Reputation Management Framework and its use as Currency in Large-Scale Peer-to-Peer Networks

Rohit Gupta and Arun K. Somani
Dependable Computing and Networking Laboratory
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
E-mail: {rohit, arun}@iastate.edu

***Abstract*— In this paper we propose a reputation management framework for large-scale peer-to-peer (P2P) networks, wherein all nodes are assumed to behave selfishly. The proposed framework has several advantages. It enables a form of virtual currency, such that the reputation of nodes is a measure of their wealth. The framework is scalable and provides protection against attacks by malicious nodes. The above features are achieved by developing trusted communities of nodes whose members trust each other and cooperate to deal with the problem of nodes' selfishness and possible maliciousness.**

I. Introduction

In this paper we propose a reputation management framework for large-scale peer-to-peer (P2P) networks, wherein all nodes are assumed to behave selfishly. By selfish we mean that nodes aim to maximize their own reputation in relation to others in the network. The proposed framework has several advantages. It enables a form of virtual currency, such that the reputation of nodes is a measure of their wealth. The framework is scalable and provides protection against attacks by malicious nodes. The above features are achieved by developing trusted communities of nodes whose members trust each other and cooperate to deal with the problem of nodes' selfishness and possible maliciousness.

The concept of currency can be used for designing network operation protocols for various distributed systems [11]. Such systems require giving incentive or payoffs to nodes in order to make them cooperate. Adopting a similar approach, several incentive-based protocols exist that reward nodes in the form of increased credit or currency for their cooperation in sharing resources. In all of these currency or "money" is used as a standard tool to achieve the desired network goals. However, a limitation of such schemes is that they either assume the existence of an infrastructure for electronic currency or some trusted centralized entity that maintains the debit/credit values of the nodes. These assumptions can be difficult to enforce in P2P networks.

However, reputation can be used as a basis to provide and procure services. These services can range from sharing data, such as audio/video files, to providing complex distributed processing capabilities as in SETI@Home. In such on-line economies, service providers are suitably compensated for the resources/services they provide. The service providers are willing to serve nodes with higher reputation to increase their own reputation. The earned reputation in turn makes it easier for them to obtain services in future. Thus, the objective of nodes is to maximize their reputation as viewed by everybody else in the system.

In order to be useful as a substitute for actual currency, we identify the following minimum guarantees that a reputation management framework needs to provide. It should be easier for nodes with higher reputation to access network services. When two nodes contend for a service and only one of them can be served, a node with higher reputation should be serviced and for this to happen the service provider should have incentive to serve the more reputed node. Our framework provides this by ensuring that the reputation of a node increases more by serving a higher reputation node. This is in contrast with traditional reputation schemes, where service providers make no distinction between the type of nodes (whether of low or high reputation) being served. Moreover, like real currency it must be possible for nodes to protect their reputation against attacks by malicious nodes. Furthermore, malicious nodes either individually or in collusion should not be able to falsely increase their reputation.

However, there are important distinctions between real currency and using reputation as a substitute. Real currency can precisely be measured. But reputation is an estimation and different nodes may assign different reputation to a node. Therefore, valuation error is inherent in systems using reputation as a measure of nodes' wealth. Moreover, unlike real currency, reputation gets depleted through expiration and malicious behavior, and so one needs to constantly provide service in order to maintain high reputation. Furthermore, reputation is not necessarily spent or reduced when acquiring service(s), which is not true if real currency is used.

The paper is organized as follows. Section II is on related work. Section III presents the system model. In Section IV, we introduce the notion of trust groups. Section V explains the proposed reputation management framework in detail, which is evaluated through simulation results in Section VI. We conclude the paper in Section VII.

II. Comparison with Related Work

In past several reputation management systems ([3], [4], [5], [6], [9], [10], [7]) have been proposed specifically for P2P

The research reported in this paper is funded in part by Jerry R. Junkins Chair position at Iowa State University.

systems. Almost all the proposed schemes assume that majority of the nodes truthfully follow a given protocol, so as to identify the malicious nodes in the system. However, this assumption does not hold true when we use reputation as a substitute for currency. This is because everyone now has incentive to behave in a manner so as to maximize their own and minimize others' reputation. Some examples of such a behavior include the following - limiting the amount of positive information regarding others, propagating false information about others' behavior, giving poor recommendations etc. A framework for reputation management for such systems should therefore explicitly take into account nodes' selfishness, and assume that nodes would follow a given protocol only if it maximizes their own reputation. Besides enabling use of reputation as currency, our proposed framework has several other desirable characteristics as outlined below.

Reputation management schemes, such as [6], [8], do not scale well to large systems. This is because they assume global knowledge and require nodes to store information about every other node in the system. This can be very difficult to implement in large systems.

The reputation ratings generated by our framework identifies the degree of trustworthiness of nodes. Proposals such as [9] and [12] consider two and four, respectively, different possible trust levels. This coarse granularity makes it difficult to reason about trust explicitly as it cannot differentiate between nodes with the same trust level even though they provide different levels of services. Also, the relative reputation rating system of [6] makes it difficult to distinguish good nodes from bad.

The authors in [7] develops a trust and security architecture for a routing and node location service that uses a trust protocol, which describes how honest nodes perform. On the other hand, the reputation management framework presented in the paper is not just limited to enable cooperative routing, but instead is generalized enough to be used irrespective of the nature of services being traded in the network.

The NICE system [10] aims to identify rather than enforce the existence of cooperative peers. It claims to efficiently locate the generous minority of cooperating peers and form a clique of users all of whom offer local services to the community. The system takes a novel approach, rather than using economics to model trust, it proposes using trust to model expected service prices.

III. System Model

We assume a P2P network where nodes' wealth is measured by their reputation. Nodes are selfish in the sense that their goal is to obtain desired services and maximize their reputation relative to others. Both the service providers and receivers benefit by obtaining accurate trustworthiness assessment of each other. Service providers want to serve highly reputable clients in order to maximize their own reputation and service receivers want to maximize the quality of service they receive. The service can be as simple as forwarding a routing message or as complex as downloading data, sending a task for remote execution etc. For concreteness, in the remainder of the paper we use file sharing as the example of service being provided in the network.

