Overlay Weaver の ALM 機能拡張と P2P 型コミュニケーションシステム Uenew への応用

坂野 遼平　佐藤 晴彦　小山 聡　栗原 正仁

オーバーレイネットワーク構築用ミドルウェアである Overlay Weaver には，Scribe アルゴリズムに基づくアプリケーション層マルチキャストの機能が実装されている．しかし Scribe ではマルチキャストグループ毎のアーカイブを取得する方法が確立されておらず，Overlay Weaver にもアーカイブの取得機能は実装されていないため，応用が狭められているという問題があった．そこで我々は，Scribe におけるアーカイブの取得手法を新たに提案し，Overlay Weaver の機能を拡張する形で実装した．実装した提案手法についてはシミュレーションによる動作実験を実施し，10 万ノードでの稼働を確認した．また本稿では，提案手法の実応用である P2P 型のウェブ閲覧者間コミュニケーションシステム Uenew についても言及する．

Overlay Weaver, a practical middleware for constructing overlay networks, implements functionalities for the application layer multicast (ALM) based on the Scribe algorithm of M. Castro and P. Druschel. However, there has been a problem of restricted applicability in Scribe and Overlay Weaver resulted from the non-existence of an established method to acquire archives for each multicast group. In this paper, we resolve this problem by presenting a new method for acquiring archives and describing its implementation on top of Overlay Weaver. We report on its practical performance based on simulation for a large-scale network. Finally, we show how the new functionality was successfully exploited in Uenew, a P2P-network application for the dynamic, archive-based communication among Web users.

1　はじめに

近年，インターネット等を利用した様々な大規模サービスの展開が為されている．それらの多くはクライアント/サーバ型のシステムとして提供されているが，インターネット利用者数の増加に加え，動画配信やオンラインストレージといった帯域の負荷が大きいサービスに対する需要が高まりつつあることで，サーバへの負荷集中によるサービス停止や品質低下が問題視されるようになってきた．

そのような中で，スケーラビリティや耐障害性に優れる P2P 型システムに注目が集まっている．本稿では，P2P 技術のひとつであるアプリケーション層マルチキャスト (Application Layer Multicast, 以下 ALM とする) に焦点を当てる．

ALM はアプリケーション層でパケットの複製を行い，1 対多通信を実現する技術である．即ち，1 つの発信ノードが全ての受信ノードにデータを送るのではなく，経路上の各ノードが転送を行うことで情報を伝播するため，負荷を分散させることができる．ALM は，IP マルチキャスト [3] と比べ，既存のインフラに対してハードウェア的な対応を要求せず，柔軟な設計が可能であるという利点を持つ．

本稿では，分散ハッシュテーブル [9][12] をベースとしたアルゴリズム Scribe [1] を扱う．Scribe を用いることで，インターネット等を下位層としたオーバーレイネットワーク上にグループを任意に生成し，メンバ同士のマルチキャストを実現可能である．

しかし Castro らの研究 [1] では，グループに新た

An Extension of Overlay Weaver with Application to a Cross-Browsing Communication System Based on Peer-to-Peer Networks.

Ryohei Banno, Haruhiko Sato, Satoshi Oyama, Masahito Kurihara, 北海道大学大学院情報科学研究科, Graduate School of Information Science and Technology, Hokkaido University.

コンピュータソフトウェア, Vol.29, No.2 (2012), pp.123–139.
[ソフトウェア論文] 2010 年 11 月 12 日受付.

に参加するノードが当該グループ内で過去に配信された情報を取得する術が示されておらず，アプリケーション開発者が実装上の工夫により実現する必要があった．そこで本研究では，Scribe について，同一グループ内で配信された情報の小規模データベースである「アーカイブ」の取得手法を新たに提案し，Peer-to-Peer ネットワーク構築用ミドルウェアである Overlay Weaver [11] の機能拡張という形で実装することで，アプリケーション開発に有用なライブラリを提供することを目的とする．

さらに，提案手法の実際の応用事例として，我々は P2P 型のウェブ閲覧者間コミュニケーションシステム Uenew を開発している．Uenew は，ウェブ上で同じ場所 (ウェブページ) に居るユーザ同士が自由にコミュニケーションをとることを可能とするシステムである．本稿では，このシステムについても言及する．

本論文の構成は，以下の通りである．まず，2 章で本研究で用いる Peer-to-Peer 技術について述べ，次に 3 章で Scribe におけるグループ毎のアーカイブ取得手法を提案する．提案手法は Overlay Weaver 上に実装し，4 章において効率等に関する実験と考察を示す．続く 5 章では提案手法の実応用であるコミュニケーションシステム Uenew のコンセプトや実装について解説し，最後に 6 章で本論文の総括を行う．

2 Peer-to-Peer ネットワーク技術

2.1 分散ハッシュテーブル

分散ハッシュテーブル (Distributed Hash Table, 以下 DHT とする) [9][12] は，ハッシュテーブル (連想配列) をネットワーク上の複数ノードにまたがって構築する技術である．ハッシュテーブルにおけるキーと格納値の組はエントリと呼ばれ，(key, value) のように表される．ハッシュテーブルでは key のハッシュ値に基づいてデータを管理するが，DHT の場合，ネットワークに参加する各ノードにも固有のハッシュ値を割り当て[†1]，巨大なハッシュ空間上に key とノードをマッピングする．そしてこのハッシュ空間上での配置に基づいて，ハッシュテーブルを各ノードが分担管理する．

従って DHT を実現するには，以下のような仕組みが必要となる．

- ハッシュテーブルを分割する仕組み
- key で検索した際，当該 key を担当するノードに到達できる仕組み

前者については，「ある key はその key のハッシュ値と数値的に最も近いハッシュ値を持つノードが担当する」「あるノードの担当範囲はハッシュ空間上における自身の Successor(あるいは Predecessor) ノードとの間に含まれるハッシュ値とする」等の規則が用いられることが多い．

後者はルーティング問題と呼ばれ，効率的な探索を実現するためにはこの仕組みをどのように設計するかが重要であり，DHT のアルゴリズムによって様々な手法が提案されている．一般的には，各ノードが，オーバーレイネットワークの距離空間上で自身に近いノードについては密な，遠いノードについては疎な情報を経路表に持つことで，経路表のサイズと検索時のホップ数を共に小さく抑えるようになっている．多くのアルゴリズムでは，ノード数を n とした時，次数 (経路表の大きさ) と直径 (最大所要ホップ数) が共に $O(\log n)$ であるようなグラフを構築可能である．

2.2 アプリケーション層マルチキャスト

アプリケーション層マルチキャスト (Application Layer Multicast, 以下 ALM とする) は，IP マルチキャストの動作をオーバーレイネットワーク上で実現する技術である．パケットの複製をアプリケーション層で行うため既存のインフラに対してハードウェア的な対応を要求せず，柔軟な設計が可能である．ここでは，2.1 節で述べた DHT の一種 Pastry [9] をベースとした ALM 手法である Scribe [1] について述べる．

Pastry を含む DHT では，同じ key で検索をすれば，どのノードからであっても同一のノードにたどりつく．即ち，ある key を複数のノードから検索すると，それらの検索時のルーティング経路の集合が図 1 に示すような木構造となる．図中で，実線の矢印はルーティング経路を表している．Scribe では，これを配送木として利用する．

