Distributed Wikis: A Survey

Alan Davoust, Hala Skaf-Molli, Pascal Molli, Babak Esfandiari, Khaled Aslan

Abstract

"Distributed Wiki" is a generic term covering various systems, including "peer-to-peer wiki," "mobile wiki," "offline wiki," "federated wiki" and others. Distributed wikis distribute their pages among the sites of autonomous participants to address various motivations, including high availability of data, new collaboration models and different viewpoint of subjects. Although existing systems share some common basic concepts, it is often difficult to understand the specificity of each one, the underlying complexities or the best context in which to use it. In this paper, we define, classify and characterize distributed wikis. We identify three classes of distributed wiki systems, each using a different collaboration model and distribution scheme for its pages: highly available wikis, decentralized social wikis and federated wikis. We classify existing distributed wikis according to these classes. We detail their underlying complexities and social and technical motivations. We also highlight some directions for research and opportunities for new systems with original social and technical motivations.

Distributed Wikis: A Survey

Alan Davoust[1], Hala Skaf-Molli[2], Pascal Molli[2],
Babak Esfandiari[1] and Khaled Aslan[2]
[1]Carleton University, 1125 Colonel By Drive,
Ottawa, Ontario K1S 5B6, Canada
[2] LINA, Nantes University, 2 Chemin de la Houssinière,
44300 Nantes, France

January 14, 2015

1 Introduction

Wikis are typical Web 2.0 applications [1]; they demonstrate how a system can turn strangers into collaborators. In these *groupware* [2] systems, a typical task for a group of users is to write and maintain wiki pages, which are web pages –hypertext documents– that can be edited directly in a web browser. Traditional wikis are centralized: the wiki engine and collection of pages are hosted by a single organization, which users access remotely. The best known wiki is certainly Wikipedia, one of the most accessed sites on the Web. Despite their success, wiki systems still suffer from many issues, such as:

(i) Scalability and cost: as the number of users and contributions increase, so do the storage and bandwidth requirements. Every year, the Wikimedia Foundation (which hosts Wikipedia) needs to raise millions of dollars to finance its huge infrastructure costs;

(ii) Centralized control/single point of failure: a centralized wiki is controlled by a single organization. If the organization disappears –e.g., if the Wikimedia Foundation runs out of funds– all the knowledge contributed to the wiki may disappear as well;

(iii) No offline access: if a user's internet connection or the central wiki system are temporarily unavailable, the user cannot work;

(iv) Handling transactional changes: a user cannot modify multiple pages simultaneously before making modifications visible to other users. All incremental modifications are immediately visible to other users. In some cases, users may observe inconsistent states of the wiki, as in a database system without transactional support [3];

(v) Handling disagreements: even when changes to a page are not made with malicious intent, they can still lead to "edit wars" [4] when different groups of users disagree on the content that is being contributed and they attempt to

cancel or override the other groups' input;

(vi) A single point of view: there is only one authoritative entry on each topic, either because only one version is ever kept by the system, or because the latest version is the one that is displayed to the user, and the user typically is not encouraged to compare that latest version with previous ones. This is the case for most wikis, including Wikipedia. In fact, Wikipedia strives for a single point of view, the one that is deemed "neutral." But neutrality is not always easy to achieve, and it is often more desirable to be exposed to all point of views on a given topic, particularly when it is a controversial one.

While large systems such as Wikipedia can manage scalability using a distributed infrastructure, this comes at a massive cost, for storage, bandwidth and system maintenance. To address the different issues outlined above, many systems have been proposed, such as Wooki [5], UniWiki [6], DistriWiki [7], DSMW [3], git-wiki [8], Smallest Federated Wiki [9] and P2Pedia [10]. These systems extend the wiki concept to a distributed infrastructure, under the names "peer-to-peer wiki," "mobile wiki," "offline wiki," "federated wiki" and others. Although existing systems share some common basic concepts, it is often difficult to understand the specificity of each one or the best context in which to use it. Currently, there is no classification that can be used to compare existing systems.

In this paper, we propose to formally define the general concept of a "distributed wiki," and to classify and characterize the different existing systems that fit this concept. The main characteristic of distributed wikis is that the wiki engine and the collection of wiki pages are distributed or replicated across several physical sites, managed by *autonomous participants* [11]. The idea of autonomous participants is that there is no single organization managing the different sites. By this definition, Wikipedia's distributed infrastructure [12] is not a distributed wiki, as it is managed by a single organization. There are many different motivations for distributing a wiki among autonomous participants. We have identified three fundamental motivations, each enabling us to define a category of distributed wiki systems:

(1) *Highly available wikis*: in this category, the infrastructure is decentralized between the participants, but for the users, the system's functionality is identical to that of a traditional wiki. The decentralized infrastructure allows the participants to share the cost of storage, bandwidth, and maintenance [13], and provides availability, scalability, and fault-tolerance for the wiki. This category of distributed wikis mainly tackles issues traditionally related to distributed systems.

(2) *Decentralized social wikis:* this category of systems relies on an explicit social network of participants and aims to support the multi-synchronous collaboration model [14] and offline access. In these systems, the collaborative editing process follows cycles of divergence/convergence, where users publish their changes, and acquire those published by others, at the time of their choosing. In addition, users can be selective in the subset of changes that they integrate. However, these changes are integrated automatically by algorithms that enforce particular consistency models, which limit the freedom of the user

to select changes from others.

(3) *Federated wikis*: In this category of systems, users are connected via an explicit social network of wikis, which are not expected to be consistent. Users access each other's pages through the social network and have complete freedom to copy and reuse each other's work. Federated wikis therefore promote multiple viewpoints on topics, by allowing the participants to maintain different versions of the pages, which are materialized by having multiple pages with the same title. The users can view the different page versions and perceive disagreement between participants [15]. This category of distributed wikis allows for the emergence of different viewpoints.

The different categories of distributed wikis can be obtained by applying different approaches to the distribution and replication of wiki pages across the participants, ranging from the full replication of the wiki content, to the partition of the collection of pages among the sites. The choice of distribution scheme, change propagation and conflict resolution strategies will significantly affect the performance of the system and its collaborative writing model [16].

In section 2, we define a formal model and some basic operations for traditional wikis. Then we show how this basic model is extended in the different categories of distributed wikis identified above. For the description of the three categories, we follow the same presentation plan. We start with an overview of the main characteristics of the category, then we define the data model, specific operations for the category and a collaboration scenario in the category. Finally, we describe some representative systems in the category. Section 3 presents highly available wikis. Section 4 presents decentralized social wikis. Section 5 presents federated wikis. The last section concludes the paper and points to future work.

2 Traditional Wikis

According to Ward Cunningham –the creator of the original Wiki– a wiki is "the simplest online database that could possibly work" [1]. More specifically, it is a collection of Web pages hosted on a server running a wiki engine, which gives them the following characteristics:

(1) Page title uniqueness: Each page has a unique *title* within the wiki;

(2) Read-write: A wiki page can be edited directly through a web browser. The textual content of a wiki is usually referred to as *wikitext*, and is typically written using a simple mark-up language rendered by a wiki engine;

(3) Wikilinks: Wikis include internal hyperlinks; these are hyperlinks to other pages of the same wiki, called *wikilinks*, which are part of the wikitext. The target page of a wikilink can be specified simply by its title, as the title uniquely identifies a page.

In this section, we formalize the concept of a traditional wiki. Later, we will show how this formalization is extended in the different categories of distributed wikis.

2.1 Data Model

We model a wiki as a graph of wiki pages connected by wikilinks:

Wiki A wiki is a tuple $\langle id, G \rangle$, where id is the identifier of the wiki (this is usually the domain name of the wiki server); $G = \langle P, E \rangle$ is the graph of the pages. The nodes P are the pages of the wiki, and the edges $E \subseteq P \times P$ are the wikilinks connecting them.

The wiki identifier is a handle whereby users can locate and access the wiki.

Wiki Page A wiki page P is a pair $(L, content)$ where L is the title of the page and *content* is the *wikitext* of the page; i.e., a sequence of text, wikilinks (defined below), and embedded media content.

Wikilink A *wikilink* with label L is an annotated instance of L appearing within the content of a wiki page. We use the notation $[L]$.

Intuitively, the annotation means that L is no longer simply text, but a symbol recognized by the system.

Edges Let W be the wiki $\langle id, G \rangle$ where $G = \langle P, E \rangle$. The edges E of the wiki graph are defined as follows: Let $P_1 = (L_1, content_1)$ and $P_2 = (L_2, content_2)$ two pages of P. Then: $(P_1, P_2) \in E$ iff $content_1$ contains the wikilink $[L_2]$.

Notice that while the annotation may be present in the wikitext, if the target page does not exist, then, formally, the edge does not exist. The edges in the graph represent the possibility of browsing from the page containing the wikilink to its target page. If the target page does not exist, then this navigation is not possible: typically, the link would then redirect to a page creation form. This last functionality is not expressed in our model.

In a traditional wiki, the title L uniquely identifies a page on a wiki server identified by id. The following property holds:

Unique Page Content Let W be the wiki $\langle id, G \rangle$ where $G = \langle P, E \rangle$, $P_1 = (L_1, content_1)$, $P_2 = (L_2, content_2)$ and $P_1, P_2 \in P$. If $L_1 = L_2$ then $content_1 = content_2$.

Note that this property does not hold in all distributed wikis. For instance, it does not hold for Federated Wikis, described in section 5.

Example

As an example, we represent the page "Nantes" in the English-language Wikipedia. This page and its outgoing edges to other pages (those visible in the example wikitext) are illustrated in figure 1.

The identifier of this wiki is its domain name "`en.wikipedia.org`". It can be defined as $\langle en.wikipedia.org, G \rangle$ where $G = \langle P, E \rangle$ is the graph of English

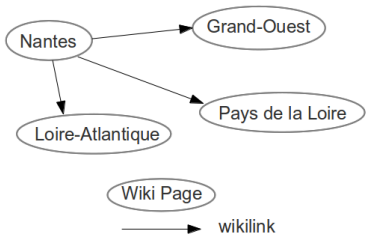


Figure 1: Example: A small section of the English Wikipedia Graph.

