ホームネットワークのためのセンサ連携を支援するイベント駆動フレームワーク

松浦　知子[†]　田頭　茂明[†]　北須賀輝明[††]　中西　恒夫[†]
福田　晃[†]

Event-Driven Processing Framework Supporting Sensor Cooperation for Home Networks

Tomoko MATSUURA[†], Shigeaki TAGASHIRA[†], Teruaki KITASUKA[††], Tsuneo NAKANISHI[†], and Akira FUKUDA[†]

あらまし　通信機能を有する家電製品を接続してホームネットワークを形成し，実世界の情報に応じた処理を行うコンテクストアウェアネス技術の実現が望まれている．コンテクストアウェアアプリケーションでは，センサの測定データに現れるイベントをサービスの開始，変更，終了の契機とするために，それらのイベントを検出することが重要である．本論文では，ホームネットワーク上に分散しているセンサを利用して，コンテクストアウェアネス実現に必要なイベントを効率的に検出するフレームワークを提案する．具体的には，多様なセンサを抽象化し，センサを利用するアプリケーションに対して統一的なインタフェースを提供する．また，分散する複数のセンサをイベント駆動で連携させて，それらのセンサに関係する複合したイベントを効率良く検出する仕組みも提供する．これにより，アプリケーション開発者の負担を軽減し，ホームネットワーク上のセンサを連携するために必要な通信量を削減する．また，本フレームワークにのっとったセンサによるイベント通知機能の実現法の一例として UPnP （Universal Plug and Play）を利用した実装結果を示し，効率良くイベントの検出が可能となることを確認する．

キーワード　ホームネットワーク，コンテクストアウェアネス，イベント検出，イベント駆動フレームワーク

1. まえがき

近年，家電製品など我々の身辺に存在する多くの機器が通信機能をもつようになり，家庭内の機器を接続して，ホームネットワークを実現しようという動きが活発である．ホームネットワークを用いることで，機器はネットワークでつながった他の機器やセンサを利用して，より多くの情報を集めることができる．これにより，機器がユーザの置かれている状況（コンテクスト）を自動的に判断し，その状況に合った適切なサービスを提供するコンテクストアウェアネスの実現が現実味を帯びてきた [1] ．コンテクストとは人やものの状況を特定するために利用できるあらゆる情報である [2]．そして，コンテクストアウェアアプリケーション（CAA）とは，コンテクスト情報を利用して，ユーザに適した情報やサービスを提供するアプリケーションである．コンテクストアウェアシステムの実現を目指し，この多種多様なコンテクスト情報を扱うシステムも数多く発表されている [3], [4].

CAA は，センサを利用して，実環境の状況を判断する．特に，コンテクストの変化に対応して，提供するサービスの開始・内容の変更・終了といった振舞いを決定することが多い．コンテクストの変化は，コンテクストに関連するセンサの測定データが，何らかの条件を満たしたことを検出することで知ることができる．本論文では，センサの測定データがある条件を満たしたことをイベントと定義する．CAA がイベントを検出する方法としては，CAA がネットワークを介して

[†] 九州大学大学院システム情報科学府・研究院，福岡市
Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka-shi, 819–0395 Japan
[††] 熊本大学大学院自然科学研究科，熊本市
Graduate School of Information Science and Electrical Engineering, Kumamoto University, Kumamoto-shi, 860–8555 Japan

定期的にセンサの測定データを取得し，CAA 側でイベントを検出する方法と，センサ側で条件判定し検出したイベントを，CAA に対してイベント駆動により（非周期的に）通知する方法の二つが考えられる．一般的に，イベントの発生頻度の低いときにはイベント駆動による方法が，逆に発生頻度が高いときは定期的に取得する方法が通信量の点で有利とされている．本研究では，イベント駆動によりイベントを検出する手法に着目する．センサ内部でイベントを検出し，CAA に通知するためにはセンサが高機能である必要があるが，今後発展が期待されるホームネットワークにおいては，定期的に取得する手法と同時にイベント駆動によるイベント検出の実現も重要な技術課題であると考える．

本論文では，様々な種類のセンサと家電機器とで構成するホームネットワークを想定し，ホームネットワーク上に分散しているセンサを利用して CAA を効率的に実現するフレームワークを提案する．具体的には，多様なセンサを抽象化し，センサを利用する CAA に対して統一的なインタフェースを提供する．加えて，分散する複数のセンサ間で連携して，それらのセンサに関係する複合したイベントを効率良く検出する仕組みも提供する．提案フレームワークでは，イベント駆動方式でイベントを検出することで，センサの連携に必要な通信量を削減している．また，提案フレームワークの実現法の一例として，提案フレームワークにのっとった機能を提供するセンサを UPnP(注1)を用いて実装した結果についても記述する．具体的には，UPnP にセンサ連携のためのモデルを新たに追加し，そのモデルを通してセンサの抽象化及び複数のセンサ間の統一的なインタフェースを提供している．提案フレームワークを適用可能なセンサとして，UPnP が動作する程度の性能を有する必要がある．これは本研究では，ホームネットワークを想定しており，その利便性の高さから UPnP を利用することを想定しているためである．更に，イベント検出をセンサ側に負担させるために，センサ値に対する条件判定が行えるネットワークサービスを実行できる程度の性能が必要である．

本論文の構成は，以下のとおりである．まず，**2.** でコンテクストの変化を検出するために重要となるイベントの検出について述べる．次に，**3.** で本論文で提案するフレームワークについて述べ，そして，**4.** で本提案フレームワークにのっとったイベント検出機能を提供するセンサの実装及び評価について述べる．**5.** で関連研究についてまとめる．最後に **6.** でまとめと今後の課題を述べる．

