センサデータアップロードストリームの集中管理アーキテクチャ

Architecture for Centralized Management of Sensor Data Upload Stream

落合 秀也† 松浦 知史†† 砂原 秀樹†† 江崎 浩†

† 東京大学
東京都文京区本郷7-3-1 工2号館102B1 江崎研究室

†† 奈良先端科学技術大学院大学
奈良県生駒市高山町8916-5 インターネットアーキテクチャ講座

E-mail: †jo2lxq@hongo.wide.ad.jp, ††sato-mat@is.naist.jp, †††{suna,hiroshi}@wide.ad.jp

概要

本研究は，将来的な自律分散型センサデータ収集システムを念頭に置き，各組織で独自に所有している多数センサの管理を各々集約化し，管理コストを削減するアーキテクチャの提案を行っている．既存のセンサデータ収集システムをそのまま将来的なシステムに対応させようとすると，センサのデータストリームの制御を手作業で個々のセンサに対して行う必要があり，センサ数が多くなると極端に管理コストが増大する．また，既存システムにおいては，想定されているストリーム形態が不十分であり，将来的なシステムで導入されていくと考えられる冗長サービスには対応していない．本研究では，将来的なセンサデータ収集システムで必要となるセンサからのデータの送信形態を概観し AND/OR 木で表現できることを示すと共に，多数のセンサデータストリームを集中的に管理できるアーキテクチャの提案を行う．本研究で実装されたアーキテクチャの検証実験により，簡単な設定を集約点に登録するだけで，複数のセンサストリームを短時間で制御可能であることが示された．

1. はじめに

IP ネットワークで接続されたセンサのデータ収集システムにおいては，ファイヤウォールおよび NAT 下での運用も可能にするため，センサが自律的に観測データを遠隔サーバへ送信している場合が少なくない．既存のシステムは，基本的にサーバ集中型を念頭に設計されており，データ送信先情報がセンサ上に手作業で設定されている場合が多い．将来的なセンサデータ基盤(e.g., Live E![1])においてはサーバ集中型から自律分散型へ移行していくことが考えられている．センサデータ収集システムのアーキテクチャの変遷に伴い，以前は固定的に設定され運用されていた送信先情報が，今後は必要に応じて変更できるようになるべきだろう．現在のように送信先情報を手作業で設定しているシステムにおいては，送信先情報の変更が迫られるたびに，場合によっては現地で，センサに対して人の手で設定変更を行わなければならない．そのため管理しているセンサ数が増えると，管理コストが急激に増大してしまうと考えられる．

本研究は，組織単位で管理された複数のセンサを集中制御し，上位データ収集システムからの送信先変更などに関する要求に対し，比較的短時間で，低い管理コストで対応可能なシステムを提案するものである．本研究では，センサデータの流れをセンサデータストリームと呼び，このセンサデータストリームの制御に関して，重点的に，分析，考察を行い，検証している．なお，本研究では上位データ収集システムが，

(1) 複数のデータ・ドメイン
(2) 複数の冗長サービス

から構成されていることを想定している．

今後想定される基盤システムとしてのセンサデータ収集／配信システムは，自律分散システムとして構築され，センサデータはストリームとして配信されることが考えられる[2,3]．基盤システムは，運用上の都合により，たびたび構成が変更されることを想定する必要があり，センサは柔軟に設定変更が可能でなければならない．センサデータのアップロードサービス率を改善するために，既存のデータ送信スキームでは想定されていない冗長化が行われる必要もある．

本研究で提案するシステムアーキテクチャでは，センサデータ送信設定はサーバに集約され，センサデータストリームの構成を管理組織ごとに自由に設定できるものとなっている．センサデータストリームの構成としては，マルチ・ユニキャスト型とマルチホーム型，およびそれらの複合型が考えられ，これらは AND/OR 木で表現できる．提案アーキテクチャにおいては，各センサは定期的にこのサーバからデータストリーム構成を取得するように設計されており，センサの管理組織が，新しい構成