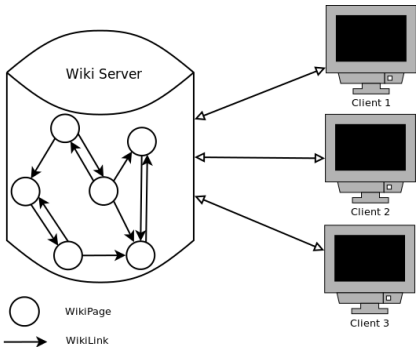


Figure 2: Traditional Wiki

Wikipedia pages. We consider the wiki page ("*Nantes*", *content*) $\in P$ describing the French city of Nantes, in the English-language Wikipedia. The title of this page is the string "Nantes," and the *content* of this page (the "current" version, as of August 6 $^{\text{th}}$, 2013), includes the following *wikitext*:

```
Nantes, labeled art and history city, is the capital city
of the [[Pays de la Loire]] region and the [[Loire-Atlantique]]
department and also the largest city in the [[Grand-Ouest]],
North western France in English.
```

In Wikipedia, a *wikilink* is encoded by double square brackets; e.g., `[[Loire-Atlantique]]`. Therefore, if the page ("*Loire-Atlantique*", *content*2) is in P, then the edge (("*Nantes*", *content*), ("*Loire-Atlantique*", *content*2)) is in E.

2.2 Use Cases

We now present a functional view of a traditional wiki. Figure 2 shows the client-server architecture of a traditional wiki system. In this example, the wiki graph has six wiki pages that are connected by different wikilinks. The client is a web browser that renders the user interface to the wiki. The main use cases available to the users in a traditional wiki are *view page*, *create page*, *delete page* and *edit page*. The server provides backend storage for the wiki system. We

formalize its functionality by a list of operations on the wiki graph. We first present the use cases, then detail the formal operations that implement these use cases.

2.2.1 Use Case: View page

Summary: the user requests a specific wiki page, which is displayed in the browser.

The user can initiate the use case by either manually entering the URL of a page, or following a wikilink from another page. This use case relies on a single operation on the server: the operation $lookup(L)$. The key information sent to the wiki server is the page title L; the wiki server returns the content associated to this title. If there is no page with the given title, then the user may be redirected to an error page, or to a form to create the page (see use case *create page*). The page content is then displayed to the user.

2.2.2 Use Case: Create page

Summary: the user creates a new page from a blank editing form.

This use case may be initiated through a special link in the user interface, or by following a wikilink to a non-existent page. A blank editing form is displayed to the user; the page title may already be filled in. The user writes some content, then may decide to either submit the form and save this content, or cancel. If the user chooses to submit the new page, the operation $save(L, content)$ is called. An error may occur if two users concurrently try to create a page with the same title; see section 2.4 below.

2.2.3 Use Case: Delete page

Summary: the user deletes a wiki page.

When a user is viewing a page, the user interface may provide a link for users to delete the page. When the user selects this link, the operation $delete(L)$ is invoked on the server, where L is the page title. The result is that the wiki no longer has a page with the given title. In modern wikis, the user interface typically does not allow end users to delete pages immediately. Instead, they must initiate a deletion process, in which approval of other users is requested and, after a delay, an administrator (a user with special priviledges) deletes the page.

2.2.4 Use Case: Edit page

Summary: the user modifies an existing wiki page.

The user selects a link to modify an existing wiki page. The page is then displayed in "write mode"; i.e., as an editing form, containing the page's current wikitext. The user modifies the page content by inserting or deleting text, wikilinks and media content. Typically, the user cannot modify the page title.

The user may then submit the changes, which are sent to the server. This invokes the operation $save(L, content)$.

In the case where multiple users edit the page at the same time, there is a risk that some of the changes may be lost. This problem and several possible solutions are described in section 2.4.

2.3 Operations

We now formalize the semantics of each atomic operation handled by the server. We express the operations *lookup*, *delete*, and *save*, as applied to a wiki $W = \langle id, G \rangle$ where $G = \langle P, E \rangle$; these operations are specified by algorithm 1. In these definitions, we give the minimal definition of each operation, where concurrent modifications are ignored. These definitions indicate the basic operation semantics and must be extended to handle concurrent modifications (see section 2.4).

```
1  function lookup(L):-
2   PRE: ∃p ∈ P, p = (L, content)
3   EXECUTE:
4    return content
5
6  function delete(L):-
7   PRE: ∃p ∈ P, p = (L, content)
8   EXECUTE:
9    P ← P \ {p}
10   E ← E \ {e ∈ E|∃p_1 ∈ P, e = (p, p_1) ∨ e = (p_1, p)}
11
12 function save(L, newContent):-
13  EXECUTE:
14   if ∃p ∈ P, p = (L, oldContent) then
15    oldContent=lookup(L)
16   else
17    oldContent = ∅
18   links_old ← {(L, L_i)|[L_i] ∈ oldContent ∧ ∃c_i, (L_i, c_i) ∈ P}
19   links_new ← {(L, L_i)|[L_i] ∈ newContent ∧ ∃c_i, (L_i, c_i) ∈ P}
20   E ← (E \ links_old) ∪ links_new
21   p ← (L, newContent)
22  POST:
23   links_new ⊆ E ∧ (links_old \ links_new) ∩ E = ∅
```

Algorithm 1: Operations of a traditional wiki

Complexity Executing these operations involves a request sent from the client to the server, local processing on the server, and a response to the client. For such simple processing, the main latency factor is the HTTP request/response. For comparison with the distributed setting, and in particular different strategies for the *save* operation, we therefore measure the *message complexity* of operations; i.e., the number of messages exchanged between different net-

worked components (here, clients and servers) until the final state is reached. For all of these basic operations, exactly 2 messages are exchanged.

2.4 Collaboration in a Traditional Wiki: the Concurrent Editing Problem

The problem of concurrent editing occurs when different users simultaneously open the same page in editing mode, then make different changes. Without any safeguards to detect this case, the last user to save her changes will simply override the changes made by the others, without being aware that she has done so.

Among the solutions to this problem, there are two main strategies [16]: the *sequential writing strategy*, in which pages are locked while they are edited, and the *parallel writing strategy*, in which the pages are not locked but concurrent editing is detected at save time. Within the latter strategy, we further distinguish the "first arrived wins" approach — where the first save is applied then all subsequent saves must go through a manual conflict resolution (merge) step — from the solutions that use automatic merge algorithms. However, automatic merge algorithms may produce undesirable results in the meaning of the text; therefore, many algorithms aim to detect the more problematic cases and revert to manual merging. This is the case of Wikipedia.

Both strategies require extending the data model of a page. Additional information is necessary to detect editing conflicts: either a flag must be added to indicate the page being locked, or a timestamp must be used to identify out-of-date versions.

The cost of the edit operation, as measured by the message complexity, differs depending on the chosen strategy. In the case of manual merges, the process will require many back-and-forth messages, as each user must reload the page and resubmit the resolved conflicts. If m users concurrently edit a page and they all save their changes, then the total number of messages exchanged between the clients and the server is $\mathrm{m}^2 + 3\mathrm{m} - 2$ [17].

Collaboration Scenario

A scenario in which three contributors edit a Wikipedia page is illustrated in figure 3. In this scenario, t represents a timestamp variable associated with a page.

In this scenario, three contributors, $user_1$, $user_2$ and $user_3$, concurrently edit the page $(p_1, content)$. For simplicity, we assume that the content is a list of characters: $content = \langle abc \rangle$ and each character has a position, starting from position 0. We assume that users will make simple changes, such as inserting or deleting a character.

First a lookup operation is called and a copy of the content of the page is displayed in the browser of each user. The users then modify the content of the page: (i) $user_1$ inserts the character X between a and b (positions 0 and 1): $insert(X, \langle abc \rangle, 0, 1)$; (ii) $user_2$ deletes b: $delete(\langle abc \rangle), 1)$; and (iii) $user_3$

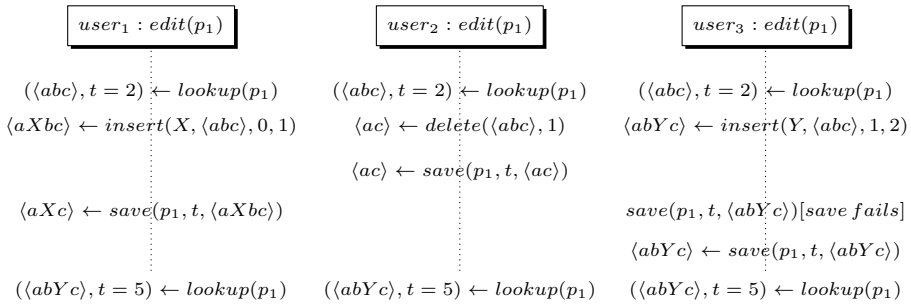


Figure 3: Collaborative editing scenario in Wikipedia

inserts Y between b and c: $insert(Y, \langle abc\rangle, 1, 2)$. First, $user_2$ saves her changes: no conflict is detected, and a new version of the page is submitted to the server with a new timestamp. Later, $user_1$ and $user_3$ want to save their changes: the system first handles $user_1$'s request, and $user_3$'s request is aborted. A conflict is detected with the current version on the server published by $user_2$, so $user_1$ has to solve the conflict manually. In this case, $user1$ keeps $user_2$'s changes (deleting b), and the content of p_1 on the server is $\langle aXc\rangle$. $user_3$ then re-tries to save, and as a conflict is detected, she must solve it manually. She dismisses the other users' changes and imposes her own version. The final content of p_1 is $\langle abYc\rangle$.

By delegating conflict resolution task to users, Wikipedia cannot ensure that all users' contributions will be included. The last contributor decides on the final content of the page[1]. Collaboration in Wikipedia produces content that is validated by at least the last writer, with the risk of producing lost updates.

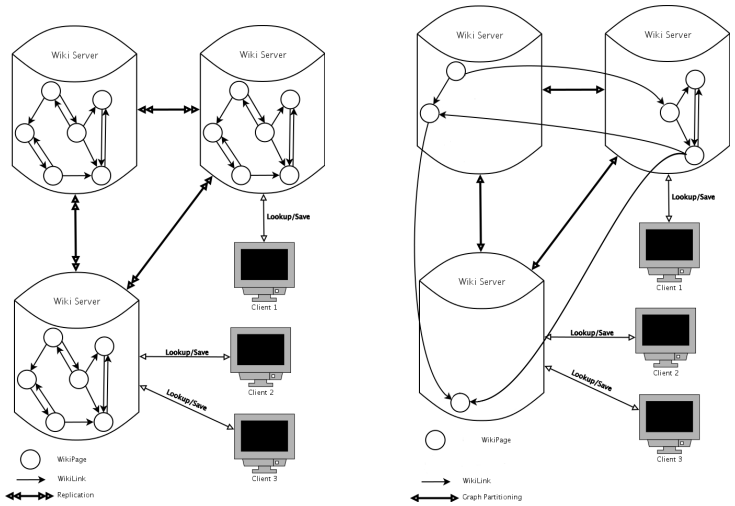
2.5 Distributed Wikis

Distributed wikis extend this basic model to a distributed setting, in which several physical sites, interconnected but managed by *autonomous participants*, host interoperable wiki engines and collections of wiki pages. In this setting, the traditional wiki model must be adapted. The infrastructure can be modeled as a graph $N = \langle S, C\rangle$, representing the participants $S = \{S_1, S_2, \ldots, S_n\}$ and their physical connections $C \subset S_i \times S_j$, where $S_i, S_j \in S$.

