

Piki - A Peer-to-Peer based Wiki Engine

Patrick Mukherjee

Christof Leng

Andy Schürr

Technische Universität Darmstadt

{mukherjee@es, cleng@dvs.informatik, andy.schuerr@es}.tu-darmstadt.de

1. Introduction

The wiki technology [4] proved to be an appropriate tool for sharing global knowledge, as it provides an easy way to contribute and consume information. The way wikis are used, e.g. consuming and contributing information, naturally matches the peer-to-peer (p2p) paradigm, which inherently follows the structure of its distributed users. Compared to client-server solutions, it has significantly lower maintenance costs, no bottleneck or single point of failure, and resources are used and offered by all users.

Achieving the same functionality like in server based systems, on a fully decentralized fashion brings numerous challenges. As a user is not just a consumer of the system but a system's part, its failure has to be carefully considered in the design. Controlling and keeping track of concurrent changes, full-text keyword search on articles, tracing and analyzing metadata between multiple semantically connected articles is trivial on a central database but needs sophisticated algorithms in p2p solutions. Other p2p based wiki engines (e.g. [5], [8], [9]) do not solve the problem of concurrent updates with completely reliable version control.

We propose a purely p2p based wiki engine (PIKI) with the focus on concurrent editing, version control, and decentralized full-text search (in a structured overlay). Further we elaborate tracking semantic information between linked articles through different versions, where a typed link itself holds meta-data about the linked files.

2. Peer-to-Peer Wiki (PIKI): Features

Our solution is based on FreePastry [1], which implements the Pastry overlay maintenance and routing [6]. As its implementation is abstracted by the KBR [3] interface, our application may use any overlay which routes key-based, i.e. which retrieves always the same peer when routing for a key from any peer. Articles are edited and viewed using the GUI shown in Figure 1. It provides tabbed browsing and keeps a history of visited pages.

Storing and Fetching Articles Articles can be retrieved by their name, by following a link, or by a full-text search. A data key is calculated by hashing the article name. Routing to this key retrieves the article deterministically or informs that no article with this name was written before. In the last case, the user gets informed that he can create a new article with the given name. Articles are stored on a peer (*primary owner*) whose id is numerical closest to the articles key. As proposed in [6], the primary owner replicates its articles to a number of *replica peers* from its direct neighborhood set. This enables the numerical closest peer to automatically take over all requests if the responsible fails, as the routing algorithm chooses the substitute automatically as the new target.

Version Control Using JLibDiff [2] to compare versions, only updated lines of an article are sent to the article's primary owner. Concurrent changes are serialized by this peer, as it receives all changes on a particular article. Only the first arriving change is applied. Other changes are automatically merged with the newly formed version and sent back to the authors to confirm the changes. The same procedure applies, if a change is based on an outdated version. If a conflict occurs, i.e. changes on the same line in an article, the last author is asked to resolve it.

The primary owner sends an acknowledgment to the submitting peer after updating all replica peers. A failed primary owner gets replaced by the replica peer who is numerical closest to the data key. The replacing peer overwrites the other replica peers storage to ensure a consistent state. It might happen that the committed changes were not applied before the primary owner failed. In any case, the substitute does not send an acknowledgment to the committing peer, forcing it to resubmit its changes after a time-out.

In addition to the latest version of an article, deltas to the older versions are stored, allowing the responsible peer to reconstruct older versions on demand. A peer, which requests an article, attaches its current version number. If it does not specify a desired version, the requester gets the most recent one. Article versions are sent as patches. A user can compare differences between any two versions of an article in a *visual diff*.

