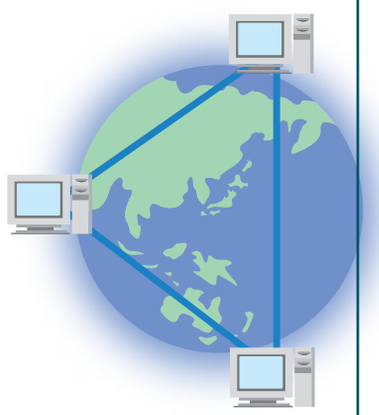
7

Peer-to-Peer技術



斉藤 賢爾
慶應義塾大学政策・メディア研究科
ks91@sfc.wide.ad.jp

飯村 卓司
奈良先端科学技術大学院大学情報科学研究科
takuji-i@is.naist.jp

WIDEプロジェクトのワーキンググループの1つであるIDEON（Integrated Distributed Environment with Overlay Network）では，IP（Internet Protocol）上のピアグループによるオーバレイネットワークに注目し，Peer-to-Peer（P2P）技術にかかわるさまざまな研究開発活動を行っている．

本稿では，その中から分散ゲーム環境構築を目的としたインフラ構築ライブラリであるlibcookaiとP2P補完通貨である*i*-WATについて紹介する．

WIDEプロジェクトとPeer-to-Peer技術

WIDE プロジェクト IDEON ワーキンググループは，「統合分散環境（Integrated Distributed Environment）」という WIDE プロジェクトの目標の実現に対し，「オーバレイネットワーク（Overlay Network）」という道具を用いてアプローチする有志が集うワーキンググループである．

IP ネットワークはグローバルな接続性を提供するが，多くのアプリケーションでは，IP ネットワーク上にアプリケーション固有の仮想的なネットワーク（オーバレイネットワーク）を形成し，その上で通信を行うことにより機能を実現している．IRCNet や，SMTP における MTA の網などはその典型的な例である．

IDEON では，このようにアプリケーションごとに独自に形成してきたオーバレイネットワークを整理・系統立てた上で，さまざまなアプリケーションで用いることができる共通のインフラストラクチャを構築することを最終目的としている．

本稿では，その活動の中から分散ゲーム環境構築を目的としたインフラ構築ライブラリである libcookai と P2P 補完通貨である *i*-WAT について紹介する．

libcookai：分散ゲーム環境構築を目的としたインフラ構築ライブラリ

ネットワークの広帯域化により，ネットワークを用いたゲームが一般的となった．この中で多人数が同時に同じゲーム世界を共有する Multi-player Online Game（MOG）と呼ばれるオンラインゲームは，ネットワークを用いて遠方の友人と同時に遊ぶことができるなど，ネットワークを十二分に活かしたゲームとして注目を集めている．

MOG のシステムの多くはサーバに中央集権的に処理を集中させるクライアント･サーバ方式で動作している．この方式は，データの管理などのゲーム上の処理を一極集中することができデータの一貫性を取りやすいなどの利点があるが，参加ユーザの数が増えることによってサーバに求められる計算能力やデータ容量，ネットワーク帯域などが増大することにより，クライアント・サーバ

型の運用には限界が生じてしまう．そこで，サーバに一極集中する処理を増大するクライアント・ノードへと分散させることができれば，クライアント・ノードの数に応じて増えていたサーバの負荷を，クライアント・ノードの数が増えるに従って減らすことができると考えられる．また，サーバにおける処理をクライアント・ノードへと分散させることによって，サーバが存在することによるサーバ停止の問題も同時に解決できると考えられる．

サーバで扱われるデータと Zone

ゲームクライアントはプレイヤーにゲーム世界を演出するためにそのゲーム世界を記述したデータを用いる．したがって，多人数が同時に同じゲーム世界を共有するためには，ゲーム世界を表現するデータを共有する必要がある．また，ある 1 つのクライアント・ノードはゲーム世界のすべてを見渡す必要はない．これは，あるクライアント・ノードが必要とするデータは共有されているデータのすべてではないことを意味する．以上のことから，クライアント・サーバ型ゲームにおけるサーバはゲーム全体のデータを管理し，個々のクライアント・ノードへとそのクライアント・ノードが必要とするデータを伝える仕事としていると考えることができる．この時，ゲーム全体のデータを Global Status Data（GSD）と呼ぶ．

GSD は個々のゲームの性質によってさまざまな構造を持つことが考えられる．たとえば，ユーザ・キャラクタの名前，身長などのキャラクタに分類されるもの，ゲームが日本を題材にしたものであれば東京や大阪などの土地に存在する建物のデータなどである．また，クライアント・ノードが必要とするゲーム世界を表現するデータは GSD の一部のみであることから，GSD には構造と局所性があることが分かる．