Each node has a specific area of interest and provide services related to that interest category. For example, downloading and caching of certain data types, such as audio/video files belonging to a certain music group or artist. We assume that a node supports at most one service category. Service categories are denoted by $S = \{S_1, S_2, \ldots, S_n\}$, where *n* is the number of different services provided. The network incurs a cost equal to C_S for completing service S_i. This cost is a well-known value and is due to network bandwidth consumed, buffer space occupied etc. For simplicity, C_S is assumed to be constant for all services and/or location of service providers and receivers in the network.

We assume that each node has a unique public-private key pair, and there is a mechanism to reliably obtain a node's public key given its IP address.

There may be malicious or bad nodes, which provide bad service and/or aim to disrupt the normal operations of a network. Example of malicious activities include spreading viruses, propagating wrong information regarding others etc. Non-malicious or good nodes, although selfish, do not provide bad service or try to disrupt the operations of a network. Good nodes are differentiated based on the probability (called *service probability*) with which they provide service, if selected by a client. All nodes either belong to a set of good nodes (*Good*) or bad nodes (*Bad*). The total number of nodes are denoted by *N* (=$|Good| + |Bad|$).

A node *i* maintains three sets $Good_i$, Bad_i, and ϕ_i, which are the set of good, bad, and unknown nodes, respectively, as perceived by it. ϕ_i represents the set of nodes about whom there is insufficient (or no) information, which allows them to be classified as either good or bad. Initially all the nodes are put in ϕ_i. With time, as more experience is gained about these nodes, their reputation ratings are updated and if appropriate they are moved to either $Good_i$ or Bad_i. For each positive (negative) information that a node receives, it increases (decreases) the reputation of the service provider based on the reputation of the service receiver.

Nodes form trusted communities whose members trust each other to be good nodes and rely on each other for protection from malicious nodes. Such communities are called *trust groups* (TGrps). The members of a TGrp share the same reputation as viewed by the outside nodes. For example, if the reputation of a TGrp with *S* number of nodes is *R*, then the reputation of an individual member node is R/S. We assume that each node belongs to at most one TGrp. TGrp members are also referred to as *peers*. TGrp of a node *x* is denoted by $xTGrp$ and the same notation is used to denote the set of nodes that comprise a TGrp. Thus, $xTGrp$ is a set of nodes consisting of *x* and its peers. If two nodes *x* and *y* are members of the same TGrp, then we say $xTGrp = yTGrp$.

Reputation of a node is a decaying function of time, i.e. nodes need to continuously share resources and provide services. A node's reputation is incremented only if it serve someone outside its TGrp.[1] If this is not the case then malicious nodes can easily collude and increase their reputation by only serving

[1]As we would see in Section V-B, when a node serves someone in its TGrp, only the peers increase the serving node's reputation.

each other. The reputation of a node is updated after each service transaction and is represented as follows.

$$R_{provider-new} = f(R_{provider-old}, R_{receiver}) \quad (1)$$

Here, f is a function that takes as input the reputation of a service provider ($R_{provider-old}$) and receiver ($R_{receiver}$), and outputs an updated reputation value for the service provider ($R_{provider-new}$). The output value is directly proportional to the reputation of the service receiver. Thus, nodes have incentive to serve reputable good nodes. The above expression holds true for updating a TGrp reputation also, i.e. if node *x* serves node *y*, the reputation of *xTGrp* is increased in accordance with the reputation of *yTGrp*.

A. Service Information Propagation

If a node provides good service then only the network neighbors and TGrp members of the concerned nodes (i.e. the service provider and receiver) become aware of the service transaction. This is because selfish nodes do not propagate positive information in order to prevent others from increasing their reputation. On the other hand, negative information is readily propagated to lower the reputation of the affected nodes. We assume that before a client receives service, it provides information on its neighbors and TGrp members to the service provider. At the end of a successful service completion, the client signs a *satisfaction certificate* (a form of digital certificate) and gives it to the server. The satisfaction certificate is a proof that the server provided good service to the client. The server on receiving the satisfaction certificate sends it to the nodes known to it. These nodes (except the peers), however, do not have incentive to further pass on this positive information.

On the other hand, if a server provides bad service, then that information is readily propagated, first by a client to the nodes known to it and then recursively to other nodes known to each of them. As a result, negative information is known to a large fraction of the total nodes and its propagation can be assumed to follow a flooding mechanism. Although, it is not straightforward for a node that received bad service to prove that fact, it can still send a *complaint* against the server. The complaint message contains digitally signed information about the service provider from which bad service was received. Since, complaints can be easily (even falsely) initiated, it is difficult for the receiving nodes to ascertain the validity of such messages, and consequently reputation of both the nodes (complainant and complainer) is reduced. The decrease in each node's reputation is in proportion to the other node's reputation. This works as a disincentive for selfish good nodes to falsely propagate negative information. Still, sending a complaint against a malicious node would not significantly affect a good node, as malicious nodes typically have low (zero or negative) reputation.

IV. Trust Groups

A. Advantages of TGrps

- Members of a TGrp cooperate to minimize the damage that malicious nodes can cause to their reputation. This is because a node's reputation is derived from its TGrp's reputation, which in turn is dependent on the aggregate service provided by all the member nodes. Thus, even if malicious nodes target a good node by sending false complaints against it, the reputation of the target node is not severely affected.
- TGrp members assist in positive information propagation, i.e. when a member node provides good service, its peers help to propagate that information to as many other nodes as possible.
- TGrps provide scalability to the reputation management system because nodes are judged based on their TGrps. Therefore, one needs to keep reputation information about TGrps only, which are much less in number than the total number of nodes in a network.
- Recommendations from peers are more reliable. This is because peers have different areas of interest (and thus minimum conflict of interest), and so have little incentive to provide misleading recommendations.