†1 ノードの IP アドレス等をハッシュ関数に入力した結果を用いる．

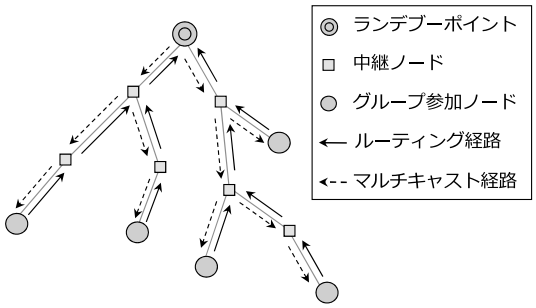


図 1 Scribe におけるマルチキャスト木の構成

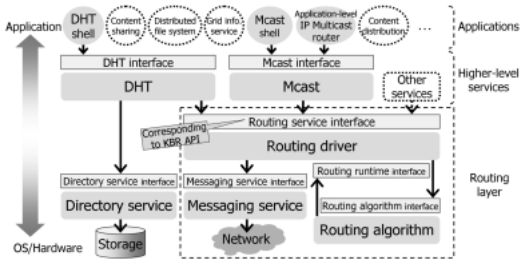


図 2 Overlay Weaver の構成
(論文 [11] の図 2 より転載)

配送木の根にあたるノードはランデブーポイント (Rendezvous Point) と呼ばれ，ランデブーポイントに渡されたデータが順に子ノードに転送されていくことでマルチキャストが実現される．つまり，配送木上の各ノード (Forwarder と呼ばれる) が IP マルチキャストにおけるルータのように振舞うのである．図 1 では，破線の矢印がマルチキャスト時の転送経路を示している．グループに参加したいノードは，グループ名を key としてルーティングを行うことで配送木に連なることができる．

このように，Scribe では Pastry のリンク関係をそのまま用いて配送木を構築するため，Forwarder にはグループに参加しているノードと，参加しておらず中継のみ行うノードの 2 種類が存在することになる．

なお，Scribe の配送木構築手法は Pastry 以外の DHT アルゴリズムにも適用可能である．実際，2.3 節で述べる Overlay Weaver では Scribe と同様の仕組みによる ALM 機能を実装しているが，ベースとなる DHT のアルゴリズムは自由に切り替えられるようになっている．

2.3 Overlay Weaver

Overlay Weaver [11] は，key-based routing [2] の概念に基づき Java 言語で実装されたミドルウェアである．

Overlay Weaver を構成する階層は，図 2 に示したように，下位側からルーティング層，ハイレベルサービス層，アプリケーション層となっており，ルーティング層において各種アルゴリズムに共通する処理を括り出す事で，各アルゴリズムの実装に個別に必要とされるプログラムコード量の大幅な削減を実現している．このため，アルゴリズム研究者にとっては，このミドルウェアへのアルゴリズムの追加が非常に容易であり，また他のアルゴリズムとの公正な比較が可能であるという利点がある．標準では Chord [12]，Kademlia [7]，Koorde [6]，Pastry [9]，Tapestry [14] の各 DHT アルゴリズムが実装されており，アプリケーション開発者はこれらの中から利用するアルゴリズムを簡単に差し替えることができる．

Overlay Weaver のハイレベルサービス層では，DHT と Mcast の 2 種類のインタフェースが提供されている．Mcast は ALM の機能を提供しており，アルゴリズムとして 2.2 節で述べた Scribe が採用されている．

3 Scribe におけるアーカイブ取得手法の提案

3.1 Overlay Weaver に標準で実装されている ALM

まず，Overlay Weaver に標準で実装されているハイレベルサービス Mcast について述べる．Mcast は，Scribe に基づく ALM 機能を提供する．Mcast におけるグループへの参加処理を，図 3 に示す．

Mcast のグループ参加処理では，まずグループに参加するノードが，グループ名を key とした DHT の探索経路沿いに Join 要求を転送する．Join 要求を転送されたノードでは，当該グループに関する経路表を作成し，ランデブーポイントへ向かって Join 要求をさらに転送していく．但し，Join 要求を受け取ったノードが当該グループに関する経路表を既に持っている場合，そのノードからランデブーポイントまでの経

新規参加ノード

```
グループ参加( グループ名 : gn ) {
    参加済グループリスト.add( グループ名 : gn )
    if ( gnの経路表が存在 ){
        return
    } else {
        dest = DHT経路表.getNextHop( キー : gn )
        参加要求送信( 宛先 : dest )
        gnの経路表を作成( 親 : dest, 子 : null )
    }
}
```

各ノード

```
メッセージ受信( グループ名 : gn, 転送元 : src ) {
    if ( gnの経路表が存在 ) {
        gnの経路表.add( 子 : src )
    } else {
        dest = DHT経路表.getNextHop( キー : gn )
        参加要求転送( 宛先 : dest )
        gnの経路表を作成( 親 : dest, 子 : src )
    }
}
```

図 3　Mcast におけるグループ参加処理

マルチキャスト発信ノード

```
マルチキャスト( グループ名 : gn, データ : d ) {
    parent = gnの経路表.getParent()
    children = gnの経路表.getChildren()
    マルチキャスト送信( 宛先 : parent, データ : d )
    for each( child in children ) {
        マルチキャスト送信( 宛先 : child, データ : d )
    }
}
```

各ノード

```
メッセージ受信( グループ名 : gn, 転送元 : src ) {
    parent = gnの経路表.getParent()
    children = gnの経路表.getChildren()
    if( parent ≠ src ∧ parent ≠ null) {
        マルチキャスト転送( 宛先 : parent, データ : d )
    }
    for each( child in children ) {
        if( child ≠ src ) {
            マルチキャスト転送( 宛先 : child, データ : d )
        }
    }
}
```

図 4　Mcast におけるマルチキャスト処理

路上ノードには既に当該グループの経路表が作成されているため，Join 要求はそこで破棄される．

一方，図 4 はマルチキャスト処理を示したものである．この Mcast におけるマルチキャストの配送経路は，Scribe の原著論文における仕様とは異なっている．図 5 において，実線の矢印は Mcast での配送経路，破線の矢印は Scribe での配送経路である．Scribe ではまずランデブーポイントへデータを渡し，そこから子ノードへ向かって転送していくが，Mcast ではマルチキャスト発信ノードから親ノード側と子ノード側の双方向に伝播していく仕様となっている．

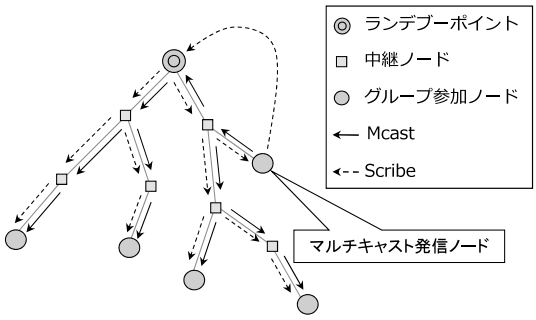


図 5　Mcast と Scribe におけるマルチキャスト経路の違い

3.2　提案手法

3.2.1　クエリのマルチキャストによるアーカイブ取得手法

概要

Mcast をアーカイブ取得可能なように拡張する場合，最も単純な方法はアーカイブ取得要求をマルチキャストすることである．即ち，グループ参加ノードにはアーカイブの保持を義務付けることとし，グループ参加時に他の参加ノードからアーカイブを受信するようにすればよい．この方式を，以降 QMAM(Query Multicasting Archivable Multicast) と呼ぶ．QMAM では，3.1 節で述べた Mcast のグループ参加と同様の処理を行った後に，図 6 に示した手続きでアーカイブ取得を行う．

図 7 において，実線の矢印はアーカイブ取得要求のマルチキャスト経路，破線の矢印はそれに対する応答を表す．アーカイブを要求したノードは，最初に応答したノードからアーカイブを取得する．つまり，要求元ノードでは最初に届いた応答に対してのみ確認応答を返し，確認応答を受けたノードはアーカイブを送信するという仕組みである．

また，グループに参加している各ノードは，マルチキャストを受信した際にローカルのアーカイブを更新するようにする．従って，QMAM ではグループに参

アーカイブ要求ノード

```
アーカイブ要求( グループ名 : gn ) {
    parent = gnの経路表.getParent()
    children = gnの経路表.getChildren()
    waiting = TRUE
    アーカイブ要求送信( 宛先 : parent )
    for each( child in children ) {
        アーカイブ要求送信( 宛先 : child )
    }
}

応答受信( 応答元 : responser ) {
    if( waiting = TRUE ){
        waiting = FALSE
        確認応答送信( 宛先 : responser )
    }
}
```

