Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 350 – 357

Procedia Computer Science

The 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015)

Effects of knowledge base quality on peer-to-peer information propagation

Michael Gibson[a], Wamberto W. Vasconcelos[a]

[a]*Department of Computing Science, University of Aberdeen, Aberdeen, AB24 3UE, United Kingdom*

Abstract

Peer-to-peer (P2P) networks have become popular through the use of file-sharing over a decentralised network. They rely on keyword searching to allow peers to exchange files. However, for networks handling real-time information, such as those from sensor networks or computer games, information cannot easily be categorised by keywords. We developed a knowledge-based P2P mechanism in multiplayer games, based on roles individuals play and their information needs for game playing. In this paper, we show how we evaluated this mechanism when knowledge quality degrades.

Peer-review under responsibility of the Conference Program Chairs
Keywords: Agents; Computer games; Information propagation; Knowledge-base mechanism; Peer-to-peer

1. Introduction

P2P has gained momentum by allowing files to be shared over a decentralised network. This was possible by assigning unique identities (IDs) to each shareable file as well as for each peer within a network so that certain peers will be responsible for holding some of the files and propagate queries for files to the peer with the requested file. Techniques have been developed to assist in query propagation, including keyword-searching so that an identical ID for the requested file does not have to be known in the first place and improving search pattern from 'blind' *flooded request* to neighbour selecting *document routing* [1]. Other improvements include *structuring* a P2P network to organise what peers share among themselves [2] and sharing other types of information, such as from databases [3].

Although *keyword searching* is useful for 'static' forms of information, such as files, this method is not suitable for *real-time* information. Real-time information is produced and consumed in narrower time-frames, and have brief useful lives before becoming out-of-date and useless. Since this type of information has a short useful life, it is unrealistic to identify the information so that it can be queried by peers through keyword searching.

We, therefore, designed a knowledge-based mechanism which uses elements from the document routing mechanism, but can be used for real-time information propagation/exchange, through the use of *roles*. Instead of assigning keywords and IDs to information, we assign roles to peers which *produce* and *consume* types of information. This

E-mail address: {michael.gibson, w.w.vasconcelos}@abdn.ac.uk

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.106

means instead of peers querying for information itself, peers will query other peers based on the role which is able to produce information for the querying peer. Another advantage of our mechanism is that peers can *volunteer* information based on what a peer will *consume* according to its role. This is similar to *publish/subscribe*[4], where if a peer knows what another peer will always want, the volunteering peer will automatically give information to the consuming peer without the need for extra query messages.

To test our mechanism, we have developed an experiment to show how the *accuracy* of knowledge can affect the performance of our mechanism and, hence, overall performance of a P2P network. Since we focus on real-time information, we use a *computer game* as the domain of our experiment. The game is a simple resource collecting game, where each player (a peer) has to collect as many resources as possible. Using our role system, we limit players to collecting only one type of resource per role and players must coordinate with each other to collect other types of resources. By changing the accuracy of knowledge a player acquires during a game, we can see how degrading knowledge can negatively affect the overall performance of peers playing the game.

This paper is organised as follows. Section 2 cover related work in sharing knowledge over P2P networks. Section 3 explains how our mechanism works. Section 4 describes our experiment to show how knowledge accuracy affects peer performance. Section 5 looks into how the experiment was evaluated and the results acquired. Section 6 concludes the paper with a reflection on our work and how it can be improved upon.

2. Related work

Typical mechanisms involve selecting the best neighbouring peer to contact in order to request of provide information. The exchange can be a query for a resource, for example, a file, a resource itself or about the network itself which can be used to indicate whether peers are joining or leaving the network. How the best peer is selected depends on the relationship between the information to be propagated and the peers themselves. Examples of such mechanisms include *Freenet*[5], which uses *document routing* to store files with generated IDs on peers with similar IDs in an *unstructured network*[6], and *Chord*[2], which uses *distributed hash-tables*[7] to partition peers in a *structured network*[8] and be responsible for resources that belong to specific partitions. These mechanisms are similar in the sense that each resource and peer can be uniquely identified and their identifications are related to each other.

