

TwoHop: Metric-based Trust Evaluation for Peer-to-Peer Collaboration Environments

Dimitris Glynos*, Patroklos Argyroudis†, Christos Douligeris* and Donal O’Mahony†

*Dept. of Informatics, University of Piraeus, Karaoli & Dimitriou 80, Piraeus 18535, Greece

email: {daglyn, cdoulig}@unipi.gr

† CTVR, Dept. of Computer Science, University of Dublin, Trinity College, College Green, Dublin 2, Ireland

email: {argp, omahony}@cs.tcd.ie

Abstract—Communications are increasingly relying on peer-to-peer models of interaction in which all participating entities have the same level of authority. Such models allow developed systems to overcome single points of failure, since there is no central management of security parameters. In order to select trustworthy service providers, peers in such environments require collaborative metrics that build trust relationships based on evidence of previous interactions. In this paper, we present a collaborative metric and an algorithm for the evaluation of trust in peer-to-peer networks called *TwoHop*. Our proposal requires no central management and scales to accommodate the fundamental need for growth that characterises peer-to-peer systems. Our experimental analysis and simulation results indicate the robustness of our design against dishonest participants and its ability to flexibly accommodate the needs of majorities.

I. INTRODUCTION

According to the peer-to-peer communication model nodes that exist on the edges of the network become empowered with the functionality of utilising and, more importantly, of providing services to each other. In such computing environments the need for self-management becomes paramount, since there are no entities that can be universally trusted to provide service guarantees. Therefore, decentralised and collaborative algorithms are being developed in which all peers that participate in a network contribute to knowledge that can be used to assess service providers [1].

In this paper, we present a novel system called *TwoHop*, that utilises recommendations of peer nodes in order to evaluate the trustworthiness of service providers. We assume that there are direct relations between participants that have interacted in the past, and our goal is to establish indirect relations between unknown (i.e. without previous immediate interaction) participants. In a manner similar to previous work on the subject [1], [2], [3] we model the evaluation of indirect relations between *TwoHop* participants as a directed and weighted graph problem; vertices represent participants, edges represent relations and the weights on the edges represent the strength of the relation between two connected participants. *TwoHop* allows nodes to be directly connected via multiple edges, each edge representing a different type of relation. While establishing an indirect relation, the contribution of each edge towards the indirect relation weight depends on the edge type and the distance from the source vertex. As the name implies, nodes may build indirect relations only if they are within a 2-

hop radius. On this model we propose a metric that describes the quality of provided services as experienced by the querying participant’s neighbourhood of peers.

In the next section we present related work on the subject, and in section III the design of our system. Results are analysed in section IV and we conclude in section V.

II. RELATED WORK

The main characteristic of all collaborative trust evaluation metrics is *peer assertion*. Participants make assertions about each other certifying certain beliefs they have. Since all participants are allowed to express their opinion without the validation of a trusted third party (TTP), they are free to assert false facts with selfish or malicious intent. The main goal of trust metrics is to be resistant to such attacks, even when high percentages of dishonest participants collude.

One of the first simple and successful trust metric systems was the one employed by eBay. After a transaction, participants make integer statements of -1, 0, or 1 for each other. These are collected, added, associated with a user identity and made publicly available to all users. One of the disadvantages of the eBay system when applied to peer-to-peer environments is that it relies on a centralised infrastructure to collect and publish user ratings. Furthermore, *karma suicide* attacks have been particularly effective against it. In karma suicide attacks malicious participants behave correctly for a large number of small value transactions and sacrifice their rating by stealing in a single high value exchange.

In Pretty Good Privacy (PGP) each user maintains a key ring that contains the keys and the associated identifiers (names and email addresses in this case) of other users [4]. When a key is entered in the key ring the owner must decide on a trust value for the new key. The trust value is basically a rating on how much the key ring owner trusts PGP key certificates issued by this new key. The lower the trust value the more key certificates are necessary to validate a given key. The key ring and the associated trust values stay local to each owner. This essentially means that the PGP trust evaluation algorithm does not provide a way to key ring owners to compute trust values for the keys of unknown users.

