A Highly Flexible Data Structure for Multi-level Visibility of P2P Communities*

Debmalya Biswas[1] and Krishnamurthy Vidyasankar[2]

[1] IRISA/INRIA, Campus Universitaire de Beaulieu, Rennes, France 35042
dbiswas@irisa.fr
[2] Dept. of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada A1B 3X5
vidya@cs.mun.ca

Abstract. Peer to Peer (P2P) communities (or "interest groups") are referred to as nodes that share a common interest. Each peer in the system claims to have some interests and, accordingly, would like to become a member of these groups. The available interest groups are arranged according to a hierarchical semantics ontology, and managed with a semantic overlay network. P2P community structure is highly dynamic: a peer may be added to or deleted from a community; communities may be added or deleted; communities may be merged or split; and sub-communities may become parent-level communities and vice versa. In this paper, we propose a highly flexible multi-level data structure to capture the visibility aspect of P2P communities. The data structure is simple, facilitates dynamic changes easily and efficiently in a decentralized fashion, and is highly scalable.

1 Introduction

Over the past few years, Peer to Peer (P2P) systems have gained widespread acceptance as a result of their decentralized control, high scalability and availability. However, their commercial use has been mostly restricted to information/file sharing systems. As a result, work has already been initiated towards the use of P2P systems for collaborative work [1,2]. A related application area where we also need to consider "groups" of peers is that of Interest Groups, e.g., Yahoo Groups [3]. Basically, the peers in an interest group share some common interests. The notion of P2P Communities [4] has been proposed to model such interest groups. We generalize P2P communities as an abstraction for a group of peers, which work collaboratively to perform a specific task or share some common interests.

By default, each peer has knowledge of (visibility over) the rest of the peers in its own community. Now, for a community to grow, it needs visibility over other

* D. Biswas's work is supported by the ANR DOCFLOW and CREATE ACTIVEDOC projects. K. Vidyasankar's work is supported in part by the Natural Sciences and Engineering Research Council of Canada Discovery Grant 3182.

S. Rao et al. (Eds.): ICDCN 2008, LNCS 4904, pp. 363–374, 2008.

communities and their peers. Similarly, a peer would like to have information about communities catering to "related" interests. Current P2P systems either function as independent entities (peers/communities) or assume that each peer is aware of all the other peers and communities ([8]). Allowing each peer to have full information, about all the other communities and their peers, is not a practical solution. For dynamic and heterogeneous environments such as P2P systems, trust and anonymity issues may force a peer (or community) to be restrictive in the visibility it allows to others. Thus, a fundamental issue to address in P2P communities is how to capture the visibility a peer (or community) has over the other peers and communities.

Towards this end, we introduce a visibility graph formalism for P2P communities. The communities are often organized hierarchically corresponding to a hierarchical semantics ontology, for example, as shown in Fig. 1. Our formalism is a multi-level graph to facilitate community visibility at different levels of the hierarchy. Note that the P2P community structure is highly dynamic: a peer may be added to or deleted from a community; communities may be added or deleted; communities may be merged into bigger ones or split into smaller ones; and sub-communities may become parent-level communities and vice versa. The multi-level visibility graphs facilitate performing query evaluation and structural updates in a decentralized and scalable fashion to accommodate the inherent dynamism.

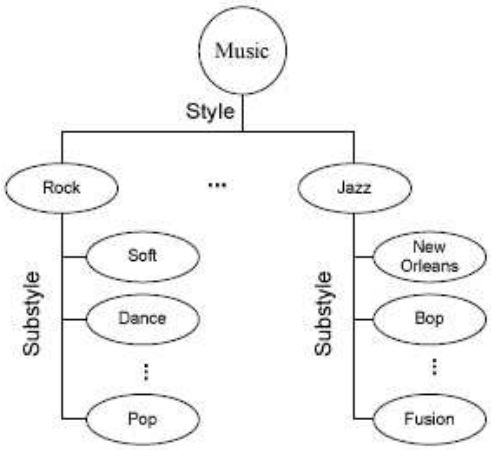

Fig. 1. A sample hierarchical ontology

The rest of the paper is organized as follows: The visibility graph formalism is introduced in section 2. Sections 3 and 4 outline algorithms for query evaluation and for performing structural updates on visibility graphs, respectively. In section 5, we discuss some implementation details of visibility graphs. Section 6 presents some related work. Section 7 concludes the paper and provides some directions for future work.