One of the fundamental issues in P2P networks is how to share knowledge among peers, which is what we are interested in. A growing field in researching this issue is utilising *knowledge bases* to share knowledge over P2P networks. Frameworks, such as *Edutella*[9] and *HELIOS*[10], make use of how knowledge can be organised on each peer to assist query creation and propagation. *Distributed Knowledge Management*[11,12] focuses on how smaller communities of peers in a P2P network contribute to knowledge creation as well as sharing with other communities. With *social networking* becoming an important part of general life, research is also looking into how *virtual communities* can share knowledge among themselves[13,14]. The problem with focusing on small communities within a large community is the phenomena of *small-worlds*[15], where peers will only tend to interact with peers it has interacted with before. To ensure the whole network is up-to-date, it is important for all peers to attempt contact with distant peers.

Since our chosen domain focusses on computer games, we also look into existing work involving games and P2P networks. One of the most important issues in running any multiplayer game is to ensure that all players have the same *game state*. This means all players operate within the same state of the game and any changes made, for example, player location, must be reflected on all players' view of the game. One way to ensure players (peers) are kept up-to-date is to use *interest management*[16,17], which prioritises player updates based on interest factors, such as close player positions in-game and visibly looking at events in-game. Players which exhibit interest in each other will receive more updates to ensure better gameplay, whereas non-interested player will receive fewer updates as their actions will not have as much affect. Another way is to partition game areas through *zoning*[18,19], which makes certain peers responsible for handling all events within a particular area of a game. Interest management and zoning are only useful mechanisms within games (and even to certain types of games). Although they share some similarities with other approaches, such as distributed hash-tabling for zoning, these techniques cannot be easily adapted to other domains. Although we use a game for our experiment domain, our mechanism can be adapted to other domains.

3. Routing mechanism

The disadvantage of using an ID system for searching is *how* to uniquely identity a resource so that it can be queried upon. For *static* resources like files, a hashing algorithm can be used to produce a unique identification. Mechanisms can then be designed to work with these identifications, for example, a storage mechanism to store files with similar IDs together on specific peers and a query mechanism to search for a file's ID to retrieve the file. If the resource is *dynamic*, for example, real-time information from a sensor, then it becomes much harder to uniquely identify, and therefore query, the resource because it is constantly changing. Querying for such resources can be even harder if the resource cannot be easily searched for, for example, the resource cannot be explained by a series of keywords. Using IDs or keywords for querying dynamic information is not feasible and, therefore, requires another approach. However, the idea of associating information with an indexable handle is still valid. Therefore, we look at how to query *types* of information rather than the information itself.

3.1. Role-based querying

Instead of indexing information itself for querying, we have implemented a *role system* which categorises different types of information to be handled by specific roles. Depending on the information to be shared among the peers, different roles can be made which are used by the peers. For example, for a file-sharing network, one role can be created to handle text files, another role for video files and so on for other types of files. This means when a peer requires a specific type of file, instead of querying for the file itself, it will query a peer with a role which handles the required type of file. Peers which demand a specific type of resource are what we call *consumers* of that type and will have a role reflecting this. Although this example would not be viable for file-sharing networks, primarily because better systems like keyword-searching already exist, it shows how information which cannot be as easily identifiable or indexed can be handled by peers.

Another advantage of using roles to handle information is the ability of peers to *volunteer* information to other peers. In some P2P networks, especially file-sharing networks, there is the problem of *free-riders*[20], whereby peers only obtain information but never provide any in return. If a peer obtains information which may not be beneficial for itself, but knows another peer which may benefit from it, then the information can be volunteered to the other peer. For a network where peers have to handle real-time information, being able to query and provide information through the use of roles means information can be propagated to the most appropriate peers. Peers which volunteer a type of information, whether the peer provided it itself or obtained it from elsewhere, is a *producer* of that type and will have a role reflecting this.

