Vol.2016-IOT-32 No.7
2016/3/3

PIAXを用いたP2P型Annoteaサーバ”Wasabi”の設計と実装

山岸 一貴[1,a)]　安藤 大志[1]　萩原 威示[1]

概要：ウェブアノテーションとは，ウェブページに対し付与された注釈である．W3C のプロジェクトである Annotea は，ウェブアノテーションを扱うデータ型と通信プロトコルを定めている．本研究では，ウェブアノテーションを相互参照する P2P 型 Annotea サーバ”Wasabi”を，P2P フレームワークである PIAX を用いて設計・実装した．Annotea では，URI に対して注釈付けられたウェブアノテーションを検索して取得することができる．しかし P2P 環境において，ウェブアノテーション検索のたびにネットワーク内すべてのピアへ問い合わせをしていては，検索結果の収集に時間がかかることが予想される．そこでアプリケーションへの検索応答性能を良くするため，ウェブアノテーション検索時の Wasabi の協調動作の仕組みを設計した．URI に対して注釈付けられたウェブアノテーションをネットワーク内全ピアから収集することにより，ウェブアノテーションのキャッシュを作成する．ウェブアノテーション検索で，Wasabi は PIAX ネットワークからキャッシュを取得してそれを返答し，その後にキャッシュのアップデートを行う．Wasabi の動作実験をする．

キーワード：Annotea，P2P，Skip Graph

Implementation of P2P type Annotea Server ”Wasabi” using PIAX

Yamagishi Kazuki[1,a)]　Andou Taishi[1]　Hagiwara Takeshi[1]

Abstract: Web Annotation is an annotation that is appended to Web page. W3C Annotea defines data type and communications protocols to deal with an annotation. This study designed and implemented ”Wasabi” that is able to refer to an annotation with each other by using P2P framework PIAX. In Annotea, It can get an annotation by URI. It is expected that asking all peers of network takes response time in P2P environment. To improve performance of application responce, we designed the mechanism of Wasabi collaboration. An annotation cache is created by collecting an annotation from all peers of network. Wasabi gets an annotation cache from PIAX network. In an annotation searching, Wasabi gets an annotation cache and responds to application, after that Wasabi update an annotation cache. We experiment motion of Wasabi.

Keywords: Annotea, P2P, Skip Graph

1. はじめに

アノテーション技術やコメント技術は，様々な形で使用されている。代表的な例では，Youtube が提供する動画上に表示させることができるクリック可能なテキストといった Youtube アノテーション機能，ニコニコ動画における配信される動画再生軸上に対してユーザーがコメントを投稿できるコメント機能，Windows 10 で利用できる Web ブラウザ「Microsoft Edge」の機能として Web ページに直接手書きでメモを書き込み，そのメモの共有を行う，といった

[1] 新潟大学
Niigata University
[a)] k-yamagishi@cs.ie.niigata-u.ac.jp

ものがある．

ウェブアノテーションとはウェブページに対するオンライン付箋としての注釈やコメントであり，ウェブページにコメントしたり，付けられたコメントを閲覧したり，コメントのやりとりによってコミュニケーションを行う，といったことが可能となる．

Annotea は W3C 主導の，ウェブアノテーションに標準規格を定めるプロジェクトである．Annotea では Annotea プロトコル [1] を規格しており，これを実装したアノテーションサーバを Annotea サーバ，Annotea サーバと通信を行うクライアントアプリケーションを Annotea クライアントと呼ぶ．

我々の研究室では以前より Annotea を実装した Annotea サーバとして”Wasabi”の開発を進めている．Wasabi は個人用アノテーションサーバとしての利用を想定しており，他の Wasabi と P2P ネットワークを介してアノテーションの相互参照が可能となるように，P2P 通信フレームワーク PIAX[2] を用いて実装を進めている．

