参加型センシングにおける協調制御機構の設計

尾崎凌介[†1] 小川亮二[†2] Niwat Thepvilojanapong[†3] Teemu Leppänen[†4]
狐崎直文[†2] 戸辺義人[†2]

多彩なセンサを内蔵するスマートフォンの普及が近年進んできている．これらのセンサを用いてデータを取得し，サーバに送信する参加型センシングの試みが広がりつつある．しかし，参加型センシングでは全体の調停がないため取得データに過不足が生じ，加えてデータの重複は過剰な消費電力につながる．そこで，本研究では参加者全体の省電力化を達成するために，周囲のクライアントの数に応じてデータ取得・送信間隔を動的に変更する参加型センシングにおける協調制御機構 CSSP (Cooperative Sensor-data Stream Protocol)を提案する．本論文は，CSSP のシステム設計と実装について述べる．

A Design of Cooperative Control Mechanism for Participatory Sending

RYOSUKE OZAKI[†1] RYOJI OGAWA[†2] NIWAT THEPVILOJANAPONG[†3]
TEEMU LEPPÄNEN[†4] NAOFUMI KITSUNEZAKI[†2] YOSHITO TOBE[†2]

1. はじめに

近年，普及が急激に進んでいる携帯電話の中でも特に独自 OS を備えたスマートフォンの普及率が高くなっている．スマートフォンは加速度センサ，照度センサ，GPS などの多様なセンサを搭載しており，データを取得することができる．また，スマートフォンは携帯回線や Wi-Fi を利用したインターネット接続と，GPS 機能を利用した位置情報測定により,取得したデータと位置情報の送信が可能である．最近では，ユーザがスマートフォンを用いて内蔵センサによりデータを取得し，それを特定の Web サーバや Web サイトに送信するユーザ参加型センシング[1]という試みが広まってきている．提供された情報は，リアルタイム性と精度が高く，地域に特化したものとなっている．例えば，天気予報のサイトでは，ユーザから提供された情報を反映することで，元の気象データをより正確で直近の粒度の細かいデータとしてフィードバックしている．今後，スマートフォン利用者が増加していくにつれて，ユーザ参加型センシングもより身近なものになっていくと予想される．

ユーザ参加型センシングでは，刻々と変わるデータを定期的に取得することで,そのセンシングの価値は向上する．しかし，ユーザが何度も手動でデータを送信するのは煩雑である．送信アプリケーション自体が自動でデータを取得し定期的に送信すればその問題は解決されるが，そこで新たにバッテリや消費電力の問題が出てくる．スマートフォンユーザにとってスマートフォンのバッテリ継続時間や消費電力は非常に重要なファクターである．定期的にデータを送信する場合，固定された送信間隔ではデータが過不足する場所が出てくる．さらに，データの重複は過剰な消費電力につながる．これはユーザ側にとっても情報収集する側にとっても効率的ではない．そこで，サーバ側がセンシングを行なっている周囲のスマートフォンの数に応じて動的にデータ送信・取得間隔を変更することで，それぞれのスマートフォン同士を間接的に協調させることを可能とする参加型センシングにおける協調制御機構 CSSP(Cooperative Sensor-data Stream Protocol)を提案する．CSSP を用いることによって,効率的にセンシングを行い，データを収集することで，ユーザ 1 人あたりの消費電力を減らし，それによって全体としての省電力化を図ることが目的である．

本論文では第 2 章にて関連研究を述べ,第 3 章にて CSSP の設計,第 4 章にて CSSP の実装について詳しく解説する．第 5 章にて CSSP を用いた評価実験を行い，得られた結果について考察する．第 6 章にて CSSP の今後の予定について言及する．第 7 章にて，全体に対するまとめを行う．

