情報処理学会論文誌 Vol. 49 No. 6 1794–1802 (June 2008)

EntityCollaborator：SIP と DHT を用いたユビキタスコンピューティングのためのフレームワーク

粕谷貴司[†1] 中西泰人[†2]

ユビキタスコンピューティングにおける多くのアプリケーションでは，センサ情報から取得されたイベントやコンテキストといった離散的情報，動画や音声などのストリーミング情報といった連続的情報を利用するが，それらを統合的に扱えるフレームワークやミドルウェアは少ない．また，それらを統合的に扱う手段として SIP の利用が考えられるが，SIP では様々な仕様から構成されるため習熟が難しい．そのため我々は離散的，連続的情報を統合的に扱うため，バックエンドに SIP を用いた Java によるフレームワークである EntityCollaborator を提案する．EntityCollaborator の要件としては，SIP メソッドやセッション管理の抽象化と隠蔽，分散環境中のサービス探索機能の実装があった．そのため我々はイベント駆動型の API により SIP の隠蔽と抽象化を行い，P2P 探索技術の 1 つである DHT を利用することによりサービス探索機能を実現した．それにより離散的，連続的情報を統合的に扱うことが可能になり，SIP を採用したことで既存の SIP アプリケーションとの連携も可能になった．また，DHT を採用したことで，その特徴である高いスケーラビリティ，アドホック性，耐故障性なども得ることができた．

EntityCollaborator: A Ubiquitous Computing Framework Using SIP and DHT

Takashi Kasuya[†1] and Yasuto Nakanishi[†2]

Developing Ubiquitous Computing application is quite difficult, because of the needed for deep knowledge for network and devices. Therefore researches of framework and middleware are becoming active area. However, these frameworks mainly aim to handle discrete information such as event messages, therefore they seem to insufficient for handling continuous information such as sound and movie. Then we propose a framework called EntityCollaborator, which aim to converge discrete and continuous information using SIP. EntityCollaborator neaded to hide SIP methods and session management, and to provide a lookup service for finding component in a distributed environment. Therefore, to settle these specifications, we provided event-driven APIs and lookup service by using DHT, and which led EntityCollaborator to obtain these features; cooperation function with SIP-based applications, high scalability, adhoc nature, and robustness.

1. はじめに

携帯デバイスの小型化や高性能化やセンサネットワークの普及，ブロードバンド技術の発展によりユビキタスコンピューティング実現の可能性が高まっている．しかし，そうしたアプリケーション開発はネットワークやデバイスなどに関する高度な知識が要求されるために困難であるといわれている．そのため開発を支援するためのフレームワークやミドルウェアの研究がさかんに行われている[3]．しかし，既存研究の多くはセンサ情報やコンテキスト情報といったイベント情報の配布を主な目的としているため，動画や音声の利用が難しい．

そうした問題解決のための手段として，SIP（Session Initiation Protocol）[9] や XMPP [10] の利用が考えられる．SIP は IP 電話やビデオ会議，IM（Instant Messanger）に利用されているプロトコルであり，情報家電や携帯電話への適用も検討されている．XMPP は Google Talk などで利用されている XML ベースのプロトコルであり，IM によるメッセージ交換やログイン状況といったプレゼンス情報の通知に使われている．また，XMPP では Jingle というマルチメディアセッションのための規格も策定されており，VoIP や動画，アプリケーションの共有などのために利用されている．いずれも同様の能力を有しているが，XMPP/Jingle は現在 IM だけに適用されている規格であり，Jingle は RFC に採択されておらず一般的ではない．一方，SIP は IP-PBX や NGN（Next Generation Network）といった通信インフラのバックエンドにも利用されており，今後もより普及することが予想される．このような相互運用性の高さにより，SIP がより効果的であるといえる．

以上の背景より，イベント情報などの離散的な情報と動画などの連続的な情報を統合的に扱うため，バックエンドに SIP を利用した Java のフレームワークである EntityCollaborator（以下 EC）を提案する．EC の実現には SIP の隠蔽と抽象化，分散環境中のサービス探索などを実現する必要があった．我々はそれらをイベント駆動型の API と，P2P データ探索

†1 慶應義塾大学政策・メディア研究科
Graduate School of Media and Governance, Keio University
†2 慶應義塾大学環境情報学部
Faculty of Environmental Information, Keio University

技術の 1 つである DHT（Distributed Hash Table）によって実現した．また，IP 電話やソフトフォンなどの SIP クライアントをシステムの部品として扱い，プロトコルレベルでの連携を可能にするためにコンポーネント指向を採用した．