各ノード

```
メッセージ受信( グループ名 : gn, 要求元 : requester, 転送元 : src ) {
    if( 参加済グループリスト.contain( グループ名 : gn ) ) {
        応答送信( 宛先 : requester )
    } else {
        parent = gnの経路表.getParent()
        children = gnの経路表.getChildren()
        if( parent ≠ src ∧ parent ≠ null) {
            アーカイブ要求転送( 宛先 : parent )
        }
        for each( child in children ) {
            if( child ≠ src ) {
                アーカイブ要求転送( 宛先 : child )
            }
        }
    }
}

確認応答受信( 送信元 : src ) {
    アーカイブ送信( 宛先 : src )
}
```

図 6 QMAM におけるアーカイブ取得処理

加する全てのノードが最新のアーカイブを保持することになる．

QMAM の課題点

Mcast のグループ参加処理やマルチキャスト受信処理に，アーカイブ要求に関する処理を追加するだけで良いため，QMAM の実装は比較的容易である．しかしながら，QMAM ではグループへの参加ノード数が 0 となった時にネットワーク上からアーカイブが消えてしまい，アーカイブを取得できなくなるという問題がある．

また，QMAM 及びそのベースとなっている Mcast には他にも課題となる点がある．まず，各ノードにおけるメッセージの到着順序が一致することを保証できない点である．例えば図 7 においてノード A と E

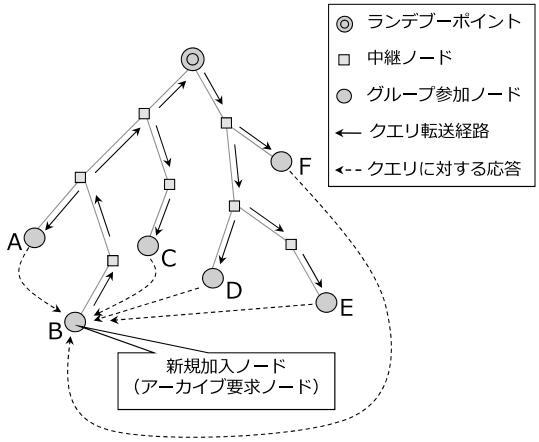


図 7 QMAM におけるアーカイブ取得経路

がほぼ同時にそれぞれメッセージ M_A，M_E を発した際，ノード C とノード D とでは M_A，M_E の到着順が異なる可能性がある．

もう 1 つの問題は，churn やネットワーク障害によりメッセージは欠落し得るため，あるメッセージを一部のノードは受信し，他のノードは受信しないといった状況が生じる可能性がある点である．本稿では，グループに参加するノードが同一のアーカイブデータを保有可能な性質を整合性 (consistency) と呼ぶ．

QMAM ではグループ参加ノードのいずれか 1 つからアーカイブを取得するため，整合性の欠如が発生している場合，新たに参加するノードが不完全なアーカイブを取得する可能性がある．特に，順序も含めた整合性が必要とされるアプリケーションにおいては，前述の順序不一致の性質も相まって，大きな問題となる．従って，整合性の欠如が発生した場合にそれが解消されるような仕組みが求められる．具体的には，ノード間で順序を一致させる工夫を施した上で，メッセージの欠落を補完する必要がある．

まず，順序の不一致への対策としては，メッセージ発信時にタイムスタンプを付与することで，各ノードにおいて受信メッセージを同一の順序に並び替え可能とすることが挙げられる．しかし各ノードの時計は一般に一致していないため，例えば 5 章に述べるような応用では不自然な会話順となってしまう等の問題が考えられる．この問題はタイムスタンプ付与時に毎回

共通の時刻サーバから時刻を取得することで回避できるが，そのようなサーバの利用を前提とすることはP2P システムの利点を損ねる要因となる．

また，メッセージ欠落に対する方策としては，各ノードが定期的に整合性確認メッセージを流すことが考えられる．その際，各ノードが持つアーカイブのハッシュ値を算出し比較すれば不一致の検出は可能であるが，不一致である 2 ノードのアーカイブデータのうちどちらが正しいのかは判断できず，互い違いに欠落しているメッセージがある可能性もある．そこで，メッセージを発信する際，あらかじめ固有のメッセージ ID を割り当てておけば，その ID 集合を定期的にマルチキャストすることで，欠落を相互発見し補い合うことが可能となる．あるいは，アーカイブ取得処理の発生時に，全ての既参加ノードが要求元ノードへメッセージ ID の集合を送るようにすれば，要求元ノードで整合性をチェックすることができ，グループへの新規参加が発生する都度確認が行われることになる．しかしいずれにしても，グループ参加者数や保存する過去メッセージ数が多い場合，整合性維持のための通信のコストが嵩んでしまうという問題がある．

3.2.2 DHT とのハイブリッドによるアーカイブ取得手法

概要

QMAM におけるアーカイブ消失の問題を解決するためには，ネットワーク上にアーカイブを保存する特定のスペースが必要となる．そこで，DHT とのハイブリッドによる手法 DBAM(DHT-Based Archivable Multicast) を提案する．

DBAM では，アーカイブの蓄積と取得を DHT により行う．ALM とは別に DHT 用のオーバーレイネットワークを用意するわけではなく，ALM の下位ネットワークとして構築されている DHT をそのまま用いる．

Scribe のアルゴリズムより，各グループのランデブーポイントは DHT においてグループ名を key として put/get(登録/取得) する際にデータが格納されるノードである．従って，ランデブーポイントを各グループのアーカイブが集積されるノードとすればよい．

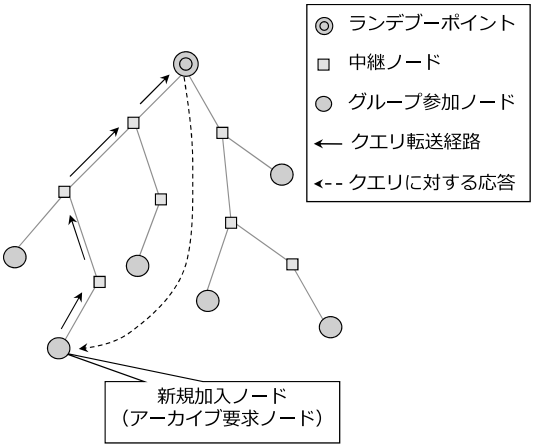


図 8　DBAM におけるアーカイブ取得経路

Overlay Weaver の標準の実装では，ハイレベルサービス層の DHT と Mcast は単体で用いることが想定されており，同一オーバーレイネットワーク上で共存させることはできない．DBAM は，DHT と Mcast の機能を統合し新しいハイレベルサービスを構築したものである．

DBAM の動作について，要点のみをまとめると以下のようになる．

- グループ参加
  - 通常のグループ参加処理を行う．
  - ランデブーポイントからアーカイブを get する．
- マルチキャスト
  - ランデブーポイントへアーカイブを put する．
  - ランデブーポイントが，put されたアーカイブを子ノードへ向けてマルチキャストする．

図 8 に，DBAM におけるアーカイブの取得経路を示す．DHT のルーティングにより実線の矢印に沿ってランデブーポイントまで要求が転送され，ランデブーポイントからは破線の矢印に示したように要求元ノードへアーカイブが送信されることになる．

DBAM では，ランデブーポイントに蓄積されているアーカイブデータの churn 等による消失が発生していないならば，グループに参加しているノード数が 0 であってもアーカイブを取得することができる．ま