3.2. Virtual network

As mentioned earlier, queries propagate through neighbouring peers by comparing an ID for the queried information and peer IDs and the closest matching peer will further process the query. For our role-based system, this comparison mechanism is difficult – how can a peer decide on who to send a query, best handled by a particular role, if none of its neighbours have that role; how can a peer choose the 'next best' role to send a query to? Comparing values, such as IDs, is simple compared to comparing concepts, such as roles. We handle this in two ways: roles are *ordered* so that another role can be selected and a *virtual network* is formed by each peer by examining *message paths* from messages.

For any file-sharing P2P network, even if a less than perfect match was made between a query and a neighbouring peer, that peer will process the query if that was the only option – *all* paths are explored before the query fails. This is because of the nature of *depth-first search* that the likes of Freenet use. For this reason, queries in our network should have alternative paths to follow if the best-matching role is not available from the set of neighbouring peers. Since the design and assignment of roles to handle the different types of information is arbitrary, the alternative roles to try will have to be decided beforehand and be agreed by all peers. As long as all alternative roles can be attempted if the first option is not available, each query will have a chance to be propagated to a peer with the required role. This does not mean, however, that all role alternatives follow a similar pattern – roles could be designed which handle query routing

and not necessarily process any information, similar to super-nodes in Gnutella 0.6.[1] Peers with these 'specialist' roles may not contribute with any information, but can assist with propagating queries to the best-matching peers. These roles can have more prominence in the list of alternate roles to try.

Additionally, each peer has the ability to create a *virtual network* based on previous interactions with other peers. This is achieved by recording a message's *path* (a message being a query or reply to a query), containing a list of peers which have processed the message. The main purpose of such a path is to ensure replies to queries are sent back to the querying peer. When a peer processes a query (including creating a message), it adds its details (name and role) to the path before sending it to another peer if the query could not be answered. An answering peer inspects the path and adds it to its view of the network. When a reply to a query is sent back along the same path, the querying peer will also inspect the message path of the reply and add it to its view of the network. Details of how virtual networks are formed and managed are outside the scope of this paper; however, initially peers will only know about their neighbours and their roles. As the peers query each other, and in conjunction with re-routing queries if the ideal role cannot be found, each peer will develop a view of the network, which will lead to future queries being directed to peers rather than having to blind-search through the network.

3.3. Knowledge base

To model the view of the network, as well as the information to assist propagation, we use a *knowledge base*, specifically an *ontology*. An ontology allows *relationships* to be formed between concepts, such as resources and roles. These relationships can be used to determine how concepts are related with each other in order to determine, or *reason*, new information about an existing fact. In our system, we make use of these relationships to determine which peer to query for a specific type of information. We give an example of how these relationships are used in our experiment in Section 4. We can also use these relationships to create the virtual network of each peer. When a peer inspects the message path from a message, adjacent peers in the path are added as 'neighbours' in the ontology. This means as more message paths are analysed, more of each peers' neighbours are discovered, leading to the creation of a virtual network for each peer, even though each peer's view of the network may be different. When a new query has to be created, a peer will be able to select a peer with the necessary role to answer the query and also generate a path for the message to travel. This reduces querying time since a message does not have to be blindly routed to the destination peer.

4. Description of the experiment

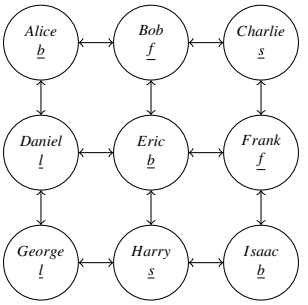
The knowledge used for routing must be accurate to ensure messages are sent to the best-matching peer. Otherwise, messages may not reach the intended destination and have a negative impact on the querying peer, especially if the information is required within short time. To evaluate our knowledge-based routing mechanism, we have created an experiment to show how the accuracy of knowledge affects peers in terms of their performance in completing a task.

4.1. Resource collection game

The task itself is based on a game we have developed, whose objective is to collect as many *scoring* resources as possible. These scoring resources can be obtained by collecting *consumable* resources from a 2-dimensional grid. Each player (a peer) is able to move along adjacent grid positions. Fig. 1 shows various aspects of the game.