Beyond this common defining characteristic, the different distributed wiki systems are driven by different motivations, and therefore make different choices regarding the distribution of resources between the participants, and the functionality that they offer.

The design of a distributed wiki can be characterized by the following criteria:

[1]Extensions to MediaWiki such as http://www.mediawiki.org/wiki/Extension:FlaggedRevs allow for the introduction of more sophisticated collaborative models. These extensions are out of the scope of this paper.



(a) Replicated graph in an unstructured wiki

(b) Partitioned graph in a structured wiki

Figure 4: Highly Available Wikis

- The topology of the graph of participants;
- the physical distribution and replication of pages;
- the change propagation and conflict resolution strategies.

In the following sections, we will represent these choices using the formal concepts introduced so far. We show how the data model and operations of traditional wikis are extended for distributed wikis: *Highly Available Wikis*, *Decentralized Social Wikis* and *Federated Wikis*.

The change propagation and conflict resolution strategies strongly define the collaboration model supported by a distributed wiki. We will illustrate the different collaboration models supported by these distributed wikis through extensions of the scenario given for the Wikipedia example (fig. 3).

3 Highly Available Wikis

Highly available wikis use a peer-to-peer (P2P) infrastructure to provide scalability, by sharing the storage and workload, and/or fault-tolerance, by replicating the content in different locations. In these systems, the distribution and replication of the content is "transparent" to the end users and maintained in the background, beyond the users' direct control. We distinguish between structured and unstructured highly available wikis, according to the underlying P2P overlay network architecture.

3.1 Highly Available Structured Wikis

Highly available structured wikis are designed to share the load (and cost) of the wiki across a distributed infrastructure, and thus provide scalability without requiring a single organization to bear the cost of the infrastructure. This distributed infrastructure is usually a Distributed Hash Table (DHT) [18, 19].

3.1.1 System Model

The collection of wiki pages is *partitioned* across the different nodes of the DHT. Wikilinks can connect pages hosted at different nodes, and the entire set of pages can still form a connected graph. No peer has a global knowledge of the wiki graph.

Definition A *highly available structured wiki* is a tuple $\langle G, N, M \rangle$, where:

- $G = \langle P, E \rangle$, is a (traditional) wiki graph;
- $N = \langle S, C \rangle$ is a graph representing the structured overlay network: the nodes $S = \{S_1, \ldots, S_n\}$ are participants; each participant is uniquely identified and the edges $C \subset S_i \times S_j$ represent their physical interconnections. These connections C, and therefore the graph topology, are usually dictated by a protocol ensuring connectedness of the graph;
- $M : P \to S$ is a function that maps the wiki pages to the participants; this function is also implemented as part of the DHT protocol. Therefore, the graph $G = \langle P, E \rangle$ is distributed among the autonomous participants $\{S_1, \ldots, S_n\}$; i.e., each page $p \in P$ is mapped to a host $S_i \in S$.

Figure 4b shows an example of a structured P2P wiki. The wiki graph of the traditional wiki in figure 2 is partitioned among the wiki servers, and the requests associated to the different use cases are transparently routed to the participant responsible for storing the relevant pages.

Highly available structured wikis may also include some degree of replication to ensure fault-tolerance. In this case, the number of replicas is limited and determined by the system. The replicated pages are stored as backup and not directly accessed by the users. Optimistic replication [20] techniques can be used to manage the consistency of replicas. In optimistic replication, modifications are applied immediately to the replica where they were generated, then are propagated to other replicas to be integrated; conflict resolution is needed in some situations.

3.1.2 Use Cases and Operations

Highly available wikis aim to reproduce the use cases of a traditional wiki; they can therefore be considered identical.

```
1  lookup(L):-
2   PRE: ∃p ∈ P, p = (L, content)
3   EXECUTE:
4    return DHT.get(L);
5
6  delete(L):-
7   PRE: ∃p ∈ P, p = (L, content)
8   EXECUTE:
9   DHT.remove(L)
10
11 save(L, newContent):-
12  PRE: ∃p ∈ P, p = (L, oldContent)
13  EXECUTE:
14  content ← DHT.put(L, newContent)
15  return content
```

Algorithm 2: Wiki operations implementation in a DHT

The traditional operations *lookup*, *delete* and *save* also have the same semantics with respect to the system model (specified in algorithm 1). Typically, the page title is used as a key for the data being stored, and the operations are implemented using the classical DHT functions [19] *get*, *remove* and *put*, respectively, as shown in algorithm 2. The functionality handled by the DHT is mainly *key-based routing*, which consists of routing the operation requests to the peers responsible for storing the pages being retrieved, deleted, or modified. Formally, requests about a page P_i must be routed along the edges of the graph N to the node $S_i = M(P)$.

Additional Use Cases In addition to the traditional wiki use cases, peers can also join and leave the underlying network. These use cases are directly implemented by the underlying DHT's *join* and *leave* functions. These operations normally involve significant overhead, as the DHT routing tables must be updated, and the pages in the network must be redistributed among the peers to ensure load balancing and availability. In terms of our model, this involves modifying the mapping function M.

Complexity The message complexity of the operations in a structured highly available wiki is higher than those of a traditional wiki, due to the communication between the different DHT nodes. In a system with n participants, $(n = |S|)$ as defined in 3.1.1, where each wiki page is replicated k times $(k < n)$, the complexities of the operations have the complexities of the underlying DHT functions, as follows:

- *lookup*: Complexity of *get*; i.e., the routing complexity, typically [21] `O(log(n))`.
- *remove*, *save*: routing complexity, plus the cost of propagating the changes to the k page replicas; if these are stored in neighbouring nodes of the main

data location, this can be done with $O(k)$ messages. The complexity of a *remove* or a basic *save* operation is therefore $O(\log(n) + k)$.

- *join*, *leave*: Depending on the DHT protocol, the message complexity of updating routing tables can range from $O(1)$ to $O(\log(n))$. The cost of redistributing pages depends on the size of the wiki itself: if the pages are uniformly distributed, we can estimate that the number of pages to be redistributed is $O(k.|P|/n)$, where k is the degree of replication and $|P|$ the size of the wiki.

3.1.3 The Concurrent Editing Problem

The different strategies for handling the concurrent editing problem (sequential writing, and parallel writing with automatic or manual merge, discussed in section 2.4) are also applicable to highly available structured wiki systems. For each wiki page P_i, there is a single node $S_i = M(P_i)$ responsible for managing this page. S_i can therefore apply the same strategies that a centralized wiki server would.

3.1.4 Highly Available Structured Wiki Systems

Piki [22], DistriWiki [7], DTWiki [23] and UniWiki [6] are examples of highly available structured wiki systems. Piki uses Key-based Routing [19] and a DHT-based version control system [24] as a storage backend. Each page is assigned to one primary owner and replicated by the primary owner on a number of peers. Piki allows concurrent modifications, which are handled by the primary owner using the "first arrival wins" rule (with manual merges). Unlike most wiki systems, which are accessed through web browsers, Piki is a standalone application. DistriWiki [7] uses the JXTA [25] protocol to build its peer-to-peer network. There is no automatic replication: each node stores a set of wiki pages, and users are expected to search for the latest version of each page in order to edit it. The problem of concurrent modifications is not addressed in DistriWiki.

DTWiki [23] is a wiki system that addresses the problem of operating a wiki system in an intermittent environment. It is built on a delay-tolerant network (DTN) [26] and the TierStore [27] distributed file system. DTN manages communications links as they go up and down, and TierStore provides a transparent synchronization of file system contents, partial replication of shared data, and detection and resolution of concurrent update conflicts on a single file. TierStore manages concurrent update to file replicas by appending a suffix to each remotely conflicting replica. DTWiki detects the presence of the conflict and sends the user a message stating that a conflict has occurred and displays a merge of the contents of the conflicting revisions; the user can choose the final content of the file. Conflict resolution in DTWiki is similar to those in traditional wikis, as explained in section 2.4.

UniWiki [6] consists of multiple wiki front-ends that fetch and store their data over a DHT. It combines optimistic replication [20] techniques and DHT techniques. The originality of UniWiki is that it performs automatic merges of

concurrent edits by running the WOOT [28] synchronization algorithm directly in the DHT nodes, which allows the system to support unstable content with a high rate of modifications while ensuring CCI consistency (see section 3.2.3 for a definition of this concept).

3.2 Highly Available Unstructured Wikis

Most wikis in this category rely on a self-organized unstructured peer-to-peer network. The collection of wiki pages is *fully replicated* across all the participants' sites. The total replication scheme requires that all peers have the same storage capability. The users are connected to one peer and interact with the local page replicas: wikilinks are resolved locally; consequently, a user browsing pages hosted at one physical site cannot follow a wikilink to a page hosted elsewhere. Users are able to work even when their node is disconnected from the rest of the network. When the network is connected, changes are automatically propagated to the other nodes.

3.2.1 System Model

An unstructured highly available wiki is conceptually similar to a set of n interconnected and automatically synchronized wikis.

Definition A *highly available unstructured wiki* is a tuple, $\langle \Omega_G, N \rangle$ where:

- $\Omega_G = \{G_1, G_2, \ldots, G_n\}$ is a set of wiki graphs;
- $N = \langle S, C \rangle$ is a graph representing an unstructured and self-organized overlay network: the nodes $S = \{S_1, \ldots, S_n\}$ are participants, each participant is uniquely identified and the edges $C \subset S \times S$ represent their physical interconnections. Participant S_i hosts G_i;
- The wiki graphs G_i are *eventually consistent*: this notion is defined below.

The wiki appears centralized because the participant directly interacts only with the local system, which is the wiki $\langle S_i, G_i \rangle$. Propagation of updates happens behind the scenes, and to the user is indistinguishable from operations that might happen concurrently on the local wiki if it was an isolated system. Eventually, the set of local page replicas at each node should converge to be identical. As we will see further, ensuring that this happens is difficult. In order to define eventual consistency, we must consider a highly available unstructured wiki to be a system that evolves over time as a result of the user's actions. We note $G_i^{(t)}$ the state of a graph G_i at time t.