ここで，GSD を分割し別々のノードが分割された GSD を管理したとしても，クライアント・ノードは自らに必要な GSD の一部が得られることによってゲーム世界を共有したと錯覚することができる．このとき，GSD の構造も考慮して分割することにより，局所性のみの時よりもより柔軟な分割が可能であると考えられる．この分割された GSD を Zone と呼ぶ．

Zone への分割には分散ハッシュテーブル（Distributed Hash Table：DHT）の技術を使用する．DHT は参加しているノードそれぞれがハッシュテーブルを分割して管理し，ハッシュキーに対する検索という形式で目的のハッシュキーのデータを持っているノードを効率的に探し出すことができる手法である．本提案では個々の Zone それぞれに異なるハッシュキーを割り当てることでこれを利用する（図 -1）．

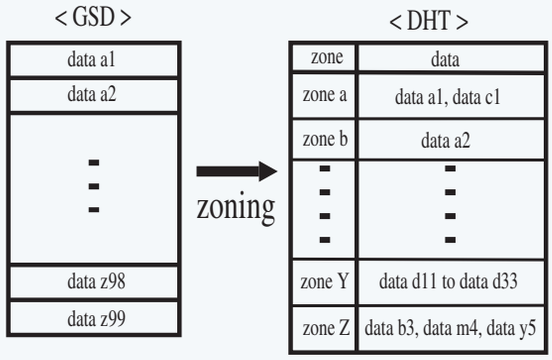


図-1　GSDのZoneへの分割

Zoned Federation Model

以上のことから Zoned Federation Model（ZFM）を提案する．ZFM はネットワークゲームにおけるサーバに要求される機能を満たしたまま，サーバに集中する負荷を Zone というデータの分割単位を用いて効率的にクライアント・ノードに分散させることでサーバに集中する負荷を減少させることを目的とする．

個々の Zone のデータは GSD と比べると十分に少なくできることから，1 つの Zone をクライアント・ノードが処理することが可能となる．したがって，ZFM ではこれら Zone の管理をクライアント・ノードへと分散する．これにより，クライアント・サーバ型でのサーバの役割であるデータ管理をクライアント・ノードへと分散させることが可能となる．

Zone のデータを管理するとき，複数のクライアント・ノードで管理する場合には調停などのオーバヘッドが発生する．これに対して Zone のデータの管理を 1 つのクライアント・ノードで行う方が調停などのオーバヘッドが存在せず，より高速にデータを扱うことができる．ZFM においては応答性に敏感なゲームでの使用を考慮するため，オーバヘッドが少ない方法をとる．したがって Zone のデータを管理するノードは 1 つにすべきである．このデータを管理するノードを Zone owner と呼ぶ．また，Zone のデータを必要とするノードを Zone member と呼ぶ．Zone member は Zone owner を経由して Zone のデータを共有することでゲーム世界を共有す

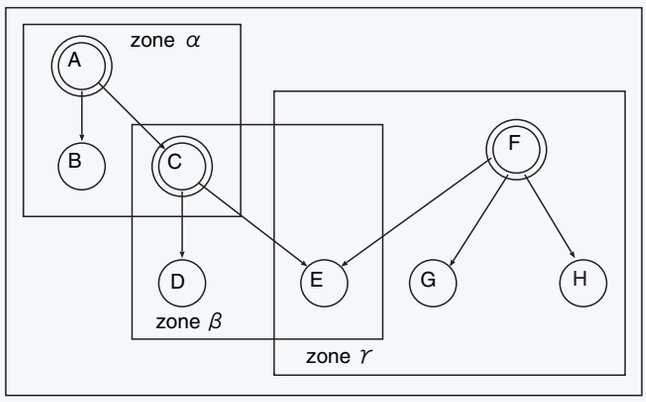


図-2　Zoneの利用形態

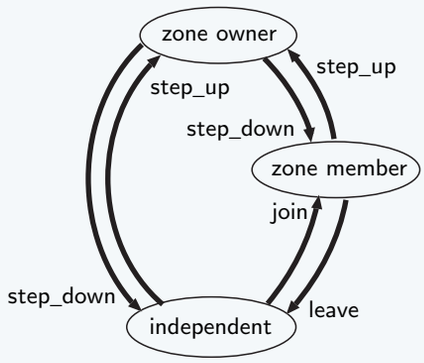


図-3　ノードの状態遷移

る．このとき，Zone member と Zone owner は Zone のデータを扱うという小規模のクライアント・サーバであると見ることができる．また，応答性に敏感なゲームでの使用を考慮して Zone member と Zone owner の間を直接接続する．これにより，Zone 内部の情報の伝達を高速に保つことができる．

