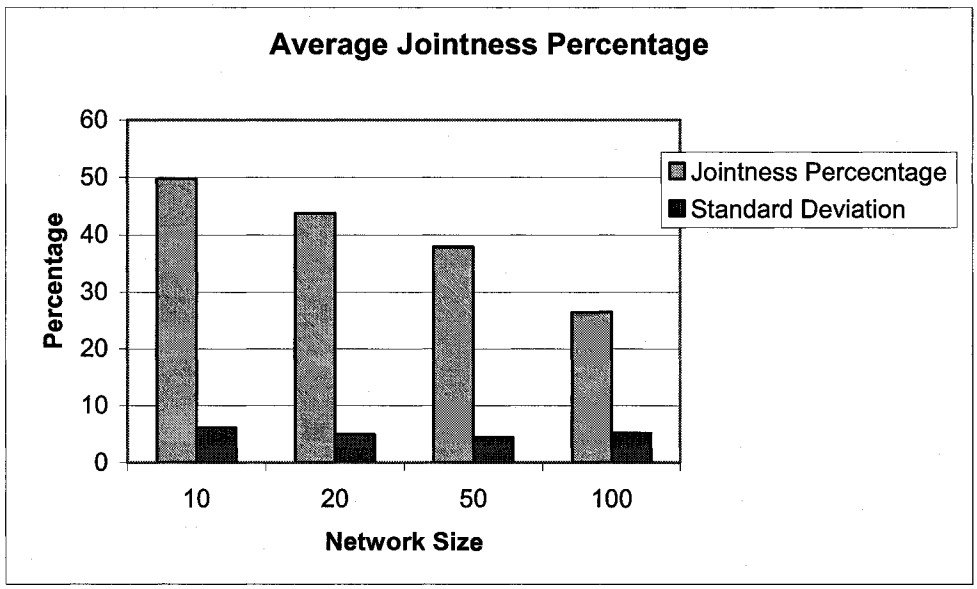


**Figure 5.4 - Average Jointness Percentage**

As the figure above shows, when network size increased the jointness percentage actually decreased dramatically. When the network size increased 10-fold, the jointness percentage decreased nearly 50 percent. If this trend were to continue in a

similar manner, in a network of one thousand nodes, the jointness percentage would be expected to be near 13 percent. As the number of expected hops is 7 (discussed in detail in Section 6.5), this value of 13 percent is less than one of these 7 hops that is expected to see more than a single share of a message. This would become smaller as the network size increased even more.

## *5.3 Identity Protection*

Identity protection is achieved by hiding the parties involved in a communication, and by hiding the fact that communication takes place. Hiding the parties involved is achieved through the use of pseudonyms. These are generated in a manner that makes them unique in both space and time, and are generated using a hash function that makes deciphering them computationally infeasible. These pseudonyms provide a global measure of anonymity.

Through the use of hop-by-hop (link-) encryption it is not possible to map a specific incoming message to a node to a particular outgoing message. In essence, every node acts as a MIX. This eliminates the need of reordering messages (as occurs in a MIX network) as incoming messages do not look the same as outgoing messages at a node, and as all messages are the same size (due to the use of Shamir's threshold scheme), there is no way to correlate a message going into a particular node with a message leaving the same node. This in conjunction with the techniques presented in

Section 4.5 for sender and receiver anonymity, make these types of correlations, even over long periods of time very difficult if not impossible.

There is still the issue of a local view of anonymity, however. This is the view of the nodes near a particular node, as opposed to the previously mentioned global view, which is the view of someone overseeing the network as a whole. These local views are addressed by the extra sender and receiver anonymity measures discussed in the previous chapter. Sender anonymity is provided through the attachment of a generator state. As this generator state is based on a one-way hash function, an adversarial node could not learn any information about how far a message has already travelled. All this adversarial node could do would be to switch modes early (i.e. start decrementing the TTL value before the triggering generator state is reached), or not switch modes when it is supposed to. Neither of these approaches would gain an adversarial node any advantage. Switching early would only slightly decrease the expected hop count of a message, while not switching would only slightly increase the expected hop count. The notion of receiver anonymity is provided through an independent probability decision, and provides the notion that no adversary can positively "corner" any node through TTL manipulations, as discussed in the previous chapter. However, in smaller sized networks, there will be a small number of nodes that actually forward messages to other nodes. For example, in a 10 node network, only 1.25 nodes are expected to forward to one other node, and one other node may possibly forward to 2 nodes. This is enough to introduce doubt to an adversary

attempting to corner a specific node, as they can not be certain that a node is not also forwarding the message on to other neighbours, but could likely be improved by fine-tuning the tail-node behaviour probabilities for specific network sizes.

Hiding the fact that communication takes place is done through the use of an ant-based routing algorithm, wherein direct connections between communicating nodes are not established. Through the use of pseudonyms, the avoidance of creating direct connections, and link encrypted communications between neighbouring nodes, tracking communications between two nodes is extremely difficult.

