Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

Voluntary Sensing/Computing Architecture の提案

久保 健[†]　杉山 浩平[†]　田上 敦士[†]　長谷川 輝之[†]
Jean Walrand[††]

スマートフォンなどの高性能携帯デバイスがもつセンシング・コンピューティング資源を第三者に開放することで，スマートフォン群を新たな情報インフラとして利用可能とするアーキテクチャを提案する．具体的には，依頼者が他者のスマートフォン（実行者）に様々な情報を取得・加工するためのアプリケーションソフト（プラグインプログラム）を送り込み，実行者がそのプログラムを実行し，実行結果を依頼者に返答する．近年注目を集めている participatory sensing と異なり，提案アーキテクチャでは，実行者は「依頼者が誰で，どのようなプログラムを実行すべきか」を事前には知らない，という前提を置く．そのため，提案アーキテクチャは，依頼者から任意のタイミングで送付された任意の処理を行うプラグインプログラムを，実行者が受け入れ実行する仕組みを提供する．さらに，第三者に資源を提供するモチベーションを持ち得ない実行者に資源提供を促すインセンティブメカニズム実現に必要な，報酬の支払いを支援する仕組みを提供する．本論文ではアーキテクチャの提案とともに，Android への具体的な実装方法についても説明する．

A Proposal on Voluntary Computing/Sensing Architecture

Takeshi Kubo[†]　Kohei Sugiyama[†] Atsushi Tagami[†]
Teruyuki Hasegawa[†] and Jean Walrand[††]

In this paper, we propose a new architecture that opens smartphones' sensing/computing resources to other users. This means that smartphones themselves become a new infrastructure for sensing and computing. In the proposed architecture, a client feeds a program code (plugin program) to smartphones (performers) to obtain and process information from the performers, and they reply the results to the client. Different from participatory sensing, we assume that each performer does not know the plugin program that it should execute a priori. Therefore, the proposed architecture provides the mechanism of dynamic loading and executing arbitrary plugin programs at any timing, and the mechanism of rewarding for performers that is an essential for incentive mechanism for encouraging smartphone users to offer their resources. We also present the implementation of the dynamic loading and executing mechanism on Android.

1. はじめに

スマートフォンが携帯電話端末の主流となりつつある．米ガートナー社の調べ[1]によれば，世界の年間スマートフォン販売台数は 2011 年で約 5 億台（全携帯電話端末の販売数の約 3 分の 1），2015 年には約 11 億台（全携帯電話端末数の販売数の約 2 分の 1）と予想されている．これらのスマートフォンに搭載される Android OS や iOS 等のプラットフォームは，サードパーティによるアプリケーション開発を可能としており，スマートフォンが備える各種センサ（GPS，加速度，カメラなど）や高解像度タッチスクリーンを駆使した様々なサードパーティ製アプリケーションが登場している．また，これらのプラットフォームは，マルチタスキングをサポートしており，複数のサービス，アプリケーションを同時に実行させることができる．そのため，年々向上するスマートフォンのコンピューティング資源（現在では，1GHz 超のマルチコア CPU・数 GB のメインメモリを搭載するものが存在する）を有効活用することができる．

至る所に存在し豊富なセンシング・コンピューティング資源を持つスマートフォンなどの携帯端末を利用し，様々なアプリケーションを実現する取り組みとして，近年 participatory sensing が注目を集めている[2][3][4]．Participatory sensing は，スマートフォンをセンサ情報源として用い，収集した情報を用いてヘルスケアなどのアプリケーションを実現する仕組みである．

Participatory sensing では，スマートフォンユーザが，自分が利用したいアプリケーションを実現するソフトウェア（アプリケーションソフト）を，事前にインストールしておくことが暗に仮定されている．例えば，ヘルスケアの場合，ユーザは自分の脈拍や体温などの情報を定期的にサーバにアップロードするアプリケーションソフトを自分のスマートフォンなどにインストールしておく必要がある．つまり，Participatory

† 株式会社 KDDI 研究所
KDDI R&D Laboratories, Inc.
†† University of California, Berkeley

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

sensing において，情報の発信者（脈拍や体温の情報を送る人）は，アプリケーションの利用者（送った情報を元に医師に健康状態を管理してもらう人）と等しい．

しかし，スマートフォンユーザがアプリケーション利用者にならない場合でも，第三者がそれらのスマートフォンに情報を加工・発信させ，それを利用できるようになれば，さらに新しいアプリケーションが実現できるようになる．その例としては，ある離れた場所（交差点など）の渋滞状況調査が挙げられる．このアプリケーションでは，アプリケーション利用者が渋滞状況を知りたいと思う付近にいる多数の人々がもつスマートフォンに，加速度センサの情報を集めさせる．そして，さらにそれらの情報を連携処理させることで，アプリケーション利用者が自分自身のわずかなコンピューティング資源を使うだけで渋滞状況をリアルタイムに測定する．もう一つの例としては，リアルタイムアンケート調査（市場調査）が考えられる．これは，アンケートを行いたい人が，特定の属性を持つ人のスマートフォンにアンケートを送りこむ．そして，スマートフォンのユーザに画面操作によって様々な情報を入力してもらい，さらにスマートフォン同士が連携してその情報を集計するアプリケーションである．