2. イベント駆動によるコンテクストの変化の検出

2.1 想定環境

本論文で想定している環境は，図 1 のような様々なセンサと家電とからなるホームネットワークである．ホームネットワーク上の機器は，ネットワークでつながったセンサから実環境の情報を取得し，その情報をもとに現在のコンテクストに適したサービスを，ユーザに対して提供する．従来のホームネットワークでは，各機器自身が必要とするセンサを内蔵し，自身の制御のために利用するというものであった．ネットワーク経由でセンサを共有できるようになれば，一つのセンサをホームネットワーク上の多くの機器で共有することが可能である．また，家電に内蔵されたセンサだけではセンサの種類に限りがあるため，不足しているセンサを単体でホームネットワーク上に存在させることも考えられる．このように，機器に内蔵されたセンサや単体で設置されたセンサをネットワークを介して共有することにより，センサを内蔵していない機器も，CAA を提供することが可能となる．CAA を提供する多くの機器がそれぞれ専用のセンサをもつのではなく，少数のセンサをネットワークを介して共有することで，ホームネットワーク上で効率良く CAA を実現できると考えている．

2.2 イベントの検出

CAA は，センサを利用してユーザの置かれている実環境の情報を取得し，コンテクストを推定する．しかし，センサにより得られた情報がそのままコンテクス

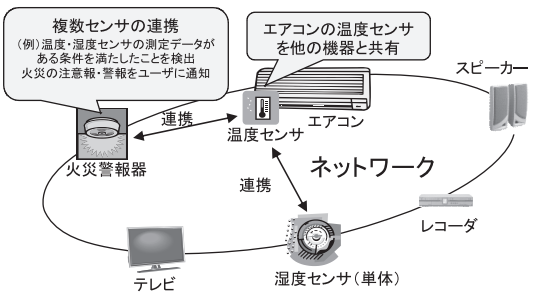


図 1　想定環境の概観．
Fig. 1　Overview of our target environment.

(注1)：UPnP は，UPnP Implementors Corporation の商標である．

トとして利用できることは少なく，センサの測定データをコンテクストに変換する処理と，そのコンテクストの変化を検出する仕組みが必要である．すなわち，CAA はセンサの測定データを変換して得られたコンテクストの変化に応じて，サービスの開始，内容の変更，終了といった振舞いを決定する．コンテクストの変化は，コンテクストに関連するセンサの測定データが何らかの条件を満たしたことを検出（イベントの検出）することで知ることができる．また，複数のセンサの測定データを利用してコンテクストを推定する場合では，構成される測定データに関する複数のイベントを組み合わせて単一のイベントとして取り扱うことができる．

例えば，温度センサ，人感センサ，及びエアコンがホームネットワークにより接続された環境を考える．エアコンは，温度センサの測定値が 27 度以上であれば，その部屋のコンテクストを「暑い」と判断する．また，人感センサの測定値が人を検知していれば，コンテクストを「人がいる」と判断する．そして，その部屋のコンテクストが，人がいてかつ暑いというコンテクストに変化すると，冷房サービスを開始するものとする．このコンテクストの変化は，温度センサの測定値が 27 度以上という条件を満たしたというイベントと，人感センサの測定値が人を検知したというイベントを組み合わせたイベントにより検出できる．

2.3 課　　題

多種多様なセンサと家電が分散して存在するホームネットワーク上でイベントを検出するためには，いくつかの課題が存在する．以下に，想定するホームネットワーク環境でイベントを検出する際の問題点をまとめる．

- ホームネットワーク上に存在するセンサは，その種類や精度，提供ベンダなどが様々で，それぞれ扱い方が異なる．CAA はセンサごとのインタフェースの違いを考慮して，センサを利用しなければならない．
- CAA により検出したいイベントが異なるため，センサは様々なイベントに（複数のイベントを組み合わせたイベントを含めて）柔軟に対応する必要がある．
- センサと CAA が分散していることから，イベントの検出に多くの通信量がかかる．

イベントを検出するためにセンサの測定データをどのように利用するかは，実現する CAA に依存するが，センサの測定データがある条件を満たしたというイベントを検出する処理は，多くの CAA が共通して必要とする処理である．このために，本研究ではイベント検出を効率良く行うフレームワークとして提供することを目指す．

3. 提案フレームワーク

本章では，センサが分散して存在するホームネットワーク上でイベントを検出するためのフレームワークを提案する．本論文で提案するフレームワークでは，イベント検出処理の中の共通部分をくくり出し，その機能をセンサが自身の機能の一つとして提供することで，CAA 側での処理を減らし CAA 開発者の負担を削減することを目的とする．具体的には，提案フレームワークが提供する機能を以下に概説する．

［多様なセンサの抽象化］　ネットワーク上のすべてのセンサが，CAA に対して，共通のインタフェースを提供することで，CAA は，センサの多様性を意識することなくセンサを利用できる．本フレームワークでは，センサの固有情報を開示する統一された仕組み，及びイベント検出処理の統一されたインタフェースを提供する．

［分散するセンサの連携］　CAA が複数のセンサを利用する場合，CAA がそれらのセンサに個別にイベント検出を要求するのではなく，複数のセンサが連携して CAA の要求にこたえる．特に，複数のイベントの論理積（AND）及び論理和（OR）に着目し，これらをセンサが連携することで対処する．

提案フレームワークでは，以上の機能を限られたセンサのリソース・機能を使って実現することを目指す．特に，センサの電力消費に着目しており，分散するセンサの連携に関して，イベント駆動によりセンサ間及びセンサと CAA 間の通信量を抑えること，及び連携中のセンサをできる限り休止することを目指している．以下，各機能の詳細について記述する．

