DEIM Forum 2012 A10-3

WebSocketを用いた大規模センサデータストリームの可視化システムの開発

中根　傑† 江田　政聡† 横山　昌平†† 福田　直樹†† 峰野　博史††
石川　博††

† 静岡大学大学院情報学研究科　〒 432–8011 静岡県浜松市中区城北 3–5–1
†† 静岡大学情報学部情報科学科　〒 432–8011 静岡県浜松市中区城北 3–5–1
E-mail: †{gs10040,gs10010}@s.inf.shizuoka.ac.jp, ††{yokoyama,fukuta,mineno,ishikawa}@inf.shizuoka.ac.jp

あらまし　Web 上で可視化するためのシステムについて提案する．提案システムでは，WebSocket と呼ばれる相互通信用のプロトコルを用いて，センシングデータを受信し，Tiled Display Wall 上で可視化を行う．従来の Web アプリケーションでは，Web 上でストリームデータを受信する際に LongPolling を用いることが多いが，毎回クライアントからサーバへ問い合わせる処理が発生する．本研究では WebSocket の相互通信を用いてクライアントからサーバへの無駄な問い合わせ減らし，サーバから Push 通信を行うことでよりリアルタイム性を高める．それらのセンシングデータを用いて可視化を行うことで，エネルギー消費の無駄や人の行動などが発見可能である．
キーワード　可視化，センシングデータ，WebSocket

Development of the visualization system of a large-scale sensor data stream using websocket

Takashi NAKANE†, Masaaki EDA†, Shohei YOKOYAMA††, Naoki FUKUTA††, Hiroshi MINENO††, and Hiroshi ISHIKAWA††

† Graduate School of Informatics, Shizuoka University　Johoku 3–5–1, Nakaku, Hamamatsu-shi, Shizuoka, 432–8011 Japan
†† Department of Computer Science, Faculty of Informatics, Shizuoka University　Johoku 3–5–1, Nakaku, Hamamatsu-shi, Shizuoka, 432–8011 Japan
E-mail: †{gs10040,gs10010}@s.inf.shizuoka.ac.jp, ††{yokoyama,fukuta,mineno,ishikawa}@inf.shizuoka.ac.jp

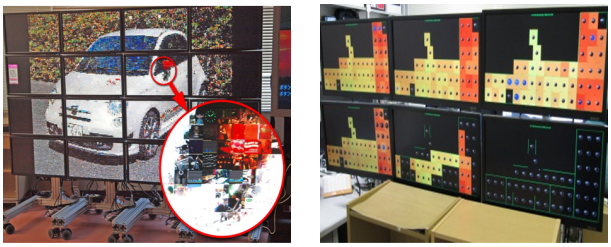
Abstract
Key words　Visualization，Sensingdata，WebSocket

1. はじめに

近年，さまざまなセンサが普及し，家電製品などでも用いられるようになってきた．センサの例は，モーションセンサや照度センサ，扉開閉センサなどがあり，それらのセンサからセンシングデータを取得するための環境として無線センサネットワークの研究が盛んに行われている．無線センサネットワークは，複数のセンサを無線ネットワークに繋げて，センシングデータの取得や，センサ制御などを行う．センシングデータの収集間隔は，目的によって変化し，センサノードの台数が増加すればその分負荷も増加する．従来，これらのセンサデータストリームの処理は，LongPolling 技術や Commet を用いた Push 通信で実現している．Web 技術の発展により，よりデータストリームの処理に適したプラットフォームや規格が整いつつある [1]．本研究では WebSocket の相互通信を用いてクライアントからサーバへの無駄な問い合わせ減らし，サーバから Push 通信を行うことでよりリアルタイム性を高めた可視化システムについて論じる．

2. 関連研究

Tiled Display Wall を利用した研究は様々な分野で行われている [2] [3]．NASA の Hyperwall [2] は横 7 台×縦 7 台の SXGA ディスプレイで構成された Tiled Display Wall である．また，University of California, San Diego の HIPerSpace3 [3] は，よ



左）WDiM1号機　　右）WDiM2号機

図 1　WDiM(Wall Display in Mosaic)

り高解像度なディスプレイを利用する事により総画素数 225 メガピクセルの巨大な Tiled Display Wall を構築している．これらの既存研究と本研究との違いは，Web ブラウザを利用している点である．従来の Tiled Display Wall は、OpenGL の分散レンダリング等高度な処理を行う為に最適化されており，そこで動作するアプリケーションも複雑な実装を必要とした．一方，我々の手法においては，Web ブラウザを利用する事により，一般的な Web コンテンツとして Tiled Display Wall 上のアプリケーションを開発を目指している．そのため，インターネット上のデータやサービスへ容易にアクセスする事が可能である点が最大の特徴である．

また，センサネットワークを利用した研究はさまざまな分野で行われている [6].