我々は上記のようなアプリケーションを実現するために，他者のスマートフォンにアプリケーションソフト（プラグインプログラム）を送り込んで実行させ，その実行結果を得るためのシステムアーキテクチャを提案する．提案アーキテクチャでは，何らかのアプリケーションを実現したいと望みプラグインプログラムを送り込む人（すなわちアプリケーション利用者）を依頼者と呼ぶ．また，プラグインプログラムを実際に実行するスマートフォンを保持する人を実行者と呼ぶ（なお本論文では，実行者が持つスマートフォン自身を指して実行者と呼ぶ場合もある）．実行者は，プラグインプログラムを実行することで，自身がもつセンサ情報など様々な情報を入力として，実行者同士で連携して情報処理を行い，依頼者にその実行結果を返答する．提案アーキテクチャはさらに，実行者の属性情報（位置や嗜好など）の管理，依頼者の要求に応じた実行者の選択，実行者へのプラグインプログラムの転送および依頼者への実行結果の中継機能を提供する．また，依頼者から実行者への報酬支払の管理機能を提供する．

提案アーキテクチャのメリットは，依頼者が，所望のタイミングで所望の処理を行うプラグインプログラムを実行者に実行させられることである．これにより，Participatory sensing，あるいは，Twitter などでユーザが自主的に発信した情報を受動的に収集・解析する方法に比べ，より能動的かつ的確に所望の情報を得ることが可能になる．また依頼者は，自分自身のコンピューティング資源に余裕がない場合でも，実行者を連携させることで，実行結果を取得することもできる．さらに，実行者の資源提供に対する依頼者から実行者への報酬の支払いを支援することで，「第三者（依頼者）に資源を提供するモチベーションを持ち得ないスマートフォンユーザに資源を提供させる」ために必要なインセンティブメカニズムを働かせることができる．

本論文では特に，アーキテクチャの提案と Android を搭載したスマートフォンへの実装について報告する．本論文の章構成は以下の通りである．2 章では関連研究について述べる．3 章では提案アーキテクチャについて説明する．4 章ではその実装に向けた指針と Android への実装について述べ，5 章でむすびとする．

2. 関連研究

スマートフォンをはじめとする携帯電話を利用して様々なセンサ情報を取得し利用する考え方は，Participatory sensing [2][3][4]と呼ばれ，近年注目され始めている．Participatory sensing では，スマートフォンが様々なセンサ情報を取得し，それを中央のレポジトリサーバに蓄積する．そして様々なサービス（ヘルスケアや人や車の流動調査など）が，レポジトリから情報を引き出し，加工することで，所望の結果が得られる．

しかし，既存の Participatory sensing には，センサデータ取得・加工・発信のためのアプリケーションソフトを，「どうやって」ユーザがスマートフォンにインストールするのかという点が明確ではない．Participatory sensing では，各ユーザがアプリケーションソフトをスマートフォンに事前にインストールすることが暗に仮定されている．その理由は，既存方式のアプリケーションには，「資源提供者」と「アプリケーション利用者」が本質的には同一であることを仮定したものが多いためである．例えばヘルスケアは，個々人が資源提供者であり，自身のスマートフォンの資源を使って，情報の取得や加工，発信を行い，医者にその情報を託して健康管理を依頼する．つまり，資源提供に対する受益者（＝アプリケーション利用者）は資源提供者本人である．流動調査（人や車などの移動に関する調査）のアプリケーションでは，資源提供者が町の住民であり，そこから得られた情報を行政に託して，都市計画を依頼する．つまり，上記の例と同様に，資源提供者＝アプリケーション利用者といえる．

我々の提案は，携帯端末群を情報源とするという点では Participatory sensing と同様であるが，資源提供者とアプリケーション利用者が全く異なることを許容する仕組みを提供する．