2. 関連研究

・Participatory Sensing

参加型センシング(Participatory Sensing)[1]は, スマートフォンが搭載する多様なセンサを使用して，ユーザ参加型のネットワークを構築し，様々なユーザからセンサを使用して得た情報を収集することが可能である．参加型センシングでは，携帯電話を持ったユーザが移動しながら周囲のセンシングを行うことができるのが特徴であり,特定の場所,

†1 青山学院大学理工学研究科理工学専攻
†2 青山学院大学理工学部情報テクノロジー学科
†3 三重大学工学部情報工学科
†4 Department of Computer Science and Engineering, University of Oulu

広範囲の場所の情報を収集したり，リアルタイム性の高い情報を収集したりすることが可能である．また，携帯電話は日常生活で持ち歩いて行動する場合がほとんどであるため，センシングへの参加や脱退が容易である．

ヒューマンプローブは，車載器や CAN を利用した交通状況などのプローブ情報や，スマートフォンや計測機器を使用して人々の状態・行動を把握するためのセンシング情報など，移動体によるセンシングである．ヒューマンプローブや参加型センシングによって得られた情報は，ダイナミックに変化する人やモノの状態・行動・周辺環境をリアルタイムに把握することができる．また，それらの情報は適切に加工・蓄積され，大気モニタリング[3]，騒音モニタリング[4]，道路モニタリング[5]，ユーザ行動履歴収集[6]，ソーシャルネットワークでの利用[7]など様々な研究が進められており，情報を有効活用することで，都市生活・環境・交通などの諸問題の解決や，日常生活をより豊かにする可能性がある．

・Aquiba

Aquiba(Architecture of Qualitative Urban Information Blending and Acquisition)[2]では，複数のスマートフォン同士での協調に，近距離無線などの携帯通信回線以外の通信手段を用いることで，スマートフォン同士で直接調停をし，センシングおよび送信のレートを調節する．これによって省電力化を図る．

3. CSSP の設計

本章では，サーバとクライアントで用いるセンサデータストリーム協調制御プロトコル CSSP の設計について述べる．

3.1 全体システム

CSSP は，参加型センシングにおいて参加ユーザがセンシングを開始した後，サーバが周辺の参加ユーザの数に応じて動的に送信間隔を制御することで，周辺の参加ユーザと意識することなく協調してデータ収集することができ，消費電力を少なくすることでスマートフォンのバッテリの消耗を防ぐ．協調することで一台一台の時間あたりの消費電力は少なくなり，全体としての省電力化を図ることが目的である．CSSP では，参加するユーザの操作をできる限り簡単なもので手順の少ないものにし，スマートフォン内部での処理を可能な限り少なくすることで消費電力を抑える．CSSP におけるスマートフォン同士の協調の手法を図 1 に示す．協調処理，データの管理はすべてサーバが行うこととなる．スマートフォン同士は，サーバに従うことで，周辺のスマートフォンと協調してセンシングを行なうことになる．

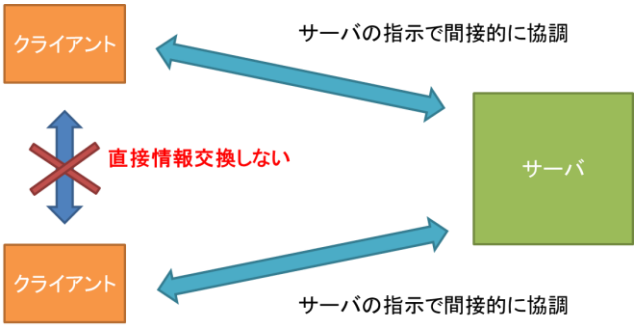

図 1 CSSP における協調の手法

CSSP は，CSSP サーバと CSSP クライアントの 2 つによって構成する．CSSP サーバは，Java により記述された HTTP サーバとして，CSSP クライアントは Android アプリケーションとして実装する．