An updated version of Table 3.6 is presented below, which includes the protocol presented in this document. In all instances shown in the table, from an adversaries point of view, the user in question appears no more likely to have originated the action than to not to have. The group of neighbouring nodes surrounding the node in question would all have to be considered as likely suspects, and thanks to the use of pseudonyms, the measures to prevent time-to-live attacks, and the use of link encryption (making outgoing packets appear different than incoming ones) all nodes within this group are equally likely to have performed an action. As nodes are required to have at a minimum four neighbours, this corresponds to a particular node having a 20% probability of having initiated an action if the smallest set of surrounding nodes is considered. This corresponds to the notion of Probable Innocence presented in Chapter 2, as the probability of this node having performed the action is less than 50%. Even if the minimum number of neighbours was reduced to two, the probability of a specific

node having performed an action would still only be 33%, corresponding to the same degree of anonymity.

The use of hop-by-hop routing and avoiding direct connections between peers, again along with the use of pseudonyms and link encryption, makes linking a sender and receiver extremely difficult, even to a global adversary. The use of threshold schemes to break apart messages into shares further obfuscates this linkage, as it creates a notion of security through obscurity, with additional shares being required to decipher the message contents.

**Table 5:3 - Anonymity Provided By Different Designs**

| | Freenet | MIX networks | Onion Routing | Ant-Based | **Secure Communications** |
|---|---|---|---|---|---|
| Sender anonymous to Global Adversary | No | No | No | No | **Prob. I.** |
| Responder anonymous to Global Adversary | No | No | No | No | **Prob. I.** |
| Sender anonymous to Responder | Prob.I. | B.S. | B.S. | Prob. I. | **Prob. I.** |
| Sender anonymous to Node | Prob. I. | No | No. | Prob. I. | **Prob. I.** |
| Responder anonymous to Sender | No | No | No | Prob. I. | **Prob. I.** |
| Responder anonymous to Node | No | No | No | Prob. I | **Prob. I...** |
| Sender-Responder unlinkable to Node | Prob. I. | B.S. | B.S. | Prob. I. | **Prob. I.** |
| Sender-Responder unlinkable to Global Adversary | No | B.S. | B.S. | No | **Prob. I.** |

The use of ant-based routing provides Probable Innocence for all views of anonymity with the exception of a Global Adversary. However, as previously mentioned, the use of hop-by-hop routing, along with the incorporation of pseudonyms to protect the actual location of a network participant, the use of link encryption to obfuscate the correlation between incoming and outgoing messages at a node, and the TTL anonymity measures all combine to provide Probable Innocence for sender-receiver unlinkability to a global adversary. All of these techniques make the origin and final destination of a message equally likely among a set of nodes, and make tracking the message across the network extremely difficult. In addition, the use of threshold schemes to break the message apart into shares introduces another level of complexity, as communication between two nodes is not simply the passing of a single packet back and forth across a path, but involves tracking several shares between nodes over varying paths. To a global adversary watching traffic across the network, this makes it equally likely that any two nodes are in actual communication.

## *5.4 Data Integrity*

As with several of the protocols presented in Chapter 4, data integrity within this system is chiefly provided through the use of Shamir's threshold scheme. As a single share of a message gives no meaningful information about the message as a whole, any modifications to a single share could not be accomplished in a manner to achieve any predictable result. In addition, any modified shares would not be useable

in the reconstruction of the full message, making modified shares easy to detect and remove from use. See section 5.2 above for the discussion and simulation results of the use of threshold scheme in combination with routing the shares across disjoint paths as an effective measure of providing both data integrity and data confidentiality.

## *5.5 Security Measures Impact*

This section will examine simulation results to demonstrate the minimal impact that the measures used to implement secure communications have on the performance of the routing protocol as a whole. Three main aspects will be examined in the following sections. First, the validity of the maximum TTL values are examined with respect to average hop count. Then the impact on splitting a message into shares with respect to the additional time it takes to receive additional shares and reconstruct is examined, and finally the routing load at a node is examined with and without the TTL anonymity measure in place.

### 5.5.1 Average Hop Count

A commonly used maximum TTL value within P2P networks is 7 [RF02]. This was the maximum TTL value that the system was attempting to achieve with the exception of the Hello messages. As neighbours were attempted to be selected in a proximal fashion, a lower value for maximum Hello TTL value is used. For the sender TTL anonymity measures employed, messages had a 1 in 3 chance of switching modes, based on the initial generator state. This leads to an expected hop count of 3 for this

portion of the routing. The expected number of additional hops generated by the receiver TTL anonymity measures is 0.5. This leads to an overall expected additional hop count of 3.5. Thus, in order to achieve a value close to 7, the maximum TTL value in the simulations was set to 3. The lowest maximum value that could be used for Hello messages was 1, giving an expected hop count of just over 4. Figure 5.5 below presents the results from the simulations for the average hop count of a packet within the system.

