A Hybrid Peer-to-Peer Recommendation System Architecture Based on Locality-Sensitive Hashing

Alexander Smirnov[*†], Andrew Ponomarev[*]
[*]St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, Russia
[†]University ITMO, St. Petersburg, Russia
{smir, ponomarev}@iias.spb.su

***Abstract*—Recommendation systems have become ubiquitous recently as they help to mitigate information overflow of the nowadays life. The vast majority of current recommendation system approaches are centralized. Although centralized recommendations have several significant advantages, they also have two main drawbacks: single point of failure and the necessity for users to share their preferences. In this paper, a system architecture of a peer-to-peer recommendation system with limited preferences disclosure is proposed. The proposed architecture is based on a locality-sensitive hashing of user preferences and an anonymized distributed hash table approach to peer-to-peer design.**

I. INTRODUCTION

Recommendation systems play important role in dealing with information overflow of the nowadays life. Most of the current recommendation systems have a centralized design. It is beneficent mainly because it allows to employ a broad spectrum of user preference models to predict future user behavior. It also puts all the relevant user information under the control of recommendation system provider allowing to perform various research activities on this data beside providing users with online recommendations.

Centralized approach has several drawbacks. First, it introduces a quandary about rights on the preferences data collected about a user. A user is usually not aware of what information a system collects about his/her behavior and cannot extract this information from the centralized system. Second, the centralization is in the matter of fact only partial. In other words, a user may communicate to several recommendation systems, sharing with each system some part of his/her preferences profile but all user preferences become spread between several recommendations with no chance of being united. It is not desirable, because a complete preferences profile can lead to recommendations that are more accurate. Third, centralization usually leads to single point of failure, but in modern computer systems, this drawback is usually alleviated by multilevel duplication and replication.

In this paper, a user-centric approach to recommendation systems design is examined. According to this approach, a user holds all his/her preferences on his/her own. This entirely removes the quandary about rights – a user fully controls his/her preferences storage. This can also remove preferences partitioning as all the user preferences become centralized in a device, controlled by the user. When in need of recommendations, a users' device queries other devices for them.

Decentralized recommendation systems carry two main advantages:

- the recommendations can be distributed among all users, removing the need of costly central server and enhancing scalability;

- a decentralized recommendation improves the privacy of the users, as there is no central entity owning private information of the users (though this is a subtle topic due to immanent security issues of peer-to-peer systems).

Albeit all enumerated issues of centralized recommendation design are addressed by user-centric decentralized recommendation system design, there are some other issues to be solved. To achieve in decentralized recommendation the same characteristics as of centralized one is an acute problem, because a large amount of distributed data need to be managed and resource usage need to be balanced.

In this paper, a recommendation system architecture that follows the user-centric approach is proposed. It is a structured peer-to-peer (P2P) network, where each peer corresponds to one user and holds preferences thereof. Recommendations are made by means of anonymized communication between peers. The proposed architecture provides limited preferences disclosure. It means that there is no way to reliably match ratings and a user network address having no global control over the entire P2P network. The proposed architecture is a hybrid P2P as it uses one special node for data-driven coordination that, however, is not used directly in the recommendation process.

The rest of the paper is structured as follows. Section 2 presents an overview of existing P2P recommendation systems and approaches. In section 3, locality-sensitive hashing approach to recommendations is discussed. Section 4 contains the description of the proposed recommendation

ISSN 2305-7254

system architecture. Main results are summarized in the conclusion.

II. Related Work

Peer-to-peer recommendation systems design is already addressed in literature.

In [1], [2] the P2Prec system is proposed. The idea of this system is to recommend high quality documents related to query topics and content hold by useful friends (or friends of friends) of the users, by exploring friendship networks. To disseminate information about relevant peers, they rely on gossip algorithms. For publishing and discovering services they use a distributed hash table.

