ISR Institute for Software Research
University of California, Irvine

SIFT: A Simulation Framework for Analyzing Decentralized Reputation-based Trust Models

Girish Suryanarayana
University of California, Irvine
sgirish@ics.uci.edu

Richard N. Taylor
University of California, Irvine
taylor@ics.uci.edu

August 2007

ISR Technical Report # UCI-ISR-07-5

Institute for Software Research
ICS2 217
University of California, Irvine
Irvine, CA 92697-3455
www.isr.uci.edu

SIFT: A Simulation Framework for Analyzing Decentralized Reputation-based Trust Models

Girish Suryanarayana and Richard N. Taylor
Institute for Software Research
University of California, Irvine
{sgirish,taylor}@ics.uci.edu

ISR Technical Report # UCI-ISR-07-05

August 2007

Abstract: Open decentralized applications are susceptible to the attacks of malicious entities. In such applications, each autonomous entity must adopt protective measures to safeguard itself. One set of such countermeasures are reputation-based trust management systems. However, designing these systems is arduous because the impact of factors introduced by decentralization on such systems is largely unknown. There is a lack of knowledge in existing literature that can guide the design of an appropriate trust solution. To address this shortcoming, we present a simulation-based framework called SIFT that allows a designer to explore and analyze the interplay of various trust and application settings. SIFT-based experiments with various trust and application settings have not only helped expose the pros and cons of different trust settings but also revealed several interesting insights that guide the selection and refinement of a set of trust settings for a given operating condition.

SIFT: A Simulation Framework for Analyzing Decentralized Reputation-based Trust Models

Girish Suryanarayana and Richard N. Taylor

Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
{sgirish,taylor}@ics.uci.edu

ISR Technical Report # UCI-ISR-07-05

August 2007

Abstract

Open decentralized applications are susceptible to the attacks of malicious entities. In such applications, each autonomous entity must adopt protective measures to safeguard itself. One set of such countermeasures are reputation-based trust management systems. However, designing these systems is arduous because the impact of factors introduced by decentralization on such systems is largely unknown. There is a lack of knowledge in existing literature that can guide the design of an appropriate trust solution. To address this shortcoming, we present a simulation-based framework called SIFT that allows a designer to explore and analyze the interplay of various trust and application settings. SIFT-based experiments with various trust and application settings have not only helped expose the pros and cons of different trust settings but also revealed several interesting insights that guide the selection and refinement of a set of trust settings for a given operating condition.

1. Introduction

Consider a decentralized auctioning system where distributed entities buy and sell commodities. There is no single centralized authority that controls and coordinates trade between individual entities. Instead, each entity, also called a peer, is autonomous, interacts directly with other peers, and makes local independent decisions towards its individual goals that may possibly conflict with those of other peers. In an open decentralized auction, peers including those with malicious intentions can freely enter the system. These peers may execute a variety of attacks such as impersonating other peers to leverage their privileges, promising non-existent services, failing to fulfill promised obligations, and spreading false information about other peers in the system. These attacks pose a significant threat to the integrity of the system and therefore must be effectively countered.

In the absence of a centralized authority that can regulate the entry of malicious entities and help safeguard the system against such malicious attacks, it becomes the responsibility of each decentralized entity to adopt suitable measures to counter these attacks. Reputation-based trust management systems have been found to serve as potential countermeasures against such attacks [25]. These systems rely upon a peer's reputation molded by its past interactions to determine its trustworthiness in the future.

Our previous work [21] proposed a novel architectural style, PACE, that guides the incorporation of reputation-based systems within the architecture of a decentralized peer. PACE provides guidance on how to build trust-centric decentralized applications, but does not specify what trust model should be chosen for a given application. In order to realize a secure decentralized application, choosing an appropriate and reliable trust model for that application is as important as building it right.

Our next focus, therefore, was to investigate how to select a trust model for a particular application. We, however, discovered that researchers have devoted little attention towards this goal. Moreover, existing reputation models are geared towards different needs and applications, and so it is rather difficult to decide what model should be adopted for one's use.

To address this need, we have built a simulation framework called the **SI**mulation-based **F**ramework for **T**rust models (SIFT). SIFT uses scenarios based on the critical threats to a decentralized system to explore and analyze the interplay of trust and application settings. Specifically, SIFT simulates how a trust model, characterized using specific parameters, will behave when subjected to different types of attacks in a particular application setting.

We have used SIFT to simulate numerous trust and application settings. Our experience with using SIFT has highlighted its usefulness. SIFT enables the designer to quickly experiment with different trust and application settings. SIFT simulation results expose the benefits and deficiencies of trust settings under different operating conditions which provides much-needed guidance towards the selection of an appropriate trust model for a given application and points at future refinements to those models.

The rest of the paper is structured as follows. Section 2 provides a brief background to trust and reputation. Section 3 discusses relevant related work. Section 4 describes the approach and various aspects of SIFT. SIFT design and implementation details are presented in Section 5 while Section 6 describes our experiments with SIFT and presents several interesting insights based on our simulation results. Limitations and future work are discussed in Section 7.

2. Background

The concept of trust is not only basic to society but also undoubtedly significant. Therefore, interest in it is not limited to electronic communities but reaches across several research disciplines such as psychology, sociology, computer science etc. [17]. We limit our discussion here to our definitions of trust and reputation.

2.1 Trust

A number of definitions for trust exist in the literature. Some of the more popular ones include those proposed by Deutsch [7], Marsh [17], Diego Gambetta [9], and Grandison and Sloman [10]. For our purposes, we adopt the Gambetta's definition of trust who defined trust as *a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action*. Gambetta also introduced the concept of using values for trust and defended the existence of competition among cooperating agents.

2.2 Reputation

The concept of reputation is closely related to trust and can be used to determine the trustworthiness of an entity [25]. Abdul-Rahman [1] defines reputation as *an expectation about an individual's behavior based on information about or observations of its past behavior*. In online communities, where an individual may have very little direct information to determine the trustworthiness of others, their reputation information is typically used to determine the extent to which they can be trusted. An individual who is more "reputed" is generally considered to be more trustworthy. Reputation can be determined in several ways. For example,

a person may either rely on his direct experiences, or rely on the experiences of other people, or existing social relationships or a combination of the above to determine the reputation of another person.

3. Related Work

As previously mentioned, recognizing the significant role of reputation-based trust management in decentralized applications, researchers have recently devoted increasing attention to the exploration of such systems. Researchers have not only invented a number of decentralized reputation models [20] but have typically also evaluated those models using custom-made simulators. The approach adopted in SIFT draws from some of these existing custom-made simulators. We, therefore, examine these simulators and where relevant point out similarities and differences in their approaches with that used in SIFT. We will also discuss briefly four decentralized reputation models that have been used in our evaluation of SIFT.

3.1 Trust Model Simulators

Yu and Singh [24] present a weighted majority trust algorithm that assigns initial weights to recommenders and after every successful or unsuccessful interaction fine-tunes these weights accordingly. The evaluation of this algorithm consists of alternating phases of querying and trust evaluation in every simulation round. A random number of peers are chosen to evaluate trust in the rest of the peers. However, the simulation testbed assumes a ring-based topology for the peers which does not model the structure of real-world decentralized applications.

Credence [22] is a reputation-based system that evaluates the reputation of objects in a peer-to-peer file-sharing application. It uses a voter correlation scheme that weighs peer opinions and helps identify objects that may be tainted. Credence has been evaluated in a custom-made Java-based discrete event simulator. However, due to the nature of the trust model, the application context used in the simulations is strictly limited to file-sharing. Additionally, the use of a simple randomized topology to model the underlying peer-to-peer network does not help predict the behavior of the trust model under real-world application settings.

The NICE platform [14] includes trust algorithms that enable good peers to form robust cooperative groups with other good peers in order to counter the threat of false information reported by malicious peers. These algorithms are evaluated through simulations. However, an in-depth description about the simulation setup is lacking. Assumptions made about the underlying application topology are also missing.

PeerTrust [23] is a reputation-based trust model for eCommerce communities that uses P-Grid for distributed data storage and retrieval. Like other trust models, it also focuses on incorporating the credibility of recommenders in the evaluation of trust. But PeerTrust additionally includes the number of transactions and transaction and community context to make trust evaluation more accurate. PeerTrust has been evaluated using a simulator built in Mathematica 4.0. Simulations include an initial number of interactions between random peers followed by the evaluation of trustworthiness of a fraction of the peers by a fixed number of randomly selected peers. SIFT adopts a similar structure in its simulations. However, similar to some of the above-described systems, the evaluation of PeerTrust does not clearly specify what the application topology is and how close the assumed application structure is to real-world eCommerce applications.

PET is a trust model that combines reputation and risk evaluation to determine trustworthiness in P2P resource sharing applications [15]. Simulation for PET is thread-based and written in Perl. The objective of the simulation is to assess the effect of PET components in the context of a P2P Web server sharing application. In such an application, only the web servers form the P2P system. However, the evaluation

approach used in PET does not specify assumptions made about the underlying topology of this P2P system and whether the simulation models a real-world scenario. Further, no simulation results are presented for a case where malicious peers actively collude to subvert the system.

Marti and Garcia-Molina [18] propose a limited reputation sharing scheme to reduce the number of failed interactions between peers. This scheme is evaluated using a custom P2P simulator that assumes a Gnutella-like flat unstructured network topology in order to model real-world applications. The simulation process consists of a set of timesteps with a query being generated at every timestep and being completely evaluated before the next timestep. An interesting feature of this simulation testbed is that peer turnover is modeled, albeit in a simplistic manner by having a peer leave the system and replace that peer's place in the network by another peer.

The EigenTrust reputation system [12] uses a distributed method based on Power iteration to compute global trust values for all peers in P2P file-sharing applications. The simulation platform interconnects networks in a power-law network fashion which models prevalent real-world P2P networks [19]. Simulations with different threat scenarios are executed to examine the behavior of EigenTrust in the face of those threats. Each simulation cycle consists of a number of querying cycles at the end of which a peer downloads the file from a selected node. At the end of each simulation cycle, global trust values for each peer is computed. Though there are several similarities between approach used in EigenTrust simulation and SIFT, one drawback of the EigenTrust simulation is that it seems to be limited to file-sharing applications.

3.2 Decentralized Reputation-based Trust Models

We describe below four reputation-based trust models that are used in our evaluation of SIFT which is discussed in Section 6.

3.2.1 Distributed Trust Model

In the Distributed Trust Model (DTM), a trust relationship is always between exactly two entities, is non-symmetrical, and is conditionally transitive. Mutual trust is represented as two distinct trust relationships. Two different types of trust relationships are distinguished. When one peer trusts another, it constitutes a direct trust relationship. But if a peer trusts another peer to give recommendations about another peer's trustworthiness, then there is a recommender trust relationship between the two [1]. Trust relationships exist only within each peer's own database and hence there is no global centralized map of trust relationships. Corresponding to the two types of trust relationships, two types of data structures are maintained by each peer - one for direct trust experiences and another for recommender trust experiences. Recommender trust experiences are utilized for computing trust only when there are no direct trust experiences with a particular peer.

Trust categories are used by peers to classify trust towards other peers depending upon which aspect of that entity is under consideration. For example, a peer may trust another peer on a certain issue but may not trust it in another context. Similarly, since a peer may trust a certain peer more than other peers, comparable trust values are needed. A reputation is defined as a tuple consisting of a peer's name, the trust category, and the specific trust value. A recommendation is defined as communicated trust information which contains reputation information. Discrete integral trust values are used to represent the trustworthiness of peers with -1 indicating distrust, 0 indicating lack of knowledge, 1-3 indicating increasing trust and 4 indicating complete trust.

Three types of messages are exchanged between peers in this trust model: Request for Recommendation, Recommendation, and Refresh messages. When a peer needs a service offered by another peer for the first time (no prior transactions between the two), the peer sends out a Request for Recommendation message to the peers it trusts as recommenders. These recommender peers can respond by sending Recommendations if they know the target peer, else they forward the request to other peers whom they trust as recommenders. Since the opinion of peers may change over time, recommendations are valid only for a limited time. When recommendations expire or the trust values associated with them change, they are updated using Refresh messages. Refresh messages are also used to revoke Recommendations by sending Refresh messages with trust values 0.

3.2.2 Complaint-based Model

A complaint-based reputation model relies on negative feedback or "complaints" to convey reputation information. In such a model, peers do not store information about successful interactions or trustworthy peers, but rather record their negative experience in the form of complaints against interacting peers. These complaints are also forwarded to another peer or peers in the system in order to disseminate information about the malicious peer. When a peer wants to evaluate the trustworthiness of a target peer, it first searches its own history to locate any previous complaints registered by itself. It can also query other peers for other existing complaints about the target peer. Complaints received from other peers are included in the determination of the target peer's trustworthiness. This kind of complaint-based scheme has been adopted by trust management systems such as the one based on the P-Grid data structure [2].

3.2.3 XREP

XREP is a distributed protocol that allows these reputation values to be maintained and shared among the servents. It consists of the following phases: resource searching, resource selection and vote polling, vote evaluation, best servent check, and resource downloading. Resource searching is similar to that in Gnutella and involves a servent broadcasting to all its neighbors a Query message containing search keywords. When a servent receives a Query message, it responds with a QueryHit message. In the next phase, upon receiving QueryHit messages, the originator selects the best matching resource among all possible resources offered. At this point, the originator polls other peers using a Poll message to enquire their opinion about the resource or the servent offering the resource. Upon receiving a Poll message, each peer may respond by communicating its votes on the resource and servents using a PollReply message. These messages help identify reliable resources from unreliable ones, and trustworthy servents from fraudulent ones.

In the third phase, the originator collects a set of votes on the queried resources and their corresponding servents. Then it begins a detailed checking process which includes verification of the authenticity of the PollReply messages, guarding against the effect of a group of malicious peers acting in tandem (collusion) by using cluster computation, and sending TrueVote messages to randomly selected peers suspected of belonging to a cluster. The TrueVote messages request confirmation on the votes received from the recipients. At the end of this checking process, based on the votes received through TrueVoteReply messages, the peer may decide to download a particular resource.

Since multiple servents may be offering the same resource, the peer still needs to select a reliable servent. This is done in the fourth phase where the servent with the best reputation based on the votes received from other peers) is contacted using a AreYou message to check the fact that it exports the resource. Upon receiving a reply from the servent in the form of a AreYouReply message, the originator downloads the resource from that servent in the final phase. It also updates its repositories with its opinion on the downloaded resource and the servent who offered it.

3.2.4 eBay Reputation Management System

eBay is an electronic marketplace where users sell and buy different kinds of goods [8]. Sellers advertise items and buyers place bids for those items. After an auction ends, the winning bidder buys the advertised item from the seller. Both buyers and sellers rate each other after the completion of a transaction. A positive outcome results in a +1 rating and a negative outcome results in a -1 rating. These ratings form the reputation of buyers and sellers.

This reputation information is stored and maintained by eBay which allows this to be viewed through user profiles. A user can click on the profile of a buyer or seller to view their past interaction history and trust information. The profile includes the number of total interactions a user has been involved in along with his total trust score. This score is obtained by adding all the unique positive ratings and subtracting all the unique negative ratings the user has received from other users. The profile also lists the total number of positive, negative and neutral ratings the user has received. Further, the profile also displays the aggregation of the most recent ratings received in the last one month, six months, and twelve months. A user viewing a profile can also choose to read all the comments written about the particular buyer or seller.

The eBay reputation system does not include a trust computation mechanism. It only presents reputation information about a user and does not provide an algorithm that computes a trust value for the user. The eBay reputation system leaves the trust determination and decision-making to its users. While this definitely gives the users flexibility to use the reputation information in their specific ways, it is quite difficult for an average user to determine who can be trusted. A user may have several good comments and may have a high rating but this does not mean that that particular user is the most trustworthy or that he will not renege on the delivery of the promised service or items. This can potentially affect the reliability of the eBay reputation system.

4. SIFT Approach

The main objective of SIFT is to be able to experiment with different trust and application settings under specific threat conditions. Towards this objective, the SIFT approach identifies and utilizes three sets of parameters: trust model parameters, application parameters, and simulation parameters. In order to be able to quantify how well each trust setting fared against these threats, SIFT also uses suitable assessment metrics. Figure 1 summarizes how these different parameters and metrics are used in SIFT. In the following sections, we introduce each of these parameters and metrics and the symbols used to denote them. An overview of all these parameters is also presented in Table 1.

4.1 Trust Model Parameters

In SIFT, a trust model is an instantiation of a specific set of values assigned to trust parameters. It should be noted that this set of trust parameters is not exhaustive, but is so chosen because these parameters are generally found in most reputation-based trust models. We, therefore, believe these parameters serve to allow a generic representation of trust models for the purposes of their examination. These parameters are described below.

4.1.1 Trust weight (alpha)

Trust models typically base trust decisions on two information sources - a peer's personal experience and recommendations received from other entities in the system. Depending upon the nature of the trust model, these two information sources are assigned different weights and appropriately combined in a weighted-average manner to give a resultant trust value. We model this aspect of trust models by assigning a trust

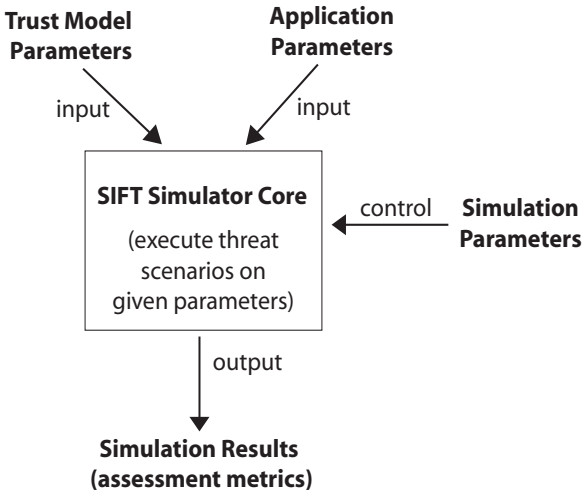

Fig 1. Overview of SIFT Simulator

Table 1. Overview of SIFT Parameters

Parameter	Symbol	Value	Description
Trust			
Trust Weight	alpha	0 - 1	Weight assigned to personal experience. Influences trust computation
Trust Threshold	thr	0.1 - 0.9	Level above which a peer is trusted and vice versa.
Recent Interaction Weight	val	0-1	Weight assigned to recent interactions during trust computation
Interaction Limit	lim	1-100	Percent of interactions considered recent in trust computation
Degree of trust in unknowns	deg	0 - 1, 2	Denotes how much an unknown peer is trusted.
Hop Count	hc	1-5	Determines the depth to which trust queries traverse
Application			
Number of interactions	int	0-100000	Number of initial peer interactions in each simulation
Connection Density	oe	1-5	Minimum number of peers each new peer must be connected to
Number of Peers	peers	100	Total number of peers in the system
Malicious Peers	mal	0-75	Number of malicious peers in the system
Malicious Probability	malProb	0-1	Probability with which malicious peers exhibit fraudulent behavior in their interactions
Simulation			
Number of assessing peers	aPeers	1-20	Number of peers that query and evaluate other peers
Number of assessments	aCount	1-5	Number of times querying and evaluation is repeated in a simulation
Type of attack	attack	0-3	Describes the attack scenario to be executed
Number of targeted peers	target	1-10	Number of good peers targeted by colluding peers
Number of simulations	simCount	50	Number of times each set of settings is simulated

weight, alpha, to the personal experience-based trust value. The recommendation-based trust value is correspondingly assigned a trust weight equivalent to '1 - alpha'. The value of alpha determines the impact that personal experience and received recommendations have on the final computed trust value. The trust weight thus plays a significant role in determining the nature of the trust model.

4.1.2 Trust threshold (thr)

The trust threshold represents the trust value below which all entities are considered untrustworthy. Correspondingly, any entity that has a trust value above the trust threshold value is considered trustworthy.

The trust threshold value used depends upon the nature and needs of the application. If peer interactions are of a critical nature, such as in e-commerce applications, peers in the system would prefer to not interact at all rather than face the risk of interacting with a malicious peer. In such a case, the trust threshold is typically set a high value to reduce the risk of interacting with a malicious peer. However, if failed interactions do not have significant repercussions and the focus is on encouraging a greater number of interactions, the trust threshold can be set to a low value. In SIFT, we assume that trust is expressed as continuous values between 0 and 1. The trust threshold value is therefore also a continuous value between 0 to 1.

4.1.3 Interaction Limit (lim)

Certain trust models specify with every trust value a trust expiration date that stipulates the time for which that trust value is considered "good". This is to ensure that data that is no longer considered valid is not included in the evaluation. Instead of using a validity period for trust values, some trust models limit the data used for trust evaluation by constraining it to a specific number of recent interactions. One example is Ebay which provides information about the interaction behavior of a user over the past week, month, and six months [3]. While this reputation information is certainly useful in gaining a better understanding of the future behaviors, it is susceptible to the case where a malicious peer may behave well for six months and then revert to fraudulent behavior. It is therefore important to model this aspect of trust models. In SIFT, we model this aspect of trust models through the use of the interaction limit parameter that denotes the number of recent interactions that only should be considered in the evaluation of trustworthiness.

