社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2007－ＱＡＩ－25（８）
2007／10／26

センサーネットワークに於けるサーバの負荷容量を考慮したデータ集約の実験と提案

杉山　哲弘†　落合　秀也†　江崎　浩†

† 東京大学情報理工系研究科

あらまし　センサーノードを大量にかつ広域に配置したシステムにおいては，各ノードからサーバへの大量のアクセスが発生する．センサーからのデータがある間隔で観測される場合は，それらをまとめて送ることにより，コストが下がりパフォーマンスが向上する事が考えられる．一方，この場合はセンサーの管理者にとってのデータの即時性を考慮する必要がある．データの集約によるサーバの最大サービス容量の増加の特性を知るために，Live E!のシステムを用いて効果を定量的に計測する実験を行った．本研究ではこの結果を考慮してデータの即時性を考慮しつつ効率的なデータ転送を実現する事を提案する．

キーワード　センサーネットワーク データ集約 負荷軽減

merged data upload in sensor netowrk for lower server load

Akihiro SUGIYAMA†, Hideya OCHIAI†, and Hiroshi ESAKI†

† Graduate School of Information Science and Technology, The University of Tokyo

Abstract　The server which receives data from a lot of sensors in large scale, has big load. Though we have to consider that the real time data is needed in some case, merged data from sensors will reduce the server load. In this research we had an experiment to know how the merged data upload increase the server's service capability. We propose the efficient data transfer considering the usability of data.

Key words　sensor networ, data merge, lower server load

1. はじめに

ビル管理システム，気象データの観測に於いては，センサーは観測したデータを，システムのサーバにアップロードする状況がある．特に，本研究では，センサーはインターネットへの接続性を持ち，定期的に観測したデータをサーバに送信しているものと想定する．観測されるデータは，温度や雨量など，ある間隔で観測されるもので，サイズは大きくない．

この観測されたデータをある程度まとめて送ることにより，コストが下がり，パフォーマンスが向上するようなケースに於いては，まとめて送る事が，有益と考えられる．

ただし，この場合，送信データがノードのローカルなストレージに一定期間留まる事になり，遅延に対してクリティカルなデータに対しては，適用が難しい．

そこで，本研究では，

（1）　基本的には，データ集約による送信を行う事

（2）　ユーザの要求により最大遅延時間を設定可能にする事

を行うアルゴリズムを提案する．本提案が有効に活用されるケースとして，以下の場合が考えられる．

Simple Object Access Protocol (SOAP) は異なる環境間でオブジェクトの呼び出しが可能な利便性の高いプロトコルである．一方テキストデータである Extensible Markup Language(XML) がアプリケーション間の共通フォーマットとして使用されているため，オーバーヘッドが大きくサーバアプリケーションの負荷が大きい [2]～[4]．汎用的なアプリケーション開発のために SOAP を用いていて，その上でサーバの利用効率を向上させたい場合には，データの集約によって SOAP のオーバーヘッドを軽減する事が考えられる．

Wireless Sensor Network では，無線インターフェースを持った複数のセンサーが相互に接続されている．Wireless Sensor Network に於いて，ノードの消費電力等を考慮した，データのアグリゲーションが提唱されている [5]．データのアグリゲーションにより，電力利用効率の高いネットワークを構築し，サーバの負荷も軽減する事が考えられるが，細かいデータの保存が不可能になる．

Delay Tolerant Network は通信路の誤りが多く，性能の悪いネットワークでも，通信を可能とする提案である [6]．データが転送できない場合は，一旦データを保存し，転送相手が近付いてからデータの転送を行う．

本提案の関連研究として，トランスポートレイヤのアルゴリズ

ムである nagle アルゴリズムがある [7], [8]. telnet 等のアプリケーションにより通信が行われている場合, ある 1byte の入力が行われて, それが即座に遠隔のホストに転送されれば,20byte の IP ヘッダ,TCP ヘッダと合わせて 41byte のパケットが転送される事になる.nagle アルゴリズムでは, この現象によるネットワークの輻輳を回避するために,

- 全ての送信済みデータが確認応答されている場合
- 最大セグメント長のデータを送信できる場合

という条件にあてはまらない時にはデータの送信を待つ. これにより, データが少ない場合には送信が遅れるので, ネットワークの利用効率が高まる. また, 確認応答が速く到達する場合には, データの転送は速く行われる.

一方,X Window Server をアプリケーションとして起動している場合は,nagle アルゴリズムの機能はオフにされる必要がある. なぜならば, マウスの動きのような小さいメッセージの転送をユーザのフィードバックのために遅延なしに行う必要があるからである.