ZFM の動作を図に表すと図 -2 のようになる．図 -2 における四角の枠が Zone を表し，それぞれの丸がノードを示している．このとき，2 重の丸で示されるノードは各 Zone での Zone owner を示しており，Zone owner から矢印でつながれているノードはその Zone における Zone member である．図 -2 のように，あるノードは複数の Zone に関与しており，ある Zone では Zone member であるが，別の Zone では Zone owner であるノード，というものも存在する．

ZFM においては個々のノードはそれぞれの Zone において Zone owner，Zone member，Zone に関係ないノード（Independent node），の 3 つの状態を持つ．この，それぞれ 3 つの状態から別の状態へと遷移することを図 -3 に示すように step_up，step_down，join，leave と呼ぶ．

上記の状態遷移動作の 4 つに加え，ゲームワールドへの接続 (join)，Zone owner によるデータの更新 (update)，Zone member によるデータの更新要求（commit）の 3 つを加えた 7 つが，ZFM で規定する関数である（表 -1）．

個々の Zone における Zone owner や Zone member のリストは DHT 上の Zone 名をハッシュキーとしたノードに格納される．ZFM では DHT 上に記録されるデータにリスト構造を持たせることによってこれを実現してい

関数	概要
initialize ()	ゲームワールドへ接続する
step_up (zone)	zone ownerへとstep_upする
join (zone)	zone memberとなり， zone ownerからのデータ更新情報を受け取る
updete (zone , data)	zoneのデータを書き換え， zone memberへ更新情報を伝え， DHT上のデータを書き換える
commit (zone , data)	zone memberから zone ownerへデータの変更要求を伝える
release (zone)	zone memberから independent nodeへと変化する． zone ownerとの接続は切られる
step_down (zone)	zone ownerから independent nodeへと変化する． zone memberとの接続は切られる

表-1　ZFM API

る．したがって Zone owner も Zone member もリスト構造を持っており，Zone owner はリストの最初に記録されているものが正式な Zone owner であることになっている．また，Zone におけるゲーム内部データも同じように格納されており，これは最後に記録されているものが正式なデータであることになっている．DHT 上のリストを制御するための動作を読み出し（DHT_get）追加（DHT_add），削除（DHT_del）の 3 つで表すと，Zone owner となるための step_up の非常に簡単な擬似コードは図 -4 のようになる．

ZFM において 1 つの Zone に Zone owner や Zone member が 1 つも存在しないということは容易にあり得る．このとき Zone のデータを保持するのは DHT 上での Zone 名をハッシュキーとしたノード（データ保持ノード）である．この DHT 上のノードが Zone owner でない限りは，そのデータを書き換えることはできない．また，Zone owner はデータの変更権限を持っていることから，Zone owner が存在する場合は一番新しい Zone のデータを持っているのは Zone owner であり，DHT 上で示されるデータ保持ノードはそのデータの古いコピーである．そこで，Zone owner は自分がいつ step_down してもよいように DHT 上で示されるデータ保持ノードへとデータの更新を行う．したがって，Zone owner がデータの更新を行うときには Zone member へのデータ更新の通達と，DHT 上で示されるデータ保持ノードへのデータ更新の 2 つを同時に行う．これを行うことで Zone owner や Zone member が存在しなくなってもその Zone のデータは常に最新に保たれる．

```
function step_up(zone){
  DHT_add(zone, "owner:" + myID);
  foreach data (DHT_get(zone)){
    if(data =~ s/^owner://){
      if(data == myID){
        /* step_up 成功 */
        return true;
      }else{
        DHT_del(zone, "owner:" + myID);
      }
      break;
    }
  }
  /* step_up 失敗 */
  return false;
}
```

図-4　step_up()の疑似コード

libcookai

以上の ZFM を実装したものが libcookai であり，以下の URL から入手できる．

http://libcookai.sourceforge.net/

libcookai は C 言語で記述されるライブラリである．表 -1 をすべて実装し，DHT の実装も含んでいる．

i-WAT：P2P補完通貨

P2P と補完通貨

P2P の考え方は，コンピュータの，普段は活用されていない計算能力，記憶容量，帯域といった資源をネットワーク上に解放し，その力を必要とする他のコンピュータのために使い，有効に活用するという側面を持っている．