2 Visibility Graph

2.1 Multi-level Visibility

We present a graph model to represent the visibilities of the peers and communities in a P2P system. Let $\mathcal{P}$ be the set of peers and $\mathcal{C}$ be the set of communities in a P2P system S. Peers are represented by nodes. We use the same notation to refer to the nodes as well as the peers they represent. An edge between peers P_1 and P_2 indicates that P_1 is visible to P_2, and vice versa. The visibility is assumed to be symmetric in this paper. Peers are grouped into communities, which may further be grouped into higher level communities, and so on. The communities are usually arranged according to one or more hierarchical semantics ontologies. Fig. 1 shows a sample hierarchy of music related interests (as given in [5]). Fig. 2 illustrates a hierarchical organization. At the bottom level, there are several communities. We call them level-1 communities. We use single (solid, and different dotted and broken) lines for the intra-community edges. The communities are grouped into three higher level communities. In each group, connections between communities are shown in double line edges. These are level-2 communities. All these groups of communities belong to an even higher level (level-3) community shown in triple line edges. For example, Fig. 2 may correspond partly to Fig. 1, with level-1 communities of Soft, Dance, Pop, etc., level-2 communities of Rock, Jazz, etc., and level-3 community of Music.

In our model, we represent a parent-level community through certain peers of its children communities called *seers*. The seers are connected by edges identifying the community. In Fig. 2, the double line edges connect seers of level-1 communities, and similarly, triple line edges connect seers of level-2 communities. Thus, the actual connections will be as shown in Fig. 3. We note that each node will be incident to single line edges (as long as its community has at least two peers). Then, some nodes may be incident to other (double line, triple line, etc.) edges too. In Fig. 3, there are nodes incident to (i) single line and double line edges, (ii) single, double, and triple line edges, and also (iii) single and triple line edges. That is, a peer may be a seer for a higher level (e.g., level-3) community, and not for a lower level (e.g., level-2) community. Thus, the edges of the visibility graph are of different level communities in the hierarchy.

2.2 Data Structure

Let $\mathcal{H}$ be a set of hierarchies that may exist in a P2P system S. We confine our initial discussion and the graph formalism to one hierarchy $H \in \mathcal{H}$. Fig. 4 shows the hierarchy used in our previous example. We use Fig. 4 to illustrate some notations and concepts. Each node in the figure represents a community at some level of the hierarchy. We assume that each peer forms its own *unit community*. We call this level-0 community. The nodes, and the corresponding communities, are named locally (within that level) and globally with the sequence of local labels of nodes in the path from the root to that node. For example, e is the local name and $a/b/e$ is the global name of a node in the figure. We use the