Fig. 1a shows the players, their roles and how they are connected to each other in a P2P network. Fig. 1b is an example grid world arranged in a 5×5 grid with a selection of resources, gold (g), crop (c) and tree (t), with player positions, A-I (first letter of a player's name). Fig. 1c shows which roles can convert a game resource from the grid world, *consume*, into a scoring resource, *produce*. Apart from the game-playing roles (banker, farmer and lumberjack), we have another role called *security*. Peers with this role do not play the game, but act as *directories* (or super-nodes) if a player requires to know where a cell (grid position) holds a given game resource. As the grid world is explored by the players, security peers will discover what each cell has and be able to assist players if they require help.

[1] `http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html`



(a) Topology with player names and roles

g/A		g		t
t		g	D	E
G	B		g	c
c	c/F	H	c/C	
	b	c	t/I	

(b) Grid of player positions and resources

Role	Consume	Produce
banker	gold	money
farmer	crop	food
lumberjack	tree	wood
security	(cell)	(resource)

(c) Roles with resource conversions

Fig. 1: Example of resource collection game

Each resource type can only be collected by a particular *role*. This means if a player has the appropriate role and is standing where a matching resource is, the player can *gather* the resource. Therefore, for players to obtain different resources, they must communicate to organise how to obtain the other types of resources. The reason for limiting one type of resource to be collected per role is that people tend to specialise in one type of job. This also allows us to show how the players are able to query for information about other resources based on the other players' roles.

There are two ways a player can obtain other types of scoring resources: a player can either *coordinate* or *exchange* with another player. Coordination involves agreeing with another player to rendezvous at a grid position in which the required resource is. The other player must be able to *gather* the consumable resource for itself, i.e. the other player has the correct role, but will give the resulting scoring resource to the requesting player. The other option is to exchange scoring resources with another player. This involves giving a resource, typically the resource a player can gather by itself, in exchange for another resource through the messaging system. Although an exchange for any resource can be made with any player, it is best to exchange for a particular resource with a player which has the role to gather the requested resource for itself. Details of coordination and exchange among players/peers are outside the scope of this paper and are not explored in our experiment.

4.2. Encoding gameplay

As mentioned in Section 3, we use an ontology to represent knowledge used to infer how to propagate information through the use of roles and the view of the virtual network. We also represent in our knowledge base the knowledge needed to play the game, that is the *rules* and *game state*, along with the routing ontology so that when a game rule is to be applied, any missing information to apply the rule can be queried for. For example, if a player wanted to move to an unknown cell, before a movement rule is applied, a query is created and sent to find out what is at the cell before moving to it. Also, the game rules can use knowledge from the virtual network. For example, to initiate coordination or an exchange with another player, the virtual network can be consulted to select a peer with the best role to obtain the desired game resource. Being able to combine game knowledge with routing knowledge allows automated querying for information in order to complete a task.

When using a knowledge base, it is important to ensure that the represented knowledge captures the domain knowledge, i.e. the game concepts, so that when new knowledge is inferred by reasoning (as well as game rule applications), the inferred knowledge also conforms to the domain knowledge. As the complexity of a knowledge base increases, in terms of concepts used to represent knowledge and the relationships among them, it becomes more difficult to ensure inferred and modified knowledge correctly captures all aspects of the domain. This means, in terms of our game and routing mechanism, as more knowledge is obtained about the 'game world' and the network, the higher the possibility that this knowledge is incorrect. Incorrect knowledge can occur due to initial game design being wrong or if knowledge becomes out-of-date over time and is not updated. We therefore examine the effects of varying accuracies of knowledge on the performance of a peer during gameplay.