多数のコンピューティング資源を組み合わせて大規模複雑な計算（分散コンピューティング）を行う仕組みは，grid computing をはじめ多数の提案，実施例がある．中でも，個人がコンピューティング資源を提供するという取り組みには，BOINC [5]や Flash-mob computing [6]などが挙げられる．BOINC は SETI@home [7] から発展したプロジェクトであり，ユーザがもつコンピュータの余剰コンピューティング資源を提供し，またそれらを利用するためのオープンプラットフォームを開発している．BOINC のアーキテクチャは，コアサーバが依頼された大規模計算をユニットに分割し，ユニットを資源提供者に送る．資源提供者がそれぞれ独立にユニットを処理し，処理結果

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

を収集するというものである．Flash-mob computing は，多数のコンピュータを持ち寄って，手軽に ad-hoc に super computer を構築することを目指したプロジェクトである．Flash-mob computing では，BOINC のような独立なユニットへの分割を行わず，コンピュータ同士が頻繁に通信を行い，連携して計算を進める手法が用いられている．

我々の目標は，依頼者自身が供出するコンピューティング資源をなるべく小さくしつつ，希望のセンシング・コンピューティングの処理を実現することにある．我々が BOINC などの取り組みに注目したのは，個々人が余剰コンピューティング資源を提供するというモデルによってこれらのプロジェクトが成り立っていることにある．実行者が余剰コンピューティング資源を提供するというモデルは，資源提供が気軽にできるようになれば実現の可能性が高まる．また，これらの分散コンピューティングプロジェクトの成果は，提案アーキテクチャ上でスマートフォン同士の連携にも応用可能である，と我々は考えている．

3. Voluntary Sensing/Computing Architecture

3.1 アーキテクチャ設計指針

我々が提案する Voluntary sensing/computing architecture は次のようなシステムを実現するためのアーキテクチャである．

> ある依頼者が何らかのアプリケーションを実現するために，「どのような属性をもつスマートフォンを実行者としたいか」を指定し，その実行者達にプラグインプログラムを送る．実行者はプラグインプログラムを実行してその実行結果を依頼者に返答する．そして，依頼者は実行者に対して報酬を支払う．

従って，提案アーキテクチャは以下の 3 つの要件を満たす必要がある．

1. 指定された属性を持つ実行者にプラグインプログラムを配送できること
2. 実行者がプラグインプログラムを任意のタイミングで受け付け，安全に実行できること
3. 依頼者から実行者へのプラグインプログラム実行に対する報酬の支払いを処理できること

我々は，これらの要求条件を満たすために，アーキテクチャを図 1 に示すようにプラグインプログラム，依頼機能，資源管理機能，依頼中継機能，履歴管理機能，ホストデーモン，報酬管理機能という 7 つの要素によって構成する．このうち，依頼機能，資源管理機能，依頼中継機能，履歴管理機能は，上記の要件 1 を満たすための要素であり，スマートフォン（またはその保持者）の属性（位置情報や趣味，趣向など）を管理して，問い合わせに応じて適切な実行者を優先順位付きで選択する．また，それらの実行者へのプラグインプログラム送付や実行結果の依頼者への中継を行う．ホストデーモンは要件 2 を満たすための要素で，実行者がプラグインプログラムを受け付け，実行する．そして，報酬管理機能が要件 3 を満たすための要素である．なお，要件 1 と 3 の要素はネットワーク内のサーバに配備して集中型サービスとしてサービス事業者が運用しても良いし，これらの機能も依頼者や実行者といったエンドノードに分散して運用しても良い．

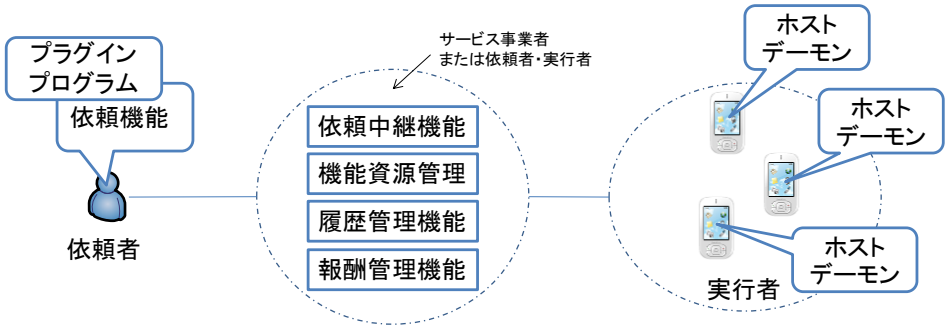

図 1　Voluntary Sensing/Computing Architecture 全体の構成

3.2 アーキテクチャの構成要素

前節で述べたとおり，提案アーキテクチャは，(1)プラグインプログラム，(2)依頼機能，(3)依頼中継機能，(4)資源管理機能，(5)履歴管理機能，(6)ホストデーモン，(7)報酬管理機能によって構成される．図 2 に，これらの要素がどのように用いられるかを，依頼者による 1 回の依頼に関する処理フローを用いて示す．なお，依頼者の 1 回の依頼は，後述の依頼機能で説明する依頼メッセージによって表される．以下では，上記 7 つの構成要素が行うべき処理の詳細を説明する．