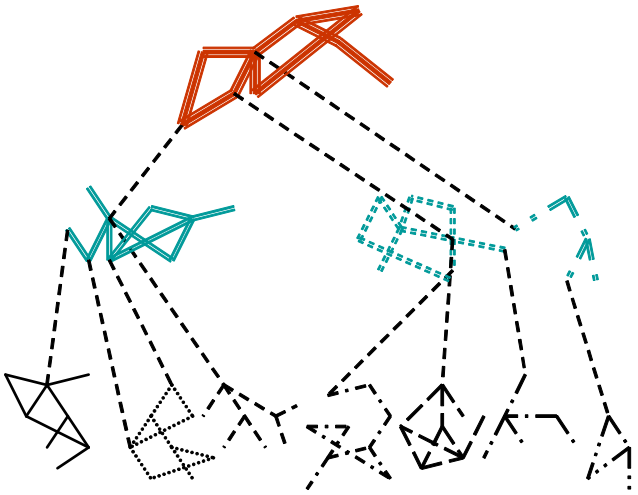

Fig. 2. Hierarchical visibility

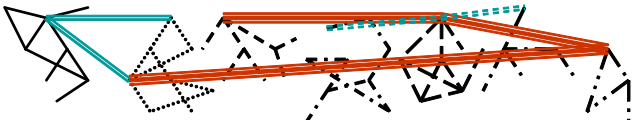

Fig. 3. Actual connections of the hierarchy in Fig. 2

global name to indicate the corresponding path also. We define a prefix α^p of a global name α as the sequence of labels of the nodes in a prefix of the path corresponding to α. On the same lines, we define an extension α^s of a global name α as the sequence of labels of the nodes of a global name β which contains α as one of its prefixes, that is, $\exists\beta, \beta^p = \alpha$. For example, for α equal to $a/b/e$, a/b and a are the prefixes, and $a/b/e/P_1$ and $a/b/e/P_2$ are extensions. We also denote the *immediate* prefix by α^I and an immediate extension by α^E. For $a/b/e$, the immediate prefix is a/b, and both the extensions in Fig. 4 are immediate extensions. Finally, for a global name α, we use ${}^*\alpha$ and α^* to denote the set of its prefixes and extensions, respectively.

We note that only the leaf nodes in H (Fig. 4) correspond to peers in S. They have labels as in the figure. (Labels of only some nodes are shown for easy readability.) All non-leaf nodes are virtual. Essentially, H describes all (lower and higher level) communities in S. Each peer belongs to several, hierarchically related, communities. To be precise, a peer with label α is a member of all the communities with labels in ${}^*\alpha$.

For each node (with local or global name) α in the hierarchy, we define two communities. The first one is the α-*full-community*, denoted with α in square brackets as $[\alpha]$-community. This consists of all the peers in the subtree rooted at α. For example, $[b]$-community membership is $\{P_1, P_2, P_3, P_4, P_5\}$. The

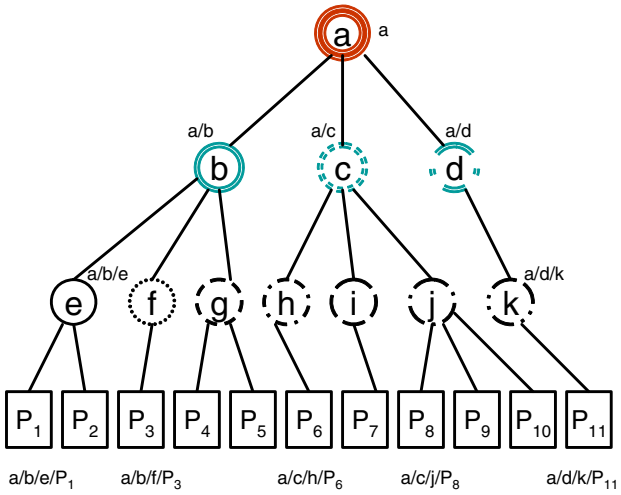

Fig. 4. Complete hierarchy with peers

second is *α-seer-community*, denoted with α in parentheses as (α)-community. For non-leaf nodes α, this will contain one or more peers from *each* of its children communities, that is from those of each immediate extension of α; these peers will be *seers* of the respective communities *for* $[\alpha]$-community. For example, (b)-community membership could be $\{P_1, P_3, P_5\}$. For leaf nodes α, the local name will be that of the respective peer, say P_i, and both the full and the seer communities will consist of just that peer. Note that each level-1 community is a full as well as a seer community. Now, a peer can be a seer for several ancestral communities. For a community C, we define a *C-graph* as the graph with node set consisting of members of C and edges, called C-edges, depicting the visibility among the members. When C is a seer community, we refer to the graph as *C-seer-graph* also. In our model, we require that each seer graph is connected.

Definition. For a P2P system S, with peers $\mathcal{P}$, hierarchy H, and related set of (full) communities $\mathcal{C}$, a *global visibility graph* is the union of C-seer-graphs of all communities C in $\mathcal{C}$ such that each seer graph is connected.