The authors of P2Prec rely on two-level Latent Dirichlet Allocation to automatically model topics. At the global level performed by a bootstrap server a sample of documents is collected from peers and a set of topics are inferred. Then, at the local level, performed by each peer, local documents are analyzed with respect to common topics. Each user maintains a friendship network. A user enlarges the friendship network by accretion of new friends relevant to queries and having overlap with this users' friendship network.

To establish friendship and P2Prec relies on gossip protocols.

Key-word queries are routed recursively through friends networks, based on user trust and usefulness.

In a number of methods described in literature, overlay network structure based on similarity between nodes is built and recommendation algorithm is defined on this network (as in [3] and [1]). Recommendations perform a search among neighbors up to certain depth or certain similarity threshold.

One of the algorithms of aligning network structure to peer similarities is T-Man [4]. T-Man relies on the ability of a peer to measure how it «likes» peers. Having defined this relation, T-Man algorithm aligns the structure of the overlay network to juxtapose peers that «like» each other.

The similarity-based overlay network structure is extensively studied at [5] with the following result. It is shown by the authors that overlay topologies that are defined by node similarity have highly unbalanced degree distributions which have to be taken into account when load-balancing P2P recommendation network. They also propose algorithms with favorable convergence of speed and prediction accuracy taking load balancing into account. They consider collaborative filtering system where similarity of users measured as cosine similarity.

In the proposed architecture exact ratings are not exposed together with node identity, so there is no way to say how similar the two nodes are. Using locality-sensitive hash values it can only be said whether they are likely to be close enough or not.

Another approach is to rely on random walk search for similar nodes in ordinary P2P network, using some form of the flooding technique [6]. Similarly, in [7], it is shown that it is enough to take a random sample of the network and use the closest elements of that sample to make recommendations.

In [8] random walks approach to collaborative filtering recommendations is examined in the context of P2P systems. The authors argue that the effect of random walk in decentralized environment is quite different from the centralized one. They also propose a system where epidemic protocols (gossip protocols) are used to disseminate user similarity information. They start from the random set of peers and then in series of random exchanges compare their local-view with the local view of the remote node, leaving only the most similar peers in the local view (clustering gossip protocol). This process converges to form some overlay based on peers similarity. Then peers that are not farther than 2 hops from given are used to make recommendations.

In epidemic protocols (also known as gossip protocols), peers have access to a Random Peer Sampling service (RPS) providing them with a continuously changing random subset of the peers of the network. When a peer joins the network, her view is initialized at random through the RPS. Each peer also maintains a view of the network. Gossip protocols are fully decentralized, can handle high churn rates, and do not require any specific protocol to recover from massive failures.

There are also research papers where structured P2P networks are used. For example, in [9], [10] distributed hash tables are used to store ratings. The proposed approach stands close to this way except that ratings are not stored in a distributed hash table, instead a fast lookup capability provided by this kind of P2P architecture is employed for searching similar peers.

Most of the approaches involve sharing rating data between nodes, while in the proposed architecture it is avoided.

Privacy concerns are directly addressed in [11]. The authors propose a file sharing network where users exchange their data only with their friends and a recommendation system on the top of it. They propose privacy-conserving distributed collaborative filtering approach that is based on exchanges of anonymized item relevance ranks between peers. Their approach, however, allows only unary ratings (initially, the fact of owning a specific file).

Distributed recommendation systems are also analyzed in quite another context, seeking for efficient parallel implementations of centralized recommendation techniques. This research direction is entirely out of the scope of this paper.

III. LOCALITY-SENSITIVE HASHING FOR RECOMMENDATIONS

Locality-sensitive hashing (LSH) is a method that is widely used for probabilistic solution of k-NN (k Nearest Neighbors) problem. The idea of this method is to hash multidimensional objects in such a way that similar objects (w.r.t some distance measure defined on them) are likely to have the same hash value.

A. The idea of LSH

The problem of finding nearest neighbors is closely related to the recommendation system domain. The reason is rather straightforward and is based on an assumption that users that had similar preferences in the past are likely to have similar preferences now (and in the future). Therefore, if user preferences are represented as a numerical vector and some measure in that vector space is introduced that corresponds to preference similarity, then the problem of finding similar users translates into nearest neighbors search.