Eventual Consistency Let W be a highly available unstructured wiki, $W = \langle \Omega_G, N \rangle$ as defined in 3.2.1. We consider a finite sequence of (arbitrary) user actions, occurring at times $t_1, t_2, \ldots t_k$.

The wiki graphs $\{G_i\}_{i\in[1\ldots n]}$ are *eventually consistent* if at some time later than the last action A_{t_k}, all of the graphs G_i are identical. Formally,

$$\exists t_f > t_k, \forall i, j \in [1 \ldots n]\ G_i^{(t_f)} = G_j^{(t_f)}$$

A highly available unstructured Wiki follows the optimistic replication technique [20], with the hypothesis of eventual delivery of operations; this is generally achieved by using the gossiping algorithm [29]. An anti-entropy algorithm [30] supports intermittent connections. Figure 4a shows an example of an unstructured wiki. In this figure, the graph of the traditional wiki from figure 2 is replicated on each wiki server.

3.2.2 Use Cases and Operations

Each user interacts with a single local site S_i hosting a wiki $W_i = \langle S_i, G_i \rangle$. Let $n = |S|$, the number of peers in the system. The local wiki supports the traditional use cases *view page*, *create page*, *delete page*, and *edit page*: the basic definition of these use cases, with respect to the local wiki W_i, is as in the traditional context (cf. section 2.2). The *view page* use case is implemented by the *lookup* operation on the local wiki W_i. However, the propagation of changes implies that the modifying use cases (*create/edit/delete page*) also affect the other wikis in the network: the expectation is that every change initiated on any node of the wiki is eventually applied to all of the other nodes as well. For this purpose, the relevant use cases are extended as follows (the base being the traditional operation, as described in section 2.2, applied to the local wiki W_i):

- the *create page* and *edit page* use cases are extended so that once the user submits her changes, the *save* operation is called on every wiki of Ω_G;
- the *delete page* use case is extended so that the *delete* operation is called on every wiki of Ω_G.

The condition of eventual consistency means that after all of the *save* and *delete* operations have been applied, the wikis are identical. In order to ensure this condition, the *save* operation must rely on an automatic merge algorithm, with adequate consistency guarantees. This issue is discussed in more detail below.

Finally, as in highly available structured wikis, peers can *join* and *leave* the network. For this class of systems, we will distinguish a node's initial *join* and final *leave* from a temporary *disconnect*, followed by a *reconnect*. For an initial *join*, the new peer must copy the local wiki of another (arbitrary) peer, as described in algorithm 3. During temporary disconnections, pending changes are simply stored so that they can be propagated once the connections are re-established. This simply results in a delayed application of the *save* and *delete* operations generated by remote peers.

The use cases *leave* and *disconnect* do not require any particular processing.

```
1 join(s, s_k):- (s is the new participant, s_k an arbitrary node in S)
2  PRE: s ∉ S, s_k ∈ S
3  EXECUTE:
4     S ← S ∪ {s}
5     C ← C ∪ {(s, s_k)}
6     G_{n+1} ←copy(G_k)
7     M(s) ← G_{n+1}
8  POST:
9   s ∈ S
```

Algorithm 3: Join in an unstructured wiki

Complexity Aside from *lookup* operations, which only involve one peer, the operations in an unstructured wiki are costly, due to full replication. Each modifying operation must be broadcast to the full network, which requires a minimum of $\mathtt{O(n)}$ messages.

The *join* operation is also costly, as the full contents of the wiki must be copied to the new peer. Here the number of messages involved is not particularly relevant; it is more appropriate to consider the number of bytes being transferred as, because of the use of automatic merges, every page must be copied with its entire edit history. Entire edit history is required once when joining the network; reconnecting after disconnection just requires some anti-entropy rounds.

3.2.3 Concurrent Editing and Consistency

We now expand on the *save* operation and its relation to the problem of concurrent editing, and consistency. Unlike structured wikis, in unstructured wikis the replicas of a page can be modified concurrently and independently *on different nodes*.

In a structured wiki, there is one "master" copy of each page, and a small number of "slave" replicas, which mirror the state of the "master" copy, with some latency. In an unstructured wiki, the different replicas may be in an inconsistent state not only because of latency, but due to modifications initiated on different replicas by different users. In order to maintain consistency and avoid lost updates, the sequential writing strategy is not applicable, as the network may be temporarily disconnected, which would prevent page locks from being propagated. With the parallel writing strategy, users may concurrently edit pages and save their changes on different nodes, temporarily disconnected from one another. This implies that conflicts may be detected only when the peers reconnect, possibly long after the editing has occurred. The "first arrived wins" rule would therefore be very impractical, if not downright impossible.

The only viable solution is to automatically merge conflicting changes. However, synchronization algorithms produce content that is not validated by human users, which implies that the text could be nonsensical. An interesting solution to this problem could be to flag the content produced by algorithms, thus producing *concurrency awareness* [31].

The *save* operation with an automatic merge can therefore be described as in algorithm 4.

```
1 save(L, newContent):-
2  PRE: ∃p ∈ P, p = (L, oldContent)
3  EXECUTE:
4  oldContent ← lookup(L)
5  content ← merge(L, oldContent, newContent)
6  p ← (L, content)
7  send(L, content, N)
8  return content
```

Algorithm 4: Save page in unstructured wiki

Concurrent modifications can be *merged* by different synchronization algorithms, such as WOOT [28] or Logoot [32].

Synchronization algorithms implement different consistency models based on the history of changes [17]. This history can be computed by classical textual differences algorithms ("diff") between the old content and the new content of the page. Causality [33] ensures that all sites have the same causal history of changes but does not ensure that all copies are identical, whereas CCI consistency [34] enforces (C)ausality, (C)onvergence and (I)ntention preservation. These notions are defined as follows: (i) *Causality:* all operations are ordered by a precedence relation, in the sense of the Lamport's *happened-before* relation [33], and they will be executed in the same order on every site; (ii) *Convergence:* the system converges if all replicas are identical when the system is idle (eventual consistency); (iii) *Intention Preservation:* the intention of an operation is the effects observed on the state when the operation was generated. The effects of executing an operation at all sites preserve the intention of the operation.

Causality does not imply necessarily convergence. Consider three insert operations, as defined in section 2.4; $op_1 = insert(X, < be >, 0, 1)$, $op_2 = insert(Y, < be >, 0, 1)$ and $op_3 = insert(Z, < be >, 0, 1)$ and the following causal ordering op_1 `happened before` op_2, op_1 `happened before` op_3, op_2 is concurrent to op_3. Therefore, it is possible for $user_1$ to receive $op_1; op_2; op_3$, which produces `<bXYZe>`, and $user_2$ receives $op_1; op_3; op_2$, which produces `<bXZYe>`. In this simple example, causality is ensured; however, convergence is clearly violated.

3.2.4 Highly Available Unstructured Wiki Systems

RepliWiki [35], Wooki [5], XWiki Concerto [36] and Swooki [37] are examples of wikis built on unstructured networks of wiki servers. RepliWiki [35] aims to provide a decentralized, multi-master implementation of Wikipedia by replicating its content. It uses the Summary Hash History (SHH) [38], in which each site maintains a tamper-evident update history that is used to determine the exact set of updates to be transferred during the automatic synchronization between peers. The synchronization algorithm of RepliWiki ensures causality

and convergence.

The aim of Wooki [5] and XWiki Concerto [36] is to support offline work; they also replicate wiki pages on all servers. A modification on a peer is immediately applied to its local copy, then it is propagated among peers using a probabilistic epidemic broadcast [29]. An anti-entropy algorithm [30] is used to recover missing updates for sites that were offline or crashed. Concurrent changes are merged using the WOOT algorithm. This algorithm ensures CCI consistency for connected peers. Swooki [37] is a peer-to-peer semantic wiki [39]. It extends the synchronization algorithm of Wooki to support semantic data.

3.3 Collaboration in Highly Available Wikis

Here we revisit the Wikipedia collaboration scenario from section 2.4.

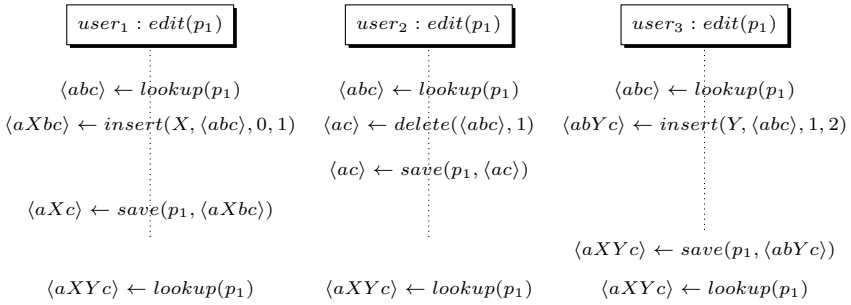


Figure 5: Collaborative editing scenario in a Highly Available Wiki

As noted previously, highly available wikis are designed to provide their users with the same functionality as a centralized wiki, with additional availability guarantees. However, the use of manual conflict resolution, as in the Wikipedia scenario, is very impractical in a distributed setting, particularly when network disconnections may occur. A viable scenario for a highly available wiki is one in which changes are merged automatically, as shown in figure 5.

In this scenario, the concurrent edits made by the three users, $user_1$, $user_2$ and $user_3$, are merged using a synchronization algorithm such as WOOT. As a result, they are all able to save their changes without errors, and they eventually see the result of the merge, which is the wikitext `aXYc`.

Highly available wikis provide traditional wiki functionality with additional performance guarantees. Structured wikis provide fault-tolerance and allow the cost of managing a large wiki to be shared between different organizations, whereas unstructured wikis allow users to work even when the network is disconnected. However, as changes are automatically propagated and integrated, users have limited control over the collaboration process. Users could be interested in sharing their changes only with trusted peers, or modify a set of pages and publish the full changeset in one transaction: this would be particularly useful in semantic wikis, where dependencies exist between pages. Transactional

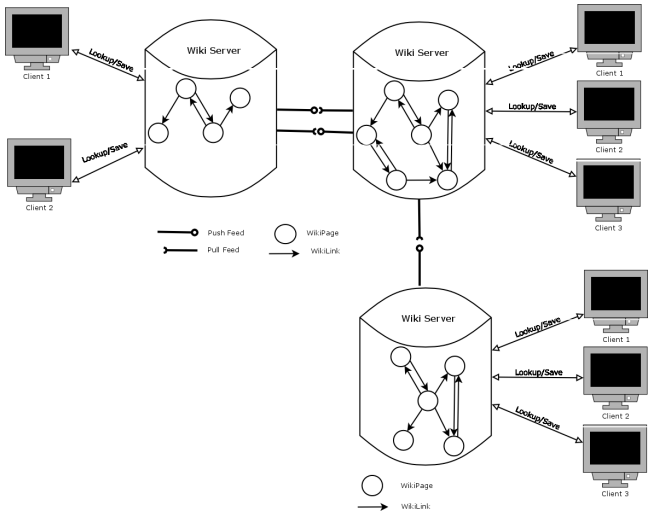


Figure 6: DSMW, a Decentralized Social Wiki

changes, trust and real autonomy for participants are the main motivations of *Decentralized Social Wikis*.