P2P システムの参加者は，自らの資源を管理する経済的な主体であるため，このことが実際に適切に行われるためには，参加者のインセンティブに注目する必要がある．システムの設計者は，たとえば何らかの形で保証された価値を代表する交換媒体をデザインしなければならない．そのような媒体がなければ，P2P システムの参加者は，資源を提供するだけ損をしてしまい，誰も資源を提供しなくなってしまう可能性があるためである．そのような媒体のデザインを含め，P2P システムにおいては，インセンティブ整合的な交換のルール，すなわち，利己的な行動が集まった結果，協調的な交換が起こるようなデザインが必要となる．

さらに，P2P システムにおける交換媒体のデザインでは，その媒体そのものが P2P であるように注意しなければならない．さもなければ，P2P であることによる，可塑性や自発性といった，システムを維持する上で有効な性質が失われかねないためである．

日本では地域通貨という呼称が浸透している補完通貨は，法定通貨を代替したり，補完するメディアとして誕生した，地域で自律的に発行される交換媒体である．我々は，元々のデザインが特に自律分散的であり，P2P の性質を備えている補完通貨である「ワットシステム」に注目し，P2P システムを始めとするネットワーク上での交換アクティビティに利用すべく電子化した，*i*-WAT の開発を進めている．この試みは 2003 年に開始され，2004 年には実用システムとして運用が可能になった．現在は，さまざまな改良を加えると同時に利用の促進を行っている．

ワットシステムは多中心的な通貨システムで，ワット

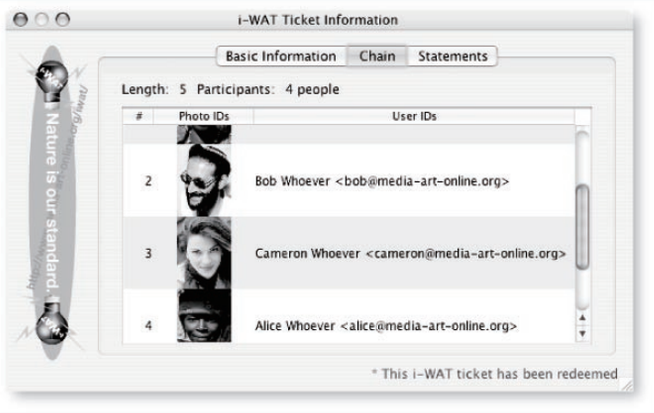


図-5　*i*-WAT券の視覚表現

券と呼ばれる交換媒体を用いる．ワット券は，手形のようなものであるが，清算のための場所や時間を指定しない．*i*-WAT では，この券を PGP 形式に基づく電子署名を利用することで電子的に表現している．

ワットシステム

i-WAT の詳細を述べる前に，ワットシステムについてもう少し詳しく紹介しよう．

ワットシステムは，地域通貨の研究会であるゲゼル研究会主宰の森野栄一氏により 2000 年に開発されたシステムである．

ワットシステムでは，以下の手順により取引を行う．

1. 振出取引ーワット券の誕生
振出人は白紙のワット券に財やサービスの提供元（貸付人）の名前と貸し付けられた負債の額面☆1，日付，および振出人の署名を記載する．これを貸付人に渡すことにより，財やサービスの提供を受ける．

2. 通常取引ーワット券の流通
ワット券を所持する者は，その券を別の取引に用いることができる．そのためには，券の裏に自分と受取人の名前を書く．受取人はワット券の新たな使用者となることができるが，これを繰り返すことによりワット券は人々の間を巡る．

3. 清算取引ー ワット券の帰還
取引の結果，ワット券が振出人の元に戻ると清算が起き，そのワット券は無効となる．

☆1 典型的には単位は kWh である．これは自然エネルギーによる発電のコストを代表している．

ワットシステムには次の利点がある．

自律性　筆記用具，および上記のプロトコルを遵守する意思さえあれば，誰でも自発的にワットシステムに参加できる．

互換性　ワット券は他のどのワット券とも互換である．振出人が信用される範囲において，世界のどこでも利用できる．

拡張性　上記のプロトコルはワットコアと呼ばれる中核部分を定義したものであり，拡張部として以下を含む条項を追加できる : 使用可能な地域，グループ，期間，負債の単位など．