本田ら [6] は，赤外線センサの発火系列を用いて，ID なしで複数人を追跡するシステムを構築した．本研究の可視化システムにおいても，単純な数値を表示するだけではなく，これらの研究のデータマイニングなどで得られた新しい知見を可視化する事を目指している．

3.　本システムで用いる Tiled Dispay Wall

本研究が「見える化」基盤として利用するのは，独自に開発した Tiled Display Wall システム，WDiM(Wall Display in Mosaic) [4] である．我々は複数の PC によって駆動される複数の液晶モニタを協調して動作させ，全体として仮想的な高解像度ディスプレイを実現する為のミドルウェアを開発してきた．本研究では大規模なセンサデータストリームをこの WDiM を使って，効果的に表示する．

我々は現在液晶モニタ 16 台構成 (1 号機) および 6 台構成 (2 号機) の 2 セットの WDiM を有する (図 1)．PC と液晶モニタは 1 対 1 で構成されており，PC は Intel Atom プロセッサを搭載する Acre Aspire Revo，液晶モニタは Full HD 解像度 (横 1920 ピクセル×縦 1080 ピクセル) のものを利用している．我々の開発したミドルウェアは JavaScript および PHP で実装されており，また，利用者自身の Tiled Display Wall アプリケーションを実装するための API を有している．各画面を協調して動作させるための処理を行うサーバには Web サーバ (Apache2) を利用し，各画面の駆動は Web ブラウザが担当している．すなわち，WDiM は Web 技術のみを利用して Tiled Display Wall の超高解像度画面を構成している．

また，WDiM はサーバ，コマンダ，レシーバの 3 つのミドル

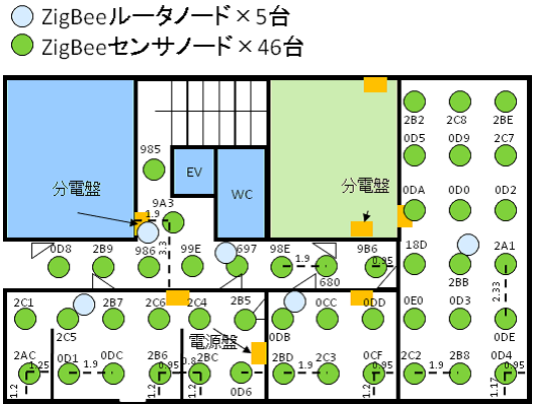


図 2　センサノードの配置図

ウェアから構成されており，それぞれ同期の制御，GUI デザイン，表示デザインの役割を持っている．図 1 左には WDiM ミドルウェアを利用して作成したアプリケーションの一例が示されている．このアプリケーションは写真共有サイト Flickr の画像を利用して，利用者が与えた写真のフォトモザイクをオンデマンドで作成するアプリケーションである．フォトモザイクとは遠くから見ると大きな一枚の画像に見えるが，近くに寄ってみると小さなサムネイル画像が格子状に並んでいるという表現方法である．この例の様に Tiled Display Wall は単に画像を大写しにするのではなく，全体を俯瞰できるとともに，近くに寄って見ると非常に高精細な情報が表示されている事がわかる．写真右は今回の開発に利用した 6 画面からなる WDiM 二号機である．WDiM は画面間の同期に係るネットワークトラフィックを除けば，理論上，任意の画面数に対応する事ができ，またアプリケーションも任意の画面数を扱うよう実装する事が可能である．本研究ではこの WDiM を大規模なセンシングデータストリームの閲覧に適用し，大規模なセンサネットワークに基づいた「見える化」アプリケーションの実現方法を検討する．

4.　センシングデータの収集環境

本研究で利用する無線センサネットワークで使用するセンサボードは，センサボード上に無線モジュール，モーションセンサ，照度センサ，温度センサを有し，乾電池で動作する．このセンサボードは拡張ボードを用いる事で，温湿度センサや臭いセンサなどの様々なセンサを後から追加するする事により，拡張する事が可能である．また，消費電力を計測するための拡張ボードで拡張した消費電力特性計測ノードを使用する事で交流実効電圧，有効電力，力率を計測する事も可能である．本研究で用いるセンサボードは 1 秒ごとに無線通信でシンクノードへデータを送信し，データベースへデータを記録する．そして，センサボードが送信するデータは，センサに割り当てられた ID，時刻，センシングしたデータが含まれており，無線通信の規格は IEEE802.15.4 として規格化されている ZigBee を用いる．センサノードは，天井ノード，消費電力特性計測ノードと在席推定用ノードの 3 種類ある．本システムでは主に天井ノードのセンシングデータを用いている．

センシングデータの収集は，静岡大学の情報学部棟 1 号館 4

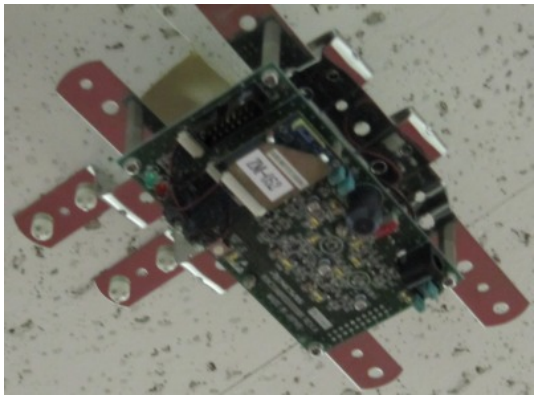


図 3　天井ノード

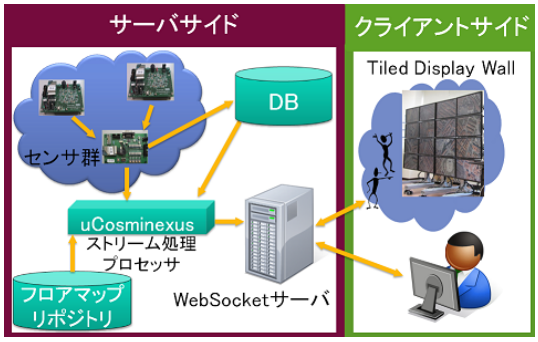


図 4　システム構成図

階にセンサノード 46 台で構築された自律分散協調ユビキタスセンサネットワークを利用する．図 2 に天井ノードの配置場所を示す．図 2 において，描かれている緑色の円が天井ノードである．図 3 に天井ノードを示す．