The Advogato trust metric [3] operates on a directed graph representing participants as nodes and evaluations as edges. Given a set of trusted nodes, the final trust value of a target

node is the probability that a random walk will end at the target. The main assumptions of this approach are that the querying node has full knowledge of the trust graph and that all the participating nodes have the same in-degree.

XRep was designed as an extension to P2PRep, a peer-to-peer reputation system where participants keep track, and share with others, information about the reputation of their peers [5]. It allows service requesters to assign reputation values to resources, as well as to providers. The P2PRep/XRep system does not have any notion of evaluating the peers that provide reputations. By treating all recommenders equally it fails to take advantage of existing trust structures. In contrast, our work in TwoHop exploits such established trust relationships, constructing a model which represents more truthfully the dynamics of real-world trust hierarchies.

In [2] Theodorakopoulos and Baras propose an algebraic method for decentralised trust evaluation in ad hoc networks. Edge weights in their system are tuples that contain a) the trust value of one participant towards another and b) the confidence value associated with the aforementioned trust value. To calculate indirect relations the authors define an operator (\otimes) to combine weights along a path, and another operator (\oplus) to combine weights across multiple paths. Using the theory of semirings it is then possible to evaluate the indirect relation between any two participants of the network, and to find the most trusted path among the existing paths between them. Unfortunately, the proposed trust model considers all edges being of the same type and thus fails to distinguish between a recommendation for a peer and an evaluation of a peer's service.

III. THE TWOHOP TRUST METRIC ARCHITECTURE

A. Design Principles

The design of the TwoHop trust metric architecture is governed by the following five basic principles:

▷ **2-hop recommendation chain limit:** Trust is not perfectly transitive, but degrades along a chain of recommenders. It has been previously analysed [6] that as the length of a recommendation chain increases, the semantics of trust change for all earlier participants of the chain. As stated in [7], assuming that such a degradation does not exist undermines the importance of a recommendation's source. We, therefore, introduce a limit on the depth of recommendation chains. A depth limit of two hops permits the creation of trust graphs that model peer interactions, where peers either communicate directly (1st hop neighbours) or receive recommendations about third parties (2nd hop neighbours) from peers that had direct communication with them in the past.

▷ **No global trust values:** Each peer keeps its own local *trust portfolio*, describing the trust associations it has built over time. Portfolios of other peers may be consulted, but their contents will be influenced by local weights. By exporting local trust information to third parties, a form of shared intelligence is realised that allows for the prevalence of community accepted evaluators and service providers.

▷ **Trust differentiation:** The TwoHop trust graph is a directed graph with multiple edges per node pair. Each edge represents a different type of trust relationship. In this way there is a clear distinction between the weight describing the quality of a node's service and the weight describing the quality of evaluations provided by the same node.

▷ **Network-wide trust metric:** TwoHop introduces a metric describing the trust towards a service, as experienced by a neighbourhood of peers. This trust metric is an algebraic aggregation of the relevant weights and evaluations found in portfolios of peers that are within a 2-hop radius from the querying node.

▷ **Multi-path contribution:** TwoHop does not ignore multiple recommendation paths. Specifically, if two or more evaluations for a provider are found they are appraised and contribute (unequally, for the details see paragraph III-D) to the trust calculation. If two or more paths are discovered that share the same service evaluator for a target service provider, the shorter one is kept. If they exhibit the same length then the one starting from the most trusted party is kept; the others are discarded.

B. Assumptions

TwoHop has been designed to operate on top of a peer-to-peer overlay network (such as Tapestry [8]) where each participating node is associated with a unique identifier. Message routing is transparent to the user of the overlay network and depends on the identifiers of the communicating nodes. In order for two peers to exchange messages, they must first come in contact in some way and exchange their respective node identifiers.