Example: A visibility graph for the P2P system in Fig. 4 is shown in Fig. 5. Here:

- $\mathcal{P}$ is $\{P_1, P_2, ..., P_{11}\}$;
- $\mathcal{C}$ is $\{[a], [b], ..., [k], [P_1], [P_2], ..., [P_{11}]\}$;
- $[P_2]$ is the unit community containing peer P_2, and the corresponding seer community (P_2) also contains P_2;
- Both $[e]$- and (e)-communities have peers P_1 and P_2; Similarly, for each of $\{f, g, ..., k\}$, their full and seer communities have all their children shown in the figure;

- Each of $\{[a]\text{-},[b]\text{-},[c]\text{-},[d]\text{-}\}$ communities have all the peers in the leaf level of the corresponding sub-trees in the hierarchy; For example, $[b]$-community membership is $\{P_1, ..., P_5\}$.
- The seer community membership is as follows: (b)-community has $\{P_1, P_3, P_5\}$, (c)-community has $\{P_6, P_7, P_8\}$, (d)-community has $\{P_{11}\}$, and (a)-community has $\{P_3, P_9, P_{11}\}$.

In this example, the connected graph of each seer community is a tree, in fact, a path. Note that P_8 is a seer for (c)-community but P_9 is the seer for (a)-community. That is, different peers of a sub-tree may be seers for different ancestral communities.

We point out that the visibility graph does not show the member peers of higher level full communities explicitly. However, the property that for every full community at every level its corresponding seer community must have at least one peer from each of its children communities guarantees that all the members of that community can be accessed if needed.

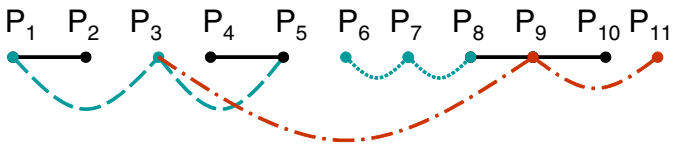

Fig. 5. Complete visibility graph of Fig. 4

3 Searching the Visibility Graph

Any search involves searching one or more communities. Search within a community typically involves flooding, that is, searching each peer in the community. Common search methods, when the peers within a community are arbitrarily connected, are Breadth First and Depth First Searches, with or without using a spanning tree. A search may be initiated from any node. It may also be initiated, in parallel, from several nodes. In each case, the search results may be forwarded to one or more initiating nodes, to some other nodes, or even to all the nodes. In this paper, we will neither be concerned with the actual search method nor with any specific way of forwarding the results. We denote searching a community C as C-search.

We explain a general search procedure with the graph in Fig. 5. Any query is initiated at a peer. Suppose a query q is initiated at P_2. Then, first, the (P_2)-search and, if necessary, an (e)-search will be performed. If q cannot be evaluated within (e)-community, then P_1 (which is the seer of $[e]$-community for (b)-community, will initiate a (b)-search. This will involve the graph consisting of P_1, P_3 and P_5. Then, P_5 may suggest that a (g)-search is appropriate, involving peers P_4 and P_5. On the other hand, P_3 may suggest that (the higher level) (a)-search, involving P_3, P_9 and P_{11} may be appropriate. Taking up one or more suggestions and continuing the search can be done either in a centralized or distributed fashion.

Thus, in comparison to traditional query resolution via flooding, our method provides the following benefits: Privacy and security constraints are maintained as queries are only forwarded to visible (trusted) peers. On the other hand, each peer may be a member of different seer communities at different levels. And, based on the information it has on each of these communities, it may be able to direct the search to any of these levels. For example, in the above search process, P_3 will be involved in the searches of (P_3)-, (f)-, (b)- and (a)-communities. Suppose (P_3)-search is the first one. The next search could be (a)-search, then (f)-search, etc. This allows the searches to be mixed rather than being strictly top-down or bottom-up. In (P_3)-search, its role is an individual peer; in (a)-search, its role is a seer for $[b]$-community, and in that capacity it is supposed to utilize some summary information of the $[b]$-community; and so on.

3.1 A Specific Example

We consider simple hierarchical path queries having the following syntax:

$I :=$ An area of interest in the underlying (hierarchical) ontology O.
$Path := \epsilon | I/Path$
$Query := (n)Path | (Query \wedge Query) | (Query \vee Query)$

where n refers to the *number of nodes* to retrieve satisfying the $Path$ criterion. $I/(sub\text{-})I$ refers to a pair of parent-child interests in O. A sample query is given below:

$Query\ q.\ (1)music/classic \wedge ((1)music/jazz/fusion \vee (2)music/rock/soft).$

Query q can be interpreted as follows: Find a peer interested in $music/classic$ and a peer interested in $music/jazz/fusion$, or a peer interested in $music/classic$ and two peers interested in $music/rock/soft$.