In this section a formal description of collaborative filtering recommendation method based on locality-sensitive hashing is provided.

Let $d_1 < d_2$ be a two distances according to some distance measure d. A family F of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every f in F [12]:

- If $d(a, b) \le d_1$, then probability that $f(a) = f(b)$ is at least p_1.
- If $d(a, b) \le d_2$ then probability that $f(a) \ne f(b)$ is at most p_2.

An important concept in the locality-sensitive hashing theory is amplification.

Given a (d_1, d_2, p_1, p_2)-sensitive family F, a new family F' can be constructed by either AND-construction or OR-construction.

AND-construction of F' is defined as follows. Each member of F' consists of r members of F for some fixed r. If f is in F', and f is constructed from the set $\{f_1, f_2, ..., f_r\}$ of members of F, $f(x) = f(y)$ if and only if $f_i(x) = f_i(y)$ for all $i = 1, 2, ..., r$.

As members of F' are independently chosen from F, F' is an (d_1, d_2, p_1^r, p_2^r)-sensitive family [12].

OR-construction of F' is defined as follows. Each member of F' consists of b members of F for some fixed b. If f is in F', and f is constructed from the set $\{f_1, f_2, ..., f_b\}$ of members of F, $f(x) = f(y)$ if and only if $f_i(x) = f_i(y)$ for all $i = 1, 2, ..., b$.

Similarly, F' is an $(d_1, d_2, 1 - (1 - p_1)^b, 1 - (1 - p_2)^b)$-sensitive family.

Generally, it is desirable that p_1 is as large as possible and p_2 is as small as possible. If p_1 is less then 1, then there is some possibility that similar objects will have different hash values. On the other hand, if p_2 is greater than 0, there is some possibility that distant objects will have similar hash values. Therefore, family F is chosen in such a way that p_1 is large (close to 1) and p_2 is small (close to 0). There is a finite set of well-studied locality-sensitive function families and the desired levels of p_1 and p_2 can not always be achieved with one "pure" family. This is where amplification comes into play.

If family F^{Ar} is obtained as AND-construction of r functions from family F, and G is then obtained as OR-construction of b functions from family F^{Ar}, then G is a $(d_1, d_2, 1 - (1 - p_1^r)^b, 1 - (1 - p_2^r)^b)$-sensitive family. Informally, AND-construction mostly lowers the initially low p_2 probability and subsequent OR-construction raises the initially high p_1 probability.

The idea of nearest neighbors search based on the LSH is described, for example, in [12], [13]. First, a hash family F (it is discussed in greater detail later on) is chosen and L ordinary hash tables are arranged. Each hash table corresponds to some hash function f^{Ar}_i, $i = 1,...,L$, where f^{Ar}_i is an AND-construction of r random functions from F. Every object x is stored to each of the L hash tables. Key is the $f^{Ar}_i(x)$ and value is either some identity of x or x itself. It is natural that several objects can fall into one hash table bucket.

When searching for the nearest neighbors of an object y, first, $f^{Ar}_i(y)$, $i=1,..,L$ is calculated and then all values from the corresponding hash maps are retrieved resulting in a set of nearest neighbor candidates. Precise distance to each of the candidates is then assessed and false positives are removed.

Particular choice of hash function family depends on data representation and distance function d. For Hamming distance a bit sampling locality sensitive hash was proposed in [14], for cosine distance a random projections method was proposed in [15], a well-performing hash function for Euclidean distance is proposed in [16].

In the proposed architecture random projections method is used, i.e. function f from F corresponds to one random hyperplane and checks whether point being hashed is above or under this hyperplane.

B. Recommendations generation

User-based rating prediction collaborative filtering system is a recommendation system that infers recommendations (unknown user ratings) from the similarity of users measured by the extent known user ratings coincide.