5.　システム構成

本システムでは，無線センサネットワークから得られたセンシングデータをよりリアルタイム性を保つために，サーバ・クライアント間の通信を WebSocket で行い，Tiled Display 上で可視化する事を試みる．Tiled Display 上に可視化する利点として，無線センサネットワークの規模やセンサデータストリームのデータ量によってディスプレイの台数を増減することが容易で，最適な可視化環境が構築可能である点である．本システムで用いられている言語は Javascript や PHP である．また，WebSocket や Canvas 要素を用いて通信・表示を行っているため，現在一番対応している GoogleChrome を Web ブラウザとして使用している．開発したシステムの構成は図 4 のようになっており，サーバサイドとクライアントサイドに分かれている．このサーバ・クライアント間の通信は WebSocket で行い，WebSocket サーバには Node.JS v0.4.8 [5] を用いている．センサデータストリームの処理エンジンとして uCosminexus Stream Data Platform を用いており，センサデータストリームを WebSocket サーバに送信している．この処理エンジンで行っている事は，各センサノードの 15 分間分のセンシングデータを保持し，新しいセンシングデータが入力された際に，対応するセンサノードのモーションセンサの積算値を活動量として

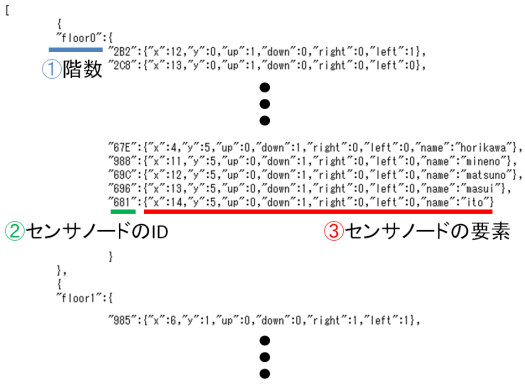


図 5　フロアマップリポジトリの一部

加えて WebSocket サーバへ出力する仕組みになっている．

また，クライアントサイドで用いているライブラリには，YUI2（Yahoo! User Interface Ver.2）[7] や Ext JS 3.0 [8] を用いている．

5.1　DB 構成

本システムではセンシングデータを蓄積するための DBMS として PostgreSQL8.4.1 を用いた．DB に格納・管理される要素として，センシングデータの要素やセンサボードが実際に置かれている場所などがある．センシングデータを蓄積しているテーブルのスキーマは表 1 のようになっている．表 1 では行番号，センサの発火時刻，センサボードの個体番号，照度センサの値，モーションセンサの値，温度センサの値，湿度センサの値，センサボードの電池残量，交流実効電圧，有効電力，力率が格納される．

5.2　フロアマップリポジトリ

地図データは JSON 形式で図 5 のように格納されている．図 5 中の 1 には階数が要素として記述される．2 にはセンサノードの ID が列挙され，その中にセンサノードの要素が記述される．センサノードの要素には，x，y，up，down，right，left，name の 7 つの要素が記述される．x と y はセンサノードの xy 座標を意味する．up，down，right，left はそれぞれセンサノードの上，下，右，左方向に壁や扉が存在するかを表す要素であり，この値が 0 の場合が障害物なしを表し，1 の場合が壁が存在する事を，そして 2 の場合が扉が存在する事を示す数値となっている．また，name 要素はそのセンサノードが在席判定用のセンサノードであるときに存在する要素である．name 要素に記述されるのは，その場所に在席する人の名前となる．フロアマップリポジトリを独立させている利点としては，センサノードの追加や削減に対応する事が容易である事や，可視化させるのに必要なセンサノードの情報のみを取得できる事が挙げられる．

5.3　処理の流れ

システムの処理の流れは，まずクライアントサイドからの接続要求を受信した際に，サーバ・クライアント間でコネクションを確立し，フロアマップリポジトリにアクセスして特定の階に置かれているセンサノードのデータだけを JSON 形式で返す．JSON 形式で返されたデータは Web ページ内で連想配列とし

属性	データ型	索引型	説明
id	integer	primary key	行番号
sensingtime	timestamp		センサが発火した時刻
sensorid	character varying（5）	foreign key	センサボードの個体番号
light	smalllint		光センサの値
motion	smallint		モーションセンサの値
temperature	double precision		温度センサの値
moisture	double precision		湿度センサの値
battery	real		センサボードの電池残量
acvrms	double precision		接続先の交流実効電圧
effectivepower	double precision		接続先の有効電力
powerfactor	double precision		接続先の力率

表 1　sensingdatatable のスキーマ

て格納される．そして，uCosminexus Stream Data Platform と WebSocket サーバを通してセンサデータストリームをクライアントサイドに Push 通信で送信する．クライアントサイドは連想配列に格納されたデータを基に必要なセンシングデータを取捨選択し，可視化を行う．このような仕組みにする事でセンサノードが増えても必要なセンサノードのセンシングデータのみを可視化する事ができ，スケーラブルな可視化を行う事が可能であると思われる．