B. TGrp Management

In this section, we briefly discuss how nodes are added and deleted from a TGrp. The most important requirement for selecting peers and forming a TGrp is that the constituting members trust each other to be good nodes. This is because the reputation of a node is determined by the reputation of its TGrp, which in turn depends on the aggregate service provided by all the member nodes. Thus, new members are carefully screened before admitting them to a TGrp. Likewise, new nodes carefully select which TGrp (if any) they should join.

A node is added to a TGrp if it is not in set, Bad_i, of any of the existing member nodes, and its reputation is greater than that of any of the existing TGrp members as viewed by some member node. Another important criterion for forming a TGrp is to ensure that nodes have no (or minimum) conflict of interest among themselves. Conflict of interest arises if a node suffers a potential loss by increasing the reputation of one of its peers. This can happen, for example, if two peers having same area of interest contend for the same resource. If the resource can be awarded to only one of them, then it is in each node's best interest to not cooperate and in fact lower each other's reputation. Therefore, we assume that TGrp members belong to different service categories.

Now we look at the reverse process, i.e. how nodes leave or are forced out of a TGrp. It is generally difficult for a malicious node to join a TGrp. A stringent screening function is applied by the member nodes before admitting a new node into their TGrp. Still, it is possible for malicious nodes to join TGrps by providing good service early on till they get admitted. On joining a TGrp they can then provide bad service and/or send false negative information about good nodes. Therefore, it is important to identify and remove such nodes from a TGrp. This removal process is also termed as *eviction*.

A member node is evicted only if it provides bad service several times, and its reputation goes below zero as viewed by the majority of its TGrp members. This is because peers trust and give each other benefit of doubt even if complaints are received regarding each other. Several external factors can cause nodes to provide bad service, like broken ISP connection, network congestion, unintended virus uploads etc. A member node

might also possibly be a victim of false negative information propagated against it by malicious nodes. Thus, member nodes are given a lot of leeway before being evicted from a TGrp.

Evicted nodes are put in Bad_i by all the peer nodes. In addition, a node's eviction is made known to as many other nodes as possible. Since eviction signals negative information about a node, propagation of this information follows the same rule as that for normal complaints.

C. Trust Group Membership Validation

It is important for nodes to be able to correctly prove their credentials regarding their TGrps. Moreover, malicious nodes should not be able to fake their membership to some highly reputable TGrp. Furthermore, as nodes join and leave a TGrp, it should be possible to seamlessly update TGrp membership information.

TGrp members create an *affiliation certificate* that include the IP addresses of all the nodes that comprise the TGrp and successively encrypted by each node's private key. For example, say nodes *1*, *2*, and *3* form a TGrp. The IP addresses and private keys of nodes are represented by IP_i and E_i, respectively ($\forall i \in \{1,2,3\}$). The following is the affiliation certificate for this TGrp: $E_3(E_2(E_1(IP_1, IP_2, IP_3)))$. One can obtain the public keys of *1*, *2*, and *3* and verify that they all belong to the same TGrp. The affiliation certificate also contains a time stamp such that old certificates cannot be used by (evicted) nodes to falsely claim membership to a TGrp.

A node in order to prove its membership to a TGrp can include the affiliation certificate (containing its IP address as well as that of the TGrp members) when communicating with other nodes. Other nodes on receiving the affiliation certificate can update (if they do not already have) their information about the TGrp. When a TGrp member is evicted, the remaining member nodes create a new affiliation certificate and include it in their message for propagating the information that a node has been evicted. This enables TGrp membership information to be updated by a large fraction of the nodes that receive the new affiliation certificate. Likewise, when a new node joins a TGrp, the updated affiliation certificate includes information about the new node.

It is assumed that joins and leaves from a TGrp occur one node at a time, and that the reputation (and also the reputation counter value) of the old TGrp is inherited by the new TGrp. More elaborate mechanisms are needed to handle scenarios involving concurrent joins and leaves. For example, the case when a TGrp disintegrates into multiple TGrps each consisting of a large subset of the original set of member nodes and/or when multiple TGrps fuse into a single TGrp. The study of such mechanisms are part of our future work.

V. Reputation Management Framework

Owing to the difficulties caused by nodes' selfishness, we define the following goals for our reputation management framework. With high probability, nodes with higher reputation should get better services, i.e. nodes which provide good service should get good service in return. Moreover, malicious nodes should not be able to fake high reputation and lower the reputation of good nodes in any significant manner, i.e. good nodes should be regarded as good and bad nodes as bad by all good nodes in a system. The chances of a node being identified correctly is directly proportional to how good or bad it is, i.e. its service probability. These goals are summarized below.

- $Good \not\Rightarrow Bad_i \; \forall i \in Good$, i.e. good nodes should not be viewed as bad by other good nodes.
- $Bad \not\Rightarrow Good_i \; \forall i \in Good$, i.e. bad nodes should not be viewed as good by good nodes.
- $Good \Rightarrow Good_i, Bad \Rightarrow Bad_i \; \forall i \in Good$, i.e. good nodes should be viewed as good and bad nodes should be viewed as bad only by good nodes.

It must be noted that these goals cannot be perfectly achieved in a large and dynamic P2P system with imprecise information. Therefore, the proposed framework strives to achieve these goals with high probability only. In our discussion, all services are considered at par with each other. Thus, the reputation of a service provider depend on the clients it serve and not on the category of service it provides. Similarly, all bad services are indistinguishable from each other, i.e. the reputation of a service provider is reduced based on the reputation of a client that sends a complaint against it.

A. Reputation Representation and Calculation

A node computes the reputation of every other TGrp that it is aware of. Since the reputation of a node is simply the reputation of its TGrp divided by the number of nodes in that TGrp, we only describe how the reputation of a TGrp is estimated. Nodes that are not part of any TGrp are also considered as TGrps comprising of a single member node. In addition to maintaining the reputation of other TGrps, a node also maintains the reputation of its peers. This is important so as to determine when to add a new member or to remove an (possibly malicious) existing one from the TGrp.