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

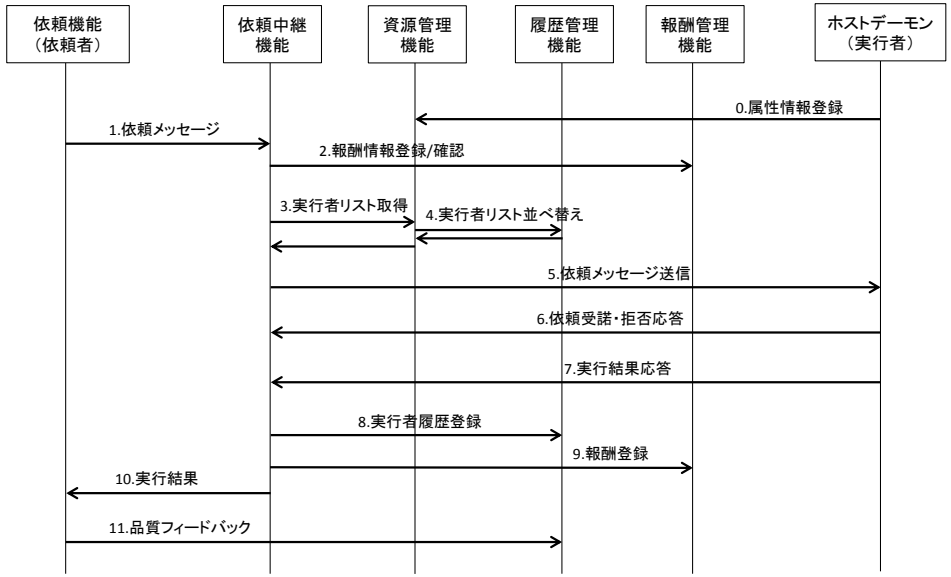

図 2　ある 1 回の依頼の開始から完了までの処理フロー

(1) プラグインプログラム

プラグインプログラムは，依頼者が実施したい処理を記述した以下のような特徴を持つプログラムである．

- プラグインプログラムには，任意の処理を実施するコードを記述できる．
  - ただし，実行者が許可しない機能(例えば GPS 情報の取得など）を利用するコードを記述した場合エラーが返される．
- 複数の実行者間で連携して情報処理を行わせることも，そのようにコードを記述すれば可能である．
  - 既存の分散コンピューティングの方式[5][6]が利用可能．

(2) 依頼機能

依頼機能は，依頼者のコンピュータ上で動作し以下のような処理を行う．

- 依頼メッセージを構築し，依頼中継機能に送信する（図 2 のステップ 1）．なお，依頼メッセージは以下の項目からなる．
  - 依頼文：プラグインプログラムがどのような処理を行うか等の，依頼内容を記述する．これは，スマートフォンユーザがその内容を見て，依頼を受けるかどうかを判断するために用いられる．
  - 実行条件：どのような属性を持つ実行者に処理を依頼したいか，またその人数や結果応答の期限を記述する．
  - 報酬：依頼が遂行されたときに支払う報酬を記述する．
  - プラグインプログラム：実行したいプログラム本体
- 依頼中継機能から実行結果を受け取る（図 2 のステップ 10）．
- 実行結果の品質に対する評価（主観評価）を履歴管理機能に送信する（図 2 のステップ 11）．
  - 例えば-1～+1 までの値で評価する．評価を何も送信しなければデフォルト値(=0)とする．
  - 依頼者が複数の実行者から個別に実行結果を受けた場合，そのそれぞれに対して評価できる．なお，個別実行結果の送信元実行者は依頼中継機能によって匿名化されている（後述）．
  - 依頼者が，集約された実行結果を受け取る場合には，実行者全体に対して一括して評価する．
  - どのように評価を行うか（例えば UI 画面の表示）などはここでは特に規定しない．