C. Definitions

Peers participating in a TwoHop system play the roles of the service *provider*, the service *evaluator*, the *judge* to service evaluators and that of the *reviewer* to evaluator collections. A peer may play more than one roles at a time depending on his activities during his presence in the overlay network.

A service *provider*, as the term suggests, is a peer offering a service to other peers. This service is associated with an identifier s , which is a publicly accepted description for this type of service. The format of s depends on the application.

A service *evaluator* is a peer that at one point has used a service offered by a service provider and has made an evaluation on the quality of this service. An evaluation record contains the triplet:

$$[i, s, v], i \in I, s \in S, v \in (0, 1),$$

where v are the actual evaluation points awarded to the service of type s offered by the service provider with peer identifier i , I is the set of all possible peer identifiers and S is the set of all possible service type identifiers.

A peer acting as a *judge* to a service evaluator, assesses the quality of evaluations made by this service evaluator, for a given service type. The assessment record is similar to the evaluation record and contains the triplet:

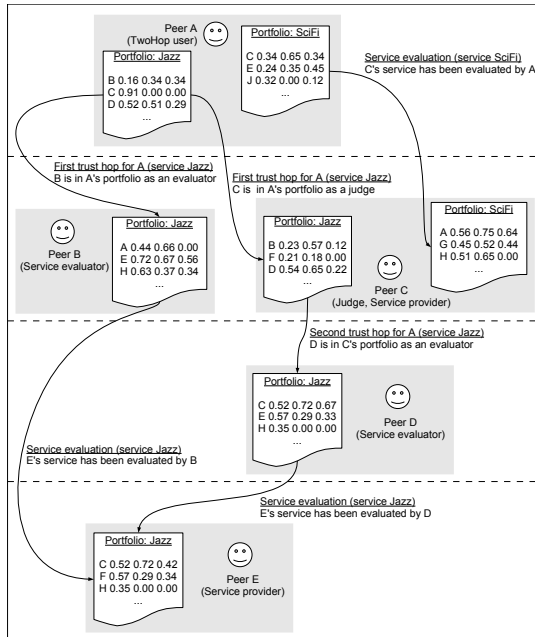


Fig. 1: Example of a TwoHop Trust Hierarchy.

$$[i, s, a], i \in I, s \in S, a \in (0, 1),$$

where a are the assessment points awarded to the service evaluator with peer identifier i when it was assessed on service evaluations of type s .

The evaluator collection *reviewer* examines the assessments made by a peer acting as a *judge* for a given service type. The *reviewer* awards points both for the quantity and the quality of these assessments. The review record contains the triplet:

$$[i, s, r], i \in I, s \in S, r \in (0, 1),$$

where r are the review points awarded to the peer with identifier i , for the assessments made on evaluations of service type s .

Each peer holds a collection C of *trust portfolios*, with one portfolio P_s per service type s . We define C as follows:

$$C = (P_j, \dots, P_k), j, k \in S, j \neq k$$

A trust portfolio contains the combined information coming from the evaluations, assessments and reviews performed by its owner, and is defined as:

$$P_s = \{(i_0, r_0, a_0, v_0), \dots, (i_k, r_k, a_k, v_k)\}, k \in \mathbb{N}^*, s \in S,$$

where v_k is the evaluation for the service of type s offered by peer i_k , a_k is the assessment for peer i_k 's evaluations and r_k are the review points peer i_k received for assessments on service type s . If any one of these values (r , a , v) is not available, it is initialised to 0. The method by which r , a and v are updated is beyond the scope of this paper.

Fig. 1 illustrates a high-level example, presenting the concepts explained so far and their relations.

D. Trust Metric Calculation Algorithm

To calculate the network-wide trust value for a service, TwoHop utilises the following three functions:

- `getPortfolio(i, s)`: Retrieves the portfolio for service type s from the remote peer identified as i .
- `peerExists(P, i)`: Returns TRUE if an entry for peer i exists in portfolio P .
- `calculateTrust(P_{root}, i, s)`: Calculates the trust value for the service of type s , offered by the provider with peer id i , starting from portfolio P_{root} .

