位置コンテンツ共有システムの P2P モデルによる実装

寺西 裕一†, 川上 朋也‡, 石 芳正†, 春本 要§, 下條 真司†
† 大阪大学サイバーメディアセンター,
‡ 大阪大学 大学院情報科学研究科, § 大阪大学大学院工学研究科

概要

本稿では，Web ブラウザを用いてユーザが自由にコンテンツを追加・編集・閲覧が可能な位置コンテコンテンツ共有システム 'MapWiki' の P2P モデル に基づく実装について述べる．従来，MapWiki はサーバ・クライアント構成による実装が行なわれてきたが，この構成には，スケーラビリティ上の問題，収集情報の管理コストの問題，および，プライバシ上の問題があった．そこで本研究では，これらの問題を解決すべく，P2P 型位置コンテンツ管理機構 'Diploid' を開発し，MapWiki の P2P 実装を行なった．本実装を実機評価した結果，大幅にスケーラビリティを向上できることを確認した．

A P2P Implementation of Location-based Content Sharing System

YUUICHI TERANISHI†, TOMOYA KAWAKAMI‡, YOSHIMASA ISHI†,
KANAME HARUMOTO § AND SHINJI SHIMOJO †
†Cyber Media Center, Osaka University,
‡Graduate School of Information Science and Technology, Osaka Univeristy,
§Graduate School of Engineering, Osaka University

Abstract

In this paper, a new P2P implementation of the location-based Web content sharing system 'MapWiki' is described. Our former implementation of MapWiki was based on the legacy client-server architecture. In this implementation, there were mainly three issues that must be solved, scalability problem, management cost and privacy problem. To solve these issues, we have developed a P2P location-based content sharing mechanism called 'Diploid' to implement P2P-based MapWiki. We already finished early implementation and confirmed that the system vastly improved its scalability.

1 はじめに

近年の携帯型コンピューティングデバイスの小型化や高性能化，モバイルネットワーク技術の発達にともない，ユビキタスコンピューティングを実現する，いわゆるユビキタス環境が現実的なものとなりつつある．ユビキタス環境の実現により，だれでも，どこでもコンテンツを発信，受信できるようになる．このような環境では，限られたコンテンツ発信者が，一定の統制のもとコンテンツを体系的に提供する従来型のコンテンツ配信よりも，オープンなプロトコルのもと，個人が自由にコンテンツを配信可能な，いわゆる口コミ型のコンテンツ流通の重要性が増すことになると考えられる．

我々は，ユビキタス環境においてこうした口コミ型のコンテンツ流通を実現するための一手法として，地図上でだれでも簡単かつ自由に情報を追加・共有可能とする 'MapWiki' システムの研究開発を行なってきた [1, 2]．MapWiki により，ある場所にある店舗に関する口コミ情報や，その場で撮影した写真画像といった位置コンテンツを，地図上で自由に発信・共有することができる．

これまでに我々が開発してきた MapWiki システムは，単一のサーバ上にあらゆるコンテンツが保存され，参照されるサーバ・クライアント構成であった．

しかし，この構成には，まず，コンテンツへのアクセス数の増加に伴ないサーバの負荷が上昇し，表示速度が遅くなる問題があった．また，画像の動的な合成等，コンテンツに応じた複雑な処理が含まれるため，アクセス者が少人数であっても，利用コンテンツの量によっては処理が追い付かない場合があった．

また，MapWiki に追加されたコンテンツをユーザの状況にあわせ，実フィールドにおいて活用していく上では，その場の状態や，個人の状態を反映するために，センサや個人から発信される情報を，リアルタイムに扱えることが求められる．しかし，頻繁

に更新される位置情報や，膨大なセンサ情報などを，サーバ側で全て掌握し，管理していくことは困難である．

一方，ユーザの状況を取得する手法として，携帯電話などの機能を利用して，ユーザの位置，付近にある物の撮影画像，その場でのおすすめ情報などを入力・発信可能とする研究が進められてきている．こうした情報は，ユーザのプライバシに関わるものが含まれるため，サーバを運営する第 3 者に管理を委ねることなく知人や家族に知らせたい場合がある．

我々はこうした課題を解決するため，MapWiki の Peer-to-Peer (P2P) モデルに基づく実装を行なった．実装にあたって，我々の研究グループで研究開発を進めている P2P プラットフォームである PIAX [4] を用いて，位置コンテンツの分散管理機構 'Diploid' を開発した．本稿ではこの P2P モデルによる MapWiki の実装法および評価について述べる．