(3) 依頼中継機能

依頼中継機能は，ネットワーク上のサーバなどで動作し以下のような処理を行う．

- セッションを作成する（依頼者からの依頼を 1 つのセッションとみなす）．
- 依頼者が報酬管理機能に登録済みかを確認する（図 2 のステップ 2）．
- 依頼内容（実行条件と報酬条件）を報酬管理機能に登録する（図 2 のステップ 2）．
- 依頼者から送られた依頼メッセージを，得られた実行者リスト（図 2 のステップ 3）に従って実行者に転送する（図 2 のステップ 5）．
- 実行者が実行を拒否したかどうか，または実行結果を応答したかをチェックし（図 2 のステップ 6，7），これらの情報を履歴管理機能に登録する．（図 2 のステップ 8）また，実行結果を応答した場合は，報酬管理機能に通知する（図 2 のステップ 9）．
- 実行者から受け取った実行結果を依頼者に中継する（図 2 のステップ 10）．
  - その際，個別の実行者から実行結果が送られる場合には，送信元実行者の ID を匿名化する．
  - 真の ID と匿名化された ID の対応表（匿名ラベルリスト）をセッションに紐づけて管理しておく．

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

(4) 資源管理機能

資源管理機能は，ネットワーク上のサーバなどで動作し以下のような処理を行う．

- 実行者のホストデーモンから送られてくる属性情報（位置や趣味嗜好など）を登録する．
- 依頼メッセージに含まれる実行者条件を元に，実行者リストを取得する（図 2 のステップ 3）．
- 履歴管理機能に実行者リストの並べ替えを依頼する（図 2 のステップ 4）．これによって実行者の優先度を決定する（序列の高い方から実行者として選択される）．

(5) 履歴管理機能

履歴管理機能は，ネットワーク上のサーバなどで動作し以下のような処理を行う．

- 依頼者がこれまでに行った依頼や，実行者の資源提供実績を記録する（図 2 のステップ 8，11）．
- 依頼者の依頼機能からのフィードバックされた実行結果の評価値を実行者ごとに記録する．
  - どの依頼者がどんな評価を与えたかも記録する．
  - どの実行者に対する評価値であるかは，匿名化された ID でラベリングされているので，対応するセッションの匿名ラベルリスト（依頼中継機能が持つ）を参照して，解決する．
- 与えられた実行者リスト（図 2 のステップ 4）を，記録している評価値に基づいて並べ替える．
  - 具体的な評価手法はここでは規定しないが，実行者の被評価値だけでなく，依頼者の評価値も記録しているため，例えば「まともな」評価を行っている依頼者の評価に重み付けする，などの処理が可能である．

(6) ホストデーモン

ホストデーモンは，実行者のスマートフォン上で動作する以下のような処理を行う機能である．なお，バックグラウンドで動作するためデーモンと名付けている．

- 依頼の受け入れ処理や，プラグインプログラム起動，終了，実行結果送信などを行う（図 2 のステップ 5，6，7）．
  - 依頼受信時には，依頼内容の提示と依頼を受諾・拒否の選択を実行者に促す．
- 実行者によるプラグインプログラムの実行に関する様々な設定の受付や，実行者への情報提供を行う．
  - 実行者がプラグインプログラムに利用を許可する機能を設定し，その設定に従って，プラグインプログラムを制御する．
  - 許可できる機能とは，各種センサへのアクセスや，Wi-Fi 等による通信などである．
  - スマートフォン自身が持つプライバシー情報などへのアクセスは常に不許可として制御を行う．
- 現在のプラグインプログラムの稼働状況を表示するとともにプラグインプログラムの強制終了の指示を受け付ける．つまり，実行者は任意のタイミングでプラグインプログラムを強制終了できる．
- 資源管理サービスに対して，属性情報（位置など）を定期的に通知する（図 2 のステップ 0）．
- 実行者による属性情報に関する様々な設定を受け付ける．属性情報の変更があればそのタイミングでも通知する．

(7) 報酬管理機能

報酬管理機能は，ネットワーク上のサーバなどで動作し，依頼者の支払い額と実行者の獲得報酬額の管理や，セッションごとの報酬支払い処理を行う（図 2 のステップ 9）．

4. ホストデーモンとプラグインプログラムの実装

本章では，3 章で提案したアーキテクチャの中でも中心的な役割を果たすホストデーモン（3.1 節の要件 2 の部分）およびプラグインプログラムが満たすべき要件を示す．また，その指針に基づく Android への実装について報告する．

4.1 実装要件

(1) ホストデーモンが満たすべき要件

- 複数のプラグインプログラムを同時に実行できる．
- 実行者が任意のタイミングで実行中のプラグインプログラムを強制終了できる．
- 起動中のプラグインプログラムが画面表示を行える．ただし，どのプラグインプログラムの画面を表示するかは実行者が選択できる．
- プラグインプログラムが，スマートフォン上のプライバシー情報（端末設定や電話帳など）にアクセスすることを防止する．
- プラグインプログラムを他のすべてのアプリケーションプロセスやプラグインプログラムからアイソレートする．
  - すなわち，同一スマートフォン上で実行されている他のアプリケーション

