Decentralisation of ScoreFinder: A Framework for Credibility Management on User-Generated Contents

Yang Liao, Aaron Harwood, and Kotagiri Ramamohanarao

Computer Science and Software Engineering,
The University of Melbourne, Victoria, Australia
{liaoy,aaron,rao}@csse.unimelb.edu.au

Abstract. *User-generated content* (UGC) from Internet users has significant value only when its credibility can be established. A basic approach to establishing credibility is to take an average of scores from annotators, while more sophisticated approaches have been used to eliminate anomalous scoring behaviour by giving different weights to scores from different annotator profiles. A number of applications such as file sharing and article reviewing use a decentralised architecture. While computing a weighted average of static values in a decentralised application is well studied, sophisticated UGC algorithms are more complicated since source values to be aggregated and their weights may change in time. In our work we consider a centralised credibility management algorithm, ScoreFinder, as an example, and show both structured and unstructured approaches for computing time-dependent weighted average values in *peer-to-peer* (P2P) networks. Experimental results on two real data sets demonstrate that our approaches converge and deliver results comparable to those from the centralised version of ScoreFinder.

1 Introduction

User-Generated Content (UGC) is an increasingly important information source on the Web. UGC applications process individual data streams from a large number of Internet users and make this information available globally, e.g. Social Networking, Collaborative Content Publishing, File Sharing, Virtual Worlds and other collaborative activities. Examples of UGC include purported factual information, user opinions or reviews on public events and issues, files and documents. The value of the information from these applications is proportional to the information credibility – users need to be able to ascertain the credibility of the information in the UGC.

Confirming the credibility of a given content item using a centralised authority is infeasible due to the scale of UGC, and so most systems allow the users themselves to provide feedback, or score the content items that other users have provided. ScoreFinder [4] was proposed addressing the problem of aggregating feedback.

Not all UGC applications can be hosted centrally. For example, file sharing applications are more effective when they are decentralised, using a peer-to-peer (P2P) model, and it is widely accepted that the P2P model is applicable to a large range of UGC applications. Such applications are emerging, and they do not have a central trusted authority that will undertake the computations in a secure way. Indeed one of the motivations for

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 272–282, 2010.

these applications is to the contrary that there is no central authority in which trust must be placed. The distribution of trust therefore removes the so called "trust bottleneck" but creates a different problem of distributed trust. Furthermore, distributed functionality that benefits the majority of peers in a peer-to-peer system, such as UGC scoring, is likely to be supported in the sense that the required computational load put onto the peers will be accepted. As a result, sophisticated UGC credibility methods are very desirable but significantly challenging to apply in a decentralised system.

Both *structured* and *unstructured* peer-to-peer networks are considered in this paper, and different strategies are accordingly used. Structured networks provide unified access to each shared resource, namely all peers in a structured network can map an arbitrary resource identifier – usually file names or hash values of the file content – to a certain host. On the other hand, each peer in an unstructured peer-to-peer network sees only a local area of the network, and searches intended resources by propagating request messages.

In this paper we consider the problem of aggregating users' feedback for both a structured and an unstructured P2P architecture, and we use ScoreFinder as an instance of sophisticated credibility management algorithms. We provide a structured P2P approach for implementing ScoreFinder, and show how is it implemented on a structured peer-to-peer platform. We also provide an unstructured P2P approach for implementing ScoreFinder, based on gossiping. Particularly, the method of using time-slots defined on the Real-Time Clock to synchronise decentralised computation is novel and useful in other gossiping protocols where restricted synchronisation is required. We simulate network topology and churn to show its affect, as compared to a centralised or ideal computation.

2 Related Work

2.1 Article-Annotator Model and ScoreFinder

We in [4] introduce a model of credibility management, called the *Article-Annotator* (A-A) model and a comprehensive framework for Credibility Management, called *ScoreFinder*. The participants and shared content items are named *Annotators* and *Articles* respectively, and the evaluations made by annotators are called *Scores*, which are numbers between 0 and 1.

The key operation of ScoreFinder, denoted as `ExpertnessEstimation`, is to iteratively calculate the weight of each score contributed by annotators to articles, and accordingly calculate the weighted average of scores to an article as the inference of the value or quality of the article, denoted as `AggregateResult`. An additional operation is to shift scores from each user to remove general biases, denoted as `ScoreShifting`.

