various types and how they can be used to organize digital memories. Section IV explains how the communities are formed according to memory threads. This section explains how EMT and VMT are reflected in network and the joining of a peer of a memory thread-bases community. The next section which is section V describes the searching carried out in our community. The work is concluded in Section VI.

II. RELATED WORK

The idea of organizing digital memories in a meaningful way was originally drawn by Vannevar Bush in 1945 in his famous article "As We May Think" in the form of a machine called *Memex*: "A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory" [2]. It was assumed that only one button push could retrieve all the data you need in a small amount of time. Gorden Bell's *MyLifeBits* [3], utilizing the tools, high processing devices and the relatively-speaking large storage devices available at the time, it would be possible to collect and organize all of our digital data easily. The MyLifeBits software is able to store text, images, links, videos etc. in a database and annotate, which was manual. Jim Gemmell *et al.* [4] described the four principles for designing MyLifeBits. First, there should be no strict hierarchy for organizing data. Second, many visualizations of their *life bits* were desirable to help understand what they would be looking at. Third, the value of non-text media is dependent on annotations. Fourth, authoring tools create two-way links to media that they included with new media. Azizan *et al.* [5] describe a Human Life Memory system for collecting, storing and organizing different life events which they call "Serendipitous Moments", as well as discussing sharing via P2P networks. To share digital memories, it is important to be properly organized in network such that the properties of entity are not lost. Following are a few approaches which use data contents to organize and search peers and data in network.

Upadrashta *et al.* [6] utilise the experience of a peer in a network. Peers analyse queries and find the interest of a peer from the search queries that are received from other peers. In this way each peer stores information about other peers, resulting in the formation of virtual communities. Whenever a search query is received, it is analysed and then forwarded to those peers that have similar interests to those reflected in the search query. An in-experienced peer has less knowledge to find contents or peers, which can be a problem for newly joined peers.. In another approach proposed by Modarresi *et al.* [7], peers with similar interests are grouped together to form a community. This approach is similar to semantic overlay networks [8], the difference being that in semantic overlay networks peers having similar data are connected to same super peer. Data lookup in interest-based communities is performed by sending queries only to those members that have similar interests. Interest-based communities bring peers with similar data or interests together and avoid peers that do not have the required data. Such a community only brings similar peers together, but does not provide any information about the status, characteristic or personality of the entity. These communities are in an un-organized form in which a peer connects with other peer, inside a community, without any further similarity between them. The location of data is not known. When a peer is searching some data, it sends queries to all members – or up to a certain number of hops – within a community. When searching for data, sending queries to all peers creates overhead since those peers that have no relevant data will also receive queries. If a query is restricted to a certain number of hops inside the community, the chance of finding the required data is reduced. We also believe that our memory thread-based communities will be more stable due to the entities in our Entity-based social P2P network.

III. MEMORY THREADS

A human mind store information in an encoded form, which is carried out by biological events and becomes a reason to store the information [9]. For example, emotions are a stimulus to generate such signals in brain, which further becomes a cue to remember an event. The cues are used to recall those memories. People collect and store digital memories of their happy, sad, etc. times in different ways, e.g. pictures, text, videos and so on. These memories are collected at different times, places and events. Various hardware or software tools, e.g. Memory for Life (M4L) systems [10], are able to analyze and meaningfully define memories in a digital form. Some systems also allow manual annotation e.g. MyLifeBits [4], where it is difficult to analyze data by a tool. The information, which defines a data, is stored in the form of metadata. We call each tagged information as *memory key*. As described above, people remember their memories based on some reference to memories, which are then used to recall their memories. The metadata, stored by hardware or software tools, contains such reference(s) to a digital memory in the form of memory key(s). A single memory key or combination of memory keys which form a reference to a digital memory are named as *reference key*. A digital memory should have at least one reference key. A reference key for a data is either set by the owner/user of data explicitly or obtained, by a M4L system, from the data or a portion of data for which the data can be known.

We will now explain the idea with an example of a picture of Eiffel Tower. In Figure 1 [11], a picture of Eiffel Tower is being captured by Jim and Emmy Humberd in March 1989 using a certain brand of camera in a cloudy weather at the 2nd year of its 100th anniversary. In this example the type of data (picture), date (March 1989), device (camera), name of object (Eiffel Tower), weather (cloudy), the photographer, and the event (2nd year of 100th anniversary) are memory keys. A possible reference key to the memory in Figure 1 can be the name of the object which is Eiffel Tower or the photographer's name because of a part of their memories during visit to the place.