6.　アプリケーション

アプリケーションの試作として，6 階建ての建物を可視化するアプリケーションの開発を行った．しかし，本実験ではセンシングデータの収集環境が 1 階分となっている．そのため，6 日分のセンシングデータを各階のデータに見立てて，6 階がその日のデータ，階が下がるごとに 1 日前のデータを表示する事とする．また，WDiM では GUI 部分とデザイン表示部分が分けられているが，ここでは主にデザイン表示部分について論じる．

ディスプレイは図 6 のようになっており，青色で囲まれている部分は日付，赤色で囲まれている部分をセルを表しており，また図 7 において，紫色で囲まれているのがセンサノードの ID，赤色で囲まれている部分が温度，青色で囲まれている部分が 15 分間の活動量を表している．セルとは，背景の四角，中心の円，およびそれらの色・大きさを変更する機能，数値を表示する機能のパッケージの事である．

- 中心に表示されている円の色を変更する機能
- 中心に表示されている円のサイズを変更する機能
- セルの背景色を変更する機能

これらの機能を有するセルを，センサの台数分，それらが設置されている場所のフロアマップ上に配置する事により，任意の数のセンサデータストリームを任意の液晶モニタ数の Tiled Display Wall で可視化する事ができる．また，これらセルの機能をアプリケーションのコンテキストに応じて，適切なセンシングデータと関連付ける事ができる．我々はこのセルを用いて，人の行動と室温を「見える化」するというシナリオでアプリケーションを試作した．この試作アプリケーションにおいて，「見える化」するセンシングデータとセルの機能の割り当てを次節より詳述する．

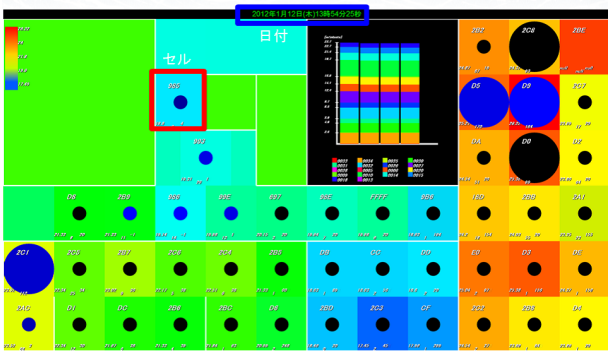


図 6　可視化の例

6.1　中心に表示されている円の色の変更

試作アプリケーションでは円の色を利用して，モーションセンサの値の表示を行っている．モーションセンサの出力は人がいる・いないの 2 値であり，またセンシング間隔は 1 秒である．単純に考えると，センサが反応した時を青，していない時を黒とする事が考えられるが，モーションセンサは人の動きに反応する為，デスクワークなどを行っている場合は必ず毎回反応する訳ではない．そこで，時間軸上直近に反応した時刻から現在までの時間を算出し，0 秒なら青とし，20 秒で黒になるまで色を減衰させる事とした (図 7)．これは在席判定に使うほか，例えば廊下を人が歩くとき等，歩いている方向などを把握する事ができる．

6.2　中心に表示されている円のサイズの変更

モーションセンサは人が居るか，通ったかの判定に利用するほか，そのセンサの探知圏内での人の活動量を見積もる用途でも利用可能である．試作アプリケーションでは，この活動量を円の大きさとして表している．活動量として直近の 15 分間の反応回数の積算値を元に円の大きさを制御している．

図 7 の例では，活動量の多い場所のセル (D5, D9) に比べ，活動量の低い場所のセル (2C7,DA,D0,D2) の方が円の大きさが小さい事が表れている．この機能は人がどれくらい活動しているのかという事の「見える化」を行っている．

