School of Computing Science,
University of Newcastle upon Tyne

A proposal for Trust-Enabled P2P Recommendation Systems

Georgios Pitsilis and Lindsay Marshall

Technical Report Series

CS-TR-910

May 2005

A proposal for Trust-Enabled P2P Recommendation Systems

Georgios Pitsilis[1] and Lindsay Marshall

School of Computing Science, University of Newcastle Upon-Tyne
Newcastle Upon Tyne NE1 7RU, U.K.
{Georgios.Pitsilis,Lindsay.Marshall)@ncl.ac.uk

Abstract. In this paper we present a trust-oriented solution that can be used when building Peer-to-Peer recommendation systems. We discuss its benefits in comparison to a centralized solution, its requirements, its pitfalls and how these can be overcome. In our approach, we base the formation of trust on evidential reasoning and made the proposed design with ease of adoption by existing infrastructures in mind. The paper also includes a preliminary analysis of performance based on a simple model we use to investigate the impacts on scalability and thus show the applicability of the protocol.

Keywords: *Recommendation Systems, Subjective Logic, Trust Modeling*

1 Introduction

Recommendation systems (RS) are used widely in e-commerce where suggestions are offered to customers about products they might potentially like [1]. However, Recommender Systems are not perfect and they seem to have weaknesses such as their vulnerability to various attacks and the low quality of predictions caused by the use of sparse datasets. Using *Trust* in recommendation systems is known to have a positive effect with regard to the quality of the service [3,5], but due to the extra resources needed to run trust protocols it is not clear whether or not its use is appropriate.

The work presented in this paper investigates the applicability of a trust-enabled recommendation system in a distributed environment intended to overcome the scalability barriers of centralized approaches. The emergence of P2P networks has changed the way that people (and entities in general) collaborate to achieve common goals. Therefore, in our research we focus on a P2P solution in which trust will be supported by a virtual over-net built up from the user's trust relationships.

The rest of the paper is organized as follows. Section 2 discusses the problems of recommendation systems and how trust can help to solve them. In section 3 we describe the proposed protocol and explained the advantages that it offers when

[1] Scholar of the Greek State Scholarships Foundation (IKY)

applied to P2P architectures. Section 4 includes the testing and the results so far. Finally in section 5 we discuss some future issues and conclusions.

2 Recommendation systems and their problems

Today's recommendation systems operate essentially as centralized services [2,4]. The main idea behind them is to correlate users based on the opinions they have expressed in the past and to provide them with suggestions either as list of products or as predictions of ratings for items they want to know about.

Trust-enabled recommendation systems can provide some improvement in the quality of recommendations. This is achieved by decreasing *sparsity*, but if this is not done correctly then the extra effort that is required may cause more side-effects than benefits [3]. By *sparsity* we mean a lack of users' shared-experience data that is required for Collaborative Filtering systems (the best known type of RS) to work. The extra effort required results because such a technique requires foremost a vast amount of computation to be carried out, to support the trust infrastructure, which increases with the number of users. Given the requirement that such systems need to operate in a live, interactive fashion, response times must be kept within strict limits and any additional computational load may have disastrous consequences for usability.

If the system uses a flooding algorithm to propagate trust values to neighbouring users, this would require $O(n^t)$ unicast operations since a query running in depth d would need roughly $L = n\sum_{t=1}^{d} n^{t-1}$ operations for each search operation to propagate to *n* trusted neighbours. In reality the number of messages will not be as many as the formula shows due to sparsity in the datasets.

High sparsity is also responsible for obscuring the scalability problem since it is never possible for all users to be correlated with each other. This means that predictions cannot be made about every prospective choice for any user within the community. *Computability* is a measure we used to express how well a system can provide predictions about a wide range of products, and, as can be seen in [3], the deployment of trust in a *Collaborative Filtering* system, helps *Computability* to increase without significant impact on the *Prediction Error* in comparison to a plain CF system.