3.2 CSSP の要件

CSSP で達成すべき要件は，次のように定義される．

・インターネット接続可能な携帯端末を保持し，センシングに参加する CSSP クライアントの数に応じて，CSSP サーバが動的に送信間隔を変更する

・CSSP サーバがすべての CSSP クライアントのセンサデータ，送信間隔を管理し，CSSP クライアントは送信間隔通りに取得したセンサデータを送信する

・CSSP サーバは，自ら直接任意の CSSP クライアントへメッセージを送ることはできず，CSSP クライアントからのメッセージの応答として返す

3.3 プロトコルスタック

図 2 に示すように，CSSP は HTTP 階層の上に実装する．RESTful 設計に基づき，CSSP サーバと CSSP クライアント間でデータを送受信するのに使用するプロトコルとして HTTP を選択する．CSSP メッセージは，HTTP メッセージに含めて送信する．CSSP ではレート制御をアプリケーション層で実行するため，トランスポート層のレート制御と異なり，数十分単位の粒度で制御する．

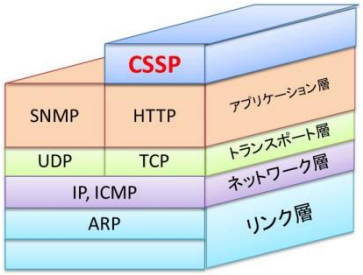

図 2 CSSP のプロトコルスタック

3.4 CSSP での役割定義

以下は CSSP 実装下における役割定義である．

・ユーザ

スマートフォンを保持し，場所の移動・決定，スマートフォンの操作を行う人．

・スマートフォン

CSSP のクライアントの要求に応じて通信機能やセンサ機能などを提供する端末．

・CSSP クライアント

スマートフォンの機能を使用して，各種メッセージの送信やデータ取得･送信間隔の管理を行う Android アプリケーション．

・CSSP サーバ

CSSP クライアントの情報・データを管理し，CSSP クライアントの数に応じてデータ取得・送信間隔を変更し，CSSP クライアントより受信した各種メッセージに対する ACK を送信する HTTP サーバ．

3.5 CSSP サーバの設計

3.5.1 状態遷移図

CSSP サーバは起動すると，CSSP クライアントからのメッセージを受信するため受信待機状態になり，CSSP クライアントからメッセージが届くたび逐次スレッドを作成し，その中で CSSP コードやクライアント ID でそのメッセージの種類を判別し，メッセージに対応した処理を行う．

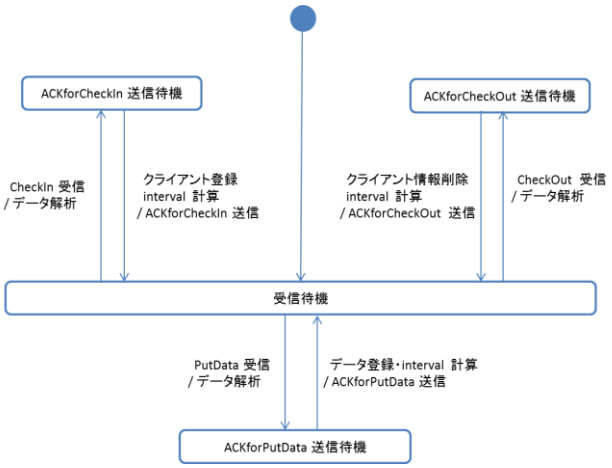

図 3 CSSP サーバの状態遷移図

3.5.2 CSSP コード

CSSP サーバと CSSP クライアント間で受信したメッセージを判別するため，共通のコードを用いる．

3.5.3 メッセージ・メッセージフォーマット

CSSP サーバは 3 種類のメッセージを持ち，CSSP クライアントから受信したメッセージに応じて，適切なメッセージを送信する．メッセージは，ACKforCheckIn，ACKforPutData，ACKforCheckOut である．メッセージは JSON 形式で記述する．