2. SIP を用いたフレームワーク

2.1 要　　件

本研究の対象としているアプリケーションは離散的，連続的な情報を統合的に利用する分散型のアプリケーションである．なお，本研究における離散的情報とは，センサ情報から生成されるイベントやコンテキスト，IM といったテキストベースの情報を意味し，連続的情報とは IP 電話における音声通話やビデオチャットによる動画などのストリーミング情報を意味する．これらの情報を統合的に扱うために，SIP を利用することが有効であることはすでに述べた．しかし，SIP は様々な仕様から成り立っており，またその運用にはセッション管理などについての理解も必要であるため，習熟に多くの時間がかかるといわれている．また，分散型のアプリケーションの開発には，分散環境中のサービスの探索機能が必要である．そのため SIP を利用したフレームワークの要件としては以下があるといえる．第 1 は複雑な SIP の隠蔽と抽象化であり，具体的には SIP によるセッション管理などの隠蔽や INVITE/BYE，MESSAGE，REGISTER などのメソッドの抽象化がある．第 2 は開発と運用を効率化するための API とミドルウェアの提供である．ミドルウェアの要件としては，分散サービスの記述，発見，管理機能，および通知されるイベント処理のための機能があるといえる．なお，SIP を利用しているライブラリやフレームワークの既存研究としては以下に示すものが提案されている．

2.2 関 連 研 究

SIP を利用したフレームワークの先行研究としては CINEMA [1]，SPLAT [11] などがある．

CINEMA はマルチメディアコラボレーションのためのソフトウェア基盤であり，様々な SIP サーバから構成されている．サーバ群は SIP プロキシ，プレゼンスサーバ，レジストラなどから構成されており，CPL（Call Processing Language）[6] などによって制御できる．また，Bluetooth による位置情報取得のための機能が実装されており，位置情報に応じたマルチメディアサービスの提供が可能である．なお，提供されている C，C++ライブラリによる独自アプリケーションの開発も可能である．

SPLAT は SIP によるマルチメディア会議やゲームといったサービスと既存のアプリケーションを連携させるためのプラットフォームであり，SIP の機能を提供するための高レベルの API 群と，その運用を可能にする Client-side SIP service，SIP による独自のサービスを定義した Network infrastracture blocks から構成されている．また，低レベルな API も提供しており，SIP のフローごとに細かな設定も可能である．

なお，SIP を利用したアプリケーションの具体的な例としては，“取り込み中”や “退席中” といったプレゼンス情報の更新をセンサ情報を用いて行うシステムが Web システムと SIP アプリケーションの統合を目的とする SIP Servlet [5] において提案されている．

これらの関連研究では SIP アプリケーション開発のための有用な API を提供しているが，いずれもクライアント・サーバ型のアーキテクチャを採用しており，ホームネットワークやオフィスでの運用を対象とするためスケーラビリティやアドホック性が低い．また，提供されている API は SIP に詳しい技術者以外での習熟，アプリケーションのプロトタイピングが困難である．

そのため我々はイベント駆動型の API による SIP の隠蔽と抽象化を行い，アドホック性を考慮した木構造による P2P アーキテクチャによりノードの管理を行うフレームワークの設計と実装を先行して行った [15]．しかし，木構造によるノード管理には特定のグループに対するマルチキャストが容易であるという利点があったが，ユーザがつねに参加しているオーバレイネットワークの構造を考慮する必要があったこと，動的なノードの管理が難しいという問題があった．

2.3 提 案 手 法

それらの問題解決のため，DHT によるノード管理の採用を試みた．その理由としては，ユーザはハッシュテーブルの管理だけを行えばよいため学習も容易であること，動的なノードの管理も提案されている既存アルゴリズムの適用で可能であることがある．なお，SIP と DHT を利用することでスケーラビリティを向上させている研究としては，SOSIMPLE [2] などがある．ロケーションサーバを用いないルーティングの実現性などから注目を集めているが，いずれも適用の対象を IP 電話としており，フレームワークとして提供する研究は少ない．また，現在提供されている SIP-DHT のライブラリとしては，SIPDHT [12] があり，SIP を利用した DHT のルーティングを可能にしている．

3. EntityCollaborator

前章で述べた機能の実現を目指して我々が開発しているフレームワークが EC である．ユーザは提供されたライブラリを利用することで離散的な情報と連続的な情報を統合的に扱う SIP アプリケーションの開発が可能であり，その実装には Java（JDK 5.0）を用いて

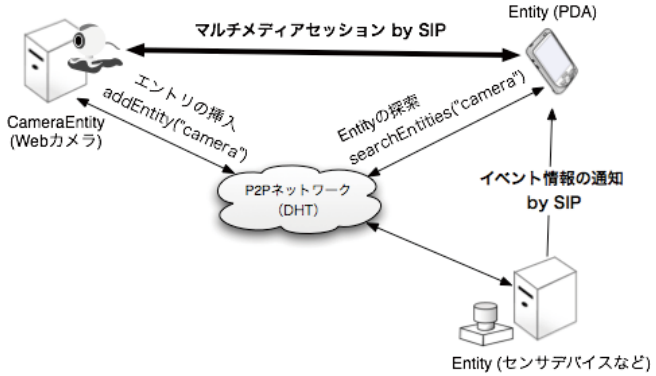

図 1　EntityCollaborator による分散アプリケーションの構成例
Fig. 1　An example of composition of distributed application with EntityCollaborator.