3.1 多様なセンサの抽象化

ホームネットワーク上のセンサは，単体で存在することも，家電に内蔵された形で存在することもある．また，その種類や精度，提供ベンダなども様々で，それぞれ扱い方が異なる．このような多種多様なセンサを同一のインタフェースで扱える仕組みが必要である．このために，まずイベントを検出する基本的なステップを，要求受付，判定，及び通知として定義する．本節では，特に要求受付のステップについて詳細を記述する．判定や通知ステップは，要求受付が設計できると，その要求に従った判定を行い，判定後通知するだ

けのステップであり，特に複雑な部分がないためである．要求受付で重要な機能としては，センサ固有情報の開示機能と要求受付機能がある．以下，各機能について詳細を説明する．

3.1.1 センサ固有情報の開示

CAA は，自身が知りたい情報を得るために，適切な位置に設置された，十分な精度とサンプリング周期をもったセンサを選ぶ必要がある．そのために，センサは CAA に対して自身の固有情報を開示する．本フレームワークに準拠するセンサは少なくとも次の情報を開示するものとする：センサの種類（測定対象），設置場所，精度，及びサンプリング周期である．これらの情報をもとに，CAA は自身が知りたいコンテクストの変化を検出するのに，適当なセンサをホームネットワーク上から探し出し利用する．センサの設置場所の情報は，CAA がセンサを利用する際，重要な情報である．センサを利用する CAA は，目的のイベントを検出するのに適した場所にセンサが設置されているかどうかを判断するために利用する．また，精度やサンプリング周期については，目的のイベントを検出するために必要なセンサの精度とサンプリング周期を判断するために用いる．

3.1.2 イベント検出の要求受付

CAA は，センサに対して，興味のあるイベントを何らかの形で表現し，イベントの検出を要求する．本フレームワークに準拠するセンサはすべて，次の統一された方法で表現されたイベント検出要求を受け付ける機能を提供する．

まず要求受付の手順について説明する．単一の単位条件からなるイベント検出要求の場合，CAA は，利用する一つのセンサに対して要求を出せばよい．複数の単位条件からなるイベント検出要求の場合は，その条件中に出現するセンサのうち任意の一つに対して要求を出せるものとする．イベント検出要求を受け付けたセンサは，要求内容を解析し，複数の単位条件からなるイベント検出条件の場合は，他の関係するセンサに要求があったことを通知する．このとき，要求されたイベント検出条件も同時に通知する．CAA が要求をキャンセルする場合は，はじめに要求を出したセンサに対して，キャンセル要求を出すものとする．複数の単位条件からなるイベント検出要求に対するキャンセルを受け付けたセンサは，他の関係するセンサに要求がキャンセルされたことを通知する．

次に，要求条件の表現方法について説明する．前章で述べたように，コンテクストの変化はセンサの測定データが何らかの条件を満たしたというイベントを検出することにより認識できる．測定データとコンテクスト情報との関係を考えると，しきい値を超えたというしきい値超過条件のみでも十分にコンテクストの変化を検出することができると考え，提案フレームワークでは，イベントの条件としてしきい値超過条件を適用する．

しきい値を設定する測定データとしては，センサの測定値に加えて，測定値の傾きも考える．この傾きの値についても，指定されたしきい値を超えたか否かを判断することにより，測定値の急激な変化，すなわち，コンテクストの変化を検出することができる．また，しきい値の設定は時刻に対しても行えるものとする．時刻を指定したしきい値を超えた場合の通知では，測定値または測定値の傾きを通知する．更に，測定値，測定値の傾き，または時刻がしきい値を超えたか否かという条件を単位条件とし，複数の単位条件の論理積や論理和も条件としてを扱うことで複雑な条件を表現することもできる．各単位条件はネットワーク上の異なるセンサの測定データに関するものを指定することができる．

しきい値の設定は，各センサが測定値と時刻のグラフと，測定値の傾きと時刻のグラフを想定し，そこに値若しくは時刻を指定してしきい値の線を引く作業とみなすことができる．図 2 に温度センサの測定値のグラフにしきい値を設定していく例を示す．このグラフは横軸は時間，縦軸は温度センサの測定値を表している．はじめはグラフ上に線が引かれていない状態である．ここに，測定値が 25 度を超えたら通知するようしきい値を設定する場合は，25 度に水平方向の線を引くことになる．これは測定値に対して直接値を指定するしきい値設定である．また，10 分ごとに測定値を通知するよう時刻に対する等間隔のしきい値設定を行う場合は，垂直方向に 10 分間隔で線を引くことになる．複数の線を引いた場合は，その線が表す条件同士の関係を AND または OR で指定する．この例では，温度センサの測定値に関する二つの条件を OR でつないでいる．よって，図 2 の実線のように温度が変化した場合には，10 分ごとの垂直方向の線を超えるたびに，そのときの温度センサ測定値を通知し，加えて 25 度の横線を超えたときには，しきい値を超えた旨を通知する．

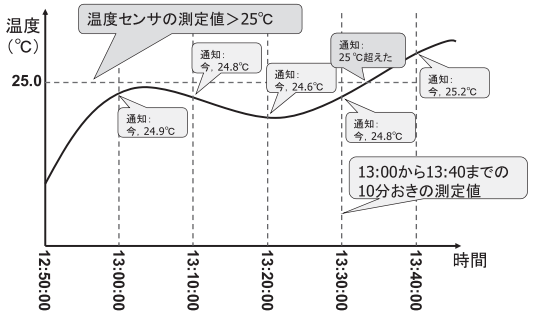


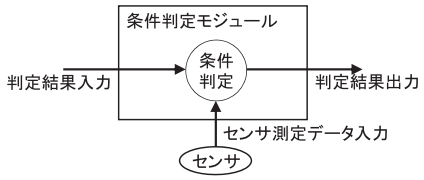
図 2　イベント条件とイベント検出のイメージ
Fig. 2　Illustration for event condition and event detection.