6.3　セルの背景色の変更

試作アプリケーションでは，モーションセンサで観測された人の行動と，室温の関係を「見える化」するシナリオで実装し

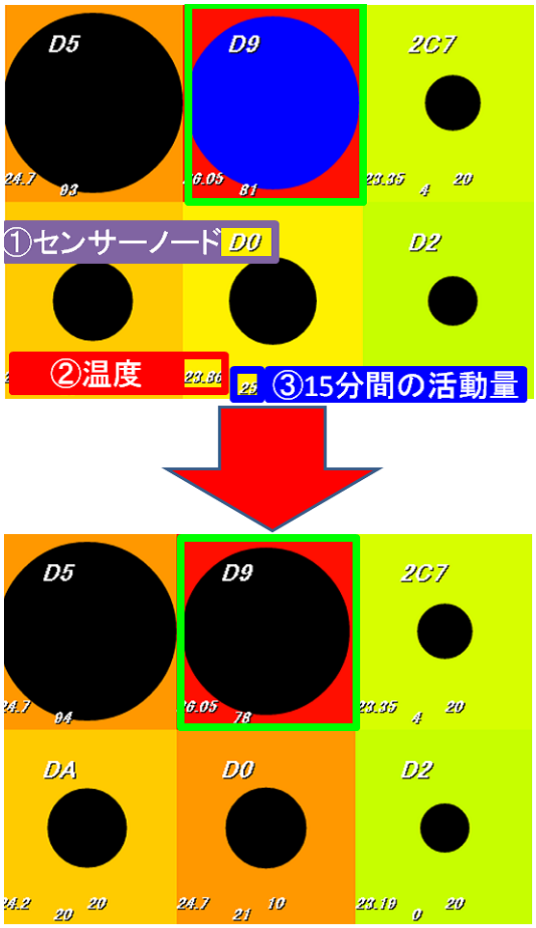


図 7　モーションセンサが反応したときの可視化例

ている．室温は天井ノードに搭載されている温度センサを用いて計測し，このセンサの値を，セルの背景色で表している．背景色は温度が一番高い部分を赤，低くなるに従って青色になるように表示している．また，センサノードがない場所については，隣り合うセルの平均値をその場所の温度として表示している．図 6 に示されているように，このようなフロア全域の温度データを並べて表示すると，室温にもフロア内，部屋内で偏りがある事がわかる．また，詳細なデータの分析をせずとも，人が在席している部分がより温度が高くなっているという関係もみてとれる．これにより例えば，空調に係るリソースを人の活動量の多い部分へ集中させたりという，快適制御へつなげる事ができると考えている．既存の空調機器もセンサを搭載しており，人の行動などを監視し，適切な制御を行っている．しかながら，それらは単にその機器に搭載されたセンサによってその機器単体が制御されているに過ぎない．それに対し無線センサネットワークを利用すれば，フロア，ビルなどの大きな区域全体をセンシングする事ができ，また複数の機器を統合的に操作する事ができるようになる．これにより，エネルギーを必要な箇所へ選択・集中させる事ができ，快適性を維持したまま，消費エネルギーを抑制する事ができるようになる．本研究は，そのような大規模センシングデータの利活用基盤としたい．

センサ ID	レイテンシ差 [sec]
18D	1.76
2A1	1.72
2B2	1.74
2B5	1.67
2B6	1.64
D0	1.74
D1	1.70
全ノード平均	1.71

表 2　レイテンシ計測実験結果の一部

計測 1	計測 2	計測 3	平均 [sec]
0.278	0.115	0.236	0.210

表 3　遅延の実測値

7. WebSocket のリアルタイムの検証

実験として，本システムの WebSocket 版と Ajax 版を用意して，そのレイテンシを計測することでどちらがよりリアルタイム性があるのかを検証する．ここでいうレイテンシとは，無線センサネットワークのシンクノードにセンシングデータが到着してから，そのデータを Web ブラウザで受信するまでの時間のことである．計測時間は 1 時間とし，3 回計測した平均を結果とする．また，3 回の内一度でも電池切れによって計測できなかったセンサノードは省く．3 回とも計測できたセンサノードは４１台あり，その内の一部の結果と全センサノードの平均を表 2 にまとめた．表 2 のセンサノード ID とは各センサノードの個体識別 ID を意味し，レイテンシ差とは Ajax のレイテンシから WebSocket のレイテンシを減算したものである．表 2 より，Ajax 版と WebSocket 版のレイテンシ差は全センサノード平均で 1.71sec あり，WebSocket 版のほうがレイテンシが小さかったため WebSocket のほうがよりリアルタイム性があると判断できる．

実験で利用している無線センサネットワークの発火時間とは無線センサネットワークのシンクノードに到達した時間であるため，実際にセンサノードが発火した時間とは言えない．そこで，実際にセンサノードが発火してから Web ブラウザにそのセンシングデータが到達するまでにどれだけの遅延があるのかを実験した．なお，WebSocket 版のシステムを用いることとする．実験手順を次に記述する．

（1） スクリーン動画キャプチャソフトを用いてブラウザの表示を動画で撮影

（2） ブラウザを表示すると OK ボタンが表示されるので，そのボタンとストップウォッチを同時に押して計測開始

（3） 実際にセンサノードの下まで行き，センサノードの LED が発光したとき発火したことにし，ラップを計測

（4） 3 回発火させたあと，動画をみて発火したと思われる表示とその時の経過時間秒を確認し，正確な遅延の実測値を算出

この手順で実験した結果が表 3 である．平均して，200 ミリ秒のレイテンシでセンシングデータを Web ブラウザで受信可

項目	仕様
OS	Windows 7 Home Premium 32bit　日本語
CPU	Intel(R) Atom(TM) 1.60GHz
メモリ	4.0G(3.0G 使用可能)

表 4　実験に用いる PC のスペック

能であることがわかった．

表 2，表 3 の実験結果により，WebSocket を用いたことにより，Ajax を用いた実装よりも平均 1.71 秒もリアルタイム性を向上させることができ，WebSocket を用いた実装では平均 0.21 秒でセンシングデータ受信することが可能であった．中でも，WebSocket の実測値では，最速約 0.1 秒で Web ブラウザで受信することが可能であった．Ajax での実装が WebSocket よりも，2 秒近くも余分に時間がかかってしまう理由として，クライアント側からサーバ側へ毎回 HTTP 接続する必要があるのと，通信経路の途中で DB が存在していることが挙げられる．前者ではデータをクライアント側で受信するためには，クライアント側とサーバ側の通信だけでも WebSocket に比べると 2 倍の時間が必要であるし，後者はセンシングデータを DB に格納する時間と，その格納したデータを検索し，取得する時間が必要となるため致命的なボトルネックとなっている可能性がある．

8.　システムパフォーマンスの評価実験

ここでは，可視化するセンサノードの台数を増加させた場合のディスプレイ 1 枚で描画できる限界を把握し，Tiled Display Wall を用いることでその限界の台数を超えてもコマ落ちせずに描画できるのかを検証・評価する．ここで用いる PC（Aspire R3610）の仕様は表 4 にまとめる．