5. Evaluation

For our evaluation, we ran a version of the game described in Section 4 using knowledge bases with varying accuracies. Whilst it would be ideal to have various knowledge bases to represent different accuracies, we cannot control how accurate a knowledge base will be for evaluation. Therefore, we use a success rate to determine when a reference knowledge base (a knowledge base of the game and routing mechanism engineered to the best of our ability) produces an accurate answer. An accurate answer in this case is returning a peer which has the best chance at answering a query. As mentioned in Section 4, roles are used to determine which peers to attempt first in order to answer a query regarding a type of resource. The success rate determines how likely a relevant peer will be given as an answer compared to a peer which is unlikely to answer the query. This simulates the accuracy of a knowledge base in terms of how much knowledge is correct. We use five categories to determine knowledge base accuracy: 0% where an incorrect answer is always returned, 25% where there is a one in four chance of a correct answer being returned, 50% where there is a one in two chance of a correct answer being returned, 75% where there is a three in four chance of a correct answer being returned and 100% where a correct answer is always returned.

5.1. Set up

To measure the performance of a peer, we use two different measurements: *satisfaction* and *score*. Satisfaction is a measure of successful replies received against the total number of queries sent and score is the total number of scoring game resources collected. These measurements are recorded for each peer, since they make use of their own representation of the game state (in their knowledge bases).

In our game, we do not impose a limit for collecting game resources. However, we cannot let a game run forever, or the experiment would never terminate. During a game, each player can perform one action at a time: *move*, *gather*, *coordinate* or *exchange*. We therefore place a limit on the total number of actions allowed to 300. This limit ensures that all players have a good chance to gather resources for themselves and to communicate with other players in the network to achieve better scores.

To acquire a suitable sample for statistical analysis, we use 100 peers arranged in a $k = 4$ *closed graph* topology, which means all peers will have four neighbours. The topology looks similar to the one shown in Fig. 1a, but peers along the edge of the graph will 'wrap round' to the opposite edge, ensuring all peers have four neighbours. In addition to using 100 peers in a closed graph $k = 4$ topology, for each independent variable (the knowledge base accuracies), we run each experiment 20 times to eliminate any outlying results.

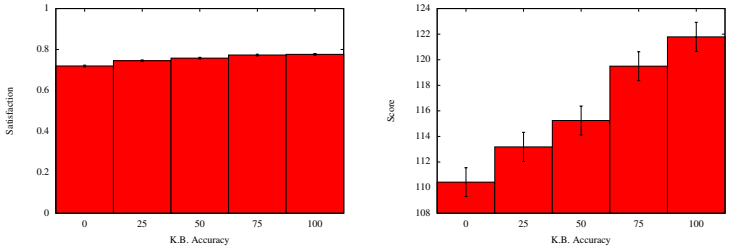
To control this amount of peers, we use *software agents* to control the P2P mechanism and game playing. The agents are designed to act as human players to the best of their ability. This means each agent will focus on gathering game resources for themselves first and then attempt to coordinate and exchange with other players. Each agent agent also has a level of *patience* to wait for replies to queries, including finding information about a grid position and attempts to coordinate with another player. If replies to these queries do not arrive within the patience value, then the agent will assume no response which will lower its satisfaction. Even if a successful reply arrived much later, the knowledge from the reply will be too old for the querying player to benefit from.

Each agent (and its peer) is assigned a random role from six possibilities – five game playing roles (similar to those shown in Fig. 1c) and a security role. We also have five different types of game resources to collect from a 20×20 grid, which is randomised for each game. With each experiment having 2000 peers (20 runs × 100 peers), we will have approximately 1667 peers ($2000 - 2000/6 \approx 1667$) for statistical analysis. We use these sample sizes to calculate the *standard error of the mean* for the satisfaction and score metrics to show the precision of our mean values.

Our hypotheses for this experiment are as follows:

- **Hypothesis 1** There is a difference in performance as knowledge base accuracy changes.
- **Hypothesis 2** A knowledge base which provides correct answers will improve the performance of peers.

The rationale behind our hypotheses is since better knowledge bases will lead to correct answers, this will improve the success rate of sending messages to the most appropriate peers (since knowledge about the game is adequately represented). This success rate will lead to better satisfaction and scores among the peers as positive replies will be received and efficient tasks will be carried out.