3. 2　分散するセンサの連携

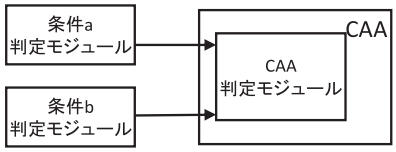
センサは，CAA からの要求を受け付けた後，自身の測定データに対して条件判定を行い，要求されたイベントを検出する．複数の単位条件からなるイベント条件の場合は，効率良く目的のイベントを検出するために，センサ間での連携の仕方を工夫する必要がある．

まず，センサは CAA から新たなイベント検出要求が出されると，イベント検出要求に含まれる要求条件を解析し，要求条件を判定するモジュールを用意するものとする．要求条件が複数の条件がなる場合は，条件ごとにモジュールを用意する．この場合，そのセンサ以外のセンサに関する条件が含まれている場合は，関係するセンサに要求を転送し，転送先のセンサでモジュールが用意される．これらのモジュールが互いに連携し，目的のイベントを検出する．条件判定モジュールの概念図を図 3 (a) に示す．各モジュールは他のモジュールの判定結果を受け取る入力と，センサから測定データを受け取る入力をもつ．また，判定結果を出力する．モジュールは測定が行われるたびに条件判定を行い，条件を満たしたなら，その旨を出力する．要求条件が複数の条件からなる場合は，論理和でつながれた条件の場合は，この条件判定モジュールを並列に（図 3 (b)），論理積でつながれた条件の場合は，この条件判定モジュールを直列に（図 3 (c)）接続する．また，これらは入れ子にして接続することも可能である．条件判定結果の出力先は，CAA から要求された条件が，単一の単位条件からなるか，複数の単位条件の論理積からなるか，複数の単位条件の論理和からなるかにより異なる．

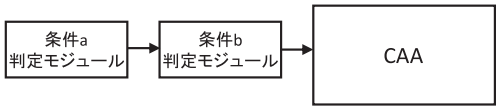
単一の単位条件からなるイベント検出条件の場合は，出力が，要求元である CAA に対して行われ，イベント検出が実現される．要求を受け付けると，単位条件判定モジュールが条件判定を開始し，イベントが検出されると CAA に対して通知する．この条件判定は，イベント通知要求がキャンセルされるまで続けられる．



(a) 条件判定モジュールの概念図



(b) 条件 a OR 条件 b の場合



(c) 条件 a AND 条件 b の場合

図 3　条件判定モジュールの連携
Fig. 3　Cooperation for condition check modules.

複数の単位条件の論理和からなる要求条件の場合の処理について説明する．CAA は，要求条件に含まれるセンサの中から一つセンサを選択し，そのセンサに対して要求を送信する．更に，CAA 上の条件判定モジュールで OR の最終的な判断を行う条件判定を開始する．選択されたセンサが要求を受け付けると，そのセンサは，自身の条件判定モジュールの条件判定を開始し，更に関係のある他のセンサに対して要求があることを通知し，連携を要求する．その後は，それぞれの条件判定モジュールが条件判定を行い，イベントを検出した際には，それぞれ CAA 条件判定モジュールに対して通知する．例えば，図 3 (b) において，条件 a が成立すると，条件 a 判定モジュールが CAA 条件判定モジュールに通知し，更に CAA に通知されることになる．条件 a が満たされなくなった場合は，成立した場合と同様に，CAA に通知されることになる．こ

こで，条件 a と条件 b が同時に成立していた場合において，どちらか一方の条件が成立しなくなった状況を考える．この状況では，成立しなくなった条件判定モジュールがそのことを CAA 条件判定モジュールに通知することになるが，CAA 条件判定モジュールはもう一方の条件が成立しているために，CAA には通知しない．

複数の単位条件の論理積からなる要求条件の場合の処理について説明する．あるセンサが要求を受け付けると，そのセンサは，関係のある他のセンサに対して要求があったことを通知し，連携を要求する．そこで，それぞれの測定データの過去の履歴から条件が成立する頻度を予想し，予想成立頻度の高いものほど CAA 側にくるように直列に連結する．例えば，図 3 (c) では，条件 b の方が成立頻度が高いと予想されたとしている．まず，CAA から一番遠いモジュールの条件判定を開始し（図 3 (c) の場合では条件 a），その他の判定モジュールは省電力化のために休止する．条件 a が成立し，条件 b 判定モジュールに通知されると，条件 b 判定モジュールは条件判定を開始する．条件 b も成立すれば，条件がすべて満たされイベントが検出されるので，条件 b 判定モジュールから CAA に対してイベントが通知される．条件 a 判定モジュールは，条件 a が成立し，条件 b 判定モジュールにその旨を通知した後，条件不成立の判定を開始する．条件が満たされなくなった場合は，条件 b 判定モジュールに対して通知する．これにより，条件 b 判定モジュールは再び休止状態に戻り，条件 a 判定モジュールは再び条件成立を判定する．このようなセンサ間の連携を可能とする枠組みにより，複数の単位条件の論理積からなるイベント通知条件の場合も，通信量と消費電力の点で効率良くイベントの検出ができる．