As can be seen in Figure 5.5, as the network size increases, so does the average hop count. In the 100-node networks both the hop count of Hello and all other messages are very close to their expected values. In the smaller network sizes, these values are below what is expected. There could be numerous explanations for this, the most compelling being that in the smaller networks, the average network diameter is not large enough to actually accommodate such a large hop count. Another reason is that duplicate messages are being dropped more frequently, as there are fewer paths between nodes, and nodes themselves, to propagate the messages forward. The important aspect of these results is that with the anonymity measures in place, within a reasonably sized network, the expected average number of hops can be achieved. This allows for, as in a normal TTL scheme, a measure of control over the scope limiting mechanism, allowing for fine tuning if desired, with predictable results.

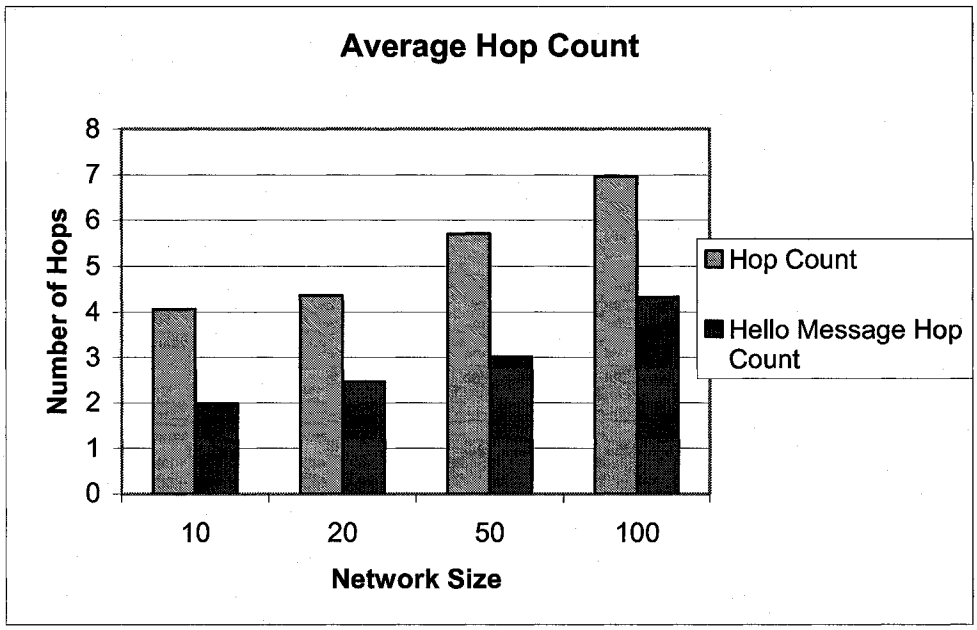


**Figure 5.5 - Average Hop Count**

### 5.5.2 Message Reconstruction Delay

The introduction of several shares of a message introduces a delay into the system as opposed to a system where messages are sent as a whole. This delay is the result of a destination node having to wait for several shares to arrive before it can reconstruct a message, as opposed to only needing to wait for a single packet that carries the entire message. The ratio between the average time it took for a packet (single share) to reach the intended destination and the time it took from when the message was created until it could be reconstructed are shown in Figure 5.6 below.

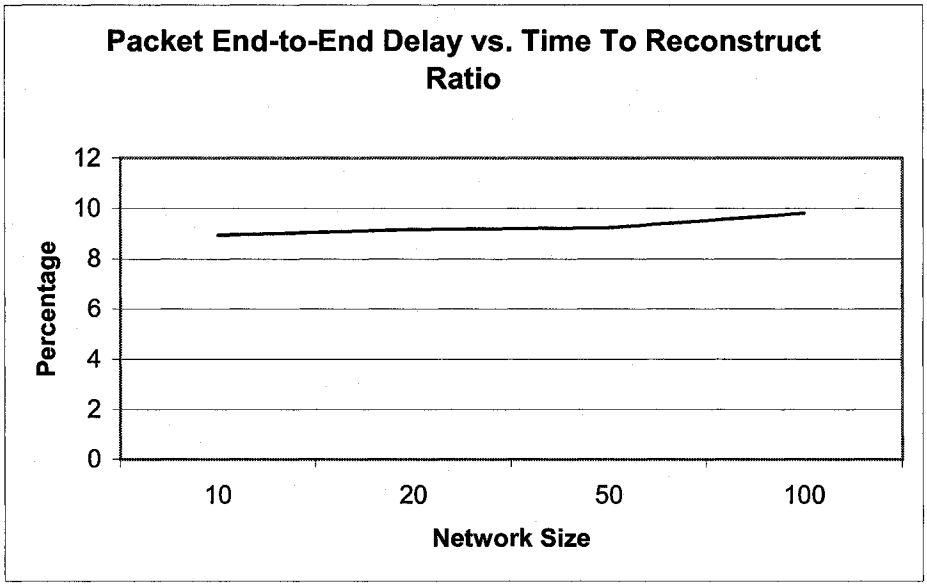


**Figure 5.6 - Packet End-to-End Delay vs. Time to Reconstruct Ratio**

In a network of 100 nodes, the additional latency introduced by routing shares across disjoint paths is 9.8%, compared to 8.9% in a network of 10 nodes. This is a minimal increase in latency percentage when compared to a ten-fold increase in network size. As can be seen in Figure 5.6, the slope of the line is near zero (0.01), indicating a relatively constant ratio between the time it takes a single packet to reach the destination as compared to the time it takes k shares to reach the intended destination and be reconstructed. As network size increases, so does the average end-to-end delay. The increase in end-to-end delay is a result of the average path length increasing as the network size increases. A longer path length results directly in a longer end-to-end delay. However, even in situations with larger end-to-end delays, the ratio presented in Figure 5.6 stays nearly constant. As the network of 100 nodes has