Given this, a search to evaluate query q over the global visibility graph (Fig. 5) would be as follows. (We assume that a, b, c and d correspond to the $music$, $music/rock$, $music/jazz$ and $music/classic$ communities, respectively.)

We assume that q is submitted at peer P_3.

1. P_3 initiates an (a)-search, that is, evaluates q within the (a)-community.
2. If unsuccessful, P_3 chooses nodes in the (a)-community (peers P_9, P_{11}, and peer P_3 itself) for propagating (the entire query or parts of) q for further evaluation. Peer P_3 makes this decision based on the "similarity" between the path criteria in q and the global names of the nodes in the (a)-community. Here, P_3 would choose itself for the (sub)query $(2)music/rock/soft$, while P_9 would be chosen for $(1)music/jazz/fusion$ and P_{11} for $(1)music/classic$.
3. The nodes P_3, P_9 and P_{11} initiate a (b)-search, (j)-search and (d)-search, respectively. In the course of (j)-search, P_9 might find a need for, and initiate, a (c)-search.
4. This continues until the appropriate peers are found, or all the peers in H have been explored (in which case, there may not exist peers satisfying the query).

4 Visibility Structure Changes

P2P systems are highly dynamic in nature, with peers being added to or deleted from communities, communities being added to or deleted from higher level communities, merging and splitting of the communities, etc. All these operations change the visibility graph. Our formalism facilitates these changes easily. We outline the general procedures in the following, with examples from the hierarchy of Fig. 4, and Fig. 5.

I. Basic operations

(a) *Adding a peer P to a community C.* Add the node P and a C-edge between P and some node already in C. Note that the addition of this single edge is sufficient to keep the new C-graph connected. Of course, additional edges can be added too. We restrict our description to the minimum requirements.

(b) *Deleting a peer P from a community C.* This may require several operations: (i) P is deleted from the C-graph; (ii) If this disconnects the C-graph, then additional C-edges, between other peers in C, are added to keep the C-graph connected; (iii) Suppose P is a seer of a sub-community C' of C for C. If P is the only such seer, then some other peer P' of C'-community should be designated as a seer for C and added to C.

(c) *Deleting a peer P from the P2P system S.* Then P must be deleted from every community it is part of (as a seer).

II. Other structural changes

Some of the other structural changes are:

- adding a community C to a higher level community C';
- deleting a community C from a higher level community C';
- merging two same level communities;
- splitting a community to two communities under the same parent community;
- merging two different level communities; etc.

These, and several other operations involving combinations of those considered above, can be executed. All these involve essentially some generic operations like adding a node to a graph, deleting a node from the graph but keeping the graph connected after the deletion, etc. Depending on how the graph (C-graph for community C) is maintained (as a tree, arbitrary connected graph, complete graph, etc.), these operations can be implemented efficiently. The key property of our formalism is that a node may be incident to C-edges of several communities C, and so the node may have to be deleted from some C-graphs and not from some others.

5 Implementation and Discussion

As we know, P2P systems are very highly decentralized. Peers are far apart, they are highly autonomous, they may join the system and drop out in an ad hoc

manner, their storage and processing capacities may be varied and limited, their availability and accessibility may be different at different times, communication between them may not be reliable, and so on. Therefore, implementation of the algorithms for both searching the peers for query evaluation and for modifying the structure of the visibility graph, outlined in the previous two sections, requires special attention. For instance, in any distributed implementation, the algorithms will be executed only lazily, that is, asynchronously. The algorithms can be fine tuned for "safe" execution. For example, when a community is deleted from another, and added to a different community, the addition part can be done first, and then the deletion. This may result, in the worst case, in a concurrent execution exploring more peers than necessary, but without loss of visibility.

