UNL

平成１４年度　研究開発成果報告書

ユビキタスコンピューティング環境を実現する基盤ネットワークプロトコルの研究開発

目次

第一章
研究開発課題の背景

20 世紀後半より，情報通信技術・IT（Information Technology）の急速な進展と広範な普及によって，我々の社会は大きく変革し，いわゆる情報社会へと突入した.我が国も情報通信技術に関しては世界を牽引した数少ない国の一つして自負するに十分であり,多くの研究開発がなされてきた．現在も次世代携帯電話を始めとして，世界に貢献する成果を輩出している.

1.1 ユビキタスコンピューティング

1990 年代からの情報通信網基盤の爆発的発展は，ネットワークの大容量化と接続機器（コンピュータ）の高性能化によって，より高度なユーザサービスを実現してきた.それと同時に,近年の我が国では,これとは異なる情報通信網の急速な発展も起きている．それは，従来は情報処理能力や通信能力を持たなかった,身の回りに存在する無数の小さな「モノ」に対して，計算力と通信力を与える方向への爆発的な拡大である.こうした身の回りのあらゆるものをインテリジェント化することで,高いユーザサービスを実現する情報通信のパラダイムは，ユビキタスコンピューティング （Ubiquitous Computing）や「どこでもコンピュータ環境」と呼ばれている．このパラダイムは，1980 年代後半に日米で同時に提唱され，その後ポスト PC 時代の情報通信技術のパラダイムとして，専門家の間で広く受け入れられている.

このユビキタスコンピューティングこそが今後の日本型の IT 技術開発のパラダイムとして有望なものであり,本研究開発課題はこのパラダイムの実現に対して正面から取り組むものである.

1.2 我が国の産業構造との関係

情報通信分野は極めて広範で深いため，単一の国や会社，組織ですべてをカバーすることは，もはや不可能である．世界全体でみた情報通信分野は,様々な国の様々な組織がそれぞれに得意な部分を分担し,世界規模で協調と競争をしながら発展していくのが健全な姿である.

現在，すでにインターネットや PC の分野の技術的主導権は米国が握っている．実際，パーソナルコンピューティング分野は，心臓部である CPU や OS といった根幹技術を"Wintel"という造語が示すように,米国の特定ベンダーの独占状態にあり,我が国の研究開発は壊滅状態である．しかし，情報通信分野で十分に大規模な収益が見込める分野は，インターネットや PC の分野だけではない．特に，ポスト PC 時代

の主力産業と考えられる情報家電,ネットワーク機能をもった電子機器に目を向ければ,ITRON(Industrial TRON: The Real-time Operating system Nucleus) を始めとして，我が国の独自技術が世界的に強い競争力を維持し続けている.我が国の産業構造からみて得意な部分とは,小さく緻密な機器を生産するところにある．こういった視点からも,効率的な研究開発投資を考えると,むしろ,我が国が最も得意な分野を更に発展させることを目指すべきである.こうした技術は今だ世界の先端を走っており，この点を活かした新しい産業を創造し，世界をリードする分野を積極的に開拓すれば,当該分野の世界的イニシアチブを獲得することも可能である.

1.3 緻密でクリーンな IT 技術開発への要求

21 世紀を迎え，光ファイバー網を使ったインターネット等が目指している，より速く・より大容量・より広帯域を追求する情報通信技術の重要性は高いものの,現在それに加えて更に,次のようなより緻密でクリーンな情報通信技術への要求も高まっている．第一に,豊富な容量・帯域・速度をもった IT 基盤上のデジタル情報の流れを，人間や社会の意志に基づいて確実に制御できること.権利がある人にだけに情報のアクセスを許可し,権利の無い人の不正な盗聴を防ぐこと,また,不正な情報複製ができないようにするといったことが,確実に行えることが重要である．従って，これには，広い意味での，暗号技術や認証技術などが含まれる．第二に，省電力をはじめとして，資源を浪費せず,環境への悪影響を最小限にとどめる,クリーンな情報通信技術．こうした考え方は，カームコンピューティング（Calm Computing）とも言われる.