をサーバに登録すると，センサはその構成に従って，センサデータの送信を行うようになる．

N 台のセンサを考えた場合，1 回の送信先変更要求に対し，管理者が設定変更を行う回数は，従来のシステムは N 回であったが,提案アーキテクチャでは 1 回になる．提案アーキテクチャでは，1 回の設定で 10 分以内にすべてのセンサに変更を反映させることができるが，従来のシステムでは，この値を超えてしまうだろう．センサの設置環境やユーザインタフェースによっては，設定変更にさらに時間がかかることも考えられる．

本研究で述べるデータ送信を行う際の AND/OR 木の評価アルゴリズムは，TCP/IP プロトコルスタックに比べて単純であるため TCP/IP 対応の組込みシステムには実装科のだろう．実際的な環境においては，AND/OR 木のノード数は高々10 個と見積もっており，メモリ使用量も大きくないと思われる．

本研究では，集中管理サーバ，および動的に管理サーバから設定を読み込む仮想的なセンサを実装し，アーキテクチャの検証を行っている．

1.1. 関連研究

ワイヤレスセンサネットワークの分野においては，センサのデータ送信設定を動的に変更できる TinyDB[4]があるが，これはデータ内容を動的に設定できるものであって，マルチ・ユニキャストやマルチホーム構成などの送信形態を制御するものとはなっていない．近傍の Sink ノードに自動的にデータを流し込む Directed Diffusion[5]もあるが，これは，資源的に弱いセンサを補助するための機構で，データの送信スキームを集中管理するためのものではない．本研究は，インターネット上に構築されるバックボーンとしてのセンサデータ収集／配信システムへ，センサを接続するための管理を集約化するものである．

1.2. 本論文の構成

第 2 章では，将来のシステムで想定されるサービスの構造を概観する．第 3 章では，提案アーキテクチャを解説する．第 4 章では，実装および実験システムの設定について説明し，第 5 章では，実験結果を述べる．第 6 章で考察し，第 7 章でまとめる．

2. データストリームの制御パターン

ここでは将来的なセンサデータ基盤へデータ送信を行う場合のセンサデータストリームの構成(i.e., 送信スキーム)を概観する．センサデータストリームの形態は下記に示す 3 つの構成が考えられ，AND/OR 木で表現することが可能である．

(1) マルチ・ユニキャスト型(AND 型)

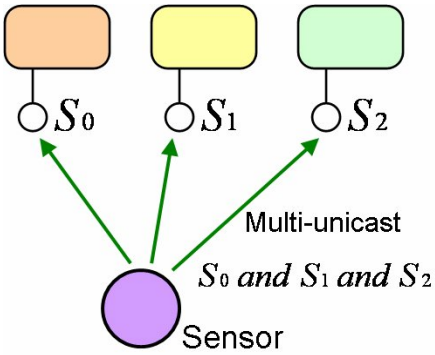

図 1. マルチ・ユニキャスト構成

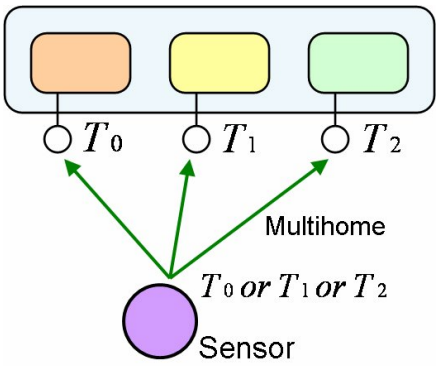

図 2. マルチホーム構成

(2) マルチホーム型(OR 型)

(3) 複合型

2.1. マルチ・ユニキャスト型 (AND 型)

完全に独立なシステムへセンサからデータを直接送信する構成が，マルチ・ユニキャスト構成となる．具体的には，情報基盤としてのセンサデータ収集システムへのデータ送信に加え，組織内利用のために，組織で独自に設けられたサーバへもデータを送信する場合が考えられる．また，バックボーンの情報基盤が完成するまでには，新旧サーバの同時運用，実験サービスの導入も考えられ，それらに伴い，複数のセンサデータストリームを生成する必要が出てくるだろう．