Since the implementation of the first two functions is straightforward, we will concentrate on the third one, namely `calculateTrust`, which forms the basis of the TwoHop system. The function's first argument P_{root} is a portfolio, which the system will use as a starting point for finding peers related to this type of service. Since all searching starts from this portfolio, we call it the *root trust portfolio*. This portfolio is usually the user's own portfolio, but this need not be the case. If the user has no entries in his portfolio for the requested service type, a portfolio of another peer (trusted entity) may be obtained via `getPortfolio` and used as P_{root} .

The first operation of `calculateTrust` is to scan P_{root} for entries referring to peers other than provider i (see Function 1). In this way, we avoid taking into account evaluations the provider may have made about his own service (which would probably be biased), evaluations made by other parties known by this provider (i.e. biased "friends" of this provider, mentioned in his portfolio), but also previous evaluations made by the owner of the P_{root} portfolio.

The portfolio (P_{onehop}) of each peer found in the remaining entries ($peer_0$) is then fetched (via `getPortfolio`) and serially examined. As the name P_{onehop} implies, we are examining node portfolios that are one hop away from the P_{root} owner. We are looking for two types of entries: a) evaluations of the provider's service and b) assessments regarding peers who are "experts" on the particular type of service. If an evaluation of the provider's service is found, its value is multiplied by the assessment ($assess_0$) linked to this one-hop peer ($peer_0$) in P_{root} . The resulting value is added to sum_{trust} , while the weight (i.e. $assess_0$) is added to $sum_{weights}$.

For any assessment found in P_{onehop} , we check (via `peerExists`) if the peer it is referring to is already part of P_{root} and if so we discard it at this point. In this way, we favour checking a peer directly at P_{root} level (using the P_{root} owner's previous assessment), rather than indirectly, through some other peer's portfolio. This also prevents us from considering a peer's portfolio twice (both at one-hop and two-hop levels), should P_{onehop} contain a reference to its owner.

Assessment entries in P_{onehop} are used as pointers to second-hop portfolios. For each of these portfolio's (P_{twohop}), we record the assessment ($assess_1$) linked to the portfolio owner as described in P_{onehop} . Since our system is bounded to 2-hop searching, each P_{twohop} is fetched and scanned just for evaluation records of the given provider. If such a record is found, its value is multiplied by a weight product, which comes from $review_0$ (the root portfolio owner's review of the peer that suggested this 2-hop node) and $assess_1$. The weight product is also recorded in $sum_{weights}$.

If `calculateTrust` reaches a certain evaluation made by a peer through more than one paths, it records the evaluation value only once, favouring the path that started with the peer trusted most (i.e. highest $review_0$ in P_{root}).

The trust value T returned by `calculateTrust` is the weighted average of the collected evaluations, i.e.:

$$T = \frac{sum_{trust}}{sum_{weights}} = \frac{\sum_{j=1}^{|V_{i,s}|} w_j v_j}{\sum_{j=1}^{|V_{i,s}|} w_j},$$

$$w_j = \begin{cases} r_j a_j & \text{if } v_j \text{ was collected from a second-hop peer} \\ a_j & \text{if } v_j \text{ was collected from a first-hop peer} \end{cases},$$

$$i \in I, s \in S, v_j \in V_{i,s}, r_j, a_j, v_j \in (0, 1), T \in (0, 1)$$

$V_{i,s}$ represents the set of collected evaluations for provider i offering service s . The weights r_j and a_j represent the review and assessment points associated with an evaluation v_j . If no evaluations were found, `calculateTrust` returns 0.

Service evaluations found in the P_{root} level do not influence the end result during the first stage of the algorithm (*Root Portfolio Loop*), but they may influence it later on. If a peer links back to the P_{root} owner, then the owner's evaluations will also be incorporated into the trust calculation, along with the respective weights of the consulted peers. This effectively means that any peer's evaluations are of value, only if other peers believe so too.