実験方法は，Web ブラウザで Web ページを開き，1 分間経過したところで，センシングデータを格納する配列の長さを確認し，1 分の間にその配列がセンサノード台数のよりも大きくなった場合に，描画が追い付かずにコマ落ちしたと判断する．また，実験で用いるセンシングデータは，擬似データを作成し，各センサノードにつき，1 秒に 1 回データを送信することとする．つまり，1 秒にセンサノード台数分の擬似データが uCosminexus Stream Data Platform を通して，Web ブラウザに送信される．

評価指標は，10 回実験した際にコマ落ちしなかった回数の割合とし，これを達成率と呼ぶことにする．予備実験をおこなったところ，Web ブラウザで受信したデータ数が，想定される数よりも極端に少なかったため，まず，uCosminexus Stream Data Platform の受信するデータの個数について述べる．

8.1　uCosminexus Stream Data Platform の評価実験

センサノードの台数を増加させたとき，計測時間 1 分の間に uCosminexus Stream Data Platform が受信しているデータの個数をグラフ化したものが図 8 である．図 8 の縦軸が受信したデータ数で，横軸がセンサノードの台数である．各センサノードにつき，1 秒に 1 回擬似データが uCosminexus Stream Data Platform へ送信されているので，受信データの最大数は 60 である．

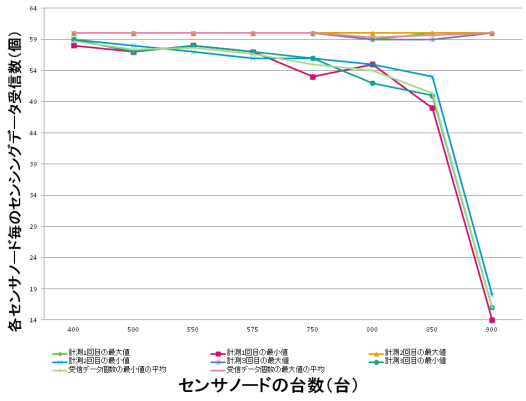


図 8　uCosminexus Stream Data Platform のデータ受信数

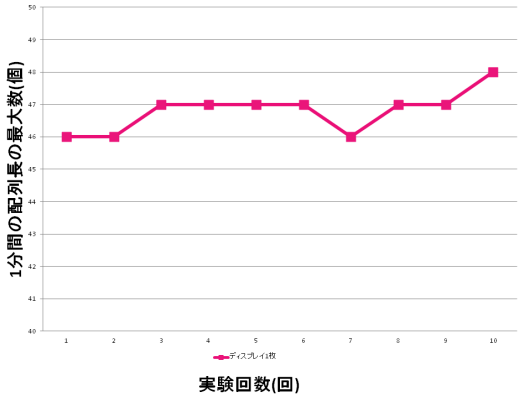


図 9　センサノード数が 50 個の場合の格納配列の最大長

図 8 から，センサノード数が 400 台のときから受信数の最小値が減少してき，900 台のとき急激に受信数の最小値が減少している．しかし，最大値はほとんど減少していないため，特定のセンサノードだけ想定されるデータよりも極端に少なくなっていることがわかる．そこで，netstat コマンドで UDP のパケットエラーを確認したところ，900 台から極端に増加していることが確認できた．そのため，900 台から UDP のパケットロスが大量に発生し，uCosminexus Stream Data Platform で台数分のセンシングデータを受信できなかったと判断できる．より拡張性の高いシステムにするためには，センシングデータを収集する部分も分散させたり，単純に PC のスペックを上げる必要がでてきた．

8.2　画面描画の評価実験

現在の実装では，uCosminexus Stream Data Platform が受信できるデータに制限があることが確認できたため，画面描写の評価をする際には，uCosminexus Stream Data Platform へ擬似データを送信せずに直接 WebSocket サーバに送信することにした．まず，コマ落ちしたかしてないかの判断基準であるセンシングデータ格納配列の最大長がセンサノード台数よりも大きいこととする．配列の最大長がセンサノードの台数よりも大きいことをコマ落ちであるとすることの妥当性を検証するため，実際に運用しているセンサノードの台数と同程度である 50 台の場合の配列の最大長を確認した．図 9 から，コマ落ちしていない時の格納配列の最大長は，センサノードの台数よりも少

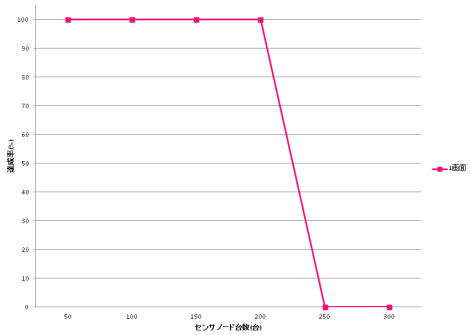


図 10　ディスプレイ 1 枚でセンサノード数が増加させたときの 結果

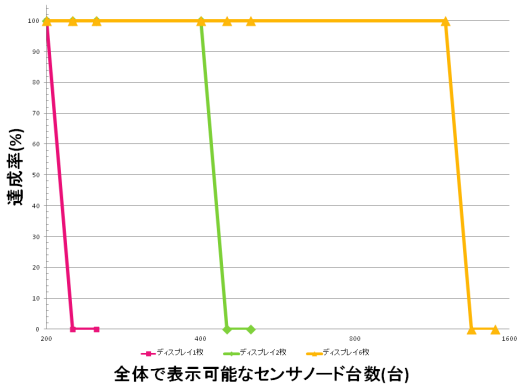


図 11　センサノード数を増加させたときの結果 (ディスプレイ 1 枚，2 枚，6 枚)