More formally, let r_{uj} be the rating assigned to the item j by the user u, which corresponds to how user u liked item j, or what was the subjective utility of j for u. Let U be the set of all users, I – the set of all items, I_u – the set of items that was rated by user u, and I_{uv} – the set of items rated by both

user u and user *v*. Usually, a user has ratings for relatively small number of items, $|I_u| << |I|$. User-based rating prediction collaborative filtering is one of the neighborhood methods, i.e. it relies on some similarity measure between users which is calculated based on common ratings $(sim(u, v) = f_s(\{r_{uj}, j \in I_{uv}\}))$ and estimate unknown rating r^*_{uj} based on known ratings r_{vj} and estimated similarities $sim(u, v)$.

In the recommendation systems research there were introduced several user similarity measures: Pearson and Spearman correlation coefficients, Jaccard similarity, Hamming distance, cosine similarity. The choice of similarity measure mostly depends on user rating encoding.

In this paper, a cosine similarity is employed as a similarity measure between users. Therefore:

$$sim(u,v) = \frac{\sum_{I_{uv}} r_{uj} r_{vj}}{\sqrt{\sum_{I_{uv}} r_{uj}^2} \sqrt{\sum_{I_{uv}} r_{vj}^2}}.$$

User ratings are normalized in such a way that $r_{uj} = 1$ corresponds to strong positive attitude of user u to item j, and $r_{uj} = -1$ corresponds to strong negative attitude respectively.

The prediction of an unknown rating r^*_{uj} requires the search of users v that are similar to r^*_{uj}, or the nearest neighbors of u according to cosine similarity measure.

Recommendation system using LSH follows the nearest neighbor approach. Known a set of hash values for some user u, it checks the respective hash tables and retrieves all users whose interests are likely (due to hash function properties) similar to u's. Then exact similarity may be assessed and high rated items of similar users are provided to u.

To sum it up, in the proposed system architecture, a profile of user u is a set of pairs (i, r_{ui}), where i are item identifiers. Each of L locality-sensitive hash functions is represented by b vectors, which dimensionality equals to the number of items known. After application of all these hash functions L b-dimensional binary vectors are obtained and stored into hash table.

IV. System Architecture

The proposed hybrid architecture enables personalized recommendations exchange with limited user preferences disclosure. In this section, target use cases are discussed, as well as components of the proposed system and scenarios that implement target use cases.

A. Functions

Recommendation systems may provide slightly different end-user features. Specifically, in this paper following recommendation functions are considered: a) rating prediction for a given item (or set of items); b) recommendations query.

Rating prediction for a given item (or a set of items) is involved when a user encounters some item and wants to assess if it is potentially interesting or useful for him/her. In this case, user passes the item (item identity) to recommendation system and recommendation system should return expected attitude of this user to this item. Certainly, a user is not required to perform this request intentionally by hand; some other program or GUI element acting on behalf of the user can mediate this action. Rating prediction request may contain several items. It may be very convenient in some situations. Though rating prediction for multiple items can always be implemented as a series of single item rating predictions, it is interpreted here as a use case extension, because in some circumstances rating prediction for multiple items is potentially more efficient that multiple separate single item requests.

Recommendations query is launched in quite another situation. Here, a user just wants to see some recommendations – maybe recommendations of new, previously unseen and actual items.

B. Components

In the proposed architecture, recommendation system is split into two parts: Peer-to-Peer (P2P) recommendations network and the Master node (Fig. 1). The Master node breaks the conceptual purity of the P2P design, making it a hybrid P2P system, but it does not play a significant role in the primary use cases of the system, namely rating prediction for the given item and recommendation query. Both enumerated earlier functions are implemented by P2P network solely and the Master node is responsible for synchronizing supplementary information between peers.

In Fig. 1 two types of connection between nodes are shown: connections between similar peers used to get recommendations are shown by solid lines, and occasional connections of peers to the Master node for retrieving supplementary information are depicted by dashed lines.