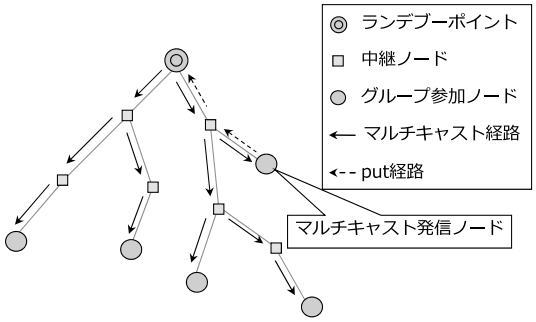


図 9 DBAM におけるマルチキャスト経路

た，QMAM とは異なりグループ参加ノードがアーカイブを持つことを義務付ける必要は無い．

一方，図 9 は DBAM におけるマルチキャスト経路を示したものである．DBAM では必ずランデブーポイントを経由してマルチキャストが行われるため，図 5 に示した Scribe のマルチキャストに似た振る舞いとなる．

DBAM におけるノード間の整合性と耐障害策

DBAM では，QMAM とは異なり，グループ参加ノードが全メッセージを受け取るならば，各ノードが受信するメッセージの順序は一致する．これは，全てのマルチキャストメッセージがランデブーポイントを経由するためである．

但し，時刻 t_1 が t_2 よりも先であるとして，t_1 にノード A から put されたものより t_2 にノード B から put されたもののほうが先にランデブーポイントに到着するといった状況は発生する可能性がある．従って，ほぼ同時刻に発生したマルチキャスト (ランデブーポイントへの put) に関する，実時刻に照らした順序関係が大きな意味をもつ用途においては，3. 2. 1 項に述べたような共通の時刻サーバの利用等の方策が必要となる．

しかし実際の応用においては，より意味的な順序が重要であることが多い．例えば 5 章のような応用を考えた場合，順序に関する重要事項は対話順が保たれること，及びノード間で順序が一致することである．後者が欠ける場合，ある一連のメッセージに起因するメッセージをやりとりする際に矛盾が生じる可能性がある．例えば「コップの中身は水だ」「コップの中身は酒だ」という 2 つのメッセージがマルチキャストされた後に,「正解は前者である」というメッセージが流れた場合，最初の 2 つのメッセージの順序が全ノードにおいて一致していなければ意味を正しく解釈できなくなってしまう．

QMAM では順序の一致保証が難しいためこういった問題が生じ得るが，DBAM では回避可能である．先に述べた 2 つの重要事項のうち，前者については，ある発言に対する返信が元の発言より先にランデブーポイントに到達することは起こり得ないため問題とはならず，また後者についても，前述のとおり DBAM では順序を一致させることが可能である．

一方，メッセージの欠落によるノード間の整合性欠如については，QMAM と同様 DBAM でも考慮する必要がある．しかし DBAM では，ランデブーポイントからの送出時にメッセージに番号を付与することで，各ノードは連続するメッセージ群の部分的欠落については容易に検知できる．従ってメッセージ ID の集合をやり取りする必要はなく，検知された欠落についてはランデブーポイントから再取得することができる．さらに，アーカイブデータの取得先がランデブーポイントに一元化されているため，QMAM とは異なり他の全参加ノードと照らし合わせる必要がなく，アーカイブデータの整合性欠如による影響が新規参加ノードに伝播してしまう状況も発生しない．

但し，DBAM ではアーカイブデータの churn 耐性について考慮が必要となる．これは，churn によるランデブーポイントの離脱や入れ替わりが発生した場合，当該グループのアーカイブデータが消失してしまう可能性があるためである．DBAM では，アーカイブの取得/蓄積に DHT の機構をそのまま用いているため，アーカイブの取得可能性は DHT での get 可能性と同義である．従って，耐 churn 手法に関する既存研究 [10] の成果を適用することができる．

この既存研究で挙げられている耐 churn 手法は，以下の 4 種類である．

- 複製配置
  DHT 上にデータを put する際，当該 key の担当ノード以外のノードに複製を配置する．
- 加入時委譲

オーバーレイネットワークに新たにノードが参加した際，必要に応じて周辺ノードから (key, value) ペアを譲り受ける．

- 複数 get
  DHT 上からデータを get する際，当該 key の担当ノードのみではなく複数のノードに問い合わせを行う．
- 自動再 put
  各ノードが，保持している (key, value) ペアを自動的に put し直す．

例えば，あるグループにおいてノード root0 がランデブーポイントとして定まり，ノード root1 が次点候補であったとすると，ランデブーポイントが消失した場合，以下の流れによりネットワーク上にアーカイブが維持される．

1. root0 へのアーカイブ蓄積時，複製配置により root1 に複製が保持される．
2. root0 が消失する．
3. GroupRefresher[†2]により，配送木が root1 をランデブーポイントとして再構築される．
4. root1 からアーカイブが取得される．

また，root0 を次点候補に押しのけてランデブーポイントとなる rootX が新たにオーバーレイ上に参加してきた場合，以下のようにしてアーカイブを取得可能な環境が継続する．

1. rootX が参加する．
2. 加入時委譲あるいは自動再 put により，rootX へアーカイブが渡される．
3. GroupRefresher により，配送木が rootX をランデブーポイントとして再構築される．
4. rootX からアーカイブが取得される．あるいは，複数 get により root0 からアーカイブが取得される．

以上のように，アーカイブデータに関しては複製配置等の手法で churn 耐性を高めることができる．

3.3 提案手法の実装

3.2 節で述べた QMAM 及び DBAM を，図 2 のハイレベルサービス層 (Higher-level services) に実装した．

まず QMAM と DBAM に共通する仕様について述べる．グループに参加するノードはローカルにアーカイブを保持しており，マルチキャストを受信するとローカルのアーカイブを更新する．また，グループ離脱時にローカルに保持している当該グループのアーカイブを削除する．

アーカイブの取得はグループ参加時に自動的に行われるが，GroupRefresher によるグループ参加処理ではアーカイブ取得は行わない．

アーカイブデータについては，マルチキャストでやり取りされるデータを保持するラッパークラスを用意し，データ本体とともに番号，ユーザ，日時が記録されるようになっている．「ユーザ」はオーバーレイ上での ID(ノードに割り当てられたハッシュ値) によって記録する．「日時」はマルチキャスト発信元ノードにおいて発信をした日時であり，「番号」は QMAM では各ノードへの到着順，DBAM ではランデブーポイントへの到着順に付与される．

以下は，DBAM のみの仕様に関する記述である．ランデブーポイントに集積されるアーカイブは，グループ毎に最新の N 件を保持するようになっている．N を超えると，古いものから順に破棄される．また，アーカイブの件数が N を上回らなくても，配信から一定期間 T が経過したアーカイブは消えるようになっている．これらのパラメータ N, T は容易に変更が可能である．

さらに，DBAM ではアーカイブデータの削除機能を実装している．元々Overlay Weaver の DHT にはユーザ毎のパスワード (secret) を設定することで put したデータを削除できる機能があり，それを拡張する形で実装した．アーカイブデータの削除は，マルチキャストしたユーザのみ行うことができる．削除要求を行うと，ランデブーポイント上の当該データが削除されると共に，要求がマルチキャストされ，グループ参加各ノードが持つローカルのアーカイブからも当該データが削除される．

なお，Overlay Weaver では耐 churn 手法を実装した DHT(Churn Tolerant DHT) が提供されてい

†2 Mcast に実装されている，配送木修復のために定期的にグループへの再参加を行う機能.