ないことが確認できた．

次に，ディスプレイ 1 枚の場合のセンサノードの台数の限界を把握するために，センサノード台数を 50，100，200，250，300 と増加させた場合の実験を行い，その結果が図 10 である．図 10 は横軸がセンサノード台数 (台) をとり，縦軸には達成率 (%) をとる．図 10 より，達成率が 100% であるセンサノード 200 台をディスプレイ 1 枚で描画可能な限界台数とする．

最後に，Tiled Display Wall を用いることで，ディスプレイ 1 枚では描写できなかったセンサノードの数までも，コマ落ちせずに描画できるのかを確認する．1 枚のディスプレイの限界台数が 200 台程度であることが図 10 より確認できたので，実験するセンサノードの台数は 200，225，250，400，450，500，1200，1350，1500 とし，ディスプレイは 1 枚，2 枚と 6 枚の場合をそれぞれ実験する．200 台よりも少ない台数や 250 台よりも大きな台数は 10 の実験結果の傾向から，実験を省き，その代わりに新しく 225 台を加えて実験を行うこととする．結果をグラフ化したものが図 11 である．図 11 の横軸にはセンサノードの台数をとり，縦軸には達成率（%）をとる．この図のピンク色線は，ディスプレイ 1 台にセンサノードの台数分のセンサノードを表示したときの結果であり，黄緑色線はディスプレイ 2 枚にセンサノードの台数分のセンサノードを表示したときの結果，黄色線はディスプレイ 6 枚にセンサノード台数分の台数分のセンサノードを表示したときの結果である．なお，このときの達成率はディスプレイが複数枚あることを加味し，1 枚のディスプレイでも最大配列長がセンサノードの台数を超えた場合は，コマ落ちしたこととし，ディスプレイ 6 枚とディスプレイ 2 枚の達成率とする．図 11 の結果から，1 枚ではコマ落ちをして描画することができなかったセンサノードの台数を，Tiled Display Wall を用いてディスプレイの台数を増やして，分散させて表示することで描画可能となったことがわかる．また，図 11 より，ディスプレイが 1 枚のときの限界台数は 200 台，2 枚ののきの限界台数は 400 台，6 枚のときの 1200 台となっており，ディスプレイの枚数に限界台数が比例している事が確認でき，1 枚当たりの限界台数は同じであることが確認できた．つまり，Tiled Display Wall を用いた場合の限界台数は，1 枚あたり 200 台であり，総限界台数は 1 枚当たりの限界台数にディスプレイの総数を乗算すれば良いことがわかる．また，限界台数を超えた台数のセンサノードを表示したいときには，単純に，Tiled Display Wall で使用するディスプレイを増やしていけば，表示することが可能な拡張性を持つことがわかった．次に，急激に達成率が減少している結果についての考察を述べる．uCosminexus Stream Data Platform の結果を振り返ってみると，ちょうどセンサノード台数が 900 を超えるとパケットロスが極端に増加してしまうことがあった．uCosminexus Stream Data Platform の場合は UDP での通信のため，パケットロスしてしまうと，受信するデータの量が減少してしまうことになるが，WebSocket の通信は TCP で行っているため，パケットロスした場合は再送される．つまり，パケットがうまく送受信できなかった場合，同じデータが再送されるので，Web ブラウザ側では通常受信するデータに加えて再送されたデータも処理する必要がでてくる．そもそも，1 枚のディスプレイでは表示できなかった 225 台を大きく上回る 1200 台を Tiled Display Wall で描写することが可能であったため，本実験でパケットロスしてしまう原因として，データ量が多すぎたためにブラウザ側でうまく処理しきれなかったことが挙げられる．うまくデータを処理できなかったためにパケットロスをした状態で，同数のセンシングデータに再送分のセンシングデータを加えたセンシングデータを受信しようとするために，センシングデータ格納配列の最大長が急激に増加することで達成率が急激に減少する結果になったと考えられる．また，実際にパケットロスをしていることを確認するために，netstat コマンドを用いたところ，1 秒当たりのパケット再送数が 1 0 程度増加しつづけているのが確認できた．

9.　システム全体の考察

本章では，実験結果からシステム全体のリアルタイム性の検証と拡張性の分析・考察を行う．リアルタイム性の検証では，表 2，表 3 の実験結果により，WebSocket を用いたことにより，Ajax を用いた実装よりも平均 1.71 秒もリアルタイム性を向上させることができ，WebSocket を用いた実装では平均 0.21 秒でセンシングデータ受信することが可能であることがわかった．中でも，WebSocket の実測値では，最速約 0.1 秒で Web ブラウザで受信することが可能であった．これは，WebSocket 版で

は，経路途中にボトルネックになるであろう DBMS が存在せず，uCosminexus Stream Data Platform を用いてデータ集約を行っていることに加えて，Ajax 版のように毎回クライアント側からサーバ側に対して通信を行う必要がなく，サーバ側からの Push 通信でデータを取得することが可能であることが挙げられる．