2.2 Decentralised Frameworks

There are decentralised frameworks for implementing algorithms in structured and unstructured peer-to-peer networks. In unstructured p2p networks, there are algorithms

employing flood or gossip messages to gradually aggregate values distributed over all peers. In [5], the authors proposed an approach to compute an overall average of values on each peer by continuously exchanging the status in each small-scaled sub-network (particularly, between the direct neighbourhoods). Similar approaches are adopted in GossipTrust [11], where opinions are uninterruptedly exchanged between each peer and a randomly selected peer (or all other peers) to reach the commonly agreed values. In [1], a general model of such problems is established for computing a linear combination of values distributed in peer-to-peer networks. More emphasis in the paper is placed on privacy of participants. On the other hand, most structured networks maintain a *Distributed Hash Table* (DHT) among peers. A DHT provides a many-to-one map between keys and peers; in most cases the distribution of keys over peers is uniformly random. It can be used for distributing data or for assigning logical roles over peers. Upon a given set of peers, a search to a given key always hits the same peer. These characteristics are employed in our framework, among other things, to map each of items to a peer that is in charge of organising the computation. Sophisticated frameworks also provide the ability of maintaining integrity of the Hash Table, by duplicating and/or migrating data items from left peers to other available peers. In practice, we used Pastry [9] as the DHT in our experiments; other DHT platforms like CAN [7] and Chord [10] provide the same function.

3 Decentralisation of ScoreFinder

3.1 Decentralisation in Structured Peer-to-Peer Networks

In this section we explain how we implemented a structured P2P application that uses ScoreFinder to rank articles. A decentralised architecture is proposed in Figure 1. The centralised ScoreFinder algorithm is simply split into two parts, the *annotator component* and the *tracker component* that could run on each peer; data exchange between such peers is through the underlying structured peer-to-peer network. The principles in splitting the centralised algorithm are: (1) to minimise the network transfer, and (2) to minimise computation on the tracker side because trackers are likely to become bottlenecks in the system. Thereby, the annotator component provides its expertness estimate and score shifting factor (respectively e and b) to each related tracker (i.e. trackers hosting articles that this annotator has read) and the tracker component provides the most recent results ($\mathbf{r} = \{r_j\}$) on hosted articles to the annotators that have read them.

Algorithm 1 and Algorithm 2 outline the computation in the tracker component and the annotator component, respectively. $\mathbf{S}^T$, $\mathbf{E}^T$ and $\mathbf{r}^T$ denote the relevant part (i.e. those for articles hosted in this peer) of $\mathbf{S}$, $\mathbf{E}$ and $\mathbf{r}$ in each tracker peer, while $\mathbf{S}^A$ and $\mathbf{r}^A$ denote the relevant part of $\mathbf{S}$ and $\mathbf{r}$ in each annotator peer. In particular, $\mathbf{r}^{AT}$ denotes the set of temporary results in the intersection of the corresponding $\mathbf{r}^A$ and $\mathbf{r}^T$. We use $\mathbf{B}^T$ to denote the score shifting vector (i.e. b_i) from annotators who gave scores to articles that are hosted by this tracker.

These two algorithms continue to iterate for the lifetime of the objects which instantiate them. We consider that in a peer-to-peer network, there are always new peers, new content items and new annotations to be included, so the process of refining the ranking

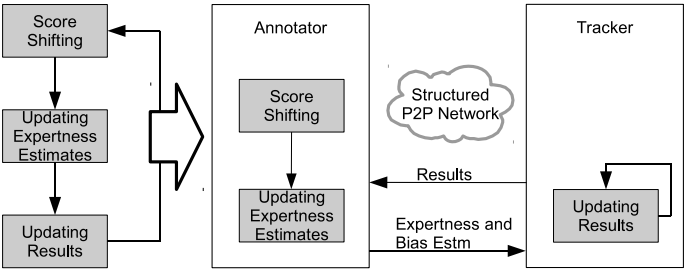


Fig. 1. Decentralising ScoreFinder on structured peer-to-peer networks

Algorithm 1. Process on Each Tracker

Require: All s_{ij} and score shifting factors
Ensure: Temporary result of each content item hosted by this tracker

