P2P Discovery

Maintaining P2PXWiki documents namespace

In the current XWiki, each wiki has a URL which can be used to contact a server (such as http://www.bikash.xwiki.com). Such unique server URL doesn't make sense in p2pxwiki since a wiki can be found at multiple peers with different IP addresses. Current P2P systems maintain a lose naming scheme where multiple documents in a p2p system can have the same name. Some P2P systems generate a unique ID based on the hash of the content of a document.

So, the design question is shall we maintain a lose naming scheme for p2pxwiki where the each user is free to chose what name he wants to give? The current xwiki has the advantage that each wiki page has a "particular link" which is easy to memorize and present. The URLS of a wiki have the advantage that they represent a point of contact for a person/company.

Another potential problem with lose naming scheme for p2pxwiki is what I call **wiki hijacking**. Suppose user A downloads a wiki page with name "cnnwiki" from Peer B and stores it in his local database, and suppose there is already a document named "cnnwiki" in the local database of A, the new peer document may overwrite the old one. A malicious user may create a local document named "cnnwiki" and send it to other peers making them think that this is the original "cnnwiki". Such situations doesn't arise in traditional file sharing because there is no guarentee provided about the name of the files, and the users are on their own in trusting the file based on its name.

Handling the above two problems (inconvenience of not having a URL for a wiki and security problems such as wiki hijacking with arbitrary names) require a *logically* central authority for handing out **just wiki names** (although it alone doesn't solve the wiki hijacking problem). So the proposed architecture has a peer group (set of peers) servers (called **xwikiname-servers**) in charge of maintaining a losely consistent database of wiki names currently in use. Any newly created wiki (not the internal pages) need to get a name from one of the xwikiname-servers before it can be accessed by others. A user can assign a name to his newly created wiki, the system will contact a wikiname-server and inform the user if that name (which is a URL like http://www.bikash.p2pxwiki.com) is already taken. If there is a conflict, the user will select another name. The wiki names are internally going to be translated into local document space (separate directories for each different wiki) at a peer, i.e., there is going to be an indirection for mapping wiki URLs to local files.

The above very similar to assigning a domain name, and resolving a domain name, except that it is decentralized by making use of a peer group which use a protocol to keep a consistent view of wiki names already in use.

Just for my information: how is the uniqueness of wiki URL maintained in the current XWiki? Perhaps through a central database check?

Locating a XWiki

With P2PXwiki, we no longer have a central server "URL". Even though he may have a url, say of the form http://bikash.p2pxwiki.com, it doesn't actually refer to a central server. If the user has a URL, and he doesn't have the document in local cache, he can use it to query. Otherwise, the user can also query for a xwiki based on keywords. A maximum number (configurable - say 50) of wiki list whose description matches the query keywords are returned to the user. The user can then select retrieve a particular xwiki by clicking on it. In case the user has a URL, a peer holding a copy of that wiki returns it to the user.

Granularity of query advertisements: individual pages or a wiki?

There will be advertisement messages that are sent out when a wiki is created so that others can locate it in a scalable manner. Advertisements can be sent out for the existence of each page or each wiki. Sending advertisement for each page has very high network routing and query caching overheads, whereas sending advertisement for each wiki may result in lookups at multiple intermediate nodes that have some pages of the wiki but not the requested page. It is better that advertisements are sent at the wiki level in order to

minimize network overheads. A peer which has the wiki will look at internal cache to see if it has the requested page for the wiki.

Offline Capabilities

Getting a Xwiki document on a peer's local machine, i.e., when exactly the document is cached?

It is not feasible to have all the wikis accessible local to the user. Simple solution is that the user starts of with no local documents the first time he uses P2PXWiki. As the user queries for interesting xwikis, they are stored locally.

Way a cache is used when a user tries to get a wiki?

On every access to a XWiki, the system first check to see if it is replicate at the user's local machine, if not, contact peers and load the xwiki from a peer. Note that after the user accesses a particular wiki, other pages that are linked within that wiki can be retrieved pro-actively as an optimization (perhaps in future).

Possibility of creating a new xwiki while offline ?