4 Decentralized Social Wikis

Decentralized social wikis aim to support a social collaboration network and adapt many ideas from decentralized version control systems (DVCS) used for software development. They promote the *multi-synchronous* collaboration model [14], in which multiple streams of activity proceed in parallel. The main structure of a decentralized social wiki is similar to that of a replicated wiki; however, the unstructured overlay network is a social collaboration network: its edges represent relationships between users who have explicitly chosen to collaborate.

The synchronization of the nodes is not fully automated; instead, users can choose pages to replicate and manually publish changes, including sets of changes affecting multiple pages. The changes are propagated along the edges of the social network, and users can select which changes to integrate.

As the published changes are propagated through the network, each wiki graph incorporates a subset of the global sequence of changes, filtered through the participants' trust relationships. The task of integrating selected changes can be automated by algorithms that may enforce different consistency models, as in *highly available wikis*.

The explicit collaboration network and the manual publishing and integration of changes define the class of *decentralized social wikis*, an extension to the main wiki concept.

4.1 System Model

Definition A *decentralized social wiki* is a tuple $\langle \Omega_G, N \rangle$, where:

- $\Omega_G = \{G_1, G_2, \ldots, G_n\}$ is a set of wiki graphs, as in a highly available unstructured wiki;
- $N = \langle S, C \rangle$ is a graph representing a socially organized overlay network: the nodes $S = \{S_1, \ldots, S_n\}$ are uniquely identified participants, and the edges $C \subset S \times S$ represent social connections through which operations are exchanged. Each participant S_i hosts G_i.

The social connections can be defined as "follow and synchronize" relationships [40], in which a user can follow the changes made by specific peers and periodically integrate some or all of these changes. Decentralized social wiki systems may provide a full-blown publish-subscribe protocol (e.g., DSMW, which uses "feeds," shown in figure 6), or simply a social acquaintance relationship that underlies the fact that the "follower" regularly "pulls" (in DVCS terminology) and integrates changes from the "followed" user.

In decentralized social wikis, the content of a wiki graph could be different from one wiki to the next; there is no expectation of consistency at the level of the full wiki. However, automatic synchronization algorithms can still be used; we discuss the consistency issues that they raise in section 4.3.

4.2 Use Cases and Operations

4.2.1 Use Cases

Each user interacts with a single local site S_i hosting a wiki $W_i = \langle S_i, G_i \rangle$, as defined in 4.1. This local wiki supports the traditional wiki use cases, with unchanged semantics: *view page*, *create page*, *delete page*, *edit page*.

The social propagation of the changes requires additional use cases, whereby the users *publish* and *integrate* changes:

- *publish changes*: The user selects a set of changes from one or several pages, represented as a list of atomic insertions and deletions, and stores this *changeset* in a location available to other users, using the operation *publish*. A user can publish other users' changes. Therefore, users can receive the same changes by different channels several times. Consequently, the merge algorithm has to be idempotent to avoid duplication;
- *integrate changes*: After retrieving a changeset from another user through a social connection (operation *retrieveChanges*), the user selects a subset of the operations in the changeset and applies them to her local wiki, using the operation *integrateChange*. Integrating the changes may be automated, using algorithms such as those used in highly available wikis (e.g., WOOT). We discuss the issue of consistency in section 4.3.

We note that the synchronization process assumes that the users discover the published changes in some way, either through a formal publish/subscribe protocol or by a query protocol. We do not represent this aspect, which may vary between systems and does not really affect the overall collaboration model.

Finally, users can establish or remove social connections to other users: the *follow* and *unfollow* use cases. These two use cases could happen in very different ways in different systems and we simply describe them below as operations, giving their semantics on the system model.

4.2.2 Operations

In addition to the traditional operations on the local wiki, the publishing and integration of changes is supported by the following additional operations: *publishChanges*, *retrieveChanges*, *integrateChange*. Conceptually, *publishing* a set of changes consists of making the state of the local wiki visible to other users, so that they can at least partially synchronize their local wikis with the published wiki. However, it would be impractical and extremely inefficient to transfer the full state of the wiki over the network, so most systems manipulate a representation of the wiki that describes the new state of the wiki as a list of changes from a shared previous version. The representation is a *changeset*, which includes a reference to a previous version, and a list of atomic *operations*. Therefore, while these notions are not indispensable to the DSW concept, they are the most sensible data model for synchronization. The synchronization operations in a decentralized social wiki are sketched in algorithm 5 and make use of these concepts. See reference [3] for a formal description of the above operations in the decentralized social wiki DSMW.

```
1  publishChanges(changeSet, url):-
2   PRE: the peer has selected a changeset to publish
3   EXECUTE:
4    bind(url, changeSet)
5    notifySubscribers()
6   POST: the changeset is available at URL url
7
8  retrieveChanges(url):-
9   PRE: a peer S_i has published a changeset at URL url
10  EXECUTE:
11   changeset ← connect(url)
12   return changeset
13
14 integrateChange(op):-
15  PRE: op applies to page p ∈ P, p = (L, localContent)
16  EXECUTE:
17   localContent ← lookup(L)
18   content ← merge(localContent, op)
```

Algorithm 5: Synchronization operations in a decentralized social wiki

Complexity The average communication complexity is the number of messages exchanged by a group of trusted participants to converge to the final state. This complexity is $\mathtt{O(M)}$, where M is the average degree of the nodes in N (number of "follow and synchronize" relations of a participant). Convergence is achieved only on shared objects among participants that defined a "follow and synchronize" relation on that object.

4.3 Consistency in a Decentralized Social Wiki

In Highly Available Unstructured Wikis, synchronization algorithms with strong consistency guarantees (WOOT, Logoot...) ensure that once the system is idle, the wikis on the different nodes converge and eventually reach a state where they are all identical. In a decentralized social wiki, users can choose which users they collaborate with, and can choose to ignore some changes published even by the users they collaborate with. It can therefore be expected that the users' local wikis will be inconsistent. The rationale of this approach is that groups of users should collaborate on subsets of the wiki, and within such groups, sets of pages should be consistent while the collaboration lasts. Once a user chooses not to integrate an operation op_0, then all the operations that follow op_0 can no longer be integrated by consistency-ensuring algorithms. This implies that there is a trade-off between the benefits of user autonomy and the consistency guarantees provided by synchronization algorithms.

4.4 Decentralized Social Wiki Systems

Gollum [41], git-wiki [8] and Olelo [42] are wiki systems based on the distributed version control system *Git* [43]. These systems support the multi-synchronous collaboration model, in which users can work in parallel on their local replica and synchronize their modifications when they decide to, using git primitives such as *pull* and *merge*. We note that the *Git* merge algorithm is designed to identify edit conflicts at the granularity of a line (changes are conflicting if they affect the same line) and does not resolve these conflicts automatically. *Git* ensures convergence on shared histories. Convergence on shared objects in a workspace is ensured only if the merge operation is commutative, associative and idempotent, as defined in Commutative, Replicated Data Type CRDT [44] and Summary Hash History (SHH) [38]. This is not the case for the merge algorithm in *Git*.

Distributed Semantic MediaWiki (DSMW) [3, 40] is an extension of Semantic MediaWiki (SMW) [45] that allows SMW servers to be connected and form a decentralized social semantic wiki network. The social links in DSMW are "follow and synchronize" relations. Users create their own collaboration network by creating and subscribing to *feeds*, which are named communication channels for propagating operations. In DSMW, when a wiki page is updated on a participating node, an *operation* is generated, describing the change. The operation is executed immediately against the page and is logged for future publication. A user can then decide to publish a set of changes to a feed called

push feed, and subscribers to this feed may then pull the changeset and integrate the changes to their local wiki graph through a pull feed, as shown in figure 6. A pull feed cannot exist alone: it must be associated with at least one push feed. If needed, multiple changesets can be merged in the integration process, either generated locally or received from other participants. DSMW manages the synchronization of shared pages with the Logoot [32] algorithm, ensuring CCI consistency.

4.5 Collaboration in Decentralized Social Wikis

As mentioned earlier, decentralized social wikis promote the *multi-synchronous* collaboration model, in which multiple streams of activity proceed in parallel. The collaborative work is made up of divergence/convergence cycles: participants can work in isolation from each other, during which time divergence occurs; then, from time to time, users share their changes with each other and integrate these changes to achieve a consistent state.

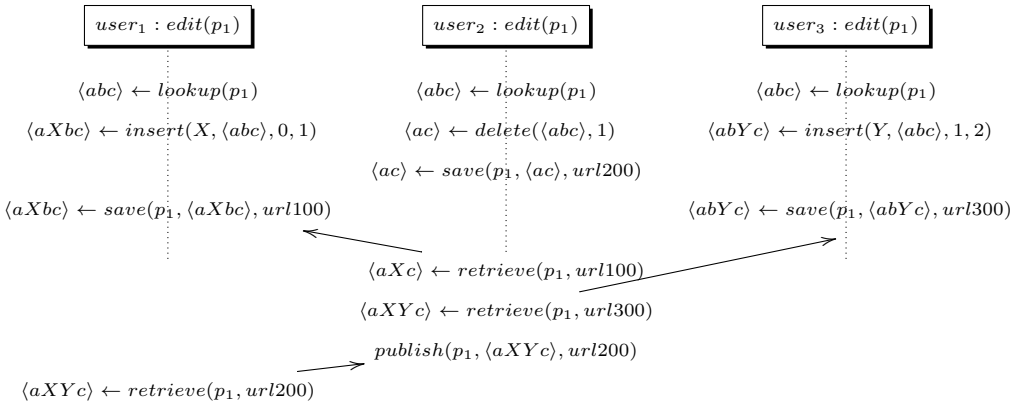


Figure 7: Collaborative editing scenario in Decentralized Social Wikis

Again we revisit the scenario from section 2.4. Each user can work in isolation on her own copy of the page. We suppose that the decentralized social wiki implements a synchronization algorithm that ensures CCI consistency. While saving their modifications, $user_1$, $user_2$ and $user_3$ decide to make them available at the addresses $url100$, $url200$ and $url300$, respectively.