4.1.4 Recent Interaction Weight (val)

Some trust models also adopt a time-based degradation of trust values. In such models, recently reported trust values are given more importance during trust computation than those previously reported. We model this in SIFT through the use of the recent interaction weight parameter. This parameter specifies the weight that should be assigned to the recent interactions. Correspondingly, a weight of value (1-val) is assigned to the old interactions while determining trustworthiness.

4.1.5 Hop count (hc)

For querying trust information, SIFT adopts a communication mechanism similar to that of a gnutella-based system. Specifically, a peer sends out a query requesting trust data to its neighbors who in turn forward the query to their neighbors and so on. The query flooding is limited by a hop count parameter that is decremented at every peer that receives the query. If the hop count value is greater than 1, the query is forwarded to each of the peer's neighbors. Upon receiving a query, a peer checks whether it has the data requested in the query. If so, the peer sends back a response message with the requested data back to the query originator. The hop count parameter thus decides the extent to which the trust query traverses in the system and therefore also indirectly influences the number of received responses. This in turn affects the trust decision-making process at the query originator. While a higher hop count means a larger number of entities are queried, it also leads to the possibility of query flooding that may lead to issues with scalability [16]. Thus, depending upon the nature and needs of the application, a designer must evaluate this trade-off and choose an appropriate hop count value.

4.1.6 Degree of trust in unknowns (deg)

Depending upon how critical peer interactions are, a trust model may decide to trust unknown peers to varying degrees. A very liberal trust model that wants to encourage the participation of peers, may decide to trust new unknown peers completely. On the other hand, a more conservative trust model may decide to

bestow little or no trust on such unknown peers. A trust model may also choose to just plainly report that it has no information about the unknown peer. “Degree of trust in unknowns” parameter helps model this aspect of trust models.

4.2 Application-specific Parameters

Similar to the trust parameters, we model an application through the use of application-specific parameters that help capture the essential features of the application. The application-specific parameters described below do not form an exhaustive list by any means. They have been so chosen because of their significant role in the choice of a particular trust model, and serve as a good starting point for our experiments.

4.2.1 Number of initial interactions (int)

This parameter denotes the number of interactions that peers have with each other in every simulation round before the trust querying and evaluation phases begin. Depending upon the nature of the application, peers may engage in varying number of interactions. The number of interactions peers engage in with each other impacts the amount of trust-related data that is available subsequently to make trust decisions. Thus, the number of interactions can play a significant role in determining the effectiveness of the trust model against malicious attacks.

4.2.2 Connection density (oe)

The connection density helps capture the overlay topology of the application. Peer-to-peer decentralized applications have been observed to have a power law network topology with most nodes sparsely connected and a few nodes that are densely connected [19]. Such a topology has also been commonly observed across various systems including the internet and biological and sociological networks [4]. Therefore, in order to be able to simulate real-world application environments to provide results that would match the real-world context, SIFT assumes a power law-based topology for all applications and incorporates an algorithm that constructs a power law-based topology using the peer connection density parameter. This parameter, represented by “oe”, denotes the minimum number of peers each new peer must be connected to when it first joins the system.

4.2.3 Number of peers (peers)

This parameter denotes the total number of peers in the system.

4.2.4 Number of malicious peers (mal)

This parameter denotes the number of peers in the system that will engage in malicious activity.

4.2.5 Malicious Probability (malProb)

This parameter denotes the probability with which a malicious peer will engage in fraudulent interactions. Typically, malicious peers in decentralized applications first build up a good reputation through successful interactions with other peers and then leverage their good reputation to successfully engage in malicious activity. How frequently a peer exhibits malicious behavior depends upon the nature of peers in the system. The malicious probability parameter helps capture this frequency with which malicious behavior is exhibited by peers. A higher value of the malicious probability indicates that peers in the system more frequently engage in fraudulent interactions and vice versa.

4.3 Simulation Parameters

In addition to the trust model and application parameters, SIFT also includes the following simulation parameters that help control the simulation setup in order to vary the conditions that are used to examine a trust model's behavior. In SIFT, each simulation round consists of a specified number of interactions between randomly chosen peers followed by one or more rounds of querying and assessing the trustworthiness of all peers in the system. A random set of peers are chosen for this querying and assessment in every simulation round. Further details of the SIFT simulation setup are provided in section 5.

4.3.1 Number of peers that assess (aPeers)

This parameter represents the number of peers that originate queries, aggregate the responses, and combine them along with their personal experiences to assess the trust level of all peers in the system. This evaluation averaged over all assessing peers is used to determine how well the trust model helps identify malicious peers.

4.3.2 Number of assessments (aCount)

This parameter denotes the number of rounds of querying and assessment that each assessing peer engages in each simulation round.

4.3.3 Type of attack (attack)

SIFT subjects each trust model to four different types of attacks. These four types of attacks do not form an exhaustive set, but are chosen to cover two critical but common threats to decentralized applications: misrepresentation and collusion. Misrepresentation is when a peer lies about the trustworthiness of other peers in the system. Collusion is when a group of peers collectively indulge in misrepresentation either to boost their own trustworthiness or to reduce the trustworthiness of others. Each of these attacks is modeled as a threat scenario in SIFT.

Type 0 - This attack is when a malicious peer misrepresents its trust in other peers including other malicious peers. Specifically, if the target peer is trustworthy, the malicious peer reports otherwise and vice versa.

Type 1 - In this case, a malicious peer misrepresents its trust in good peers but always reports complete trust in other malicious peers.

Type 2 - Type 2 attack is a type of a collusion attack where malicious peers collectively target a particular good peer and report a trust value of 0 for that target peer. However, unlike a type 1 attack, these malicious peers do not report complete trust in each other.

Type 3 - Type 3 attack is a combination of type 1 and type 2 attack. Specifically, malicious peers collectively misrepresent good peers as well as report complete trust in other malicious peers.

4.3.4 Number of peers being targeted (targets)

This parameter denotes the number of good peers that are targeted by colluding malicious peers in type 2 and type 3 attacks.

4.3.5 Number of simulations (simCount)

SIFT chooses malicious peers and assessing peers randomly at the start of each simulation. Further, in each simulation round peers pick other peers randomly for interactions. If a simulation were to be carried out only once, it would be hard to decide whether the simulation results obtained are indeed representative of typical behavior. Repeating simulations a number of times with the same set of trust, application and simulation settings increases confidence in the simulation results. The number of simulations parameter denotes the number of times a simulation is repeated for each set of trust, application and simulation parameters. The results obtained from all the simulations are averaged over the number of simulations to arrive at a final set of values.

4.4 Assessment Metrics

In order to quantify and compare the behavior of trust models, SIFT identifies the following metrics.

4.4.1 Malicious peers identified correctly (M)

This metric describes how well the trust model identifies malicious peers in the system. Specifically, it represents the number of malicious peers whose trust values are found to be below the specified trust threshold value thus allowing the trust model to correctly identify them. A metric related to this is the remaining number of malicious peers **(N)** that could not be identified by the trust model.

4.4.2 Peers incorrectly identified as malicious (F)

This metric denotes the number of false positives. Specifically, it represents the number of peers that are not actually malicious but whose trust value is below the specified trust threshold value. While it may be considered better to err on the side of caution and be suspicious about every peer, identifying a large number of peers incorrectly as malicious may result in a significant reduction in the number of interactions. A metric related to the number of false positives is the remaining number of good peers **(G)** in the system that are correctly identified as trustworthy by the trust model.

4.4.3 Number of peers considered unknown (U)

This metric denotes the number of peers whose nature remains unknown to the assessing peers. This could be because no information is found about these peers and the trust model decides to mark them as unknown instead of considering them as trustworthy or untrustworthy.

5. SIFT Design

5.1 Peer model

In SIFT, each peer communicates directly with other peers to exchange information. Every peer is assumed to have the necessary communication, information storage, and computing capabilities that enable it to be an independent decentralized entity. Each peer maintains a list of neighboring peers that it is directly connected to and through whom the peer communicates with other peers in the system. Also associated with each peer is an attribute that reflects whether the peer is malicious or not.

5.2 Communication mechanism

Communication between peers is asynchronous and event-based. Similar to other decentralized peer-to-peer applications, communication between peers occurs without the intervention of any central routing authority. The communication relies on a broadcasting mechanism with a static hop count to spread queries throughout the system. Specifically a peer broadcasts a query message with an initial specified hop count value to all its neighbors. The query message also contains a sender list to store the path through which the query is forwarded. Each neighboring peer, upon receiving the query message, decrements the hop count, appends the sender's name to the sender list, and forwards the message to all its neighbors except the sender. This exclusion helps prevent unnecessary forwarding of messages to peers that have already received the message. This process of rebroadcasting the query message is repeated as long as the hop count value is greater than zero.

Each peer along the path that receives the query message also checks to see if it has the information requested by the originator. If it does, the peer encapsulates the needed information within a response message and sends the response message back along the path the query was received using the sender list. A peer needs to know only its neighboring peers in order to communicate with peers in the system. Communication and information exchange between peers is strictly restricted to messages exchanged between peers. In other words, peers cannot use out-of-band communication to exchange information.

5.3 Simulation Algorithm

When the simulation first starts, each peer is created and randomly initialized with the relevant parameters that determine whether the peer is malicious or an assessing peer and/ or one of the peers targeted by malicious peers. Only peers that are not malicious are chosen as assessing peers. The network topology with the specified peer connection density is then constructed and each peer is given a list of peers whom it must connect to.

At the start of the simulation, peers interact with each other and build up some initial reputation history. Depending upon the value of the malicious probability, malicious peers will engage in some fraudulent interactions. If the malicious probability is 0, then the malicious peers will never engage in any fraudulent interaction. Similarly, if the malicious probability is 1, malicious peers will always engage in fraudulent interactions. The result of a fraudulent interaction is that the fraudulent peer is assigned a reputation value of 0 by the other peer. If this other peer does not exhibit fraudulent behavior, the fraudulent peer assigns a reputation value of 1 for that peer. If both peers engage in fraudulent actions, they both assign each a other a reputation value of 0. However, if an interaction is successful, both peers assign each other a reputation value of 1 for that interaction.

Next, the randomly selected assessing peers are asked to query and assess the trustworthiness of all peers in the system. Two kinds of messages are used in the system - QueryMessage and ResponseMessage. Each assessing peer broadcasts a QueryMessage to its neighbors. This message is spread in the system using the broadcast mechanism specified in Section 5.2. The QueryMessage encloses the name of the originator, the context for which the trust value is sought, and a sender list to which is appended the name of each peer that forwards the message. For the simulation results presented in this report, SIFT assumes that trust is uni-dimensional and all interactions are categorized under a single context.

Since the goal of the simulation is to help identify the nature of all peers in the system, each peer that receives the QueryMessage computes its trust of all the peers in the system. The subjective trust for every peer is computed as follows. First, the interaction limit (lim) specified by the trust model is used to break up the interactions into recent and old interactions. If the number of recent interactions is zero, it means

that no interactions should be considered. In that case, a trust value corresponding to the degree of trust in unknowns (deg) is assigned to that peer and returned to the originator. However, if this value indicates that the peer being evaluated should be considered unknown then no value for that peer is returned. If the number of old interactions is zero, then all interactions are considered recent. Then, as described below and illustrated in equation (2), the interactions are combined to compute a trust value. If the number of recent and old interactions is non-zero, then both recent and old interactions are combined separately using equation (2) to arrive at two values. These values are then weighted using the recent interaction weight (val) to compute the final trust value for that peer. Thus,

Equation 1:

*Trust Value Computed = (Trust value from recent interactions)*val + (Trust value from old interactions)*(1-val)*

Before combining data from interactions to compute a trust value, interactions to be included are further filtered based on whether they match the context specified in the query. As described earlier, in the current SIFT simulations, all interactions are categorized under a single context; thus, the resulting set of interactions after context matching remains unchanged. Values from the resulting set of interactions are then simply averaged to arrive at a single value.

Equation 2:

Trust Value from recent or old interactions = Summation of values from all considered recent or old interactions / number of recent or old interactions

Once the trust value is computed, then depending upon whether the peer is malicious or not, a peer may choose to lie or tell the truth about its perceptions. It should be noted that the malicious probability only dictates how often malicious peers will indulge in fraudulent interactions. SIFT currently does not include a parameter to model how often malicious peers lie about other peers; instead, it assumes that malicious peers will always misrepresent their trust in others. The primary reason for this assumption is to limit the further expansion of the huge design space that already exists currently due to the various trust and application parameters in SIFT. However, as will be discussed in section 7, we anticipate that such a parameter can be added to SIFT in the future.

If the computed trust value for a peer is 0 or 1, the malicious peer returns 1 or 0 respectively. This helps model applications where binary reputation is used. Otherwise, if the computed trust value falls below the trust threshold of the trust model, a malicious peer returns a random value greater than the trust threshold and vice versa. This result is then encapsulated within a ResponseMessage which is then sent back to the query originator. When the assessing peer that originally sent the QueryMessage receives a ResponseMessage intended for it, it stores that information for future use.

After each assessing peer has completed the process of querying and has aggregated the received responses, it evaluates the trustworthiness of all peers in the system. For this, it combines its personal experience of every peer with the reputation reported by the recommending peers using the trust weight (alpha) parameter of the trust model. Equation (3) describes how each peer's final trust value is determined.

Equation 3:

*Final Trust Value = (Personal Trust)*alpha + (Recommender Trust)*(1 - alpha)*

The Personal Trust is computed using equations (1) and (2) and follows the process described earlier. The Recommender Trust is computed using the following mechanism. The value reported by each recommending peer is first multiplied with the assessing peer's subjective personal trust in the recommending peer. Then these values are averaged over the number of recommending peers. Equation (4) illustrates how the Recommender Trust for each peer is computed.

Equation 4:

*Recommender Trust = (Summation over all recommenders (reported trust value * personal trust in recommender)) / number of recommenders*

Finally, at the end of the simulation, the final trust values from all assessing peers are combined to get an average final trust value. This value is then compared against the trust model's specified trust threshold value to determine how many and which peers are trustworthy or malicious. Peers with trust values above the trust threshold value are considered trustworthy, otherwise peers are considered malicious. Simulation results are expressed using the assessment metrics described earlier. These metrics are then used to analyze how well the trust model behaved in the face of threats.

5.4 SIFT Implementation

We implemented SIFT as a multi-threaded asynchronous application in Java. Each peer is modeled as a separate thread to preserve the autonomy of each peer. This also prevents any form of implicit synchronization between their interactions. No explicit synchronization is performed except when transmitting and receiving messages.

6. SIFT Evaluation

The primary goal of SIFT is to investigate the interplay between different trust and application settings towards providing guidance for the adoption of a suitable trust solution for a particular application. Hence, our evaluation of SIFT was directed at examining the degree of usefulness of the simulation results. Towards this objective, in this section, we pick some interesting simulation results and explain the resulting insights. A detailed discussion of all simulation results is not possible due to space constraints.

Table 2 lists the typical values of the different trust, application, and simulation parameters that were used in our experiments. Values different from those specified in these tables are explicitly specified as and when they occur. As explained earlier, in SIFT a collection of trust parameters, each set to some value represents a specific trust model. Thus, each point in the SIFT simulation results represents a unique trust model.

Simulation results are presented through bar charts that present the evaluation of a trust model in terms of the assessment metrics described in Section 4.4. 'M' denotes the number of malicious peers identified correctly by the trust model. Ideally, the values of 'mal' and 'M' should be equal. 'N' denotes the number of malicious peers that were not detected by the trust model and were instead considered trustworthy. 'F' denotes the number of false positives i.e. the number of peers that were not malicious but were considered so by the trust model. Ideally, values of both 'N' and 'F' should be as low as possible.

Table 2. Typical values used for SIFT simulations

Parameter	Symbol	Values
Trust		
Trust Weight	alpha	0.5
Trust Threshold	thr	0.5
Recent Interaction Weight	val	1
Interaction Limit	lim	100
Degree of trust in unknowns	deg	0
Hop Count	hc	3
Application		
Number of interactions	int	5000
Connection Density	oe	3
Number of Peers	peers	100
Malicious Peers	mal	50
Malicious Probability	malProb	1
Simulation		
Number of assessing peers	aPeers	20
Number of assessments	aCount	1
Type of attack	attack	0
Number of targeted peers	target	5
Number of simulations	simCount	50

'G' represents the number of good peers identified correctly and should ideally have a higher value as this would indicate that the trust model is also able to detect good peers correctly. Thus, in the charts below, higher values of G and lower values of F and N will indicate a better trust solution. Finally, 'U' represents the number of peers who are marked as unknown. The trustworthiness of such peers cannot be determined and the trust model chooses to mark them as unknowns instead of choosing to trust or distrust them. Ideally, the number of unknown peers should be as low as possible so peers can determine whether those peers can be trusted or not. Table 3 summarizes these assessment metrics and includes their symbols and ideal values for the set of simulations presented in this report. These symbols are used as legends in the bar charts that are discussed in the rest of this section.

Table 3. SIFT Assessment Metrics

Assessment Metric	Symbol	Ideal Value
Number of malicious peers in the system that were identified	M	50
Number of malicious peers in the system that were not identified	N	0
Number of good peers in the system that were identified as malicious	F	0
Number of good peers in the system that were correctly identified as good	G	49
Number of peers that were marked as unknown peers	U	0

6.1 Effect of 'number of interactions'

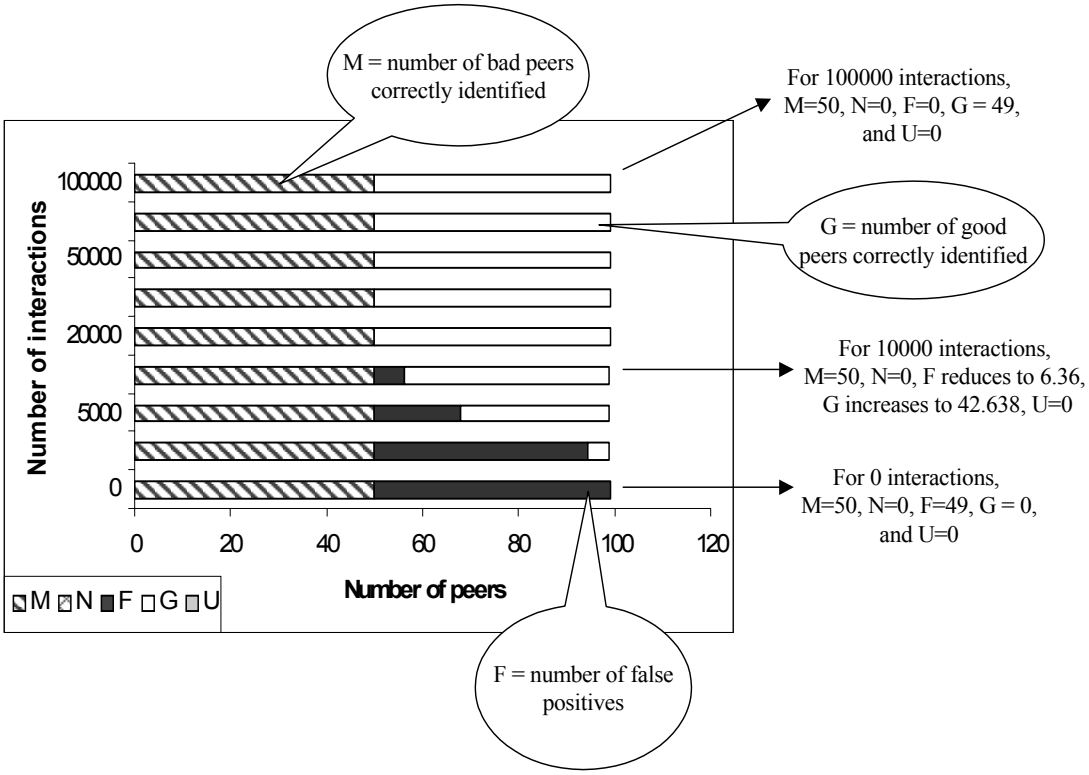

Fig 2. Varying the number of interactions with deg=0.1

Figure 2 shows the effect of increasing the number of peer interactions in a system where malicious peers always indulge in fraudulent interactions (malProb=1) and the trust model has a degree of trust in unknowns value equal to 0.1. Since unknown peers are given a low initial trust value of 0.1 and the threshold value is 0.5, it can be seen that at low interaction counts, a number of the peers are classified as malicious. Increasing the number of interactions decreases the number of false positives and increases the number of identified good peers while keeping the number of malicious peers identified correctly constant. This is because the simulation assumes a malicious peer will always act in a fraudulent fashion. Thus, increasing the number of interactions results in a greater number of failed interactions for a malicious peer (which is why malicious peers are identified correctly) and also results in a greater number of successful interactions for a good peer (which is why number of good peers identified correctly increases).

6.2 Effect of degree of trust in unknowns

Figure 2 showed the behavior of the trust model when the degree of trust in unknowns has a value equal to 0.1. Figures 3 to 7 show the effect as the degree of trust in unknowns is varied through the following values: 0, 0.5, 0.75, 1, and 2. Value 0 for the degree of trust in unknowns represents an extremely conservative trust model that completely mistrusts every unknown peer. Value 1 on the contrary represents a very liberal trust model that completely trusts every unknown peer. Value 2, for the purposes of our simulation studies, represents the case where the trust model reports no information about unknown peers and simply considers them as "unknowns". It should be pointed out that the simulation results presented in

this section are for an application with malicious probability value equal to 1. The effect of malicious probability on the degree of trust in unknowns will be discussed later in this section.