2 位置コンテンツ共有システム MapWiki

2.1 MapWiki の基本モデルと現状

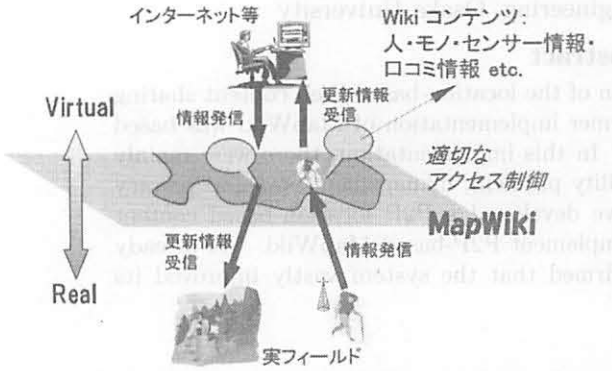

図 1: MapWiki の基本コンセプト

図 2: MapWiki によるコンテンツ追加例

Wiki は，Web ブラウザからユーザが自由に編集可能で，直感的に記述できる簡易言語でマークアップされる特徴をもち，多くのサイトで活用されている技術である．'MapWiki' はこの Wiki システムを共有地図上で実現し，その長所を生かしつつ，ユビキタス環境の口コミ型コンテンツ流通を実現する一種のコラボレーション環境を提供する (図 1)．MapWiki では，地図上に，だれでも，どこからでも自由にコンテンツを追加・編集可能とすることで，実フィールドに居るユーザ，インターネット上のユーザにかかわらず，自由にコンテンツの発信・共有を可能とする．コンテンツの表現および管理に Wiki を用いることで，直感的かつ容易にコンテンツを発信できる利点がある．高機能なツール等は必要なく，Web ブラウザ機能があれば携帯端末であってもコンテンツの発信が可能である．また，移動する人やモノに関する情報が通常の Wiki コンテンツと同様に地図上で閲覧でき，かつその状態は常に更新されるため，実フィールドの状態に合わせたコンテンツが得られる．

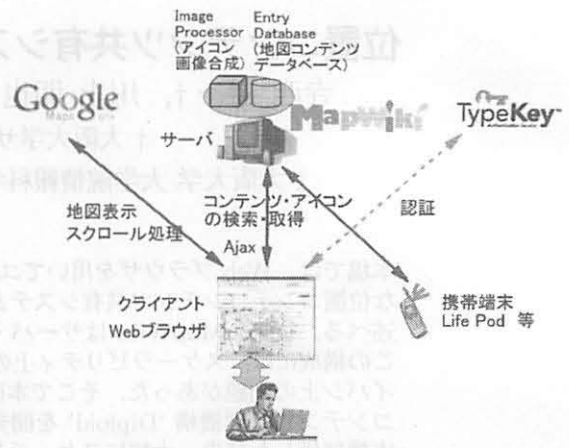

図 3: MapWiki のサーバ・クライアント構成による実装

これまでに我々は Google Maps API [7] を用いた MapWiki の実装を進めてきた[1]．この実装では，MapWiki のクライアントは Google マップと同様に JavaScript を用いて実現されており，特別なソフトウェアをインストールすることなく Web ブラウザ上で MapWiki を動作させることができる．この実装の MapWiki のシステム構成を図 3 に示す．図の通り，MapWiki のデータベースは，一つのサーバ上で稼働するサーバ・クライアント構成である．また，ユーザがコンテンツを発信した位置には，マーカとして地図上にアイコンが表示される．このアイコン画像はコンテンツやユーザに応じて変更可能であり，表示時に動的に合成される．例えば，定点カメラ映像にその場の温度を合成するといった応用も可能である (図 4)．

また，MapWiki に対して，PHS や携帯電話を用いたフィールド上のユーザが，情報の発信を手軽に行なうための開発も進めている．現在，携帯電話を入力機器とする Life Pod [6] から発信されるコンテンツを扱えるようにするシステム連携の開発を進めている (図 5)．Life Pod は，ユーザの位置，付近にある物の撮影画像，その場でのおすすめ情報などを，携帯電話から簡単に「ライフログ」として発信可能とするシステムである．Life Pod との連携により，フィールドに居るユーザが簡単に口コミ情報やその場の情報を発信し，共有することが可能となる．