4. 評　　価

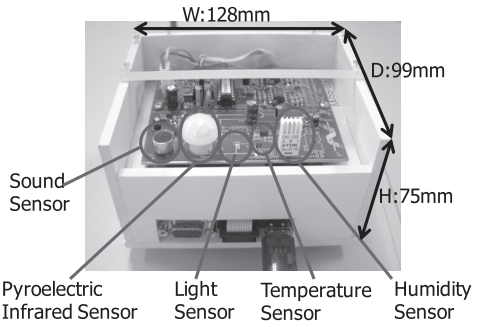
本章では，**3.** までに述べた提案フレームワークの実装について述べ，また提案フレームワークを用いた開発効率の評価と，そのフレームワークがもつ機能である複数センサの連携の基礎的な評価を行う．提案フレームワークの実装には，本研究室で開発しているセンササーバ UDSS（Ubiquitous Data Source Server）[6] を用いる．UDSS をホームネットワーク上で稼働する単体のセンサと考え，このセンサが連携して動作することを想定する．また評価では，提案フレームワークのイベント駆動方式とポーリング方式を比較する．具体的には，開発効率として各方式の実装に必要なコード数を考察し，また，複数のセンサを連携した場合の遅延時間及び通信量について定量的に比較する．

4.1 提案フレームワークの実装

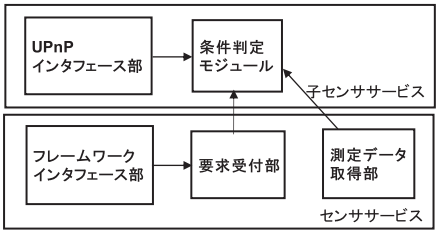
提案フレームワークを UDSS 上に実装した．UDSS は，温度・湿度・照度・音圧・焦電赤外線の 5 種類のセンサを搭載するセンサボードと ARM プロセッサと有線 LAN インタフェースを搭載する CPU ボードからなる．図 4 (a) に UDSS の外観を示す．

UDSS では，UPnP ライブラリである libupnp [17] を用いて UPnP を実装しており，提案フレームワークのイベント検出を UPnP のイベンティング機能を用いて実現している．UPnP のイベンティング機能とは，サービス上の状態変数の変化を通知する機能である．イベンティング機能では，状態変数の変化の通知を要求することをサブスクライブと呼び，その要求元をコントロールポイントと呼ぶ．

今回の実装で追加した，UDSS ソフトウェアのイベント検出機能部の構成を図 4 (b) に示す．UDSS のソ



(a) 外観



(b) ソフトウェア構成

図 4　UDSS の概要
Fig. 4　Overview of UDSS.

フトウェアは，全 5 種類のセンサで現在の測定値を要求に応じて提供する機能を提供し，焦電赤外線センサについては測定値の変化をサブスクライブ済みのコントロールポイントに対して通知する機能も提供する．本実装ではセンサの機能は，UPnP のサービスとして動作している．またセンサ固有情報の開示は，デバイスディスクリプション内にこれらの情報を記述し提供している．CAA からのイベント検出要求は，サブスクライブを用いてセンササービスに対して行う．センササービスは，要求を受け付けると，各単位条件ごとの条件判定モジュールを新たなサービス（子センササービス）として生成するものとする．UPnP ではサービスの動的な増減を考慮していない．よって今回の実装では，条件判定のための子センササービスを動的に生成するのではなく，条件判定のための子センササービスの静的インスタンスをいくつか用意しておき，要求ごとにインスタンスの中から一つを割り当て，条件判定を行う．今回の実装では子センササービスの最大数を 10 とした．

4.2 実験環境

今回実装したイベント検出機能を用いて検出する場合と，UDSS の現在の測定値を提供する機能を用いてポーリングにより検出する場合（周期的に測定値を取得し検出する場合）とで，CAA の開発効率，通信遅延，通信量を比較する．本評価においては，単純な環境上での提案フレームワークの基本的な振舞いを評価することが目的である．実環境における提案フレームワークの評価は今後の課題としたい．このために，実際のセンサの測定データの代わりに，0 から 9 までの乱数を擬似データとして用いた．擬似データは一定時間変化しないと仮定し，擬似データの更新のたびに 5 秒から 35 秒までの乱数で，次回の擬似データ更新時刻を決定した．イベント検出条件としては二つ，四つ，八つの単位条件を AND によりつないだイベント検出条件を想定した．センサを起動させてから 30 秒後にアプリケーションを起動させ，10 分間ログをとりながら，同じ条件のイベントを検出し続けた．今回，検出を行ったイベントの条件を表 1 に示す．表中の $V_i (i = 1, 2, \ldots, 8)$ は乱数による擬似データである．ポーリングによる実験では，5 秒に 1 回問合せを行った．また，CAA はすべての条件にかかわるセンサに対して，同時にポーリングを行うようにした．すなわち，条件にかかわるセンサに対して同時にコネクション確立し，ポーリングの処理を行うようにした．ブロードキャストなどを用いて一度に問い合わせる方法も考えられるが，パケット損による影響があり信頼性の点で公平に評価するためにこのような方法を採用した．

表 1 実験で用いたイベント要求条件
Table 1 Requested event conditions used in experiment.

番号	条件数	条件
条件 (1)	2	$V_1 > 4.0$ AND $V_2 > 2.0$
条件 (2)	4	$V_1 > 4.0$ AND $V_2 > 3.0$ AND $V_3 > 2.0$ AND $V_4 > 1.0$
条件 (3)	8	$V_1 > 3.0$ AND $V_2 > 2.0$ AND $V_3 > 1.0$ AND $V_4 > 1.0$ AND $V_5 > 1.0$ AND $V_6 > 1.0$ AND $V_7 > 1.0$ AND $V_8 > 1.0$

表 2 アプリケーションのコード行数
Table 2 Number of source code lines in CAAs.