2. 予備知識

2.1 Annotea

Wasabi ではアノテーションサーバ実装のために Annotea に従った実装を行っている．ここでは Annotea について説明する．

Annotea は 2001 年頃に W3C で進められたプロジェクトである．Annotea は HTTP,RDF,XML を利用して構築されており、任意のウェブリソースに対するメモ，説明，コメントといったアノテーションを付加することが可能になる．Annotea においてアノテーションは、作成者や注釈日時といった注釈情報を持つプロパティ部と，コンテンツを持つボディ部によって構成される。プロパティにはアノテーションを区別するためのアノテーション ID が割り当てられる．

Annotea プロトコルは，**表 1** のようなプロパティを持つアノテーションを，クライアント・サーバ間で通信する手順を定めている．通信の際，アノテーションを RDF ／ XML 形式のデータで表現する．Annotea クライアントは，HTTP リクエストを用いてウェブアノテーションの Post,Query,Download,Update,Delete を Annotea サーバに要求する．Annotea サーバは要求に従いデータベースを操作する．

Annotea プロトコルでは，アノテーション ID の生成規則は実装者にゆだねられている．Wasabi におけるアノテーション ID の生成規則は，Wasabi の設計で述べる．

2.2 SkipGraph

Wasabi では，より早くアノテーション検索に応答するための，アノテーションキャッシュの仕組みを設計している．簡単に説明すると，アノテーション検索の際にはいずれかの Wasabi がもつアノテーションキャッシュを素早く取得しそれを返答するという仕組みである．Wasabi ではより早くアノテーションキャッシュ取得を行うため，P2P フレームワーク PIAX で使用可能な構造化オーバーレイネットワーク SkipGraph[3] を用いてキャッシュをもつピアの特定をしている．

Wasabi において SkipGraph に対してどのようなキーを登録し検索を行って最新のアノテーションキャッシュを取得しているか，というアノテーション検索の仕組みを 3.7 節で説明する．

3. Wasabi の設計

Wasabi では，複雑な設定なしで複数の Wasabi と協調動作することができるように，Annotea サーバの P2P 化を行った．Wasabi は，笹川 [4]・佐藤 [5] によって P2P フレームワーク JXTA[6] を用いて設計・開発されてきたが，現在 JXTA の開発がストップしており，古いフレームワークとなっていた．そこで使用する P2P フレームワークを PIAX へと変更し，開発を進めている．

3.1 システム構成

Annotea プロトコルにおいてアノテーション操作は，Post/Query/Download/Update/Delete の 5 つあり，それぞれアノテーションの新規投稿/検索/取得/更新/削除を行う．Wasabi は Annotea クライアントと通信できる Annotea サーバとして動作するよう，Annotea プロトコルに従い実装する．図 1 では，単体サーバでの構成を示している．要求に従いアノテーションデータベースを操作する．

Wasabi は P2P ネットワークに接続することで，ネットワーク内 Wasabi のアノテーションデータベースに保存されているアノテーションの参照が可能となる，P2P 型 Annotea サーバである．ネットワーク例を図 2 に示す．

表 1 アノテーションプロパティ
Table 1 Annotation Property

プロパティ名	プロパティ値	内容
annotatesTo	URI	ウェブリソースの URI
body	URI	コンテンツ
Description	URI	アノテーション ID
creator	テキスト	作成者
created	`yyyy-mm-ddThh:mm:ssZ`	作成日時

3.2 Wasabi の方針

Wasabi の協調動作の設計方針について述べる．Annotea クライアントからのアノテーション検索要求に対して検索結果を素早く返答でき，かつ P2P ネットワーク内すべての Wasabi のアノテーションデータベースを対象として検索できるようにする．だが，「検索返答時間を短縮すること」と，「P2P ネットワーク内全 Wasabi で検索に該当したアノテーションすべてを集めること」は両立できずトレードオフであるといえる．そこでアノテーションの検索結果を集めたキャッシュを作成し，アノテーション検索要求にはこのキャッシュを返答する，という仕組みを設計した．詳しい説明は 3.7 節で述べる．