いる．なお，EC はネットワーク部の隠蔽と抽象化を行うものであり，デバイスの抽象化などはサポートしていない．

EC によって開発されるアプリケーションは分散環境中のソフトウェアコンポーネントである Entity を，メッセージの交換やマルチメディアセッションの開設によって協調させることで動作する．Entity は SIP クライアント，センサ，ウェブカメラなどを表すコンポーネントであり，任意のキーワードに紐付けることで分散環境中での探索が可能になる．たとえば，ウェブカメラによるストリーミング機能を持つ CameraEntity に対して “camera” というキーワードを付加することで，同じネットワーク中に属する Entity から同じキーワードによる探索によって CameraEntity の参照が得られる．ユーザは得られた CameraEntity の参照に対して，マルチメディアセッションのリクエストやメッセージの送信を行うことができる（**図 1**）．

3.1　要 素 技 術

EC では主な要素技術として SIP と DHT を利用している．本節ではそれぞれの利点とフレームワークに適用する際の課題，および提案手法について述べる．

3.1.1　SIP

SIP は通信のセッションの開設を制御する「シグナリング」実現のためのプロトコルである．SIP はセッションの開設以外の機能は持たず，それらの機能は他のプロトコルと組み合わせて実現することを特徴としている．たとえばセッションのトランスポート・アドレスやマルチメディアデータの圧縮方式などの交換には SDP（Session Description Protocol），動画など送信には RTP（Realtime Transport Protocol）を用いる．

フレームワークに SIP を採用することの利点としては以下の 3 つが考えられる．第 1 に離散的な情報と連続的な情報を統合的に扱えること，第 2 に既存の SIP ベースのアプリケーションや通信インフラとの連携が容易に行えること，第 3 に SIP はアプリケーションレイヤのプロトコルであるため，メーラやブラウザのように多様なクライアントの提供が可能なことである．

しかし，その問題点として次の 2 つが考えられる．第 1 は SIP の複雑さである．この問題については 2.1 節ですでに述べたとおりであり，我々はイベント駆動型の API によって SIP の隠蔽と抽象化を行うことで対応した．第 2 は SIP のオーバヘッドである．SIP はそのヘッダ部分だけでつねに 200 バイトほどになり，メッセージの受け手側も同様のヘッダでレスポンスを返す必要があるためオーバヘッドが大きい．そのため即時性を必要とするアプリケーションに対して有効でない可能性がある．こうした課題には，Fukuda らが提案している SIP と RTP を併用した手法[4] による解決が考えられる．Fukuda らは位置情報などの離散的な情報を RTP を用いて通知することでオーバヘッドの削減を行っており，この手法は通知間隔の短い情報の受け渡しには有効である．ただし，EC においての実装や検証はまだ行っていない．

3.1.2　DHT

DHT は分散環境中でハッシュテーブルを構築することにより，情報の分散とデータ探索機能を提供する技術である．DHT の特徴としては，サーバが存在しないピュア P2P であること，高いスケーラビリティやアドホック性を持つこと，単一障害点の回避により，ネットワーク中に存在するオブジェクトはたいてい発見できることがあげられる．そのためフレームワークに適用することで，設計課題の 1 つであったスケーラビリティの解決が期待できる．また，アドホック性や耐故障性などの特徴も取り入れることも可能であると考えられる．

また，その他の特徴として Chord[13]，CAN[8] など様々なアルゴリズムが提案されていることがあげられる．最も一般的なアルゴリズムである Chord はリング上のハッシュ空間を持つことを特徴としており，$O(\log N)$[⋆1] の探索コストと，$O(\log^2 N)$ の参加・離脱コストをそれぞれ平均計算量として実現している．また，きわめて頑健性が高く，Chord は全体の半分のホストが同時に Fail Stop しても各ホストは $(1-1/N)$ の確率で即座に正確なリン

⋆1 N はノード数．

ク情報の取得ができる．

DHT の問題点としては探索の柔軟性に欠けることがあげられる．我々は DHT の機能を利用して，任意文字列と分散コンポーネントである Entity を対応させた参照方式を考案したが，現在の実装では単一のキーワードによる探索しか行えない．しかし近年，ハッシュ関数の工夫や特徴ベクトルの導入などにる類似探索を実現する技術の研究[16),17)] が行われており，これらの技術を応用することで，柔軟な探索も可能になると考えられる．たとえばキーワードから特徴ベクトルを抽出し，抽出したベクトル値をもとに探索を行う手法を用いることで，複数のキーワードによる探索も可能であると考えられる．

3.2　システム概要

図 2 に EC のシステム構成を示す．システムはユーザが実装するコンポーネントである Entity，Entity を保持するコンテナである EntityContainer，SIP メッセージの処理モジュールである SipCore，DHT の機能を提供するモジュールである Chord，システムのフロントエンドである *EntityCollaborator* の 5 つのモジュールで構成されている．それぞれについて以下で詳細を述べる．