2.2 従来実装の課題

MapWiki の実装および実験的な運用にともない，これまでにいくつかの課題が明らかとなってきた．以

[1] MapWiki のサーバ・クライアント構成による実装は次の URL で実験的に公開している．http://gmapwiki.org

図 4: アイコン画像の動的合成例

図 5: MapWiki と Life Pod の連携

下に，これらの課題について述べる．

- スケーラビリティ
  現状のシステム構成では，サーバ上で全てのコンテンツが保持・管理されているため，スケーラビリティの問題がある．個々のコンテンツの位置に基づく検索処理はデータベースシステムで行なわれ，個々の処理に大きなオーバヘッドはないが，アクセスが集中するとシステム負荷が高くなり，応答速度が低下する．また，前述のとおり，アイコン画像が画面表示時に動的に合成される際，コンテンツ量が増えると処理が追いつかず，表示速度が低下することがある．このような場合，サーバ設備のクラスタ化等，処理能力の向上による対処が考えられるが，莫大な構築コストがかかってしまう．

- 情報の収集・管理コスト
  MapWiki は Web ブラウザからのコンテンツの登録を可能としているものの，基本的にユーザによる手入力が必要である．位置を含めたユーザのプロファイルや，センサー情報のように頻繁に更新されるべき情報を，毎回変更があるたびに更新し続けることは困難である．また，センサのように，広範囲に設置されるモノが発信する膨大な情報を，全て集中管理することは現実的ではない．実際に利用されるかどうか分からないデータをサーバのストレージ上に保持することとなるため，無駄も大きい．

- プライバシ
  Life Pod のようなアプリケーションでは，ユーザの状況，すなわちユーザの位置，付近にあるモノの撮影画像，その場でのおすすめ情報などが，携帯端末から発信される．こうした情報にはプライバシに関わるものが含まれるため，サーバの運営者に管理を委ねず，ユーザのパソコン上に保存し，整理した上で友人や家族とのみ共有したい場合がある．また，サーバ上に全ての情報が格納される構成では，サーバが攻撃され，侵入されてしまうと，一度に全ての情報が流出してしまう危険も伴なう．

上記に挙げた課題は，いずれも，サーバ・クライアントベースでシステムを構築している限り解決することは本質的に難しい．我々は P2P モデルによるシステムアーキテクチャを取ることにより上記課題の解決を目指す．

3 P2P モデルによる実装

前節で述べたサーバ・クライアント構成による MapWiki 実装の課題を解決するため，我々は P2P モデルに基づく分散位置コンテンツ管理機構 'Diploid' の開発を行なった．Diploid は，我々の研究グループで研究開発を進めている P2P プラットフォーム PIAX を用いて開発されている．本節では，Diploid と，Diploid に基づく MapWiki の実現について述べる．

3.1 Diploid: P2P による位置コンテンツの管理

我々の研究グループでは，複数のオーバレイネットワークを構成可能なマルチオーバレイ機能と，分散エージェント機能とを融合させた P2P プラットフォーム PIAX の研究開発を行なってきた [4]．PIAX は，オーバレイネットワークとして，指定された地理的領域に含まれる資源をスケーラブルに探索する LL-Net[3] 機能を持っており，ユビキタスサービスの実現において重要となる，指定した位置の近傍にある資源の探索を効率的に行なうことができる．LL-Net では，主にピア (端末) が位置情報を持つオーバレイネットワークの構成法であるが，さらにピアが複数の位置情報を持つ場合についての対応も検討を進めている [5]．我々はこうした PIAX の機能を用いて，MapWiki などで用いることができる位置コンテンツ分散管理機構 'Diploid' の開発を行なった．

Diploid は PIAX の LL-Net の位置に基づくオーバレイネットワークの探索機能を用いることで，P2P モデルでの位置コンテンツの管理をスケーラビリティを保ちつつ実現する．また，基本的に P2P ネットワークに参加するだけで，自らが保持・管理する情報をネットワーク内のピアから発見させることができる．このとき，更新があるたびにサーバに情報を集約させるといった手間は不要であり，自ピア内のコンテンツを更新するだけで最新の情報を発信できる．さらに，コンテンツが基本的にユーザ端末上，もしくはユーザ自身が管理可能なコンピュータ上に存在することとなるため，プライバシの問題にも対応できる．