an average hop count of near 7, and this is the expected hop count for all network sizes, this ratio should continue to stay consistent for larger network sizes. Further experimentation would need to be explored in order to confirm this observation.

### 5.5.3 Routing Load

Within the simulation framework, routing load was defined as the number of protocol specific packets that were processed at a given node per second. This was measured to help ascertain the additional routing load that the TTL anonymity measures introduce into the system. As can be seen in Figure 5.7 below, as the network size increased, so did the average routing load at a node. This is as a result of more nodes being within the network, resulting in a larger number of protocol messages being introduced into the system. In addition, as the average node degree increases with network size, this also means that a greater number of shares are being created per message. Another possible reason for the lower load at smaller network sizes is that the expected hop count is also not achieved at small network sizes. Messages therefore do not travel as far through the system, decreasing the overall network load. Thus, there is a twofold increase in the number of messages to process as network size increases. The standard deviation remains fairly constant across varying network sizes, indicating a fairly balanced load across the network.

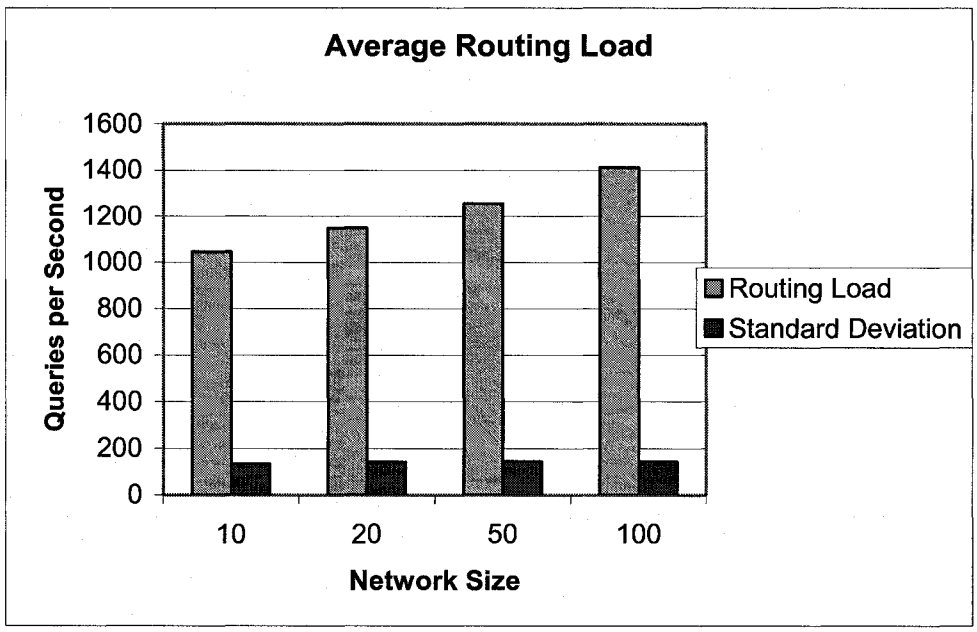


**Figure 5.7 - Average Routing Load with TTL Anonymity**

Figure 5.8 below illustrates the average routing load in the system with the TTL anonymity measured disabled. As can be seen, while there is a decrease across the board in terms of routing load, the difference is acceptable in terms of the objections put forth in the contributions section of Chapter 1. Again, as network size increases, so did the routing load per node, but not at a similar rate. This is a result of smaller network sizes attaining the desired expected hop count, whereas with the TTL anonymity measures in place, the average hop count at smaller network sizes was lower than expected. Therefore, with the TTL anonymity measures turned off, there is one less contributing factor to the variance in routing load across network sizes.