Finding an evaluation in P_{twohop} does not require sequential searching of all records in P_{twohop} . If such a record exists it will be the only record concerning the particular provider i and hence it could be fetched in $O(1)$ time by using an indexing scheme. If any given peer has an average of k items in his portfolio, then `calculateTrust` scans k entries in P_{root} and for each of these entries it scans k entries in P_{onehop} . This induces a $O(k^2)$ average time of execution, assuming that `peerExists` and `getPortfolio` complete in $O(1)$ time.

If all three portfolios (P_{root} , P_{onehop} , P_{twohop}) are in memory at once, the algorithm requires at least $3 \times \lambda \times k$ memory units to store these (where λ is the cost per portfolio entry, expressed in memory units) and therefore the average space requirements are $O(k)$. The number of items k in a peer's portfolio is naturally bounded by the number of peers a user trusts per service type. If a given portfolio reflects a peer's view of his social network, then the size z of this network places a bound on k , i.e. $k \in [0, z], k \in \mathbb{N}$. According to [9], $\ln z \in (1, 6)$, $z \in \mathbb{N}$, $\lceil \bar{z} \rceil = 154$.

In a trust network of a given service type with N peers, each having a portfolio size of k entries, the probability of finding a given peer α as an entry in a foreign portfolio within a 2-hop radius, is $P(\alpha) = 1 - (\frac{N-k}{N})^{k+k^2}$. Thus, for a large network of 1,000,000 peers and portfolio sizes of 150 entries, there is a 96% probability of successfully locating an entry for α .

IV. SIMULATION

For simulation purposes we have developed a prototype of TwoHop in Python. Each experimental result presented in this section is an average value coming from 5 simulations of a TwoHop network on a 3GHz Pentium IV host with 1GB RAM.

Function 1 calculateTrust

```

Require:  $P_{root} \neq \emptyset, i > 0, s > 0$ 
 $sum_{trust} \leftarrow 0, sum_{weights} \leftarrow 0$ 
 $old_{review_0} \leftarrow \emptyset, old_{assess_1} \leftarrow \emptyset$ 
for all  $entry_0 \in P_{root}$  do {Root Portfolio Loop}
   $peer_0 \leftarrow entry_0[peer\_id]$ 
   $assess_0 \leftarrow entry_0[assessment]$ 
   $review_0 \leftarrow entry_0[review]$ 
  if  $peer_0 = i$  then
    continue {Root Portfolio Loop}
  end if
   $P_{onehop} \leftarrow getPortfolio(peer_0, s)$ 
  for all  $entry_1 \in P_{onehop}$  do {1-Hop Loop}
    if  $entry_1[peer\_id] = i$  then
       $sum_{weights} \leftarrow sum_{weights} + assess_0$ 
       $sum_{trust} \leftarrow sum_{trust} + (assess_0 \times entry_1[evaluation])$ 
    else
       $peer_1 \leftarrow entry_1[peer\_id]$ 
       $assess_1 \leftarrow entry_1[assessment]$ 
      if  $peerExists(P_{root}, peer_1) = TRUE$  then
        continue {1-Hop Loop}
      end if
      if  $peer_1 \notin old_{review_0}$  then
         $old_{review_0} \leftarrow old_{review_0} \cup \{peer_1\}$ 
         $old_{review_0}[peer_1] \leftarrow 0$ 
         $old_{assess_1} \leftarrow old_{assess_1} \cup \{peer_1\}$ 
         $old_{assess_1}[peer_1] \leftarrow 0$ 
      end if
       $P_{twohop} \leftarrow getPortfolio(peer_1, s)$ 
      for all  $entry_2 \in P_{twohop}$  do {2-Hop Loop}
        if  $entry_2[peer\_id] = i$ 
          AND  $old_{review_0}[peer_1] < review_0$  then
             $sum_{weights} \leftarrow sum_{weights} + (review_0 \times assess_1) - (old_{review_0}[peer_1] \times old_{assess_1}[peer_1])$ 
             $sum_{trust} \leftarrow sum_{trust} + (review_0 \times assess_1 \times entry_2[evaluation]) - (old_{review_0}[peer_1] \times old_{assess_1}[peer_1] \times entry_2[evaluation])$ 
             $old_{review_0}[peer_1] \leftarrow review_0$ 
             $old_{assess_1}[peer_1] \leftarrow assess_1$ 
          end if
        end for{2-Hop Loop}
      end if
    end for{1-Hop Loop}
  end for{Root Portfolio Loop}
if  $sum_{weights} \neq 0$  then
  return  $\frac{sum_{trust}}{sum_{weights}}$ 
else
  return 0
end if