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

プロセス，および他のプラグインプログラムにアクセスする（悪影響を与える）ことを防ぐ．

・ プラグインプログラムが実行者のコンピューティング資源（および通信資源）を不当に大量に消費することを防止する．

(2) **プラグインプログラムが満たすべき要件**

・ プラグインプログラムには，依頼者が行いたい独自処理のみを記述する．
・ プラグインプログラムの記述は一般的なプログラミング言語を用いる．

4.2 Android への実装

本節では，前節で述べた要件を満たす Android 上での実装について説明する．以下，Android アプリケーションに関して，本実装に必要な基礎知識を概説し，次にホストデーモンおよびプラグインプログラムの実装について述べる．

4.2.1 Android アプリケーション開発のための基礎知識

Android アプリケーションは，Android コアライブラリ群を利用して開発される．Android コアライブラリの多くは Java で記述されており，アプリケーションも Java で記述するのが主流である[a]．

Android アプリケーションは通常，Service および Activity と呼ばれる Android コアライブラリのクラスを利用する．Service とは，ユーザへの画面表示を伴わない場合に用いるクラスである．デーモンプログラムのように，バックグラウンド処理を行うアプリケーションは必ずこのクラスを継承したオブジェクトを含んでいる．一方 Activity とは，ユーザへの画面表示に関連する処理を司るクラスである．従って，何らかの画面表示を行うアプリケーションは必ずこのクラスを継承したオブジェクトを含んでいる．なお，1 つの Android アプリケーションが Service，Activity の両方のオブジェクトを持つことも可能である．Android コアライブラリが提供する，例えば各種センサや電話機能へのアクセス，コンタクトリストの管理，システム設定などの機能（本論文では，これを Android 固有機能と呼ぶ）は，Service または Activity のオブジェクトからしか利用できない[b]．

我々は，上に述べた Android プラットフォームの特徴を利用して，ホストデーモンおよびプラグインプログラムを実装する．なお，これらの開発言語は Java とする．

a) 最新の Android プラットフォームでは，NDK を利用してネイティブコード（C 言語など）によるアプリケーション開発が可能である．
b) さらにマニフェストに許可情報を設定しておく必要がある．

4.2.2 ホストデーモン

図 3 にホストデーモンのオブジェクト構成を示す（なお括弧内は継承しているクラス，Thread は生成されるスレッドを表す）．

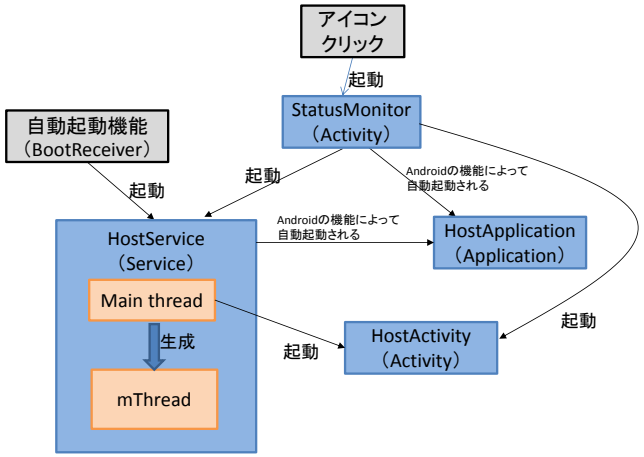

図 3　ホストデーモンの実装

ホストデーモンを構成する主要オブジェクトは，HostService（Service クラスを継承），StatusMonitor（Activity クラスを継承），HostActivity（Activity クラスを継承），HostApplication（Application クラス[c]を継承）である．

HostService は起動直後に新規スレッド(mThread)を生成しバックグラウンドで動作する．そして生成された mThread でプラグインプログラムの待ち受け，起動，終了処理を行う．なお，Main thread とは，1 つのアプリケーション（この場合ホストデーモン）全体で共有される thread であるため，このスレッドで待ち受けを行うのは適切ではない．

StatusMonitor は，ホストデーモン自体の設定やプラグインプログラムの稼働状況表示を行う．図 3 に示すとおり，StatusMonitor はスマートフォンのホーム画面などに置かれたアイコンがクリックされたときに起動し，ホストデーモンの設定画面を表示する．この設定画面上には，「起動」や「停止」などのボタンがあり，HostSerivce の起動や停止が行える．また，3.2 節の(6)で示したように，StatusMonitor は起動中のプラグインプログラムの一覧などを表示する機能や，HostService からの指示を受け，新し

c) Application クラスはひとつのアプリケーション全体が共有することができるオブジェクトを提供するクラスである．

Vol.2011-MBL-60 No.6
Vol.2011-ITS-47 No.6
2011/11/10

くプラグインプログラムを受け付けるときに画面上に通知を表示する機能等を備える．

HostActivity は，プラグインプログラムが画面表示や UI による入力処理を実施するために用いられ，収容しているすべてのプラグインプログラムがこの Activity を共有する．そして，HostActivity が動作中のプラグインプログラムのいずれか一つの画面を表示させる．具体的には，新しいプラグインプログラムが起動した直後にそのプラグインプログラムを，また起動中のプラグインプログラム一覧から選択したプラグインプログラムを画面に表示する．