以上のように, ある程度の遅延時間を許容してデータを集約すれば, ネットワークの利用効率や, データアップロードを受けるサーバの利用効率が向上することが考えられる. トランスポート層を対象としている nagle アルゴリズムに対し, 本提案ではアプリケーション層でのデータ集約を対象としている. また, データの種類・性質の違いを考慮して, ユーザによる設定の変更を可能とする.

本研究では, データの集約に対して, サーバの利用効率がどれだけ向上するかを, 定量的に把握したいと考えた. そのために行われた実験の内容・結果は章 3. で述べられている. 実験は,Live E! [1] の最新システムを用いた.

2. 即時性を考慮したデータ集約アルゴリズム

2.1 本アルゴリズムの対象とするシステム

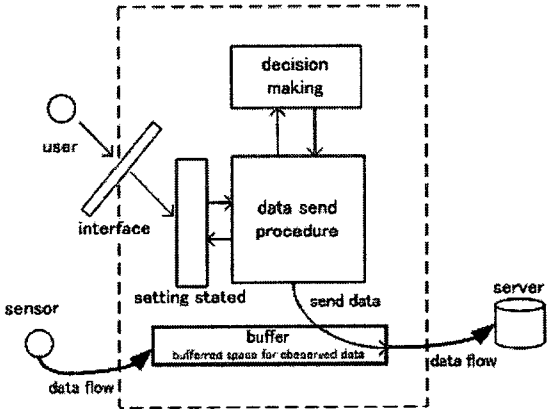

図 1 観測されたデータの流れ

観測されたデータをある程度まとめて送ることによりコストが下がり, パフォーマンスが向上するような場合がある.

5 秒毎に温度を観測しているセンサーを考える. このセンサーがインターネットにつながって, 観測したデータをサーバに送る場合に, データの取得毎にサーバと通信するのでなく,1 分毎に通信を行うようにすれば,TCP のセッション数は 12 分の 1 になり, サーバの負荷が下がる事が考えられる. 同様に考えれば,1 時間毎,1 日毎,1 週間毎と, データ送信の間隔を広げれば広げる程, 通信効率は向上する事が考えられるが, 送信されるデータの価値は下がる. センサーが温度でなく, 地震の揺れを計測している場合は数分後のデータ送信でさえ, そのデータの価値を失わせるかもしれない. 通信効率向上のためのデータ集約は, データを取得できる時間とのトレードオフとなる.

2.2 観測されるデータの性質を考慮した設定

そこで本アルゴリズムでは, 観測したデータの性質を考慮して, ユーザが手動でデータ送信の最大遅延時間を設定できる事とする. 図 1 のうち,user,interface の部分がそれに当たる.

ここで設定される項目は

- delay_threshold : データの送信についてどれだけの遅延を許すか
- size_threshold : 集約すべきデータの適切な数

の 2 つである. delay_threshold は完全にセンサー管理者の意志により決定される. すなわち, 該当センサーによって観測されているデータが即時性を要求するものであれば小さい値が, そうでなければ大きい値が設定される. 集約されるデータの数に最適な数があれば size_threshold で設定される. 章 3., 章 4. で述べるように, 集約されればされるだけサーバの負荷が下がるわけでなく, データの集約には限界が存在する.

```
M := the set of the all stored data in the sensor

initialize delay_threshold by setting file
initialize size_threshold by setting file
While true
   wait for a while
   If Determin-Data-Trans(delay_threshold,
                          size_threshold) is true then
      Send M to the server
      Delete all elements from M
   EndIf
EndWhile
```

図 2 データの送信

2.3 データの流れ

図 1 が本アルゴリズムに於けるデータの流れである.

本アルゴリズムでは, データの送信を主に対象にしている. データの送信は, 図 2 に示すように行われている. 図 1 のうち,data send procedure がこれに当たる. データ送信の判断が真であれば, 観測されたデータの集合 M をサーバに送信し, 送信された M の要素は削除される.M の要素は図 3 に示すように

別のプロセスで追加されている.

```
d := observed data in the sensor
While true
   add d to M
EndWhile
```

図 3 観測されたデータの取得

2.4 データ送信の判断

データ送信の判断は図 4 のように行われる. 図 1 のうち,decision making の部分に当たる.

この判断の真は溜ったデータをサーバに送信することを示し, 偽は送信しない事を示す. 最後のデータアップロードから 2.2 で述べた delay_threshold 以上の時間が経過している場合, あるいは, 十分なデータの数が確保されている場合, 真を返す. そうでなければ, データはまだ溜められるべきと判断し, 偽が返る.

```
Determin-Data-Trans(delay_threshold, size_threshold)
   If current_time - last_sent_time
                        > delay_threshold then
      Return true
   ElseIf the number of M > size_threshold then
      Return true
   EndIf
   Return false
```

図 4 データの送信判断

3. データ集約によるサーバ負荷軽減の検証

データ集約によるサーバ負荷の軽減を検証するために以下に述べるような実験を行った.