In this scenario, $user_1$ decides to communicate her modifications to $user_2$. A connection is created between $user_1$ and $user_2$. $user_3$ also decides to communicate her modification to $user_2$, and $user_2$ communicates her modification to $user_1$ only. The social network is as follows:

$$user_2 \xrightarrow{follow\ and\ synchronize} user_1$$
$$user_1 \xrightarrow{follow\ and\ synchronize} user_2$$
$$user_2 \xrightarrow{follow\ and\ synchronize} user_3$$

Now $user_2$ can retrieve the new published modifications of both users, and $user_1$ can retrieve modifications of $user_2$. As shown in figure 7, the replicas of p_1 of $user_1$ and $user_2$ converge, but they are divergent from $user_3$'s replica.

The divergence/convergence cycles also occur in unstructured wikis, but the divergence is supposed to be temporary and the role of the system is to ensure convergence. In decentralized social wikis, divergence is a possible choice for any user, and is observable and measurable [46]. Users can define their own collaboration networks and synchronize their work with others at their chosen frequency.

A decentralized social wiki can have the same properties as an unstructured wiki if the social network is connected through the "publish and synchronize" relation and the users publish all of their changes; the system can then ensure eventual consistency. The eventual consistency is defined only for shared objects of each strongly connected component of the social graph. The multi-synchronous collaboration model creates communities with different focal points within the wiki. In a way, decentralized social wikis allow divergence and multiple points of view, but they do not allow users to search and browse the global network and discover these different points of view. Federated wikis support this process, as we will discuss in the next section.

5 Federated Wikis

The term "Federated Wiki" was coined by Ward Cunningham, for his *Smallest Federated Wiki* [9] project (hereafter SFW). The main principle of federated wikis is to allow divergence with no restrictions; that is, two participants can host pages on the same topic (identified by the page title), without having to synchronize them. This allows for multiple points of view to be represented. The key difference with decentralized social wikis is that users can search and browse the global network, thus being exposed to the different points of view expressed by the participants. The participants of a federated Wiki are organized in a social network and use this social network to search and browse the pages hosted by their peers. In federated wikis, users collaborate by copying and reusing material from their peers, without directly altering it in another user's repository.

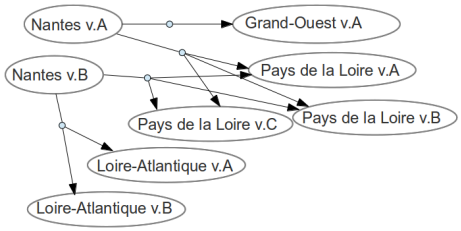


Figure 8: Example: A hypergraph of pages from a Federated Wiki

5.1 System Model

5.1.1 Wikilink Semantics and the Hypergraph Model

In a Federated Wiki, the title of a page no longer uniquely identifies a page: property 2.1 no longer holds. Wikilinks therefore acquire different semantics. A wikilink is defined by a page title and gives access to all or any of the pages in the network sharing this title. In functional terms, following a wikilink implies selecting one of the target pages of the hyperedge. This selection can be done automatically by the system, or else by the user. We therefore model wikilinks as directed hyperedges, and the federated wiki as a directed hypergraph[2].

Figure 8 shows an example hypergraph, a small set of pages from a hypothetical federated wiki. In contrast with the traditional wiki graph shown in figure 1, for each page title, there are several page versions. Version A of the "Nantes" page has wikilinks to the pages "Grand-Ouest" and "Pays de La Loire," whereas version B of the "Nantes" page has wikilinks to the pages "Pays de la Loire" and "Loire-Atlantique." Each of these wikilinks is a hyperedge, because following it will retrieve all the versions of its target page.

Definition A *Federated Wiki* is a tuple, $\langle H, N, M \rangle$ where:

- $H = \langle P, E_h \rangle$ is the hypergraph of wiki pages; H is composed of nodes (the pages P) and directed hyperedges $E_h \subset P \times \mathcal{P}(P)$;
- $N = \langle S, C \rangle$ is a graph representing the network of participants, where the nodes $S = \{S_i\}$ are the participant sites and the edges $C \subset S \times S$ represent their social connections;
- $M : P \to \mathcal{P}(S)$ is a function that maps the wiki pages to sets of participants; this is not a function that can be expressed by a defined algorithm (as in the case of structured wikis), but rather describes a relationship that is under the control of the users. Each page $p \in P$ may be hosted by one or several participants.

[2]Specifically in the sense discussed by Gallo et al. [47]

5.1.2 Unique Page Identification

Although page titles are not globally unique in a Federated Wiki, they may be *locally* unique (property 5.1.2). In this case, any single participant of a Federated Wiki, taken in isolation, is a traditional wiki. In some Federated Wikis (such as P2Pedia), this weaker property does not hold either.

Locally Unique Page Titles A federated wiki $\langle H, N, M\rangle$, where $H = \langle P, E_h\rangle$, has *locally unique page titles* if the following holds: if $P_1 = (L, content_1)$, $P_2 = (L, content_2)$, $P_1, P_2 \in P$ and $content_1 \neq content_2$, then: $\forall s \in S, s \in M(P_1) \Rightarrow s \notin M(P_2)$.

As pages are not uniquely identified by their title, additional identifiers can be introduced to act as globally unique identifiers (GUID), so that a page is a triple $(id, L, content)$, where id is unique.

If page titles are locally unique, then for any page p, the combination of the page title L with $M(p)$ (or any element of $M(p)$) uniquely identifies p and can be used as a GUID.

Alternatively, a GUID can be obtained by hashing the page contents. Once each page is uniquely identified, the GUID can be used to create hyperlinks to specific pages, provided the system implements a mechanism to dereference a GUID. Such "version-specific" links induce a graph structure and can complement the wikilinks.

5.2 Page Distribution

As users are free to make any changes they like to a page, they are also free to host pages on whichever topics they like. In addition, they can copy another user's entire page without changing it, and this page is therefore *replicated*. As a result, the different pages of the hypergraph are replicated "socially": for each unique page, there may be any number of copies, distributed in arbitrary locations.

Figure 9 shows an example distribution of the federated wiki pages of figure 8. In this example, pages titles are not locally unique. Instead, the page title plus the version act as a GUID for illustrative purposes. Some pages are more replicated (more "popular") than others: the page "Pays de la Loire v.C" is hosted by all three participants, whereas the "Loire-Atlantique" pages are only hosted at one site.

5.3 Use Cases

The "social" distribution of pages requires an additional use case, which consists of copying a page version from one participant to another: the *fork page* use case. We note that for decentralized social wikis, content is also transferred manually between participants. However, a page version is not fully transferred; rather, the changes –the differences from a previous version shared by the transfer initiator and recipient– are transferred. In a federated wiki, as there are no

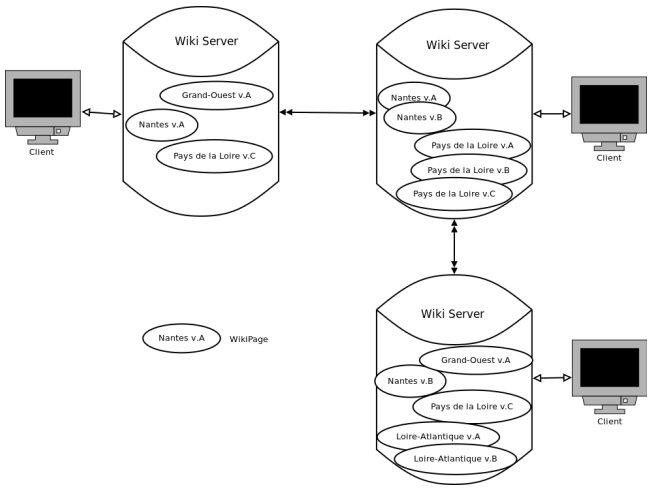


Figure 9: Federated Wikis without the Local Unique Page Assumption. Wikilinks are not shown, to avoid overcrowding the graph.

consistency guarantees or assumptions, no previous version can be expected to be shared between the participants. Pages are therefore copied in full. In federated wikis with local page title unicity, any previously existing local version (i.e., another page sharing the same title) is deleted.

The semantics of traditional wiki use cases are modified as follows:

- View Page: In a traditional wiki, users request specific pages by entering the page title in their browser, or by following wikilinks. As several pages may share the same title, pages are requested in a two-step operation:

  1. Find a list of pages with the requested title in the network;
  2. Select one page to display: the selection can be manual or automatic. This triggers the *lookup* operation, which retrieves the page content based on the unique page identifier.

  These two conceptual steps can be illustrated by the following sequence:

$$L \xrightarrow{search} \{\underbrace{\langle P_1, M(P_1)\rangle, \ldots \langle P_j, M(P_j)\rangle}_{\forall i, P_i = (L, content_i)}\} \xrightarrow{select} content_k$$

- Create Page: users can create new pages, which are stored locally. There is no longer any precondition limiting which pages may be created, unless the page titles are locally unique. In the system model, all hyperedges pointing to other pages with the same title now also point to the new page;

- Delete Page: users can delete only local pages. If other copies of the page exist elsewhere, they are not deleted;

- Edit Page: As users have control over their local material only, the two-step edit-save action cannot be reduced to the effect of saving a new version. Instead, there are now three steps: fork-edit-save. The "fork" step consists of making a local copy of the original page to be edited. This local copy is then edited, then saved, while the remote page is unchanged. The fork operation is specific to federated wikis and is detailed below. Once the page has been edited, it is saved locally, as for a new page (see above). Then, outgoing links must be updated, as in a traditional wiki.

The manual management of the social network requires the following additional use cases:

- Joining and Leaving the network: the participants can join or leave the network. Formally, this corresponds to the node being added to or removed from the graph N, as defined in 5.1.1. No pages are necessarily transferred as a result of the operation.

- Social Network Operations: the participants maintain their social network by establishing or dropping links with others. These operations are, again, edges added or removed in N.