・ACKforCheckIn

CSSP クライアントへ，固有 ID，センサの種類，送信間隔を送信する．

表 1 ACKforCheckIn のメッセージと役割

変数名	データ型	役割
id	int	クライアント固有 ID
sensortype	int	センシングを行うセンサの種類
interval	int	送信間隔（分）
csspcode	int	サーバ-クライアント間の共通コード

・ACKforPutData

ClientManager より CSSP クライアントの送信間隔を調べ，その送信間隔と CSSP サーバの現在の送信間隔が異なる場合は，CSSP サーバの現在の送信間隔を新しい送信間隔として付加して送信する．変更がない場合は，元の送信間隔を送信する．

表 2 ACKforPutData のメッセージと役割

変数名	データ型	役割
csspcode	int	サーバ-クライアント間の共通コード
interval	int	送信間隔（分）

・ACKforCheckOut

CheckOut メッセージを送信した CSSP クライアントの情報を CSSP サーバの ClientManager から削除したことを示す ACK とメッセージを送信する．

表 3 ACKforCheckOut のメッセージと役割

変数名	データ型	役割
csspcode	int	サーバ-クライアント間の共通コード
message	String	登録削除完了メッセージ

3.5.4 データ取得・送信間隔決定アルゴリズム

M：60 分あたりに必要なデータ数，N：参加クライアント数とすると，送信間隔(min)は，次式で決定される．

送信間隔 = (60 / M) * N

3.6 CSSP クライアントの設計

3.6.1 状態遷移図

CSSP クライアントは初めに CheckIn メッセージを CSSP サーバに送信し，ACKforCheckIn メッセージを受信したら，バックグラウンドで動作するサービスを起動し，ACKforCheckIn メッセージに含まれる送信レートに従ってタイマを設定し，スリープ状態に入る．時間が来ると CSSP クライアントはスリープ状態から復帰し，CSSP サーバから指定されたセンサでデータを取得し，センサデータ・取得日時を PutData メッセージに格納して CSSP サーバへ送

信する．CSSP サーバから ACKforPutData を受信したら，送信レート変更の有無を確認し，変更があれば新しい送信レート，変更がなければ元の送信レートでタイマを再設定する．CSSP クライアントはユーザから停止命令を受信するまでセンシングを繰り返し行う．ユーザのタイミングで CSSP クライアントに停止命令を出し，その際 CSSP クライアントはセンシングを終了する CheckOut メッセージを CSSP サーバへ送信する．

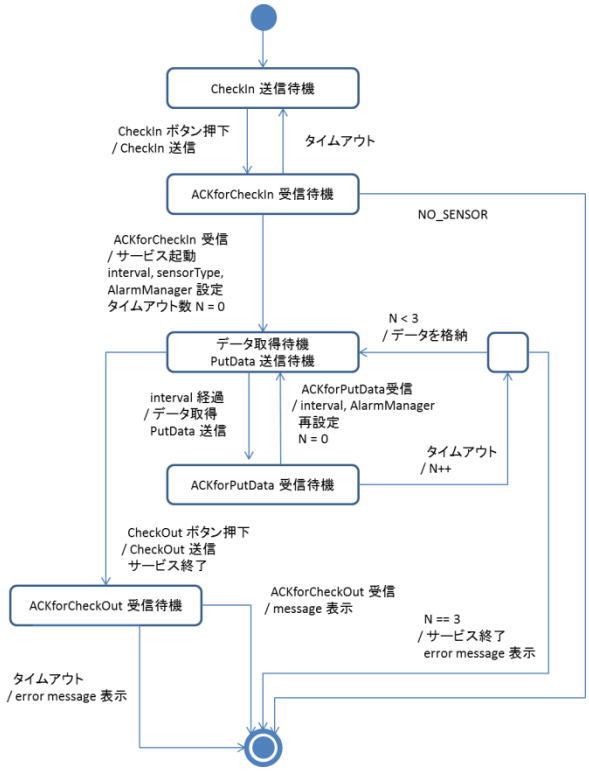

図 4 CSSP クライアントの状態遷移図

3.6.2 メッセージ・メッセージフォーマット

・CheckIn

CSSP サーバへ端末情報を送信する．

表 4 CheckIn のメッセージと役割

変数名	データ型	役割
csspcode	int	サーバ-クライアント間の共通コード
latitude	double	緯度
longitude	double	経度
sensortypes	ArrayList<Integer>	センサ ID リスト