Since the information available at nodes vary, nodes may have different views or reputation of the same TGrp. The consistency in views is proportional to the service probabilities of the member nodes of the target TGrp. A very active TGrp whose member nodes provide service often, is considered good by a large fraction of the nodes. Whereas, a TGrp whose member nodes provide service infrequently, is viewed as good by only a few and unknown by a large fraction of the nodes. Thus, reputation of a TGrp and service probabilities of its members are directly related to each other.

The reputation of a TGrp is always a value between -1 and 1. Higher value corresponds to higher reputation, and vice versa. More precisely, good nodes have positive reputation ratings and bad nodes have negative ratings, and value zero is assigned to nodes in set ϕ. The reputation value is updated after every time period τ, which is a well-defined system parameter. τ can be either timer-driven or event-driven, for example, based on the number of requests observed by a node. A TGrp's reputation is dependent on two factors - number of times its members provided service (and to whom) out of the total service instances in the current time period, and its reputation at the end of the last time period. For illustration, we consider an observer node *o* and see how it updates its reputation of other TGrps (and also

its peers). Before that we define some definitions that are useful in our subsequent discussion.

Definition 1: Network neighbors of *o* denoted by N_o, are the set of neighboring nodes as governed by the overlay routing substrate. The neighboring nodes are utilized for carrying out resource lookups.

Definition 2: Network view of *o*, represented by Γ_o, is the set of all nodes that *o* is aware of. This set will include, *o*'s own ID, its network neighbors, its TGrp members, and other nodes that it becomes aware of on receiving messages containing satisfaction certificates and complaints. For simplicity, we assume that the size of this set can only grow as *o* learns about new nodes. In a dynamic system, it is possible for both network neighbors and TGrp members to change. However, as long as these nodes are still part of the network, it is useful to include them in one's network view so as to be able to send satisfaction certificates to them in future.

For each TGrp, *o* maintains a reputation counter C_{oiTGrp}, $\forall i, iTGrp \subseteq \Gamma_o$. C_{oiTGrp} is initialized to zero at the start of every time interval. The counter is increased (decreased) whenever a member of $iTGrp$ provides good (bad) service. Since one gets higher reward for serving more reputable nodes, the increase (decrease) in the counter value is proportional to the reputation of the node being served (cheated). More precisely, if *i* serves *j*, where $i, j \in \Gamma_o$ and $iTGrp \neq jTGrp$, *o* updates *iTGrp*'s reputation counter as follows.

$$C_{oiTGrp} = 1 + C_{oiTGrp} + C_{ojTGrp}/|jTGrp| \tag{2}$$

At the same time *jTGrp*'s counter is decreased by C_S, which is the cost that the network incurs due to the service transaction and is charged to *jTGrp*. Therefore, we have,

$$C_{ojTGrp} = C_{ojTGrp} - C_S \tag{3}$$

If *i* cheats *j*, the counters for both *iTGrp* and *jTGrp* are decremented as follows.

$$\begin{aligned} C_{oiTGrp} &= C_{oiTGrp} - C_{ojTGrp}/|jTGrp| \\ C_{ojTGrp} &= C_{ojTGrp} - C_{oiTGrp}/|iTGrp| \end{aligned} \tag{4}$$

For example, say there are three nodes *a*, *b* and *c* in Γ_o, that belong to TGrps *aTGrp*, *bTGrp*, and *cTGrp*, respectively. Let the cardinality of all the three TGrps be one. The counter values C_{oaTGrp}, C_{obTGrp}, and C_{ocTGrp} are initially set to zero. *o* first receives a satisfaction certificate stating that *a* provided service to *b*. As a result, C_{oaTGrp} is updated to contain value 1. Subsequently, if *c* provides service to *a* then using Equation 2, C_{ocTGrp} is updated to value 2 (=1+1). Now, if *b* is a bad node and sends a complaint against *c*, C_{obTGrp} is reduced to -2, but C_{ocTGrp} remains unchanged. After these three messages, *a* and *c* are put in $Good_o$, and *b* in Bad_o, respectively. In other words, *o* views *a* and *c* as good, and *b* as a bad node.

As can be seen, if *o* receives a complaint message that *i* cheated *j*, it decrements the reputation counters for both *iTGrp* and *jTGrp*. This is because complaints can be falsely initiated and it is difficult for *o* to ascertain their validity. However, it is known for sure that one of *i* or *j* acted maliciously and therefore the counters for both the TGrps are reduced in proportion to the others' current value. This simple rule ensures that malicious nodes suffer more (incur a larger reduction in counter values) than non-malicious nodes. This is because non-malicious or good nodes provide service to others and typically have higher counter values. This also prevent good nodes from falsely propagating negative information against other (good) nodes. Here we assume that of the two nodes indicated in any complaint, one belongs to $Good$ and the other to Bad.

At the end of the current time interval, *o* calculates the reputation of all the TGrps in Γ_o using the following equation.

$$R^{curr}_{oiTGrp} = \frac{C_{oiTGrp}}{\Sigma_{jTGrp \subseteq \Gamma_o} C_{ojTGrp}}, \forall iTGrp \subseteq \Gamma_o \tag{5}$$

The denominator in Equation 5 represents the sum of counter values of all the TGrps that *o* is aware of. The reputation (R^{curr}_{oiTGrp}) of TGrp, *iTGrp*, obtained above for the current time interval is combined with the TGrp's reputation at the end of the previous interval (represented by R^{prev}_{oiTGrp}) to obtain its new reputation value (R_{oiTGrp}) as shown below.

$$R_{oiTGrp} = \zeta * R^{curr}_{oiTGrp} + (1 - \zeta) * R^{prev}_{oiTGrp} \tag{6}$$

Here, ζ is the importance given to the current performance of a TGrp as opposed to its past performance for estimating its reputation. In general, when it is not important (or is implied) to specify the observer node (here *o*), the reputation of *iTGrp* is simply written as R_{iTGrp}. The reputation of any node, *i*, represented by R_{oi}, is given as $R_{iTGrp}/|R_{iTGrp}|$. Again, when it is not important (or is implied) to specify the observer node, the reputation of *i* is simply written as R_i.