The two challenges – *reduction of sparseness* and *scalability* – are in conflict, since the less time that is spent on a search query, the worse the quality of the results the system will give. The increasing number of computations is the main reason for the reduced scalability of CF systems and this is why, theoretically, a centralized recommendation system of this type would not scale to massive number of users and products.

3 Our Proposed Architecture

In this section we outline the operations that our proposed architecture requires and associate them with relevant functions that can be found in a typical CF system. Sarwar et. al. [6] distinguish in total 3 phases in the recommendation production and presents *knowledge representation, neighbourhood formation* and *recommendation generation* as the key tasks. Our P2P-based concept fits this categorization almost perfectly since Knowledge Representation and Neighbourhood Formation overlap. So, we distinguish *Trust discovery*, *Recommendation Search* and *Recommendation Generation* respectively as the main operations of the distributed proposal.

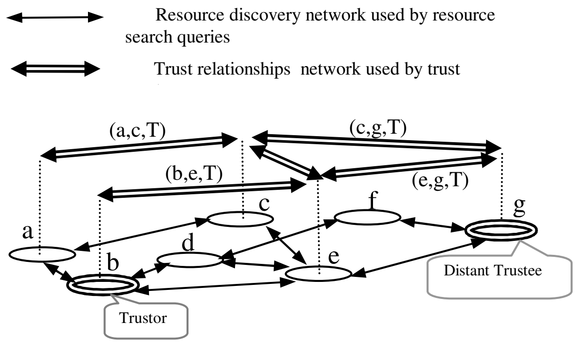

Resource discovery network used by resource search queries
Trust relationships network used by trust
(a,c,T)
(c,g,T)
(b,e,T)
(e,g,T)
a
b
c
d
e
f
g
Distant Trustee
Trustor

Figure 1. A typical Trust Over-net.

In order to explain these operations we first present a common scenario for searching in a P2P system that provides trust-based recommendations:

1. User U initiates a query searching for some product A she is interested in.
2. The system detects, through the resource discovery operations, the subset S of entities that can provide their own experiences with the product A.
3. The system tries to estimate the trustworthiness (so called *Secondary Trust*) of the S entities through the trust graph and informs A, and after that attempts to reach them by initiating trust queries via its neighbours.
4. Once the trust information has been received back by A and the trust of every S has been established, U estimates what she believes about the trustworthiness of each one, from which she finally derives the expected rating of product A

Figure 1 shows an example of two entities **g** and **b** which share some experience with product A. In the scenario, entity **a** is also interested in product A and is trying to derive the trustworthiness of **g** and **b** - which are both untrusted neighbours of **a** - though the trust graph.

We perceive trust establishment as the operation of estimating the levels of belief between users using their behavioural data as evidence. *Subjective logic* is a framework of artificial reasoning suitable for modelling such relationships which conveniently also deals with the fact that knowledge is always imperfect [9]. In this framework the ratio of trust, distrust and absence of data are expressed as triplets of belief, disbelief and uncertainty (b,d,u).

3.1 Trust Discovery

The *Trust Discovery* phase we mentioned is concerned with the representation of trust relationships and the formation of neighbourhoods of users with respect to the level of trust they place on each other. This kind of trust is known as *Primary Trust*. This phase must take place before a search operation commences and every entity that wishes to participate in the trust scheme must implement this phase. In contrast to centralized *recommendation systems*, in our design there is no central customer-product matrix since users maintain their own tables for selecting their neighbours. In [7] we presented a model for trust derivation using evidence in which trust values are derived from the common experiences that every pair of users have had in the past. In that model the existence of common experiences is a requirement for establishment of trust. The operations of the Trust Derivation phase can be carried out jointly off-line by the parties involved, who periodically broadcast messages indicating they are looking for entities with which they wish to establish primary trust.

Once the *Primary Trust* relationships have been established the trust over-net is set and ready to resolve trust requests. To keep the protocol as simple as possible we rely solely upon the honesty of each counterpart and allow the calculations to be done by the entities involved themselves. Otherwise, if we consider attacks from malicious users who might want to influence their counterparts by providing fraudulent evidence, the calculation of trust should be undertaken by a third party that will perform the trust calculation and transmit the result confidentially to the parties involved. We consider the detection and prevention of such attacks an issue for further research.