3.1 実験の環境

3.1.1 サーバ

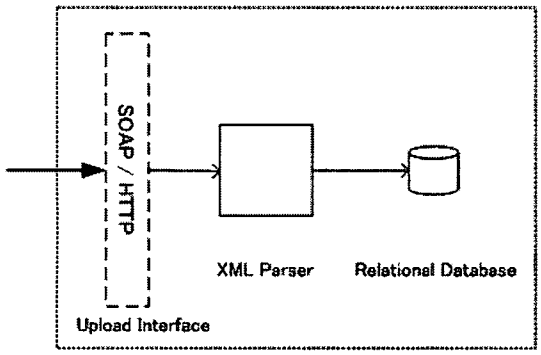

図 5 サーバの構成

章 1. で述べたように, 本実験ではデータのアップロードシステムとして, Live E! [1] のソフトウェアを用いた.Live E!のソフトウェアのうちデータのアップロード部分は, 図 5 のように構成されている.

XML で表現されたアップロードデータは,SOAP/HTTP を用いて送信される. ,SOAP の実装は axis である. アップロードされた XML データは java で解析され,postgresql で構築されたデータベースに保存される.

サーバのマシンは以下のスペックのものを用いた.

- OS:Linux 2.6.20-16
- CPU:Intel(R) Pentium(R) 4 CPU 2.80GHz
- Memory:1GB

3.1.2 クライアント

データを送信するクライアントとして以下のマシンを用意した.

- OS:Linux 2.6.20-16
- CPU:Intel(R) Pentium(R) D CPU 2.80GHz
- Memory:2GB

このマシンから多数のデータアップロードを実行し,1 秒当たりのデータアップロードセンサー数に対する, 応答時間を計測した.

3.1.3 ネットワーク

サーバの負荷に対する計測を行うために, ネットワークは可能な限り理想的な状況を用意した.RTT の平均は 0.100ms, 帯域は iperf での計測で 900Mbps であった.

3.2 実験方法

サーバにデータをアップロードする単位時間当たりの回数を徐々に増やすと, サーバの処理時間が急激に大きくなる時点がある. この処理時間が急激に大きくなるデータアップロード数をサーバの最大サービス負荷容量となる点とし, データ集約数の変化によって, この点がどう変化するかを計測した.

3.2.1 サービス負荷容量計測手順

サービス負荷容量を求めるクライアントノードの動作は, 以下である.

(1) データをアップロードするスレッドを m 個用意し, サーバに対し大量のアップロードを実行する. サービス負荷容量の限界に十分なだけのアップロードを行う.

(2) アップロードを終了したスレッドは s 秒間スリープする.

(3) データアップロードを行うスレッドを徐々に減らしていきながら, データアップロード, スリープを繰り返す.

(4) 全てのスレッドが終了したら, 単位時間 t の間にアップロードを開始したスレッドの数を計測する.

アップロードするデータの集約数を 1,2,4,8,16,32,64,128,256 と変えて, それぞれに対して (1)〜(4) を行った.

3.2.2 データ表現

アップロードするデータの表現形式は図 6 のようになる.

sensorGroup は複数の sensor で構成されている. 例えば, 温度, 湿度, 気圧, 雨量を計測する機能を持っているセンサーは, それ自体を sensorGroup とし, それぞれの項目を sensor として

集約数	m	s	t
1	200	2	10
2	200	5	10
4	200	10	10
8	200	20	10
16	200	40	10
32	200	80	100
64	200	80	100
128	25	100	200
256	25	125	1000

表 1　データ集約数に対する各実験条件

表す.

sensor タグは, そのセンサーの id により区別される. ある時間に観測されたデータを value タグとして持つ. この value タグは複数持つ事が DataUpload200703 [1] より可能になった. データを集約して送信するとは, 上記の XML の value タグを増やして送信する事を意味する.

```
<?xml version="1.0" encoding="UTF-8"?>
<sensorGroup authorization="MTIzNDU2Nzg=" class=
"combinedSensor" id="hongo.wide.ad.jp/WM918/elab/"
xmlns="http://live-e.org/DataType/2006/08/">
<sensor id="hongo.wide.ad.jp/WM918/elab/Temperature">
<value time="2007-06-01T09:35:07+09:00">25.6</value>
</sensor>
<sensor id="hongo.wide.ad.jp/WM918/elab/Humidity">
<value time="2007-06-01T09:35:07+09:00">56.7</value>
</sensor>
<sensor id="hongo.wide.ad.jp/WM918/elab/Pressure">
<value time="2007-06-01T09:35:07+09:00">1013.2</value>
</sensor>
<sensor id="hongo.wide.ad.jp/WM918/elab/RainFall">
<value time="2007-06-01T09:35:07+09:00">0.0</value>
</sensor>
</sensorGroup>
```