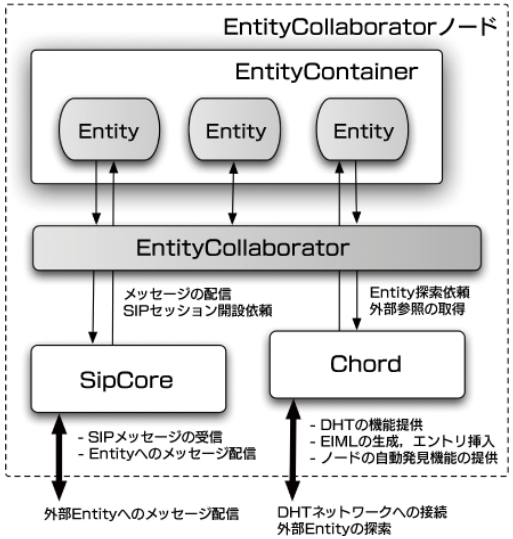

図 2　EntityCollaborator のシステム構成
Fig. 2　The system architecture of EntityCollaborator.

3.2.1　Entity，EntityContainer

Entity はアプリケーション開発の中心的な要素であり，インタフェースとして定義されている．ユーザは任意の機能を持った Entity を抽象クラスである AbstractEntity を継承することで開発し，それらを組み合わせることでアプリケーションの構成を行うことができる．Entity の粒度や構成数は任意であり，同じノード内に存在する Entity は EntityContainer により保持されるため，1 つのノードに複数存在させることが可能である．このようなコンポーネント指向の利点としては，コンポーネントの再利用による生産性，システムの保守性，冗長性や拡張性の向上があるといわれている．たとえばセンサ情報を配信する SensorEntity を開発した場合，同じ SensorEntity を他のアプリケーションでも同様の目的で利用することができる．さらに同じアプリケーション内に複数の Entity を配置することで，冗長性の高いアプリケーション構成も可能である．

Entity には SIP による通信を隠蔽，抽象化したメソッドが定義されており，それによりイベント駆動型のプログラミングが可能である．**表 1** に各 SIP メソッドに対応したメソッドとコールバックメソッドであるイベントハンドラを示す．なお，EC のシステムは各 Entity にシステムが起動時にユニークな ID である SIP URI を与えることで，イベントの配布先の Entity の特定を行っている．

図 3 に Entity のクラス構成を示す．Entity はイベントハンドラを定義した EntityListener，抽象クラスである AbstractEntity，センサを表す SensorData，位置情報を表す Location から構成されている．AbstractEntity はヘルパークラスであり，sendMessage() などのメソッドがあらかじめ実装されている．ユーザが独自の Entity を開発する際はこのクラスを拡張する必要があり，継承したメソッドを利用することでメッセージ送信やマルチメディアセッションの開設を行う．なお，EntityListener に定義されているイベントハンドラは AbstractEntity に空のメソッドとして記述されているため，ユーザは必要なハンドラだけ

表 1　EntityCollaborator で使用される SIP メソッド
Table 1　Methods for SIP used in EntityCollaborator.

SIP メソッド	対応するメソッド，ハンドラ
INVITE	sendOffer(), receiveOffer(), receiveAnswer()
BYE	sendBye(), receiveBye()
MESSAGE	sendMessage(), receiveMessage()
SUBSCRIBE	subscribe(), receiveSubscribe()
NOTIFY	sendMessage(), receiveNotify()
REGISTER	-(Entity にメソッドは定義されていない)

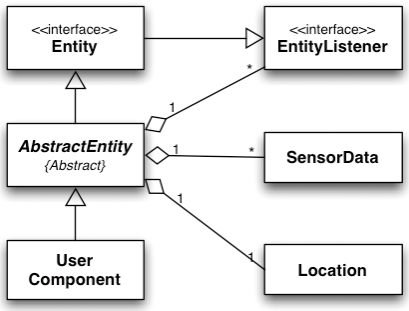

図 3　Entity の UML
Fig. 3　The UML description for Entity.

をオーバライドして使用する．また，SensorData, Location は Entity に対して任意に設定，追加することが可能である．これらのクラスには数値データや位置情報を保持させることで，外部の Entity からでも参照が可能になる．

3.2.2　SipCore

Entity は SipCore に対して処理を委譲することで SIP メッセージの送信を行い，外部からの SIP メッセージを受信することで SIP URI で指定された Entity に対応したイベントハンドラを呼び出す．そのため SIP プロキシのような機能を持っているといえる．**図 4** に INVITE メッセージによるオファー/アンサーモデルによるマルチメディアセッションの開設の例を示す．通信元の Entity は sendOffer() メソッドに SDP を引数とすることにより，通信先の Entity の receiveOffer() を呼び出す．次に送信先の Entity は SDP から通信可能なフォーマットを選び出し，それを receiveOffer() の返り値とすることで通信元の receiveAnswer() に対してその SDP が渡すことができる．以上の手順により，それぞれの Entity は渡された SDP をもとにマルチメディアセッションを開設することができる．