図 1 の例では，センサは，3 つの独立したシステムのサービス S_0, S_1, S_2 のすべてにデータをマルチ・ユニキャストする．このとき，データ送信スキーマを S_0 and S_1 and S_2 と表現することができる．

2.2. マルチホーム型 (OR 型)

サービスが冗長的であるシステム(つまり，複数のサービスが提供されているが背後ではデータの共有などの連携が図られているシステム[2]) へセンサデータを提供

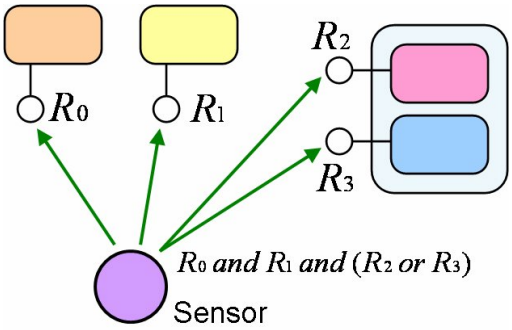

図 3. 複合構成

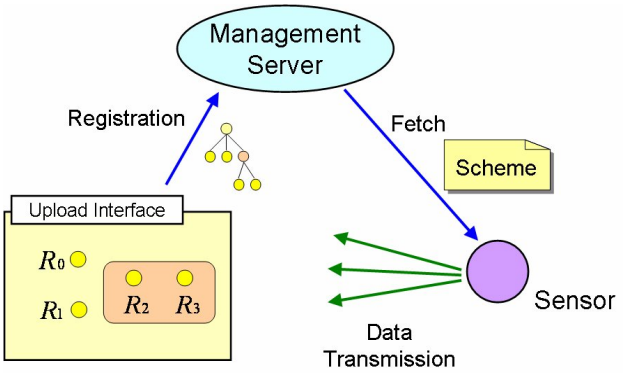

図 4. センサデータストリーム制御集中管理システムアーキテクチャ

する場合は，データストリームの形態をマルチホーム構成と捉えることができる．このようなシステムにおいては，マルチ・ユニキャストされると，システム内部での重複検出が必要となってしまうため，いずれか 1 つのサービスを通じてデータを投稿することが望ましい．図 2 の例では，*T0,T1,T2* のサービスは背後において連携が図られている．マルチホーム構成のデータ送信スキームは，*T0* or *T1* or *T2* と表現することができる．

2.3. 複合型

現実的な運用形態として，マルチ・ユニキャスト構成とマルチホーム構成が組み合わさった複合構成を考えることができる．図 3 のように，複数の独立なシステムへデータ配信を行うが，あるシステムについては，サービスが冗長化されている場合がある．図 3 の例では，*R2* と *R3* は冗長サービス(*R2* or *R3* と記述)であり，*R0* と *R1* と (*R2* or *R3*) へマルチ・ユニキャストしている．この構成は，*R0* and *R1* and (*R2* or *R3*) と表現することができる．

一般に，ストリームの構成は，AND/OR 木で表現することができ，(*U0* and *U1*) or (*U2* and *U3*) や *V0* and (*V1* or (*V2* and *V3*)) などの様々な構成を考えることも可能ではあるが，現実的にありえる構成はマルチ・ユニキャストの中に，マルチホーム構成が存在しているものぐらいだろう．

3. ストリーム集中管理アーキテクチャ

本研究で提案するセンサデータストリーム制御，集中管理システムのアーキテクチャを図 4 に示す．

センサデータアップロード先のサービス構造は，管理者によって，管理サーバに登録されていて，センサが定期的にその情報を取得し，自身の送信スキームを更新することで，多くのセンサデータストリームの統括的制御を可能にしている．以下では，それぞれの機能，構成および動作を具体的に解説する．