5.4 Operations

The traditional operations of a wiki, and the additional $fork$ operation, are modified as shown in listing 6. We describe them as implemented by a node S_0.

```
1  lookup(id):-
2   PRE: ∃p_id ∈ P, p_id = (id, L, content)
3   EXECUTE:
4    return content
5
6  delete(id):-
7   PRE: ∃p_id ∈ P, p_id = (id, L, content)
8   EXECUTE:
9    M(p_id) ← M(p_id) \ {S_0}
10   if M(p_id) = ∅ then:
11     P ← P \ {p_id}
12     forall h in E such that h = ⟨p_id, p_set⟩ do:
13       E_h ← E_h \ {h}
14     forall h in E_h, h = ⟨p, p_set⟩, p_id ∈ p_set do:
15       h ← (p, p_set \ {p_id})
16       if p_set = ∅ then E_h ← E_h \ h
17
18 save(L, newContent):-
19  EXECUTE:
20   id ← generateGUID(L, newContent)
21   p_id = (id, L, newContent)
22   P ← P ∪ {p_id}
23   M(p_id) ← {S_0}
24    forall h in E_h, h= ⟨p, p_set = {(i_1, L, c_1), ... (i_n, L, c_n)}⟩ do:
25      h ← ⟨p, p_set ∪ {p_id}⟩
26
27 fork(L, content):-
28  EXECUTE:
29   M((p_id) ← M(p_id) ∪ {S_0}
```

Algorithm 6: Operations of a federated wiki

Complexity Most of the operations have very low communication costs, as they are often local. However, the *lookup* operation implies searching the entire network for a copy of the requested page, as pages are not automatically assigned to a particular location. In an unstructured network, a broadcast search requires at least $O(n)$ messages to reach all of the peers.

5.5 Federated Wiki Systems

The Smallest Federated Wiki project [9], led by W. Cunningham, is a set of interconnected and interoperable wiki servers, belonging to different users. Users can seamlessly browse the pages of the different wiki servers: wikilinks are dereferenced by automatically selecting the first available page according to a preference function over the known participating servers. Pages from the user's local server are selected in priority, then pages from the neighboring servers in the network. If no page with a given title is found, then the user is directed to a page creation form. The user interface shows several pages at the same time,

side by side. This makes it easy to edit pages by dragging and dropping from other users' pages. The pages also have a "fork" button. As the page titles are locally unique, when a page is forked, any existing local page with the same title is deleted. The network of servers is automatically discovered and maintained by browsing and forking pages (forks are recorded in the history of pages), which gives the user only indirect control over the social network structure. The full Federated Wiki (i.e., the known network) can also be searched, and users can see all the available versions of each page. Search results are visible as version-specific links, identifying the host of each version.

The P2Pedia wiki [48, 10] explores a very similar idea, but implemented over a P2P file-sharing network. The peers may share any version (or set of versions) of each page. Wikilinks are dereferenced by a P2P file-sharing "search" function, and manual selection of the target page by the user, among the search results. In order to assist the user in this choice, search results can be ranked according to different trust indicators, based on the popularity of each page version, and on the social network. Page replicas are identified by a GUID based on the page content's hash. When the user browses pages, each page is automatically forked; i.e., it is downloaded not only to the browser cache, but also to the user's local repository. When a page is edited, the previous version is also kept, unless the user explicitly deletes it. In addition to these two federated wikis, the different language editions of Wikipedia share some aspects of federated wikis. Articles on the same topic in different languages may represent alternative views on the topic and interlanguage links provide a means of navigating between them. Cap [15] also discusses the different points of view adopted by different wikipedia languages and proposes an "Every Point Of View" approach that matches the motivation of the federated wiki idea. The approach is not implemented, and its technical details have only been sketched out. Presumably, the different versions of each page would be still stored in a centralized repository and users would have access to all of them. This proposal could be a centralized equivalent of the federated wiki concept. However, the centralization goes against one of the fundamental principles of federated wikis: their decentralized control model, in which each user can modify only her own set of pages.

6 Conclusion and Research Directions

6.1 Summary and Comparison

Distributed wikis consist of a network of autonomous participants that host a set of wiki pages. A distributed wiki must handle the maintenance of the network, the distribution of the pages in the network, the corresponding retrieval of pages, and the propagation and integration of the edits made to the pages. The different existing systems are motivated by different social and technical issues and therefore adopt different solutions for each of these technical problems, with different complexities. We have classified them into three general classes, defined by their general motivation.

Highly available wikis, the largest category of distributed wiki systems, are designed to address the technical limitations of a centralized infrastructure: lack of scalability, high cost, and central point of failure. Their defining characteristic is that all of the problems above are handled automatically by the system: wiki pages are either partitioned or replicated across a self-organized network of wiki servers, and edits to the pages are automatically propagated and integrated, using algorithms that guarantee the consistency of the replicated pages. The complexity of the lookup and save operations depends on the size of the network.

Decentralized social wikis are designed to support the multi-synchronous collaboration model and to allow users to organize social collaboration networks. The participants are therefore organized in a social network and modifications to the wiki are manually published, and propagated through communication channels following the social network edges. Different communities will be formed around different focal topics. Within each collaborative community, the convergence of page replicas is ensured by automatic synchronization algorithms.

Federated wikis allow and encourage divergence, in order to accommodate multiple points of view. Federated wikis achieve this by giving users the greatest level of control over the different functionalities. In particular, the system does not enforce any consistency between pages on the same topic, and collaboration is limited to manually copying and reusing the work of others. Many pages may therefore share the same title, and wikilinks cannot simply point to a single wiki page. Instead, they point to all the versions sharing a given title, allowing users to browse the different versions. This is best modeled by a hypergraph of pages that are socially replicated across the sites. A key difference with decentralized social wikis is that users can browse pages from the whole network.

These classes of systems are summarized in table 1. For each system, the table indicates the network organization, the page distribution scheme, the consistency model, the change propagation method, the scope of page retrieval (i.e., the set of pages that users can directly retrieve for viewing), and finally the complexities of the *lookup* and *save* operations. These parameters for the complexity values are mainly n, the size of the network, and d is the average node degree (for decentralized social networks).

6.2 New Challenges and Opportunities for Distributed Wikis

Recent developments in Web techologies are offering new opportunities for distributed wikis.

Technological opportunities and real-time editing Existing distributed wikis are complex to deploy. This is a severe limitation for their adoption by end users. Recent advances in web protocols such as webRTC[3] make it easy to deploy complex distributed infrastructures, even for end users. New

[3] http://www.webrtc.org/

Table 1: Classes of Distributed Wiki Systems

	Highly Available Wikis		Decentralized Social Wikis	Federated Wikis
	Structured	Unstructured		
Network Organization	self-organized		social	social
Distribution Scheme	partition	full replication	user-controlled (partial)	user-controlled (partial)
Page Retrieval	global	local	local	global
Change Propagation	system		social	social
Consistency	N/A	eventual consistency	eventual consistency (sets of pages)	no consistency required
Lookup Complexity	O(log(n))	O(1)	O(1)	O(n)
Save Complexity	O(log(n))	O(n)	O(d)	O(1)
Collaboration Model	parallel editing		multi-synchronous	N/A
Systems	Piki [22], DistriWiki [7], DTWiki [23], UniWiki [6]	RepliWiki [35], XWiki Concerto [36], Wooki [5], Swooki [37]	Gollum [41], git-wiki [8], Olelo [42], DSMW [3, 40]	SFW [9], P2Pedia [48, 10]

systems such as ShareFest[4], PeerCDN[5] or webtorrent[6] demonstrate how direct connections, unstructured P2P networks or DHTs can be deployed directly in web browsers. Such technology can transform any of billions of devices running compatible browsers into distributed wiki participants in one click. This can greatly improve the user experience with distributed wikis and allow researchers to set up new experiments much more easily.

Furthermore, such technologies also introduce distributed real-time editing for web authoring. It is already possible for a wiki instance to integrate a real-time editor such as ShareJs [7], but WebRTC can improve the user experience with fully decentralized browser-to-browser data channels. This raises issues about collaboration models where some contributors edit together in real-time while others may prefer to edit offline. How should slow coarse-grained changes be combined with fast fine-grained changes? How can real-time editing sessions served by different wiki instances be detected and accommodated?

Federated Semantic Wikis The Semantic Web is a major opportunity and an interesting challenge for distributed wikis. Thanks to the Linking Open Data Project (LOD) [49], the Semantic Web makes millions of RDF triples from a network of autonomous participants available to the public. However, data quality is a major issue [50, 51]. Wiki systems have demonstrated how communities of users can improve the quality of shared documents, and semantic wikis [39] or wikidata [52] apply the wiki approach to improving semantic documents and data. Other work [53] aims to transform wikis into collaborative integrated development environments mixing text, data and applications. Such wikis allow simple semantic web applications to be quickly developed and shared.

Surprisingly, in such approaches, wikis remain centralized while the data is hosted in a federation of linked data. In fact, current semantic wikis allow authoring of one local dataset, even if this dataset is linked to others datasets of the LOD cloud. A federated semantic wiki should allow authoring of linked data across the federation, as part of it, or as a new federation accessing the existing federation of linked data.

The decentralized social wiki approach has been already applied to building a decentralized social semantic wiki [40]. However, this system has several drawbacks:

- Semantic data is modified as a side-effect of text modification. The advantage is that the text is kept synchronized with semantic data embedded in the text, but the draback is that semantic data cannot be modified directly. Consequently, other authoring tools for seman-

[4] sharefest.me
[5] peercnd.com
[6] https://github.com/feross/webtorrent
[7] http://sharejs.org/

tic web such as Protégé or SPARQL Update cannot be used safely on the same semantic data authored through a semantic wiki.

- If different autonomous participants collaborate directly on semantic data in a system such as [54], then text and semantic data get out of synch.
- Fundamental replication techniques used to build distributed wikis force all the participants, in the worst case, to have the same storage capacity, and generate considerable amounts of traffic on the network. As the amount of data hosted can be very large, this approach is problematic.

The federated class of wikis seems more appropriate for building a *federated semantic wiki*. A federated semantic wiki should be able to "editorialize" data collected from LOD, *i.e.*, to author meaningful, human-readable documents from linked data. Such documents could then be further edited, and changes would be propagated to the federation of semantic wikis and the federation of linked data. In other words, a federated semantic wiki should be able to make the web of data understandable and editable by humans.

However, a major issue is that the federated wiki and semantic wiki models cannot be applied directly to federated semantic wikis. In particular, there is a mismatch between the graph model of most semantic wikis, where concepts are generally associated with a unique page, and the hypergraph model of federated wikis, which represents the multiplicity of perspectives over shared concepts.

Many more issues can be added to this list: what is a suitable graph model? how should collected data be "editorialized"? Since linked data is mainly read-only, how should federated semantic wikis push back changes? How should data providers trust changes authored in federated semantic wikis? How would concurrent changes be managed,specifically the changes coming from human users within federations of semantic wikis and the changes computed by algorithms on federations of linked data?