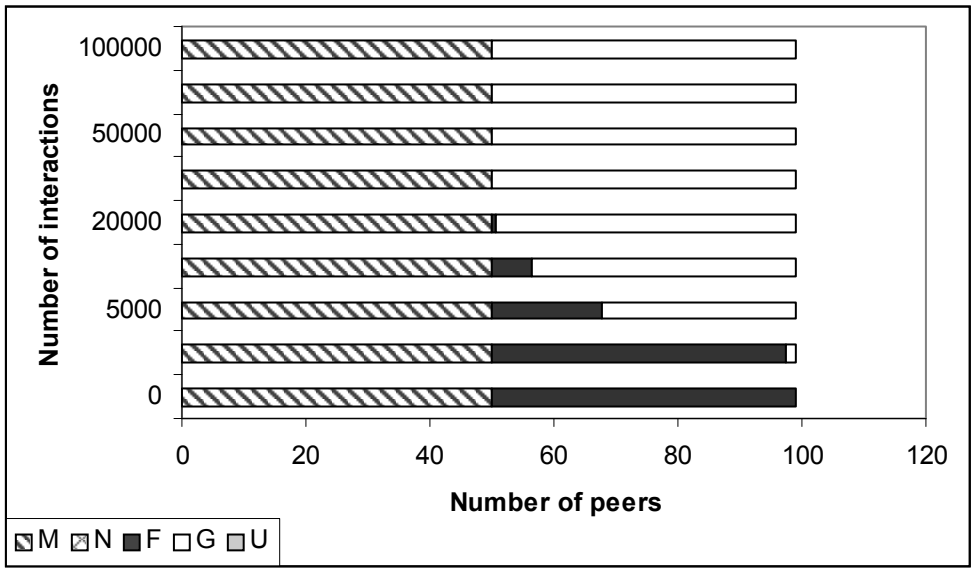

Fig 3. Varying the number of interactions for deg=0

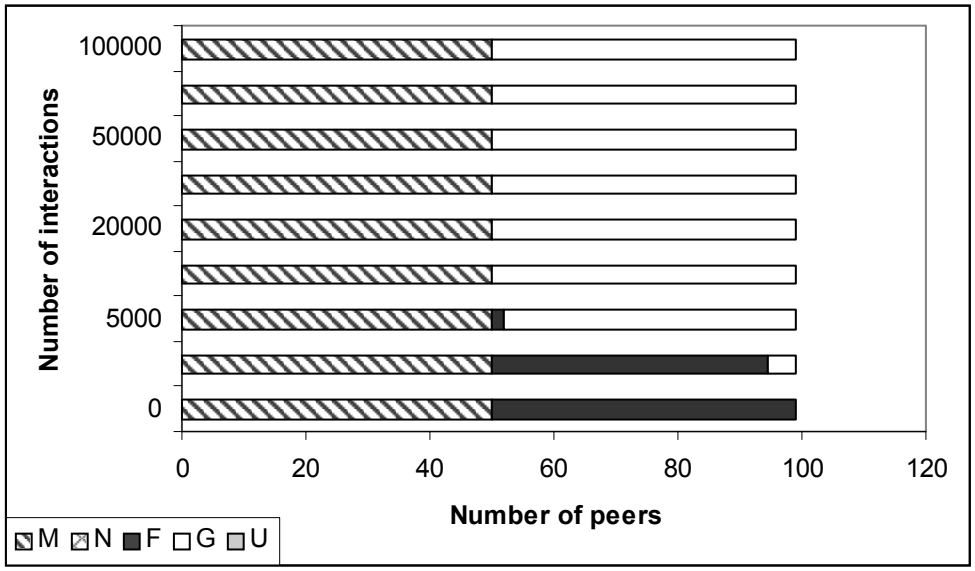

Fig 4. Varying the number of interactions for deg=0.5

Figures 3 and 4 reveal that as the number of interactions increases, the number of good peers identified increases and the number of false positives decreases. Figures 5 and 6 show the behavior of a liberal trust

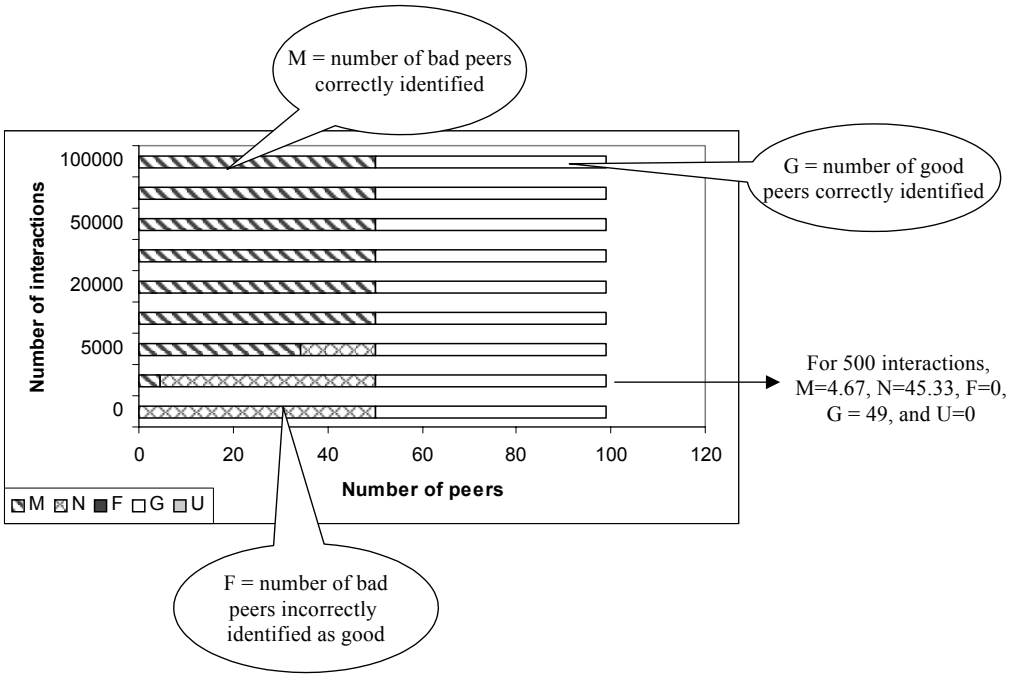

Fig 5. Varying the number of interactions for deg=0.75

model. It can be seen that when the number of interactions is low, a large number of malicious peers remain unidentified and in fact are erroneously treated as trustworthy. However, as the number of interactions increases, the number of malicious peers identified starts increasing until all such malicious peers have been correctly discovered.

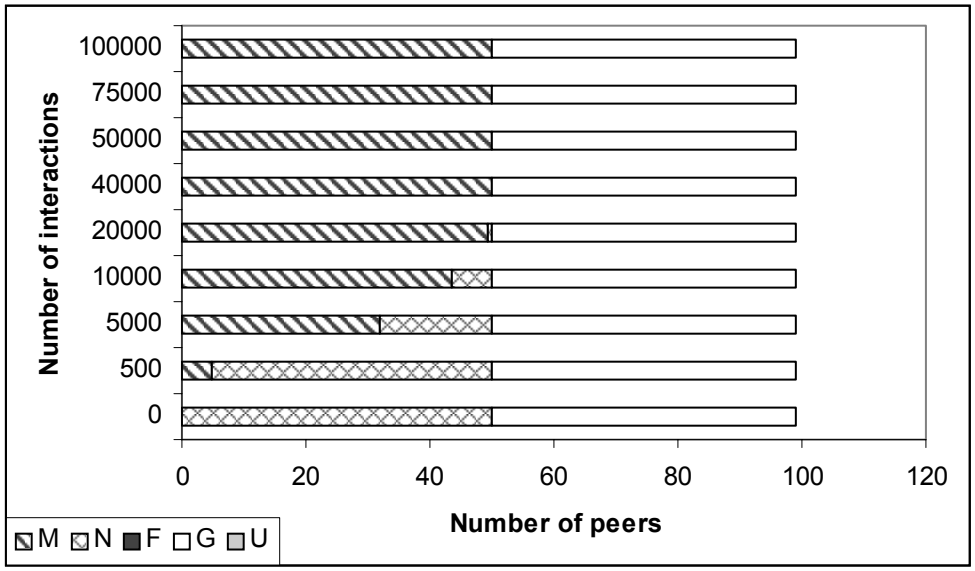

Fig 6. Varying the number of interactions for deg=1

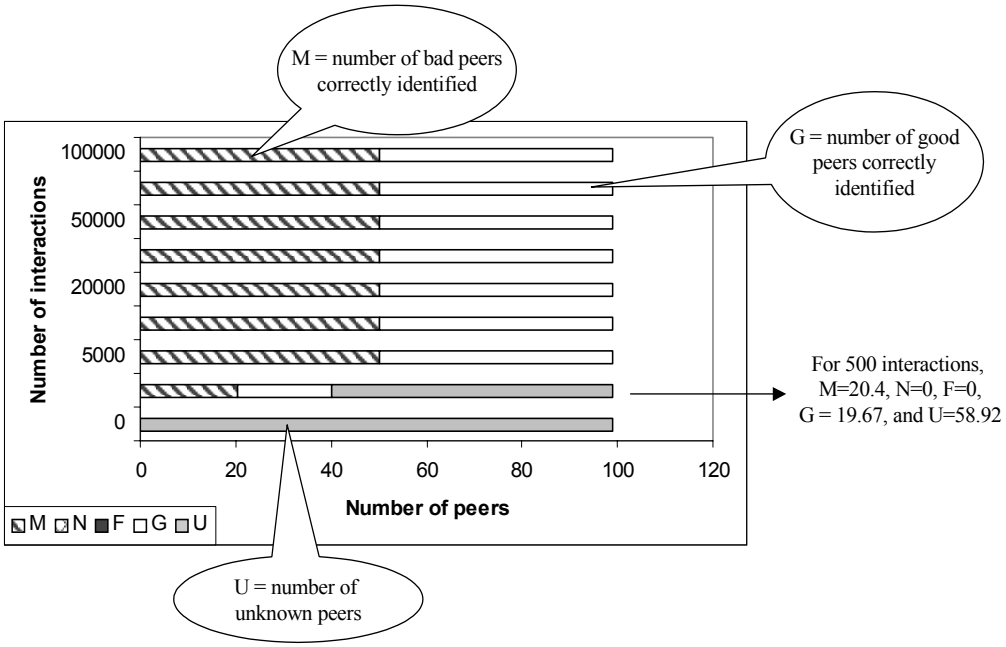

Fig 7. Varying the number of interactions for deg=2

Figure 7 shows the behavior of a trust model that reports unknown peers as "unknown" rather than trustworthy or untrustworthy. When the number of interactions is low, a large number of peers remain unknown to each other resulting in a significant number of unknown peers. However, as the number of interactions is increased, more peers are known to each other resulting in the increasingly correct identification of malicious and good peers.

Figure 8 presents the effect of varying the "degree of trust in unknowns" with 5000 interactions and shows that for an application with 5000 interactions, it may be better to choose a trust model that has a "degree of trust in unknowns" value of either 0.5 or 2. However, if the number of interactions is large, then, as

illustrated in Figures 2 to 7, it does not matter how conservative or liberal the trust model is because all peers are correctly identified for large number of interactions.

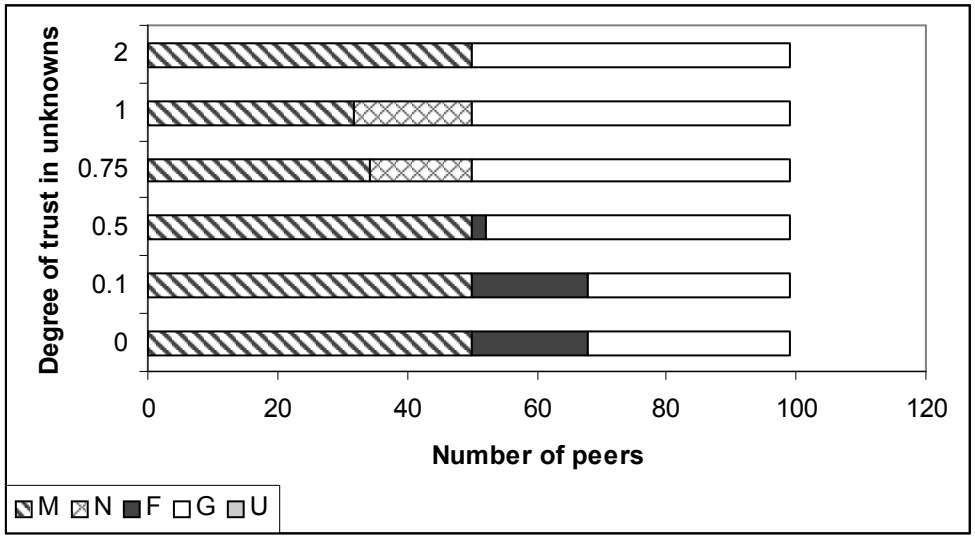

Fig 8. Varying the degree of trust in unknowns

6.2.1 Effect of 'trust threshold'

This section discusses the interplay between the trust threshold and the number of interactions and degree of trust in unknowns for a system with malicious probability value 1.

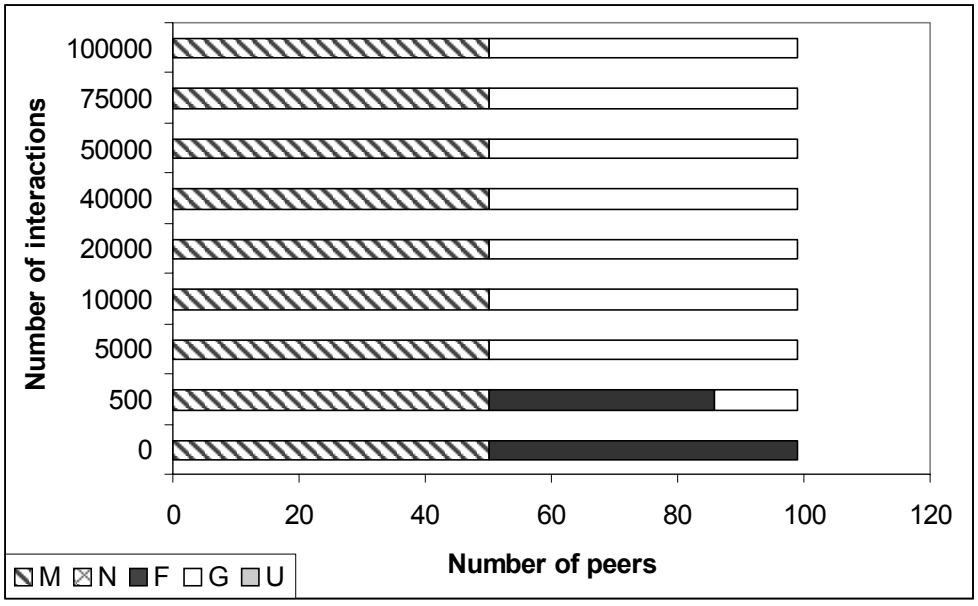

Fig 9. Varying the number of interactions with thr=0.1

Figure 9 shows that with threshold value equal to 0.1, the number of false positives is extremely high for lower interaction count. This can be explained by the fact that at lower interaction counts, peers don't have enough information about each other and since deg value is 0, the trust model treats unknown peers as malicious. However, as the number of interactions increases, more trust information is available and peers

are correctly and quickly identified. The low threshold value of 0.1 implies that as long as a peer has a trust value above 0.1, it is considered trustworthy. However, since malicious peers are assumed to always act maliciously, it can be seen that all peers are quickly and correctly identified in spite of the low threshold value. Figure 10 illustrates how it requires a greater number of interactions to correctly identify peers as the threshold value increases. Figure 11 presents the case where the threshold value is 1. Clearly, in such a case where the trust model is the most conservative, all peers will be considered malicious.

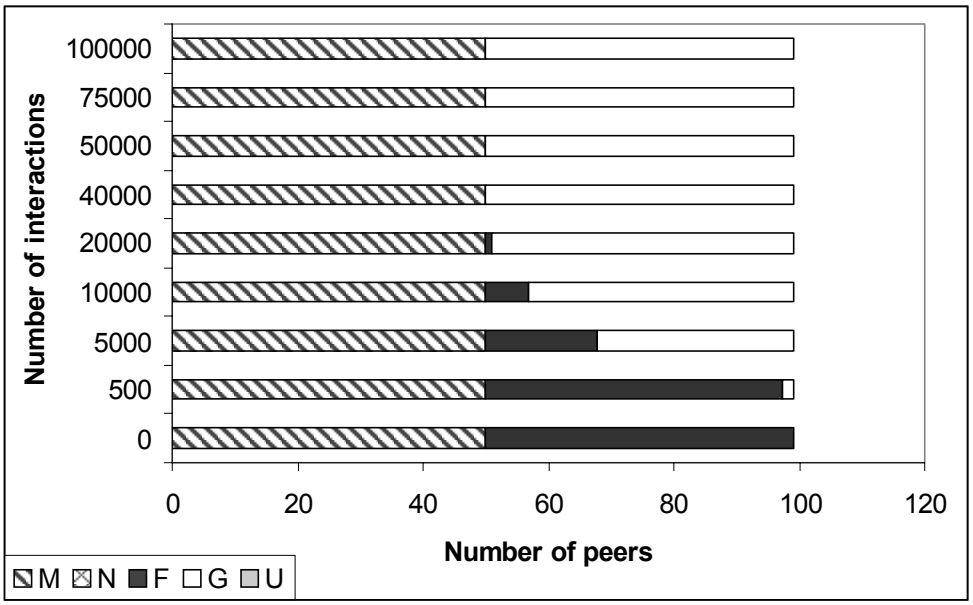

Fig 10. Varying the number of interactions with thr=0.5

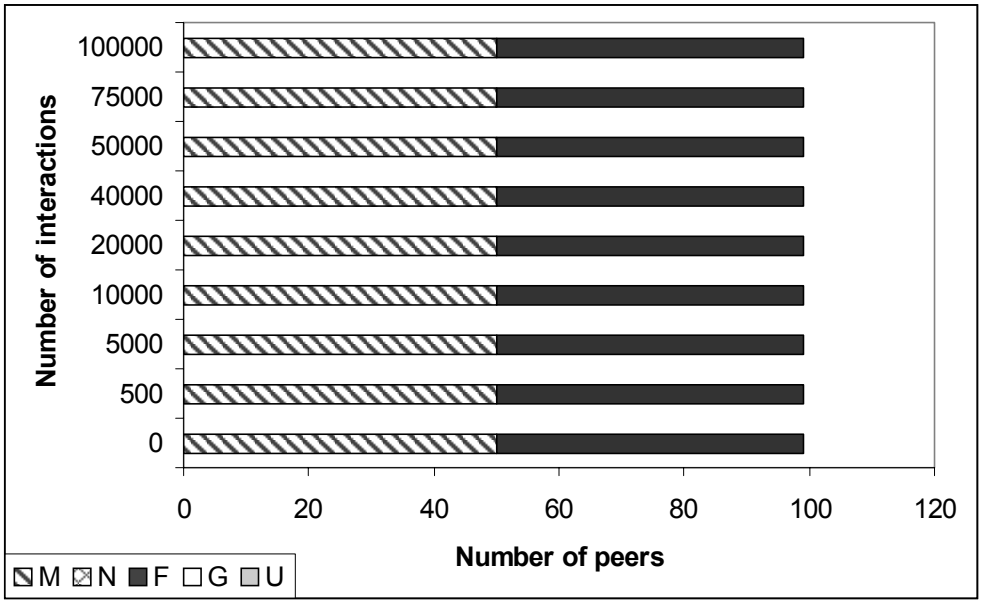

Fig 11. Varying the number of interactions with thr=1

Figure 12 summarizes the effect of varying threshold for a system with 5000 peer interactions. It reveals that for a system subject to the above assumptions, if an application designer were to choose a trust model

that does not trust unknown peers (deg=0), an optimal behavior may be obtained if the trust threshold value is set to a value below 0.25.