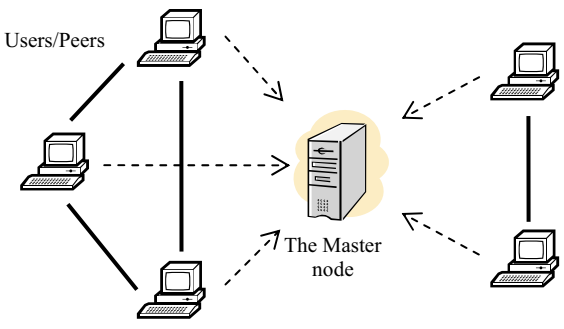

Fig. 1. Connections between nodes in the proposed architecture

1) Peer-to-Peer recommendations network: In the proposed architecture, each user corresponds to exactly one node (or peer – these terms are used here interchangeably). That node holds all the information about one user's preferences, ratings, browsing history etc, but does not share

this information with the other nodes, instead it shares only the locality sensitive hash value of this information in order to find similar users to query for recommendations.

P2P network is based on the Distributed Hash Table (DHT) [17] model widely employed in various P2P networks. The general idea of the DHT is rather straightforward. It holds a collection of key/value pairs scattered over a distributed set of nodes, supporting key/value pair migration in case node disconnection. DHT usually refers to a class of systems rather than to some specific system or algorithm.

Original DHT has some severe security vulnerabilities. To overcome these vulnerabilities a variety of secure and anonymous DHT lookup implementations was designed [18]. The proposed architecture relies on one of these anonymized implementations, namely Octopus. The idea behind most of secured DHT implementations is that all DHT lookups are made through other nodes accessible by anonymous path through anonymization relays. Each node in anonymization path knows only neighbor nodes, but does not know whether some request was originated in the neighbor node, or was passed from some other node.

The DHT in the proposed system is used as a set of hash tables needed for nearest neighbor search, as described in section III. Each key/value pair stored in the DHT holds information about one locality-sensitive hash value and the list of nodes corresponding to that hash value. As it was discussed in the respective section, several (L) hash tables are needed to perform nearest neighbor search. Each of the *L* tables uses its own locality-sensitive hash function. It is proposed to store all of these *L* hash tables in one DHT. In order to achieve this, key of the DHT pair should include global unique identifier of the locality-sensitive hash function and the value of that function.

Each node of the P2P network has a unique identifier, which is assigned to the node when it first connects to the network. In most DHT implementations, node identifier is a 160-bit value that is produced by applying SHA-1 to the network address of the node.

Before a node advertises itself in a DHT, it creates an anonymized path and uses the endpoint specification of this path as an address it tells to other nodes. These anonymized paths are created each time node connects network, resulting in different public identifiers of the same node.

As user preferences expressed in ratings are not changing very fast, it is reasonable for each node to locate through DHT and store other nodes with the similar profiles. Therefore, a new overlay network of similar users is formed over P2P network. It is important to differentiate between the three employed connection layers (Fig. 2). The first layer is the underlying network, which provides physical connection between P2P nodes. The second layer is DHT connection layer which provides DHT key search, key

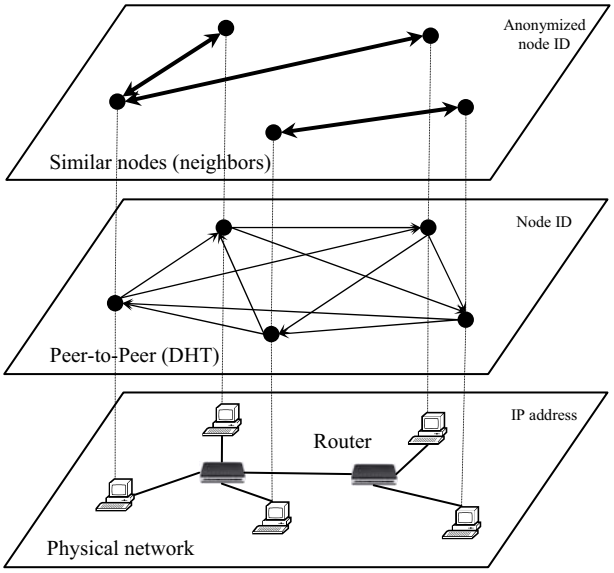

Fig. 2. Peer-to-Peer layers

redistribution etc. This layer is provided by links to adjacent nodes in structured P2P, so called “fingers”. The third layer is formed by connections between similar nodes, where similarity is interpreted like equality of locality-sensitive hashes.