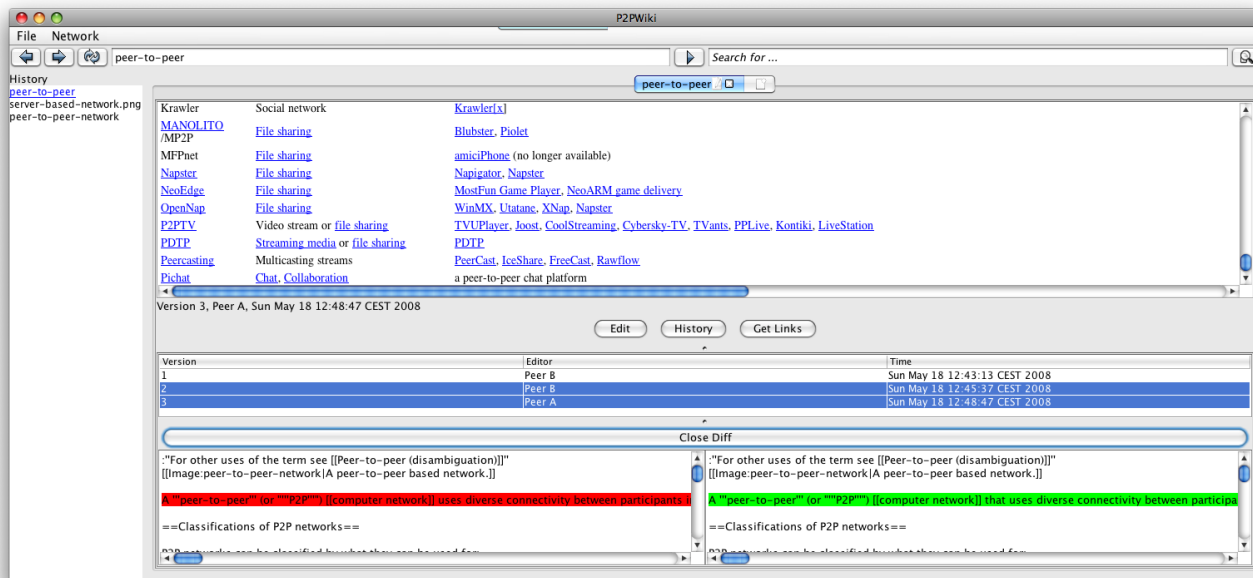


Figure 1. PIKI's user interface.

Full-text Search Beside requesting an article by its name, a full-text search for articles is offered. An inverted index is formed which includes all article names where a certain word occurs. This list is created for each word and it is maintained in the DHT. Queries can contain multiple words connected by AND, OR, and NOT. The peer responsible for the first word's list, sends it to the peer which hosts the second word's list. According to the specified concatenation in the query, it merges the lists and forwards the result to the peer responsible for the next searched keyword. Eventually, the peer responsible for the inverted index of the last word sends the result to the searching peer. In the case a peer fails before forwarding the received query, the initiator has to send the query again after a time-out. The search results, which are likely to contain multiple articles, are ordered by how often the searched term occurs.

Meta-data on Articles and Traceability Links Additionally to an article, a *metadata file* is stored and maintained. It contains arbitrary meta information about this article (e.g. author, version number, all links from and to that article), semantic information *between* articles, enabling analysis of their relationship (e.g. which articles point to a observed one). Our approach handles semantic information and provides traceability by saving constraints to ensure that only correct versions of articles are linked. By assigning types to articles and links, an article of a type 'city' could be linked by a link-type 'city-in-country'. When the article about the country changes its type, for example to 'zone-in-amusement-park', the wiki engine could direct links from the city-article to the most recent version of the country-article, which is still of type country. Note that this ability

is unique. The related approaches always retrieve the latest version of a linked article, which might not be appropriate.

3. Future Work

In our future efforts, we will extend our wiki with access control capabilities, improve the full-text search with BitZipper [7] and refine our version management implementation with the focus on metadata management, in order to propagate and trace changes.

References

- [1] Freepastry. <http://www.freepastry.org/>.
- [2] Jlibdiff. <http://jlibdiff.sourceforge.net/>.
- [3] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *IPTPS'03*, pages 33–44.
- [4] B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc., 2001.
- [5] J. Morris and C. Lürer. Distriwiki: A distributed peer-to-peer wiki. In *WikiSym'07*, pages 69 – 74.
- [6] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01*, pages 329–350.
- [7] W. Terpstra, S. Behnel, L. Fiege, J. Kangasharju, and A. Buchmann. Bit zipper Rendezvous—Optimal data placement for general P2P queries. In *EDBT'04*, pages 466–475.
- [8] G. Urdaneta, G. Pierre, and M. v. Steen. A decentralized wiki engine for collaborative wikipedia hosting. In *WebIST'07*.
- [9] S. Weiss, P. Urso, and P. Molli. Wooki: A p2p wiki-based collaborative writing tool. In *WISE'07*, pages 503–512.