アノテーション検索をいち早く行う場合はアノテーション検索を Wasabi へ要求すれば，ネットワーク内の他の Wasabi が作成し保持しているキャッシュを取得しそれを返答する．検索結果として返答されるアノテーションはキャッシュしてあるアノテーションとなるため確実性は保証されない．その後 Wasabi はバックグラウンドでネットワーク内全 Wasabi へアノテーション検索を行いキャッシュを作成する．そのため確実にネットワーク内全ての Wasabi のアノテーション検索結果がほしい場合はアノテーション検索要求をもう 1 度行う必要がある．

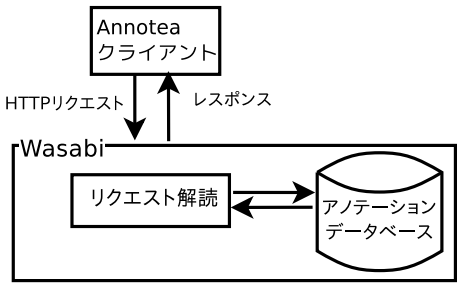

図 1　単体サーバ構成図

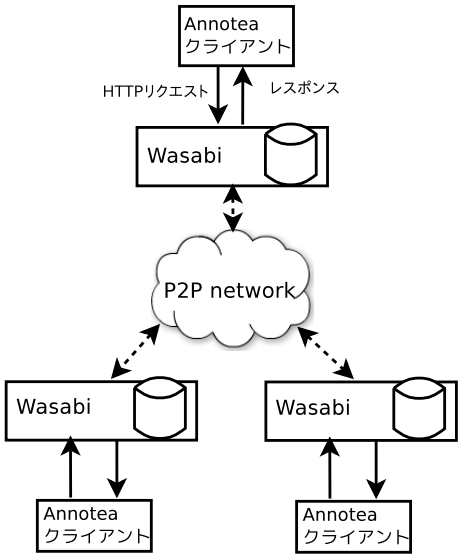

図 2　Wasabi ネットワーク例

表 2　アノテーションデータベーステーブル
Table 2　DataBase Table of Annotation

フィールド名	データ型	説明
recordId	bigint	(主キー) シーケンシャルナンバー
root	varchar	ウェブリソース URI
unixTime	bigint	投稿時刻
annotation	blob	プロパティ部
body	blob	ボディ部

3.3 Wasabi への一斉リクエスト送信

Wasabi は，PIAX ネットワーク参加時に「自分が Wasabi のノードであることを表すための固有キー」を SkipGraph に挿入する．他の全 Wasabi ノードに対するリモート操作は，このキーを指定し，discoveryCall という機能を用いることで行った．

3.4 アノテーションデータベース

アノテーションデータベーステーブルを**表 2** に示す．以下の操作を行う．

- recordId で指定した項目の更新／削除．
- recordId で指定したプロパティ部またはボディ部の取得．
- root としてウェブリソース URI を指定しプロパティ部データの一覧の取得．
- 最新のシーケンシャルナンバーの取得．

3.5 Wasabi におけるアノテーション ID

Wasabi で生成されるアノテーション ID について説明する．

Wasabi でのアノテーション ID には UUID，シーケンシャルナンバー，リソース種類，という 3 つの情報を埋め込む．

UUID　アノテーション用データベーステーブルを新たに作成するとき，ランダムな文字列を生成してそれを Wasabi の UUID とする．

シーケンシャルナンバー　データベースでアノテーションを識別するシーケンシャルな整数値．

リソース種類　リソース種類を区別するための文字列．”annotation”ならばプロパティ部，”body”ならばボディ部を表す．

例として `184a5e812613/29/annotation` というアノテーション ID が生成されアノテーションに割り当てられたとき，そのアノテーションは Wasabi の UUID が `184a5e812613` のデータベースに，シーケンシャルナンバーを 29 として保存される．