At the end of every time interval, nodes with reputation values below zero are put in Bad_o. In other words, if a TGrp has a reputation value of less than zero, then all the member nodes are considered as malicious and put in Bad_o.

In the above, we have shown how the reputation of different TGrps (including one's own) is calculated by *o*. In addition, *o* calculates the reputation of each of its peers by maintaining a counter C_{oi} for each of them, $\forall i, i \in oTGrp$. These counters are handled similarly to those for the TGrps; their exact updation mechanism is given in the next section. Peer reputation values are given by the following equations, which are analogous to the ones used above for TGrps' reputation calculation.

$$R^{curr}_{oi} = \frac{C_{oi}}{\Sigma_{jTGrp \subseteq \Gamma_o} C_{ojTGrp}} \tag{7}$$

$$R_{oi} = \zeta * R^{curr}_{oi} + (1 - \zeta) * R^{prev}_{oi} \tag{8}$$

B. Reputation Counter Updation Algorithm

The algorithm presented here is used by good nodes to update the reputation counters for different TGrps as well as their peers; the reputation values based on these counters are calculated at the end of every time period of length τ as explained in Section V-A. As before $Good_o$, Bad_o, and ϕ_o represent the set of good, bad, and unknown nodes, respectively, as viewed by an observer node *o*.

The underlying principle of the algorithm is that a TGrp's reputation is dependent on the reputation of the TGrp(s) it has served (or cheated). Also, reputation ratings are increased by non-TGrp members only if the service provider and receiver

belong to different TGrps. We divide the algorithm into two categories - for dealing with positive and negative information, respectively.[2]

Category 1: Node *o* receives a message (M_{ab}) that *a* provided service to *b*. Note that this message is originated by *a* and contains the satisfaction certificate given to it by *b*. We use the following notation to describe this event, $o : a \rightarrow b$. Based on this information, node *o* takes an appropriate action as outlined in Figure 1.

1) **if** $(b \in Bad_o) \vee (oTGrp \neq aTGrp = bTGrp)$
2) **then**
3) return;
4) **if** $(oTGrp = aTGrp = bTGrp)$
5) **then**
6) $C_{oa} \leftarrow 1 + C_{oa} + C_{ob}$
7) $C_{ob} \leftarrow C_{ob} - C_S$
8) return;
9) **if** $(oTGrp = aTGrp \neq bTGrp)$
10) **then**
11) $C_{oa} \leftarrow 1 + C_{oa} + C_{obTGrp}/|bTGrp|$
12) $C_{obTGrp} \leftarrow C_{obTGrp} - C_S$
13) send M_{ab} to $i, \forall i \in \Gamma_o$
14) return;
15) **if** $(oTGrp = bTGrp \neq aTGrp)$
16) **then**
17) $C_{oaTGrp} \leftarrow 1 + C_{oaTGrp} + C_{ob}$
18) $C_{ob} \leftarrow C_{ob} - C_S$
19) return;
20) **if** $(oTGrp \neq bTGrp \neq aTGrp)$
21) **then**
22) $C_{oaTGrp} \leftarrow 1 + C_{oaTGrp} + C_{obTGrp}/|bTGrp|$
23) $C_{obTGrp} \leftarrow C_{obTGrp} - C_S$
24) return;

Fig. 1. Reputation counter updation upon receiving a satisfaction certificate

Category 2: Node *o* receives a message (M_{ab}) that *a* cheated *b*. Note that this message is originated and signed by *b*. We use the following notation to describe this event, $o : a \nrightarrow b$. Based on this information, node *o* takes an appropriate action as outlined in Figure 2.

Figure 1 is self-explanatory, so we focus on Figure 2. α, β, γ (where $0 \leq \gamma \leq \beta \leq \alpha \leq 1$) limit the reduction in reputation counter values for one's TGrp members. Peers are given benefit of doubt even if bad service is received from them. This is because nodes trust their TGrp members to be good and attribute their malicious behavior to some external factor as was described in Section IV-B. For example, in Steps 3 and 4, *o* minimizes the reduction in reputation counter of its peer node by a factor, α (which is less than 1). Likewise, if a member node receives a complaint involving two of its peers, the reputation counters for both are reduced, but scaled down by a factor, β, as shown in Step 2. The value of β is typically less than α because it is not known for certain which of the two nodes is really a malicious node. Thus, a low value of β minimizes wrongfully penalizing a good peer node. Moreover, there is a possibility that none of the two nodes are malicious and poor

[2]In the reputation counter updation mechanism presented in Figures 1 and case2, if a counter value on the right-hand side of an assignment statement is negative, it is set to zero.

1) **if** $(b \in Bad_o)$
2) **then**
3) return;
4) **if** $(oTGrp = aTGrp = bTGrp) \wedge (o \neq b) \wedge (o \neq a)$
5) **then**
6) $C_{oa} \leftarrow C_{oa} - C_{ob} * \beta$
7) $C_{ob} \leftarrow C_{ob} - C_{oa} * \beta$
8) return;
8) **if** $(oTGrp = aTGrp = bTGrp) \wedge (o = b)$
9) **then**
10) $C_{oa} \leftarrow C_{oa} - C_{ob} * \alpha$
11) return;
12)**if** $(oTGrp = aTGrp = bTGrp) \wedge (o = a)$
13)**then**
14) $C_{ob} \leftarrow C_{ob} - C_{oa} * \alpha$
15) return;
16)**if** $(oTGrp = aTGrp \neq bTGrp)$
17)**then**
18) $C_{oa} \leftarrow C_{oa} - (C_{obTGrp}/|bTGrp|) * \gamma$
19) $C_{obTGrp} \leftarrow C_{obTGrp} - C_{oa}$
20) return;
21)**if** $(oTGrp = bTGrp \neq aTGrp)$
22)**then**
23) $C_{ob} \leftarrow C_{ob} - (C_{oaTGrp}/|aTGrp|) * \gamma$
24) $C_{oaTGrp} \leftarrow C_{oaTGrp} - C_{ob}$
25) return;
26)**if** $(oTGrp \neq aTGrp \neq bTGrp)$
27)**then**
28) $C_{obTGrp} \leftarrow C_{obTGrp} - C_{oaTGrp}/|aTGrp|$
29) $C_{oaTGrp} \leftarrow C_{oaTGrp} - C_{obTGrp}/|bTGrp|$
30) send M_{ab} to $i, \forall i \in \Gamma_o$
31) return;
32)**if** $(oTGrp \neq aTGrp = bTGrp)$
33)**then**
34) return; /* do nothing - both a and b belong to the same TGrp. Not much can be derived from this information */

