Project 2 – Appendix

This document highlights those portions that might be interpreted in different ways that it was actually intended. Moreover, if you have any question, please email me at [Aref.Asvadishirehjini@utdallas.edu](mailto:Aref.Asvadishirehjini@utdallas.edu) before emailing Professor Prakash. I will ask him, and I will update this file.

* **Test case file**

In a few days, I will provide a test case, which would include the course of events that are expected from your program to follow.

* **User**

This is a user-oriented program, that is, user will decide when a node should join the network by entering a command in terminal, user will issue query by using a suitable command, and also the same approach will hold for departure of a node.

* **Search Query Hold Duration**

Let’s assume any intermediate node that receives a search query, will keep the query until initial (generated by search originator) t\_hop seconds (Related to item 1 of assignment section).

* **Index File Structure**

Each node will maintain a Fi file. One approach can be use of an xml file.

<Keywords>

  <term name="term1" \>

  <term name="term2" \>

</Keywords>

<FileReferences>

  <File name="name 1" directory="c://">

    <Keywords>

      <term name="term1" \>

    </Keywords>

  </File>

</FileReferences>

Still, you can decide on subtleties such as whether a file might be associated with multiple keywords or a single keyword. Please do consider that it is only a suggestion. It is your choice to select the appropriate methods for maintaining the Fi file (JSON, binary file, …).

* **Joining to the network**

For simplicity, you can consider a master node that is always the first node that joins the network. Other nodes upon joining the network can ask this node to retrieve the list of the current nodes of the network to select from.

Another approach that is suggested by Professor Prakash is to maintain a single file that is accessible by all nodes in the file system. This file contains the nodes that are already in the network. This file is initially empty. When a node joins the network, it must add itself to this file, and when it’s about to leave, it should remove itself from the list of entries in this file.

* **Search Query**

A search is uniquely identified by the following tuple: <initiator id, search keyword or filename, timestamp of search initiation> (also hop count ). So, multiple searches could concurrently be on for the same file or keyword, initiated by different nodes.

* **Network Growth**

The network growth is a function of number of searches. So, after some constant number of searches that have finished, add a new node. Then, let another set of searches finish and add another node, and so on. You can select a small value like 4-5 searches between node additions.