3.1. サービス構造の登録

アップロード先のサービス構造は，管理者によって，管理サーバに登録される．サービス構造は第 2 章で述べたように，AND/OR の木構造になっている．これは，複雑な構成ではないため，実際的には簡単に Web インタフェースを用いて，構成設定を行うことが可能だろう．

3.2. 冗長化された管理サーバ

管理サーバは，センサから定期的に参照される．サービス構造の更新後に，更新情報をセンサが取得しないうちにサービスが何らかの原因で停止してしまうようなことがあれば，更新結果がセンサに反映されないまま動き続けることになる．このような現象をできるだけ回避するため，本研究で提案するアーキテクチャにおいては，管理サーバを冗長化することにした．冗長化の仕組みはマスタ・スレーブ方式を取っている．管理者がマスタサーバの設定を更新すると，スレーブサーバにもその設定が反映される．

3.3. センサの管理情報

センサが管理サーバから取得する情報は，センサの管理情報 MI(Management Information)であると同時に，センサへの要求事項であると考えることもできる．この要求で必要な情報は，管理サーバのサービスポイントリストと，データの送信スキームである．冗長化のため，複数ある管理サーバのサービスポイントは，リストで表現でき，データの送信スキームは，AND/OR 木で表現できる．

3.4. センサの動作

センサは定期的に下記の操作を行う．

(1) 管理サーバから管理情報を取得
(2) アップロードスキーマに従いセンサデータを送信

センサ側のローカル実行環境に保持された管理情報を

```
M:=parseManagementServerList(m_MI);
For each m belong to M
   response:=getManagementInfo(m);
   If response.succeeded then
     m_MI:=response.mi;
     break;
   End-If
End-For
```

図 5. 管理情報取得アルゴリズム

```
scheme:=parseUploadScheme(m_MI);
postData(scheme,data);
```

図 6. データ送信手順

```
BOOL postDataUnicast(scheme,data){
   return uploadData(scheme.url,data);
}

BOOL postDataMulticast(scheme,data){
   C:=scheme.children;
   succeeded:=true;
   For each c belong to C
     If !postData(c,data) then
       succeeded:=false;
     End-If
   End-For
   return succeeded;
}

BOOL postDataMultihome(scheme,data){
   C:=scheme.children;
   index:=scheme.defaultIndex;
   S:=createRandomSeq(index,scheme.count); // e.g., {4,2,1,3}
   For i=1 to scheme.count;
     If postData(C[index],data) then
       return true;
     Else
       index:=S[i];
       scheme.defaultIndex:=index;
     End-If
   End-For
   return false;
}

BOOL postData(scheme,data){
  switch(scheme.type){
  case “unicast”:
    return postDataUnicast(scheme,data);
  case “multicast”:
    return postDataMulticast(scheme,data);
  case “multihome”:
    return postDataMultihome(scheme,data);
  }
  return false;
}
```

図 7. 送信スキーム解釈アルゴリズム

m_MI で表現することにして，これらの操作を，図 5 および図 6 に示すアルゴリズムで定義する．これらの操作が，定期的に実行されることで，センサは集中管理サーバから，送信スキームを自動的に読み込み，センサデータストリームが制御される．なお，図 6 のデータ送信手順にある postData 操作は，AND/OR 木構造の送信スキームに従ってデータ送信を行うため，図 7 に示したように再帰的に定義されている．

4. 検証システム

提案アーキテクチャを実装し，検証システムを構築した．検証する内容は，センサデータストリーム集中管理可能性(i.e., 期待通りのアップロードパターンが得られるか)と，一つの組織で管理可能なセンサ数の見積もりである．構築したシステムは，管理サーバと，データアップロードサービス，そして仮想的なセンサである．以下に,実装の詳細と検証に利用したシステムの構成を示す．