Fig. 2. Reputation counter updation upon receiving a complaint

service occurred because of some external factor. The value γ in Steps 5 and 6 represents the fact that nodes trust their peers more than they trust someone outside their TGrp. Thus, a low value of γ makes it difficult for bad nodes to cause a node to be evicted from a TGrp by propagating false negative information against it.

VI. Framework Evaluation

We have evaluated our proposed reputation management framework using extensive simulations and found that it satisfies the requirements for using reputation as a form of currency, and is robust against possible attacks by malicious nodes. Simulation results show that reputation and extent of awareness about TGrps is directly proportional to the service probabilities of their respective member nodes. Moreover, with time all bad nodes are detected and isolated, thus limiting the amount of bad service they can provide to good nodes.

A. Simulation Setup

The network contains a specified number of good and bad nodes. Good nodes are differentiated based on their service probabilities, and never provide bad service when selected for a

Network size	300
α	0.3
β	0.3
γ	0.1
ζ	0.1
C_S	0
Simulation rounds	100

TABLE I

DEFAULT VALUES OF THE PARAMETERS USED IN SIMULATIONS.

transaction. The goal of bad nodes is to maximize the instances of bad service in the network, i.e. their intent is to get selected for a transaction and then provide bad service. Moreover, they may fake high reputation to increase their priority and prevent legitimate requests from being serviced. Furthermore, bad nodes may propagate false negative information against a target node to reduce its reputation and cause it to be evicted from its TGrp. Evicted nodes are considered as bad and are given a very low reputation value (say -1) by the peers and others that learn about the node being evicted. We refer to such attacks as the denial-of-reputation (DoR) attacks and the only way a good node's reputation can appreciably be decreased is by causing it to be evicted from its TGrp.

In our simulations, we do not penalize nodes for accessing services, i.e. C_S in Equation 3 is set to zero and a node's reputation is unaffected if it accesses service. This is based on the assumption that serving a request is much more expensive than receiving a request and an overloaded node can simply ignore a request if it is unable to serve it.

Nodes have fixed number of network neighbors equal to *O(log N)*, where *N* is the network size.[3] All good nodes belong to one of the specified service categories, i.e. they originate and serve requests only related to that service category. The total number of service categories is dependent on the network size and in all simulations it is set to *N/15*. We also ran simulations with the number of service categories equal to *N/10* and obtained similar results as presented here. Good nodes configure themselves into various TGrps and no two member nodes in a TGrp belong to the same service category. Bad nodes can also join TGrps in order to subsequently provide bad service and/or target good nodes.

To keep our framework general and independent of any specific routing protocol, we assume that all nodes belonging to a requested service category are equally likely to be selected by a client. The probability of selection is governed by the service probabilities (i.e. reputation) of the candidate nodes. Moreover, bad nodes can intercept requests and claim to provide service, even if they are not capable of doing so. The probability of interception is directly proportional to the number of bad nodes.

Unless otherwise stated, Table I gives the value of various parameters used in our simulations. We divide the total simulation time into multiple simulation rounds. In every round, each node initiates a single request that can be satisfied by any of the potential service providers. The limit of one request per node per simulation round simplifies the handling of scenarios where bad nodes repeatedly send out messages to enhance each others' reputation. Intuitively, if one receives multiple service transaction information initiated by the same node in a short span of time, then this information is discarded.

The time interval for reputation updation, as defined in Section V-A, is equal to one simulation round. This was adopted so as to simplify the simulation code and any other timer value could also be used without altering the qualitative nature of the results.

In our simulations, since we want to evaluate the effectiveness of TGrps and not worry about how they are formed, we assume the TGrps as given with all the good nodes belonging to one of them. Initially, the bad nodes are not part of any TGrp, however, they can join one if they provide sufficiently good service over a period of time. This scenario is valid in real-world also, as malicious nodes usually do not join a network in the early stages of its formation. Usually it is much easier for malicious nodes to operate when the network has reached a certain critical size.

Now we demonstrate the effectiveness of the proposed framework in identifying and isolating malicious nodes, and in minimizing the instances of bad service in the network.

B. Attack Models

To take into account different possible strategies of the bad nodes, we examine several attack models in this section. These attack models reflect the main requirements outlined in Section I for using reputation as a substitute for currency, and also indicate the robustness of the proposed framework in dealing with nodes' selfishness and maliciousness.

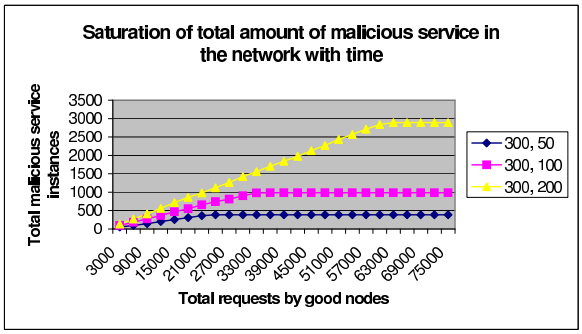

Fig. 3. Even when there are large number of malicious nodes in the system, the percentage of bad service in the network continuously decrease as the number of requests increase. Bad nodes are identified and not selected for future service transactions. The simulation results are for a network with 300 good nodes, with increasingly higher number of bad nodes. For example, (300, 50) denotes the network configuration with 300 good nodes and 50 bad nodes.