HostApplication は，ホストデーモンに関する設定情報や，後述のプラグインプログラムに関する情報を保持する．これらの情報は，HostService, HostActivity, StatusMonitor およびすべてのプラグインプログラムからアクセスされる．

4.2.3 プラグインプログラム

プラグインプログラムがホストデーモン上で動作する際のオブジェクト構成を図 4 に示す．

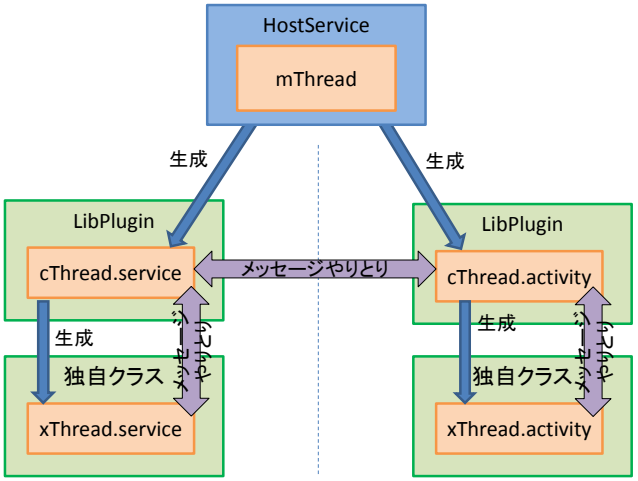

図 4　プラグインプログラムの実装

プラグインプログラムは，3.2 節の(2)で示したような依頼メッセージに格納されており，図 2 のステップ 6 で実行者が依頼を受諾した後に，図 4 のように HostService の mThread から別スレッドとして起動される（cThread および xThread については後述する）．

プラグインプログラムはバックグラウンド処理を行うことも，画面表示（UI 処理）を行うことも可能である．図 4 は，一つのプラグインプログラムを表しているが，左側（cThread.service や xThread.service のスレッドが含まれる方）は Android アプリケーションにおける Serivce に対応する処理を，同図右側は同じく Activity に対応する処理を担当する（図 4 では Service と Activity の両方の特徴を含む例を示しているが，片方だけしか必要ない場合は，もう一方は不要である）．ただし本実装では，プラグインプログラムは Service および Activity クラスを直接継承することができない．このような制約を与えるのは，プラグインプログラムが Android 固有機能，すなわちプライバシー情報（電話帳，Web 閲覧履歴，システム設定情報の閲覧・変更など）や，センサへのアクセスなどを制限するためである（4.1 節の要件(1)）．本実装では，すべての Android 固有機能に対して，ラッパメソッドを用意し，それらのメソッド経由で利用したい機能を呼び出す仕組みをとる．ラッパメソッドでは，まずその機能を利用することを実行者が許可しているかを確認する．なお，どの機能（例えばセンサなど）を使用許可するかは，StatusMonitor の設定画面で指定する．

上述のようなプラグインプログラムの起動やアクセス制限を実現するために，本実装ではプラグインプログラムがホストデーモン上で動作するための基本クラス LibPlugin を用意し，すべての依頼者（プラグイン開発者）にそのクラスがライブラリとして公開される．そして，すべてのプラグインプログラムが必ずこの LibPlugin を継承する．

LibPlugin は，プログラム起動のためのエントリポイント，プラグインプログラムの動作制御（強制終了など），HostActivity との情報のやりとり，Android 固有の機能へのラッパメソッドを提供する．LibPlugin のプログラムコード例を図 5 に示す．

```
class LibPlugin {
   ...
   private Activity act;
   ...
   public startPluginProgram(param) {
   };
    ...
   public getGPSinfo() {
      GPSを利用していいかどうか設定をチェックする;
         (利用不可ならreturn);
      actオブジェクトを利用して、GPS機能にアクセスする;
       return (GPS情報);
   }
   ...
};
```

図 5　基本クラス LibPlugin のコード例

LibPlugin クラスには，プラグインプログラムのエントリポイントとして

startPluginProgram メソッドが宣言されている．これは，HostService (mThread)がプラグインプログラムを起動するために新規スレッド(図 4 の cThread.service や cThread.activity)を生成し，そのスレッドの中で最初に呼ばれるメソッドである．各プラグインプログラムは，LibPlugin を継承した独自クラスを持ち，その中で startPluginProgram メソッドをオーバーライドして，所望の初期化処理を行う．例えばその初期化処理の中で，必要に応じて新たなスレッド(xThread.service など)を生成し，様々な処理を行わせることができる．