・PutData

CSSP サーバへ測定し取得したセンサデータを送信する．

表 5 PutData のメッセージと役割

変数名	データ型	役割
id	int	固有 ID
date	String	データ取得日
time	String	データ取得時刻
value	double	データ値
pre_times	ArrayList<String>	ACK 不受信時に現在日付時刻格納
pre_values	ArrayList<String>	ACK 不受信時に現在データ格納

・ACKforPutData メッセージが来ない場合

一定時間経過しても CSSP サーバから ACKforPutData がなければ，タイムアウト回数に応じた処理を行う．タイムアウト回数が 3 回未満であれば，pre_times と pre_values に先程取得したデータを格納する．タイムアウト回数が 3 回になった場合，通信不能と見なしセンシングを中止し，サービスを終了する．

・CheckOut

CSSP サーバへセンシングの終了を送信する．

表 3.7 CheckOut のメッセージと役割

変数名	データ型	役割
id	int	固有 ID
csspcode	int	サーバ-クライアント間の共通コード

3.6.3 チェックイン・チェックアウト方式

参加型センシングは，ユーザが容易にセンシングへ参加できる．そのため CSSP ではチェックイン・チェックアウト方式を採用した．ユーザは任意のタイミングでセンシングを開始し，任意のタイミングで終了する．

4. CSSP の実装

4.1 CSSP サーバの実装

CSSP サーバは，Java による HTTP サーバとして実装する．

4.1.1 CSSP サーバの機能

CSSP サーバは以下の機能を持つ．

・CSSP クライアントの情報・データの登録
・メッセージの解析
・メッセージに対する ACK の返信
・送信間隔の更新
・CSSP クライアントの情報削除

4.1.2 CSSP サーバの構成

CSSP サーバは，CSSPserver と ConnectionThread によって構成する．

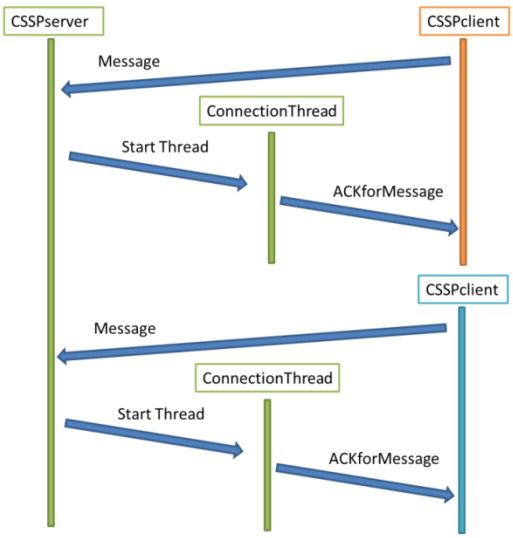

図 5 CSSP クライアントの構成とメッセージ送受

・CSSPserver

CSSPserver は常時受信待機状態であり，ソケットを受信したら逐次 ConnectionThread を作成する．

・ConnectionThread

ConnectionThread は受信したソケットに対して以下の変数と主要メソッドを用いて処理を行う．

表 7 ConnectionThread の変数と役割

変数名	データ型	役割
interval	int	全体のデータ取得・送信間隔（分）
SENSOR_TYPE	int	ClientManager から CSSP クライアントを削除
csspclient	CSSPclient	CSSP クライアント情報を保持
ClientManager	ArrayList <CSSPclient>	すべての CSSPclient の管理