It is important to note, that links to neighbor nodes in the third layer are not exactly identifiers of nodes in P2P network, but are entrances to anonymized paths to that nodes.

2) The Master node: The distributed nature of the proposed system causes one hindrance. LSH-based nearest neighbor search implies that when searching for neighbors of object *x*, all the locality-sensitive hash functions that were used to hash other objects and fill hash tables are applied to *x*. In the proposed architecture, an object being hashed is a vector of all ratings assigned by the user to different items of interest and hashing functions family is random hyperplane projections. To define a hyperplane the dimensionality of the space have to be known. In some cases, for instance, when rating storage is centralized, when ratings are immutable or all possible items are known in advance, knowing dimensionality is not a problem. However, in case of distributed rating storage when each node holds only ratings of one user, overall item space dimensionality can be found out only though communication between nodes. Dimensionality not only means the number of dimensions, but also their order. It is easy to see that if one user encounters items in the following order: (Item1:1, Item2:-0.5, Item3:1), and another user encounters and rates the same items in another order: (Item1:1, Item3:1, Item2:-0.5), then their hashes with hyperplane (0,1,0) would be different although ratings match perfectly.

Hence, it is needed to synchronize item space characteristics and random projection hyperplanes across all nodes. The problem of maintaining global shared state in P2P network is rather nettlesome, and there are numerous papers dedicated to this problem, e.g. [19],[20],[21]. In the proposed system this problem is addressed in a way similar to one from [22], sacrificing the P2P-purity of the system. It is the Master node that, first, collects all new items discovered and rated by peers, maintains their ordering and generates new locality-sensitive hash functions. So, each peer must connect to the Master node in two situations: first, to notify about some previously unknown item (which should become a new dimension), second, to get a new set of locality-sensitive hash functions. It must be noted, that there is no necessity in generation of new hash functions after each new assessed item. Using outdated hash functions with lower dimensions is still possible, but it gradually decreases the quality of recommendations. So, each user node collects new rated items (which was not assigned identifiers yet) and then sends batch of these items to the Master node. The Master node, in its turn, accumulates new items, and when there are many of them, assigns them an ordering and issues a new set of locality sensitive hash functions. It is also important that the new set is not an entire replacement of the previous, but contains only several new functions.

C. Scenarios

1) Rating prediction for a given item: Rating prediction in a node is possible only after this node has integrated into the P2P network and has located the nodes of the users with similar ratings (hereinafter these nodes are referred to as neighbor nodes). Let the neighbor nodes for the given one are stored in the Neighbors list. Then rating prediction for the item Item is performed by sending requests to each node from the Neighbors list passing the item identifiers.

Each neighbor node answers with rating estimation, or empty value if it can not estimate rating for the requested item.

Rating prediction for the set of items is done mostly in the same way, except that requester node passes a list of item identifiers instead one identifier and the answer contains a list of pairs (*itemId*, *rating*) for all items that the neighbor node is able to estimate.

Informally, rating prediction scenario can be interpreted as asking an advice from co-minded people. In centralized systems it is performed in some conceptual way, in the proposed hybrid P2P system it is performed literally sending requests to the respective nodes.

When answering rating prediction request, a node can report the rating that is stored for the given item, or infer the rating from some other information. This is an extension point of the proposed system architecture.

These requests are send and answered through anonymization relays, so a node does not expose both its identity and an exact rating for an item.

2) Recommendations query: In this case, node that needs recommendations just sends corresponding requests to each of the neighbor nodes. Each neighbor node answers with a list of (item, rating) pairs. Unlike the previous scenario, here neighbor node needs to send not just an identifiers of the recommended items, but their entity, something that the receiver side can use directly.