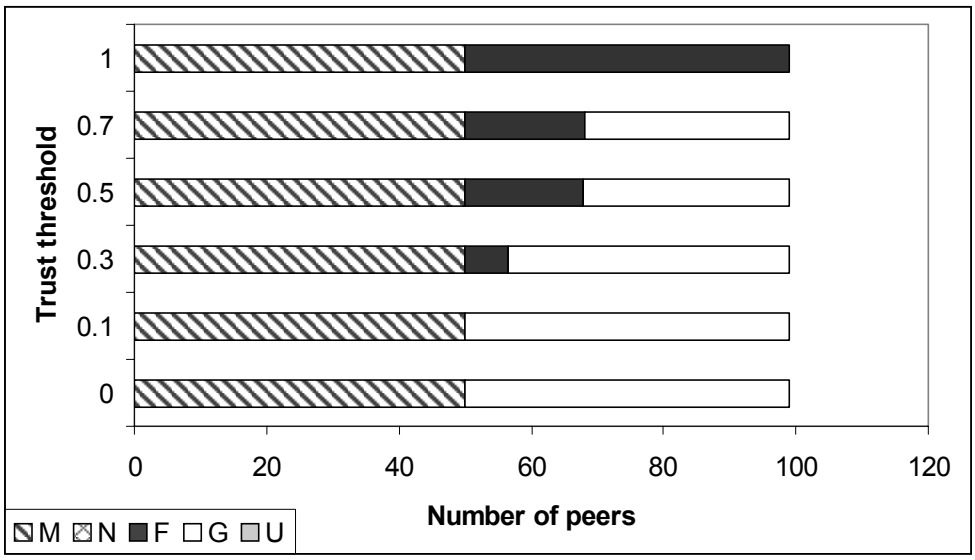

Fig 12. Varying trust threshold for deg=0

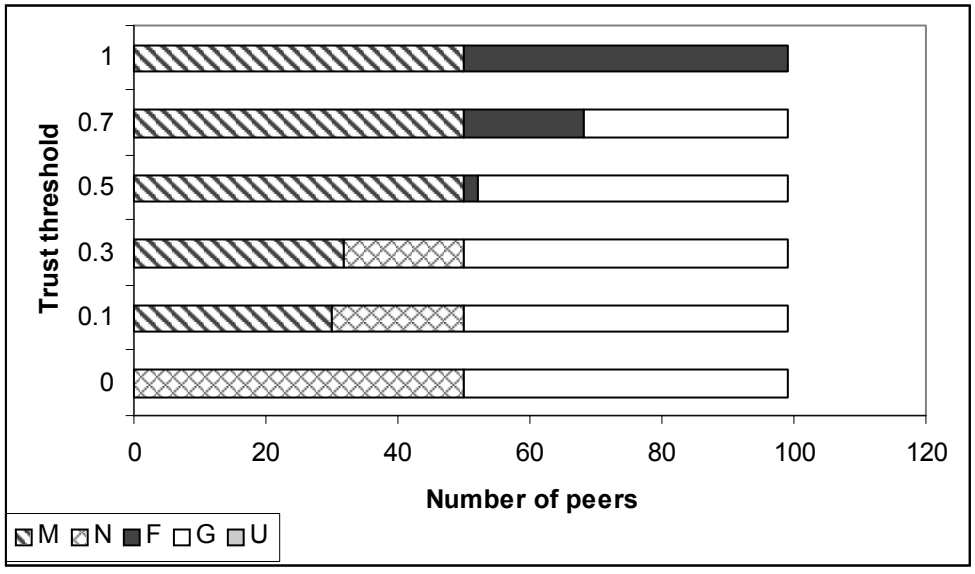

Fig 13. Varying trust threshold for deg=0.5

Figure 13 shows that if the value of the degree of trust in unknowns were to be increased to 0.5, the ideal trust threshold value for the same system is around 0.5. For a threshold value lower than 0.5, due to the higher degree of trust in unknowns, a larger number of malicious peers may be considered trustworthy.

Figures 13 and 14 show that as the degree of trust in unknowns is increased, the trust threshold must also be increased correspondingly to correctly identify the nature of peers.

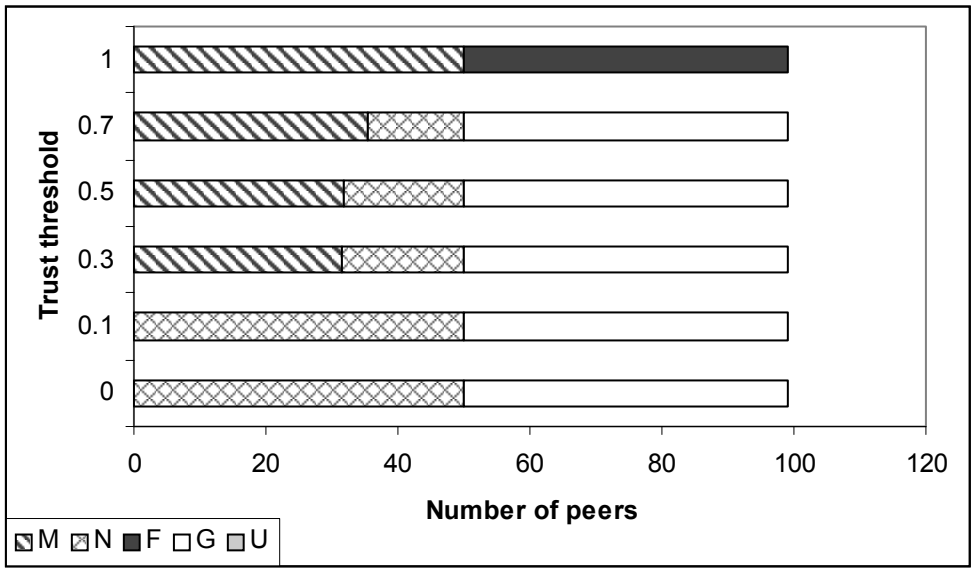

Fig 14. Varying trust threshold for deg=1

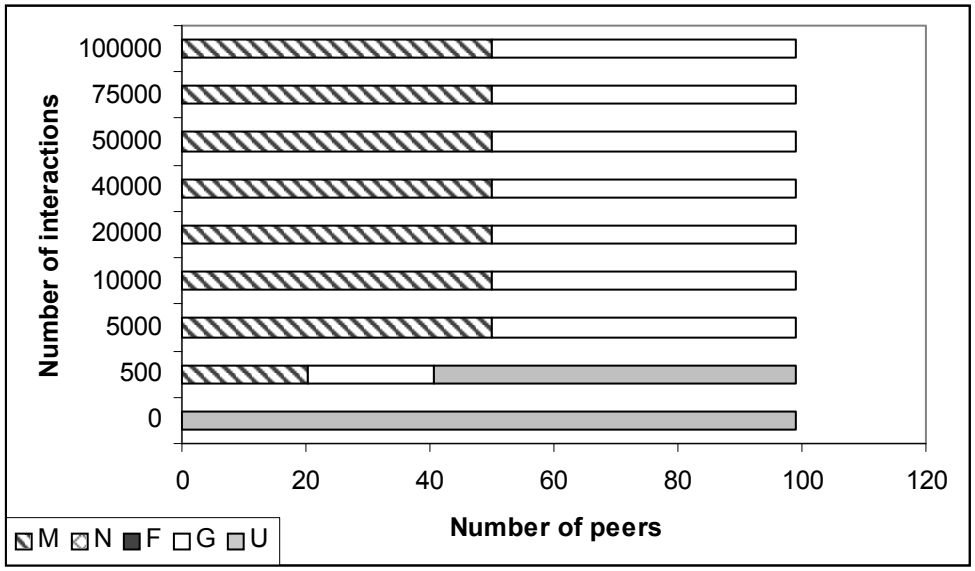

Fig 15. Varying the number of interactions for thr=0 and deg=2

Finally, Figure 15 illustrates the case when the trust threshold is 0 and peers simply report unknown peers as "unknown" (deg=2). The chart shows that for low interaction count, a larger number of peers remain unknown to each other. However, this number decreases with increasing interaction count. This is further aided by the liberal nature of the trust model (thr=0) implying that peers are completely trusted as long as they have a non-zero trust value.

6.2.2 Effect of 'trust weight'

Trust weight dictates the proportion in which personal perception and reported information are combined to compute trustworthiness. All the results discussed in this section assume a malicious probability value of 1. The relationship between the malicious probability and trust weight is discussed later in section 7.7.8.

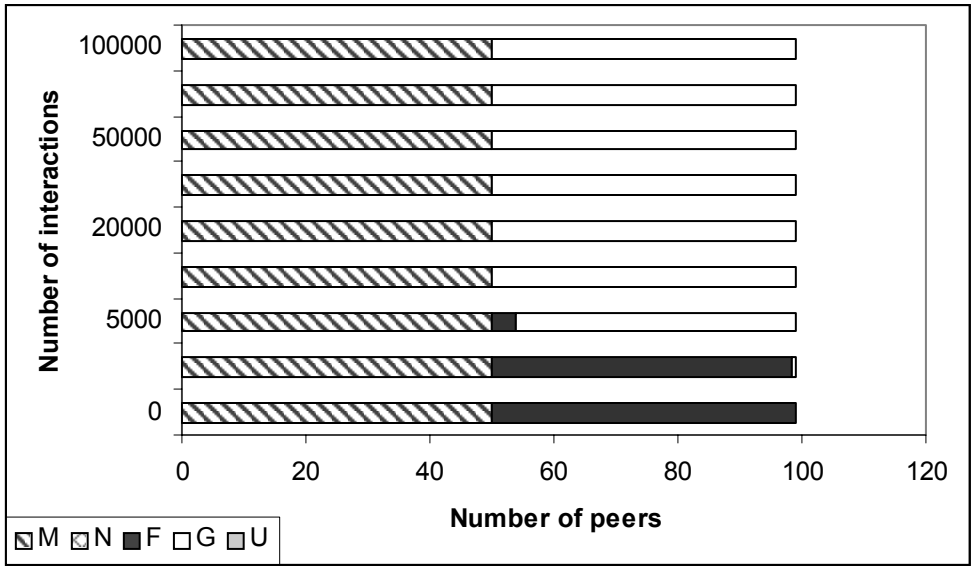

Fig 16. Varying the number of interactions for alpha=0

Figure 16 shows the behavior of a trust model with alpha=0 and deg=0 as the number of interactions is varied. When the number of interactions is small, peers do not have complete information about each other and report low values of trust as dictated by the degree of trust in unknowns (deg=0). Since trust is strictly computed on the basis of reported values for alpha=0, most peers are treated as malicious. However, as the number of interactions increases, trust data maintained by each peer rapidly increases and peers can be correctly identified.

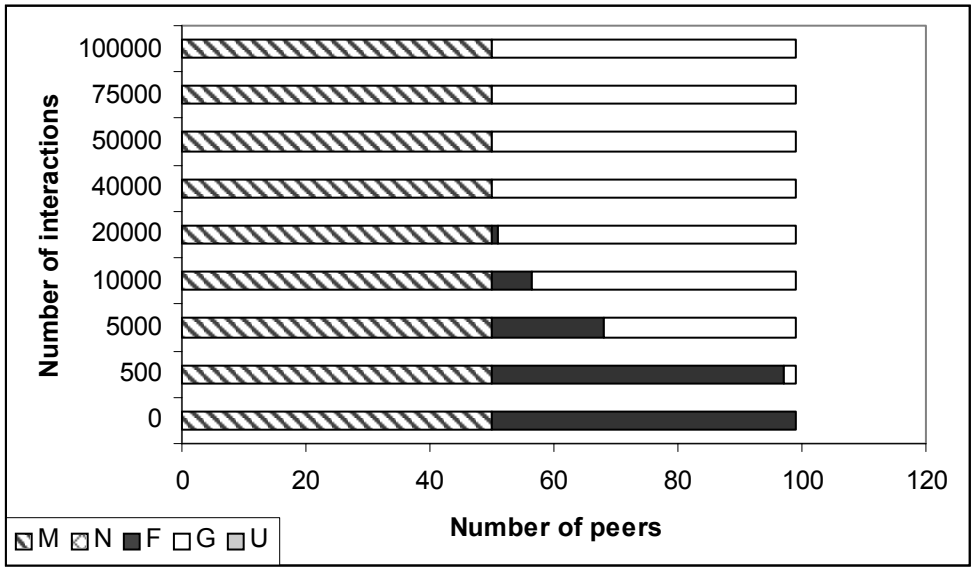

Fig 17. Varying the number of interactions for alpha=0.5

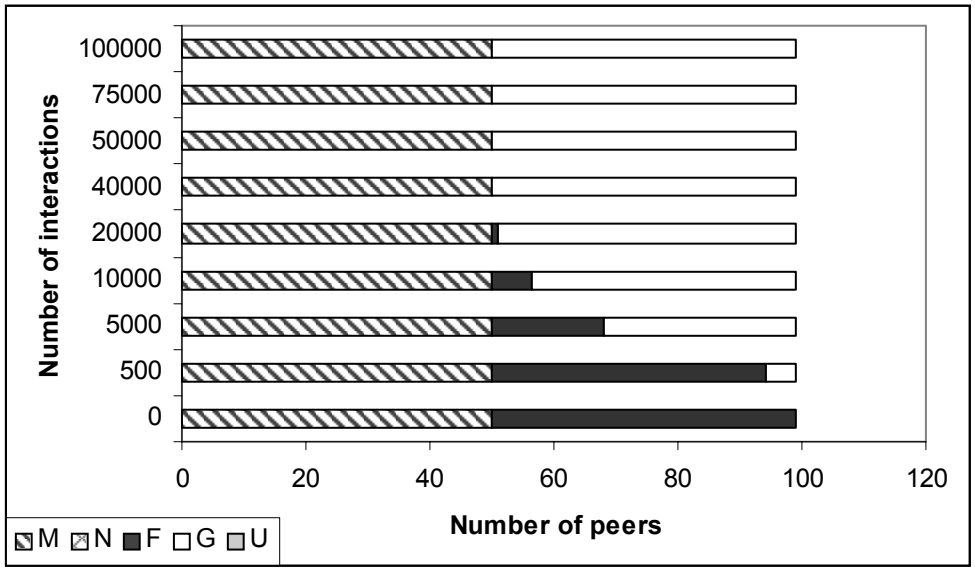

Fig 18. Varying number of interactions for alpha=1

Figures 17 and 18 illustrate the effect of varying interaction count as a trust model increasingly relies on its personal perception to compute trustworthiness. It can be seen that the behavior of the trust model actually worsens as the value of alpha is increased. This is because low interaction counts imply that each peer has little trust information about other peers. A high value of alpha in such a case means that the trust determination will be made mostly on the basis of the little personal information each assessing peer has. Thus the trust determination will not be accurate. However, as the number of interactions increases, the trust information available increases and thus, personal trust information maintained by each peer becomes increasingly sufficient to compute trustworthiness correctly.

Figure 19 summarizes the effect of varying alpha with degree of trust in unknowns equal to 0 for an application with 5000 peer interactions. This chart reveals that for lower interaction counts, it may be better for the conservative (deg=0) trust model to rely on the trust information supplied by other peers in the system.

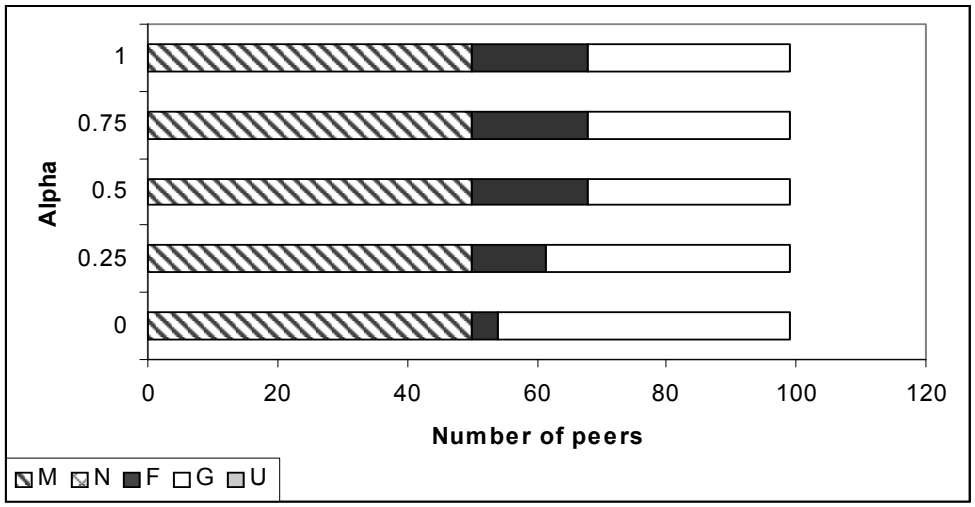

Fig 19. Varying alpha for deg=0

Figures 20 and 21 similarly illustrate the effect of varying alpha with higher degree of trust in unknowns for a system with 5000 interactions. It can be seen that in such a system, peers still lack trust data about some peers. Increasing the degree of trust in unknowns results in the trust model trusting a greater number of unknown peers. Thus, it can be observed in Figure 20 that in spite of relying increasingly on personal perception (increasing alpha), the number of false positives decreases considerably. However, at alpha=1, the number of false positives increases because when alpha=1, reported trust information is not used in trust computation. Instead trust determination is made strictly on the basis of personal perception which at 5000 interaction count is not enough to make correct trust determination.

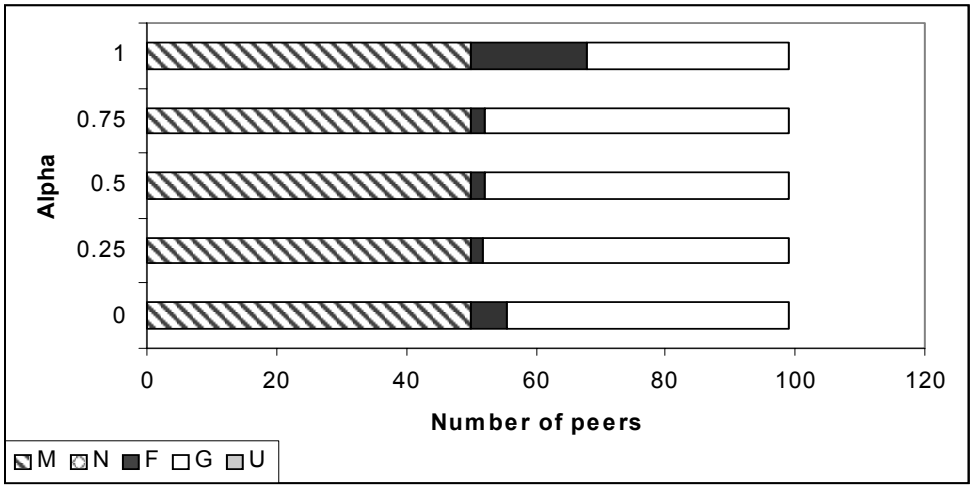

Fig 20. Varying alpha for deg=0.5

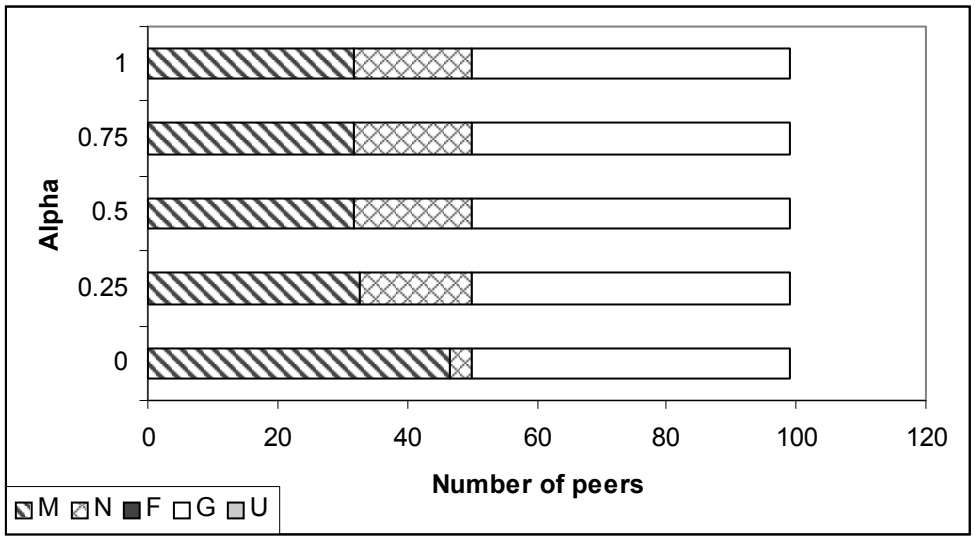

Fig 21. Varying alpha for deg=1

Similarly, Figure 21 shows that when the degree of trust in unknowns has a value equal to 1, the trust model behavior worsens with increasing alpha. In fact, since the trust model now trusts all unknown peers completely and moreover relies increasingly on its own limited personal perception, even malicious peers are considered trustworthy.

Finally, Figure 22 shows the effect of increasing interaction count with trust weight value equal to 0.25, degree of trust in unknowns equal to 2, and malicious probability value equal to 1. It can be seen that the

number of unknown peers is high for small number of interactions and reduces as more trust information is available with increasing number of interactions. Simulations results have also shown that for a system with 5000 or more interactions and with malicious probability 1, there is no visible effect of varying the trust weight when the degree of trust in unknowns is 2.

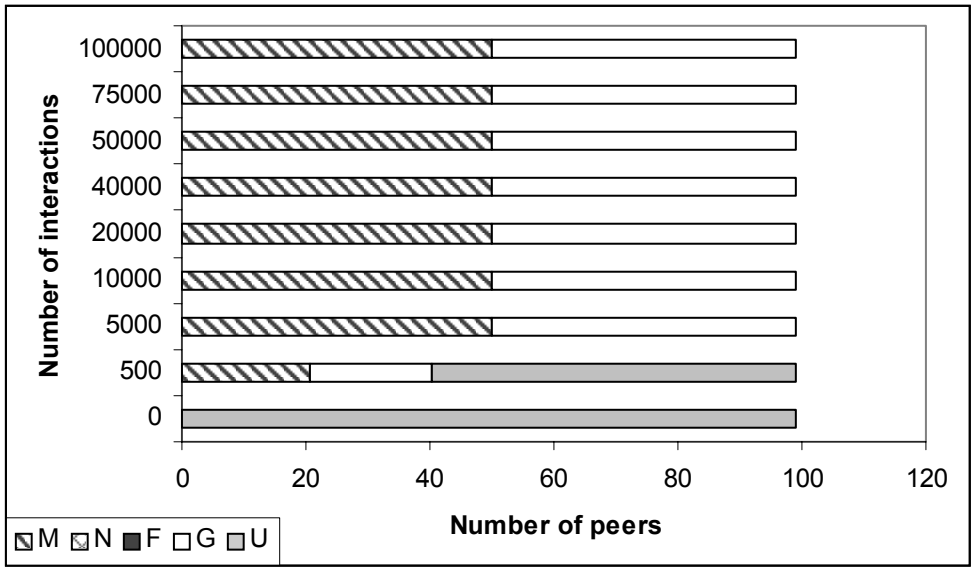

Fig 22. Varying the number of interactions for alpha=0.25 and deg=2

6.2.3 Effect of 'interaction limit'

This section discusses the effect of interaction limit. Every application setting discussed here assumes that malicious peers always indulge in fraudulent interactions (malProb=1).