Diploid には，Heavy Peer 層と Light Peer 層の 2 つの Peer 層が存在する (図 6)．Heavy Peer 層は，PIAX の P2P ネットワークに参加し，LL-Net などのオーバレイネットワークの機能を実現するピア群である．一方，Light Peer 層は，P2P ネットワークの機能は持たず，Heavy Peer 層のピアに P2P としてオーバレイネットワークの検索機能を委託するピア群である．Light Peer は，Heavy Peer に寄生するかたちで存在するピアであるといえる．

Heavy Peer が保存する情報としては，実際にコンテンツデータの提供を行なう Real Entry と，実際の

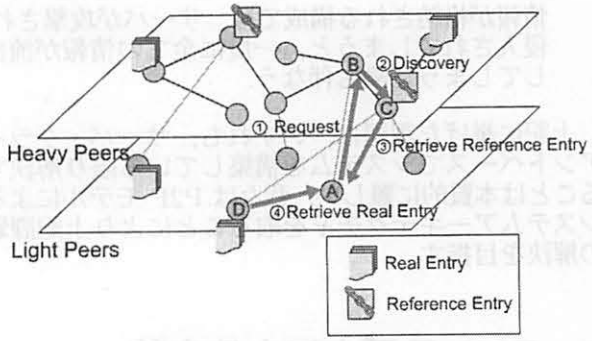

図 6: Diploid の構成

コンテンツデータは持たず，他のピアが持つコンテンツへの参照のみを扱う Reference Entry の 2 種類が存在する．

ネットワーク接続が安定しないピアは Light Peer となり，かつ自らコンテンツを保持せず，Heavy Peer に Real Entry を登録することを想定している．また，センサのように自身のデータが頻繁に更新されるが，処理能力的に P2P ネットワークのピアとしての能力を提供できない場合は，Light Peer として参加し，Heavy Peer に自ピアのコンテンツへの Reference Entry を登録する．この場合，Reference Entry が探索されて当該データへの参照が起きたとき，コンテンツデータを相手に送信する役割のみを果たす．

ユーザのデスクトップ PC 端末など，常時接続する端末は，Heavy Peer となるが，文献 [5] にも示している通り，オーバレイネットワークとしての探索処理の分担効率上，他のピアが探索のインデックスを持ち，問い合わせに応答する方が良い場合が考えられる．この場合は，他の Heavy Peer に Reference Entry の保持を委託することとなる．

Life Pod のようなアプリケーションでは，ユーザが出先で携帯電話などから情報を発信するが，こうした場合の携帯電話は Light Peer であり，例えば自宅の PC 端末やホームサーバがその Heavy Peer となってコンテンツの Real Entry を登録するかたちとなる．これにより，サーバのような第 3 者を介することなく，ユーザの情報をユーザ自身が管理することができる．

図 6 において，ピア A，ピア D は Light Peer であり，ピア B，ピア C は Heavy Peer である．ピア D は ピア C に Reference Entry を置いている．例えばピア A からピア B へ探索依頼が出され，ピア C 上にあるピア D の Reference Entry が発見されたとする．このとき，ピア B がピア A に応答として Reference Entry を送信する．ピア A が得られた応答の内容にアクセスすると，この Reference Entry が指すピア D 上の Real Entry がピア A へ転送される．

3.2 実装

我々は，前述の MapWiki におけるコンテンツ共有機能を，Diploid を利用して実装した．ただし，サーバ・クライアント構成において用いている Web シングルサインオンサービス TypeKey[9]，地図表示エンジン Google Maps API については P2P 構成においても引き続き利用している．すなわち，Web ブラウザ上での認証，地図表示については，サーバからサービスが提供され，地図上に表示されるコンテンツは，P2P のピアからサービスが提供される構成である (図 7)．

この構成による実装において問題となるのは，Web ブラウザから P2P ネットワークに接続し，コンテンツの探索を行なうための手段である．通常の Web ブラウザではセキュリティ上の配慮から，Java Script の主な通信手段である XMLHttpRequest は配信元のページ以外のサーバと通信を行なうことができない．したがって，Google Maps API を利用したページを配信する Web サーバ以外とは通信できないことになる．[2]．配信元のサーバが P2P ネットワークとのゲートウェイの役割を果たすことも考えられるが，それでは結局サーバにボトルネックが残ることとなってしまう．また，各端末に，Google Maps API を利用したページを配信する Web サーバを動作させる方法も考えられる．しかし，Google Maps API を利用する Web サーバは，サーバ毎に API Key と呼ばれるキーを取得する必要があるため，各端末ごとに API Key を取得しなければならなくなり，現実的ではない．