	条件 (1)	条件 (2)	条件 (3)
イベント検出	124 行	124 行	124 行
ポーリング	124 行	148 行	196 行

4.3 実験結果

4.3.1 開発効率

アプリケーション開発効率の評価として，表 2 に各条件での CAA のコード行数を示す．センサによるイベント検出機能を利用したときの条件 (1)，条件 (2)，条件 (3) の場合で，コード行数が変わらないのは，検出するイベントの条件部分の記述が違うのみで，その他の部分は同じコードであるからである．一方，ポーリングによる検出においては，条件が増えるほど，コード行数も増えていることが分かる．この理由は，条件判定及び検出を CAA 側で実装する必要があり，その部分のコードが条件数に応じて増えたためである．このことは，条件が複雑になるほど CAA での処理も複雑となり，開発者の負担を増加することにつながる．提案するフレームワークを利用することで，複雑な条件であっても CAA 上での条件部分の記述を変更するだけで対応できる．

アプリケーション開発効率とは別の指標として，提案フレームワークを含めたコード量や，センサに要求される処理性能が考えられる．これらの指標においては，必ずしも提案フレームワークがポーリング方式や連携のない単純なイベント駆動方式と比較すると劣っているといえる．しかしながら，本研究で想定するホームネットワークにおいては，コード量の増加を許容してアプリケーションの開発効率を優先させることができると考えている．

表 3　遅延時間の結果（秒）
Table 3　Result of delay time (s).

	条件 (1)		条件 (2)		条件 (3)	
	平均	分散	平均	分散	平均	分散
ポーリング	6.14	2.17	5.89	4.57	6.66	14.83
提案方式	0.81	0.56	2.30	8.50	-	-

表 4　通 信 回 数
Table 4　Number of transmitted control messages.

	条件 (1)	条件 (2)	条件 (3)
ポーリング	466 回（問合せ:234 回, 応答：232 回）	952 回（問合せ:476 回, 応答：476 回）	1895 回（問合せ：951 回, 応答:944 回）
提案方式	57 回（サブスクライブ要求：2 回，イベント通知：55 回）	57 回（サブスクライブ要求：4 回，イベント通知：53 回）	125 回（サブスクライブ要求：8 回，イベント通知：117 回）

4.3.2 遅 延 時 間

遅延時間の評価を表 3 に示す．本評価における遅延時間とは，すべてのイベントの条件が成立したときから CAA にその通知が届くまでの時間を表している．表 3 では条件 (1)，(2) における遅延時間の平均と分散を示す．なお本評価では提案方式における条件 (3) は計測上の都合により省いている．表 3 から，提案方式はポーリング方式と比べて，遅延時間を削減できていることが分かる．条件数によらずポーリング方式における平均遅延時間は，2.5 秒程度（ポーリング周期の 1/2）であることが推測できるが，条件判定の処理時間，UPnP のオーバヘッド，同時にポーリングすることのオーバヘッドにより，この推測値より結果は増加している．一方，提案方式では，条件数に依存して遅延時間が増加していることが分かる．これは，最後に条件が成立するセンサから CAA までに存在するセンサの数に依存するためであり，条件数が増えれば遅延時間が増加することになる．また，提案方式の方がポーリング方式と比べて分散が大きいことが結果より分かる．これは，センサ間での遅延時間のばらつきが，センサを経由することで累積されていくためである．

4.3.3 通　信　量

通信量の評価として，表 4 に各条件における実験で要した通信の種類と回数を示す．各通信における必要なパケットサイズは，イベント通知におよそ 560 バイト，ポーリング問合せに 520 バイト，応答に 470 バイトである．通信の種類によらずほぼ同じ程度のパケットサイズが必要であることが分かる．

提案フレームワークでは，ポーリングと比較して，条件 (1)，(2)，(3) のいずれの条件においても通信量を削減できている．特に，条件 (3) の場合では，通信量を 6%程度に削減することができた．この結果から，条件 (3) のようなイベントが 10 分間に 117 回通知される環境では十分に性能を発揮できていることが分かる．ただし擬似データの更新を平均 25 秒で行い，かつポーリング周期を 5 秒とした場合との比較であることに注意が必要である．**1.** で述べたようにイベントの発生頻度によってイベント駆動方式とポーリング方式の得手不得手が変化する．本実験では，イベントの平均発生頻度がポーリングの頻度に比べて低い状況に相当するため，イベント発生頻度が高くなるにつれイベント駆動方式の通信量が増加し，ポーリング方式の通信量と逆転する．例えば，条件 (1) において，イベント更新頻度が 10 倍に増えた場合では，単純に考えてイベント通知が 550 回になり，ポーリングの通信量と逆転する．

また，ポーリング方式において，論理積により接続された条件なので，すべてのセンサへ同時にポーリングを行うのではなく，順番に問合せを行い，条件を満たさないセンサがある間は，他のセンサへのポーリングを停止することが可能である．このような手法により，今回実験で得られた通信回数よりいくらか通信量を削減できた可能性はあるが，CAA のコードは更に複雑で大規模なものになることが予想される．

5. 関 連 研 究

本章では，センサを用いたイベント検出の関連研究を紹介する．文献 [5] では，様々なコンテクストアウェアシステムが紹介されている．特に，ミドルウェアやフレームワークに焦点を絞り，システムのアーキテクチャやコンテクストモデルなどの観点から解析している．本章では，特にコンテクストの推定とその変化の検出に着目する．