Figure 23 shows the effect of increasing the number of interactions for a trust model with interaction limit 0.2%, weight of recent interactions equal to 1, degree of trust in unknowns equal to 0, and malicious

probability value equal to 1. It can be seen that it takes a large number of interactions for this trust model setting to correctly identify the malicious and good peers.

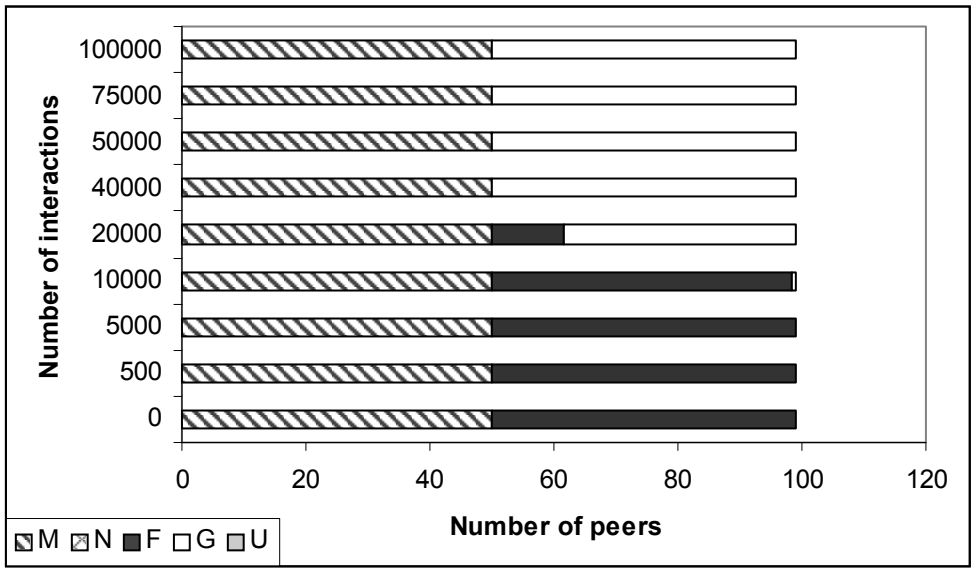

Fig 23. Varying the number of interactions with lim=0.2, val=1, deg=0

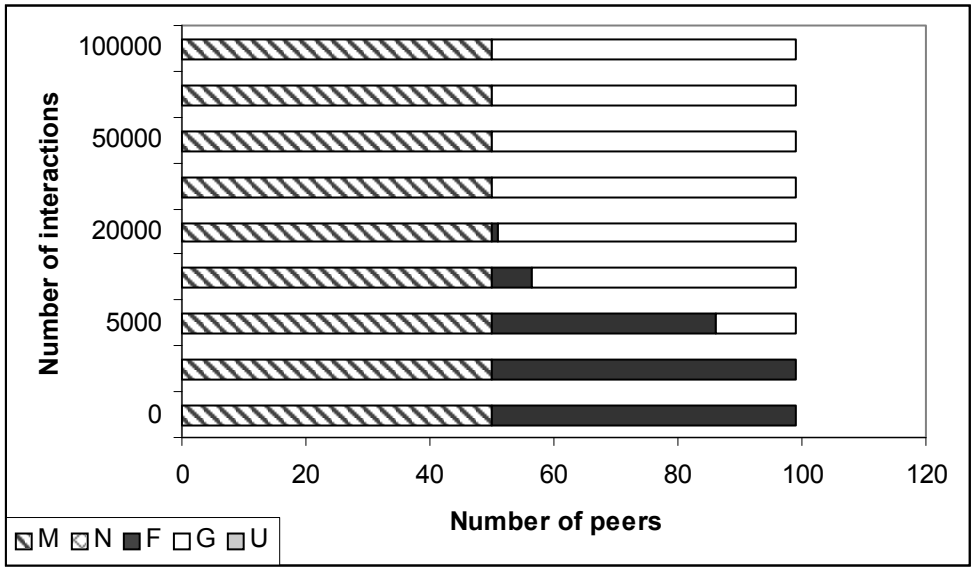

Fig 24. Varying the number of interactions with lim=1, val=1, deg=0

However, as shown in Figure 24, when the interaction limit is increased to 1%, the same result can be observed at a lower interaction count value. This can also be seen in Figure 25 which shows the effect of increasing the interaction limit for a system with 5000 initial peer interactions. Since the interaction limit

restricts the interaction history that is used in trust computation, it is seen that as more interaction data is used, the trust determination becomes more accurate and after a certain threshold achieves a steady state.

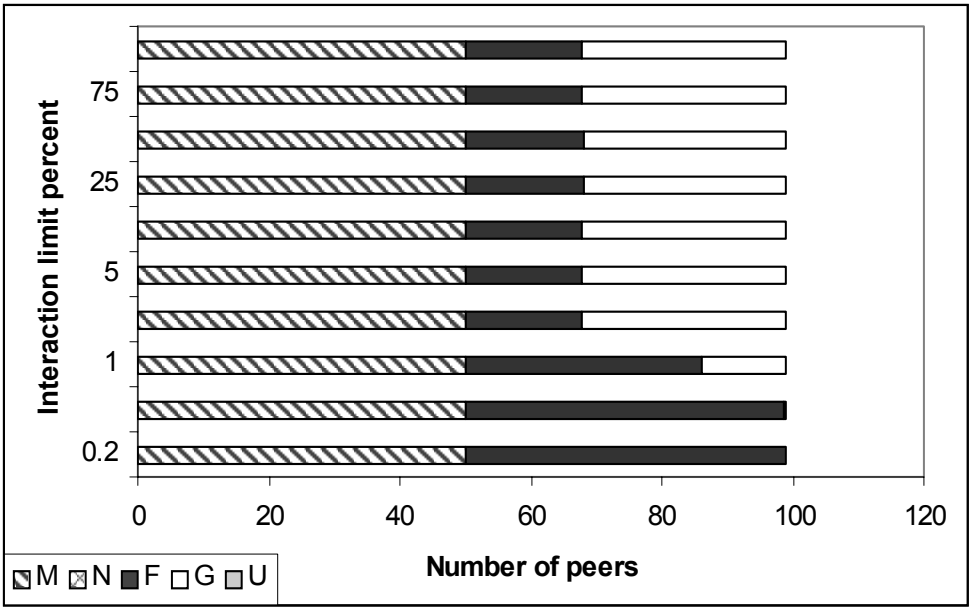

Fig 25. Varying interaction limit with val=1, deg=0

Figures 26 to 28 show the interplay of interaction limit with the degree of trust in unknowns. Comparing Figure 26 with 25 shows that as the degree of trust in unknowns is increased, the trust model behavior improves at higher interaction limits. For a trust model that completely trusts unknown peers (deg=1), it can be seen in Figure 27 that limiting the interaction history used in trust determination results in incorrectly identifying malicious peers as good peers. However, as the interaction limit increases, a greater number of malicious peers are identified correctly.

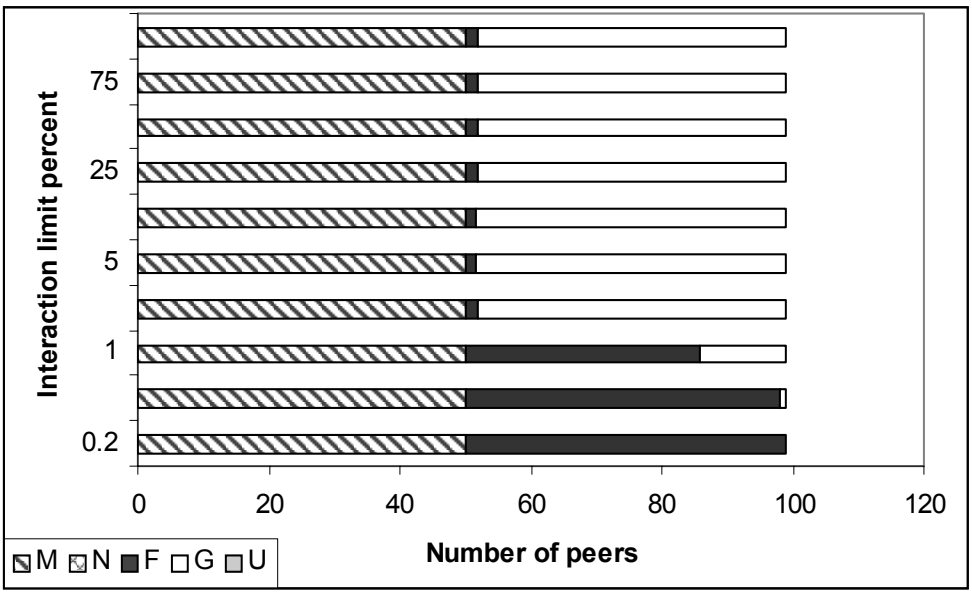

Fig 26. Varying interaction limit for val=1, deg=0.5

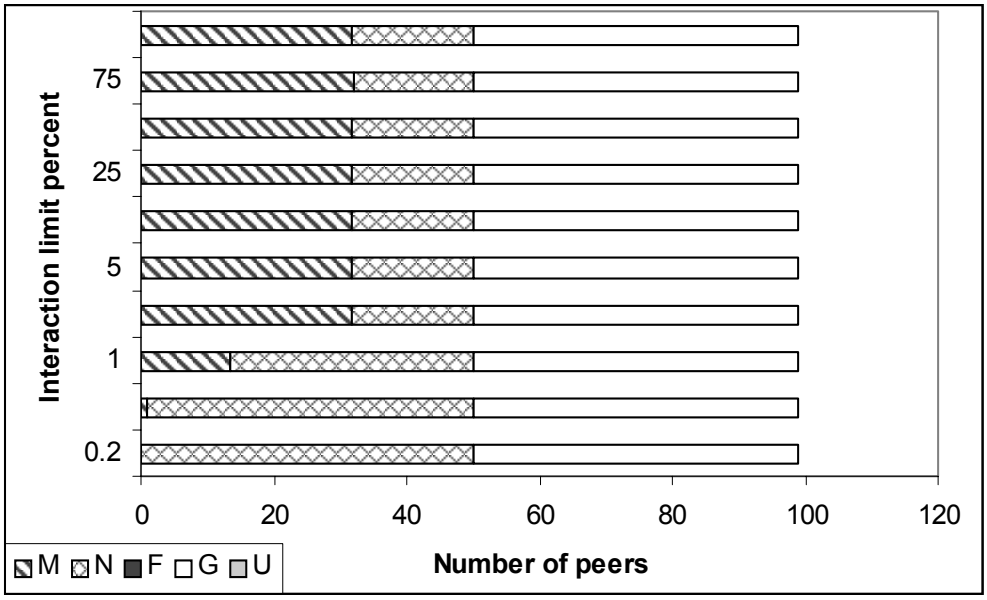

Fig 27. Varying interaction limit for val=1, deg=1

Figure 28 depicts the case when the degree of trust in unknowns is 2 and shows that as the interaction limit is increased, a greater number of peers are correctly identified.

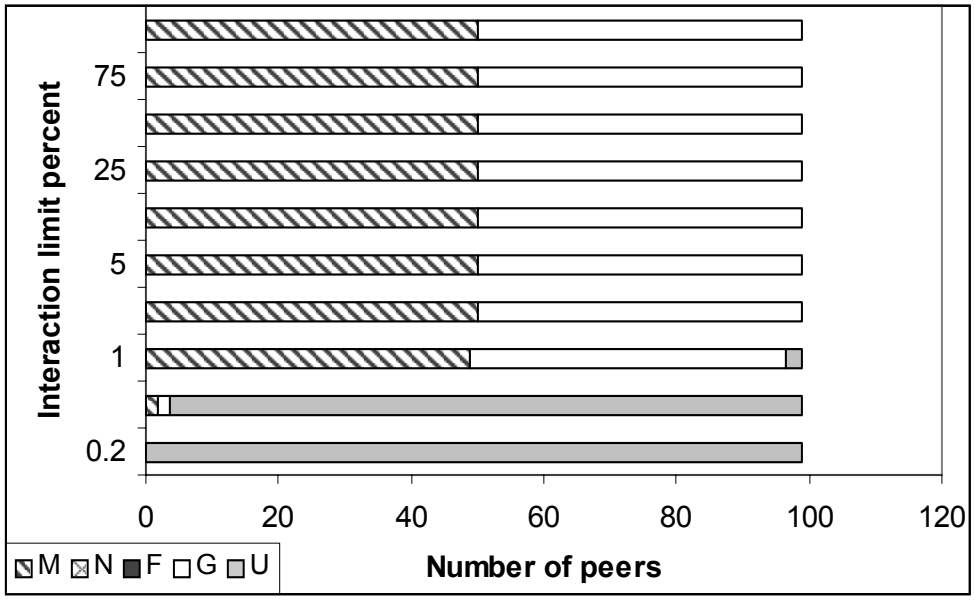

Fig 28. Varying interaction limit for val=1, deg=2

6.2.4 Effect of 'hop count'

This section discusses the effect of hop count on the behavior of a trust model for applications that have a malicious probability value equal to 1. Figures 29 and 30 illustrate the effect of varying the number of interactions for increasing values of hop count with each peer connected to at least 1 other peer in the system (represented by out edges) and degree of trust in unknowns equal to 0. In both figures, it can be seen that the number of false positives is high for low interaction counts and decreases as the number of interactions increases. It can also be seen that as the hop count value is increased, the trust model behavior improves. This increase is because as the hop count is increased, more number of peers in the system are queried for trust information. This results in more received responses and a possibly more accurate trust evaluation. Further, this increase is more significant at lower interaction counts when every peer has less

trust information than at higher interaction counts. This explains why with increase in hop count values, there is only a marginal improvement in the trust model behavior for a system with 5000 interactions.

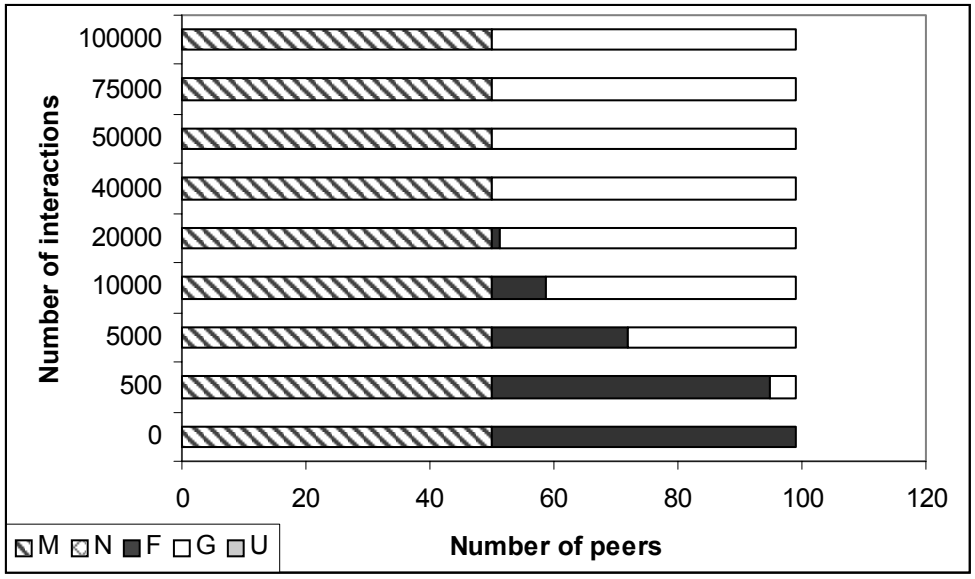

Fig 29. Varying the number of interactions for hc=1, oe=1, deg=0

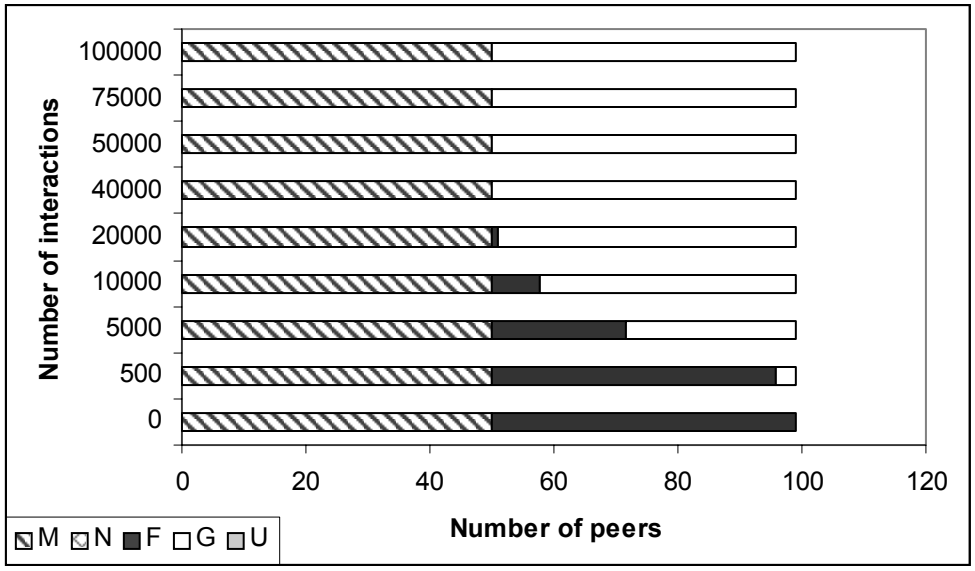

Fig 30. Varying the number of interactions with oe=1, hc=2, deg=0

Figures 31 to 34 show the effect of varying hop count for a system with 5000 peer interactions as the degree of trust in unknowns is increased. Comparing Figure 32 with Figure 31 shows that as the degree of

trust in unknowns is increased, the behavior of the trust model also becomes better. In fact, there is a marked reduction in the number of false positives as the hop count value is increased.

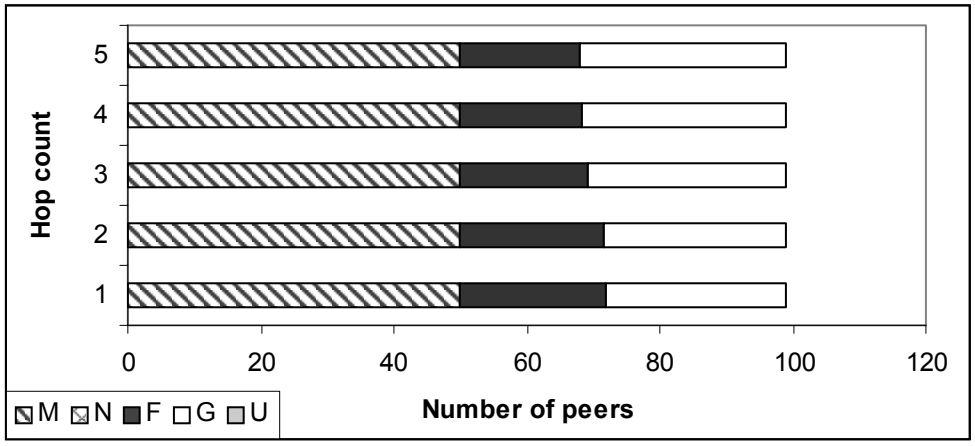

Fig 31. Varying hop count for oe=1, deg=0

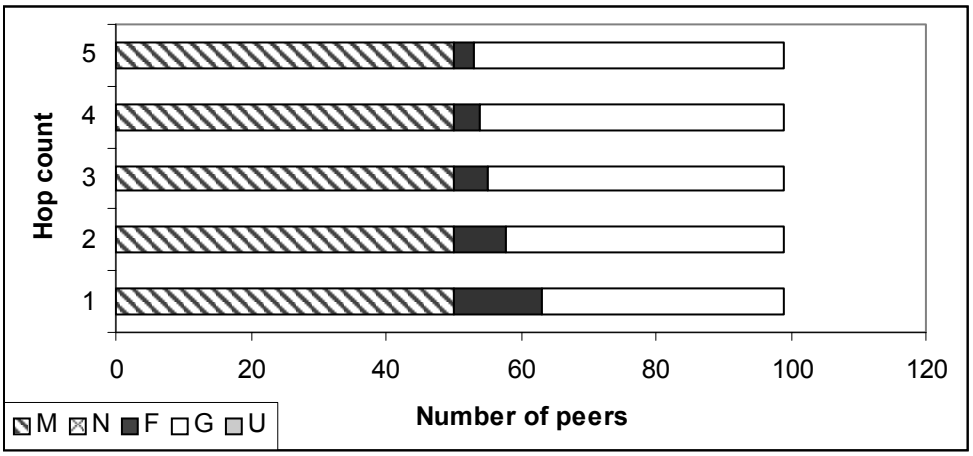

Fig 32. Varying hop count for oe=1, deg=0.5

However, when the degree of trust in unknowns is 1 (see Figure 33), the behavior of the trust model appears to slightly worsen with increasing hop count. This is because since more peers are being queried, there are more responses that tag unknown peers as trustworthy.