そこで本実装では，配信元の Web サーバ以外のピアとの間で Java Script により通信を行なう回避策として，JSONP[8] と呼ばれる手法を用いた．JSONP は，Java Script によって script タグの動的生成を行ない，通信相手が動的に生成するスクリプトを Web ブラウザ上に明示的にロードさせ，コールバック関数を起動することで，任意の相手と通信を行なう方法である．本実装では，ユーザ端末上で動作する Heavy Peer に，HTTP サーバの機能を組み込み，端末上で JSONP のスクリプトサーバと P2P ピアを兼ねるプロセス，クライアントプロセス (Web ブラウザ) の両方が動作するようにすることで，Web ブラウザから P2P ネットワークへ接続できるようにした．

また，先述の通り MapWiki にはアイコン画像の動的な合成機能があり，これがサーバにのみ存在することがスケーラビリティ低下の一因となっていた．アイコンには，カテゴリとタイトルさえ分かれば生成できるような，どのピアで処理しても同一の画像を生成できるタイプのアイコン (図 4 の (a) がこのタイプ) と，カメラ映像にその場の情報をアノテー

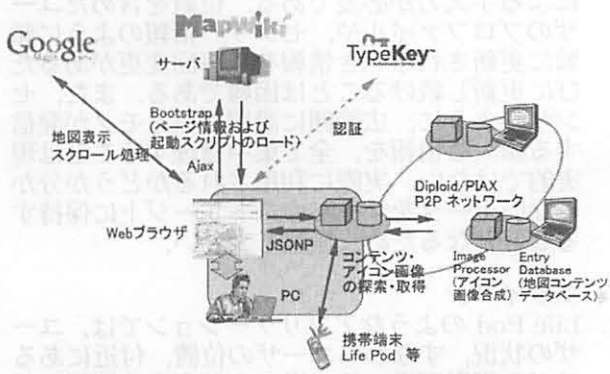

図 7: MapWiki の P2P 構成による実装

[2] Java の署名つきアプレットを利用するという方法も考えられるが，ここでは Java Script による実装を対象とし，候補から除外する．

図 8: Windows における P2P プログラムの動作

ションとして合成するといった，コンテンツ発信元ピアでなければ生成できないタイプのアイコン (図 4 の (b) がこのタイプ) がある．本実装では，前者を common 型，後者を original 型のアイコンとして分類し，各コンテンツに common 型か original 型かを区別するタグを付け，common 型の場合は，ローカルのピア上で合成処理を代行することとした．これにより，common 型のアイコンについては他のピアとの間で画像転送の通信が行なわれず，性能を向上させることができる．

図 8 は，Windows XP 上で Diploid のピア (Heavy Peer) として動作する P2P プログラムを動作させている様子を示している．本実装では，図のようにスタンドアロンのプログラムがデスクトップ上のシステムトレイで動作し，ユーザ自身が保持するコンテンツを管理できる形態となる．

4 評価

図 10 は，図 9 に示した各システム構成における MapWiki の処理性能を示している．本評価では，1 台の Web サーバ，4 台の PC 端末を用いている．図 10 および図 9 における'Server/Client' は，従来の MapWiki のサーバ・クライアント構成を示しており，'P2P (4 peers)' は，Diploid による P2P モデルの MapWiki 実装において，4 つの PC それぞれにピアを配置した構成を示している．

本評価では，地図上にランダム (一様) にコンテンツを 100 個配置し，ユーザの地図ドラッグ操作により平均 10 個の画像 (アイコン) が画面表示されるよう設定している．P2P 構成では，各ピアに 4 等分してコンテンツを保持させた．表示される画像は，コンテンツの名前をラベルとして含む common 型のアイコンである．各アイコンは，サーバ・クライアント構成ではサーバ上で，P2P 構成では，各端末ピア上で，それぞれ合成処理される[3]．P2P 構成では，各ピアで表示する画像の合成は自ピアで処理する．図 10 における「処理時間」とは，ユーザが地図ドラッグ操作を行なってから，画面の範囲に存在するコンテンツを P2P 上で探索した上で，該当するコンテンツの内容，および，合成された画像 (アイコン) の取り寄せを完了するまでの時間を示している (数値は，5 回の表示処理の平均処理時間)．それぞれ，Pentium M 1.7GHz の CPU を持つパソコン (ThinkPad X31) 上