```

A. Performance Analysis

Our initial experiments focused on the performance of TwoHop in small networks; as a typical example of small communities of friends interested in specific services we investigated networks with a maximum of 500 participants. In Fig. 2 the x axis represents the number of peers participating in the network, the y axis represents the maximum number of entries in a peer's portfolio (the exact number is randomly generated) and the z axis represents the number of seconds required to reach an output trust value for a query from a random source peer to a random target service provider.

Our results¹ show that the TwoHop algorithm primarily

¹Since our prototype is implemented in an interpreted language (rather than a more efficient, possibly compiled one), it is the relative value of the results which is of importance rather than their absolute value.

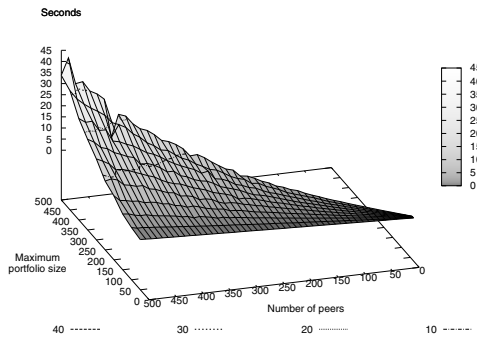


Fig. 2: TwoHop performance in networks with a maximum of 500 participants.

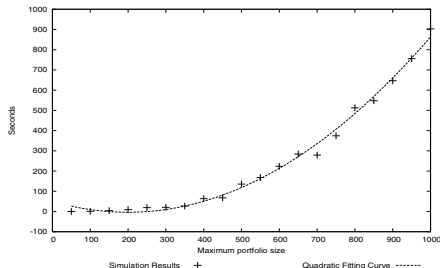


Fig. 3: Impact of portfolio size on TwoHop’s performance in a network with a fixed number of 5,000 participants.

depends on the size of the trust portfolios of the peers that participate in a network for a given service type. This experimental observation is in accordance with the theoretical execution time we calculated in paragraph III-D. To clarify this observation further, we have conducted a new experiment with a fixed number of total participating peers; Fig. 3 presents the observed results in a network of 5,000 peers. The results follow a quadratic curve, a direct consequence of the $O(k^2)$ algorithm complexity.

B. Dishonest Evaluators

In this experiment we study the impact a set of dishonest evaluators has on the trust hierarchy. We define a dishonest evaluator as a peer that provides false evaluations for a particular service. We generate a random network of 1,000 peers with a maximum portfolio size of 200 entries and we select the service provider with the most evaluations as the victim of this attack. We also select a random trust portfolio to be used as the root trust portfolio for network-wide trust evaluations of the victim’s service. Each simulation run records the change Δ_T in trust value when a given percentage of random peers become dishonest evaluators of the victim’s service. We define $\Delta_T = |T_0 - T_y|$, with T_0 being the trust value when the trust hierarchy contains no dishonest evaluators, and T_y being the trust value when the hierarchy contains a percentage y of dishonest evaluators.