なお，SIPCore は SIP クライアントからの REGISTER メソッドによる登録を受けると，登録元の SIP クライアントを Entity でラップする処理を行う．この際，SIP アドレスの Display Name をキーワードとして登録するため，同一ネットワーク上の Entity からはそれによって参照可能になり，SIP クライアントへのセッションの開設やメッセージの送信を行うことができる．なお，現在連携を確認している SIP クライアントは XLite のみである．

3.2.3　Chord

我々は DHT を利用した探索方法として，Entity に対して任意のキーワードを付与し，そ

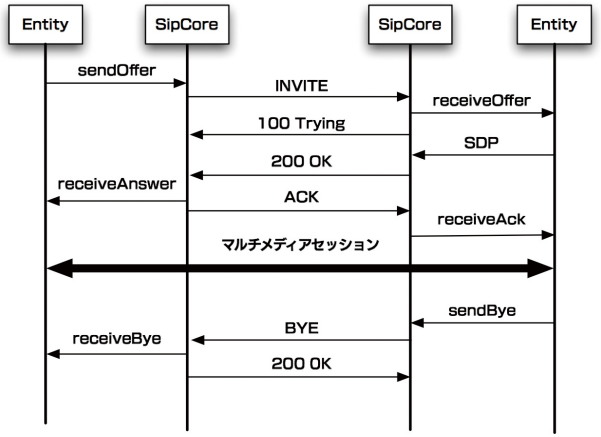

図 4　INVITE，BYE メソッドによるマルチメディアセッションの確立
Fig. 4　Multimedia session connection via INVITE and BYE methods.

のキーワードと対応した Entity のメタ情報を記載した XML を DHT のエントリとする手法を考案した．メタ情報を記載した XML である EIML（Entity Information Markup Language）は，Entity の SIP-URI や各種パラメータなどを記述したものであり，システムによって自動的に生成される．Chord はそれらのアルゴリズムを実装したクラスである．その動作の例として Web カメラの映像を配信する CameraEntity と，その情報を利用する SearchEntity による探索とセッション開設の手順を述べる．

(1)　CameraEntity に対して addKeyword() を用いて “camera” というキーワードで登録する．
(2)　キーワードが登録された Entity をシステムは EIML に変換し，キーワードを key，EIML を value として DHT に挿入する．
(3)　分散環境中の SearchEntity は，*EntityCollaborator* クラスの searchEntities() というメソッドを用いてキーワードに登録された Entity の探索を行う．
(4)　システムは DHT から指定されたキーワードのエントリを取得し，得られた EIML から CameraEntity への参照を構築し，探索元の SearchEntity に対してその参照を渡す．
(5)　SearchEntity は得られた参照に対して，メッセージの送信やマルチメディアセッションの確立などを行う．

```
public class CameraEntity extends AbstractEntity {
  public CameraEntity() {
      // キーワードの追加，および Web カメラなどの初期化（省略）
      addKeyword("camera");
      init();
  }
   public void receiveMessage(EntityEvent e) {
       // 受け取ったメッセージをコンソールへ出力
       System.out.println(e.getMessage());
   }
   public SessionDescription receiveOffer(EntityEvent e) {
       // オファー元からの SDP の取得,
       // およびアンサーの SDP の生成（省略）
      SessionDescription sdp =e.getSessionDescription();
      SessionDescription ret = getResponseSDP(sdp);
      // ストリーミングの開始（省略）
      startStreaming(sdp);
      return ret;
   }
}
```

図 5 キーワードの追加の例
Fig. 5 A sample of adding a keyword.

```
public class SearchEntity extends AbstractEntity {
  public SearchEntity() {
      // フロントエンドの取得
      EntityCollaborator ec = EntityCollaborator.getInstance();
      // camera に関連づけられた Entity をネットワーク中から探索
      Entity[] entities = ec.searchEntities("camera");
      for (Entity entity : entities) {
          // SDP の生成（省略）
          SessionDescription sdp = getRequestSessionDescription();
          // 探索された Entity に対して
          // マルチメディアセッションのリクエスト（オファー）
          sendOffer(sdp, entity);
          // メッセージの送信
          sendMessage("Hello World!", entity);
      }
  }
   public void receiveMessage(EntityEvent e) {}
   public void rceiveAnswer(EntityEvent e) {
       // アンサーの SDP の取得後，ストリーミングの受信
       SessionDescription sdp = e.getSessionDescription();
       receiveMedia(sdp);
   }
}
```

図 6 Entity の探索例
Fig. 6 A sample of searching an Entity.