As the P2PDiscovery section illustrates, assigning a URL to a new xwiki requires contacting a xwikiname-servers in order to maintain globally unique names to Xwiki. So, as a first-cut implementation, in offline mode, the user can not create a new wiki. However, the user is free to add new pages to an existing wiki in offline mode. An enhanced optimization will be to allow the user to create wiki with a provisional codition semantics that wiki names may change if there is a naming conflict when the user comes online.

Replication Capabilities

Number of replicas for each xwiki page and the way they are maintained/version controlled?

We use a master peer for each wiki in order to maintain uptodate copies of the wiki. The master can be a single peer or a group of peer (this choice is explained later). As a first cut, a wiki creator is the **master** of the wiki. The master advertises the presence of wiki to other peers through peer advertisement. The other peers when needing to commit changes will contact the master. Conflict is going to be handled through CVS versioning, and in the same manner as the current xwiki - the later commit is going to overwrite earlier ones, with the system maintaining each version (more on version control follows). The wiki is cached along the way from the master server to the requesting peer, and these cached copies may be returned as a result of later queries (more on caching follows).

Granularity at which master peer is maintained

A peer can be a master for a particular wiki or individual pages. Individual pages will have too much overhead in terms of maintaining consistency and locating a master, so the creator of a wiki will be in charge of being the master peer of all pages within that wiki.

Committing a change to a wiki page to a master?

The peer sends the update message through the p2p network. If the master is reachable, it receives the message, applies the update, and sends a success notification along with the version of the updated page (more on versioning below). Otherwise, the client keeps on trying at periodic intervals.

Local cache coherence

Peer's will receive copies from the masters. Their local copies will remain valid only for certain duration and will need to be checked periodically with the master to find out if it is uptodate. When changing contents are propagated to the master, the cached copy applies those commits locally and updates the version information (more on this later). This ensures that the clients get an uptodate copy only limited by the cache validation duration, without requiring the master to send messages to all the clients on every document change. There is a state maintainance protocol for the cached document. A cached document can be in one of the following state:

1. the cached content is unmodified content from a master with a certain version (master version)
2. the cached content has some of the other peer updates (as a result of peer commit messages) applied locally - (master version, otherpeerupdates)

3. the cached content has some of the other peer updates and some local modifications done by the current peer - (master version, otherpeerupdates, local modifications)

The local modifications, updates by other peers, and master versions are each maintained separately. All the commit, version control protocols work by looking at these separate attributes and guarentees the consistency properties (mentioned below). TODO: designing a state machine for the cache

Consistency semantic (version control and caching strategies) of P2PXWiki

The design should try to maintain the current consistency semantics of xwiki. The current Xwiki has the following characteristics:

- No edits to a Xwiki is lost since the system maintains all versions of previous edits and since all changes are committed to a central server. There is a button that shows history of each xwiki page and all versions are visible using this button. One can roll back and go back to any past version.
- Every user sees the same copy of a wiki page when no editing is being done on a wiki.

Both of the above semantics can be quite tricky to enforce in a p2p setting. In addition, there is another requirement because of p2p nature:

- A peer should be able to see the changes made by other peers even when they are not committed at the master although different peers may be inconsistent with respect to these uncommitted changes. Since XWiki doesn't guarentee any ordering in concurrent editing, different peers may see different ordering for these uncomitted changes. This requirement ensures that peers work with a "more updated" copy when they want to do local edits.

The following design is proposed which tries to maintain the above invariants.

The design is proposed first in pieces with few examples.

case 1: *one master server, multiple client editing the same document independently and commit*: similar to the current xwiki, the last change to reach the master through p2p network will survive and all other changes will be versioned at the master.

case 2: *no master server, multiple clients editing simultaneously using their cached copies, the client can talk to other clients* (Example: Peer/Client A and B both get version 1 of a wiki from master, both get disconnected from master, A does edit, commit, edit, and B does edit, sync, commit on the same document. The difficulty arises in having a simple enough protocol for handling versions that handles both offline peer as well as a peer connected to a master equally, and also provides the consistency guarentees as mentioned above. When peers are not connected to a master, their updates can be treated as concurrent updates and order in which they are applied finally doesn't matter. However, the design should ensure that all updates are retained and also that the peer should be able to see changes made by other peers even when they are not connected to a master. The design, we propose, doesn't enforce any consistent ordering of changes seen by peers, since these changes are seen as concurrent.