CSSPclient クラスは以下の変数を持ちクライアント情報を保持するクラスである．

表 8 CSSPclient の変数と役割

変数名	データ型	役割
id	int	CSSP クライアントの固有 ID
interval	int	前回 ACK を返した時の送信間隔（分）
latitude	double	CheckIn 時の緯度
longitude	double	CheckIn 時の経度
values	ArrayList<Double>	CSSP クライアントの測定したデータを保存
times	ArrayList<String>	CSSP クライアントがデータを測定した時間を保存

表 9 ConnectionThread の主要メソッド

メソッド名	機能
analyzeMessage	受信したメッセージを解析する
registerClient	ClientManager に新規 CSSP クライアントを登録
removeClient	ClientManager から当該 CSSP クライアントを削除
updateInterval	現在の CSSP クライアントの数に応じて送信間隔を更新
registerData	CSSP クライアントより受信したデータを ClientManager の DataList へ登録
ACKforCheckIn	ACKforCheckIn メッセージを JSON 形式に変換し送信
ACKforPutData	ACKforPutData メッセージを JSON 形式に変換し送信
ACKforCheckOut	ACKforCheckOut メッセージを JSON 形式に変換し送信

4.2 CSSP クライアントの実装

CSSP クライアントは，Android アプリケーションとして実装する．

4.2.1 CSSP クライアントの構成

CSSP クライアントは，CSSPclient と Sensing サービスのプログラムによって構成する．図 6 は CSSP クライアントの構成と CSSP サーバとのメッセージの送受を示す．

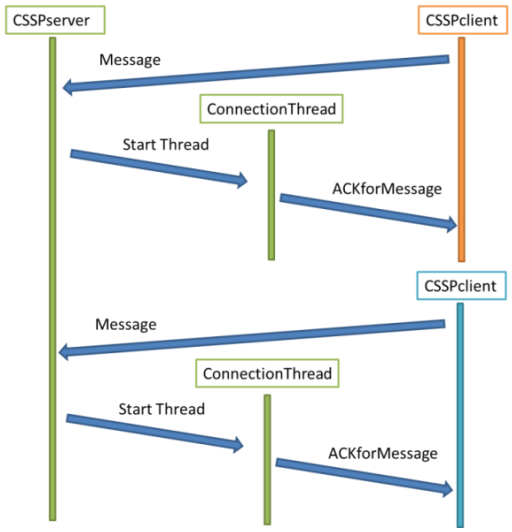

図 6 CSSP クライアントの構成とメッセージ送受

・CSSPclient

CSSPclient は以下の主要メソッドを用いて CSSPserver と通信を行う．

表 10 CSSPclient の主要メソッド

メソッド名	機能
getSensorList	内蔵されているセンサのリストを取得
getLocation	GPS または Wi-Fi を用いて現在位置情報を取得
CheckIn	CheckIn メッセージを JSON 形式に変換し送信
CheckOut	CheckOut メッセージを JSON 形式に変換し送信
startSensingService	SensingService 開始
stopSensingService	SensingService 停止
updateState	SensingService から最新データを取得

・SensingService

SensingService は以下の主要メソッドを用いてデータ・スケジューリングを行い，CSSP サーバと通信を行う．

表 11 SensingService の主要メソッド

メッソド名	機能
startResident	サービスの常駐を開始
execTask	タイマによって定期的に呼び出される処理群
startSensing	指定されたセンサによるセンシング
getSensorData	センサからデータを取得
stopSensing	センシングを終了
PutData	PutData メッセージを JSON 形式に変換し送信
makeNextPlan	AlarmManager を用いて interval に従い次回 execTask を予約
stopResident	AlarmManager を解除し，サービスを終了