References to similar memories are usually common and are recalled at the reference. Similar references, which are in the form of reference keys, of digital memories are put together and arranged on certain criteria and form a *memory thread***.** In other words, a memory thread is the collection of

Figure 1: A digital memory and the memory keys

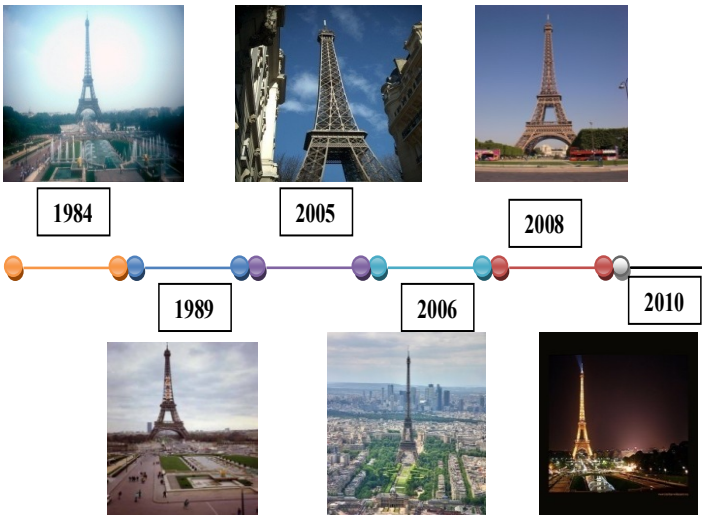

Figure 2: A virtual memory thread based on location and time

memories which has a common reference key under common criteria and can be arranged in a specific order. The order of digital memories should be in way that allows tracing via them from one point to another point. A memory thread should also provide some information about the entity for which a memory thread is formed. For example, time can be a reference to many memories of a person and are recalled as his memories at different times of his age.

To form a memory thread, certain criteria are used to select the relevant digital memories that it will be made up of. There are two types of criteria used to form a thread: *selection criteria* and *indexing criteria*. The selection criteria select those digital memories that have a similar reference point and are used by people to recall their memories. The indexing criteria arrange the digital memories in a given order. It's, therefore important that every digital memory should have at least one memory key which can be used as an indexing criterion. For example, to form a memory thread for the Eiffel Tower, the selection criterion would be the name of the entity itself (*i.e.* the Eiffel Tower) and the arranging criteria could be time, temperature, or any other total ordering. If time is the indexing criteria then the thread will be formed as the digital memories of the structure at different times since it was built in 1889. The memory thread gives us information about the history of the entity in network.

Every entity which has their own digital memories or can be a part of memories of other objects can have a memory thread(s). A memory thread stretches over many digital memories of the object. At a certain place in a memory thread, where the memories of different entities have a common reference key, overlaps with each other. A memory thread can have a single or many overlapping point. For example, as shown in Figure 2 many people visit Eiffel Tower. These people have their own memory thread but at the Eiffel Tower, which is a common reference key to their memories, their memory threads overlap with each other. We call the overlapping points of different threads as *hot points*. The hot points connect different memory threads and form a network of memory threads. The hot points allow accessing the memory threads of other entities. The encircled points in Figure 3 are the hot points for different memory threads.

Some entities e.g. human, are capable of storing and maintaining their memory threads using a device e.g. computer, mobile phone etc. These memory threads are called *extant memory threads (EMT)*. Memory threads of entities whose memories are part of extant memory threads and they cannot capture, store and maintain memories their selves, are called *virtual memory threads (VMT)*. A virtual memory thread is formed at the hot points as shown in Figure 3. Each entity finds the memories of other entities by following its extant memory thread to the hot point that can lead to the virtual memory thread of the entities. Figure 4 shows EMT connected by VMT.