1: **loop**
2: $\mathbf{S^{T'}} \leftarrow$ `ScoreShifting`$(\mathbf{S^T}, \mathbf{B^T})$
3: $\mathbf{r^T} \leftarrow$ `AggregateResult`$(\mathbf{S^{T'}}, \mathbf{E^T})$
4: **for all** connected annotators **do**
5: `SendToAnnotator`$(\mathbf{r^{AT}})$
6: **end for**
7: `Wait_A_Random_Period`
8: **end loop**

Algorithm 2. Process on Each Annotator (i)

Require: All s_{ij} and the temporary result vector r
Ensure: The expertness and score shifting factor of this annotator

1: **loop**
2: $e \leftarrow$ `ExpertnessEstimation`$(\mathbf{r^A}, \mathbf{S^A})$
3: $b \leftarrow$ `ScoreShiftingFactor`$(\mathbf{r^A}, \mathbf{S^A})$
4: **for all** connected trackers **do**
5: `SendToTracker`(e, b)
6: **end for**
7: `Wait_A_Random_Period`
8: **end loop**

is continuously running across the life cycle of the peer-to-peer application. Furthermore, there is no synchronisation between peers, i.e. every peer arbitrarily sends its most recent data at any time, and is always ready to receive messages from others. After receiving an update message, e.g. updating e or r_i, the peer updates its local cache, and uses the cache to continue with the next round of computation. Between each loop, the algorithm pauses for a short period. Network conditions largely influence the selection of this period. In our experiment, the period is set to 4 seconds since messages are delivered through an application level route, which usually consists of 2 or 3 hops.

3.2 Decentralisation in Unstructured Peer-to-Peer Networks

The ScoreFinder algorithm is also implemented in unstructured peer-to-peer networks using a gossip-based approach. We note that peers in unstructured peer-to-peer networks do not have a global perspective to the whole network, instead each peer sees a relatively small set of peers, called its *neighbours*. Therefore, there is no well-known peers, like trackers in a structured peer-to-peer network, which can be in charge of organising computation for each article; instead peers continuously exchange information with neighbours to propagate the influence of each original value to the whole network.

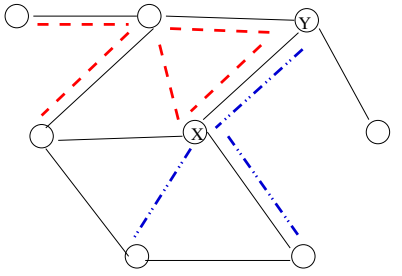
Also considered is the time-dependent change of values and their weights. Because of our expertise and score adjustment operations, the shifted scores ($\mathbf{S^{T'}}$) and the expertise estimates for each annotator (e) change in time. We need to compute time-dependent weighted average values, as shown in the following formula, for coping such changes:

$$f(\mathbf{v}, \mathbf{w}, t) = \sum_i w_i(t) v_i(t), \tag{1}$$

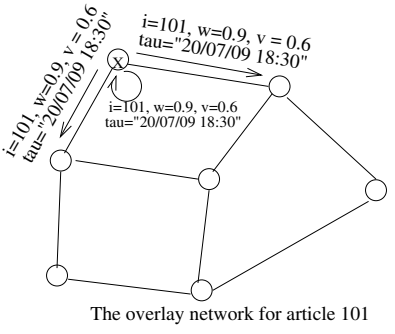
where $\mathbf{v} = \{v_i(t)\}$ and $\mathbf{w}(\mathbf{t}) = \{w_i(t)\}$ are two sets of time-depending functions. We define $\sum_i w_i(t) = 1$ for any t. A number of studies have focused on computing time-dependent weighted average values (as shown in Formula 1) in peer-to-peer networks using gossiped messages. The algorithm introduced in [3] uses epochs to divide the continuous computation into segments, in each segment the computation is restarted to validate changed values on each peer or values from new joiners. Peers in the network are synchronised by a broadcasting protocol.We then implemented ScoreFinder based on our time-slot based protocol.