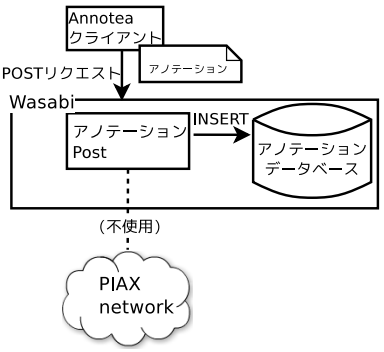

図 3　アノテーション Post の動作

3.6　UUID 指定によるリモートコール

Wasabi は，「UUID」をキーに SkipGraph に挿入する。アノテーション ID(上述) の UUID 部をもつ Wasabi の特定には，UUID をキーに指定し discoveryCall 機能を用いて行う．

3.7　アノテーション相互参照

3.7.1　アノテーション Post

アノテーション Post(投稿) は，リクエストを受信した Wasabi ノードのデータベースに保存する (図 3)．投稿時刻 (UNIXTIME) も共に保存する．

3.7.2　アノテーション Query

アノテーション Query(検索) について述べる．Wasabi では 3.3 節で説明した discoveryCall を用い，すべての Wasabi からアノテーションを収集することで，アノテーションキャッシュを作成する (図 4)．アノテーション Query リクエストを受信した Wasabi ノードは，SkipGraph を検索し，最新のキャッシュを取得し返答する (図 5)。その後キャッシュ作成を行い，URI と収集時刻の組をキーとして SkipGraph へ挿入する。

SkipGraph へ挿入するキーについて説明する．uri へのアノテーションを時刻 t に収集しキャッシュを作成した

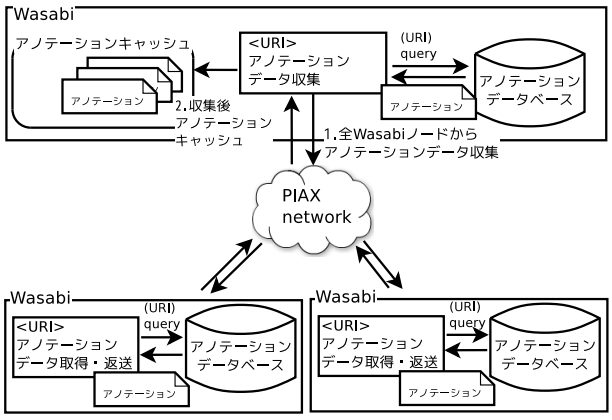

図 4　アノテーション収集の動作

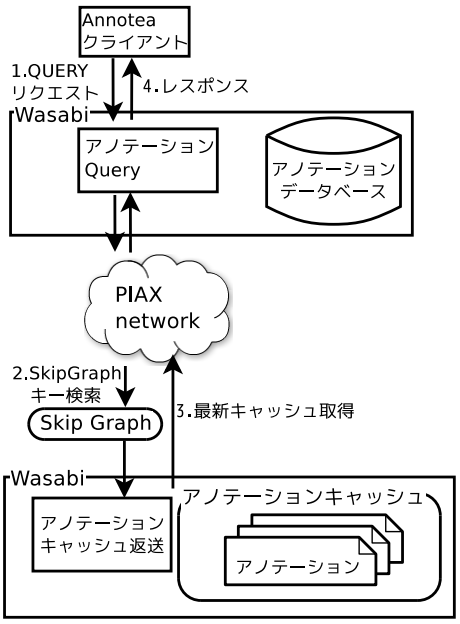

図 5　アノテーション Query の動作

場合，SkipGraph へ (uri, t) というキーを挿入する．このキーを持つ Wasabi は，「全ての Wasabi の，投稿時刻が 0 から t までの間に投稿された uri へのアノテーション」を持つ．なお，2 つのキー (uri_1, t_1) と (uri_2, t_2) の大小関係は uri_1 と uri_2 の大小関係により決定し，uri_1 と uri_2 が等しい場合は t_1 と t_2 の大小関係により決定する．