References

[1] Leuf B, Cunningham W. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley, 2001.

[2] Ellis CA, Gibbs SJ, Rein GL. Groupware: Some Issues and Experiences. *Communications of the ACM* January 1991; **34**(1):39–58, doi:http://doi.acm.org/10.1145/99977.99987.

[3] Skaf-Molli H, Canals G, Molli P. DSMW: Distributed semantic mediawiki. *ESWC (2)*, *Lecture Notes in Computer Science*, vol. 6089, Aroyo L, Antoniou G, Hyvönen E, ten Teije A, Stuckenschmidt H, Cabral L, Tudorache T (eds.), Springer, 2010; 426–430.

[4] Sumi R, Yasseri T, Rung A, Kornai A, Kertész J. Edit wars in wikipedia. *SocialCom/PASSAT*, IEEE, 2011; 724–727.

[5] Weiss S, Urso P, Molli P. Wooki: a P2P wiki-based collaborative writing tool. *Web Information Systems Engineering*, Nancy, France, 2007.

[6] Oster G, Mondéjar R, Molli P, Dumitriu S. Building a collaborative peer-to-peer wiki system on a structured overlay. *Computer Networks* 2010; **54**(12):1939–1952.

[7] Morris J. DistriWiki:: a distributed peer-to-peer wiki network. *Proceedings of the 2007 international symposium on Wikis* 2007; :69–74.

[8] Rozet S. git based wiki: http://atonie.org/2008/02/git-wiki 2008. URL http://atonie.org/2008/02/git-wiki.

[9] Cunningham W. Smallest federated wiki project. https://github.com/WardCunningham/Smallest-Federated-Wiki/.

[10] Davoust A, Craig A, Esfandiari B, Kazmierski V. Decentralized collaboration with a peer-to-peer wiki. *Proceedings of the 2012 International Conference on Collaboration Technologies and Systems (CTS 2012)*, 2012.

[11] Enslow J PH. What is a "distributed" data processing system? *Computer* 1978; **11**(1):13–21, doi:10.1109/C-M.1978.217901.

[12] Bergsma M. Wikimedia infrastructure. 2007. Http://www.nedworks.org/~mark/presentations/san/Wikimedia architecture.pdf.

[13] Lua EK, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials* 2005; **7**:72–93, doi:10.1109/COMST.2005.1610546.

[14] Dourish P. The Parting of the Ways: Divergence, Data Management and Collaborative Work. *Proceedings of the European Conference on Computer-Supported Cooperative Work - ECSCW'95*, 1995; 215–230.

[15] Cap CH. Towards content neutrality in wiki systems. *Future Internet* 2012; **4**(4):1086–1104, doi:10.3390/fi4041086.

[16] Lowry PB, Curtis A, Lowry MR. Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research and Pratice. *Journal of Business Communication* January 2004; **41**(1):66–99, doi:http://dx.doi.org/10.1177/0021943603259363.

[17] Ignat CL, Oster G, Molli P, Cart M, Ferrié J, Kermarrec AM, Sutra P, Shapiro M, Benmouffok L, Busca JM, *et al.*. A comparison of optimistic approaches to collaborative editing of wiki pages. *CollaborateCom*, IEEE, 2007; 474–483.

[18] Balakrishnan H, Kaashoek MF, Karger D, Morris R, Stoica I. Looking up data in P2P systems. *Communications of the ACM* February 2003; **46**(2):43.

[19] Dabek F, Zhao B, Druschel P, Kubiatowicz J, Stoica I. Towards a Common API for Structured Peer-to-Peer Overlays. *Lectures Notes in Computer Science* 2003; **2735**:33–44.

[20] Saito Y, Shapiro M. Optimistic Replication. *ACM Computing Surveys* 2005; **37**(1):42–81, doi:http://doi.acm.org/10.1145/1057977.1057980.

[21] Lua EK, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials* 2005; **7**(1-4):72–93.

[22] Patrick Mukherjee CL, Schurr A. Piki - a peer-to-peer based wiki engine. *Eighth International Conference on Peer-to-Peer Computing*, IEEE, 2008; 185–186.

[23] Du B, Brewer EA. Dtwiki: a disconnection and intermittency tolerant wiki. *WWW '08: Proceeding of the 17th international conference on World Wide Web*, ACM: New York, NY, USA, 2008; 945–952, doi:http://doi.acm.org/10.1145/1367497.1367624.

[24] Mukherjee P, Leng C, Terpstra WW, Schürr A. Peer-to-peer based version control. *ICPADS*, IEEE, 2008; 829–834.

[25] Oaks S, Gong L. *Jxta in a Nutshell*. O'Reilly & Associates, Inc.: Sebastopol, CA, USA, 2002.

[26] Fall K. A delay-tolerant network architecture for challenged internets. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, ACM: New York, NY, USA, 2003; 27–34.

[27] Demmer MJ, Du B, Brewer EA. Tierstore: A distributed filesystem for challenged networks in developing regions. *FAST*, Baker M, Riedel E (eds.), USENIX, 2008; 35–48.

[28] Oster G, Urso P, Molli P, Imine A. Data Consistency for P2P Collaborative Editing. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2006*, ACM Press: Banff, Alberta, Canada, 2006.

[29] Eugster PT, Guerraoui R, Handurukande SB, Kouznetsov P, Kermarrec AM. Lightweight Probabilistic Broadcast. *ACM Transactions on Computer Systems* November 2003; **21**(4):341–374, doi:http://doi.acm.org/10.1145/945506.945507.

[30] Demers A, Greene D, Hauser C, Irish W, Larson J, Shenker S, Sturgis H, Swinehart D, Terry D. Epidemic Algorithms for Replicated Database Maintenance. *Proceedings of the ACM Symposium on Principles of Distributed Computing - PODC'87*, ACM Press: Vancouver, British Columbia, Canada, 1987; 1–12, doi:http://doi.acm.org/10.1145/41840.41841.

[31] Alshattnawi S, Canals G, Molli P. Concurrency awareness in a P2P wiki system. *International Symposium on Collaborative Technologies and Systems*, IEEE, 2008; 285–294.

[32] Weiss S, Urso P, Molli P. Logoot : a scalable optimistic replication algorithm for collaborative editing on P2P networks. *32nd International Conference on Distributed Computing Systems*, IEEE Computer Society, 2009; 404–412.

[33] Lamport L. Times, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM* July 1978; **21**(7):558–565, doi:http://doi.acm.org/10.1145/359545.359563.

[34] Sun C, Jia X, Zhang Y, Yang Y, Chen D. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction* March 1998; **5**(1):63–108, doi:http://doi.acm.org/10.1145/274444.274447.

[35] RepliWiki – A Next Generation Architecture for Wikipedia. http://isr.uncc.edu/repliwiki/.

[36] Canals G, Molli P, Maire J, Laurière S, Pacitti E, Tlili M. Xwiki concerto: A P2P wiki system supporting disconnected work. *CDVE*, *Lecture Notes in Computer Science*, vol. 5220, Luo Y (ed.), Springer, 2008; 98–106.

[37] Skaf-Molli H, Rahhal C, Molli P. Peer-to-peer semantic wikis. *DEXA*, *Lecture Notes in Computer Science*, vol. 5690, Bhowmick SS, Küng J, Wagner R (eds.), Springer, 2009; 196–213.

[38] Kang B. Summary Hash History for Optimistic Replication of Wikipedia. http://isr.uncc.edu/shh/.

[39] Krötzsch M, Vrandecic D, Völkel M, Haller H, Studer R. Semantic wikipedia. *Journal of Web Semantic* 2007; **5**(4):251–261.

[40] Rahhal C, Skaf-Molli H, Molli P, Weiss S. Multi-synchronous collaborative semantic wikis. *10th International Conference on Web Information Systems Engineering - WISE '09*, *LNCS*, vol. 5802, Springer, 2009; 115–129.

[41] Gollum. A wiki built on top of git. https://github.com/github/gollum.git 2010.

[42] Mendler D. Olelo, a wiki with a git backend, http://www.gitwiki.org/ 2013. URL http://www.gitwiki.org/.

[43] Torvalds L. Git - a fast version control system. http://git-scm.com/.

[44] Shapiro M, Preguiça N, Baquero C, Zawirski M. Conflict-free replicated data types. *13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2011.

[45] Krötzsch M, Vrandecic D, Völkel M. Semantic mediawiki. *International Semantic Web Conference*, *Lecture Notes in Computer Science*, vol. 4273, Cruz IF, Decker S, Allemang D, Preist C, Schwabe D, Mika P, Uschold M, Aroyo L (eds.), 2006; 935–942.

[46] Aslan K, Alhadad N, Skaf-Molli H, Molli P. Scho: An ontology based model for computing divergence awareness in distributed collaborative systems. *ECSCW*, Bødker S, Bouvin NO, Lutters WG, Wulff V (eds.), Springer, 2011; 373–392.

[47] Gallo G, Longo G, Pallottino S. Directed hypergraphs and applications. *Discrete Applied Mathematics* 1993; **42**(2):177–201.

[48] Craig A, Davoust A, Esfandiari B. A distributed wiki system based on peer-to-peer file sharing principles. *Web Intelligence*, 2011; 364–371.

[49] Bizer C, Heath T, Berners-Lee T. Linked data-the story so far. *International Journal on Semantic Web and Information Systems* 2009; **5**(3):1–22.

[50] Ibez LD, Skaf-Molli H, Molli P, Corby O. Col-graph: Towards writable and scalable linked open data. *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, October 19-23, Riva del Garda, Trentino, Italy*, 2014.

[51] Acosta M, Zaveri A, Simperl E, Kontokostas D, Auer S, Lehmann J. Crowdsourcing linked data quality assessment. *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, 2013; 260–276.

[52] Vrandečić D. A new platform for collaborative data collection. *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, ACM: New York, NY, USA, 2012; 1063–1064, doi: 10.1145/2187980.2188242. URL http://doi.acm.org/10.1145/2187980.2188242.

[53] Arapov P, Buffa M, Ben Othmane A. Wikinext: A wiki for exploiting the web of data. *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, ACM: New York, NY, USA, 2014; 727–734, doi: 10.1145/2554850.2554962. URL http://doi.acm.org/10.1145/2554850.2554962.

[54] Ibáñez LD, Skaf-Molli H, Molli P, Corby O. Live linked data: synchronising semantic stores with commutative replicated data types. *IJMSO* 2013; **8**(2):119–133, doi:10.1504/IJMSO.2013.056605. URL http://dx.doi.org/10.1504/IJMSO.2013.056605.