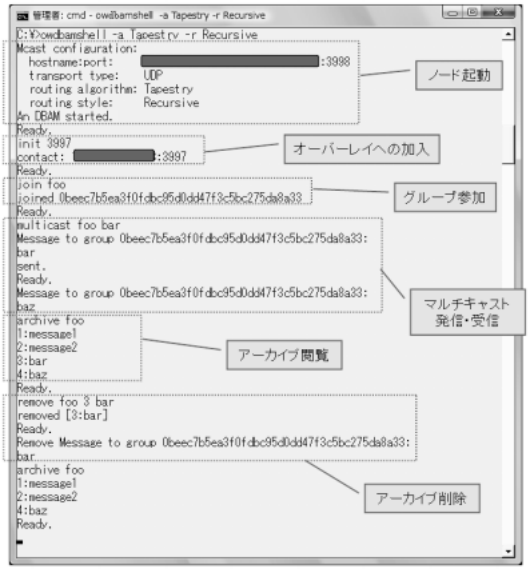


図 10 DBAM shell の動作図

る．そこで，まず Churn Tolerant DHT を拡張した DBAM 専用の DHT 実装を用意し，その DHT の上に DBAM の実装を行うことで，3.2.2 項で述べた耐 churn 手法を実際に用いることができるようになっている．

3.4 サンプルプログラムの実装

Overlay Weaver では，2 つのハイレベルサービスについてそれぞれ DHT shell 及び Mcast shell と呼ばれるプログラムを提供している．これらは，ハイレベルサービスを用いたアプリケーションの実装方法を示したサンプルであり，またこれらを Overlay Weaver 付属のエミュレータと組み合わせることで，ルーティングアルゴリズムの動作試験や比較を容易に行うことができる．

今回，ハイレベルサービスの追加にあたり，上記の DHT shell，Mcast shell に倣って QMAM shell 及び DBAM shell を実装した． 図 10 はコマンドプロンプトで DBAM shell を起動し，アーカイブ取得等の操作を実行した様子である．このように，QMAM shell や DBAM shell を用いることで，アーカイブ取得可能な ALM のネットワークを容易に立ち上げることができ，コマンドラインから操作することが可能である．

4 提案手法の考察と動作実験

4.1 概要

提案手法の特徴と，多ノード環境において実際に動作することを確かめるために，シミュレーション実験を行った．実験には Overlay Weaver に付属するエミュレータを用い，3.4 節で述べたコマンドベースのサンプルプログラムを同一計算機上で複数起動させて実施した．

実験では，トランスポートプロトコルとして UDP を用いた．ルーティング様式については，提案手法が 5 章のような応用を視野に入れていることから，安全性において Recursive より優れている [15]Iterative を設定した．

また，ルーティングアルゴリズムとしては，Pastry を用いた．本研究は Overlay Weaver の機能拡張であると同時に Scribe のアルゴリズム拡張でもあり，Scribe の原著論文では Pastry がベースとなっている．加えて，事前実験により DBAM におけるノードあたりのメッセージ数をアルゴリズム間で比較した際に，Pastry が最も少ないメッセージ数を示す 1 つであったことから，ここでは Pastry を選択した．

なお，いずれの実験においても，5 回の試行を行い最良値と最悪値を切り捨てた残りの平均をとったものを結果として用いている．

4.2 性能の考察

ここでは，以下の各点について提案手法の性能を考察する．

- ネットワーク負荷
  アーカイブ取得時，及びマルチキャスト時に，オーバーレイ上で交わされるメッセージの通信量．
- アーカイブ取得遅延
  アーカイブ取得要求を出してから実際に取得できるまでに要するメッセージのホップ数．
- マルチキャスト遅延
  マルチキャストを送信してから各ノードに到達するまでのメッセージのホップ数．

いずれも具体的な通信量についてはルーティング様式

により違いが出てくる [15] が，以下の議論は再帰探索 (fast) を想定したものである．また，以下ではランデブーポイントまでの平均経路長を n，グループ参加ノード数を m とする．

まず，アーカイブ取得時のネットワーク負荷については，DBAM の方が QMAM と比べ効率が良い．QMAM でアーカイブ取得に際しメッセージの転送が発生する回数は，要求元ノードからランデブーポイントまでが n 回，ランデブーポイントからグループ参加各ノード (要求元を除く) までが $n(m-1)$ 回，グループ参加ノードから要求元への応答とそれに対する確認応答，及びアーカイブ送信に $(m-1)+1+1$ 回である．従って通信量は，アーカイブ送信メッセージのデータ量を $k_{archive}$，それ以外のメッセージのデータ量を k_{query} とすれば，

$$(n+n(m-1)+(m-1)+1)k_{query}+k_{archive} = m(n+1)k_{query}+k_{archive}$$

となる．

DBAM では，ランデブーポイントまでの n 回とアーカイブ送信の 1 回であるから通信量は，

$$n \cdot k_{query}+k_{archive}$$

となる．

但し，QMAM についてはランデブーポイントまでの経路上にグループ参加ノードが存在すればメッセージ数が少なくなる余地がある．この余地は全ノード数とグループ参加ノード数に依存する†3 が，本稿の議論ではオーバーレイネットワークへの参加ノード数に対するグループ参加ノード数の割合が十分に小さい場合を想定する．

一方，マルチキャスト時のネットワーク負荷については，DBAM の方がやや大きくなる．これは，QMAM のマルチキャストは Mcast と同様に発信ノードから親ノード側と子ノード側の双方向に伝播していくのに対し，DBAM では必ず一度ランデブーポイントを経由するためである．QMAM でマルチキャストに要するメッセージ数は

$$n+n(m-1)$$

であるが，DBAM では

$$n+nm$$

となる．

次に，アーカイブ取得遅延について考える．通信遅延やノードのスペックを一定とした理想的環境下では，アーカイブ取得に要する遅延はホップ数に比例する．ノード間でアーカイブ取得要求の転送に要する時間を t_{query}，アーカイブデータの転送に要する時間を $t_{archive}$ とした場合，QMAM では要求元からランデブーポイントまでに n ホップ，ランデブーポイントからアーカイブ保持ノードまでに n ホップを要するため，

$$2n \cdot t_{query}+t_{archive}$$

となる．DBAM ではランデブーポイントからアーカイブを取得するため，

$$n \cdot t_{query}+t_{archive}$$

である．従って，DBAM の方が QMAM と比べ遅延が小さいと言える．

最後に，マルチキャスト時の遅延に関しては QMAM，DBAM 共に

$$2n \cdot t_{archive}$$

である．

4.3 動作実験

4.2 節を踏まえた上で，Mcast 及び QMAM，DBAM を比較する実験を行った．これらのハイレベルサービスに対する主な操作は，アーカイブ取得を伴うグループ参加と，マルチキャスト及びグループ離脱である．そこで，実験のシナリオとして以下のようなものを用いた．

1. 10 万ノードを起動する．
2. 全ノードをオーバーレイに加入させる．(5 ミリ秒毎)
3. 全ノードをランダムに 1 万個のグループへ参加させる．(10 ミリ秒毎)

†3 Pastry の場合，あるノードがグループに参加する際のランデブーポイントまでのホップ数 h は $\lceil \log_{2^b} N \rceil$ 以下である (b は Pastry のパラメータ)．全ノード数を N，グループ既参加ノード数を M とすると，グループ既参加ノードが ID 空間上で一様に分布している場合，新規参加ノードからランデブーポイントまでの経路上にグループ既参加ノードが存在する確率は $1-{}_{N-(h-1)}C_M/{}_NC_M$ である．例えば $b=4$ とすると，$N=10000$，$M=100$ の場合はおよそ 3 パーセントとなる．

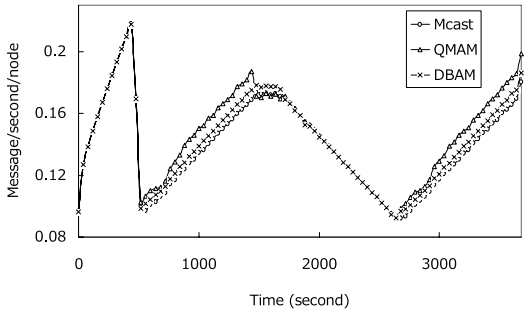


図 11 Mcast，QMAM，DBAM のメッセージ数比較

4. 1 万のノードから，参加したグループに対し 1 回ずつマルチキャストさせる．(20 ミリ秒毎)
5. 全ノードを参加したグループから離脱させる．(10 ミリ秒毎)
6. 全ノードをランダムに 1 万個のグループへ参加させる．(10 ミリ秒毎)