4.3 解析

CSSP サーバは CSSP クライアントのバッテリ残量を把握しないため，CSSP クライアントから CSSP サーバへ各種メッセージの中に，スマートフォンのバッテリ残量は含まれていない．したがって CSSP クライアントからセンサデータを送信する際，端末内部に送信日時，バッテリ残量，センサデータ，データ取得・送信間隔を CSV 形式で保存する．センシングを終えた後，CSV ファイルをパソコンに移し，単位時間あたりの消費電力(mAh/s)を計算し，解析を行う．

5. 実験

騒音センシングを行う Android アプリケーション CSSP クライアントを実装した．アプリケーションを用いた騒音センシング実験では，CSSP を使用し，データ取得・送信間隔を動的に変更した場合の各バッテリ消費レートを求め，解析した．送信間隔は 1 分，5 分，10 分，30 分の 4 種類である．各 30 分（1 分×30，5 分×6，10 分×3，30 分×1）計 2 時間，連続でセンシングを行った．

実験は以下の条件の下で行った．

・ユーザはセンシングを開始してから終了するまで，その場所から動かない
・データの送受信には携帯回線を用いる
・60 分間に必要なデータ数を 60 個とする
・参加クライアントを操作するアプリケーションを作成，使用

実験により得られたスマートフォンのバッテリ残量と送信間隔の推移を図 7 に示す．

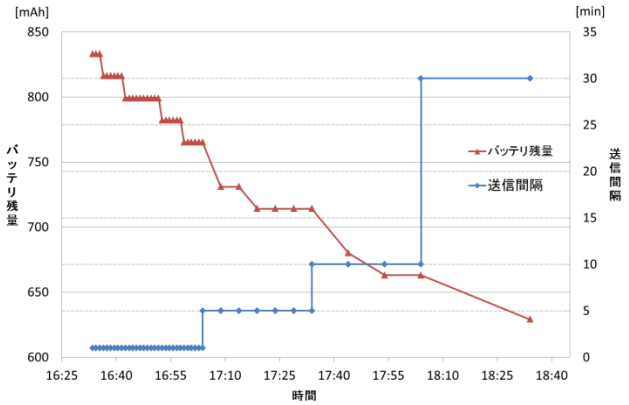

図 7 バッテリ残量と送信間隔の推移

A：バッテリ容量[mAh]，T：データ取得・送信間隔[s]，N : T における取得送信回数，BS : T でのセンシング開始時のバッテリ残量率，BE : T でのセンシング終了時のバッテリ残量率とすると，バッテリ消費量 β [mAh/s]は次式により求められる．

$$\beta = A(BS - BE)/(TN)$$

図 8 は，前式により求めた各送信間隔に対するバッテリ消費レートである．

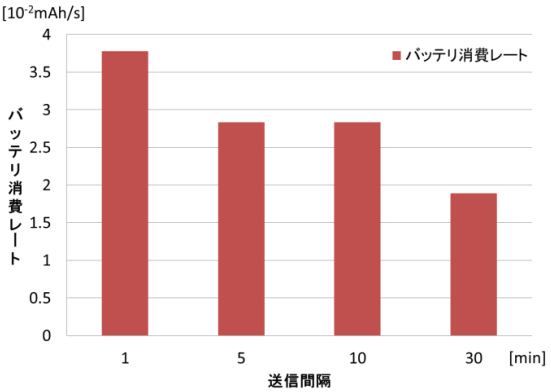

図 8 各送信間隔に対するバッテリ消費レート

実験により，送信間隔が短いほどバッテリ消費レートが高くなり，長いほどバッテリ消費レートが低くなることがわかった．それはサーバが周囲の参加クライアント数に応じて動的に送信間隔を制御した結果である．そして，CSSP を用いることによって参加クライアント同士を協調させ，参加クライアント 1 人あたりのバッテリ消費レートを減少させ，全体としての省電力化を行うことができた．

6. 今後の予定

今回の実験は, CSSP を用いた参加型センシングを行う際，ユーザがチェックインした後，その場所から動かない，という前提で行ったが，スマートフォンを用いたセンシングの特徴は移動性である．そこで今後は，ユーザの移動性(mobility)について考慮していく．