3.2 Recommendation Search

Once the trust graph has been shaped, *Recommendation Search* queries can then be serviced. The purpose of this kind of query is to make those entities that are known to have useful experiences reachable and then to derive their trustworthiness. We assume that a flooding scheme will be used, through which trust queries will propagate in the trust graph up to a defined hop distance. It is worth mentioning the requirement for *common purpose* [13] to exist in the relationships in order to make it possible for users to employ transitive trust via the graph.

Valid trust paths starting from the query originator and ending at the target node, if reached, will be sent all the way back to the origin using the same paths. Upon reception of the replies the originator should from then on maintain a collection of trust vectors which constitute a graph leading to the distant node. Then, the derived trust of the target node can be easily calculated by parsing and analyzing the graph. Various quality restrictions can be set when initiating queries, for instance to avoid using entities with low numbers of experiences or those which are not trustworthy. This policy can also reduce the number of unimportant links in the resulting graph and thus help to keep the derived trust calculation times low. [3] shows that using filters in trust queries has no serious impact on the error of the predicted recommendations. As the search goes deeper, the same happens to coverage ratio – the number of entities reached – but the resulting exponential increase in the number of messages, due to using a flooding protocol, impacts scalability.

The reason the resource greedy task of parsing the entire graph is done by the querying entity itself and not by the intermediate entities is that in this way it preserves the dependency avoidance requirement of trust. Parsing recursively backwards along the paths as the replies traverse the intermediate nodes in the graph, may lead to the wrong calculation of *Derived Trust* due to hidden topologies that might exist and of which the query originator is not aware. This is known in the literature [8] as the *Dependency Problem* in trust derivation.

3.3 Recommendation Generation

Finally, *Recommendation Generation* comprises the collection of all *Derived Trust* query results from the previous step which can now be turned into similarity measures for the entities that have provided recommendations about the product in question. The steps following are the same as those that can be found in a plain CF. The actual prediction of the rate that the querying user would give to the particular product can be approximated using a suitable formula for that purpose. For example, *Grouplens* RS uses Resnick's prediction formula [10].

3.4 Involving time in the Trust calculation

In order to improve the quality of the calculated trust, it is advisable that time be used in the processes involved [13]. The idea is that relying parties will always try to base derived trust (See 3.2) on the most recent recommendations. A simple idea is to introduce timestamps when trust opinions are derived from evidence that have the form of ratings, and discard those that become outdated. In this way, as more ratings are inserted into the system and the users increase their levels of maturity by using the newly gathered info, their opinions will as well follow their new perceptions of the 'world'.

4 Testing

As we mentioned in the previous paragraph, the scalability problems of a trust-enabled recommendation system come mainly from the requirement for opinion independence, which requires a whole graph structure to be transmitted back to the querying entity. The impact on scalability is two fold. First, the network traffic that is caused by the high number of trust vectors that go through the connection lines and second, the operating system overhead due having to parse the received structure. In the tests we performed, we attempted to approximate the behaviour of the proposed system under various conditions, which we modelled as variables during the testing.

4.1 Assumptions

In order to perform the test we made a number of assumptions regarding the conditions under which operation takes place.