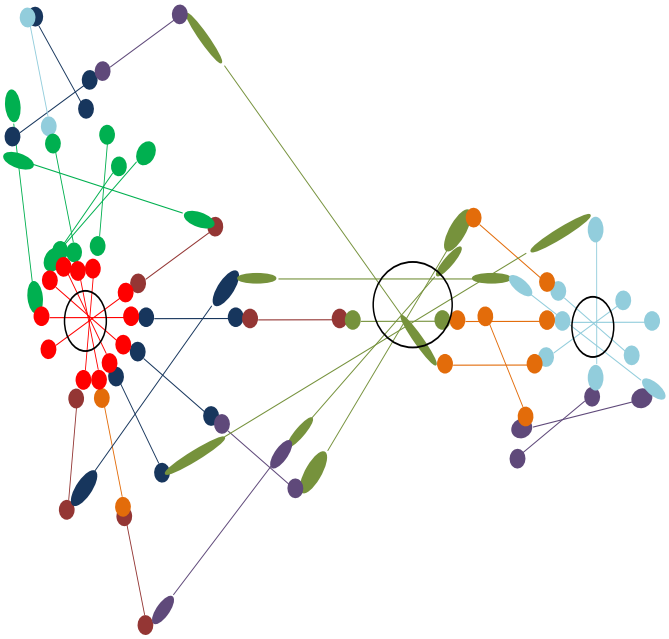

Figure 3: A network of Extant Memory Threads

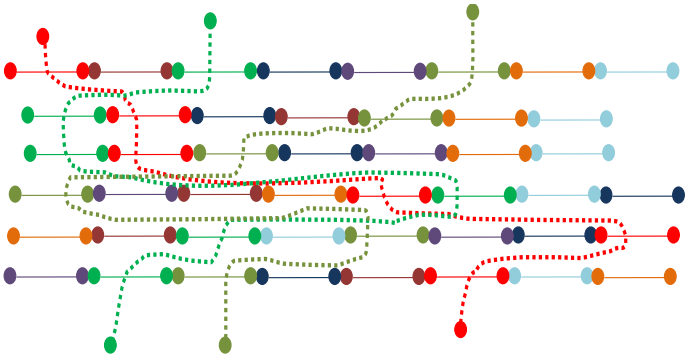

Figure 4: A network of EMTs (solid lines) connected by VMTs (dashed lines)

IV. MEMORY THREADS-BASED COMMUNITIES

Topology of our Entity-based social Peer-to-Peer (P2P) network [1] is based on the idea of memory threads. In our approach, we maintain memory thread for each entity through out the network. A memory thread is reflected in similar way in network in order to organize peers in the form of *Memory Threads-based Community* (MTC). A peer in MTC represents a digital memory in a memory thread. A peer can store many digital memories either for a single or many entities, which allows it to become a member of one or more than one MTCs. Similar to memory threads, a selection criterion is used to form a MTC and an indexing criterion defines a structure or an order for the community. In our network, a reference key for a digital memory is either set by the owner of data explicitly or by a M4L system or is obtained by aggregating search queries received for a single or a group of memory keys. As described earlier, an entity can have an extant memory thread or/and virtual memory thread; similarly our network follows EMT and VMT. An extant memory thread is maintained on a single peer and virtual memory thread across the network. In other words, a virtual memory thread spans via many extant memory threads stored at different peers. Peers form a community based on a thread criteria.

Now we will first describe how to form a memory thread-based community? And then how does it fit to develop the topology of our network?

A. *Extant memory threads*

An Extant Memory Thread (EMT) is formed by an entity from its own digital memories. A single EMT is stored on a single peer or on a personal social P2P network of an entity. To create an extant memory, an entity selects a selection and indexing criterion and simply binds their existing digital memories to the thread or whenever new digital memories are captured. Entities can have a single or more than one extant memory threads. Also, a digital memory can be a part of one or more than one extant memory threads at a time. Other peers in network join an EMT either by inviting by the peer or as members in a virtual memory thread. An EMT is simply a memory thread stored on a single peer to which other peers in network are attached.

B. *Virtual memory threads*

Creating a Virtual Memory Thread (VMT) is not straight forward. A VMT is formed for entities that cannot capture, store or maintain their digital memories and are part of other entities digital memories, which distribute their memories across many peers in network. So in the network, a VMT is formed by connecting those peers which have one or more than one digital memories of an entity. A VMT form the memory thread-based communities in our network. The following paragraphs describe the process of forming and joining a VMT in the network.

To form a virtual memory thread, a peer broadcasts a request for the digital memories of the entity. The request contains the selection criterion for the memory thread. Those peers which have data that matches the criteria of the thread, replies the availability of such data. The reply message contains information about the stored data in their EMT which can become a part of the VMT. The sender receives the reply from all the peers and starts a new community of thread by making a list of all peers which have the data. A suitable indexing criterion is applied to digital memories to order the list of peers. The list contains the addresses of the peers sorted according to the indexing criteria. The list is sent to all peers that replied with relevant data and the peers become a part of the new formed VMT. On the other hand, if the thread is available then member peer of the thread replies. The member peer either returns the data or directs it to the peers that have the possibility of holding the data. The new peer, which was requesting the data, may join the thread.