図 6　アップロードデータの表現形式の例

4. 検証の結果

4.1 アップロードノード数に対する応答時間

図 7 がアップロードノード数に対する応答時間の計測結果である. 横軸は単位時間 (1 秒) 当たりのアップロードノード数を対数軸で示したもの, 縦軸は平均の応答時間を示す.

各系列はデータの集約数を表す. この結果より, アップロードに係る時間が急激に増加するアップロードノード数を知る事が

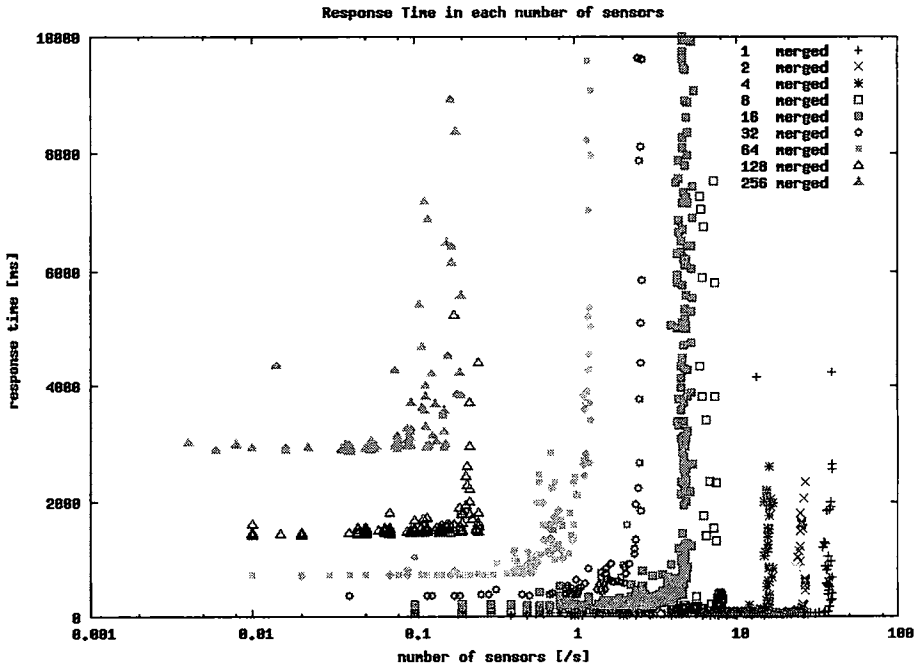

図 7　アップロードノード数に対する応答時間

できる.

4.2 データ集約数に対する最大サービス容量

各データ集約数における, 最大アップロードノード数は図 8 のようになる. 横軸はデータ集約数の対数軸, 縦軸は最大アップロードノード数を示す.

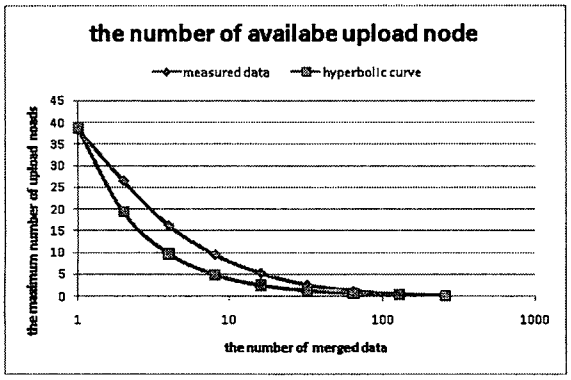

図 8 最大アップロードノード数

また, 縦軸をデータ集約数*最大データアップロード数, すなわち, 時間当たりのアップロードデータ数としたものが図 9 である.