[3]サーバ上での画像処理には，Ruby と ImageMagick (RMagick) を用いており，各ピア端末上では，Java 2D のグラフィック処理機能を用いている．

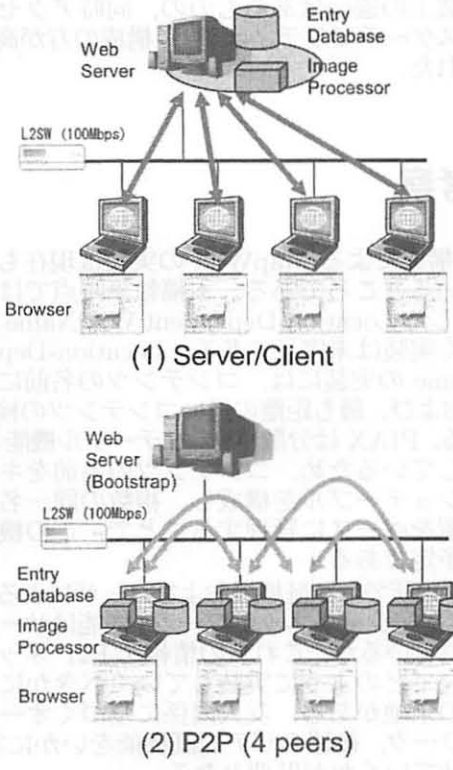

図 9: 評価システム構成

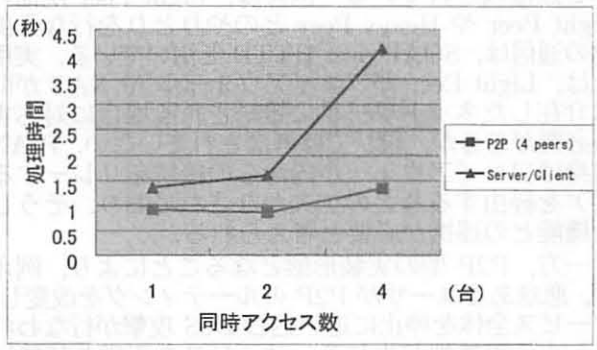

図 10: 実行結果

で動作する Mozilla Firefox をブラウザとして用い，サーバには Pentium 4 3GHz の PC サーバマシンを用いて測定している．

図 10 の通り，サーバ・クライアント構成では，同時アクセス数が 4 に増えると処理時間が極端に大きくなっている．検索処理およびアイコンの生成が同一サーバ上で同時に行なわれ，負荷が高くなっているためである．4 クライアントが同時にアクセスした状態では，約 4 秒の表示時間がかかっている．[4]

一方，P2P モデルによる実装では，負荷分散の効果により，同時アクセス数にかかわらずサーバ・クライアント型よりも処理時間は短い．また，同時アクセス数が 4 であっても，同時アクセス数が 1 のときとほとんど処理時間は変わっていない．以上によ

[4]通常，画面上には一度のドラッグ処理で高々 2〜3 個のコンテンツが表示される程度であるとすると，本評価における処理数は，通常の約 5 倍程度，すなわち，20 同時アクセス程度相当とみなすことができる．

り，実装上の違いはあるものの，同時アクセス数に対するスケーラビリティは P2P 構成の方が高いことが示された．

5 考察

P2P 構成による MapWiki の実装は現在も開発を進めているところである．本稿執筆時点では，文献 [1] に示した Location-Dependent WikiName の P2P に基づく実装は未完了である．Location-Dependent WikiName の実装には，コンテンツの名前に基づく検索，および，最も距離の近いコンテンツの検索が必要となる．PIAX は分散ハッシュテーブル機能 (DHT) を実装しているため，コンテンツの名前をキーとしたハッシュテーブルを構成し，複数の同一名エントリを位置をベースに管理することで，この機能を実現する予定である．

また，現状では認証機能およびユーザによるグループ形成 (ソーシャルネットワーク) 機能はサーバ上に実装されているが，これらの情報を P2P ネットワークにおいてどのように実装していくべきかについては検討の余地がある．友人関係に基づくオーバレイネットワーク，後述のピア認証機能をいかに実装し，連携させていくかが肝要となる．