SkipGraph 検索とキー挿入までの流れを説明する．検索対象となる URI が uri で，検索リクエスト受信時の時刻が t_x であるとする．SkipGraph で「キー (uri, t_x) より小さい値の中で最大値を持つキー」を検索する．検索に該当したキーが (uri, t_a) であるとする．このキーを持つ Wasabi は，時刻 0 から t_a の間に投稿された uri へのアノテーション (キャッシュ) を持っているので，そのキャッシュを取得して Annotea クライアントに返答する．その後，discoveryCall を用いて時刻 t_a から t_x の間に投稿されたアノテーションを全 Wasabi から収集し，時刻 0 から t_a の間に投稿されたアノテーションと統合して新たにキャッシュとして保持し，SkipGraph にキー (uri, t_x) を挿入する．

Wasabi で SkipGraph に挿入されるキーの例 (Level0) を図 6 に示した．このとき，各 Wasabi ノード w_1, w_2, w_3 は以下のキャッシュを保持している．

w_1　時刻 15 までに投稿された uri_1 と，時刻 25 までに投稿された uri_3 のアノテーション

w_2　時刻 20 までに投稿された uri_2 と，時刻 15 までに投稿された uri_3 のアノテーション

w_3　時刻 30 までに投稿された uri_2 のアノテーション

3.7.3　アノテーション Update

アノテーション Update(更新) は，更新するアノテーショ

ン ID をリクエストに含む．アノテーションを所有している Wasabi ノードを特定しなければならない．

アノテーション ID は，データベース構築時に生成した UUID を含む．discoveryCall を用いて UUID を持つ Wasabi を特定する．

Update リクエストを受信した後，以下の動作を行う (図 7)．

(1) アノテーション ID の UUID 部を参照
(2) UUID をキー挿入した Wasabi ノードへのリモートコールを discoveryCall により行う
(3) リモートでアノテーションの Update を行う．

3.7.4 アノテーション Download

アノテーション Download(取得) は，取得するアノテーション ID をリクエストに含む．

Update 時と同じく discoveryCall を用いて UUID を持つ Wasabi を特定する．Download リクエストを受信した後，以下の動作を行う (図 8)．

(1) アノテーション ID の UUID 部を参照
(2) UUID をキー挿入した Wasabi ノードへのリモートコールを discoveryCall により行う
(3) リモートで Wasabi サーバからアノテーションの本体データを取得する．

3.7.5 アノテーション Delete

アノテーション Delete(削除) は，削除するアノテーション ID をリクエストに含む．

Update 時と同じく discoveryCall を用いて UUID を持つ

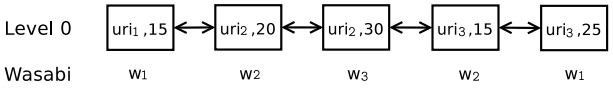

図 6 Wasabi で SkipGraph に挿入されるキーの例 (Level0)