図 5，**図 6** に実際のコーディング例を示す．これらの例は上に述べた Entity の探索を行った後，発見された Entity に対してマルチメディアのリクエスト（sendOffer）およびメッセージの送信（sendMessage）を行っている．

なお，DHT のルーティングのアルゴリズムとしては代表的な Chord を用いた．アルゴリズムの実装手法としては SIP による実装，独自プロトコルによる実装，既存のライブラリの利用が考えられるが，我々は SIP によるオーバヘッドの削減と実装の容易さ，信頼性の高さから既存のライブラリを利用している．利用するライブラリとしては各種アルゴリズムのサポートと，DHT 機能の扱いやすさにより，産業技術総合研究所で開発が行われている OverlayWeaver [14] を採用した．また，P2P アプリケーションには起動時にブートストラップノードを指定する必要があるが，ユーザビリティの向上のため，我々は LAN 内のノードの自動発見機能を実装した．自動発見機能は LAN 内に Hello パケットをブロードキャストし，もしピアが存在すれば Ack パケットを返信することで実現している．

3.2.4 EntityCollaborator

EntityCollaborator はシステムのフロントエンドとして機能するクラスであり，システムの起動や Entity の探索のためのメソッドを持つ．システムの起動は基本的に SipCore の初期化，Chord の初期化，ブートストラップノードの指定，自動発見による DHT のネットワークへの参加，Entity の初期化と追加の順に行う．**図 7** にシステムの起動のためコードを示す．

3.3 アプリケーションへの適用例

本節では EC の特徴を活かしたアプリケーションの開発例について述べる．

モニタリングシステム 従来のモニタリングシステムでは動画のみ，またはセンサのみという構成が多かった．EC はそうした情報を統合的に扱うことが可能なため，動画を送信しながらセンサ情報を送信するシステムの開発が容易に行える．**図 8** に製作したモニタリングシステムの構成を示す．

システムはセンサ情報を通知する SensorEntity，動画を送信する VideoTransmitterEntity，モニタリングを行う MonitoringEntity から構成される．MonitoringEntity は起動時

```
public class Main {
   public static void main(String[] args) throws Exception {
      EntityCollaborator ec = EntityCollaborator.getInstance();
      // SIP スタックの初期化
      ec.initiateSipCore();
      // Chord の初期化，LAN 内のピアの自動発見
      ec.findPeer();
      // Entity の追加
      ec.addEntity(new CameraEntity());
      ec.addEntity(new SearchEntity());
   }
}
```

図 7　EntityCollaborator の起動
Fig. 7　A sample of starting up EntityCollaborator.

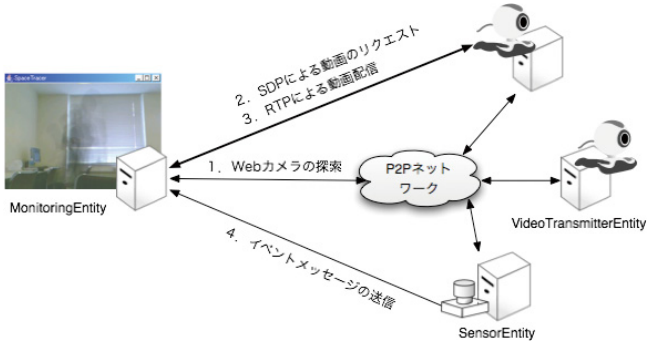

図 8　モニタリングシステム構成
Fig. 8　A monitoring system by EntityCollaborator.

にネットワーク中からそれぞれの Entity の探索を行い，モニタリングの対象とすることができる．また，新規に Entity を追加することで，容易に機能拡張やセンサ情報の共有などが行える．たとえばソフトウェアボタンによる入力をモニタリングの対象としたい場合，ボタンを押すと MonitoringEntity の探索とメッセージ送信を行う Entity を製作し，同じネットワークに追加すればよい．さらに，ほかにも同じセンサ情報を利用したいアプリケーションがある場合，MonitoringEntity と同じキーワードの Entity を登録するだけでセンサ情報の配信が可能になる．

P2P システム　PlaceEngine[7] は Wi-Fi 機器を使って簡単に現在位置を推定し，周辺の

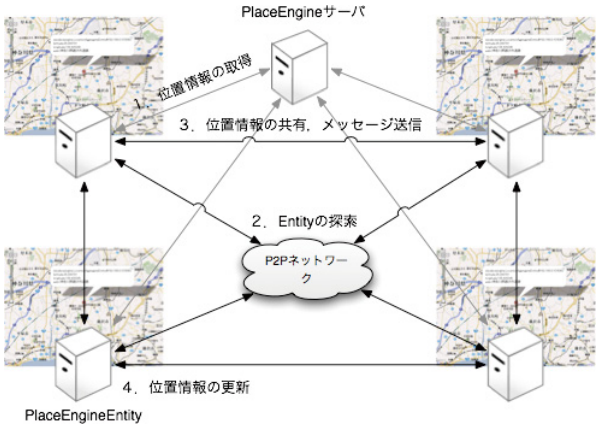

図 9　位置情報共有システム構成
Fig. 9　A location information sharing system by EntityCollaborator.