We note that peers can execute their part of the algorithms autonomously. For example (as discussed in section 3), a peer which is a seer for several (higher level) communities could employ its own heuristics to decide on which order to explore the various communities. The selection strategy could be different for a different peer even for the same set of choices.

The core element of the implementation is the manipulation of C-graph for each community C in the hierarchy. Depending on the type of connectivity maintained, the amount of work in the manipulation will be different. Searching could be more systematic and efficient with tree structure. On the other hand, updates on the hierarchical structure may be implemented more efficiently with graphs which are not trees.

With our model, several C-graphs have to be manipulated. However, each can be manipulated independent of the others. So, the complexity of the implementation will not increase very much with the increase in the number of communities. With proper extensions to the underlying hierarchy, an increase in the number of peers in the P2P system can be accommodated by increasing the number of communities without substantially increasing the size of the corresponding seer-communities. Thus, our data structure and the algorithms are highly *scalable*.

At an abstract level, multi-level visibility seems close to the notion of database views [6]. For a set of relational tables in a database, a view allows us to summarize their data based on some characteristics. Here, the peers can be considered as data. The information on the various peers in a community can be summarized and kept in the seers of that community. However, since a peer can be a seer of a higher level ancestral community without being a seer of a lower level one, hierarchical relationships of the communities cannot be represented directly.

Having said this, there is also sufficient overlap between the two concepts, for a lot of the existing work for database/views to be used here, especially, with respect to query optimization and rewriting. Also, we have not considered *concurrent* query and update of the visibility graph which is very relevant, especially, for collaborative P2P systems. Database solutions [7] appear very attractive for the above as well.

Finally (as mentioned earlier), we consider visibility graphs as an abstraction to capture the visibility aspect of P2P communities, and we expect other

middleware aspects, e.g., security, monitoring, etc. to build on this abstraction. In the sequel, we briefly show how visibility graphs facilitate P2P security. Security for P2P communities is usually provided with the help of a Trust Management scheme [8]. In this scheme, each peer maintains the trust rating of all the other members in its own group (group-mates), based on its own dealings with them. For inter-community accesses, "when a member p requests to acquire resources from a member q of another community, it sends a request to q. q checks with the group-mates of p if p is trustable and what kind of access privileges it has. q then accepts or rejects the transaction with p". The above can be modeled using visibility graphs as follows: The underlying assumption, that the seer graph of each community is connected, allows a peer to monitor the activities of its group-mates (that is, to maintain their trust ratings). Inter-community access between peers $P_1 \in C_1$ and $P_2 \in C_2$ may only occur via an access path from a C_1-seer to a C_2-seer. Further, recall that visibility is symmetric. Thus, given such an inter-community access, P_2 can backtrack along the access path, and retrieve the trust rating of P_1 from the C_1-seer.

6 Related Works

The notion of P2P communities was introduced in [4]. They also represent visibility using intra-community and inter-community edges. They use only one type of inter-community edges. Our formalism uses different edges for (seer communities at) different levels of the hierarchy.

Further works [8], [9] and [10] have extended the P2P community notion as follows: [8] presents a Trust Management scheme for P2P communities. [9] discusses efficient discovery for P2P communities based on the "type" of communities, e.g., co-operative, goal oriented, ad hoc communities, etc. [10] presents a gossip based discovery mechanism for P2P communities. However, none of them consider the visibility aspect with respect to P2P systems.

No other work (that we are aware of) has attempted to formalize the visibility aspect for P2P systems. Some of the works which have touched upon this aspect are the following: [11] identifies real-life scenarios where there might be a need to deviate from the inheritance of access rights upwards through the hierarchy in a role-based access control system. [12] considers the visibility aspect with respect to sending publish/subscribe notifications for event based systems. [13] identifies the need for visibility across levels of a supply chain management system as follows: "The information required by downstream entities are mainly material and capacity availability information from their suppliers. The information acquired by an upstream entity is information about customer demand and orders. The depth of information penetration can be specified in various degrees, e.g., isolated, upward one tier, upward two tiers, downward one tier, downward two tiers, and so forth".