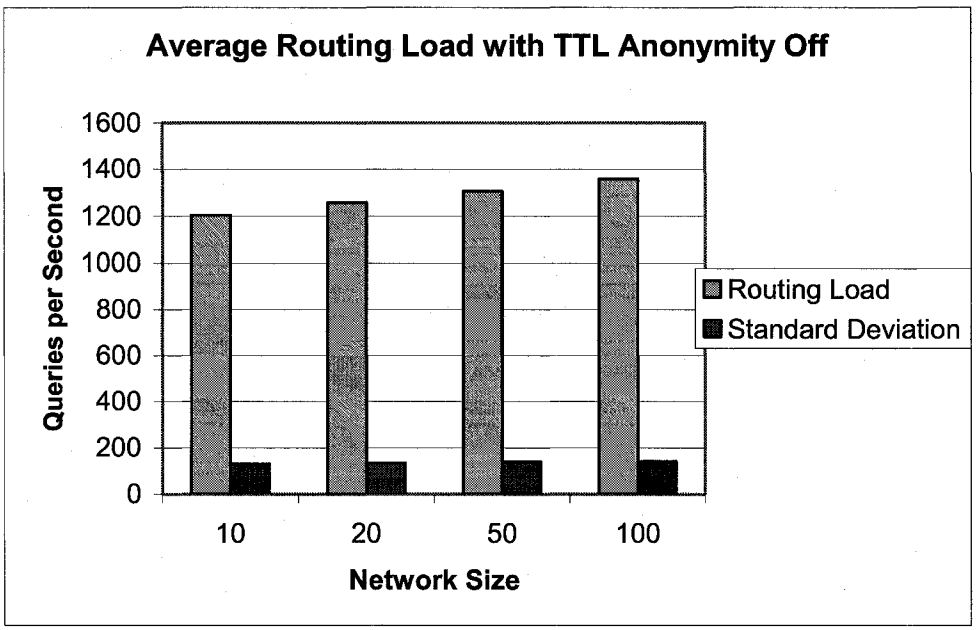


**Figure 5.8 - Average Routing Load with TTL Anonymity Off**

As Table 5.4 indicates, at larger network sizes, the system behaves in an almost identical manner, as the average hop counts are similar, along with the number of protocol participants and the number of messages being generated. For the 100 node network, results with TTL anonymity measures turned on were actually higher. This again is a result of the variable number of hops a packet can take, as opposed to a concrete number when the TTL anonymity measures are turned off. As most implemented P2P systems have hundreds of thousands of users, the impact of having these TTL anonymity measures should be negligible.

**Table 5:4 - TTL Anonymity Cost Ratio**

| Network Size | TTL Anonymity On | TTL Anonymity Off | TTL Anonymity On : Off Ratio |
|---|---|---|---|
| 10 | 1045.8 | 1203.0 | 13.1 |
| 20 | 1148.8 | 1258.1 | 8.7 |
| 50 | 1256.2 | 1304.2 | 3.7 |
| 100 | 1401.9 | 1356.5 | -3.3 (3.2 Off : On Ratio) |

## *5.6 Weaknesses*

This protocol does not solve all security issues. While it has been shown to include mechanisms to provide identity protection, data confidentiality and data integrity against several attack types, including time-to-live attacks, multiple colluding adversaries, malleability attacks, eavesdropping attacks and statistical attacks, it does not provide protection against all attack types.

Denial of service attacks may be performed against the protocol. This type of attack could impact a single node or small region of the network, but the use of routing shares across disjoint paths should make it difficult for this type of attack to have a network wide impact. Similarly, while not offering complete protection against time-based attacks, the protocol does aim to make them more difficult, through the extra stages of TTL manipulation.

There are also some aspects of the protocol which may be exploited for malicious purposes. For example, one potential weakness could include a pheromone spoofing attack, where the pheromone tables could be attacked with known pseudonyms. Another possible attack would be to manipulate the timestamps on the

packets. Further investigation is required to find methods to mitigate these types of attacks, but some of the approaches presented in [HS91] may lead to potential solutions.

## 5.7 Summary

The chapter has presented an analysis of how secure communication is achieved, through providing data confidentiality, identity protection, and data integrity. While the concepts of link encryption and the properties of Shamir's threshold scheme are proven to be secure, their application within the system required some validation. This was afforded through simulation results.

The simulation environment was presented, and then simulation results were presented and analyzed. These results showed that a high percentage of shares reached the intended destination, while a low percentage was seen by a single intermediary node or a group of colluding nodes. This reinforced the notion of the threshold schemes and disjoint path routing as affording both data confidentiality and data integrity. It was also shown that as network size increases both the jointness percentage and the percentage of shares seen at intermediary nodes decreased.

The anonymity measures put in place to help provide identity protection were also analyzed in terms of impact on the network. It was shown that the expected hop counts were experienced in larger sized networks, creating a similar routing load to a network with concrete maximum TTL values. It was also shown that the delay

introduced by having to receive several shares to reconstruct a message was minimal when compared to the average end-to-end delay of a single share.

The next chapter will review the contributions of the protocol presented in this document, along with considerations for future work.

# Chapter 6

# Conclusions and Future Work

This chapter summarizes the contributions made by this thesis, and suggests areas or directions for future work. It will begin by offering a summary and conclusions of the work presented in the thesis.

## *6.1 Conclusion*

This document has presented an adaptive routing protocol that achieves identity protection, data confidentiality and data integrity in an anonymous overlay network. It has shown how within an anonymous network these are the aspects of secure communication that are relevant, and has incorporated disjoint routing and specific anonymity algorithms to achieve these goals. Simulation results show that at larger network sizes, the average additional routing load due to incorporated anonymity measures is minimal, and can actually be a negative value, thanks to the variance in the number of hops a packet can take. The additional latency incurred by routing shares of a message across disjoint paths was shown to be near 10% in excess of the average end-to-end delay experienced by a single packet. Both of these results are below the performance objectives set forth in Chapter 1.