文献 [7] では，MobiPADS システムが提案されている．MobiPADS は，CPU，ネットワーク，バッテリーなどの状況を AND や OR で結合することでコンテクストを構築し，それらのコンテクストに応じたサービスの配置や再構築を可能にする．指定したコンテクストの変化は，イベント通知により検出することができる．コンテクストを構成する状況ソースとして，同一端末内で発生するものが想定されており，本研究が想

定しているような状況ソースがネットワーク上に分散している状況は考慮されていない．本研究の想定環境においては，状況ソースはセンサとほぼ同義である．

分散したセンサの測定データを用いてコンテクストを抽出する研究を紹介する．ハードウェア的なセンサにより測定された情報からコンテクストを扱う研究として，センサネットワークをセンサにより測定される情報のデータベースとみなし，センサの測定データを要求するクエリを SQL のような形式で記述するという手法が提案されている [8]．この文献ではコンテクストの変化を表すセンサの測定データに現れるイベントの検出も考慮されている．しかし，自然環境のモニタリングシステムに代表される大規模なセンサネットワークを想定しており，センサ数は多いがセンサノードは数種類に限られる．これに対して本研究では多種多様なセンサからなるホームネットワークを対象としている．文献 [9] では，自然言語に近い形でイベントとそれに伴う機器の動作を指定したルールを表現するルール記述言語を提案している．人間にとって分かりやすいルールの記述が可能であるが，イベントとなり得るのは，デバイスの各項目に対する大小関係比較，あるいは線形不等式に制限されている．また文献 [10] では，ユーザの行動履歴をもとにサービスの起動ルールを学習するシステム Synapse について述べている．このシステムでは人の入室など通常イベントとしてとらえられる事象のほか，現在の室温値などもイベントとしてデータベースに蓄積しておき，ユーザがサービスを起動した時刻のイベントと対応づけることでルールを学習する．文献 [11] では，ソフトウェア工学のアプローチを用いて CAA の設計と実装を容易にするフレームワークが提案されている．特に，AND や OR を含む，より高い表現能力をもつコンテクストモデルに焦点があてられている．これらの研究では，分散したセンサの測定データからコンテクストを抽出できることを示している点で有用な研究といえるが，センサの電力消費などの効率に焦点をあてたものではない．いずれも集中管理型を想定しており，ネットワークの負荷分散や耐故障性の観点から効率的であるとはいえない．

また本研究と同様に，分散しているセンサ上でイベントを検出し，最終的なコンテクストの変化を検出するミドルウェアが提案されている．文献 [12] や文献 [13] では，publish/subscribe/notify インタフェースを用いてイベント検出を実現するシステムが提案されている．しなしながら，AND や OR で接続されたイベント検出をネットワーク上でどのように構成するかは議論されていない．すなわち，それらの検出を行うインタフェースが提供されているだけであり，具体的な構成方法はユーザにゆだねられている．また，DSWare [14] や Impala [15] は，センサネットワーク上で，センサが何らかの異常を感知（イベント）したとき，即座にユーザへ通知するイベント処理指向のミドルウェアである．広域のセンサネットワークでは，途中のセンサノードでデータの集約などの加工をするものであり，本研究における AND 条件を直列接続によって実現する方法も，ある種の集約と考えることができる．この点において，本研究と同じ目標をもっている．ただし，単位条件を直列接続するにあたっては，ホームネットワークでは接続順序に対する物理的な制約がないため（すなわちマルチホップではないため），通信量や消費電力などの効率化を目的とした最適化の余地がある．本フレームワークにおいては，予想成立頻度が低いものほど常時稼動させ，高いものは頻度の低い条件の成立を待つ間休止させる．これにより通信量と消費電力に関する効率化を実現する．

6. むすび

本論文では，コンテクストの変化を検出する処理の一部をセンサ側で行い，イベント駆動型で効率良くコンテクスト変化を検出するためのフレームワークを提案した．本フレームワークは，ホームネットワーク上に存在する多様なセンサを抽象化し，センサを利用するコンテクストアウェアアプリケーションに対して統一的なインタフェースを提供する．また，分散する複数のセンサ間で連携して，複数のセンサの測定データに対する条件で表されるイベントも効率良く検出する仕組みを提供する．また，提案フレームワークによりセンサに求められるイベント検出機能の UPnP を利用した設計と実装について述べ，その実装結果を示した．本論文で提案したイベント検出機能を利用することで，通信量を少なく抑えながら十分な即応性を実現できることが示せた．

今後の課題としては，連携中のセンサの故障への対応が挙げられる．連係動作中に一部のセンサが再起動された場合など，連携する他のセンサやアプリケーションがそれを検知し，もとの状態に復帰する仕組みが必要である．また，複数のアプリケーションからの要求に対応するセンサが，内部で複数の条件判定を集

約するなど，センサ内部でのチューニングにより更に効率的にイベント検出を実現することも挙げられる．更に，実環境に即した評価実験も今後の課題である．

謝辞　本研究の一部は，科研費及び次世代研究スーパースター養成プログラム（九州大学総長裁量経費）による助成を受けている．

文　献

[1] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, and J. Altmann, “Context-awareness on mobile devices – The hydrogen approach,” Proc. 36th Annual Hawaii Int. Conf. on System Sciences, pp.292–302, 2003.

[2] A.K. Dey and G.D. Abowd, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” HCI, vol.16, no.2–4, pp.97–166, 2001.

[3] P. Prekop and M. Burnett, “Activities, context and ubiquiouts computing,” Comput. Commun., vol.26, no.11, pp.1168–1176, 2003.

[4] R.M. Gustavsen, “Condor – An application framework for mobility-based context-aware applications,” Proc. Workshop on Concepts and Models for Ubiquitous Computing, 2002.