Each document will have three types of versioning information (which will probably be maintained using branching):

Master Version number: The most recent version of this document retrieved from the master before the peer went offline **Peer Updated Versions**: Some other peers have made changes since last communication with the master. The peer changes may specify certain server versions that is not available, in which case the client requests the entire document from the peer, otherwise the client applies the diffs to the server version it has. These versioned document are maintained as branches of master versions. Master may (if it is connected) or may not (if unconnected) have this copy. **Local Modified Versions**: Some local modifications have been made since a particular master version/peer version. This has a tuple (master version number, peer changes boolean (the actual number doesn't matter), local changes) - The diff between the master version and the current version is sent to the master once connected. Each of the changes are versioned/branches and can also be sent to the server for maintaining copies. This ensures that all changes (local/remote) are retained. The other peer made changes are not propagated to the master since these will be communicated automatically by the each peer making those changes and will be applied in certain order at the master.

So the basic protocol looks as follows: The offline (disconnected from master) peers communicate and apply peer changes and maintain those versions as branches to the master version. The client gets the most recent master version, with any local/peer changes. once the client modifies the content, it sends the updates to all peers and master. The peer en-route apply those changes as peer only modifications.

case 3: *no master, a single client edit, done, commit-while-connected-to-master* No problem in this case with our versioning scheme which maintains all local, peer, and the master version numbers as branches.

case 4: *no master, a single client edit, done, commit-while-unconnected-from-master* Local branch version changes, and commits are propagated to other peers with the master version number to which commit is applied. peer-commits are just used for letting clients know about some of the changes made by their peers even though master is not around. In the absence of a success reply from the master, the peer will try after periodic interval to commit the changes.

case 5: multiple master servers (future) multiple master servers may cooperate. The biggest issue in this case is resolving conflicts among servers in case of network partitions etc. One can implement an commit protocol among servers to order changes consistently. In case of network partitions, one rule can be that if servers are time synchronized, updates are applied in time order. Or, it can be that updates are applied in random order assuming that there will be fewer conflicts. None of these violates current wiki semantics if we say that these updates are concurrent updates.

case 6: one master server but it can be handed off/moved around (future)

Client queries for a document not in the cache, and the master is not reachable: which peer cached document is retrieved?

The answer is that the network is queried for a certain amount of time, and the peer with the most uptodate server copy gets precedence. if there are two peers with same uptodate server copy, then chose a server at random. This doesn't violate any of the xwiki semantics.

Issue with CVS differencing

when clients are in offline mode, they may have different master version of the document. This is taken into account when another peer requests a copy of the wiki page. The requesting peer gets the most recent master version available. Also, this is taken into account when a peer gets an updated copy from the master - it receives the difference between it's local cached version of the master, and the master's current version.

When a peer requests a document from a master, does it get the older versions also (which are not present at the client) ?

The peer gets some of the version informations (metadata). the actual diffs for old versions are sent only when requested and not found within local cache.

Issues with having multiple master for a wiki? (Advanced features - may be in future)

The consistency problem: the network may become partitioned, wherein different master servers may end up having different conflicting copies. How should conflict resolution work in such a case? there is no clear answer unless every change that is made has an associated owner associated with it, and all the changes are merged in some arbitrary order with all changes presented to the peers as old versions. However, this doesn't lead to a very satisfactory solution when pages have multiple conflicts since the final version of the page can be arbitrary (and perhaps based on very outdated copies). The final version may require lot of further edits to incorporate changes done by partitioned groups.

Clock synchronization problem: the different master peers may have out of sync clocks which will lead to CVS version control problems when combining changes from multiple servers. Some sort of consistent orderings may need to be maintained when commits from multiple masters are to be included.

Consistent Ordering: Master peers may be out of reach with each other from time to time or receive different commits simultaneously and may need to agree on the order in which to apply the updates so that all the masters agree on the final version of a page. The actual version doesn't matter but it should be the same for all the masters to maintain xwiki semantic presented before. This requires some form of leader election to be part of the master group.

Dev.P2PXWiki Design Choices (en)
Creator: XWiki.bikash Date: 2005/07/27 09:22
Last Author: XWiki.bikash Date: 2005/07/27 09:28