In previous works [14] and [15], we have studied the visibility aspect for hierarchical systems, especially, hierarchical Web Services compositions. In [14], we introduced the notion of Sphere of Visibility (SoV) to capture the visibility

aspect, and showed its application in the context of performing compensation under visibility constraints. However, [14] only considered vertical visibility (that is, visibility over ancestors and descendants) as compared to the more generalized notion of visibility presented in [15] (visibility over siblings, uncles, cousins, etc.). In [15], we also studied some inherent relationships which might exist among the SoV's of a group of providers, e.g., coherence, correlation, etc.

7 Conclusion and Future Work

In this paper, we have proposed a multi-level visibility graph formalism to capture the visibility of peers and communities in P2P systems. The formalism caters to the grouping of peers in a hierarchical community organization. The graph model accommodates, with equal ease, any number of levels in the hierarchy and any number of communities in each level.

In this paper, we have assumed that at a level of the hierarchy, the node sets of the children components are disjoint. When we consider multiple attributes, the resulting components (at the same level) may be overlapping. This can be accommodated in our model though the search and update algorithms would become more complicated. For instance, in the event of such overlapping it may no longer be sufficient to use the hierarchical path as the global name of a peer.

Further, the visibility graph formalism, and the search and update algorithms in this paper have been defined for a single underlying hierarchy. The scenario becomes more interesting as soon as we allow for multiple overlapping hierarchies. For example, in Fig. 1, in addition to the hierarchical classification by type, the peers may also be chronologically (again, hierarchically) classified by the interest in decades, years and months. Given this, a query evaluation, after some initial search with respect to the type hierarchy and determining the year of production, might switch to a search by the chronological hierarchy. Our formalism can easily be extended for multiple hierarchies. We leave this extension and the above issues as directions for future work.

References

1. Grid Computing, `http://www.ogf.org/`
2. Active XML (AXML) Systems, `http://www.activexml.net`
3. Yahoo Groups, `http://groups.yahoo.com/`
4. Khambatti, M., Ryu, K., Dasgupta, P.: Peer-to-Peer Communities: Formation and Discovery. In: PDCS. Proceedings of the 14th IASTED Intl. Conf. Parallel and Distributed Computing Systems, pp. 161–166 (2002)
5. Crespo, A., Garcia-Molina, H.: Semantic Overlay Networks for P2P Systems. Technical report, Stanford University, USA (2002)
6. Database Views, `http://en.wikipedia.org/wiki/View_(database)`
7. Abiteboul, S., Amann, B., Cluet, S., Eyal, A., Mignet, L., Milo, T.: Active Views for Electronic Commerce. In: VLDB. Proceedings of the 25th International Conference on Very Large Data Bases, pp. 138–149 (1999)

8. In, H.-P., Meintanis, K.A., Zhang, M., Im, E.-G.: Kaliphimos: A Community-Based Peer-to-Peer Group Management Scheme. In: Yakhno, T. (ed.) ADVIS 2004. LNCS, vol. 3261, pp. 533–542. Springer, Heidelberg (2004)
9. Akram, A., Rana, O.F.: Structuring Peer-2-Peer Communities. In: P2P. Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pp. 194–195 (2003)
10. Khambatti, M.S., Ryu, K.D., Dasgupta, P.: Push-pull gossiping for information sharing in peer-to-peer communities. In: PDPTA. Proceedings of the IASTED International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 1393–1399 (2003)
11. Moffet, J.D.: Control principles and role hierarchies. In: RBAC. Proceedings of the 3rd ACM Workshop on Role-Based Access Control, pp. 63–69 (1998)
12. Fiege, L.: Visibility in Event-Based Systems. Ph.D. Thesis, Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany (2005)
13. Lin, F.-R, Tan, G.W., Shaw, M.J.: Modeling Supply-Chain Networks by a Multi-Agent System. In: HICSS. Proceedings of the 31st Annual Hawaii International Conference on System Science, pp. 105–114 (1998)
14. Biswas, D., Vidyasankar, K.: Spheres of Visibility. In: ECOWS. Proceedings of the 3rd IEEE European Conference on Web Services, pp. 2–13 (2005)
15. Biswas, D., Vidyasankar, K.: Modeling Visibility in Hierarchical Systems. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 155–167. Springer, Heidelberg (2006)