関連情報の提供を容易にするサービスである．PlaceEngine では API が公開されており，自身の位置情報を取得し GoogleMaps などと連携させることができるが，友人などの位置情報は容易に共有することができない．EntityCollaborator ではそのようなスタンドアロンのアプリケーションに対して P2P の性質を容易に付与することができる．そのため我々は PlaceEngine による位置共有のためのシステムの製作を行った．**図 9** に製作したシステムの概要を示す．なお，システムの View には GoogleMaps を利用している．

システムのクライアントである PlaceEngineEntity は位置情報の取得を PlaceEngine API を通じて行い，ネットワーク中の同じクライアントを探索，位置情報の通知を行う．通知を受けたクライアントはそれぞれのクライアントの位置情報の更新を行うことで位置の共有を行っている．

ソフトフォンとの連携　EC はバックエンドに SIP を利用しているため，SIP クライアントとの連携が可能である．また，EC は Java を用いて実装しているため，Web との連携のために開発された多くのライブラリが利用可能である．このような特徴の検証のため．ソフトフォンである XLite を利用して音声や IM のログを残し，それを Flash を用いて可視化するシステムの開発を行った．**図 10** にシステム構成を示す．

システムは以下の 4 つのモジュールから構成されている．第 1 は SIP クライアントである XLite である．XLite は正しく設定されると自動的に REGISTER メソッドを EC のシ

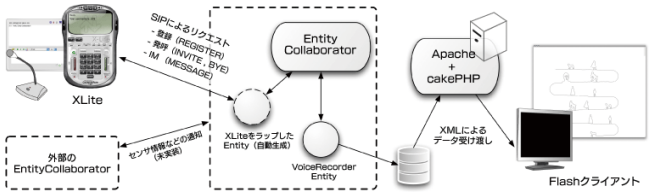

図 10　ソフトフォンとの連携システム構成
Fig. 10　A voice recording system integrated with soft-phone by EntityCollaborator.

ステムに送信し，自身を EC に登録する．EC 側では REGISTER を受けると自動的に送信元のクライアントを Entity にラップし，設定されている SIP アドレスの Display Name によってネットワーク上から参照できるようにする．第 2 は XLite からの SIP によるリクエストを受けて録音や IM の保存などを行う VoiceRecorderEntity である．この Entity は INVITE によるリクエストを受けると，RTP による音声データの受信と録音を開始し，同時に音楽ファイルからのデータを RTP によって発呼側に送信する．また，MESSAGE によるリクエストを受けると，受信メッセージの保存を行う．第 3 は保存されたデータを XML で出力するモジュールであり，第 4 は XML の情報を解析して可視化する Flash クライアントである．クライアントでは録音データが時系列に沿って再生され，再生と同時に一筆書きの要領で人物などのオブジェクトが描画される．描画されたオブジェクトをマウスでクリックすることで，録音データや IM などを確認することができる．

なお，本システムのデモンストレーション展示を 2007 年 11 月に行ったが，2 日間の運用期間中，EC によるシステムは順調に稼働し続け，障害も特に起きることはなかった．もっと厳密な頑健性の検証の必要性はあるが，運用の結果から，SIP セッションやメモリ内のリソースの管理なども正しく行われているといえるだろう．

4. 考察および今後の課題

本章では EC の設計と実装についての考察を行う．

我々は Java を実装言語とし，イベント駆動型の API やコンポーネント指向により，SIP の隠蔽やユーザビリティの向上などを試みた．これらは Java の技術者によく知られたプログラミングモデルであり，抽象クラスをヘルパークラスとして用いているためにコードの記述量も少なく，また SIP の隠蔽にも成功している．しかし，動画などを利用するためには SIP のオファー/アンサーモデルや SDP，RTP によるストリーミング，それらの実装に用いる JMF などの知識が必要となる．我々はそのためのコンポーネントを提供しているが，カスタマイズにはいずれも深い知識が不可欠となる．今後はそうしたユーザビリティの改善が課題としてあげられる．

また，先行して実装した木構造によるアーキテクチャの問題解決のため，DHT を用いたキーワードによる探索手法を採用を試みた．なお，課題としては P2P の構造を意識しないことによるユーザビリティの向上，動的なノード管理の実現があった．EC では DHT と SIP を Chord と SipCore という 2 つのモジュールに分けて設計したため，DHT の特徴である高スケーラビリティ，アドホック性，耐故障性などを損なわずに取り込むことに成功したといえる．そのため木構造での課題をおおむね解決することができたといえるだろう．また，それにより複数の Entity を利用したアドホック性の高いアプリケーション開発も可能になった．しかし，3.1.2 項で述べた柔軟なオブジェクト探索手法はいまだに実装しておらず，DHT による高いスケーラビリティやアドホック性を活かしたアプリケーションもシミュレーションレベルでしか実現していない．それらの実現が今後の課題としてあげられる．

また，エントリの value として定義した EIML は XML であり，ノード数とエントリの増加による情報量の増加がスケーラビリティ低下の一因として懸念されたため，我々は評価実験による検証を行った．なお，実験にはルータを介して有線で接続された PC3 台（OS: WindowsXP）を利用して行った．利用した PC は 2.8 GHz の CPU，1 GB のメモリを搭載した 1 台と 3 GHz の CPU，2 GB のメモリを搭載した 2 台であった．実験の結果によると，ノード数を 3 つに固定しエントリの数をパラメータとした場合，キーが 2,000 個までの平均の探索時間は 12.48 ミリ秒であった．また，エントリ数を 1,000 個に固定しノード数を変化させた場合，ノード数が 100 個までならその探索時間は平均で 15 ミリ秒程度であった．これらの結果は Chord の性質を反映しており，この範囲であればリアルタイム性の高いアプリケーションの運用が可能であるといえるが，DHT の特徴である高スケーラビリティを活かしたアプリケーションのためには，さらなる評価実験が必要であると考えられる．