To differentiate the honest from the dishonest evaluators, each honest evaluator holds a random evaluation of the victim’s service from the range $[\rho - 0.1, \rho + 0.1]$, where $\rho \in (0.1, 0.9)$. ρ is a random value that is common for all

honest evaluators per simulated scenario. In contrast, dishonest evaluators hold completely random (i.e. false) evaluations of the victim’s service, with no requirement for similarity in their respective evaluation values. Fig. 4a shows that the impact Δ_T on trust calculations increases gradually as the population of dishonest evaluators becomes larger.

C. Minimising the Impact of Dishonest Evaluators

Peers may use their trust weights as a means of defence against dishonest evaluators. In this simulation we will use the same network of peers as in our previous experiment, this time with a fixed 40% of the victim’s evaluators being dishonest. Fig. 4b shows that the impact Δ_T of the dishonest evaluators can be minimised substantially if other peers “penalise” the dishonest ones with low assessment weights. In particular, each penalising peer updates his old assessment weight ($weight_{old}$) towards a dishonest evaluator, with a new value that is significantly lower than the original ($0.2 \times weight_{old}$). Our simulation results show that by increasing the number of penalising peers, the impact of the dishonest evaluators can be minimised. It is possible to completely isolate the effect of these peers by updating the assessment weights to the minimum allowed value (e.g. 0.01) but this approach may be unrealistic, especially for peers that once held high scores in these assessments.

D. Conspiring Peers

In this experiment we will study the way in which peers acting as a team may influence the TwoHop system. Let us suppose that a set of peers have formed a group of mutual trust (i.e. they have assigned high assessment weights to each other). Peers of this group have agreed to rate a certain service provider differently from other peers. This type of evaluation could be thought of as a colluding agreement between the members of this group and we will henceforth call these peers, “conspiring” peers. Conspiring peers acting as evaluators may have a limited influence on a peer’s trust calculations (as we saw in paragraph IV-C). But if any of these conspiring peers were to be part of the root trust portfolio, then they would act as judges and their trust weights would heavily influence the trust calculation results.

To illustrate this point, we transform the dishonest evaluators of the network presented in paragraph IV-B into conspiring peers, each one connected with all other members of the group via random assessment weights from the range $[0.8, 1.0]$. Much like the honest evaluators, the conspiring peers hold random evaluations of the victim’s service with values from the range $[\sigma - 0.1, \sigma + 0.1]$, where $\sigma \in (0.1, 0.9)$. σ is a random value which is common for all members of the conspiring group, per simulated scenario. Fig. 4c shows the increase in trust change Δ_T as more and more peers evaluating the victim provider join the conspiring group.

Any member of one’s trust portfolio must be considered an “immediate acquaintance;” it is a peer that has been closely evaluated both for his services and for his expertise on a given service field. This means that if such a peer becomes a member

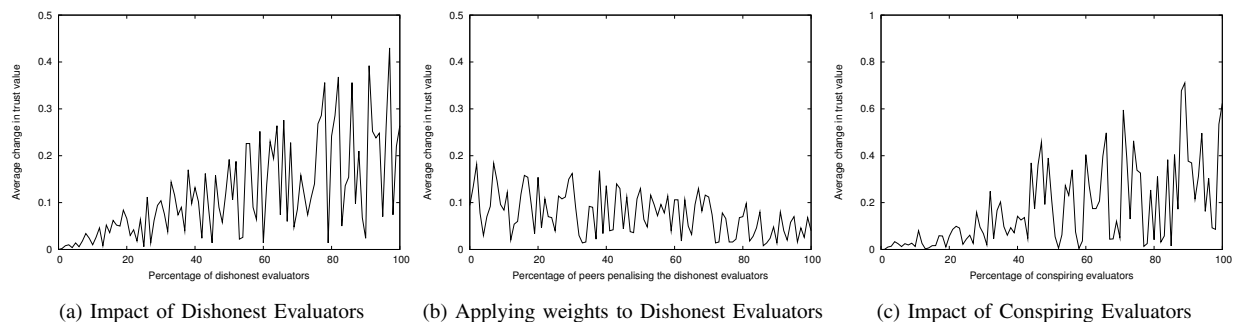


Fig. 4: Dishonest evaluators in a network of 1,000 participants with a maximum portfolio size of 200 entries.

of a conspiring group, the owner of the root trust portfolio will either join the group as well (if he identifies with the behaviour of this group) or readjust his weights (or even remove them completely from his portfolio) so that members of this group do not influence his trust calculations. Furthermore, groups of peers may be pointing to new (not established) service providers that may play a beneficial role to the network later on in the future. This type of clustering is not necessarily of malicious nature since it allows the network to follow different trends and directions that resemble more closely the needs of its users.