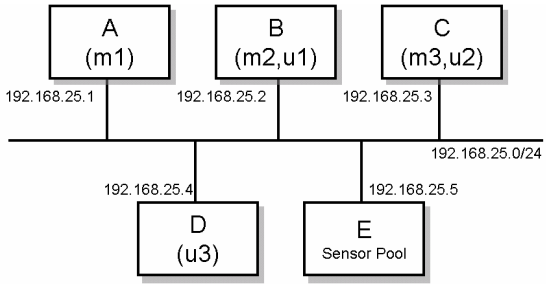

図 8. 検証システムの構成

表 1. パラメータ設定

項目	値
スレーブサーバ更新頻度	60 秒に 1 回
センサ数	100 個
送信スキーム取得頻度	8-12 分に 1 回 (この区間でランダム)
データ送信頻度	20 秒に 1 回

4.1. 実装

管理サーバおよびデータアップロードサービスは，Java と Tomcat5.5[6]により実装した．センサに対して提供されるサービスは，HTTP 上に構成され，サービスポイントは URL で表現することにした．マスタ・スレーブサーバ間の連携，および，仮想センサとのデータ交換は，すべて XML により行われる．仮想センサも Java で構築されており，センサプールの中に複数の仮想センサを同時に作り出すことで，データ送信パターン検証に必要な数のセンサを用意している．

4.2. 構成

5 台のコンピュータをプライベートネットワークで接続し，これらのサーバに管理サービス，データアップロードサービスそして仮想センサを配置して実験を行った(図 8)．ここで用いたコンピュータは，いずれも CPU が Pentium D でクロック周波数は 2.8GHz のものである．図 8 において，管理サーバは，m1 がマスタサーバ，m2, m3 がスレーブサーバとなっている．データアップロードサービス u1,u2,u3 は，ここでは独立したものとなっているが，マルチホーム実験の場合は，背後でつながれているものと想定して,データ送信パターンを検証した．図 8 において，コンピュータ E には，センサプールを用

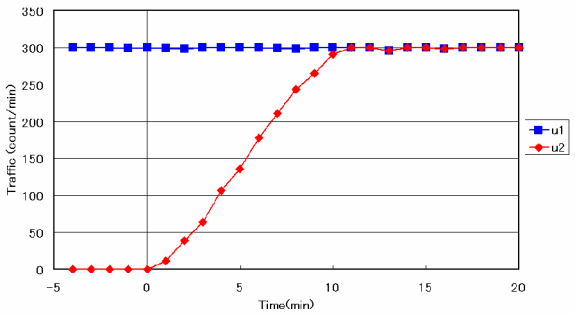

図 9. トラフィックの遷移 (多重度：1→2)

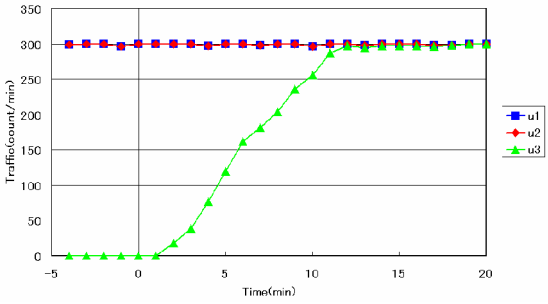

図 10. トラフィックの遷移 (多重度：2→3)

意し，仮想センサを動かした．パラメータ設定値は，表 1 に掲載する．

5. システム評価試験

本研究では，アップロードパターンの計測と，管理サーバの負荷試験を行った．アップロードパターン計測では，様々な送信スキーマにおいて，管理者による設定変更前後の過渡的状態を観測し，理論通りの結果が得られたことを示す．管理サーバの負荷試験によって，1 つの組織が集中制御可能なセンサ数の見積もりを得た．

5.1. アップロードパターンの検証

検証の設定で用いたデータ送信頻度とセンサ数では，一本のセンサデータストリームには，毎分 300(＝60／20×100)回のトラフィックがあることが算出できる．ここで述べるアップロードパターン評価試験では，毎分それぞれのアップロードサービスに対して，どれだけのトラフィック[回/分]があるかを検出してアップロードパターンの遷移をグラフ化している．