(a) Mean Satisfaction per Knowledge Base Accuracy (with Standard Error of the Mean)

(b) Mean Score per Knowledge Base Accuracy (with Standard Error of the Mean)

Fig. 2: Knowledge Base Accuracy Results

5.2. Results

Table 1 shows the results of the mean satisfaction and score among all significant players (excluding players with the security role) categorised by the knowledge base accuracy. These results are also shown in Fig. 2.

Table 1: Knowledge base accuracy results with population and standard error of the mean (SEM)

Accuracy (%)	Population	Satisfaction		Score	
		Mean	SEM	Mean	SEM
0	1681	0.719	0.004	110.428	1.129
25	1649	0.745	0.004	113.184	1.140
50	1668	0.758	0.004	115.246	1.133
75	1682	0.773	0.004	119.495	1.128
100	1650	0.776	0.004	121.788	1.139

Whilst the satisfactions and scores appear to be too close to draw any conclusions, we conducted a one-way MANOVA to evaluate the performance of the peers as accuracy of the knowledge base increased. Overall, as accuracy changed, the performance of the peers differed significantly ($F_{8,16650} = 21.945$, $p < .0001$; Wilks' $\Lambda = .979$, partial $\eta^2 = .010$); satisfaction values differed significantly to each other ($F_{4,8325} = 35.516$, $p < .0001$; partial $\eta^2 = .017$) and scores differed significantly to each other as well ($F_{4,8325} = 16.581$, $p < .0001$; partial $\eta^2 = .008$). We conducted a Tukey-Kramer post-hoc test on the satisfaction and score values to see how significant the change in values were across the accuracies of the knowledge base. We found that some neighbouring accuracies do not have significantly different values as each other: 25% and 50% and 75% and 100% do not have significantly different satisfaction values to each other ($p > .05$). Every other comparison possible have significantly different satisfaction values ($p < .0001$), but 50% was only slightly significantly lower than 100% ($p < .01$) and 50% was only slightly significantly lower than 75% ($p < .05$). Significant differences in scores share some similarities with satisfaction differences: 0% and 25%, 25% and 50%, 50% and 75% and 75% and 100% are not significantly different ($p > .05$). Other combinations of score values across the accuracies are significantly different ($p < .0001$), but there are a couple of mildly close differences; 0% was only slightly significantly lower than 50% ($p < .025$) and 25% was significantly lower than 75% ($p < .0005$).

These results allow us to draw the following conclusions:

- **Hypothesis 1 is supported** by our experiment. In most comparisons between pairs of knowledge base accuracies, the higher accuracy leads to significantly higher satisfaction and score. This is typically true in wider gaps of accuracies, such as between 0% and 100%, compared to narrower gaps, such as 75% and 100%.
- **Hypothesis 2 is supported** by our experiment. Similarly to Hypothesis 1, there is the argument that small increases in accuracy do not improve performance. However, in larger gaps, for example, between 0%, 50% and 100%, this is the case. We could evaluate several accuracies, even to infinitesimal ranges, but we believe our selection of prominent accuracies showcases our experiment and evaluation of hypothesis.

6. Conclusions, discussion and future work

In this paper, we have discussed an alternative approach to propagating information over a P2P network using roles. We have also shown how application knowledge, in this case game data, can be used alongside routing knowledge to automate query creation for retrieving missing knowledge to apply a game rule. Finally, we conducted experiments which show how our proposal performed under different controlled scenarios. The results from our experiment will help us refine our mechanism to handle poor knowledge and to operate in different types of domains.

Although it may be obvious that as knowledge accuracy increases, peer performance increases as well, it confirms that knowledge of the game concepts and structure of the network is important throughout the life of a game; from initial design of the game rules, how game data are used throughout a game and how knowledge of the network is formed in relation to the game knowledge. Any mistakes formed during knowledge creation can have an effect on peer performance, so it is important to ensure it is correct to begin with and remains correct throughout a game.