また，拡張性の検証では，本実験での全体の性能評価として，1 枚のディスプレイで閲覧する場合には，最大 200 台で複数のディスプレイで閲覧する場合には最大で 850 台となった．これには，uCosminexus Stream Data Platform 部分での原因と画面描画部分の原因が別々に存在していることがわかった．uCosminexus Stream Data Platform 部分では UDP によるパケットロスによって，データが消失することが原因となり，画面描画部分では，PC 側で少しでも処理が間に合わなかった場合にパケットロスすることで，データが再送され，PC により負荷がかかってしまう事が原因であった．また，前者は 850 台程度までは最低でも各センサノードにつき 9 割近く保障できるが，900 台にもなるとほとんどセンシングデータを受信できなくなってしまうセンサノードが出現してしまうことがわかった．後者では，図 8 から，1 枚当たりの限界台数は 200 台であり，システム全体での限界台数は Tiled Display Wall に用いるディスプレイ枚数に比例していることがわかった．この結果から，近い将来に，無線センサネットワークで収集されるセンシングデータが爆発的に増加したとしても，Tiled Display Wall で用いるディスプレイの枚数を単純に増やすことで対処可能であるといえ，拡張性も十分にあるといえる．ただし，このとき，センシングデータを WebSocket サーバに送信する段階である uCosminexus Stream Data Platform へのパケットロスが確認でき，システム全体で拡張性を向上させるためにはセンシングデータ収集環境をうまく分散させる必要性があることがわかった．

10. おわりに

WebSocket の相互通信を用いてクライアントからサーバへの無駄な問い合わせ減らし，サーバから Push 通信を行うことでよりリアルタイム性を高め，Tiled Display Wall を用いて拡張性も考慮した大規模なセンシングデータの可視化システムを開発した．またそのシステムのリアルタイム性と拡張性について評価・考察した．

本システムではモーションセンサの値によって，セルの中心に表示されている円の色の濃淡を変更したり，円のサイズを変更する事によって人の流れや活動量が把握可能となった．これにより，センシングデータをより人に見やすい形で可視化可能となった．また，セルの背景色を温度に対応した色に対応させる事が可能となり，遠距離から見ると温度の傾向が把握でき，近距離から見るとそれに加えて詳細な数値を閲覧できるので温度などを人の見やすい形で可視化する事が可能となった．本システムでは，WebSocket を用いてデータの送受信を行っており，従来の Ajax 版とのレイテンシ差を計測することで WebSocket の優位性を確認し，WebSocket のリアルタイム性を検証・確認した．また，本システムのディスプレイ 1 枚に対するセンサノードの描画限界台数を確認し，Tiled Display Wall を用いることで，その限界台数を超えたセンサノードを描画可能であることを確認した．Tiled Display Wall 全体での限界台数はディスプレイ 1 枚当たり 200 台で，ディスプレイの枚数に比例した数である 1200 台まで可能であることを確認した．そして，Tiled Display Wall 全体の描画限界が，ディスプレイの枚数に比例していることから，近い将来に，無線センサネットワークで収集されるセンシングデータが爆発的に増加した場合に，Tiled Display Wall で用いるディスプレイの枚数を単純に増やすことで対処可能であるという拡張性を検証・確認した．それに伴い，ユビキタスネットワーク社会を形成するための技術が年々進歩していく中，我々の周りのありとあらゆる場所に無線センサネットワークが張り巡らされ，すでに一般的にセンサを搭載しているもの（例えば，携帯電話や PC のセンサなど）や，あまり搭載していないもの（例えば，靴やメガネなど）にいたるまでセンサが搭載されたとしても，本システムの Tiled Display Wall のディスプレイ台数を増加することで可視化することが可能であると思われる．

今後の課題今後の課題として，本システムを無線センサネットワークの可視化のために用いるのではなく，他のデータの可視化にも容易に用いることを可能とするためにミドルウェア化を行うことを検討している．そのために，画像の表示を初めとする様々な表示機能の追加を検討している．そのようにすることで，システムを使うユーザが任意のデータソースや表示方法を指定可能にすることで，よりユーザの意図にあった可視化が行えると考えられる．

謝辞 本研究の一部は科学研究費補助金挑戦的萌芽研究（課題番号 22650012），基盤研究（B）（課題番号 20300027）および地域イノベーション戦略支援プログラム（旧知的クラスター創生事業第 II 期）「自律分散協調ユビキタスネットワーク」の助成による．

文　献

[1] I. Fette and A. Melnikov “The WebSocket protcol”,IETF HyBi Working Group. 2011.
[2] Sandstrom, T.A., Henze, C. and Levit, C “The hyperwall”,Proc. of International Conference on Coordinated and Multiple Views in Exploratory Visualization, pp.124-133 2003.
[3] DeFanti, T., Leigh, J., R., L., Jeong, B., Verlo, A., Long, L., Brown, M.,Sandin, D., Vishwanath, V., Liu, Q., Katz, M., Papadopoulos, P., Keefe, J., Hidley, G., Dawe, G., Kaufman, I., Glogowski, B., Doerr, K., Singh, R.,Girado, J., Schulze, J., F., K., and Smarr, L.“The OptIPortal, a Scalable Visualization, Storage, and Computing Interface Device for the OptiPuter”, Future Generation Computer Systems, The International Journal of Grid Computing and eScience, Vol.25, No.2, pp.114-123 2009.
[4] Wall Display in Mosaic, http://shohei.yokoyama.ac/Wall_Display_in_Mosaic
[5] Node.js, http://nodejs.org/
[6] 本田 誠一，福井 健一，森山 甲一，栗原 聡，沼尾 正行,「赤外線センサネットワークによる人物追跡」，JSAI2006，2006.
[7] Yahoo! User Interface Ver.3, http://developer.yahoo.com/yui/
[8] EXT JS 3.0, http://extjs.co.jp/