The way neighbor node forms the recommendations list is also an extension point. In the simplest case, it should return some random sample of the high-rated items, it may also return only new high-rated items.

Anonymization relays make sure that recommendations provider does not expose both ratings and its identity.

3) Rate item (supplementary scenario): The main issue of rating items is rating new items and generation of new locality-sensitive hash functions that must follow it. To address this issue each node has two lists: Known and New. The Known list holds all the items that are known of by the Master node. This list is received from the Master node during the bootstrap process of periodical synchronization process. The order of items in this list is also important as it corresponds to the order of dimensions of locality-sensitive hash functions. New list, on the other hand, holds the items that are discovered by this node and are not yet approved by the Master node. When a user rates an item, the rating is saved and then, if the item is neither in Known, nor in New lists it is added to the New list.

When New list exceeds some predefined size or once in a predefined period (whatever happens first), each node sends their New list to the Master node and queries the Master node for the global shared state. Global shared state from the Master node includes up-to-date version of the Known list. Each node augments its Known list according to the one received from the Master node and removes from New list items that are present in Known list.

4) Refresh hash functions (supplementary scenario): Each node periodically queries the Master node for the global shared state. As it was described earlier, there are L functions, and each hash function is a vector of k m-dimensional random vectors (representing random hyperplanes). To reduce the amount of information exchange and load of the Master node, each hash function posted by the Master node is represented by three integers: function unique identifier (*funcId*), random seed and current number of items m (i.e. item space dimensionality). When a node gets this information it generates random hyperplanes constituting the locality-sensitive hash function as a sequence of $k*m$ (m dimensions for each of k hyperplanes) random numbers from the specified seed using Mersenne twister [23].

5) The search for similar peers (supplementary scenario): The search for similar, or neighbor, peers is initiated when a node is registered in P2P network. Then this search is performed regularly. Before searching for neighbors a node have to refresh item list and hash functions from the Master node. Then each function from an up-to-date set of hash functions is applied to this node ratings vector. The results are merged into pairs (*funcId, value*) and these pairs are used as keys to look up in DHT. DHT look up returns a list of node identifiers similar to this one according to respective locality-sensitive function. These lists are then merged and stored as the Neighbors list.

V. CONCLUSION

In this paper an architecture of a user-centric peer-to-peer recommendation system, based on locality-sensitive hashing is proposed. In the proposed architecture user ratings are shared only in anonymized way.

Though the experimental evaluation of the proposed architecture is in progress, some limitations of this approach can already be enumerated. First, due to DHT limitations it is not suitable for P2P networks with high churn, second, it is most likely does not fit highly dynamical domains, such as news recommendation, because the need of sharing information about all objects all over the P2P network.

In the future, the authors are planning to consider alternative solutions to sharing global set of locality-sensitive hash functions among peers.

ACKNOWLEDGMENT

The research was supported partly by projects funded by grants # 13-01-00271, # 13-07-13159, # 13-07-12095, # 13-07-00039, and # 14-07-00345 of the Russian Foundation for Basic Research, project 213 (program 15) of the Presidium of the Russian Academy of Sciences, and project #2.2 of the basic research program “Intelligent information technologies, system analysis and automation” of the Nanotechnology and Information technology Department of the Russian Academy of Sciences.

REFERENCES

[1] F. Draidi, E. Pacitti, B. Kemme “P2Prec: a P2P recommendation system for large-scale data sharing”, *Journal of Transactions on Large-Scale Data and Knowledge-Centered Systems (TLDKS)*, Springer, LNCS 6790, vol. 3, 2011, pp. 87-116.

[2] F. Draidi, E. Pacitti, D. Parigot, G. Verger “P2Prec: a social-based P2P recommendation system”, *in proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 2593-2596.

[3] G. Pitsilis, L. Marshall “A trust-enabled P2P recommendation system”, *in Proc. 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2006, pp. 59-64.