Several related technologies were presented and reviewed, and it was shown how no single one of these systems achieved the same level of secure communication that the protocol presented within this document does.

Through the use of an adaptive, ant-based routing algorithm and Shamir's threshold scheme, messages were broken into shares and routed across disjoint paths between source and destination nodes. Simulation results validated that while a sufficient number of shares reached the intended destination, intermediary nodes were unable to collect enough shares to reconstruct a message. These techniques, along with the use of link encryption, provided both data confidentiality and integrity. Against a group of colluding nodes, while not compromising data confidentiality, simulations results showed that there is a possibility of disrupting communication, as the destruction of up to 30% of the shares of a message were seen. In these circumstances, however, up to 20% of the nodes in the network were colluding. These are extreme circumstances, and in reality a much smaller percentage of nodes would be expected to be adversarial, and thus a smaller percentage of shares would be expected to be intercepted.

The use of pseudonyms within the network, along with avoiding direct connections between communicating nodes helped to provide identity protection for participants. In order to strengthen this notion in a local sense, and to prevent TTL based anonymity attacks, additional measures were taken to ensure both sender and receiver anonymity in a local sense.

The combination of these techniques provided a tiered solution to each of the three aspects of secure communications identified in the problem statement. This overlap between approaches to achieve each of these aspects creates a system without a single point of failure, making it more resilient to an adversary than systems that only have a single defence mechanism.

## *6.2 Future Work*

Some enhancements to the current protocol could include incorporating other techniques to generate a more structured network design in order to try and optimize end-to-end latency. A more structured approach here could set a better foundation upon which the ant-based routing techniques could dynamically find optimal routes. Adding a mechanism to have system parameters adapt automatically to varying network conditions may also lead to improved overall performance, as would the inclusion of the ability to recover from failed message reconstruction attempts without having to initiate another complete resource request cycle. A prime candidate for this type of dynamic adjustment would the value of k within the use of the threshold scheme.

In order to help protect against the colluding node threat model, mechanisms to monitor neighbouring nodes behaviour may be a valuable addition. This could help to determine whether or not a neighbouring node can be trusted, and if not, to disassociate

from this node as a neighbour. This would most likely be accomplished by incorporating some notion of a trust model into the system.

The notion of using a threshold scheme to break apart resources to store at various peers could also be investigated. This would then add the notion of secure publishing into the system. Ants have also been shown to exhibit clustering properties in nature, and another class of algorithms exists that mimics this behaviour. The inclusion of this type of algorithm could be employed to create a Semantic Overlay Network (SOM) of nodes containing related content. This would involve another class of ant-based agents, that would aid the network in dynamically reforming to create these SOMs of similar content. Then resource request queries could be routed to the appropriate SOM, increasing the probability of a resource being found quickly, while also reducing the load on nodes with unrelated content.

The concept of routing across disjoint paths using competing pheromone types could also be used to achieve a Diffie-Helman key exchange between peers. This concept could be used to test, and possibly refute, Jason Rohr's claims that end-to-end encryption is impossible within an anonymous network. The use of routing the key across disjoint paths after splitting it apart using a threshold scheme would give senders a secure way to obtain receiver keys. Exploratory agents could reinforce the paths independent of message exchange between peers.

Simulations run on larger network sizes and with more precise resource and request distribution algorithms would be beneficial to further validate the trends seen in simulations results presented in the previous chapter.

Introducing an upper bound on the pheromone concentrations may also help to stabilize the network, preventing pheromone table entries from growing out of proportion to other entries.

Future work also exists in the area of protecting some of the currently visible fields in the packet headers, such as the timestamp or the reconstruction threshold value. Preventing unwanted modification to these fields could not only stabilize performance in the network, but could also prevent some time-based attacks from being conducted.

The distribution of the tail node behaviour could be explored to be fine-tuned for particular network sizes. As discussed in the previous chapter, smaller networks currently have a small percentage of nodes that actually forward messages in the receiver anonymity portion of the TTL handling. While this is sufficient to introduce doubt to adversarial nodes, a larger percentage would increase the security of the protocol.

Finally, the integration of the presented protocol with other similar work such as could be explored, where the underlying routing mechanisms presented in this document could be used as the anonymous communications framework layer in systems such as Freenet or Free Haven.

# Appendix A

# Simulation Parameters

All simulations were run on the following hardware:

Intel Pentium M processor, 1.73GHz

1.00 GB RAM

The following software packages were used in the simulations:

Microsoft Windows XP SP2

NS-2 (v.2.29)

- The simulation framework

Cygwin 1.5.121

- To allow NS-2 to run on Windows

Nem 0.96

- Used to generate network topologies

**Network Topologies**

Networks of 10, 20, 50 and 100 nodes were generated. For each size of network, 5 different topologies were generated using Nem. Each topology was generated using the Albert-Barabasi generation method, with the following parameters specified in the .specif file:

- m0 – 3
- m – 3
- p – 0.6
- q – 0.1
- Network Level – Router Level

This generation method is an extended scale-free power-law model.