- Due to the complexity of the operations of trust derivation which made it difficult to model them in an analytical way, we used a sample of experimental resource and trust queries [3] for which we knew the length of query responses as well as the number of simplification operations needed for the graph analysis. From these we can estimate the traffic and the computing power that each query requires.
- To estimate the scalability of the protocol it is necessary to know the impact, in terms of resource usage, on a single node rather on the whole network. Therefore, the performance measurements taken represent what is seen from the point of view of a single node.
- We assume the pattern of user demand follows a *Poisson* pdf with parameter $\lambda=5(\text{min}^{-1})$. That means every minute the physical user submits on average 5 search requests to the system. In all our experiments we have kept this parameter fixed.
- Every trust vector in the graph is stored as a triplet of *{Primary Trust value(b,d,u), Trustor, Trustee}* and it has a fixed size of 25 bytes, with 15 bytes allocated to the trust value being carried and the remaining 20 bytes split into the node addresses of *trustor* and *trustee* respectively.
- The number of trust queries that follow a resource-searching query is equal in number to the entities that have been found to have some experience with the resource.
- The time taken for a query to propagate to one hop distance is set by default to 1 sec and is standard throughout the experiment.
- The processing of the trust graph is done by applying the consensus and suggestion operators of the algebra of *Uncertain Probabilities* [8], for simplifying the parallel and serial combinations of opinion vectors respectively. The time required for a simplification operation is dependent on the number of threads that run in parallel as well as the computing power of the node's CPU. From our measurements, an elementary simplification operation takes about 1msec in a modern PC, thus we assume for the sake of the experiment that in all it takes:

$$t_s = \frac{1}{1000}\sec \times Number\_Of\_Available\_Threads$$

The time required for the collection of the graph is calculated as:

$t_c = \frac{M \times V}{AvailableBandWidth} \times Number\_Of\_Available\_Threads$, where M is the size of the replying graph and V the size of the trust vector in bytes.

Also, in the experiment we have not considered the case of a querying system serving some incoming query requested by some other node while it is processing or collecting the replies to its own queries. In this case a fraction of the bandwidth should be used for serving the incoming requests at the expense of the outgoing requests and we assume it as negligible.

4.2 Variables

We identified 3 variables that make up 18 different testing scenarios. These are:

- Trust filter. This represents the minimum level of belief that a trustor must have placed on some neighbouring trustee in order to allow a query to propagate further away. We used 3 filters b>0.5 , b>0.6 , and b>0.7.
- Max hop distance in query propagation. In the experiment we used only distances for which there were available data (1, 2 and 3 hops).
- The bandwidth capacity of the network connection that links the examined node to the rest of the world. In total we performed tests for speeds of 7 kbytes/sec and 64 kbytes/sec, simulating an analog modem and a DSL connection respectively.

4.3 Test Plan

We ran a series of simulations for every combination of variables of *propagation filter*, *hop distance* and *bandwidth capacity*. In total, every combination of parameters was left to run for 10 sessions of about 3 hours of simulated time each and the results were averaged. That length of simulated time was chosen to assure that the system came to a steady state before measurements were taken.

The measures we used for the evaluation of the model were the *Response time, Success Rate*, and the *Bandwidth Consumption* measure. The first expresses the expected response time between the query initiation and the collection of trust graph. In order to calculate this, for each query we measured the time taken and estimated the statistical distribution of the frequencies. We introduced *Success rate* because pure response times do not have a significant value.

Success Rate expresses the probability that a query completes within a preset threshold of 8 sec. We chose this as a reasonable threshold value and we aimed in this experiment to see how many times this value was exceeded thus rendering the system less usable. That threshold represents a measure called *Patient limit* and some studies in the web environment show for most users *patient limit* has the above size. If no response has been received within this threshold the user may abandon the request or might retry. Abandonment in distributed environments such as P2P is quite expensive in resources because it adds more load and make the situation even worse [12]. Therefore we treat both abandonment and retry as unsuccessful cases of queries.

As regards *Bandwidth Consumption*, we were interested to know for each setup how often the available bandwidth gets saturated throughout the system operation.