In our experiment, we assumed that all peers are *trustworthy*, meaning they will not deliberately provide false information. In games, this can be a problem since providing false information can lead to *cheating* and ruining the game experience for all players. Comparing genuine inaccurate knowledge with deliberate false information is a challenge and is a problem that should be explored. One possibility to prevent cheating and also eliminate the possibility of inaccurate knowledge is to have other peers (similar to 'security peers' discussed in Section 4) check information being sent between peers and what those peers currently know of the game state. If wrong information is sent from a peer with an inaccurate view of the game state, then either it could be updated to the most accurate view or be told to disconnect to prevent further inaccurate actions. However, if wrong information is sent from a peer with an accurate view, then it is likely that peer is cheating and can be punished accordingly.

References

1. Taylor, I., Harrison, A.. *From P2P and Grids to Services on the Web*. Springer; 2008.
2. Stoica, I., et al. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans Netw* 2003;.
3. Hoschek, W.. A unified peer-to-peer database protocol. In: *Proceedings of the International IEEE/ACM Workshop on Grid Computing*. 2002, .
4. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.. The many faces of publish/subscribe. *ACM Comput Surv* 2003;.
5. Clarke, I., et al. Freenet: A distributed anonymous information storage and retrieval system. In: *Int. Workshop on Designing Privacy Enhancing Technologies*. ISBN 3-540-41724-9; 2001, .
6. Jin, X., Chan, S.H.G.. Unstructured peer-to-peer network architectures. In: *Handbook of Peer-to-Peer Networking*. Springer US. ISBN 978-0-387-09751-0; 2010, .
7. Wehrle, K., Götz, S., Rieche, S.. Distributed hash tables. In: *Peer-to-Peer systems and applications*. Springer; 2005, .
8. El-Ansary, S., Haridi, S.. An overview of structured p2p overlay networks. *Handbook on theoretical and algorithmic aspects of sensor, ad hoc wireless, and peer-to-peer networks* 2005;.
9. Nejdl, W., et al. EDUTELLA: A P2P Networking Infrastructure Based on RDF. In: *Proc. of the 11th Int. Conf. on World Wide Web*. 2002, .
10. Castano, S., et al. HELIOS: A General Framework for Ontology-Based Knowledge Sharing and Evolution in P2P Systems. In: *DEXA '03*. 2003, .
11. Bonifacio, M.. A peer-to-peer solution for distributed knowledge management. In: *Semantic Web and Peer-to-Peer*. Springer; 2006, .
12. Mika, P.. A methodology for distributed knowledge management using ontologies and peer-to-peer. In: *Semantic Web and Peer-to-Peer*. Springer; 2006, .
13. Reychav, I., Weisberg, J.. Sharing knowledge in virtual communities. *Virtual Communities: Concepts, Methodologies, Tools and Applications* 2011;.
14. Yang, S.J.H., Chen, I.Y.L.. A social network-based system for supporting interactive collaboration in knowledge sharing over peer-to-peer network. *Int J Hum-Comput Stud* 2008;doi:10.1016/j.ijhcs.2007.08.005.
15. Watts, D.J., Strogatz, S.H.. Collective dynamics of 'small-world' networks. *Nature* 1998;.
16. Bharambe, A., et al. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. *SIGCOMM Comput Comm Rev* 2008;.
17. Gibson, M., Vasconcelos, W.. Real-time information querying over peer-to-peer networks using timestamps. In: *5th Int. Conf. on Agents and AI*. 2013, .
18. Glinka, F., et al. RTF: A Real-Time Framework for Developing Scalable Multiplayer Online Games. In: *Proc. 6th ACM SIGCOMM workshop on network and system support for games*. ISBN 978-0-9804460-0-5; 2007, doi:10.1145/1326257.1326272.
19. Hu, S.Y., Liao, G.M.. Scalable peer-to-peer networked virtual environment. In: *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. ISBN 1-58113-942-X; 2004, .
20. Feldman, M., Chuang, J.. Overcoming free-riding behavior in peer-to-peer systems. *ACM SIGecom Exchanges* 2005;.