Prioritized Sync

Sync is a CCNx facility that allows CCN components and applications to define Collections of named data in Repositories that are to be automatically kept in sync with identically defined Collections in neighboring Repositories. Further details are available at the link:

http://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html

A node defines a Collection of named data in its Repository and a sync agent is responsible for the synchronization of this Collection in another node's Repository. Sync agents periodically express a "Root Advise Interest" with the root hash of the local sync tree. The remote agent which receives this Interest compares the received root hash with its own. If they match then no response is sent, else the remote agent sends its root hash and root node in the response. Upon receiving the "Root Advise Response", the local sync agent, using "Node Fetch" Interests, iteratively fetches all nodes with combined hash that it does not recognize. During this process, the local sync agent constructs a list of names not in local sync tree which we refer to as 'difference set'.

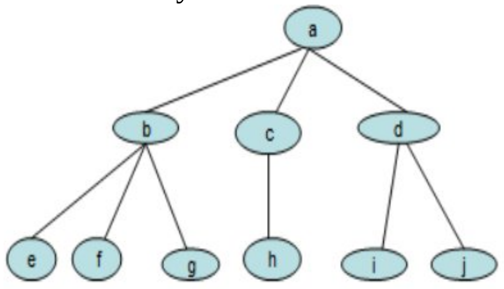

Figure 1

For the names in the difference set, the sync agent expresses a standard Interest and stores the returned content in the local Repository. When there is no prioritization, then the names are fetched in the sequential order in the difference set. For instance, let the difference set be a namespace as in Figure 1 and 'a' be the 'naming prefix' for the Collection. The content fetches are done in the order /a/b/e, /a/b/f, /a/b/g, /a/c/h, /a/d/i, /a/d/j. But, the content fetches can be prioritized by selecting the next item across the breadth of the difference tree in a round-robin manner. With such a prioritization, the content fetches will be done in the order /a/b/g, /a/c/h, /a/d/j, /a/b/f, /a/d/i, /a/b/g.

This prioritized sync helps in information maximization in a resource-constrained environment. By diversifying the content fetches, we maximize the information during sync. For instance, after fetching /a/b/e, fetching content /a/c/h is considered as maximizing information when compared to fetching content /a/b/f.

After analyzing the CCNx sync implementation, we learn that the two phases ("Node Fetch" and "Content Fetches") are sequentially done without any pipelining. So, the prioritization is applied to the difference set after the "Node Fetch" phase and the "Content Fetch" uses this prioritized set as shown in the Figure 2. But, during the content fetch phase, the fetches are heuristically pipelined.

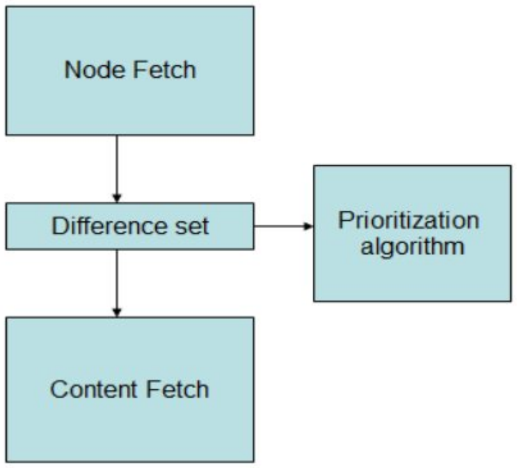

Figure 2

Following source files in Sync were updated to include the prioritization:

- <ccnx-root>/csrc/sync/SyncActions.c
    - Method : CompareAction – Updated the case “SyncCompare_waiting” to include the prioritization.
- <ccnx-root>/csrc/sync/SyncUtil.c
    - Included new utility methods to prioritize the difference set from the method CompareAction.
- <ccnx-root>/csrc/sync/SyncUtil.h
    - Included the new structures and external functions added for prioritization.

Please follow the build instructions as specified in the CCNx README files:

- <ccnx-root>/README
- <ccnx-root>/android/README

To test the prioritization:

- Deploy ccnx on 2 nodes and start ccnd & ccnr.
- Define Collections on the 2 nodes with the same <naming prefix>.
- Upload data to the Collection in one of the nodes.
- Observe the prioritized sync in the ccnr using the following configuration:
    - CCNR_DEBUG=FINE
    - CCNS_ENABLE=1
- Verify that the Collections are in sync.