図 11 において，横軸は時間，縦軸は 1 ノード及び 1 秒あたりのメッセージ数を表している．メッセージ数は 40 秒間の平均値である．オーバーレイへの参加処理 (0 秒〜500 秒)，及びグループ離脱処理 (1700 秒〜2700 秒) については，3 方式に差は無いため，ほぼ同じメッセージ数となっている．

差が出ているのはグループ参加とマルチキャストを実行している箇所である．いずれも Mcast が最も少ないが，DBAM はわずかなメッセージ数の増加に抑えられている．QMAM はグループ参加処理においてメッセージ数が多くなっていることがわかる．

Mcast と QMAM，DBAM について，アーカイブ取得の可否と，グループ参加及びマルチキャストの処理における効率について相対的な優劣をまとめると，表 1 のようになる．

アーカイブの取得については，Mcast は取得できず，QMAM ではグループ参加ノードが途切れずに存在する場合のみ取得可能である．DBAM では，churn 等によるアーカイブデータの消失が発生していなければ，常に取得することができる．グループ参加とマルチキャストについては，Mcast の転送経路が最も無駄が少ないと言える．

表 1 Mcast, QMAM, DBAM の比較

	Mcast	QMAM	DBAM
アーカイブ取得可否	×	△	○
Join 効率	○	×	△
Multicast 効率	○	○	△

4.4 配送木維持機構の有効性

5 章で述べるようなシステムを考える場合，ネットワークを構成するノードは一般ユーザのコンピュータである．従ってノードの振る舞いは予測不可能であり，頻繁な離脱と加入 (churn) が常に起こり得るものとして考慮する必要がある．

Scribe [1] では，配送木上で子ノードを持つ全てのノードが，子ノードへ向けて定期的にメッセージを送ることでノードの生存確認を行うようになっており，親ノードの消失を検出した子ノードは再度グループ参加処理を行うようになっている．

Overlay Weaver の場合，グループに参加している各ノードが定期的にグループへの参加要求を送る GroupRefresher という仕組みを実装することで，上記のような配送木の自動修復を実現している．

Mcast の拡張である DBAM にも GroupRefresher は実装されているが，DBAM の場合マルチキャスト時の振る舞いが Mcast とは異なっているため，この配送木維持機構が有効に働いているかどうかを確認する必要がある．そこで，以下のシナリオにて実験を行った．

1. 10 万ノードを起動する．
2. 全ノードをオーバーレイに加入させる．(5 ミリ秒毎)
3. 全ノードを 1 万のグループからランダムにひとつ選んで参加させる．(10 ミリ秒毎)
4. 全ノードから，参加したグループに対し 1 回ずつマルチキャストさせる．(100 ミリ秒毎)

churn はマルチキャストを実行している間起こし続ける設定とした．churn の頻度は平均 10 回/秒，発生はポアソン分布に従うようにしてある．

上記のシナリオを，Mcast shell 及び DBAM shell を用いて実行し，マルチキャスト成功率の変化を観察した (図 12)．なお，マルチキャスト成功率とは，各

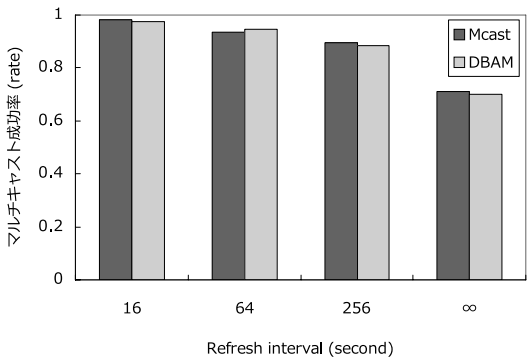


図 12　Mcast と DBAM のマルチキャスト成功率に関する比較

ノードがマルチキャストメッセージを受信すべき回数の和に対する，実際に受信した回数の和の比率である．

図 12 の横軸は，前述した Mcast の GroupRefresher 機能が働く時間間隔である．縦軸はマルチキャストの成功率を表している．Mcast と DBAM のいずれにおいても，GroupRefresher が働かない場合 (Refresh interval = ∞) には低下してしまう成功率が，Refresh interval を適切に設定することで向上している．この実験結果より，提案した DBAM では Mcast と同様に GroupRefresher が有効に働いていることがわかる．

5　ウェブ閲覧者間コミュニケーションシステム Uenew への応用

5.1　開発背景と Uenew の概要

実社会では，偶然同じイベントに参加していたり，同時に事故に遭遇するなど，ある場に居合わせて経験を共有した人同士の間で，自然にコミュニケーションが発生する場面が多々見受けられる．そのような場での情報交換によって新たな人間関係が作り上げられていくことも少なくないが，ウェブ上では，実社会とは異なり，同じ場所 (ウェブページ) に居るユーザ同士が自由にコミュニケーションをとることができない．

そこで我々は，そういった自然なコミュニケーションの発生を，ウェブ空間上で実現するシステム Uenew を開発した．開発には，3.3 節で実装したライブラリを用いた．なお，Uenew という名称はアイヌ語の「Uenewsar」(一緒に話して楽しむ) に由来している．

Uenew は，ウェブ上に存在する各ウェブページの URL 毎に ALM のグループを作成し，ブラウザのページ遷移に連動して各ユーザが常に閲覧中のウェブページに関するグループに参加している状態を実現する．例えば，現在ページ A を開いているユーザはページ A に関するグループに所属しており，ブラウザ上でリンクをクリックしてページ B に移動すると，ページ A のグループからは離脱してページ B のグループに新たに参加するという具合である．

このように URL 毎の即時的なグルーピングを行った上で，各グループ内でコメントの共有を行うというのが，Uenew の基本的なアイデアである．

このようなシステムをクライアント/サーバ型で構築した場合，ウェブ上に存在する全てのページ毎にグループを作成し管理するには，処理能力の高いサーバが必要となり，その設置・運用には大きなコストがかかる．またスケーラビリティはサーバの能力次第となるため，アクセスが増え過ぎるとサーバダウンが発生してしまうという問題がある．

Uenew は，Peer-to-Peer 技術によって一切のサーバを必要とせずにシステムを構築しているため，これらの問題からは無縁である．各ユーザのコンピュータが連携してシステムを構成することで，スケーラビリティや可用性に関して優位性が得られ，ハードウェア的な設備投資も一切不要となっている．

また，ALM によるプッシュ型の情報配信により，同じ場 (ウェブページ) を訪れている人々のコメントをリアルタイムに表示することができ，提案手法によりアーカイブの閲覧も可能である．

仮にアーカイブの閲覧ができず，チャットのようにリアルタイムなやり取りのみが可能である場合，閲覧者がまばらなウェブページでは何の情報共有も為されなくなってしまうが，Uenew であればウェブページという “場” に紐付いた経験の共有が可能となる．

5.2　既存サービスと関連研究

Uenew では，コミュニケーションのリアルタイム性が特徴のひとつとなっている．このようなリアルタイム性の高い気軽なコミュニケーションは，Twitter [13] をはじめとするマイクロブログの普及という

形で，近年ウェブユーザの間でも広まりつつある．但し，Twitter は場に紐付いたコミュニケーションではなく，人に紐付いたコミュニケーションを提供するサービスであるため，誰かとコミュニケーションをとるにはまずその相手を選定するという手間を経る必要がある．