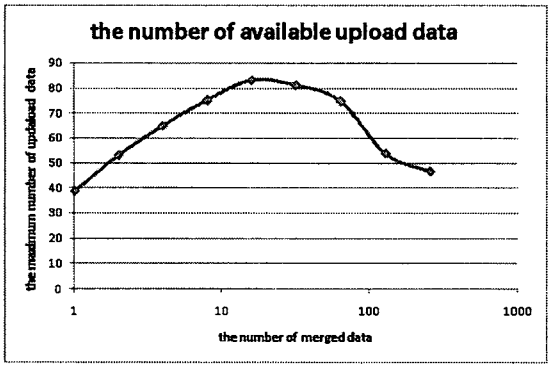

図 9 最大アップロード可能数

5. データ集約による通信のサーバ負荷軽減への考察

最大アップロードノード数はおよそ反比例となる双曲線を描く. データ集約によるサーバ負荷軽減の効果がなければ, データ集約数と最大アップロードノード数の積は一定となる. 図 8 の hyperbolic curve はデータ集約数と最大アップロードノード数の積を 38.5(集約数 1) とした時の理論曲線である. この理論曲線よりも実際に計測された値が大きい点では, サーバ負荷軽減の効果があると言える.

図 9 から, データの集約数が 16 までは, 最大アップロード可能数は単調増加する事がわかる. データ集約数が 1 から 16 になると, 単位時間当たりの最大アップロード可能数は 2.2 倍になる. また, データ集約数が 16 より大きくなると, 最大アップロード可能数は単調減少する.

章 3.1.1 で述べたように, サーバ側で行っている主な処理は

(1) 受け取った SOAP/HTTP メッセージの処理

(2) XML で表現されたデータの解析

(3) 解析されたデータのデータベースへの保存

である. このうち,(3) 解析されたデータのデータベースへの保存は, データの集約に依らず同じサイズのデータが保存されるので, サーバ負荷軽減の原因にはならない. データ集約により, 処理の負荷が小さくなるのは,(1) 受け取った SOAP/HTTP メッセージの処理,(2)XML で表現されたデータの解析である.SOAP のエンベロープの解析や XML の解析は複数に渡り何回か行われるより,1 度で行われた方が負荷は小さくなる. その結果が表れているものと考えられる.

今回のケースでは, データの集約数は 16 が最適な値である. 5 秒毎に気象データを観測しているセンサーの場合,80 秒毎にサーバにデータを送信する事になる. 気象データの場合は 80 秒に 1 回のデータアップロードは妥当であると考えられる.

一方, 観測と同時にデータをサーバにアップロードする必要があるセンサーも存在する. これらの混在を仮定した場合の最大アップロード可能数は図 10 になる.

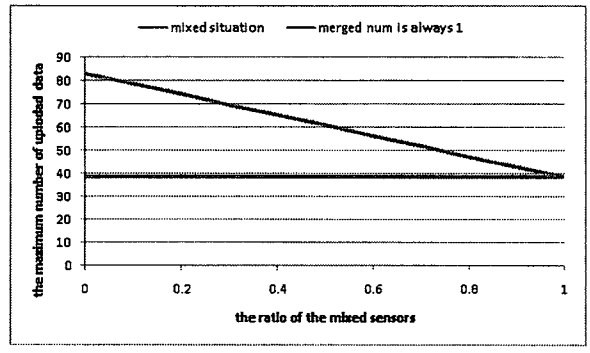

図 10 最大アップロード可能数

6. 結　　論

ノードがある間隔で生じるデータをサーバに送信する場合には, そのデータを集約して送信する事で, 通信効率が向上するケースがある. 本研究では, この問題をサーバ側におけるサービス負荷容量に焦点を当て, 実システムを模した環境で実験を行い, 集約に対する効率の向上を定量的に把握した.

一方, ユーザの要求やその性質上, 観測と同時にアップロードを必要とするデータも存在する. 本研究では,

(1) 基本的には, データ集約による送信を行う事

(2) ユーザの要求により最大遅延時間を設定可能にする事

を行うアルゴリズムを提案した.

データが集約されたセンサーとそうでないセンサーが混在した場合のサーバ負荷は図 10 となる.

文　献

[1] Live E! http://www.live-e.org/

[2] W3C, SOAP Version 1.2, http://www.w3.org/TR/soap/

[3] Kenneth Chiu et al., Ïnvestigating the limits of SOAP performance for scientific computing; In Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, pages 246-254, 2002.

[4] Christopher Kohlhoff, Ëvaluating SOAP for High Performance; WWW2003, May 2003 Business Applications: Real-Time Trading Systems

[5] Bhaskar Krishnamachari et al., T̈he Impact of Data Aggregation in Wireless Sensor Networks; IEEE 2002 Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'02), November 2002.

[6] Kevin Fall, Ä Delay-Tolerant Network Architecture for Challenged Internets, Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, 2003.

[7] RFC 896, http://www.faqs.org/rfcs/rfc896.html

[8] W. Richard Stevens, TCP/IP Illustrated, Volume1, Addison-Wesley