安全性　振出人が清算に失敗すると，貸付人が負債を肩代りする．貸付人が肩代りできない場合は，その次の裏書人が肩代りする．
裏書のチェインが長いほど，その券は信用を獲得していると言える．

i-WAT：インターネット・ワット

i-WAT はワットコアをインターネット上のプロトコルとして移植したものである．

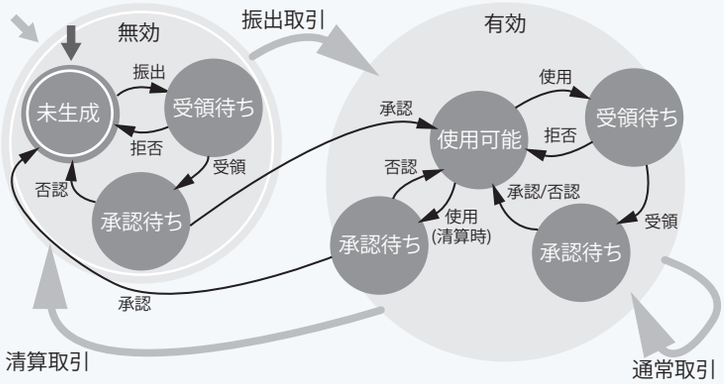
i-WAT では，PGP 形式で署名されたメッセージを用いてワット券を電子的に表現しているが，この表現形を *i*-WAT 券と呼ぶ．*i*-WAT 券のデータは XML で表現され，現在はその転送に XMPP（Extensible Messaging and Presence Protocol）を利用している．

i-WAT 券は振出人によって一意に決められる ID 番号と，負債額，および振出人，使用者，受取人といった参加者の公開鍵ユーザ ID を記載している．裏書は，PGP 署名をネストさせる（署名付きの XML データにさらに署名する）ことにより実現している．リファレンス実装では，視覚表現上，PGP のフォト ID を利用している（図 -5）．

図 -6 はワット券および *i*-WAT 券の状態遷移を表したものである．

ワットコアをディジタルネットワーク上のプロトコルとして実現するにあたり，次の問題を解決する必要があった．

1. 取引は非同期に行われなければならない．
そのため，受領待ちや承認待ちといった中間状態を用



＊グレイの矢印はワット券の状態遷移を表す
＊黒の矢印は *i*-WAT 券の状態遷移を表す

図-6　ワット／*i*-WAT券の状態遷移

意する必要があった．

2. 2 重使用を防止しなければならない．

振出人が，流通する券が不正に複製されたものではないことを保証する責任を負うという設計にした．*i*-WAT 券が不正に複製されることにより被害を受けるのは振出人であるため，この設計はインセンティブ整合的である．

3. 識別子のコストと可用性の間のトレードオフを考慮しなければならない．

参加者の識別子として，公開鍵ユーザ ID（現在の PGP の運用ではメールアドレス）を採用した．この識別子は，参加者自身が自律的に付与できるが，信用の輪（web of trust）の形成により，周囲がその正当性を確認していくという仕組みを持っている．

i-WAT の耐故障性

分散システムの特徴に，メッセージの遅延と欠落を本質的には区別できないという問題がある．そのため，メッセージが再送されて多重に受信側に届く可能性があるが，分散システムのデザインにおいては，このことが起きてもシステムとして影響を受けない，冪等性（idempotence）という概念が発達した．

電子的な通貨システムにおいて問題となる 2 重使用とその回避も，同一なメッセージが多重に届くことと，そのことへの耐性であることから，本質的には冪等性の問題であると言える．

i-WAT においては，メッセージが欠落するという障害と，貨幣が 2 重使用されるというセキュリティ上の問題に対して，振出人によるチェックという，統一的な解を与えることができたと考えている．

i-WAT は PGP の利用を基礎としているが，利用者が秘密鍵を紛失するという事故が，実用においても，またテスト中も何度か起きた．*i*-WAT での識別子は鍵の ID（ハッシュ値）ではなく，あくまで公開鍵ユーザ ID であるため，この種の事故に対しては，同一のユーザ ID で鍵ペアを再生成し，信用の輪を再構築することで対処できている．

また，通常取引において，*i*-WAT 券のデータが 3 カ所（振出人，使用者，受取人）に複製されることを利用し，実用において，振出人の持つ *i*-WAT 券のデータベースが紛失した際にも，データを復元できることを確認している．

i-WAT の入手方法

i-WAT は，同じく WIDE プロジェクト IDEON ワーキンググループにより開発されている，*wija* と呼ばれる XMPP クライアントにプラグインとして同梱されている．*wija* は以下の URL より入手できる．

```
http://www.media-art-online.org/wija/
```

広域統合分散環境の実現に向けて

本稿では，WIDE プロジェクト IDEON ワーキンググループの活動の中から，分散ゲーム環境構築を目的としたインフラ構築ライブラリである libcookai と P2P 補完通貨である *i*-WAT について紹介した．

これらの技術は，広域統合分散環境を実現するための要素技術として，今後さらに発展させていくことができると期待している．

（平成 17 年 6 月 21 日受付）