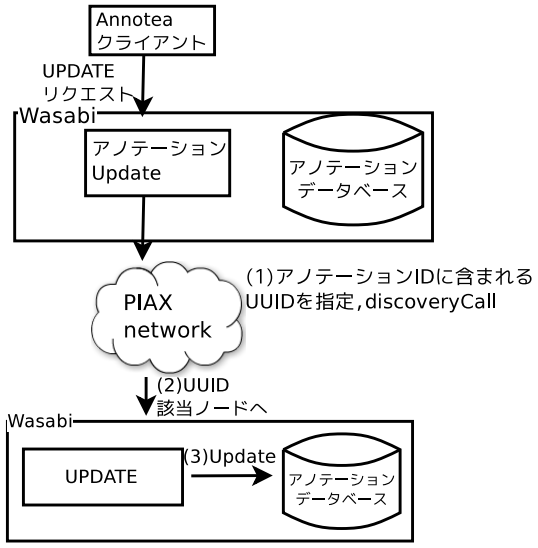

図 7 アノテーション Update の動作

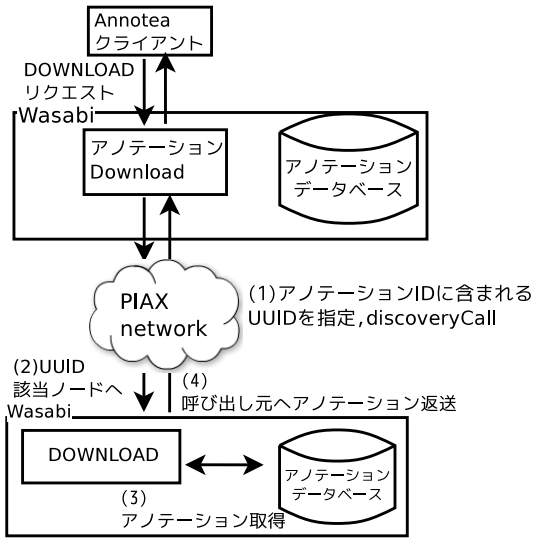

図 8 アノテーション Download の動作

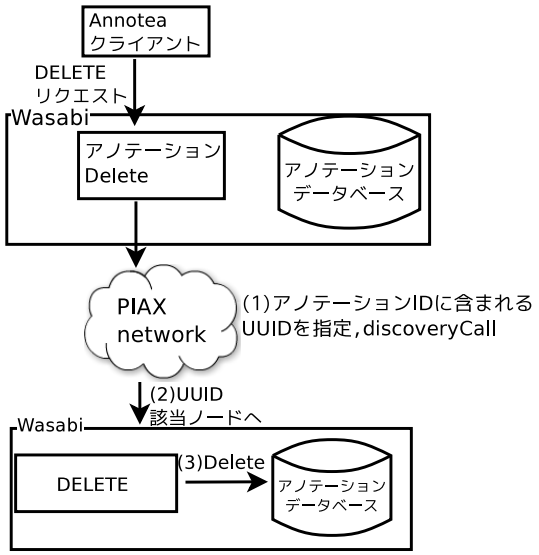

図 9 アノテーション Delete の動作

Wasabi を特定する．Delete リクエストを受信した後，以下の動作を行う (図 9)．

(1) アノテーション ID の UUID 部を参照
(2) UUID をキー挿入した Wasabi ノードへのリモートコールを discoveryCall により行う
(3) リモートでアノテーションの Delete を行う．

4. 動作試験

Wasabi では，3 節で述べたように検索応答時間を短縮することを目的としてキャッシュの仕組みを設計した．

試験では，コンテナ型の仮想化を行う Docker を用い，1 台のマシンで複数個の Wasabi を動作させる．

4.1 試験 A

ここではアノテーション検索の結果としてキャッシュを取得できること，返答されたキャッシュの内容が想定どおりの検索結果であることを試験し確かめる．

Wasabi サーバを 10 個立て，同じネットワークに接続する．それぞれの Wasabi を $w_1, w_2, ..., w_{10}$ とする．同じウェブリソース URI に対するアノテーションの投稿と検索を, 以下の順番で行う．

(1) w_1 に投稿した後 w_2 へ検索要求する．
(2) w_3 に投稿した後 w_4 へ検索要求する．
(3) w_5 に投稿した後 w_6 へ検索要求する．
(4) w_7 へアノテーション検索要求する．

アノテーション検索結果について，以下のことが想定される．

- (1) で w_2 が取得したキャッシュの内容は空.
- (2) で w_4 が取得したキャッシュの内容は，w_1 へ投稿したアノテーション.
- (3) で w_6 が取得したキャッシュの内容は，w_1, w_3 へ投稿したアノテーション.
- (4) で w_7 が取得したキャッシュの内容は,w_1, w_3, w_5 へ投稿したアノテーション.