V. DISCUSSION & CONCLUSIONS

Our experiments have demonstrated that TwoHop is robust against dishonest evaluators, given that the number of colluding attackers does not constitute a significant percentage of the total number of hierarchy participants. This is a realistic assumption; if the number of dishonest users exceeds that of honest participants then the nature of the hierarchy of interest changes accordingly and now accommodates the needs of the majority irrespectively of their intentions. Evaluations from solitary dishonest entities have little impact on our algorithm, even if the attacker resides on the first delegation step from the querying entity.

Most existing trust evaluation and reputation systems are vulnerable to *Sybil* attacks [10]. Since TwoHop does not try to regulate the creation of new identities in any way nor does it differentiate users based on identities it is, in theory, vulnerable to *Sybil* attacks. However, our experiments have demonstrated that even if a large number of users (fake or real) cross-certify each other trying to influence the output of the algorithm their impact can be limited to acceptable levels. We plan to analyse the behaviour of TwoHop under active *Sybil* attacks using the *Sybilproof* framework [11].

It is conceivable that if a user receives a lot of low evaluations he may want to discard his identity and join the TwoHop network with a new one. No existing system that we are aware of can successfully forbid this. One way to make identity discarding less attractive is to never allow a trust value to decrease below the one assigned to new participants of a system. In TwoHop a user that discards an identity loses all his connections to the rest of the network and therefore all the value that he expects to gain from it. We need to find metrics

that will allow us to compare the value a user gets from the network against the incentives to discard his identity when he gets negative evaluations.

In this paper we have presented a novel distributed trust evaluation metric for peer-to-peer collaboration environments. The shared intelligence model offered by TwoHop allows users to gain value from the collective experiences of third parties with similar interests and makes the trust network more robust against individuals advertising false claims. Finally, the 2-hop depth limit in recommendation chains creates trust hierarchies that are capable of describing peer interactions within networked environments, while keeping the complexity of the trust calculation algorithm within acceptable levels for use in Internet applications.

Acknowledgements – This work was supported by the University of Piraeus Research Center, and Science Foundation Ireland under grant number 03/CE3/I405.

REFERENCES

- [1] A. Josang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.
- [2] G. Theodorakopoulos and J. Baras, "On Trust Models and Trust Evaluation Metrics for Ad hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 318–328, 2006.
- [3] R. Levien and A. Aiken, "Attack-resistant Trust Metrics for Public Key Certification," in *Proceedings of 7th USENIX Security Symposium*, 1998, pp. 229–242.
- [4] P. Zimmermann, *The Official PGP User's Guide*. MIT Press, 1995.
- [5] E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati, "Managing and Sharing Servents' Reputations in P2P Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 840–854, 2003.
- [6] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," in *Proceedings of IEEE Hawaii International Conference on System Sciences* 33, 2000.
- [7] B. Christianson and W. Harbison, "Why Isn't Trust Transitive?" in *Proceedings of 3rd International Workshop on Security Protocols*, 1996, pp. 171–176.
- [8] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: a Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.
- [9] R. Hill and R. Dunbar, "Social Network Size in Humans," *Human Nature*, vol. 14, no. 1, pp. 53–72, 2003.
- [10] J. Douceur, "The Sybil Attack," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002, pp. 251–260.
- [11] A. Cheng and E. Friedman, "Sybilproof Reputation Mechanisms," in *Proceedings of 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON'05)*, 2005, pp. 128–132.