Article Overlay Network. Considering the very large number of articles that are shared in current Internet applications, it is infeasible to store the status for all articles in each peer. In our approach, each node maintains a separate neighbour set for each annotated article, and exchanges messages only with peers that have the same annotated articles. This is equivalent to building an overlay network for each article over the original peer-to-peer network, and computing the weighted average score for the article on this overlay network. Figure 2(a) shows two overlay networks for article 101 and article 102 resp., built upon the original peer-to-peer network. Each contains all the peers giving scores to the article. Each peer finds peers that annotated same articles by flooding search messages.

Time-slots and Gossip Messages. To compute average values that always reflect the recent status of the peer-to-peer system, we use time-slots to synchronise the computation between peers. Time slots are periods that are predefined on the Real-Time Clock (RTC) and known by all peers. For example, the application could define every 5 minutes from 0:00 o'clock to be a time-slot, i.e. there are 288 slots in 24 hours. Nowadays, a large number of computers, including desktop computers, servers, mainframes and even mobile devices, could maintain a precise Real-Time Clock by frequently synchronising their local clock with Network Time Protocol (NTP) servers. Therefore, a time-slot is started and terminated nearly simultaneously on all such peers. In the beginning of each time-slot, computation is restarted from a new status, in which scores and weights are



(a) Overlay networks for articles are built upon the unstructured peer-to-peer network. It is shown that peer x and peer y have annotated both article 101 and article 102, hence they are in both overlay networks



(b) Gossip messages are sent between each pair of directly connected peers. The ID of the current time-slot is sent in each message along with i, w and v. It is shown that before sending the three messages, peer x has $v_{101} = 1.8$ and $w_{101} = 2.7$

Fig. 2. Unstructured network overlay

both updated based on the recent results from the last time-slot, and weighted average values reflecting those updated parameters are reached in the end of the time-slot.

Figure 2(b) shows an example of three gossip messages exchanged between peers. Each node maintains two values for each annotated articles (identified by i): the summed weights, w_i, and the summed product of weights and scores, v_i. Assuming that a node is directly connected to k peers for article i, in each step (the length of which is randomly determined by the node, usually 3-5 seconds) this node sends a message to each of the k peers as well as to itself. Each message consists of $i, \frac{v}{k+1}, \frac{w}{k+1}$ and the ID of the current time-slot (τ). Afterwards, this peer re-compute its w_i and v_i by adding v and w in all messages received in the past step that have the correct slot ID. It is noticeable that there could be zero or multiple messages from the same peer that are received in a step, which does not influence the validity of the algorithm. This cycle is repeated until the end of the current time-slot, then all peers update the result of each annotated article by computing $r_i^\tau = \frac{v_i}{w_i}$, and update their expertness (e^τ). Before starting the new time-slot, each peer re-evaluate its v_i and w_i by $(e^\tau s_{ij}^\tau)$ and e^τ. This process is shown in Algorithm 3.

4 Experiments

4.1 Outline of Experiment

In evaluating ScoreFinder, we implemented a simulator to imitate the scenario that content items are shared and annotated in peer-to-peer networks, including structured and unstructured models. In our experimental unstructured peer-to-peer network, the bootstrap is done by propagating messages in a flooding way to find peers that annotated same articles. Each peer keeps at least 5 neighbours for each overlay network (i.e. for

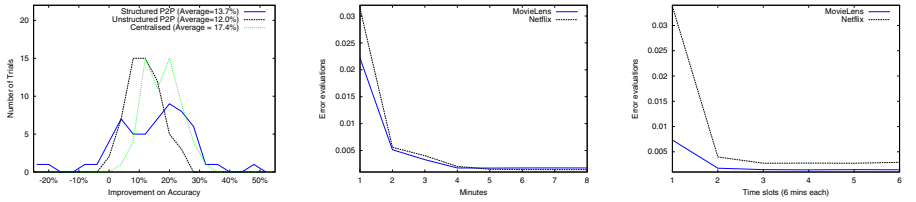
Algorithm 3. The Algorithm in each peer in a unstructured peer-to-peer network

```
τ ← NewTimeSlotID(); e^τ ← 0.5; S^τ ← S; r^τ ← S
loop
    for all annotated articles : i do
        v_i ← e^τ × s_i^τ
        w_i ← e^τ
    end for
    repeat
        for all annotated articles : i do
            if there are peers connected for i then
                for all connected peers and this peer do
                    SendMessage(i, v_i/(k+1), w_i/(k+1), τ)
                end for
                v_i ←sum of received v
                w_i ←sum of received w
            end if
        end for
        Clear_Received_Messages
        Wait_A_Random_Period
    until the end of the time-slot
    τ ← NewTimeSlotID()
    for all annotated articles : i do
        if w_i > 0 then
            r_i^τ ← v_i/w_i
        end if
    end for
    S^τ ← ScoreShifting(S, r^τ)
    e^τ ← ExpertnessEstimation(S^τ, r^τ)
end loop
```