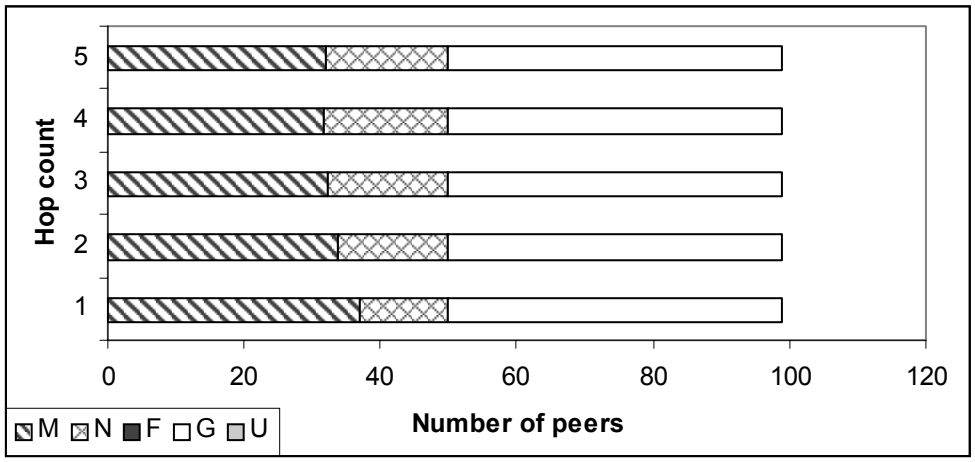

Fig 33. Varying hop count for oe=1, deg=1

Figure 34 shows the effect of varying hop count when the degree of trust in unknowns is 2 for an application with 5000 peer interactions. It can be seen that at low hop counts, since there is not enough trust data, the number of "unknown" peers is high. However, as the hop count is increased, more peers are queried leading to more received responses which ultimately results in a more accurate identification of the nature of peers.

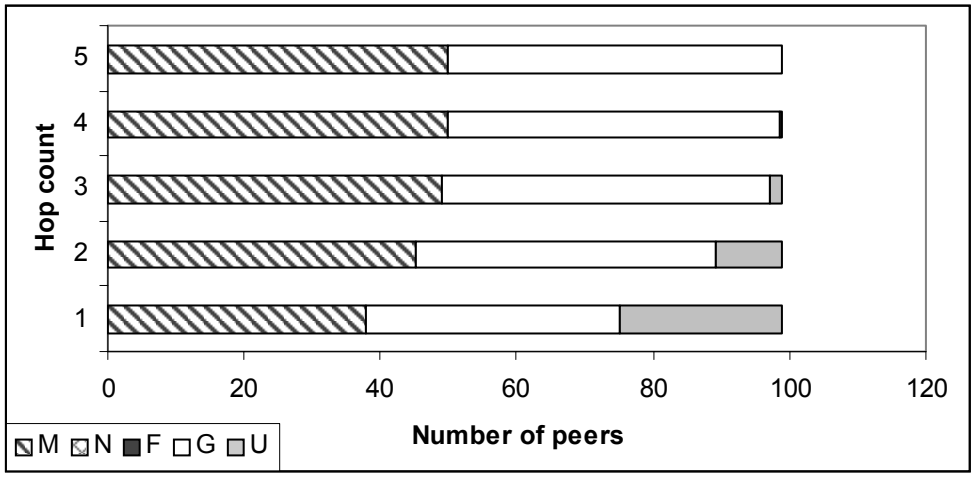

Fig 34. Varying hop count for oe=1, deg=2

6.2.5 Effect of 'connection density'

This section discusses the effect of varying the connection density of applications with malicious probability value 1. Figures 30 and 35 show the interplay of the number of interactions and the connection density for a fixed hop count value of 2 and degree of trust in unknowns equal to 0. It can be seen in both figures that at low interaction counts, there is a large number of false positives, and as the number of interactions increases, the number of false positives reduces.

Figure 36 shows the effect of varying the number of out edges with a fixed hop count value equal to 2, degree of trust in unknowns equal to 0, and 5000 peer interactions. It can be seen that as the number of out edges increases, more peers are connected to each other and this increases the amount of trust information

that is reported. Consequently, the number of false positives decreases and the number of good peers identified correctly increases.

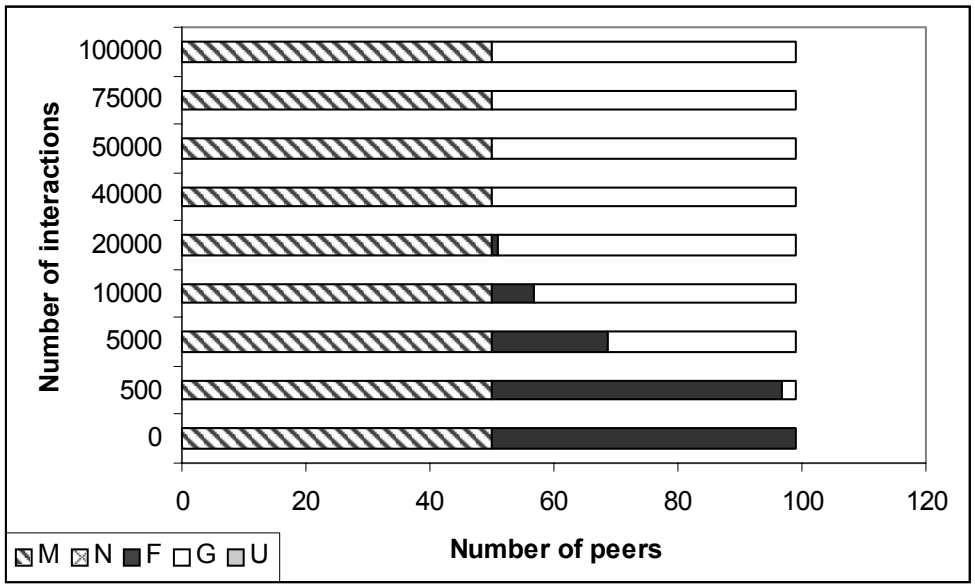

Fig 35. Varying the number of interactions with oe=2, hc=2, deg=0

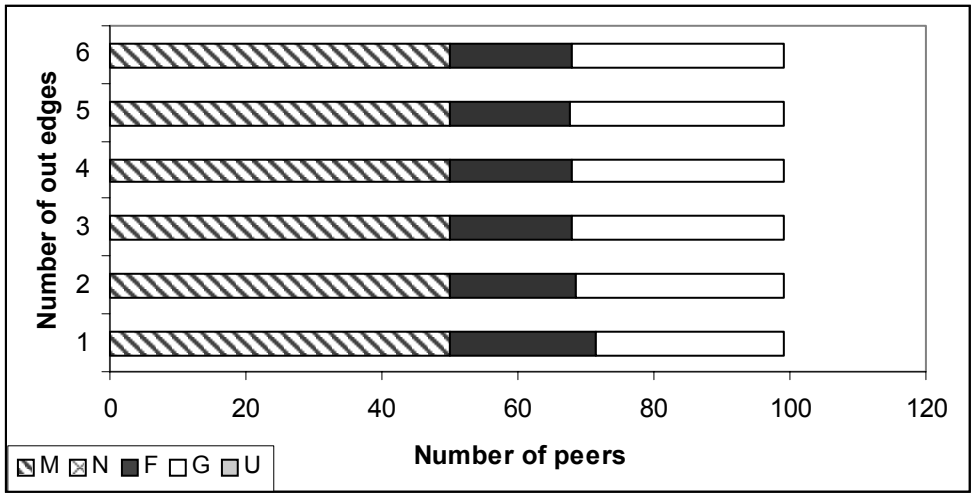

Fig 36. Varying the number of out edges for hc=2, deg=0

Figure 37 shows that as the degree of trust in unknowns is increased, the behavior of the trust model becomes increasingly better for similar application settings. However, as Figure 38 shows, when the degree of trust in unknowns is higher, the trust model tends to completely trust even malicious peers. As a

result, when the number of out edges is increased, peers report malicious peers as trustworthy and hence the number of malicious peers considered trustworthy increases.

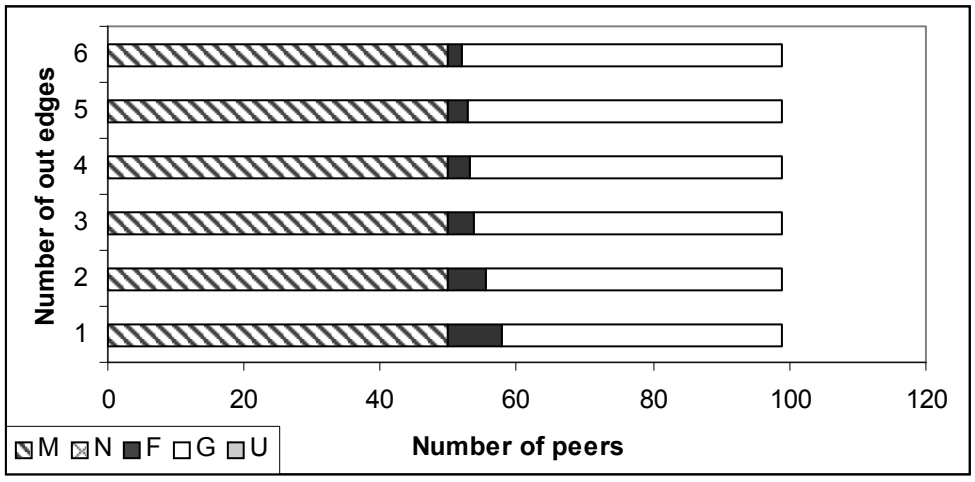

Fig 37. Varying the number of out edges for hc=2, deg=0.5

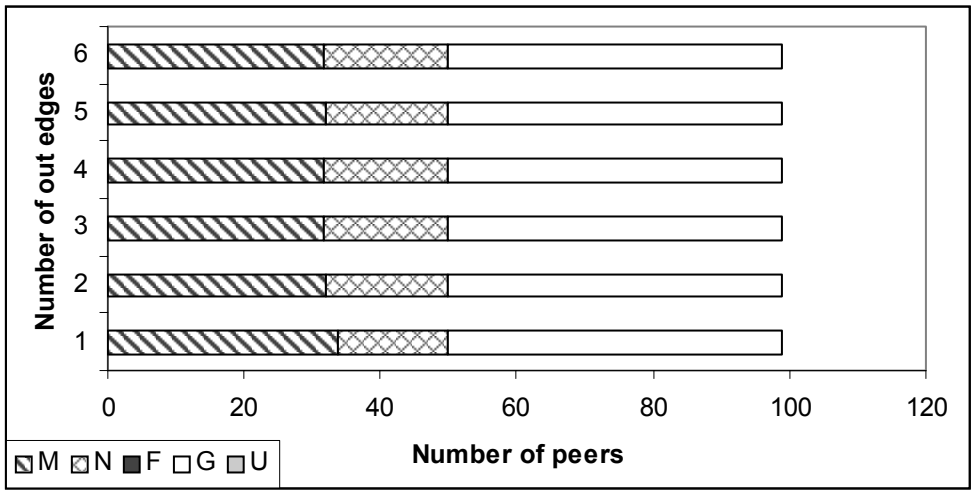

Fig 38. Varying the number of out edges for hc=2, deg=1

Figure 39 portrays the effect of varying the number of out edges for a system with 5000 peer interactions when the degree of trust in unknowns is 2. It can be seen that for low out edges value, there is a significant number of "unknowns" due to limited availability of trust data. However, when the number of out edges

increases, each peer is connected to more peers which results in more trust data being reported. Thus, a greater accuracy in identifying the true nature of peers can be seen as the number of out edges is increased.

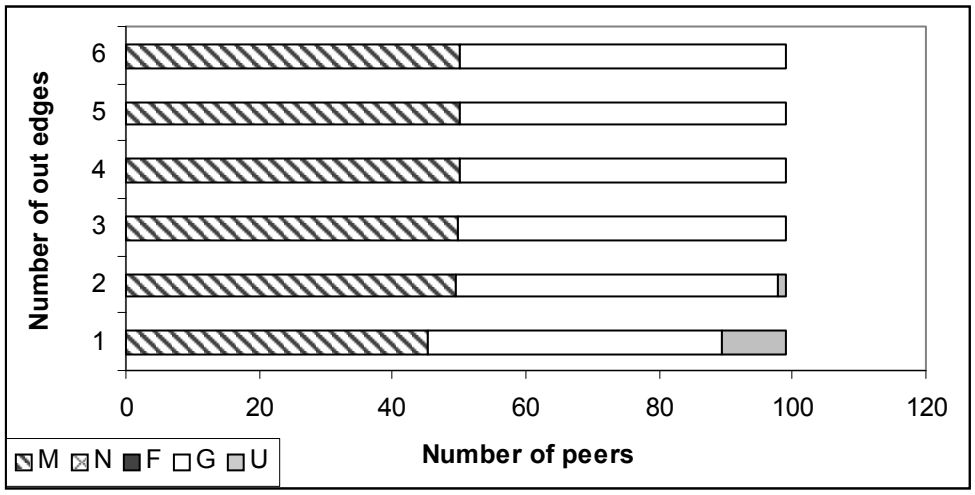

Fig 39. Varying number of out edges for hc=2, deg=2

6.2.6 Effect of 'malicious probability'

So far, the discussion of the various trust and application parameters has been focused on applications where malicious peers always consistently indulge in fraudulent actions i.e. malProb=1. This section now discusses the effect of varying malicious probability on the behavior of a trust model. Figure 40 shows the effect of varying the number of interactions with degree of trust in unknowns equal to 0 and malicious probability equal to 0.1. It can be seen that as the number of interactions increases, both the number of malicious peers successfully detected and the number of false positives decreases.

The reason for this difference in behavior from what was studied in section 7.7.1 is the value of the malicious probability. When peers always engage in fraudulent actions (malProb=1), it becomes easy to detect them, as described in section 7.7.1. However, in this case, since the malicious probability has a very low value implying that malicious peers only engage in fraudulent actions once in a while, both the assessing peers and the other peers in the system have a relatively high opinion about these malicious peers. Hence, though the information reported by these malicious peers is always untrue, their good reputation built through several successful interactions is believed by the assessing peers. The good behavior observed at lower number of interactions is because there is not enough trust data and peers completely mistrust unknown peers due to 0 degree of trust in unknown peers.

Figure 41 similarly shows the effect of varying the number of interactions for degree of trust in unknowns equal to 0 and a malicious probability equal to 0.3. Figure 42 shows that as the value of malicious

probability increases, the trust model behavior becomes better as it becomes easier to correctly detect malicious peers.

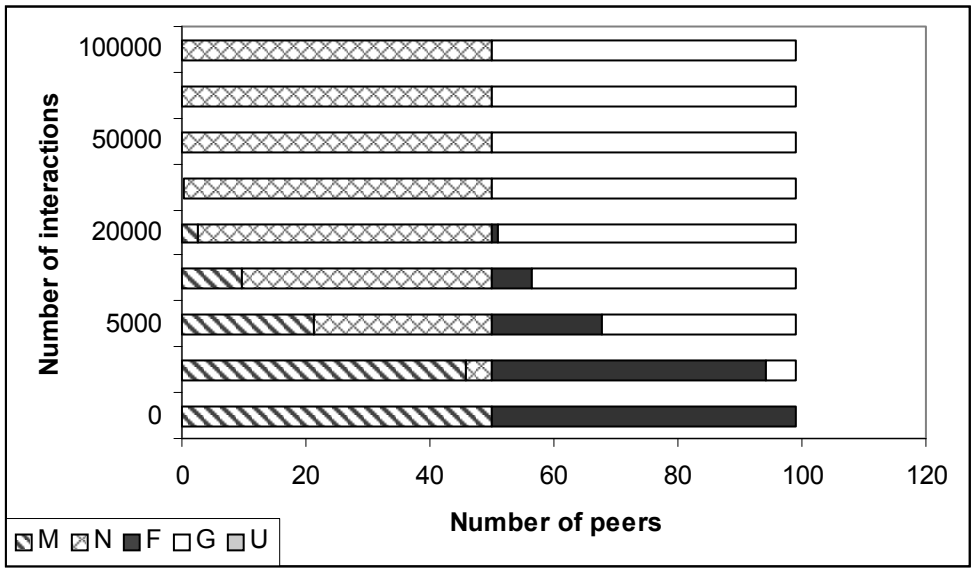

Fig 40. Varying number of interactions for malProb=0.1, deg=0

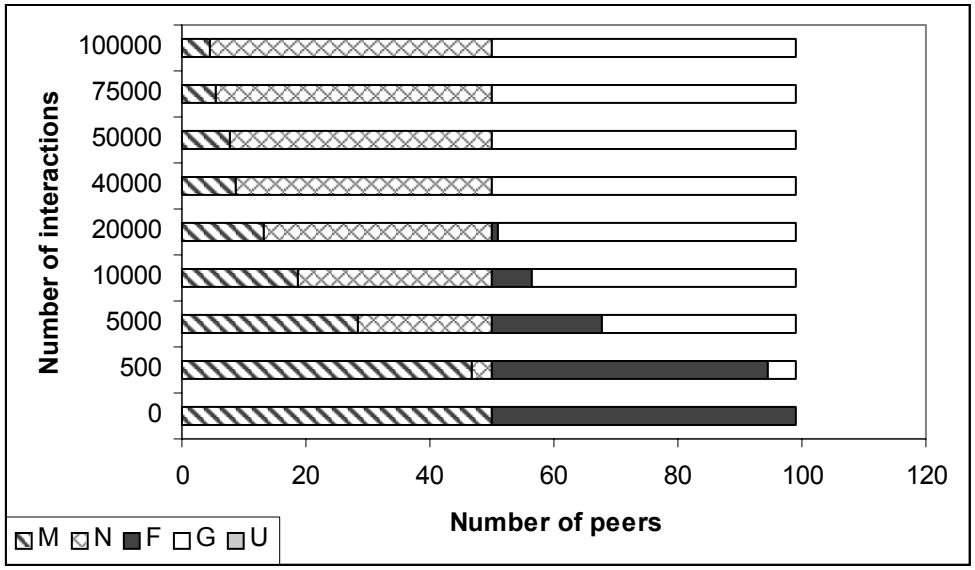

Fig 41. Varying number of interactions for malProb=0.3, deg=0

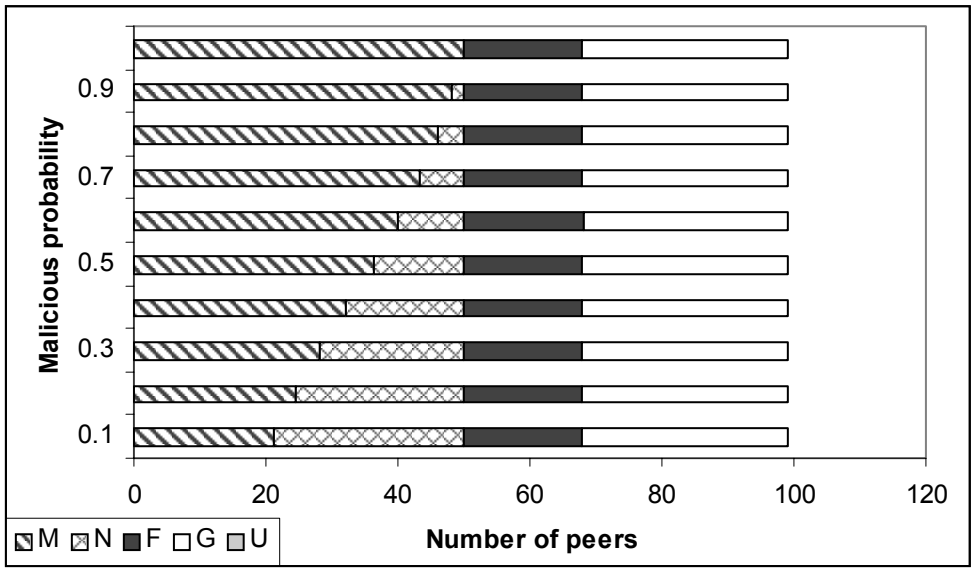

Fig 42. Varying malicious probability for deg=0

Figure 43 and 44 show the effect of varying malicious probability when the degree of trust in unknowns is equal to 1 and 2 respectively. Again, in both figures it can be seen that with increasing malicious

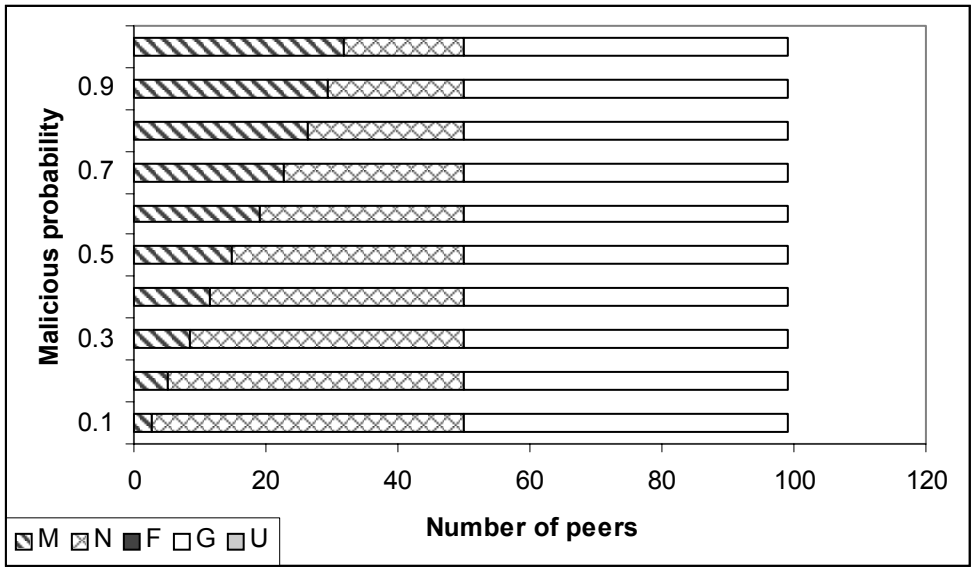

Fig 43. Varying malicious probability for deg=1

probability, the trust model detects a larger number of malicious peers correctly. It can also be seen that when the degree of trust in unknowns is 2, the trust model performs better than when the degree of trust in unknowns is 1.