In the test-bed we used, we assumed that peers internally operate like M/M/C queuing systems with single FIFO queues and multiple service points as many in number as the threads that the peer's operating system allows to run simultaneously. The use of multiple threads to serve the trust queries in parallel was done to add more realism since that feature is offered in most P2P searching user-interfaces today. Special care was taken to simulate the impact of running multiple threads at the same time that virtually share the processing power and the incoming network bandwidth. Figure 2 explains roughly the algorithm used in our testing plan.

```
while ( time has not elapsed ) do
    if probability that a new query is issued > 5/min
      $P$ = product randomly chosen
      $F$ = Set of peers that have experienced $P$
      for all $S$ in $F$ do
           $W$ = waiting time of $S$ in the queue
           $D$ = propagation delay of $S$
           $R$ = time to collect the responses of $S$
           $C$ = calculation time of $S$
           $ResponseTime = W+D+R+C$
           $Traffic = f(R)$
      end for
    end if
    display average $Delay$ & $Traffic$
 end do
```

Figure 2. The algorithm we used for the testing

4.4 Results

Figure 3 is a graph presenting the Expected Response time of a trust query for each of the different 18 configurations we tested.

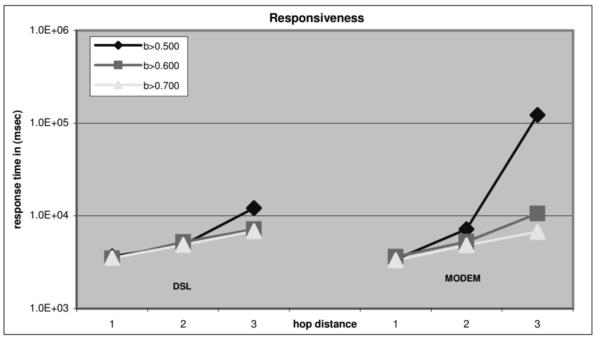


Figure 3. Expected Response times

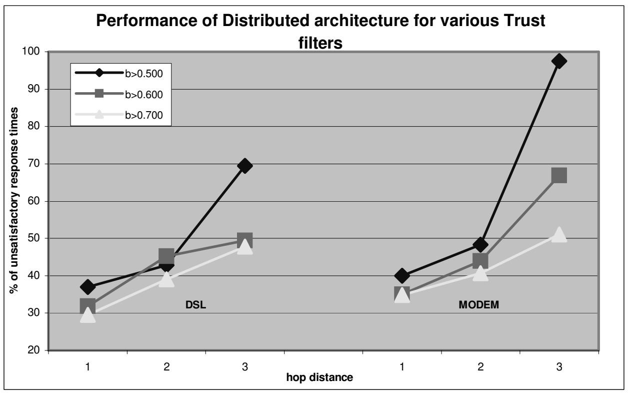


Figure 4. Responses beyond *Patient Limit*

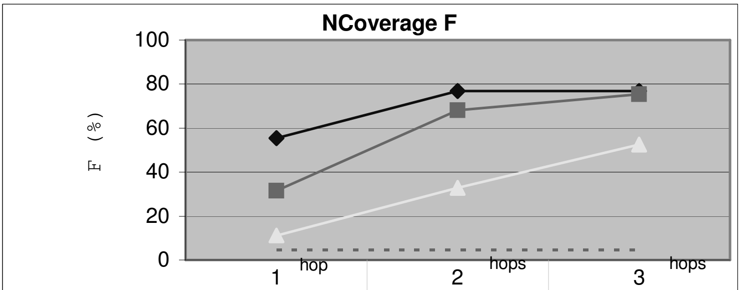
Figure 4 shows the percentage of queries whose response times exceeded the preset threshold of 8 sec and thus were considered unsuccessful.For comparison we present both results using a DSL connection and a Modem.
The figure for performance as we present it in these figures does not seem to be enough to indicate the best choice because it does not contain the gain of every individual case along with the cost. Therefore we introduce the *Satisfaction factor* (*SF*) that combines a measure of performance based on both the ratio of the successful responses and the coverage achieved by using some filtering policy. We define *SF* as:

$$SF = (1 - UnSuccessful) \cdot NCoverage$$

Figure 6 shows how the *SF* factor shapes for various depths of searching and trust filters. *UnSuccess* is the percentage of unsatisfactory responses in terms of time taken for each different configuration, and it can be taken from Figure 4. *Ncoverage* (Normalized Coverage) is the total number of services for which a user can find opinions through the trust graph divided by the total number of services that have been rated by all counterparts together. NCoverage also takes care of the fact that there is always some prediction error. The formula that gives Ncoverage is:

$$F = (1 - \overline{E}) \cdot C$$

where, C is the computability ratio and E the average recommendation error for a particular configuration. By Computability ratio we mean the number of reachable services for the group of users divided by the total number of services about which opinions can be expressed. Figure 5) shows the *NCoverage* for 3 filtering scenarios and various hop distances. The definition of *NCoverage* as well as what values it can take for various infrastructures can be found at [3]. Figure 6 shows that the maximum Satisfaction is reached when a DSL connection is used with (b>500,2 hop distance). In both configurations (modem,DSL), it declines for search depths of 3 hops and when weak filtering policies are used due to the high number of messages that required to apply such policies.



	1 hop	2 hops	3 hops
b>0.5	55.47	76.89	76.89
b>0.6	31.63	68.09	75.33
b>0.7	11.23	32.84	52.52
Intuition	4.69	4.69	4.69

Fig. 5. Normalized coverage factor

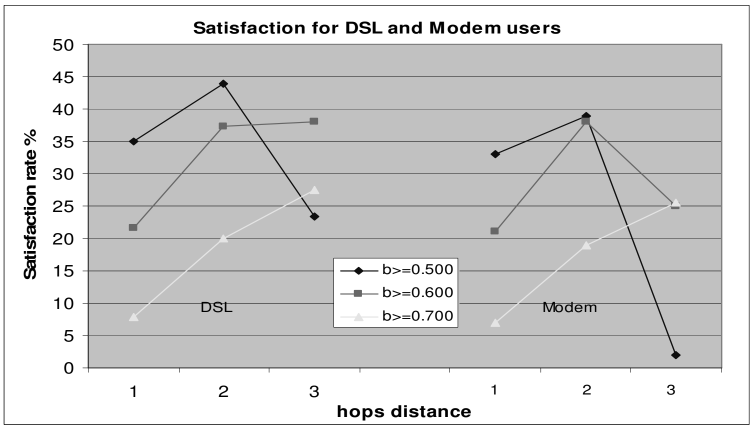


Fig 6. Satisfaction for various hop distances

Figure 7 shows the average bandwidth requirement of each node. In the diagrams we present the most likely setups that could cause saturation (hop distances of 2 and 3 and non-strong trust filters).

That measurement uses the time taken for the trust graphs to be received back through the node's connection link. We were interested in the percentage of time that the node could not cope with its own request demand. The diagram shows how the demand for bandwidth develops throughout the simulation. From all combinations the only one that suffered from bandwidth saturation was that of propagating trust up to 3 hops distance and using a weak trust filter (b>0.500). In all, the line was found to be saturated (demand over 7kbytes/sec) for only 10% of the simulation time.

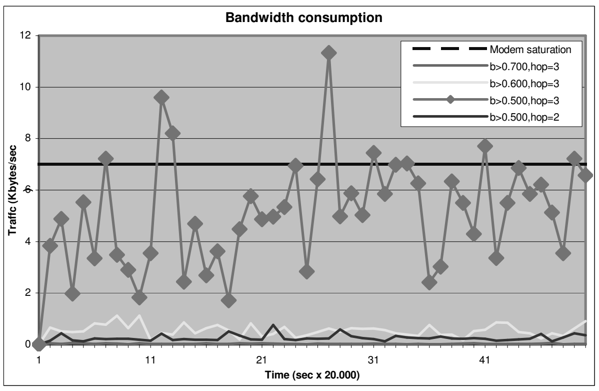


Figure 7. The bandwidth consumption

From the diagrams it can be seen that the most suitable filtering policy is to choose strong trust filters rather than weak ones in the search operations, since the latter create much more traffic which leads to increased waiting times and thus lower user satisfaction from the service. As regards the hop distances to be used for searching, there is no significant divergence (max-min) in the unsatisfactory responses when using strong filters (for filter 0.7 than 0.5 in a DSL the divergence is only 12%). As can be seen, searching as deeply as 3 hops away and using a filter for the trust propagation of 0.5 it is very unlikely that there will be an answer within a reasonable time margin.