5.1.1. マルチ・ユニキャストパターン

図 9, 図 10 は，それぞれ，ストリームの多重度が，1 から 2 へ変更された場合と，2 から 3 へ変更された場合のストリームトラフィック量の遷移を示している．これら図からわかるように，設定変更は，10 分程度ですべてのセンサに反映されたことがわかる．

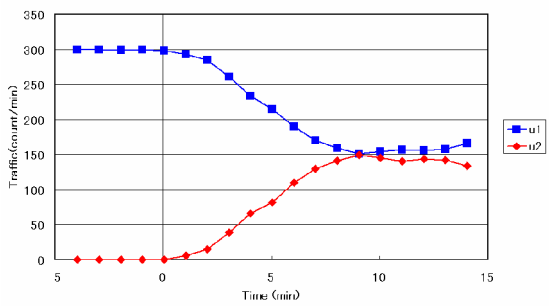

図 11. トラフィックの遷移 (冗長度：1→2)

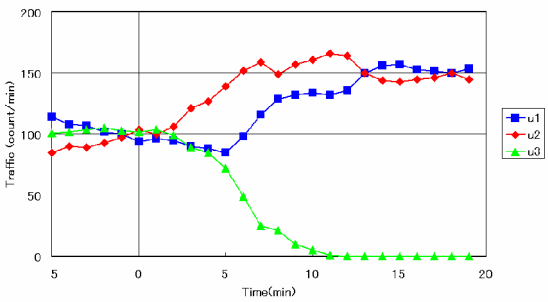

図 12. トラフィックの遷移(冗長度：3→2)

5.1.2. マルチホームパターン

図 11, 図 12 にマルチホーム型のストリーム分岐の様子を示す．図 11 は，ユニキャスト型のストリームを，2 本のマルチホーム型になるように設定変更した際の過渡現象を表している．当初，1 本のストリームで 300[回/分]のトラヒックを送り出していたものが，設定変更の約 10 分後，トラフィック約 150[回/分]の 2 本のストリームに分岐されたことが読み取れる．図 12 は，3 本のマルチホームストリームを 2 本に減らした様子を示している．トラフィック量も 100[回/分]×3[本]から，150[回/分]×2[本]に遷移した様子がわかる．

5.1.3. ロバストネスの検証

この実験では，下記の 2 種類のロバストネスを検証した．

1. 管理サーバの一部が停止した場合
2. マルチホームアップロード形態での，アップロードサービスが停止した場合

図 13 および図 14 にそれぞれ管理サーバが正常に稼動している場合と，3 台ある管理サーバのうち 2 台が停止した場合の，センサデータストリーム制御の様子を示す．双方共に u1 or u2 型から，(u1 or u2) and u3 型に変更された際のアップロードパターンの過渡的状態を示している．これらの図からわかるように，管理サーバのいくつかが停止した状態では，設定変更がセンサストリーム