[5] M. Baldauf, S. Dustdar, and F. Rosenberg, “A survey on context-aware systems,” Int. J. Ad Hoc and Ubiquitous Computing, vol.2, no.4, pp.263–277, 2007.

[6] T. Matsuura, K. Hisazumi, T. Kitasuka, T. Nakanishi, and A. Fukuda, “UDSS: Sensor device for context awareness in home network,” Proc. Fourth Int. Conf. on Networked Sensing Systems (INSS) 2007, pp.196–200, 2007.

[7] A.T.S. Chan and S.-N. Chuang, “MobiPADS: A reflective middleware for context-aware mobile computing,” IEEE Trans. Softw. Eng., vol.29, no.12, pp.1072–1085, 2003.

[8] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, “The design of an acquisitional query processor for sensor networks,” Proc. 2003 ACM SIGMOD Int. Conf. on Management of data, pp.491–502, 2003.

[9] 西垣弘二，安本慶一，柴田直樹，伊藤　実，“コンテキストに基づいた情報家電の連携を実現するためのフレームワークおよびルールベース言語の提案,” 情処学 UBI 研報, vol.2004, no.112, pp.21–27, 2004.

[10] 川原圭博，司　化，猪鹿倉知広，登内敏夫，森川博之，青山友紀，“行動履歴と制約条件を考慮した情報家電制御機構,” 情処学 UBI 研報，vol.2006, no.14, pp.55–60, 2006.

[11] K. Henricksen and J. Indulska, “A software engineering framework for context-aware pervasive computing,” Proc. Second IEEE Annual Conf. on Pervasive Computing and Communications (PERCOM’04), pp.77–86, 2004.

[12] A. Ranganathan and R.H. Campbell, “An infrastructure for context-awareness based on first order logic,” Personal Ubiquitous Computing, vol.7, no.6, pp.353–364, 2003.

[13] J.E. Bardram, “The java context awareness framework (JCAF) — A service infrastructure and programming framework for context-aware applications,” LNCS 3468, pp.98–115, 2005.

[14] S. Li, S. Son, and J. Stankovic, “Event detection services using data service middleware in distributed sensor networks,” Proc. 2nd Int. Workshop on Information Processing in Sensor Networks, pp.502–517, 2003.

[15] T. Liu, C.M. Sadler, P. Zhang, and M. Martonosi, “Implementing software on resource-constrained mobile sensors: Experiences with Impala and ZebraNet,” Proc. Int. Conf. on Mobile Systems, Applications, and Services, pp.256–269, 2004.

[16] UPnP Forum, “UPnP device architecture 1.0,” Document Version 1.0.1, (July 2006).

[17] Linux SDK for UPnP Devices 1.2.1 (libupnp); http://upnp.sourceforge.net/

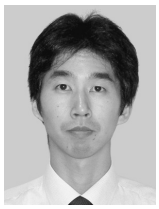
（平成 20 年 11 月 7 日受付，21 年 2 月 25 日再受付）

松浦　知子

2006 九大・工・電気情報卒．2008 同大大学院システム情報科学府博士前期課程了．主としてセンサネットワークに興味をもつ．

田頭　茂明　（正員）

1996 龍谷大・理工卒．1998 奈良先端科学技術大学院大学情報科学研究科博士前期課程了．2000 同大情報科学研究科博士後期課程了．博士（工学）．2000 広島大学工学部第二類助手．2001 広島大学大学院工学研究科助手．2007 広島大学大学院工学研究科情報工学専攻助教．2007 九州大学高等研究機構若手研究者養成部門 SSP 学術研究員，九州大学大学院システム情報科学研究院特任准教授．ユビキタスコンピューティング，システムソフトウェアの研究に従事．IEEE，情報処理学会各会員．



北須賀輝明　（正員）

1993 京大・工・情報工学卒．1995 奈良先端科学技術大学院大学情報科学研究科博士前期課程了．同年シャープ（株）入社．パーソナルコンピュータの開発に従事．2001 九州大学大学院システム情報科学研究院助手．2006 九州大学博士（工学）．2007 年 3 月九州大学大学院システム情報科学研究院助教．2007 年 10 月熊本大学大学院自然科学研究科准教授．モバイルコンピューティング，組込みシステム，並列/分散処理，コンパイラ，計算機アーキテクチャに研究的関心をもつ．情報処理学会会員．

中西　恒夫　（正員）

1993 阪大・工・通信卒．1998 奈良先端科学技術大学院大学情報科学研究科博士後期課程了．博士（工学）．1998 奈良先端科学技術大学院大学情報科学研究科助手，2002 九州大学大学院システム情報科学研究院助教授．ソフトウェアプロダクトライン，自動車ソフトウェアに研究的関心をもつ．情報処理学会，自動車技術会各会員．

福田　　晃　（正員）

1977 九大・工・情報工学卒．1979 同大大学院修士課程了．同年 NTT 研究所入所．1983 九州大学大学院総合理工学研究科助手．1989 同大学助教授．1994 奈良先端科学技術大学院大学情報科学研究科教授．2001 より九州大学大学院システム情報科学研究院教授．工博．2008 九州大学システム LSI 研究センターセンター長，教授（兼任）．オペレーティング・システム，コンパイラ，組込みシステム，モバイルコンピューティング，計算機アーキテクチャ，並列/分散処理，性能評価等の研究に従事．本会平 2 年度研究賞，平 5 年度 Best Author 賞受賞．著書『並列オペレーティングシステム』（コロナ社），訳書『オペレーティングシステムの概念』（共訳，培風館）．ACM，IEEE Computer Society，情報処理学会，日本 OR 学会各会員．