There is no special benefit of using DSL lines in the case that a non-strong filtering policy is used since there this has insignificant impact on the response times.

On the other hand, the bandwidth consumption seems to be negligible in almost all the testing setups we tried. This translates to low interference with other network applications that might be running on the user's node.

With regard to the *Combined Satisfaction Factor*, from the diagrams it can be seen that in both cases (either DSL or Modem) has the best value when using weak trust filters (b>0.500) and searching to a depth of 2 hops. Particularly in the case of DSL, the best result can also be achieved by using a medium filter (b>0.600) and hop distance of 2.

Also, the top values of Satisfaction for Modem and DSL are quite close (45% and 40%) which means that there is actually no significant gain from using DSL. Applying a strong filtering policy (b>0.7), we get the same results for both types of connections. However, since the graph for this category does not seem to have a peak value after which it declines, we can say that in that case more investigation is required to find out how it shapes beyond the 3 hop distance.

5 Conclusion - Future Work

The benefits of decentralized architectures, such as P2P, seem to be, both in theory and in practice, quite suitable for supporting services such as Recommendation Systems which were traditionally centralized. In this paper we presented the idea of using RS for those infrastructures. The simple analysis we performed based on real data shows that, within the technical limitations of the current technological infrastructure, such an architecture can favourably support a distributed trust-enabled recommendation system.

As we mentioned, the scope of our study was to provide a microscopic analysis of how the protocol behaves as seen by a single node, without considering the effects of its operation on the rest of the network infrastructure and its effects on other peers. Our future plan is to build an analytic model that will help us to study these issues from a macroscopic view and which would also give an understanding of the protocol performance in extreme situations that we have been unable to test due to lack of experimental data (e.g. propagating trust beyond 3 hops distances). Comparison with an identical centralized solution with regard to the cost/value ratio as well as the applicability to an existing P2P protocol [11] is also an important future issue.

6 References

[1] P.Resnick – H.R.Varian, “Recommender Systems”, Communications of the ACM, 40(3): 56-58, 1997
[2] http://www.epinions.com
[3] G. Pitsilis, L.F. Marshall, ”Trust as a key to improving recommendation systems”, University of Newcastle, School of computing Science, Technical report series, TR-875, November 2004.
[4] http://www.ebay.com
[5] P.Massa, P.Avesani, ”Trust-aware Collaborative Filtering for recommender Systems”, CoopIS/DOA/ODBASE (1) 2004: 492-508
[6] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, ”Analysis of Recommendation Algorithms for E-Commerce”, In Proceedings of the second ACM conference on Electronic Commerce, pg.158-167,ACM Press, 2000.
[7] G. Pitsilis, L.F. Marshall, ”A model of Trust derivation from Evidence for Use in Recommendation systems”, University of Newcastle, School of computing Science, Technical report series, TR-874, November 2004.
[8] A. Jøsang, “An Algebra for Assessing Trust in Certification Chains”, In proceedings of NDSS’99, Network and Distributed Systems Security Symposioum, The Internet Society, San Diego 1999
[9] A.Josang, “A Logic for Uncertain probabilities”, International Journal of Uncertainty, fuzziness and Knowledge based systems, Vol.9,No.3, June 2001.
[10] Resnick,P. and Varian, H.R. (1997). Recommender Systems. Special issue of Communications of the ACM. 40(3).
[11] The Gnutella protocol specification, http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
[12] K. Kant , R. Iyer,” A performance model for Peer to Peer File Sharing Service”, Enterprise Architecture Lab, Intel Corporation, Technical Report, 16 November 2001

[13] A.Jøsang – E.Gray – M.Kinateder, “Analyzing topologies of Transitive Trust”, In proceedings of the Workshop of Formal Aspects of Security and Trust, (FAST 2003), Piza September 2003.