**Simulation Runs**

For each network topology generated at each network size, 20 different runs were completed. Each run of the simulation on a network were run for 12 million time slices. Resources were randomly distributed among nodes from a resource pool that was three times the size of the number of nodes. Thus, for a 10 node network, the resource pool would be 30. Nodes request a resource after every 200,000 time slices, upon completion of the initialization phase of the protocol, presented in Section 4.2. This simplistic resource distribution model was chosen in order to more effectively test the security afforded by the protocol.

**Colluding Nodes**

Nodes were given an independent probability of being deemed a colluding node. This percentage was 10% in the simulations. In this manner, on average, 10% of the total number of nodes were colluding, adversarial nodes. If a node was neighbouring a colluding node, this probability rose to 25%. The maximum number of colluding nodes was restricted to 20% of the network size.

**Simulation Parameters**

| Parameter | Value |
|---|---|
| Alpha | 5 |
| Beta | 1 |
| Gamma | 0.1 |
| Theta | 1 |
| $q_0$ | 0.1 |
| Initial Pheromone Value | $10^{-6}$ |
| Minimum Neighbours | 4 |
| Maximum Neighbours | 10 |
| n from (k,n) Threshold Scheme | Number of Neighbours |
| K from (k,n) Threshold Scheme | The floor of 80% of n |
| $p_{\min}$ | 1 |
| $p_{\max}$ | 10 |
| MAX_HELLO TTL | 1 |
| MAX_TTL | 3 |
| $W_{best}$ Window Time | 200,000 time units |
| Received Packets Buffer garbage collection time | Every 200,000 time units |
| Received shares expiration check timer | Every 200,000 time units |
| Received shares expiration period | 500,00 time units |

## References

[AB00] R. Albert and A-L Barabasi, Topology of evolving networks: local events and universality, Physical Review Letters, 85 p. 5234-5237, 2000

[ADS03] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In R. N. Wright, editor, Financial Cryptography Springer-Verlag, LNCS 2742, p. 84-102, 2003.

[BASM04] Steve Bono, Christopher A, Soghoian, and Fabian Monrose. Mantis: A high performance, anonymity preserving, p2p network, 2004. Johns Hopkins University Information Security Institute Technical Report TR-2004-01-BISI-JHU.

[BG03] K. Bennett and C. Grothoff, GAP – Practical anonymous networking, in Proceedings of Privacy Enhancing Technologies workshop, p.141-160, 2003

[CCR04] Miguel Castro, Manuel Costa, and Antony Rowstron. Peer-to-peer overlays: structured, unstructured, or both?. Technical Report MSR-TR-2004-73, Microsoft Research, Cambridge, UK, 2004.

[CC05] Tom Chothia and Konstantinos Chatzikokolakis. A Survey of Anonymous Peer-to-Peer File-Sharing, IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS 2005)

[CDG04] Gianni Di Caro, Frederick Ducatelle, and Luca-Maria Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In Proceedings of Parallel Problem Solving from Nature (PPSN) VIII, LNCS 3242. Springer-Verlag, p. 461-470, 2004.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 4(2), February 1981, p. 84-90.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong.Freenet: A distributed anonymous information storage and retrieval system.Lecture Notes in Computer Science, 2009:p. 46+, 2001.

[DA99] T. Dierks and C. Allen. The TLS Protocol — Version 1.0. IETF RFC 2246, January 1999.

[Dou02] J. Douceur. The Sybil attack, 2002. In Proceedings of the IPTPS02 Workshop, Cambridge, MA (USA), March 2002.

[DC99] Dorigo M. and G. Di Caro (1999). The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization, McGraw-Hill, p. 11-32.

[DD98] G. Di Caro and M. Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research, 9:p. 317--365, 1998.

[DDG05] Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, 16(5):p. 443-455, 2005.

[Din04] R. Dingledine, TOR: An Anonymous Internet Communications System, 2004

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, p. 67-95, July 2000.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In Proceedings of the 13th USENIX Security Symposium, p.303-320 2004.

[For01] Behrouz A. Forouzan, Data Communications and Networking, Second Edition, McGraw-Hill, 2001, ISBN 0-07-282294-5

[Guer97] S. Guerin, Optimisation multi-agents en environment dynamique: Application au routage dans les reseaux de telecommunications, DEA Dissertation, University of Rennes I, France, 1997.

[GKB03] M. Gunes, M. Kahmer, I. Bouazizi, Ant Routing Algorithm (ARA) for Mobile Multi-Hop Ad-Hoc Networks - New Features and Results, The Second Mediterranean Workshop on Ad-Hoc Networks, 2003.