[4] M. Jelasity, A. Montresor, O. Babaoglu “T-Man: Gossip-based fast overlay topology construction”, *Computer Networks*, 53, 13 (August 2009), pp. 2321-2339.

[5] R. Ormandi, I. Hegedus, M. Jelasity “Overlay management for fully distributed user-based collaborative filtering”, *Euro-Par 2010*, pp. 446-457.

[6] A. Tveit “Peer-to-peer based recommendations for mobile commerce”, *in Proc. 1st Intl. workshop on Mobile commerce (WMC’01)*, ACM, 2001, pp. 26-29.

[7] A. Bakker, E. Ogston, M. van Steen “Collaborative filtering using random neighbours in Peer-to-Peer networks”, *Workshop on Complex Networks in Information & Knowledge Management*, 2009, pp. 67-75.

[8] A.-M. Kermarrec, V. Leroy, A. Moin, C. Thraves. “Application of random walks to decentralized recommendation systems”, *in proceeding of the 14th international conference on Principles of distributed systems*, 2010, pp. 48-63.

[9] F. Hecht, T, Bocek, N. Bär, R. Erdin et al. “Radiommendation: P2P on-line radio with a distributed recommendation system”, *in Proceedings of the IEEE 12th International Conference on Peer-to-Peer computing*, 2012, pp. 73-74.

[10] P. Han, B. Xie, F. Yang, R. Shen “A scalable P2P recommendation system based on distributed collaborative filtering”, *Expert Systems with Applications 27(2)*, 2004, pp. 203-210.

[11] K. Pussep, S. Kaune, J. Flick, R. Steinmetz “A Peer-to-Peer Recommendation System with Privacy Constraints”, *in CISIS: IEEE Computer Society*, 2009. pp. 409-414.

[12] A.Rajaraman, J.Ullman *Mining of Massive Datasets*. Cambridge University Press, 2012.

[13] M.Slanley, M.Casey “Locality-Sensitive Hashing for Finding Nearest Neighbors”, *IEEE Signal Processing Magazine*, vol.25, no.2, March.2008, pp. 128-131.

[14] P.Indyk, R. Motwani “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality”, *in STOC’98 Proceedings of the 30th Symposium on Theory of Computing*, 1998, pp.604-613.

[15] M.S.Charikar “Similarity Estimation Techniques from Rounding Algorithms”, *in STOC’02 Proceedings of the 34th annual ACM symposium on Theory of Computing*, 2002, pp. 380-388.

[16] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni “Locality-Sensitive Hashing Scheme Based on p-Stable Distributions”, *in SCG’04 Proceedings of the 20th annual symposium on Computational geometry*, 2004, pp. 253-262.

[17] D. Korzun, A. Gurtov *Structured Peer-to-Peer Systems. Fundamentals of Hierarchical Organization, Routing, Scaling and Security*. Springer, 2013.

[18] Q. Wang, N. Borisov “Octopus: A Secure and Anonymous DHT Lookup”, *in Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems*, 2012, pp. 325-334.

[19] X. Chen, S. Ren, H. Wang, X. Zhang “SCOPE: Scalable Consistency Maintance in Structured P2P Systems”, *in Proc. of IEEE INFOCOM*, 2005, pp. 1502-1513.

[20] G. Oster, P. Urso, P. Molli, A. Imine “Data consistency for P2P collaborative editing”, *in CSCW’06 Proceedings of the 20th anniversary conference on Computer supported cooperative work*, 2006, pp. 259-268.

[21] Y. Hu, L.N. Bhuyan, M. Feng “Maintaining Data Consistency in Structured P2P Systems”, *Parallel and Distributed Systems, IEEE Transactions*, Vol.23, Issue 11, 2012, pp. 2125-2137.

[22] C. Mastroianni, G. Pirro, D. Talia “Data Consistency and Peer Synchronization in Cooperative P2P Environments”, Technical Report, unpublished

[23] M. Matsumoto, T. Nishimura “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator”, *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, Issue 1, 1998, pp. 3-30.