C. *Joining a memory thread-based community*

A memory thread-based community is formed by connecting those peers which have common reference key for similar digital memories. Peers in MTC are arranged according to an indexing criterion in a specific order, which also define a structure of the community. New peers can join an existing community, as soon as new peers are discovered or new digital memories are added by entities.

If a peer wants to join a memory thread, it follows the indexing criteria to find its place in the thread. A peer finds the first peer by broadcasting in the network as described earlier; once found it connects with it. Each peer in the thread retains information about its neighbour peers either side in this thread. A peer can therefore traverse the thread, moving from neighbour to neighbour, until the correct location is found. It then stores its two or three hop neighbours on both sides of it in the thread. In case of failure or a peer leaving in immediate neighbour, the next hop neighbour is connected to avoid disconnecting the thread. There will be no dedicated peer responsible for maintaining a memory thread; if any peer – whether the peer that started the thread or one that joined it later – leaves a thread, it will not destroy the thread since each peer has equal responsibilities in maintaining the thread.

D. *Structure of Memory thread-based Community*

The purpose of memory thread-based community is not only to organize network properly, but also to maintain the characteristics of an entity across the network. For this

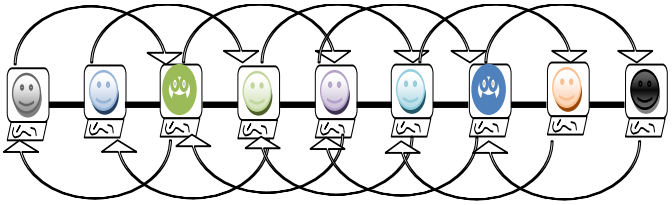

Figure 6: Memory thread-based community with 2-hop connection

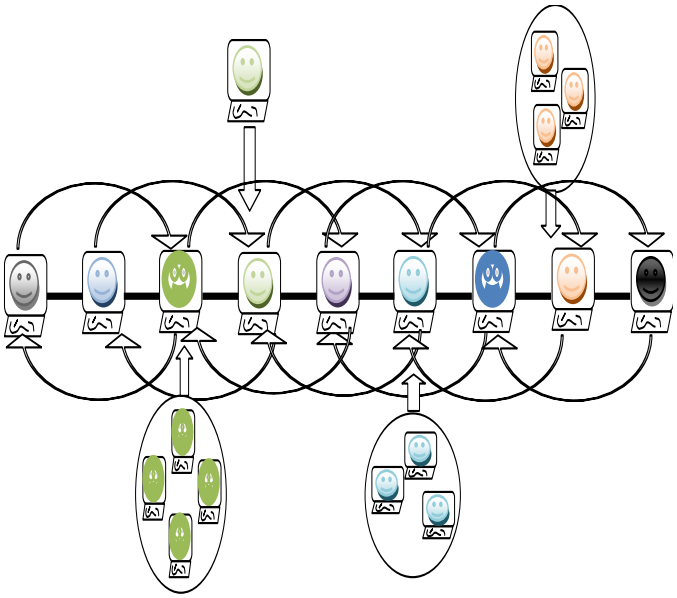

Figure 5: Memory thread-based community with duplicate digital memories

purpose, a characteristic of an entity, stored in the form of metadata, can be used to form the memory thread, in order to organize the community, for an entity. To maintain the characteristics of an entity across the network and to achieve good network performance, we are considering the history of an entity to organize its digital memories in network because the time at which digital memories are captured is usually stored with them.

According to the indexing criterion, which we have considered, the peers in our network are organized linearly such that each peer store address of immediate one and two hop neighbours, as shown in Figure 5.

Figure 6 shows peers that have duplicate files. When a peer is joining a MTC, it finds its location in the thread and joins it. Those peers which have duplicate files have the same locations in thread. When such peers are more than one, then they form a sub-community and select one peer as a representative for the sub-community. The representative takes part actively in the thread. The other peers in the sub-community are connected to the representative as well as to the peers on the both sides of representative, so that if a representative leave the thread; other peers can act as representative. The non-representative peers do not take active part in the thread until they become representative.

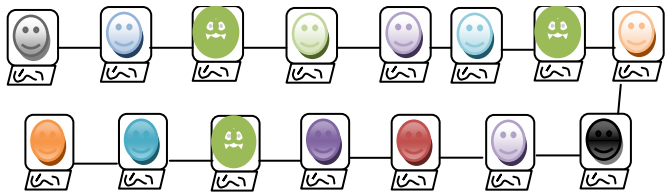

Figure 7: Memory thread-based community peers having multiple files

In figure 7, the peers, which store multiple digital memories about an entity, reoccur in a memory thread. These peers keep a sorted list of neighbours according to the indexing criterion. Whenever a query is received, it is forwarded according to the list by avoiding extra peers in the middle.