[Gnu06] Gnutella, http://www.the-gdf.org/index.php?title=Main_Page, 2006

[Har05] Jonathon B. Harris, A Scalable & Extensible Peer-to-Peer Network Simulator, A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Computer Science At Ottawa-Carleton Institute for Computer Science, Carleton University, 2005

[HLX+05] Jinsong Han, Yunhao Liu, Li Xiao, Renyi Xiao, and Lionel M. Ni. A mutual anonymous peer-to-peer protocol design. In IPDPS, vol. 1, no. 1, p 68. IEEE, 2005.

[HON05] Halpern, J. Y., and O'Neill, K. Anonymity and information hiding in multiagent systems. Journal of Computer Security 13:3, 2005, p. 483-514

[HS91] Stuart Haber and W.-Scott Stornetta. How to Time-Stamp a Digital Document. Journal of Cryptology, 3(2):p. 99--111, 1991.

[JCJ00] C. Jin, Q. Chen, and S. Jamin. Inet: Internet Topology Generator. Tech. Rep. CSE-TR-433-00, EECS Department, University of Michigan, 2000.

[Kra94] H. Krawczyk, "Secret sharing made short," in Proceedings of the 13th annual of International cryptology conference in Advances in cryptology, p. 136-146, 1994

[Mag05] D. Magoni, Network Topology Analysis and Internet Modelling with Nem, International Journal of Computers and Applications, 202-1540, 2005 Issue

[MMB00] A. Medina, I. Matta, and J. Byers. On the Origin of Power Laws in Internet Topologies. Computer Communication Review, 30(2):p. 18--28, 2000.

[MOV97] Menezes, Alfred; van Oorschot, Paul; Vanstone, Scott (1997). *Handbook of Applied Cryptography* Boca Raton, Florida: CRC Press. ISBN 0-8493-8523-7.

[Mute06] MUTE, http://mute-net.sourceforge.net/, 2006

[NEM06] Network Manipulator (Nem), https://dpt-info.u-strasbg.fr/~magoni/nem/, 2006

[NS06] The Network Simulator – NS-2. Information Sciences Institute. The University of Southern California. 2006, http://www.isi.edu/nsnam/ns/.

[NVV04] Ann Nowé, Katja Verbeeck, Peter Vrancx, Multi-type Ant Colony: The Edge Disjoint Paths Problem, in Proceedings of Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, p. 202-213

[NZ03] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (pdf) system for packet switched networks," in Proceedings of IEEE INFOCOM, p. 663-672 Vol. 1, 2003

[Qur04] Qureshi, A., "Exploring Proximity Based Peer Selection in BitTorrent-like Protocol," MIT 6.824 student. project, 2004.

[Rab89] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," ACM JACM, p. 335-348, 1989.

[RF02] M. Ripeanu and I. Foster. Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems. IPTPS, p. 85-93, 2002.

[Roh03] Jason Rohrer, routingAlgorithm.txt, http://cvs.sourceforge.net/viewcvs.py/mute-net/MUTE/doc/routingAlgorithm.txt?rev=1.1&view=log, 2003

[Roh04] 21. Jason Rohrer, End-to-end encryption (and Person-in-the-middle attacks), Technical Article, http://mute-net.sourceforge.net/personInTheMiddle.shtml, 2004

[Rot03] Martin Roth, Swarm Intelligent Networking, Cornell University, 2003

[RR98] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. ACM Transactions on Information and System Security, 1(1):p. 66–92, 1998.

[SBS02] Rob Sherwood, Bobby Bhattacharjee, Aravind Srinivasan, P5: A Protocol for Scalable Anonymous Communications, Proceedings of the 2002 IEEE Symposium on Security and Privacy, p. 58, 2002

[SGR97] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In IEEE Symposium on Security and Privacy, p. 44-541997.

[SHBR96] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, "Ant-based load balancing in telecommunication networks", Adaptive Behavior, vol. 5, p. 169-207, 1996.

[SLS01] V. Scarlata, B. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing, 2001. In Proceedings of IEEE International Conference on Network Protocols (ICNP) p.272, 2001.

[Ser02] A. Serjantov, Anonymizing Censorship Resistant Systems, in Proceedings of the First International Workshop on Peer-to-peer Systems, p.111-120, 2002

[SW02] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In Internet Measurement Workshop, p.219-232, 2002.

[Tan03] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall PTR, 2003, ISBN 0-13-066102-3

[TGJ+01] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topologies, power laws and hierarchy, Tech. Rep. TR01-746, Technical Report, University of Southern California, 2001.

[THV04] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda, Measuring Anonymity Revisited, In the Proceedings of the Ninth Nordic Workshop on Secure IT Systems, Espoo, Finland, November 2004, p. 85-90.

[Tor06] Tor: Overview, http://tor.eff.org/overview.html.en, 2006

[Whi00] White T., SynthECA: A Synthetic Ecology of Chemical Agents. Carleton University Ph.D., August 2000

[Whi05] White, T., Expert Assesment of Stigmergy: A Report for the Department of National Defence, http://www.scs.carleton.ca/~arpwhite/stigmergy-report.pdf, 2005