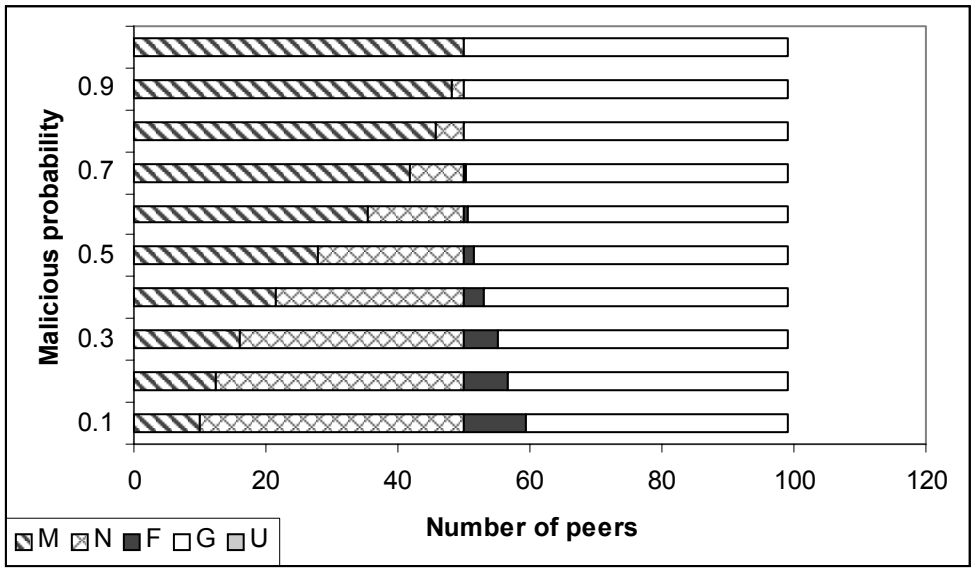

Fig 44. Varying malicious probability for deg=2

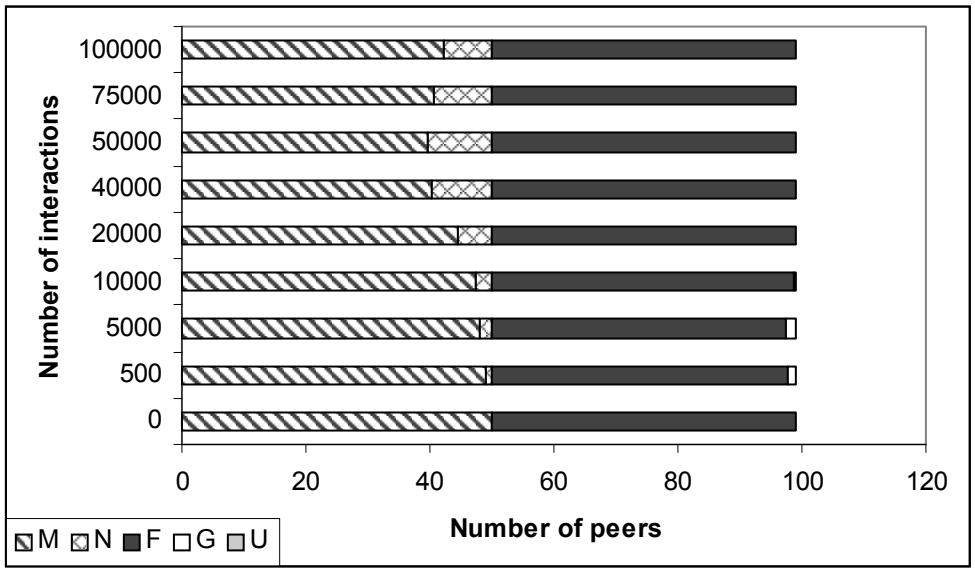

Fig 45. Varying number of interactions for malProb=0.1, deg=0, thr=0.8

Simulation results show that the effect of low malicious probability can be countered by selecting suitable values for the trust threshold and trust weights. For example, consider Figure 40 which shows the effect of varying number of interactions for malicious probability equal to 0.1 and degree of trust in unknowns equal to 0. Figure 45 illustrates how the behavior of a trust model changes when all other parameters retain the same value as in Figure 40 except for the trust threshold which is now increased to 0.8. It can be seen that compared to Figure 40 there is a significant increase in the number of malicious peers detected correctly as well as the number of false positives.

Figure 46 shows the effect of reducing the trust weight on a trust model in the case where all the other trust and application parameters are the same as in Figure 40. Figure 46 shows that when the trust weight is reduced to 0.1, other than a slight increase in the number of malicious peers detected correctly and some slight alterations in the number of false positives and good peers at lower interaction counts, the behavior

of the trust model remains similar. A further discussion of the relationship between trust weight and malicious probability is included in section 7.8.3.

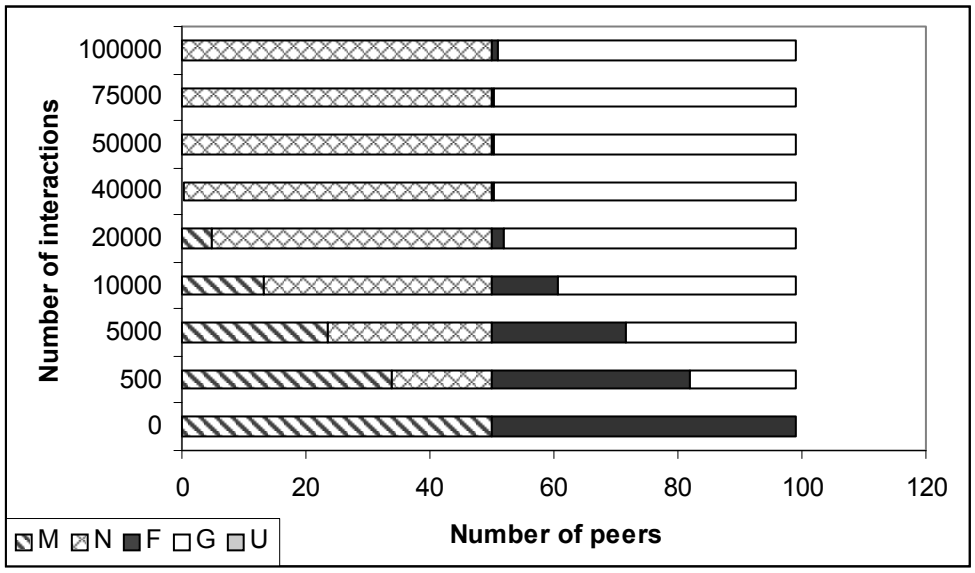

Fig 46. Varying number of interactions for malProb=0.1, deg=0, alpha=0.1

6.2.7 Effect of varying the number of malicious peers

Figures 47 and 48 illustrate how change in the number of malicious peers affects the behavior of the trust model for attack types 0 and 1 respectively. In both cases, the application has 5000 peer interactions with malicious probability value equal to 0.3, and the trust model has a trust threshold value of 0.8 and degree of trust in unknowns is equal to 0.

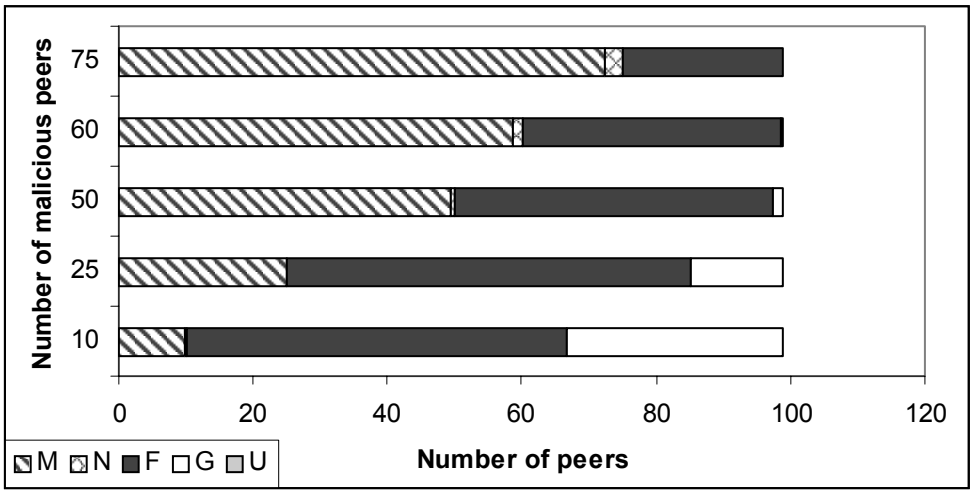

Fig 47. Varying number of malicious peers for malProb=0.3, deg=0, thr=0.8, attack=0

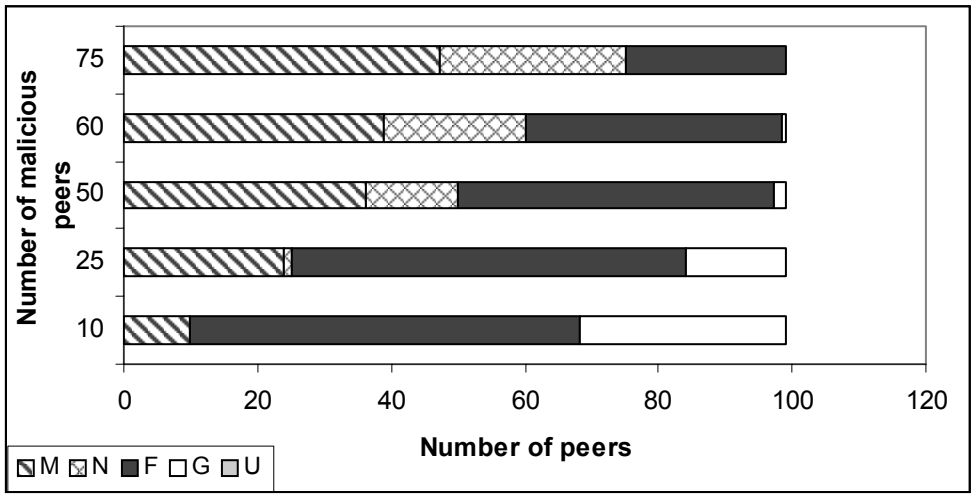

Fig 48. Varying number of malicious peers for malProb=0.3, deg=0, thr=0.8, attack=1

It can be seen that for attack type 0, the trust model manages to detect almost all of the malicious peers correctly. For attack type 1, the trust model manages to detect all the malicious peers correctly when the number of malicious peers is low. However, as the number of malicious peers increases, the number of malicious peers detected correctly by the peer starts to reduce significantly.

This difference in the behavior of the trust model in the two attack scenarios can be explained as follows. In attack type 0, malicious peers misrepresent all other peers, including malicious ones, randomly. However, in attack type 1, in addition to misrepresenting good peers, all malicious peers recommend complete trust (i.e. value 1) in other malicious peers. Hence, as the number of malicious peers increases, it becomes more difficult for the trust model to correctly identify the malicious peers from the good peers.

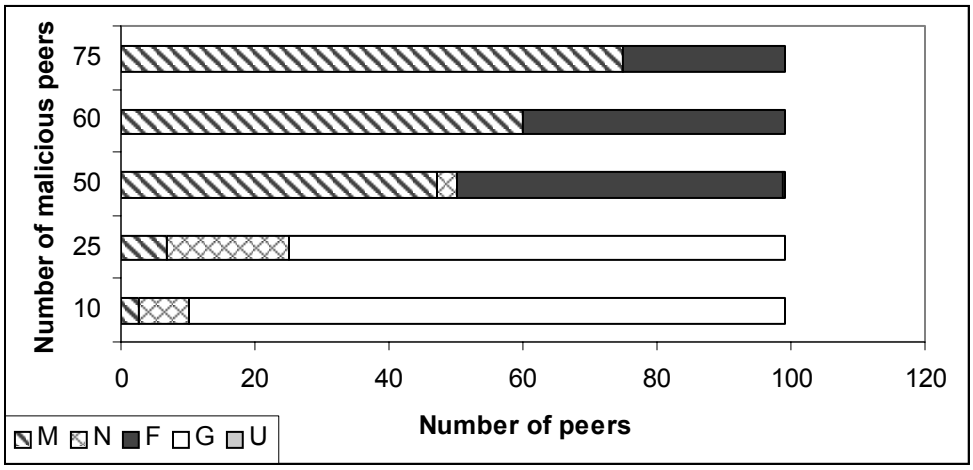

Fig 49. Varying number of malicious peers for malProb=0.3, deg=1, thr=0.8, attack=0

Figure 49 shows the effect of varying the number of malicious peers when the degree of trust in unknowns has a value equal to 1. The trust model and application settings for Figure 49 is identical to that of Figure 48 except for the different value of degree of trust in unknowns. It can be seen from Figure 49 that the trust model fails to identify all the malicious peers when there are only a few malicious peers in the system. This is primarily because the assessing peers completely trust unknown peers. A comparison of the two figures

indicates that a trust model with a degree of trust in unknowns value equal to 0 may be better suited for a system that has a lower fraction of malicious peers.

6.3 Discussion

Results from the above simulations strongly underline the relationship between the choice of a trust model and the application settings. The results also reveal substantial insights into the various trust model parameters and highlight the interplay between these parameters. This section summarizes the interplay of these various parameters and discusses the impact of these simulation results in the context of some trust models.

6.3.1 Effect of application settings on trust model

Results from the above simulations show that application settings significantly impact the choice of a trust model. We draw upon the following specific cases to illustrate this.

First, since the accuracy of trust evaluation depends upon the availability of trust information, it is important to realize the effect of connection density on the hop count value of the trust model. If the nature of the application is such that the connection density is low, then the hop count must be set to a higher value in order to ensure that queries for trust data still reach an enough number of peers so that relevant trust data can be procured.

However, it should be pointed out that just selecting appropriate values for the connection density and hop count is not enough. Other trust parameters also need to be similarly assigned suitable values to optimize trust model behavior under the specified application conditions. This is important because these other parameters may significantly affect the behavior of the trust model. For instance, consider an application that wants to encourage peer interaction and participation. If such an application has a densely connected network of peers, choosing a trust model with a high hop count value may result in the availability of more trust data. However, if the trust model were to employ an extremely high trust threshold value, it could generate a large number of false positives. Thus, instead of identifying a large number of good peers and encouraging interactions as specified by the application requirements, the trust model would discourage interactions by identifying most peers as malicious. Thus, it is important for the trust model to be designed in accordance with the system's requirements.

Second, my simulation results have also shown that the number of peer interactions has a significant impact on the trust model's ability to correctly identify malicious peers from good ones. A greater number of peer interactions implies a larger pool of available experience that typically enables better trust determination. These results seem to suggest that if the application has fewer peer interactions, then there may not be sufficient trust data for a decentralized reputation model to correctly identify good and bad peers. In such a case, we believe a centralized reputation system will provide a better trust management solution. If, however, peer interactions are voluminous, a decentralized reputation system may be well-suited for the system.

Third, results presented in section 6.2.4 point towards a strong relation between the number of interactions and the trust weight. If the number of interactions is small, it would be advisable to have a trust model with a lower trust weight. This would enable the information reported by other peers in the system to be assigned more weight than the peer's own personal perception which is limited due to the low interaction count.

Fourth, results presented in section 6.2.8 reveal how the malicious probability affects trust settings. It is seen that at low values of malicious probability, it is quite hard for the assessing peers to determine the malicious peers correctly. One way to increase the number of malicious peers detected correctly is to increase the trust threshold value; however, this will lead to an increase in the number of false positives (see Figure 45).

Finally, the results discussed in section 6.2.9 indicate that the number of malicious peers can significantly affect the behavior of the trust model under certain attack conditions. Simulation results have also shown how the number of malicious peers in the system can affect the choice of the degree of trust in unknowns. It is, therefore, extremely important for a trust model designer to properly understand the nature of an application so that he can then accordingly choose appropriate values for the various trust parameters and create a trust model that can better guard peers against attacks in that particular application.

6.3.2 Interplay of trust model parameters

In addition to throwing light on the relationship between trust and application parameters, simulation results have also revealed some interesting insights into the interplay of the various trust model parameters. For example, simulation results have highlighted the relationship between the degree of trust in unknowns and other trust model parameters. One such relationship is between the trust threshold and the degree of trust in unknowns described in section 6.2.3. Both these parameters strongly influence the extent to which the trust model is liberal or conservative. One of the specific insights that can be drawn based on these results is that as the degree of trust in unknowns is increased in a system where the malicious probability has a value equal to 1, the trust threshold should also be increased to obtain an optimal trust model behavior.

Another relationship revealed is between the trust weight and the degree of trust in unknowns. Section 6.2.4 shows that for a system with malicious probability value equal to 1 and 5000 interactions, a trust model with degree of trust in unknowns value equal to 1 or 0 performs best at trust weight value equal to 0. However, if the degree of trust in unknowns has a value equal to 0.5, a trust weight value of 0.5 is better suited.

6.3.3 Impact on trust models

This section discusses the impact of the simulation results and the subsequent insights gained from the study of the interplay between the various trust and application parameters on the trust models that were discussed earlier in the related work section.

Distributed Trust Model. As described in Section 3.2.1, in the Distributed Trust Model (DTM), peers treat unknown peers as unknown instead of trusting or mistrusting them. Thus, in the case of DTM, the value of degree of trust in unknowns is 2. Further, in DTM, each peer only relies on its personal perceptions to determine trustworthiness. A peer seeks recommendations from other peers in the system only when it has no prior experience with a peer. Thus, in the case of DTM, the value of trust weight (alpha) is 1.

Section 6.2.4 discusses that for a system with 5000 or more interactions and malicious probability value 1, there is no effect of varying the trust weight when the degree of trust in unknowns has a value equal to 2. Thus, it would seem that having a trust weight value of 1, as in the case of DTM, would not affect the detection of malicious peers. However, the malicious probability has a strong influence on the trust weight.

Figures 50 and 51 show the effect of varying the number of interactions for a system with low malicious probability (malProb=0.1) for a trust model with degree of trust in unknowns value 2. DTM is represented by Figure 50 which shows the effect when the value of trust weight is 1. Figure 51 shows the case when the trust weight is 0.

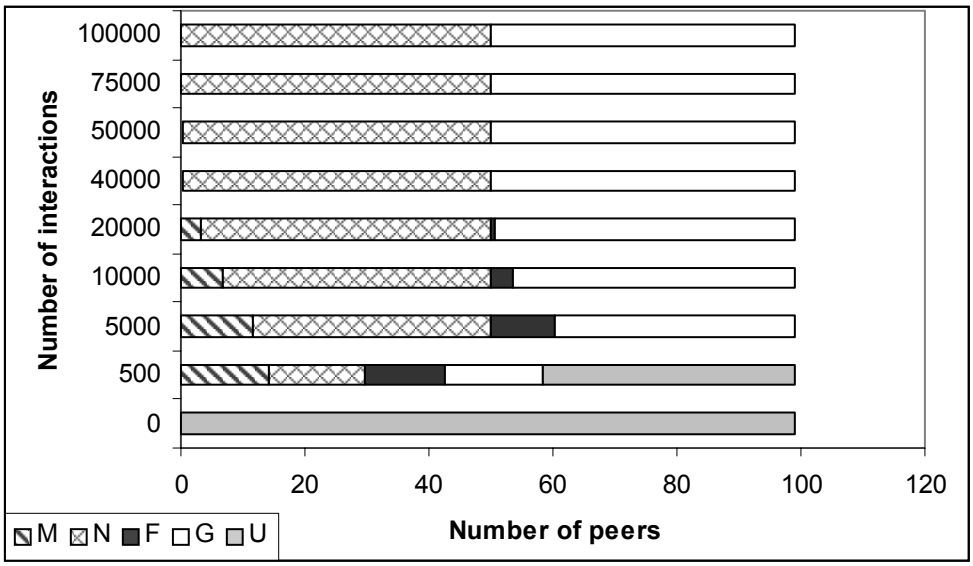

Fig 50. Varying number of interactions for malProb=0.1, deg=2, alpha=1

It can be seen from these two figures that for applications where the number of peer interactions are not high and the malicious probability is low i.e. malicious peers do not always engage in fraudulent actions, a trust model that treats unknown peers as unknown (deg=2) may be better able to detect malicious peers if it has a low trust weight i.e. peers rely less on their personal perceptions. However, this may come at the expense of significant increase in the number of false positives as indicated in Figure 51.