また，Uenew では，ウェブページに対し利用者側からの情報付与が為され，その情報がユーザ間で共有される．このような機能を実現した既存のサービスとして，ソーシャルブックマークサービスや，Google SideWiki [5]，NewLiveWall [8] などが存在する．しかし，Twitter も含めこれらの既存サービスはクライアント/サーバ型システムとして構築されているため，先述したサーバダウン等の問題を抱えていると言える．さらに，これらの既存サービスではサーバへの負担が少ないプル型の情報配信を行っているため，チャットのようなリアルタイムな対話を行うことはできない．

他に，類似のシステムを P2P 方式で実装した研究事例 [16] も存在する．この文献では，閲覧中のウェブページに対し動的にアノテーション (annotation: 注釈) を付与し，P2P ネットワーク上で共有するシステムの提案を行っている．但し，文献内で提案されているシステムはハイブリッド P2P と呼ばれる形態を採用している．ハイブリッド P2P は，情報とノードの対応付けをサーバが行い，情報の授受自体はノード間で直接行うという方式である．クライアント/サーバ型システムと P2P 型システムを組み合わせた形態であり，単一障害点となるサーバが存在する．また，定期的なデータ同期により情報の更新が行き渡る仕組みになっており，リアルタイムな情報更新・表示は行っていない．

5.3 Uenew の実装

本節では，Uenew の実装と動作について述べる．DHT は一意な key を持つコンテンツに関してスケーラブルで高効率なネットワークを構築できる技術であり，3.2 節で提案したアーカイブ取得可能な ALM は DHT によるオーバーレイネットワーク上で構築される．

そこで，ウェブの閲覧行為には必ず一意な URL が付帯することに着目し，3.3 節で実装した DBAM の API を用いて，ウェブページの URL を key としたグループ管理を行うことで Uenew の実装を行った．

耐 churn 手法については，複製配置と加入時委譲，自動再 put を行い，複数 get は行わない設定とした．これは，Uenew の場合，ユーザがウェブページを移動する度にグループの参加/離脱に伴うアーカイブのやりとりが発生することから，オーバーレイへの加入やアーカイブの蓄積 (put) に比べ，取得 (get) の頻度がはるかに高くなることが予測されるためである．

Uenew では，ウェブブラウザから現在訪れているウェブページの URL を自動的に取得し，URL 毎にマルチキャストグループを生成する．グループに対する参加/離脱は，以下のような流れでウェブブラウザの動作と連動して行われる．

1. ブラウザでウェブページを開くと，自動的に URL を取得する．
2. 取得した URL をグループ名として指定し，マルチキャストグループへの参加処理を行う．
3. ブラウザでページ遷移が起こると，連動して現在のグループから離脱し，遷移後の URL をグループ名とするグループに参加する．

ブラウザの検出には Win32API を用いており，JNI(Java Native Interface) で呼び出しを行っている．現在は Internet Explorer，Sleipnir，Google Chrome の 3 種類のブラウザに対応しており，Uenew の起動がブラウザ起動の前後どちらであってもブラウザを認識することができる．異なるブラウザを使うユーザー間でも，同一の Peer-to-Peer ネットワークに参加してコミュニケーションを行うことが可能である．

URL の取得については，ブラウザのアドレスバーに表示されているものを取得し，先頭のプロトコル部分及び末尾のクエリストリングを切り捨てた文字列をグループ名として用いている．例えば，http://www.foo.co.jp/login?bar=baz という URL であれば，www.foo.co.jp/login がグループ名となる．

図 13 は，開発したシステムが動作する様子を示している．図中ではウェブブラウザ Google Chrome が起動しており，Java のドキュメントを参照してい

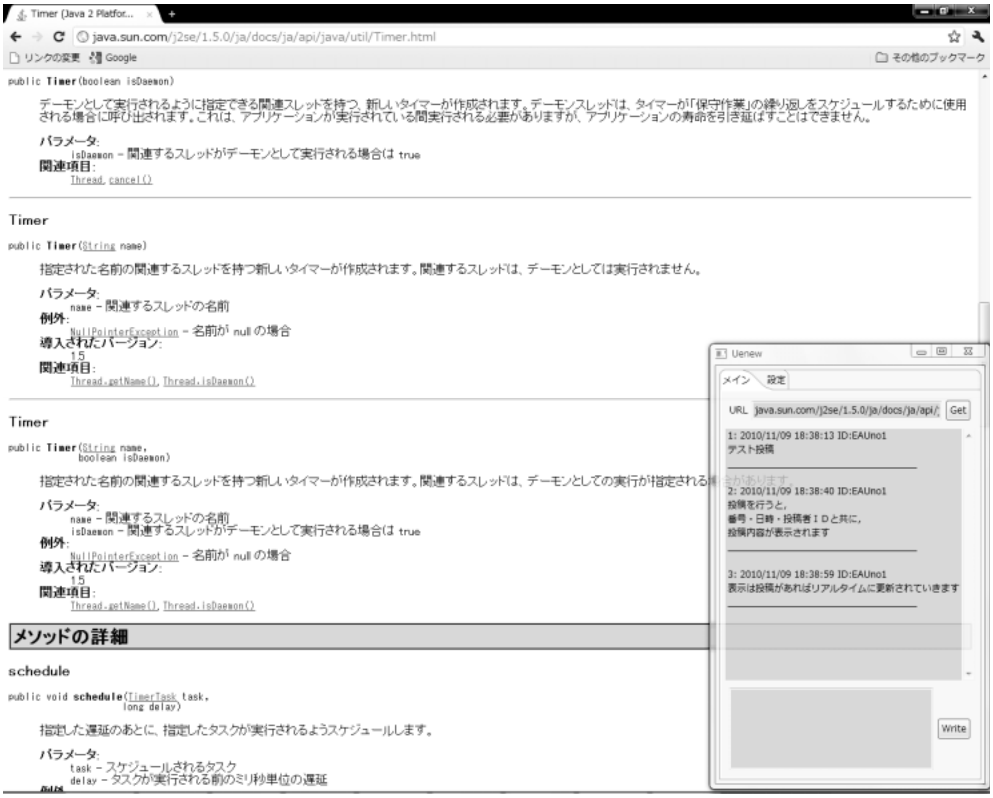


図 13　Uenew の動作図

る．右下に配置されているウィンドウが提案システム Uenew である．なお，Uenew の GUI には Standard Widget Toolkit(SWT) [4] を用いている．

Uenew のウィンドウはメインタブと設定タブの 2 つのタブを切り替えられるようになっている (図 14)．メインタブにはコメントの表示欄と入力欄の他に URL 欄があり，URL を手動で入力し，URL 欄の横にある Get ボタンを押下することでもグループ遷移が可能なようになっている．従って，予め示し合わせておけば，URL 以外の任意のグループ名でコミュニケーションを行うこともできる．また，Uenew ではプッシュ型のコメント配信が行われるため，コメント表示欄は投稿がある度に随時更新される．

設定タブでは，初期ノードアドレスや使用するブラウザ等を設定可能である．

Uenew の実装にはブラウザの検出や SWT など一部ネイティブコードを含むが，根幹である DBAM を実装した Overlay Weaver を含め，プログラムの大部分は Java で記述されているため，他プラットフォームへの移植も比較的容易であると考えられる．今後，Linux 版等も開発することを検討中である．

Uenew を用いることで，従来独立した行為であったウェブの閲覧にユーザ間のリアルタイムなつながりが生まれることが期待できる．また，誰かと偶然居合わせることや，場所 (ウェブページ) を決めて待ち合わせること，連れ立って散歩することなど，ウェブ空間上での行動選択の幅も広がると考えられる．加えて，特定の管理者が存在しないために中立性の高い俯瞰的なコミュニケーションの場を提供できるという点からも，当システムは有用であると考えられる．

Uenew も含め，本稿の提案により Overlay Weaver に追加されるモジュールを図 15 に示す．ハイレベルサービス層には QMAM 及び DBAM を，アプリケーション層には QMAM shell，DBAM shell，Uenew