Diploid の初期実装はほぼ完了しているが，改良すべき点は残されている．例えば，Light Peer が他の Light Peer や Heavy Peer とのやりとりを行なう部分の通信は，SOAP over HTTP を用いている．実用上は，Light Peer がファイアウォール や NAT が間に介在したネットワークに接続される場合に対応する必要があるが，現状では考慮されていない．PIAX 自身はファイアウォール内からの接続をリレーするピアを経由するなどの工夫がなされており，そうした機能との連携が必要と考えられる．

一方，P2P 型の実装形態となることにより，例えば，悪意あるユーザが P2P のルーティングを改変し，サービス全体を停止に追い込む DoS 攻撃が行なわれてしまう可能性が生じる．サービスを正常な状態に保つためには，P2P ネットワーク層において，PKI 等をベースとして信頼できるピアのみ接続を承認するといった機能を実現する必要があると考えている．

また，P2P 対応により膨大に存在するセンサ等が保持するデータを扱えるようになると，さらに，それらをどのように活用していくかを検討する必要がある．センサー等から得られる生データはそのままでは利用することは困難であるため，活用するための工夫を検討しなければならない．例えば，気象センサにおいて降水量が一定時間以上 0 であり，湿度が一定であれば「晴れ」という内容を持つコンテンツを提供するといった，データの値やルールに応じたコンテンツの生成やコンテンツ間の連携を実現していくことが考えられる．

一方，P2P 対応となっても，利用頻度が高いコンテンツに対しては，アクセスが集中してしまうという問題は残る．現状の実装では，全てのコンテンツは発信者のピア上に存在するが，個人のプライバシに関わる情報でなく，頻繁に更新されない場合には，別のピアに当該コンテンツをキャッシュする等の対応により，システム全体としての効率を向上できると考える．

6 まとめ

本稿では，Web ブラウザを経由してユーザが自由にコンテンツを追加・編集・閲覧が可能な位置コンテンツ共有システム 'MapWiki' の，P2P 分散コンテンツ管理機構 'Diploid' による実装およびその評価について述べた．

今後，前節にあげた検討を進めるとともに，運用における耐障害性や安定性，ユーザビリティを高め，より実用性のあるシステムとしていきたい．

謝辞

本研究の一部は，平成 17,18 年度総務省「ユビキタスネットワーク認証・エージェント技術の研究開発」の研究助成によるものである．また，同「ユビキタスネットワーク制御・管理技術の研究開発」成果である 'Life Pod' を，入力システムとして活用させて頂いている．ここに記して謝意を表す．また，Diploid の初期実装の開発にご協力いただいたオランダデルフト工科大学の Jeremy Raes, Menno Valkema, Niels Brouwers に深謝の意を表す．

参考文献

[1] 寺西 裕一, 鎌原 淳三, 下條 真司: MapWiki: 共有地図を用いたユビキタスコンテンツ流通環境, 情報処理学会第 13 回マルチメディア通信と分散処理ワークショップ論文集 (2005).

[2] Yuuichi Teranishi, Junzo Kamahara, Shinji Shimojo: MapWiki: A Map-based Content Sharing System for Distributed Location-dependent Information, *Academy Publisher: JOURNAL OF COMPUTERS*, Vol. 1, issue 3, pp. 13–19 (2006).

[3] 金子 雄, 春本 要, 福村真哉, 下條真司, 西尾章治郎: ユビキタス環境における端末の位置情報に基づく P2P ネットワーク, 情報処理学会論文誌：データベース, Vol. 46, No. SIG18(TOD28), pp. 1–15 (2005).

[4] 吉田 幹, 寺西 裕一, 春本 要, 下條真司: マルチオーバレイと分散エージェントの機構を統合化した P2P プラットフォーム PIAX, 情報処理学会研究報告：2006-DPS-128 (2006).

[5] 川上朋也, 三原慶彦, 寺西裕一, 春本 要, 下條真司: ユビキタス環境における位置依存情報の分散管理機構の設計と実装, マルチメディア, 分散, 協調とモバイル (DICOMO2006) シンポジウム, pp. 325–328 (2006).

[6] 本庄勝, 森川大補, 西山智, 大橋正良: セマンティックスを用いた携帯端末で取得されるライフログ管理基盤の検討, 情報処理学会研究報告：2006-UBI-5-35, Vol. 2006, No. 14, pp. 203–208.

[7] Google: Google Maps API, http://www.google.com/apis/maps/.

[8] Ippolito, B.: Remote JSON - JSONP, http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp.

[9] six apart: TypeKey Authentication Service, http://www.sixapart.jp/typekey/.