6.1 位置情報の取得

CSSP では移動しないことが前提条件であるため，GPS を用いた位置情報取得は CheckIn する時のみ行った．しかし，移動しながらセンシングを行う場合は，センシングをするごとに位置情報を取得する必要がある．Sensing サービスがバックグラウンドでセンシングを行う際，自動で GPS を起動終了できるようにし，位置情報を取得する仕様に変更する．取得したデータを PutData で送る際に，位置情報も付与する．

6.2 mobility を考慮したセンシング

図 9 のようにセンシングを行っている最中に，ユーザがセンシング範囲を移動する場合を考える．

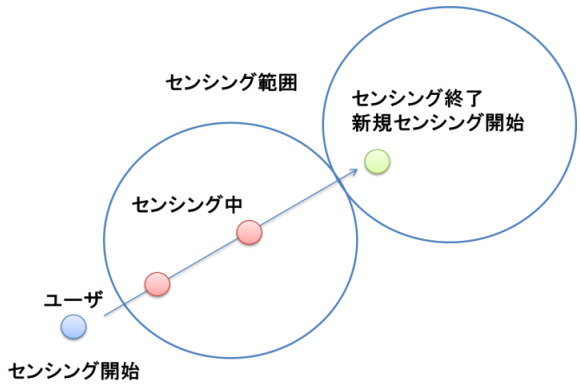

図 9 mobility を考慮したセンシングの構想

ユーザから送られてくる PutData メッセージにはその時の位置情報が含まれている．そこで，サーバは前回までユーザが送信した位置と今回送信した位置情報を元に送信間隔の時間が経過した時の位置情報を推定する．もし，送信間隔の時間が経過した時に，現在のセンシングとは別のセンシング範囲にいると推定される場合は，現在のセンシング範囲と次に行くかもしれないセンシング範囲の両方の送信間隔をユーザに対して送るようにする．

7. むすび

本論文では, 参加型センシングにおける協調制御機構 CSSP について述べた．CSSP を用いることによって，サーバが周囲の参加クライアントの数に応じてデータ取得・送信間隔を動的に変更し，参加クライアント同士を間接的に協調させることで，参加クライアント一人あたりのスマートフォンのバッテリ消費レートの減少，参加クライアント全体として省電力化という知見が得られた．これは，スマートフォンを使用する参加型センシングにおいて問題となる消費電力への不安を減少させることで，参加型センシングへ参加を容易にすることができると期待される．

参考文献

1) Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S. and Srivastava, M.B.: “Participatory sensing”, CENS report, (2006).
2) Thepvilojanapong, N., Konomi S., and Tobe, Y., "A Study of Cooperative Human Probes in Urban Sensing Environments," IEICE Trans. Comm., Vol.E93-B, No.11, pp. 2868-2878, (2010).
3) Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R. and Boda, P.: PEIR: the personal environmental impact report as a platform for participatory sensing systems research, MobiSys, ACM, pp.55–68 (2009).
4) Lu, H., Pan, W., Lane, N., Choudhury, T. and Campbell,A.: SoundSense: Scalable sound sensing for peoplecentric applications on mobile phones, Proc. ACM MobiSys 2009, pp.165–178 (2009).
5) Mohan, P., Padmanabhan, V.N. and Ramjee, R.: Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones, Proc. ACM SenSys 2008, pp.323–336 (2008).
6) Eisenman, S.B., Miluzzo, E., Lane, N.D., Peterson, R.A., Ahn, G.-S. and Campbell, A.T.: The Bikenet Mobile Sensing System for Cyclist Experience Mapping, SenSys, ACM, pp.87–101 (2007).
7) Miluzzo, E., Lane, N.D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S.B., Zheng, X. and Campbell, A.T.: Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application, SenSys, ACM, pp.337–350 (2008).