5. ま と め

本稿では離散的，連続的な情報を統合的に扱うため，バックエンドに SIP を利用した Java によるフレームワークである EC について述べた．SIP を利用することで，イベントやコンテキストといったテキストベース情報と，IP 電話による音声やビデオチャットなどの動画によるストリーミング情報を統合的に扱うことが可能になった．また，EC ではイベント駆動型の API によって SIP の隠蔽と抽象化を行っているため，既存の SIP クライアントと

の連携も可能となった．また，分散環境中のサービス探索に DHT を利用することで高いスケーラビリティ，アドホック性，耐故障性などの特徴も得ることができた．さらに，設計指針としてコンポーネント指向を採用したため，機能の拡張やアプリケーション間の協調が容易に行えるという特徴を得た．今後の展望としてはユーザビリティの改善，DHT による柔軟なオブジェクト探索の実装，高いスケーラビリティやアドホック性を活かしたアプリケーションの提案と実装がある．

参 考 文 献

1) Berger, S., Schulzrinne, H., Sidiroglou, S. and Wu, X.: Ubiquitous Computing Using SIP, *International Workshop on Network and Operating System Support for Digital Audio and Video* (*NOSSDAV'03*) (2003).
2) Bryan, D.A. and Lowekamp, B.B.: SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System, *International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications* (*AAA-IDEA '05*) (2005).
3) Endres, C., Butz, A. and MacWilliams, A.: A Survey of Software Infrastructures and Frameworks for Ubiquitous Computing, *Mobile Information Systems Journal*, Vol.1, No.1 (2005).
4) Fukuda, K., Saito, K. and Tanaka, S.: A Study on Presence System for Distribution of Positional Coordinates, *IPSJ SIG Technical Reports 2005-UBI-7*, pp.163–169 (2005).
5) Group, J.E.: JSR 289 SIP Servlet Specification.
6) Lennox, J., Wu, X. and Schulzrinne, H.: Call Processing Language (CPL): A Language for User Control of Internet Telephony Services, RFC3880 (2004).
7) PlaceEngine. http://www.placeengine.com/
8) Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S.: A scalable content-addressable network, *Proc. 2001 ACM SIGCOMM Conference*, Vol.31, pp.161–172, ACM Press (2001).
9) Rosenberg, J., Camarillo, G., Johnston, A., Peterson, J., Sparks, R. and Schooler, E.: SIP: Session Initiation Protocol, RFC3261 (2002).
10) Saint-Andre, P. and Saint-Andre, E.P.: Extensible Messaging and Presence Protocol (XMPP): Core, RFC3920 (2004).
11) Singh, A., Acharya, A., Mahadevan, P. and Shae, Z.-Y.: SPLAT: A unified SIP services platform for VoIP applications: Research Articles, *International Journal of Communication Systems*, Vol.19, No.4, pp.425–444 (2006).
12) SIP-DHT. http://sipdht.sourceforge.net/index.html.en
13) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, *Proc. 2001 ACM SIGCOMM Conference* (2001).
14) 首藤一幸，田中良夫，関口智嗣：オーバレイ構築ツールキット Overlay Weaver，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG12 (ACS 15), pp.358–367 (2006).
15) 粕谷貴司，中西泰人：EntityCollaborator：SIP を用いたユビキタスコンピューティングフレームワーク，マルチメディア，分散，協調とモバイル（DICOMO2007）シンポジウム論文集，pp.893–896 (2006).
16) 呉　俊輝，天笠俊之，北川　博：P2P 環境における構造概要を利用した XML データの検索，電子情報通信学会第 17 回データ工学ワークショップ（DEWS2006）(2006).
17) 倉沢　央，正田備也，高須淳宏，安達　淳：P2P 情報検索における索引とファイルの分散配置手法，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.2007, No.36, pp.147–154 (2007).

(平成 19 年 9 月 7 日受付)
(平成 20 年 2 月 5 日採録)

粕谷　貴司（正会員）

2003 年茨城高等専門学校電子制御工学科卒業，2005 年東京農工大学工学部情報コミュニケーション工学科卒業，2007 年慶應義塾大学大学院政策メディア研究科メディアデザイン専攻修了．ユビキタスコンピューティング・フレームワークの研究に従事．

中西　泰人（正会員）

1998 年東京大学大学院工学系研究科博士課程修了．同年より電気通信大学大学院情報システム学研究科助手．2003 年東京農工大学工学部情報コミュニケーション工学科助教授．2005 年慶應義塾大学環境情報学部助教授．2007 年同学部准教授，現在に至る．博士（工学）．感性情報処理，ヒューマンインタフェース，モバイルコミュニケーション等の研究に従事．2004 年グッドデザイン賞受賞，人工知能学会，ACM，日本建築学会各会員．