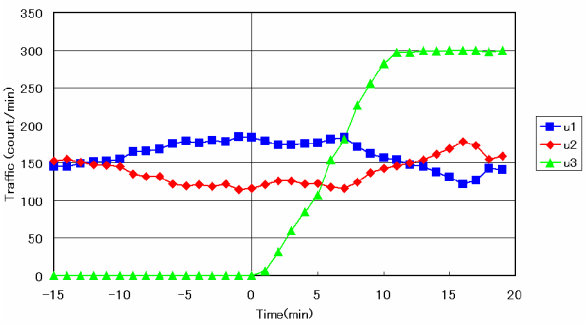

図 13. 管理サーバが正常な場合での
設定変更が反映される様子

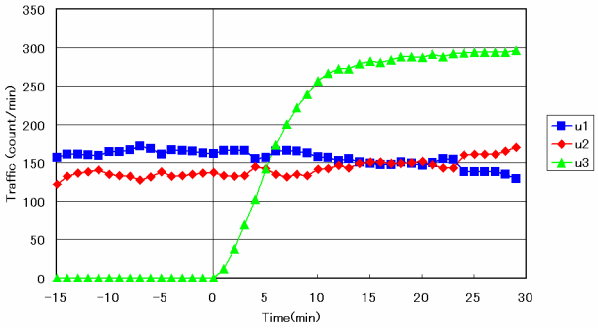

図 14. 管理サーバ(スレーブ 2 台)が落ちた状態での
設定変更が反映される様子

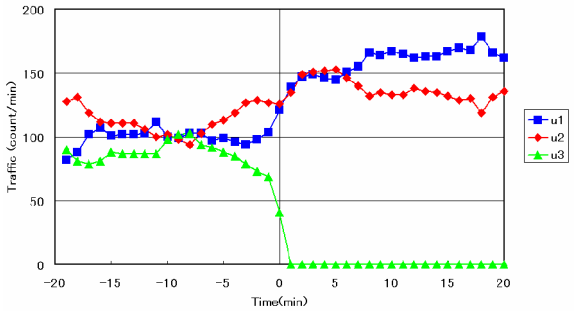

図 15. アップロードサービスが落ちたときの挙動
(1 台目：マルチホーム冗長度 3)

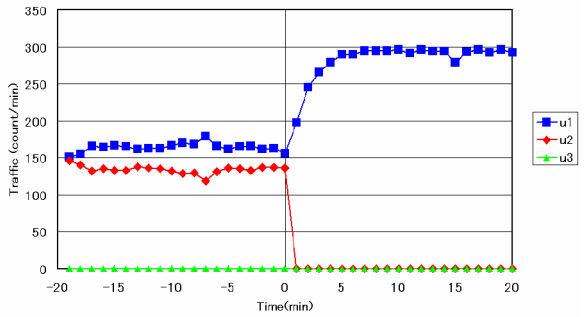

図 16. アップロードサービスが落ちたときの挙動
(2 台目：マルチホーム冗長度 3)

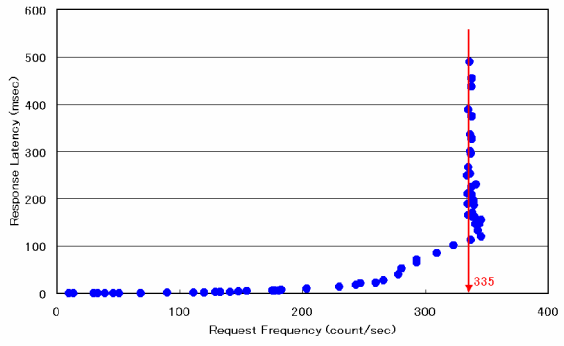

図 17. 管理サーバ 負荷試験結果

の制御に完全に反映されるまで，正常の場合と比べ 3 倍ほど時間がかかっているが，10 分後の段階ですでに 85%のセンサに反映されていることが読み取れる．

冗長度が 3 のマルチホーム構成において，アップロードサービスが 1 台，2 台停止した場合のストリームフローをそれぞれ図 15, 図 16 に示す．これらの結果から，マルチホーム構成においては，利用可能なサービスに動的にトラフィックを分配可能である，すなわち，ロバストであることがわかった．

5.2. 集中管理できるセンサの数

この検証実験では，単一の組織がどれだけのセンサを集中的に管理できるかを，管理サーバの負荷試験を元に算出している．結果，実験の設定値においては約 40 万個のセンサに対応できることがわかった．管理サーバの問い合わせ処理能力は，生成すべき送信スキームの構成によっても変わってくる．この実験では，実際の利用形態よりも計算量を多少多く必要とすると考えられる，

$$(R_0 \text{ or } R_1 \text{ or } R_2) \text{ and}$$
$$(R_3 \text{ or } R_4 \text{ or } R_5) \text{ and}$$
$$(R_6 \text{ or } R_7 \text{ or } R_8)$$