each annotated article, a peer connects to 5 other peers giving scores to this article too), and propagates search messages through the p2p network in bootstrap or when a neighbour is found to be unavailable. Each search message has a Time-to-live (TTL) field to limit the diameter of the propagation. Each peer receiving the message retransmits the message to a limited number of neighbours until a peer annotating the same article is found or the TTL is exhausted. The targeted peer then sends back an acknowledgement message to the inquirer establishing a connection with it. We used the MovieLens data set [8], containing 10 million ratings for 10681 movies from 71567 volunteers,to evaluate our algorithms; the data set was shuffled and randomly re-sampled to a smaller data set in each trial. The two decentralised variants of ScoreFinder were examined in the experiment.

4.2 Evaluation Method

We evaluated our results with a supervised approach. In contrast of the size of our samples (150 annotators), we chose movies that received more than 2000 scores in each of the two data sets, so the average scores from such a large number of annotators could



(a) Distribution of improvement on accuracy to the baseline (on the MovieLens data set, in 60 trials)

(b) Convergence speed in structured peer-to-peer networks

(c) Convergence speed in unstructured peer-to-peer networks

Fig. 3. Performance evaluation on improvements to the baseline

be regarded as the common agreed opinion, and used as the reference. The evaluation function used in the experiment was the Mean Squared Error (MSE) between inferred credibility values and the reference; a smaller evaluation value denotes a more accurate result. In peer-to-peer networks, results on each peer could be different to each other, so we accumulated errors along each annotation, i.e. if an article is annotated by two peers, the error between the oracle score and the final result on each of the two peers is taken into the MSE.

4.3 Results and Discussion

Accuracy. Figure 3(a) shows the distribution of accuracy improvement in 60 trials, each was on a randomly re-sampled data set from respectively the MovieLens data set and the Netflix data set. There were 150 annotators and 2000 content items selected for building each re-sampled data set, as well as all scores between those selected entities. In this figure, our algorithms have a general better performance than the baseline. The performance of the unstructured peer-to-peer implementation stably follows the centralised implementation, where the performance of the structured peer-to-peer implementation has a larger variance to the centralised implementation. This could be explained by the difference on the extents of synchronisation of the two decentralised implementations. In the unstructured network, the algorithm closely follow the steps of the centralised ScoreFinder, namely each iteration is strictly synchronised by time-slots; whereas in the structured network, it does not follow the steps of the centralised ScoreFinder, namely all peers arbitrarily change their scores and weights any time, and the scores are gradually injected from the annotators to the trackers. This inconsistency made the latter one may have larger variance than the centralised algorithm. We also note that there is difference between average values calculated on different peers in the end of each time-slot, this led the accuracy of ScoreFinder in unstructured peer-to-peer networks consistently lower than the centralised implementation.

Looking into a single trial, the speed of convergence of the two ScoreFinder variants is analysed in Figure 3(b) and Figure 3(c). Both variants converged in the trial, nonetheless the unstructured variant consumed more time to reach convergence. This is because agreed weighted average values are reached after a round of exchanging messages in

structured networks, but in unstructured networks it needs a whole time-slot to reach the agreed average values.

Robustness. To reveal the robustness of ScoreFinder in a variable network circumstance, we simulated two types of network conditions with different strengths. First, we randomly discarded messages between peers to examine the influence from packet loss. Second, we randomly shut down a number of peers in every 10 seconds to see how our algorithms reacted to variance on availability of peers. The same data set was used in the two simulations to facilitate comparison. Figure 4(a) and Figure 4(b) show the results in different packet loss rates and invalid peer ratios; in each condition the experiment was run 5 times. It is showed that the variant for unstructured networks is more sensitive to packet loss, whereas that for structured networks is more sensitive to churn.