LibPlugin クラスは，必要最小限の変数およびメソッドだけを public で宣言することで，Android 固有機能へのアクセス制限およびプラグインプログラムの他からのアイソレート（4.1 節の要件(1)）を実現する．LibPlugin を継承したクラスは public で宣言されたものにしかアクセスできないためこのようなアイソレートが可能になる．例えば，図 5 の getGPSInfo メソッドは，GPS 情報を取得するためのラッパメソッド例であるが，図に示したとおり，本メソッドの前半で GPS 利用が許可されているかを調べる．また，他の Android アプリケーションなどと連携するために必要な Service クラスおよび Activity クラスのメソッドへのラッパメソッドを用意しないことで，他のアプリケーションプロセスへのアクセスを防止できる．同一プラグインプログラム内の Service 部分と Activity 部分が連携する，または HostActivity の機能を利用する（画面表示や入力処理を行う）ための仕組みとして，LibPlugin が連携用のメソッドまたはメッセージハンドラを public で提供する（図 4 の「メッセージのやりとり」の部分）．これらのメソッド，メッセージハンドラは，LibPlugin クラスの startPluginProgram メソッドが呼ばれるよりも前に自動的に作成しておく（HostService がプラグイン起動処理の一環として行う）．

図 4 に示すように，プラグインプログラムは cThread.service や cThread.activity, xThread.service や xThread.activity のスレッドで動作する．そのため，これらのスレッドを mThread（親スレッド）から停止すれば，実行者が所望のタイミングでプラグインプログラムを強制終了できる．また生成される cThread の処理優先度を低くしたり，極端にコンピューティング資源を消費する cThread や xThread に対して mThread が suspend()を呼び出すことによって一時停止したりすることで，単一のプラグインが資源を浪費することを防ぐ．

4.2.4 ホストデーモンと LibPlugin の関係

前節では，プラグインプログラムが利用する基本クラスとして LibPlugin を導入した．本実装では，この LibPlugin の実体はホストデーモンが保持する．つまり，プラグインプログラムには LibPlugin クラスのバイトコードを含める必要がない．さらには，仮に依頼者が不正な改造を行った LibPlugin のバイトコードを付加したプラグインプログラムを送付したとしても，Dalvik VM[d]の仕組み（具体的には，dexclassloader と呼ばれるクラスローダー）により，必ず呼び出し側プロセス（この場合ホストデーモン）が持つクラスを優先的にロードされるため，第三者が不正な処理を行う余地が無くなる（ホストデーモンそのものが別の手段で不正に破壊されない限り）．

5. おわりに

本論文では，Voluntary sensing/computing architecture を提案した．提案アーキテクチャは，依頼者が任意の処理を行うプラグインプログラムを任意のタイミングで実行者に送り込み，それを実行者が実行する仕組みを提供する．さらに本アーキテクチャは，依頼者から実行者への報酬の支払いを可能にする仕組みを提供することで，「第三者（依頼者）に資源を提供するモチベーションを持ち得ないスマートフォンユーザに資源を提供させる」ために必要なインセンティブメカニズムを働かせることができる．本論文では更に，提案アーキテクチャを Android に実装するための要件や実装の概要について述べた．

今後は，プラグインプログラムがホストデーモン上で動作する際のセキュリティについての詳細な調査検討や，プラグインプログラムのための基本クラス LibPlugin がサポートすべきラッパメソッドの精査，提案アーキテクチャ全体を含んだシステムの試作，および本提案を普及させるためのインセンティブメカニズムの検討を行う予定である．

謝辞 日ごろご指導いただく，KDDI 研究所中島所長に深く感謝いたします．

参考文献

[1] http://www.gartner.com/it/page.jsp?id=1622614
[2] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden and J. Reich, "Mobiscopes for Human Spaces," IEEE Pervasive Computing, Vol. 5, issue 2, pp. 20-29, 2007.
[3] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo and R. A. Peterson, "People-centric urban sensing," in Proc. of the 2nd annual international workshop on Wireless internet (WICON), 2006
[4] D. Estrin, "Participatory sensing: applications and architecture," IEEE Internet Computing, Vol. 14, issue 1, pp. 12-14, 2010.
[5] http://boinc.berkeley.edu/
[6] http://www.flashmobcomputing.org/
[7] http://setiathome.berkeley.edu/

d) Android で利用される Java 仮想マシン