図 14 Uenew の設定タブ

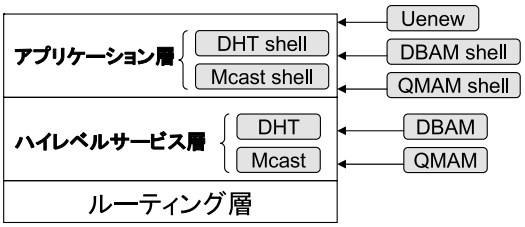


図 15 Overlay Weaver への機能追加

を実装している．なお，QMAM や DBAM の実装にあたりルーティング層等既存の実装箇所についても修正を加えているが，図には大幅な機能追加のある部分のみモジュールとして示している．

6 おわりに

近年，爆発的に規模の拡大を続けるインターネットにおいて，大規模なサービスを，高い品質と可用性を持たせて実現させることの需要が高まっている．既存のサービスの多くはクライアント/サーバ型のシステムとして構築されており，サーバへの負荷集中によるサービス停止や品質の低下が起こり得る．そこで本稿では，サーバが存在しない P2P 技術に着目し，ALM アルゴリズム Scribe におけるアーカイブ取得手法を提案した．

提案手法については Overlay Weaver の機能拡張という形でライブラリ及びサンプルプログラムを実装し，シミュレーション実験を実施した．この機能拡張は汎用的な実装であり，アーカイブ取得を必要とする様々な ALM システムに応用することができる．

また，提案手法の実応用として P2P 型のコミュニケーションシステム Uenew を開発し，サーバを介さないウェブユーザ同士のコミュニケーションが可能であることを示した．Uenew は，ウェブページを格納しているサーバに対し通信やデータ保存などの負荷増大を要求することは無く，ウェブページの管理者による掲示板設置等の作業も一切必要としない．このように，ハードウェア的な設備投資を必要とせず，ソフトウェアのみで大規模なシステムを構築できる点は P2P 技術の大きな特徴である．

なお，実装した提案手法及び Uenew のプログラムは，http://kussharo.complex.eng.hokudai.ac.jp/~software/ にて公開しており，自由に入手することができる．

本稿の提案手法における今後の課題としては，グループ毎のアクセス数の偏りによる，特定のノードへの負荷集中への対処が挙げられる．また，Uenew については，起動時にユーザが初期ノードを入力する必要があり，この操作を行わなければオーバーレイに参加することができない．現状では，上記ウェブサイトにて初期ノード情報を公開するという形態をとっているが，ユーザビリティを考慮するとある程度自動的に初期ノードとコンタクトをとれることが望ましく，これは検討課題である．他に，URL の階層構造を利用することで，ページ単位ではなくサイト単位でのコミュニケーションを可能にすること等も検討したいと考えている．

参考文献

[1] Castro, M., Druschel, P., Kermarrec, A. and Rowstron, A.: Scribe : A large-scale and decentralized application-level multicast infrastructure, *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8 (2002), pp. 1489–1499.

[2] Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J. and Stoica, I.: Towards a common api for structured peer-to-peer overlays, in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems* (*IPTPS '03*), Lecture Notes in Computer Science, Vol. 2735, Springer-Verlag, 2003, pp. 33–44.

[3] Deering, S.: Multicast routing in datagram inter-networks and extended LANs, *ACM Transactions on Computer Systems*, Vol. 8, No. 2 (1990), pp. 85–110.

[4] Eclipse Foundation: SWT: The Standard Widget Toolkit, http://www.eclipse.org/swt/.

[5] Google Inc.: Google Sidewiki, http://www.google.com/sidewiki/.

[6] Kaachoek, M. and Karger, D.: Koorde : A simple degreeoptimal distributed hash table, in *Proceedings of the 2nd International Workshop on Peer-to- Peer Systems* (*IPTPS '03*), Lecture Notes in Computer Science, Vol. 2735, Springer-Verlag, 2003, pp. 98–107.

[7] Maymounkov, P. and Mazieres, D.: Kademlia: A peer-to-peer information system based on the XOR metric, in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), Lecture Notes in Computer Science, Vol. 2429, Springer-Verlag, 2002, pp. 53–65.

[8] NewLiveWall!: NewLiveWall!, http://nlwnewg.appspot.com/.

[9] Rowstron, A. and Druschel, P.: Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Lecture Notes in Computer Science, Vol. 2218, Springer-Verlag, 2001, pp. 329–350.

[10] Shudo, K.: Churn Tolerance Improvement Techniques in an Algorithm-Neutral DHT, in *Proceedings of the 3rd International Conference on Autonomous Infrastructure, Management and Security: Scalability of Networks and Services*, Lecture Notes in Computer Science, Vol. 5637, Springer-Verlag, 2009, pp. 42–55.

[11] Shudo, K., Tanaka, Y. and Sekiguchi, S.: Overlay Weaver : An Overlay Construction Toolkit, *Computer Communications*, Vol. 31, Issue2 (2008), pp. 402–412.

[12] Stoica, I., Morris, R., Karger, D., Kaashoek, M. and Balakrishnan, H.: Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications, *ACM SIGCOMM Computer Communication Review*, Vol. 31, No. 4 (2001), pp. 149–160.

[13] Twitter: Twitter, http://twitter.com/.

[14] Zhao, B., Kubiatowicz, J. and Joseph, A.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing, *IEEE Journal on Selected Areas in Communications*, Vol. 22 (2004), pp. 41–53.

[15] 首藤一幸, 加藤大志, 門林雄基, 土井裕介: 構造化オーバレイにおける反復探索と再帰探索の比較, 情報処理学会研究報告 [システムソフトウェアとオペレーティング・システム], Vol. 2006, No. 86 (2006), pp. 9–16.

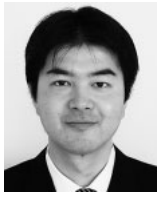
[16] 瀬川修: P2P による Web アノテーション方式, 社団法人情報処理学会第 68 回全国大会講演論文集, Vol. 4 (2006), pp. 53–54.

坂野遼平

2010 年北海道大学工学部情報エレクトロニクス学科卒業. 現在, 同大学大学院情報科学研究科複合情報学専攻修士課程に在学中. P2P ネットワーク, Web マイニングに興味を持つ. 情報処理学会学生会員.

佐藤晴彦

2006 年北海道大学大学院情報科学研究科コンピュータサイエンス専攻修士課程修了. 2008 年同大学院情報科学研究科複合情報学専攻博士後期課程修了. 博士 (情報科学). 2009 年北海道大学大学院情報科学研究科助教. 現在は関数型プログラムの形式的検証に関する研究に従事. 2011 年電子情報通信学会論文賞受賞.



小山　聡

2002 年京都大学大学院情報学研究科社会情報学専攻博士後期課程修了. 博士 (情報学). 2002 年–2007 年京都大学大学院情報学研究科社会情報学専攻助手. 2003 年–2004 年スタンフォード大学 Visiting Assistant Professor. 2007 年–2009 年京都大学大学院情報学研究科社会情報学専攻助教. 2009 年北海道大学大学院情報科学研究科複合情報学専攻准教授, 現在に至る. 2005 年度人工知能学会論文賞. 2009 年度日本データベース学会上林奨励賞.

栗原正仁

1980 年北海道大学大学院工学研究科情報工学専攻修士課程修了．北海道大学助手，講師，助教授，および北海道工業大学教授を経て，現在，北海道大学大学院情報科学研究科教授．2010 年より同研究科長．工学博士．ソフトウェア科学および人工知能の境界領域の研究に興味を持つ．1990 年情報処理学会創立 25 周年記念論文賞，2011 年電子情報通信学会論文賞受賞．情報処理学会，電子情報通信学会，人工知能学会各会員．