の構成を想定して，負荷試験を行っている．この構成において，様々なリクエスト量に対するリクエスト処理時間の計測を行った．結果を図 17 に示す．

待ち行列理論によれば，サービスの処理能力限界(実験結果では 335[回/秒])の 70%程度を現実的な処理能力と見るのが良いとされている．従って，この結果から，現実的なサービス処理能力 S[回/分]を算出すると，

$$S = 335 \times 0.7 \times 60$$
$$= 1.4 \times 10^4 \ [\text{回}/\text{分}]$$

となる．これは，1 分間に約 1 万 4 千件ものリクエストを処理できることを示している．ここで，センサの問い合わせ周期を T[分]，管理サーバの冗長度を R とすると，処理可能なセンサの数 N は，

$$N = SRT$$

$$\therefore N = 1.4 \times 10^4 \times RT \ [\text{個}]$$

となる．よって，実験での設定 T　10[分]，R ＝3 の場合は，N　40[万個]となる．

6. 考察と課題

マルチホーム型のアップロードパターン評価試験において，冗長サービスへのアップロードパターンに最大20%程度の揺らぎを観測した．これは各センサが，アップロード先を複数の候補からランダムに選択しているためで，誤差の範囲と思われる．

将来的なシステム課題としては次の機能をあげられる．

1. 数種類のアップロードインタフェースに対応
2. センサごとのストリーム制御機能

本研究では，データアップロードのインタフェースは，すでに標準化されたものであるとみなし，1 種類しかないことを前提としていた．しかし，開発途上のシステムにおいては，アップロードインタフェースはタイプが変更される場合もあり，実際的な環境においては，数種類のインタフェースが混在する状況も想定される．このような状況を考えると，センサは自身が対応しているインタフェース名を管理サーバに提示し，その要求に応じた返答を，管理サーバが生成する仕組みが必要となるかもしれない．

また，本研究においては，すべてのセンサを統括的に制御していたが，個々のセンサに対して特別にストリーム制御を行いたい場合もありうるだろう．このような状況に対応させるためには，センサが問い合わせの際に，自身のセンサ ID をサーバに提示させることが必要となる．管理サーバは，ID を提示されることで，個別の管理情報を応答することが可能になり，個々のセンサに渡った，詳細な管理も実現できる．詳細な管理を行う場合も，本研究で提案するアーキテクチャにより，集約点で管理する方が個々のセンサに直接設定を書込むよりも効率が良い．機種に依存する問題や，設置環境に依存する問題を集約点で吸収できるためである．

7. おわりに

本研究は，組織で所有している多数のセンサの管理を集約し，管理コストの削減を可能にするアーキテクチャを提案した．本研究では，まず，センサデータストリームの形態を分析後，AND/OR 木として表現した．その後，ストリームの形態を，集中管理できるように，管理サーバを導入し，センサが定期的にそのサーバから管理情報を取得する仕組みを採用した．本研究で作成した検証システムにより，集中管理サーバでセンサデータストリームの制御を短時間および低コストで実現できることを示した．

謝辞

本研究は，総務省の委託研究「ユビキタスネットワーク認証・エージェント技術の研究開発」の一環として実施したものである．

参考文献

[1] H. Esaki, and S. Hideki, “Live E! Project: Sensing the Earth with Internet Weather Stations,” In Proceedings of the 2007 International Symposium on Applications and the Internet (SAINT'07), jan 2007.

[2] D. J. Abadi, W. Lindner, S. Madden, and J. Shuler, “An integration framework for sensor networks and data stream management systems,” In Proceedings of the 30th VLDB Conference, 2004.

[3] H. Ochiai, Z. Wang, R. Oguchi, T. Sugiyama, Y. Sakamoto, S. Ishida, and H. Esaki, “Application of Content-Based Network for Sensor Data Distribution System,” In Proceedings of the 2007 International Symposium on Applications and the Internet Workshops (SAINTW'07), page 74, jan 2007.

[4] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “Tinydb: an acquisitional query processing system for sensor networks,” ACM Transactions on Database Systems, vol. 30, no.1, pages122—173, mar 2005.

[5] I. Chalermek, G. Ramesh, and E. Deborah, “Directed diffusion: a scalable and robust communication paradigm for sensor networks,” In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM 2000), pages 56—67, aug 2000.

[6] Apache Tomcat, http://tomcat.apache.org/