1) Attack Model A.: Malicious nodes always provide bad service when selected for a transaction. The probability of a bad node selection is directly proportional to the number of bad nodes in the system: In this case, the reputation ratings of bad nodes remain very low and thus they are not able to join any TGrp. Since negative information is readily propagated, all bad nodes are rapidly identified and put in Bad_i set by all the good nodes. Since the nodes in Bad_i are not selected for service transaction, the affect of bad nodes on good nodes is minimized and the network soon continues to operate as if there are no malicious nodes.

[3]This is typically the neighborhood size in several proposed P2P architecture, such as Chord [1] and CAN [2].

As shown in Figure 3, the number of bad service instances saturate to some maximum value after only a few simulation rounds. As expected, the time to reach this maximum value is proportional to the fraction of bad nodes in the system.

bad−service : instances of bad service provided by a bad node
good−service : instances of good service provided by a bad node

Network size		Zeta → 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
150	bad−service	14	8	5	5	3	3	3	3	2	2
	good−service	2	2	2	2	2	2	2	2	2	2
300	bad−service	10	5	3	2	4	4	4	3	3	1
	good−service	4	4	4	4	5	5	5	5	5	4
450	bad−service	22	10	6	4	4	3	3	3	3	1
	good−service	4	4	4	4	4	4	4	4	4	3

Fig. 4. With increasing ζ, the amount of good service provided by a malicious node to join a TGrp exceeds the amount of bad service it can provide before it is evicted from a TGrp. *Network size* refers to the total number of nodes in the system.

2) Attack Model B.: Malicious nodes first provide good service (with service probability equal to unity) and try to join some TGrp(s). Upon joining a TGrp, they always provide bad service: The intent of bad nodes here is to utilize the fact that the reputation of a node is dependent on the reputation of its TGrp. By joining reputable TGrps, malicious nodes attempt to maximize the instances of bad service in the network. This is because bad nodes are not put in Bad_i, as long as good service by their peers offset their bad service. As a result, bad nodes despite their poor service record continue to get selected for service transactions.

As described in Section IV-B, the reputation of a member node that provide bad service is gradually reduced and is eventually evicted when its reputation goes below zero as viewed by the majority of its peers. Moreover, re-entry of an evicted node into the same or any other TGrp requires higher level of service than what was required of it previously. Therefore, we only consider the damage that a bad node can cause by joining a TGrp for the first time.

We ran simulations to see the total number of bad service instances that a malicious node can provide upon entering a TGrp and before it is evicted. As shown in Figure 4, the amount of bad service that a malicious node can provide, before it is evicted, is greatly influenced by the value of ζ. For small values of ζ, the number of bad service instances far exceed the number of good service instances. But as ζ is increased, the reverse is true and this attack model becomes less and less attractive for the bad nodes.[4] These results are as expected, since higher ζ means that greater importance is given to current as compared to past performance. Therefore, malicious nodes cannot rely on past reputation to continue providing bad service and still be part of a TGrp.

We believe that it is difficult to completely avoid the damage caused by this attack model. This is due to the premise on which the notion of TGrps is based. TGrp members trust each other to be good and give each other benefit of doubt even if they provide bad service. However, in all our simulation runs, bad nodes are eventually identified and evicted.

[4]Bad nodes need to provide good service to be able to join a TGrp.

3) Attack Model C.: Malicious nodes launch DoR attacks to reduce the target nodes' reputation and cause them to be evicted from their TGrps: Although these attacks can be successful, it requires several bad nodes to provide sufficient good service to cause a target good node to be evicted. This partly defeats the goal of malicious nodes to maximize the instances of bad service in the network. We ran simulations with network size equal to 150, 300, and 450 and in each configuration successively varied the number of bad nodes. All the bad nodes target a randomly chosen good node and spread false negative information against it in each round. Also, in order to earn high reputation, bad nodes provide good service whenever they are selected for a service transaction.

round # : Simulation round number in which the target node is evicted
good−service : instances of good service provided by bad nodes (till the target node is evicted) in order to earn high reputation

Network size		|Bad| → 5	10	15	20	25	30	35	40	45	50
150	round #	–	–	–	34	27	25	23	22	22	22
	good−service	–	–	–	240	276	313	330	374	470	490
300	round #	–	–	–	–	32	31	29	27	25	24
	good−service	–	–	–	–	263	298	314	322	377	384
450	round #	–	–	–	–	38	37	29	28	28	27
	good−service	–	–	–	–	298	323	361	375	403	459

Fig. 5. With the increase in the number of bad nodes, the target node is evicted in an earlier round. At the same time, however, the total instances of good service provided by bad nodes also increase. Entries left blank indicate that bad nodes were unable to evict the target node, i.e. DoR attack failed. *Network size* refers to the total number of nodes ($|Good| + |Bad|$) in the system.

As shown in Figure 5, in all the simulation runs, at least 20-30 bad nodes were required to cause a target node to be evicted. This is expected because peers of the target node scale down the reputation counter values of the complaining nodes by a factor, γ (=0.1). Thus, large number of complaints are needed before the reputation of the target node goes below zero as viewed by its peers. With higher number of bad nodes the target node is quickly evicted, i.e. in an earlier round, but the total instances of good service provided by the bad nodes is also increased.

4) Attack Model D.: Bad nodes split themselves into two groups - $Bad1$ and $Bad2$. Nodes in $Bad1$ always provide good service and nodes in $Bad2$ always provide bad service. $Bad1$ nodes assist $Bad2$ nodes to increase their reputation: This attack model attempts to maximize the number of bad service instances by increasing the probability that $Bad2$ nodes are selected for service transactions. $Bad1$ nodes initially provide good service in order to be accepted into some TGrp. After joining a TGrp, they do not serve good nodes and only propagate positive information about $Bad2$ nodes. Since $Bad1$ nodes never provide bad service and because C_S is set to zero in our simulations, $Bad1$ nodes never get evicted from their TGrp(s). $Bad2$ nodes on the other hand use the increased reputation to get selected for service transactions and provide bad service.