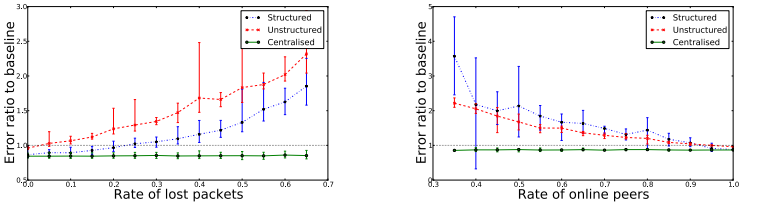


Fig. 4. Robustness testing; the error evaluations are showed by the relative ratios to the results from the baseline; the error-bars show the maximum and minimum relative errors in each condition

5 Conclusion

We introduced two decentralised variants of ScoreFinder, a credibility management framework, for respectively structured and unstructured peer-to-peer network applications. The performance of our algorithm is examined in an experiment using two real world data sets. The results reveal the merit of our approach by comparing to two baselines that are widely used in real applications. A number of issues regarding attacks and misbehaviour of users are discussed in the paper.

The primary challenge to decentralise ScoreFinder is to compute weighted average scores from parameters that change in time, by this means two approaches are used in different network models to synchronise distributed computation. In structured peer-to-peer networks, tracker components are hosted in peers which have addresses that are closest to the identification of articles, whereas globally agreed time-slots are used to coordinate paces of computation in unstructured peer-to-peer networks. The approach of using time-slots to synchronising computation across peers is equal to building a logic-time framework in a distributed environment, by which highly synchronised algorithms can be decentralised. Influence from churn and network conditions to the accuracy of the decentralisation methods was examined in the experiments.

5.1 Future Study

In this paper, no methods for identifying cliques and reducing their influence are considered. Annotators in cliques are only penalised by degrading their individual expertness levels considering that their scores may differ from other experts not in the clique. Nonetheless, unbiased scores from an annotator who gives biased scores to only a small subset of the content items are all lower weighted, hence we need an approach to say which part of their scores is unbiased and which part is not. Still, the criteria for clique identification could be closely related to the application, which is need to be further investigated.

In [6], the authors argue that weighted average values could be computed along spanning trees which is formed by multi-casting messages. We noticed that there are two advantages using spanning trees instead of exchanging gossip messages in implementing ScoreFinder. First, spanning tree-based algorithms have significantly faster convergence speed than gossip message approaches; furthermore, the upper bound of the number of messages to be exchanged before all peers reaching to an agreement is definite in a tree. Second, a peer in a spanning tree may change its local score and weight at any time, and the new agreement of weighted average value that reflects those changes will be reached in each peer under the same upper bound; namely no synchronising mechanisms, like time-slots or super peers, are necessary for spanning-tree styled algorithms. SCRIBE [2], CAN [7] and other multi-casting algorithms are useful platform, whereby spanning trees are built in unstructured peer-to-peer networks.

References

1. Bickson, D., Dolev, D., Bezman, G., Pinkas, B.: Peer-to-peer secure multi-party numerical computation. In: P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing, Washington, DC, USA, pp. 257–266. IEEE Computer Society Press, Los Alamitos (2008)
2. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.I.T.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in communications 20(8), 1489–1499 (2002)
3. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
4. Liao, Y., Harwood, A., Ramamohanarao, K.: Scorefinder: A method for topic sensitive credibility inference on documents. In: The 26th IEEE International Conference on Data Engineering, Long Beach, CA, USA (2010)
5. Mehyar, M., Spanos, D., Pongsajapan, J., Low, S.H., Murray, R.M.: Asynchronous distributed averaging on communication networks. IEEE/ACM Trans. Netw. 15(3), 512–520 (2007)
6. Ramabhadran, S., Ratnasamy, S., Hellerstein, J.M., Shenker, S.: Brief announcement: prefix hash tree. In: PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, p. 368. ACM, New York (2004)
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161–172. ACM, New York (2001)

8. Riedl, J., Konstan, J.: Movielens data sets (July 2009), http://www.grouplens.org/node/73
9. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. LNCS, pp. 329–350. Springer, Heidelberg (2001)
10. Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM, New York (2001)
11. Zhou, R., Hwang, K., Cai, M.: GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks. IEEE Transactions on Knowledge and Data Engineering 20(9), 1282–1295 (2008)