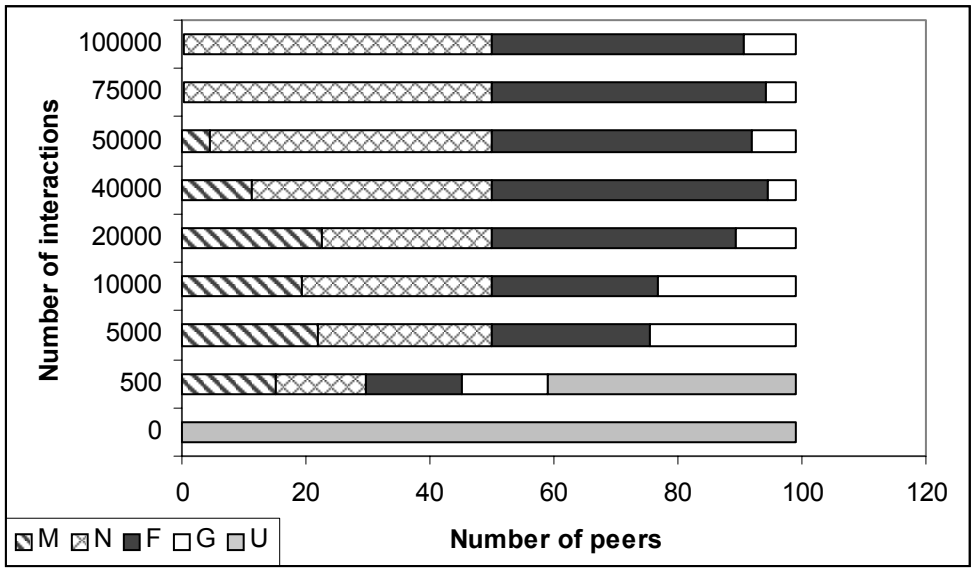

Fig 51. Varying number of interactions for malProb=0.1, deg=2, alpha=0

Complaint-based Model. The Complaint-based model introduced in Section 3.2.2 is a negative reputation-based system. Specifically, peers only store negative reputation information about other peers in the form of complaints. Before a peer interacts with another peer, it searches for complaints about that peer. If it does not receive any complaints from other peers in the system, the peer assumes that the other peer is trustworthy. Thus, this is a model where the degree of trust in unknowns has a value equal to 1.

Simulation results presented in Figure 6 indicate that the Complaint-based model would be suitable for in a system that has more than 20000 peer interactions and the remaining system characteristics match the application settings of Figure 6. However, in an application with fewer interactions between peers, it would be better for a model that has a degree of trust in unknowns value equal to 1 to rely more on the complaints reported by other peers and less on its own perception. This is true when the malicious probability is high (see Figure 21 where malProb=1) as well as when the malicious probability is low (see Figure 52 where malProb=0.1).

Though the Complaint-based model seeks complaints from other peers in the system, it does not precisely specify how these different complaints are combined to compute trustworthiness. In other words, there is no trust weight specified in the Complaint-based model. Therefore, it is important for an application

designer who is planning to use the complaint-based model or any other negative reputation-based model, to realize the significant role of trust weight and specify it clearly.

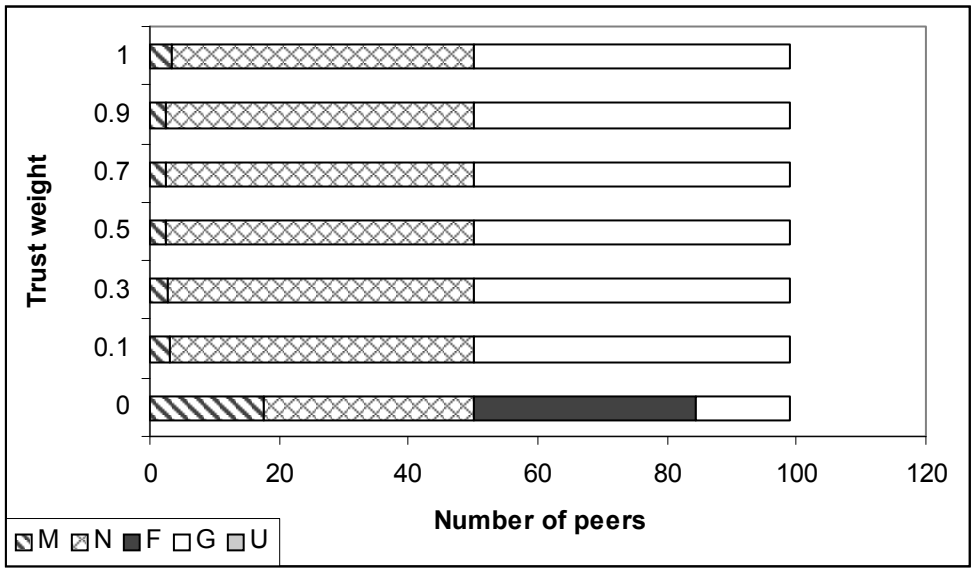

Fig 52. Varying number of interactions for malProb=0.1, deg=1

XREP. Another trust model that can benefit from an appropriate choice of trust weight is XREP [6]. As mentioned previously in Section 3.2.3, XREP in phase 4 chooses to download needed resources from the servent with the best reputation. The reputations of the servents, in this case, are based solely on the votes received from recommenders in the system. It may be possible that recommendations are more reliable than personal experiences in computing trustworthiness; however, XREP does not present any results that indicate such a case. XREP does not clearly specify if and how the personal experience of a peer is combined with these recommendations to compute the reputation.

Further, simulation results discussed earlier in this report show that there is a definite relationship between the trust weight and the degree of trust in unknowns. In other words, depending upon the value of the degree of trust in unknowns, a peer's personal experience can positively or adversely affect the accuracy of trust determination. XREP does not clearly specify to what extent it trusts unknown peers. SIFT simulation results can help a trust model designer intending to use XREP to make appropriate choices for the trust weight and degree of trust in unknowns.

eBay-like Reputation System. As discussed in Section 3.2.4, the eBay reputation system [8] provides only a simplistic aggregation of a user's ratings as reported by other users of eBay. eBay adds up all the unique positive ratings and subtracts all the unique negative ratings received by a user and displays the net value as the score of that user. While computing this net score, eBay does not take into account whether the ratings reported by a user are actually truthful or not. While eBay does enable access to the past interaction profiles of each of these recommending users, it is rather cumbersome for a user to combine all this extensive reputation information to make a correct and rapid decision about whether or not to transact with a particular user.

Specifically, there are a number of relevant questions that need to be answered to facilitate trust determination. Should a user rely only on his own past experience? Or, should a user rely on both past experience and information reported by other users, and if so, what should be the optimal way of combining those two types of information? How can a user decide if a reporting peer is lying?

The SIFT simulator can serve as an effective platform for designing a suitable reputation model that will address these questions. To illustrate this, let us assume that a trust model designer has to design a reputation model for an eBay-like system. The first step would be to characterize the system in terms of the SIFT application parameters. Let us assume that estimations by domain experts indicate that about one-fourth of peers in this eBay-like system will be likely malicious and indulge in fraudulent interactions 3 times out of every 10 interactions. This means that if the system has 100 peers, 25 of them are malicious with a malicious probability of 0.3. Let us further assume that the domain experts are not sure how popular this system will be and so estimate that each peer could on an average engage in 50 to 100 interactions. This implies that the total number of initial interactions for the SIFT simulations would be anywhere between 5000 to 10000.

Since this system, like eBay, is centralized in terms of storing reputation information, peers have access to all the trust information. In order to model this in SIFT, the connection density is assigned a high value (equal to 6) and the trust model being designed is assumed to have a high hop count value (equal to 5). Since eBay presents the entire transaction history of a peer in the peer's profile, it means that the entire information can be used in the determination of trustworthiness. Thus, the interaction limit for the trust model is set to 100%. Further, let us assume that users in this system do not form an impression or report reputation information about another user without ever having interacted with that user. This implies that the degree of trust in unknowns for the trust model is set to value 2.

Let us also assume that the trust model designer has some previous experience and realizes that when the malicious probability is low, it is better to have a higher trust threshold value. Based on this knowledge, the trust model designer sets the trust threshold value to 0.8. However, he has no idea about how the personal perception of peers must be combined with the information reported by other peers. In other words, he needs to decide the value of the trust weight.

A trust model designer can now use SIFT to rapidly experiment with different values of the trust weight for different number of initial interactions. He can then use the simulation results to decide what value of trust weight would best fit a particular type of application. Figures 53 and 54 indicate the effect of varying trust weights for the eBay-like system with 5000 and 10000 interactions respectively for attack type 0. Both figures show that as the trust weight is increased, the trust model fails to detect malicious peers correctly. However, the number of good peers correctly identified increases significantly at low trust weights and becomes roughly constant at higher trust weights.

Let us assume that the stakeholders for this eBay-like system consider it more important for the candidate trust model to correctly detect malicious users rather than good users. In other words, it is more important for the stakeholders that users of the system be aware of malicious users and refuse to interact with them even though this may prevent interaction with good users that have been wrongly tagged as malicious. Thus, increasing the number of malicious peers identified correctly is more important than reducing the number of false positives. With this in mind, the trust model designer sees that for the case where there are 5000 interactions, there is a small fraction of malicious users that have not been identified correctly. At this

point, he can decide to continue the simulations with trust weight values lower than 0.25 until he reaches the point where all malicious peers have been detected correctly.

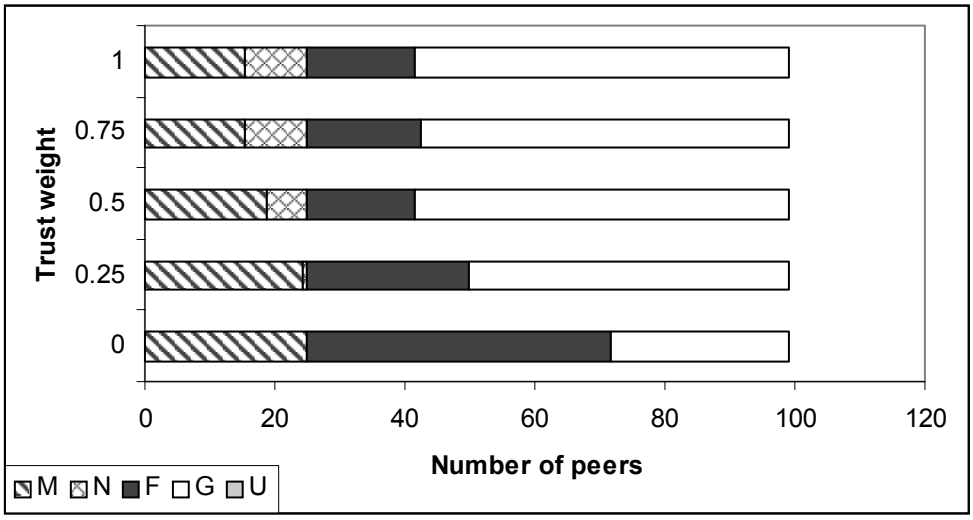

Fig 53. Varying alpha for malProb=0.3, deg=2, int=5000

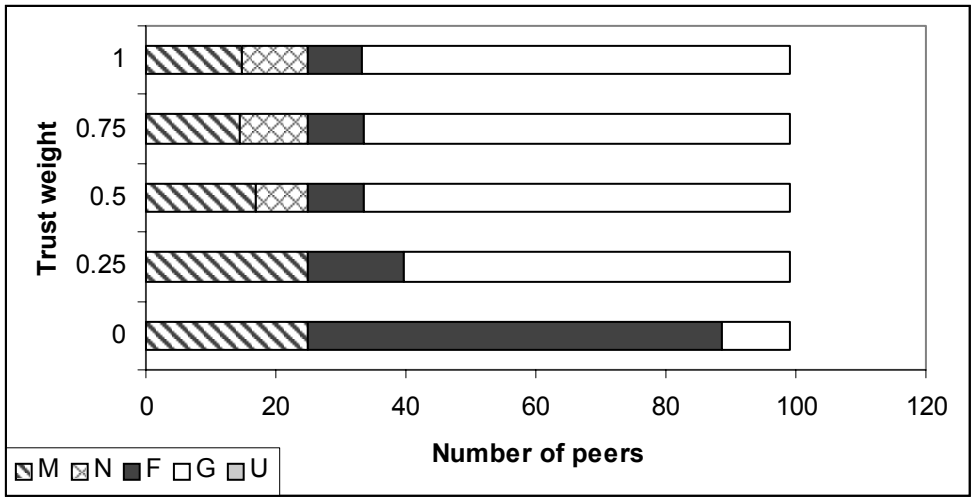

Fig 54. Varying alpha for malProb=0.3, deg=2, int=10000

Figure 54, on the other hand, shows that all the malicious peers have been correctly detected for trust weight value equal to 0.25. Since the primary goal of being able to identify all the malicious peers has been attained, the trust model designer can choose to explore whether increasing trust weight values beyond 0.25 can further reduce the number of false positives without affecting the number of malicious peers detected correctly. However, he does recognize that the upper limit for this is a trust weight value of 0.5 from Figure 54.

7. Limitations and Future Work

Using the current set of parameters for characterizing trust models and decentralized applications results in a huge design space for exploration. As can be seen from the discussion of simulation results, there exist rich and complex relationships between these parameters. The simulation results presented in this report

only explore a small part of this vast design space and are not exhaustive by any means. Even so, the results presented here have provided several insights into the relationship between the nature of an application and the choice of a suitable trust model for that application. Using SIFT to investigate these parameters further has the potential to reveal significant variation points and important insights in the future.

While the current simulation results provide several interesting insights, it should be pointed out that the sets of parameters used to characterize trust models and decentralized applications are not exhaustive in any way. These parameters have been so chosen as to capture the dominant attributes of trust models and applications. Additionally, these parameters help provide a sound initial basis to reason about the relationship between the nature of the application and the trust model that is chosen to provide trust management for that application. Finally, it should also be pointed out that the number of parameters used to characterize trust models and applications has been purposely limited in order to constrain the extensive design space for trust models. We envision that these parameters can be added to the existing set of trust and application parameters in the future to provide a richer characterization of trust models and decentralized applications. This can potentially result in a more elaborate set of simulation results and insights. We discuss a few of these additional parameters below.

As mentioned previously, the malicious probability parameter in SIFT currently only dictates how often malicious peers engage in fraudulent interactions. There is no similar parameter in the SIFT simulator to model how often malicious peers misrepresent their trust in other peers; instead, currently in SIFT, depending upon the type of attack, all malicious peers always misrepresent their trust in others. Thus, there is a need for an application parameter that would indicate the frequency with which malicious peers misrepresent trust.

Simulation results presented in this report are based on only one round of querying and assessment since SIFT currently assumes that malicious peers always misrepresent their trust. However, the number of assessments can start to play a significant effect when malicious peers inconsistently misrepresent trust. Specifically, in this case, by aggregating trust evaluations at the end of each querying and assessment round and comparing them with the evaluations of the previous round will enable peers to gradually gain a better idea of the true nature of peers.

Further, SIFT currently includes only one round of interactions followed by a round of querying and evaluation. The results of peer evaluations are not used to guide subsequent rounds of interactions and evaluations that can potentially help expose faster the nature of peers. Extending SIFT to support this could be very useful especially in scenarios where the malicious probability has a low value and it is difficult to decide whom to trust. In such cases, SIFT could be used to determine how many rounds of interactions and evaluations will be required before the trust model eventually identifies all the malicious peers correctly.

Another parameter that can be added to the set of application parameters is the degree of maliciousness. This parameter would indicate how strongly malicious peers misrepresent other peers. This parameter is important because it helps model the extent to which malicious peers disguise their misrepresentation of trust in others. For example, while misrepresenting a peer, a malicious peer may choose to propagate either extremely low or high trust values, or slightly lower or higher trust values so that they are not easily detected.

Trust is context-based and so it is important to consider the context while evaluating the reputation of a peer. Similarly, in social reputation-based systems, group reputations and group relationships play an important role in determining trust. We envision SIFT can be extended in the future to derive and match context relationships as well as use group reputations to better capture trust relationships. We also envision

that in the future SIFT can incorporate recent research concepts and approaches including risk-based and incentive-based trust evaluation mechanisms [5, 11, 13].

8. Acknowledgements

We thank Justin Erenkrantz for his feedback. This material is based upon work supported by the National Science Foundation under Grant Number 0524033.

9. References

[1] Abdul-Rahman, A. and Hailes, S. Supporting trust in virtual communities. In *Proceedings of the Hawaii International Conference on System Sciences.* Maui, Hawaii, Jan 4-7, 2000.

[2] Aberer, K. and Despotovic, Z. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Conference on Information and Knowledge Management.* Atlanta, Georgia, November 5-10, 2001.

[3] Bailey, B., Gurak, L., et al. An examination of trust production in computer-mediated exchange. In *Proceedings of the Seventh Human Factors and the Web.* Madison, WI, June 4, 2001.

[4] Barabasi, A., Albert, R., et al. Power-law distribution of the World Wide Web. *Science.* 287(5461), p. 2115a, March, 2000.

[5] Brainov, S. Incentive Compatible Trading Mechanism for Trust Revelation. In *Proceedings of the IJCAI Workshop on Economic Agents, Models and Mechanisms.* p. 62-70, Seattle, WA, Aug. 6, 2001.

[6] Damiani, E., di Vimercati, S.D.C., et al. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security.* Washington DC, November, 2002.

[7] Deutsch, M. Cooperation and Trust: Some Theoretical Notes. In *Nebraska Symposium on Motivation*, Jones, M.R. ed. Nebraska University Press, 1962.

[8] eBay. *www.ebay.com*. <www.ebay.com>.

[9] Gambetta, D. *Trust*. Gambetta, D. ed. Blackwell: Oxford, 1990.

[10] Grandison, T. and Sloman, M. A Survey Of Trust in Internet Applications. *IEEE Communications Surveys.* 3(4), December, 2000.

[11] Josang, A. and Presti, S. Analysing the Relationship Between Risk and Trust. In *Proceedings of the 2nd International Conference on Trust Management.* Oxford, UK, Mar. 29 - Apr. 01, 2004.

[12] Kamvar, S., Schlosser, M., et al. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the WWW.* Budapest, Hungary, May 20-24, 2003.

[13] Lai, K., Feldman, M., et al. Incentives for Cooperation in Peer-to-Peer Networks. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems.* Berkeley, CA, Jun. 5-6, 2003.

[14] Lee, S., Sherwood, R., et al. Cooperative peer groups in NICE. In *Proceedings of the IEEE Infocom.* San Francisco, USA, April 1-3, 2003.

[15] Liang, Z. and Shi, W. PET: A PErsonalized Trust Model with Reputation and Risk Evaluation for P2P Resource Sharing HICSS-38, January, 2005. In *Proceedings of the Hawaii International Conference On System Sciences.* Waikoloa Village, Hawaii, Jan 3-6, 2005.

[16] Lv, Q., Cao, P., et al. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the International Conference on Supercomputing.* New York, USA, June 22-26, 2002.

[17] Marsh, S. *Formalising Trust as a Computational Concept.* Thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.

[18] Marti, S. and Garcia-Molina, H. Limited Reputation Sharing in P2P Systems. In *Proceedings of the ACM Conference on Electronic Commerce.* New York, USA, May 17-20, 2004.

[19] Ripeanu, M. and Foster, I. Mapping the Gnutella Network - Macroscopic Properties of Large-scale P2P Networks and Implications for System Design. *Internet Computing Journal.* 6(1), 2002, 2002.

[20] Suryanarayana, G. and Taylor, R.N. *A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications.* UCI Institute for Software Research, Technical Report UCI-ISR-04-6, July, 2004.

[21] Suryanarayana, G., Diallo, M., et al. Architectural Support for Trust Models in Decentralized Applications. In *Proceedings of the 28th International Conference on Software Engineering.* Shanghai, China, May 20-28, 2006.

[22] Walsh, K. and Sirer, E.G. *Thwarting P2P Pollution Using Object Reputation.* Department of Computer Science, Cornell University, Report cul.cis/TR2005-1980, 2005.

[23] Xiong, L. and Liu, L. A Reputation-Based Trust Model for Peer-to-Peer eCommerce Communities. In *Proceedings of the Fourth ACM Conference on Electronic Commerce.* p. 228-229, San Diego, CA, USA, June 09-12, 2003.

[24] Yu, B. and Singh, M.P. Detecting Deception in Reputation Management. In *Proceedings of the Autonomous Agents and Multi-Agent Systems.* Melbourne, Australia, July 14-18, 2003.

[25] Zacharia, G. and Maes, P. Trust Management Through Reputation Mechanisms. *Applied Artificial Intelligence.* 14, p. 881-907, 2000.