To evaluate the effectiveness of this attack model, we compare it with attack model *A* to see if $Bad1$ and $Bad2$ nodes together are able to increase the instances of bad services beyond what is possible in attack model *A*. For this we define a metric, *net maliciousness*, which is the difference between the number of bad service instances provided by $Bad2$ nodes and good service

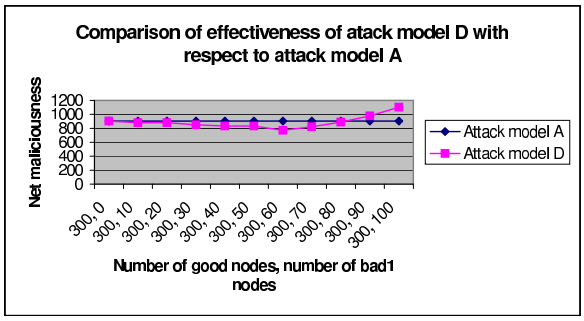

Fig. 6. Comparison of attack models *A* and *D* when the number of good nodes are 300 (i.e. $|Good| = 300$) and the number of $Bad2$ nodes are 100 ($|Bad2| = 100$). The number of $Bad1$ nodes vary from 0 to 100 in attack model *D*, whereas they remain equal to 0 in attack model *A*.

instances provided by $Bad1$ nodes.

It is clear from the results in Figure 6 that the value of net maliciousness in attack model *D* is less than that in attack model *A*, even when there are significantly large number of $Bad1$ nodes. For experimentation, in attack model *A*, we always set the number of $Bad1$ nodes to zero, while in attack model *D*, we successively increase the number of $Bad1$ nodes to see how it increases the probability of $Bad2$ nodes getting selected for service transactions. When the number of $Bad1$ nodes increase relative to the number of good nodes, $Bad1$ nodes do not have to provide as much good service to get accepted into some TGrp. This explains higher net maliciousness achieved by attack model *D* as compared to attack model *A* when $|Bad1| \geq 90$. Similar results were obtained for other network sizes as well, and it appears that this attack strategy is not any stronger than the one adopted in attack model *A*.

|Bad|	10	30	50	70	90	110	130	150
round #	43	46	45	47	47	51	53	53

Fig. 7. Illustration of the number of rounds in which the bad nodes are identified as bad by the good nodes. The number of good nodes are 300 and the number of bad nodes vary from 10 to 150. In every simulation round each bad node sends a satisfaction certificate with another randomly selected bad node as the service provider. The service probabilities of the bad nodes are set to unity so as to simulate their increased reputation, and hence higher probability of getting selected by good nodes for service transactions.

5) Attack Model E.: Malicious nodes send out false satisfaction certificates, indicating other malicious nodes as the service provider, in order to increase each other's reputation. In addition, they provide bad service when selected by good nodes for service transactions.: Malicious nodes can use the increased fake reputation to increase their priority and prevent other legitimate requests from being serviced. However, this attack strategy is naturally countered in our framework. An increase in the reputation of malicious nodes attract service requests from the good nodes. But since malicious nodes provide bad service to good nodes their reputation goes down. This decrease in reputation can be substantial depending on the value of parameter γ. In fact the simulation results confirmed this hypothesis, as it was found that all the malicious nodes were quickly identified as bad by the good nodes (see Figure 7). Interestingly, number of malicious nodes had little impact on the effectiveness of this attack model. This is primarily due to the limit imposed on the number of requests that a node can originate in any given round.

VII. Conclusion and Future Work

The most important contribution of the paper is to utilize the concept of TGrps for reputation management and use of reputation as currency in large-scale P2P networks. We find that even simple reputation updation rules, based on the notion of TGrps, are effective when there are large number of malicious nodes working in collusion to bring down the system.

Our model is easily extensible and flexible enough to be tuned as per the requirements of a specific system. The proposed framework is scalable to large P2P systems and enables reputation computation in the face of nodes' selfishness, which can cause wrong or no service information to be propagated.

In future we would extend the proposed framework to handle scenarios when nodes can simultaneously be part of multiple TGrps.

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. *In Proceedings of the ACM SIGCOMM Conference*, 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *In Proceedings of ACM SIGCOMM San Diego*, 2001).

[3] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. *In ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2003.

[4] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Data and Knowledge Engineering, 15(4):840–854*, July/August 2003.

[5] S. Marti, and H. Garcia-Molina. Limited reputation sharing in P2P systems. *In Proc. of the 5th ACM conference on Electronic commerce, New York, NY, USA*, 2004.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. EigenRep: Reputation Management in P2P Networks. *In Proc. of ACM World Wide Web Conference, Budapest, Hungary*, May 2003.

[7] T. Moreton, and A. Twigg. Enforcing collaboration in P2P routing services. *In First International Conference on Trust Management*, 2003.

[8] Z. Diamadi, and M. J. Fischer. A Simple Game for the Study of Trust in Distributed Systems. *Appeared in Wuhan University Journal of Natural Sciences*, 2001.

[9] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. *In Proc. of the Tenth International Conference on Information and Knowledge Management (CIKM '01), pages 310-317*, 2001.

[10] S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative Peer Groups in NICE. *In Proc. of INFOCOM 20003*, 2003.

[11] N. Nisan. Algorithms for Selfish Agents: Mechanism Design for Distributed Computation. *In Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, volume 1563, Springer, Berlin, pages 1-17*, 1999.

[12] A. Farag, and M. Muthucumaru. Evolving and Managing Trust in Grid Computing Systems. *In Proc. of the 2002 IEEE Canadian Conference on Electrical and Computer Engineering*, 2002.