動作試験を行い，アノテーション検索結果が想定どおりであることが確かめられた．

4.2 試験 B

ここでは，SkipGraph 検索による最新キャッシュの取得にかかる時間を計測し評価する．

Wasabi サーバを 30 個立て同じネットワークに接続する．それぞれの Wasabi を $w_1, w_2, ..., w_{30}$ とする．w_n へアノテーション投稿した後，w_n へアノテーション検索を行う．w_n がキャッシュ作成した後，w_{n+1} へアノテーション投稿・検索を行う．w_{30} まで繰り返す．

w_{30} が検索結果として取得したキャッシュは，$w_1, ..., w_{29}$ が投稿したアノテーションとなっていることが想定される．

この試験を行ったところ，w_{30} におけるアノテーション検索結果は想定どおりであることが確認された．

4.3 試験 C

大量のリクエストを処理中の Wasabi サーバが最新キャッシュを所有している場合に，他の Wasabi サーバがアノテーション検索を行ったとき，最新キャッシュが取得できるかどうかを試験し確認する．

Wasabi サーバを 30 個立て同じネットワークに接続する．それぞれの Wasabi を $w_1, w_2, ..., w_{30}$ とする．w_n へアノテーション投稿した後，w_n へアノテーション検索を行う．w_n がキャッシュ作成した後，w_{n+1} へアノテーション投稿・検索を行う．w_{30} まで繰り返す．この後，以下の操作を順番に行う．

(1) Annotea クライアントを用いて，w_{30} に多量の Post リクエストを処理させる．
(2) 新たに Wasabi サーバ w_{31} を立て，アノテーション検索リクエストを w_{31} に対して行う．
(3) 多量の Post リクエストを処理している最中の w_{30} が最新キャッシュを持っているので，この w_{30} からキャッシュを取得でき，その内容が $w_0, w_1, ..., w_{30}$ が投稿したアノテーションであるかどうかを確認する．

この試験を行ったところ，w_{31} は w_{30} からキャッシュ取得を行うことができ，キャッシュ内容も想定どおりであることが確認された．

5. おわりに

P2P フレームワーク PIAX を用いた，P2P 型 Annotea サーバ Wasabi の設計を本稿では説明した．Wasabi では，P2P 環境におけるアノテーション検索へ素早く返答することを目的としたキャッシュの仕組みを，PIAX で利用可能な構造化オーバーレイネットワーク SkipGraph を用いて設計した．動作試験を行い，Wasabi サーバ数 30 個まで，キャッシュの仕組みが想定どおりに動作していることを確認した．また最新キャッシュを持っている Wasabi サーバに処理が集中している場合でも，その Wasabi サーバからキャッシュが問題なく取得できることを試験により確認した．

P2P フレームワーク JXTA を用いた Wasabi の設計 [5] では，キャッシュを管理する Wasabi サーバが URI 毎に決められ，Advertisement と呼ばれるリソース広告機能を用いて，URI 毎にキャッシュを持つ Wasabi を広告していた．だが JXTA では広告収集のため広告検索に待ち時間が存在する．また，キャッシュ持ち主のネットワーク脱退を考慮してキャッシュのバックアップ機構を設計する必要があった．

PIAX を用いた本設計では，URI とキャッシュ時間を SkipGraph に登録することで，アノテーション検索において SkipGraph 検索が終わり次第検索に応答することができるようになった．また，たとえ最新キャッシュを持つ Wasabi サーバがネットワークから脱退しても，その次に新しいキャッシュを持つ Wasabi サーバが SkipGraph に登録されていればそのキャッシュをアノテーション検索で返答することができる．

今後は Wasabi の使用シチュエーションを設定してネットワーク環境をシミュレーションし，検索レスポンスの性能評価を行う予定である．

参考文献

[1] Annotea Protocols. http://www.w3.org/2001/Annotea/User/Protocol.html
[2] PIAX. http://www.piax.org/
[3] James Aspnes and Gauri Shah. Skip graphs. In Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 384393, Baltimore, MD, USA, 1214 January 2003.
[4] 笹川透, 瀬野瑛, 萩原威志.Annotea の P2P 型実装を用いた掲示板の構築 (2010)
[5] 佐藤慶太, 瀬野瑛, 萩原威志.JXTA を用いた自律分散型 Annotea サーバ Wasabi の実装 (2014)
[6] JXTA. https://jxta.kenai.com/
[7] 安藤大志, 山岸一貴, 萩原威示. Annotea サーバ”Wasabi”でのウェブアノテーション全文検索手法の試験実装 (2015)