V. SEARCH TECHNIQUE

To search data in memory thread-based communities, a peer trace through its own extant memory thread to the location that can lead to a proper virtual memory thread as shown in Figure 3 and 4. After the appropriate VMT is found, the criterion of the memory thread is checked. The peer specifies the selection criterion and indexing criterion for the query to be sent to network. It is then forwarded to the neighbours that can possibly hold the data or can direct to the location of data. When a member peer receives the request, it checks data in its own EMT. If data is found, then it is sent otherwise the request is forwarded to its next neighbour. The request is forwarded to neighbours in memory thread until data is found or a point is reached that indexing criteria is no longer valid and no data is found, then the query is discarded or a no data found message is replied.

Memory thread-based communities do not force the termination of queries after a few hops. If data is available in network then it ensures to find data. Also, we have defined two types of interests for a peer. *Active interest*, the one which a user is adding new digital memories with, searching data about or actively participating in the activities of the interest. *Passive interest* is the interest of a user which he holds the data about but is not participating in activities with other peers and willing to share data. The data of passive interest is still of some interest to those people whose active interest is the passive interest of other people. When a peer's interest changes but still holds the data relevant to the interest, then it will not be lost in network because peers will still be part of the thread and other peers will keep track of them.

VI. CONCLUSION AND FUTURE WORK

In this paper we presented the idea of memory threads to properly organize human digital memories and how it can support social P2P networks. Memory threads were reflected in topology of our Entity-based social P2P network to achieve fast searching and scalability. The communities formed according to memory threads provide useful information about the entity e.g. history of an entity.

In future, we are looking forward to study different structures of memory threads that can exist due to various real

world scenarios. The structure of memory thread and the communities of peers formed according to the memory thread depend upon the indexing criteria of the memory thread. The indexing criteria can change structure from one type to another. For example, history of an entity forms a linearly structured memory thread while the months of years can form a ring structured memory thread because the months repeats after each year. We will also simulate memory threads-based communities as part of the design of our Entity-based social P2P network.

REFERENCES

[1] H. Ur Rahman, M. Merabti, D. LLewellyn-Jones, and S. Sudirman, "An Entity-based Social Peer-to-Peer Network for Digital Life Memories," in *Accepted for publication in: IEEE International Conference on Computer Networks and Information Technology*, Abbottabad, Pakistan, 2011.

[2] V. Bush, "As We May Think," in *The Atlantic Monthly*. vol. 176, 1945, pp. 101-108.

[3] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong, "MyLifeBits: fulfilling the Memex vision," in *Proceedings of the tenth ACM international conference on Multimedia, MULTIMEDIA '02*, Juan-les Pins, France, 2002, pp. 235-238.

[4] J. Gemmell, G. Bell, and R. Lueder, "MyLifeBits: a personal database for everything," *Commun. ACM,* vol. 49, pp. 88-95, January 2006.

[5] A. Ismail, M. Merabti, D. Llewellyn-Jones, and S. Sudirman, "A framework for sharing and storing serendipity moments in human life memory," in *First IEEE International Conference on Ubi-Media Computing*, China, 2008, pp. 132-137.

[6] Y. Upadrashta, J. Vassileva, and W. Grassmann, "Social Networks in Peer-to-Peer Systems," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS '05*, Big Island, Hawaii, 2005, pp. 200c-200c.

[7] A. Modarresi, A. Mamat, H. Ibrahim, and N. Mustapha, "A Community-Based Peer-to-Peer Model Based on Social Networks," *IJCSNS International Journal of Computer Science and Network Security,* vol. 8 no.4, pp. 272-277, 30 April 2008.

[8] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems," in *Agents and Peer-to-Peer Computing*. vol. 3601/2005: SpringerLink, 2005, pp. 1-13.

[9] N. Hayes, "Memory," in *Foundations of psychology*, Third ed: Thomson Learning, 2000, pp. 63-96.

[10] A. Fitzgibbon and E. Reiter, "Memories for life: Managing information over a human lifetime," in *Grand Challenges in Computing Research, and Grand Challenges in Computing Education*, Newcastle, UK, 2003.

[11] J. Humberd and E. Humberd, *Invitation to France: a "why-not-travel" rather than a "how-to-travel" essay*: Em-J Pub., 2001.