1.4 セキュアコンピューティング

2001 年は，我が国でもサイバーテロが行なわれた．また，Nimda, CodeRed, Circum といったインターネットを介して大規模に感染するコンピュータウィルスやワームも発生した.既に米国では我が国以上にサイバーテロが行なわれ,情報社会を脅かしている．今後もこうした危険性は確実に拡大するだろう.現在の情報通信インフラで解決することが最も要求されている課題は,こうした攻撃に強い,セキュアな情報通信基盤である．しかもそれが,一般素人でも簡単に扱うことができなければならない．インターネットの当初の設計方針には,イ

ンターネットを通じて通信する者同士が信頼できないようなこと，また，これほどまでに素人ユーザの割合が多くなることは，想定されていなかった．現在これに対して抜本的対策を施さない限り，かつての公害問題のように，将来の情報社会に禍根を残すことになりかねない．

第二章
研究開発の全体計画

2.1 研究開発課題の概要

本研究開発課題は，我々の身の回りの，あらゆるものにマイクロコンピュータと通信機能を組み込み，それらが互いに情報を交換しながら協調動作を行い，人間生活をより高度にサポートする，ユビキタスコンピューティング環境を構築するための，次世代通信の基盤プロトコルおよびそのシステムを確立することである．

2.1.1 ユビキタスコンピューティング環境が目指す最終目標

ユビキタスコンピューティング環境とは，身の回りのあらゆるものにマイクロコンピュータと通信機能を組み込み，それらが互いに情報を交換しながら協調動作を行い，人間生活をより高度にサポートする環境のことである．今まで，こうした環境を使った IT の多様な「夢」が語られており，その「夢」を実現することが本研究プロジェクトの最終目標である．

例えば，家庭では，家に設置された温度センサーが常に外気温と室内温度を監視しており，居住者が部屋の温度を下げようとした時，もしも外気温が室温より低ければ窓を開ける．しかし，部屋でピアノを弾き，外部に騒音が漏れたら，窓を閉めて自動的に空調が入る．また，自家用車で帰宅するときに，自動車のナビゲーションシステムから，到着時刻に合わせて自宅の風呂の湯を沸かすこともできる．

こうした環境を実現するためには，膨大なコンピュータを身の回りのあらゆるものに埋め込み，人間自身も常にコンピュータを携帯し，それらが互いにネットワークで接続され，情報交換しながら協調処理を行うメカニズムが必要とされる．我々は，この埋め込まれたり携帯されるコンピュータを**「インテリジェントオブジェクト」**とよび，これらを接続する通信メカニズムのことを**「ユビキタスネットワーキング」**と呼ぶ．

本研究では生活空間を構成する大量のインテリジェントオブジェクトからなるネットワークを想定している．このネットワークと他の既存のネットワークとの違いは，まずネットワークにつながるノードの数が桁違いに多いことである．一人当たり数十から数百のプロセッサがある高密度のユビキタスコンピューティング環境のなかから，通信すべき適切なコンピュータを指定するためにはどうすればよいか．更にそれが何百，何千もの人が活動するビルや都市，最終的には世界までつながった時に，このユビキタスネットワーキングがどのように

展開されるべきか,といったことが重要な課題となっている.従って,本研究における中心課題は,この「ユビキタスネットワーキング」の根幹となる基本方式を明らかにし,更にそれを動作させるシステムを構築することである.

これらの多くのインテリジェントオブジェクトを協調させるためには,調停動作の実現がポイントである.例えば,一億個のコンピュータがネットワークにつながった場合,全部のデータを手に入れそれに基づいて中央で方針を決定するという,中央集権的な手法で全部の動作を最適化することはもはや不可能ではないかと考えている.そのためには何らかの分散的な最適化方式を考案する必要がある.

生活の場におけるインテリジェントオブジェクトは個々のエンドユーザの都合でネットワークに突然追加されたり,はずされたり,次の日には別の箇所につながったり,ということが起こる.そのような**「アドホック(ad hoc)」**性をもったネットワークを実現しなければならない.その際に一般の人でも扱え,面倒なオペレーションが不要な**「エフォートレス(effortless)」**な性質を持つ必要がある.ユビキタスコンピューティング環境において,無数のコンピュータをちりばめた時に重要なことは,その上で,これらのコンピュータ群が 24 時間 365 日正常動作するように運用できることである.そのためには,ユビキタスコンピューティング環境を構成するシステムと社会との親和性,運用技術に対する研究も重要となる.

2.1.2 技術課題の概要

本研究では,このユビキタスコンピューティング環境の基盤技術となる通信プロトコル(**ユビキタスネットワーキングプロトコル**)や,それを用いた通信網基盤の構築技術の研究開発を行う.そのために以下の研究開発項目を実施する.

1. リアルタイム通信プロトコル
2. セキュアネットワーキング,セキュアコンピューティング
3. コンパクト性
4. エフォートレスオペレーション,エフォートレスマネジメント
5. ユーザとの親和性
6. 省リソース
7. 既存通信網との親和性
8. 高度な協調・調停動作による人間生活の支援機能の実現

(1) リアルタイム通信プロトコル

ユビキタスコンピューティング環境を実現するための基本プロトコルには，人間の振る舞いや生活・社会を構成するあらゆる事象に追従して応答できるための，（ソフト）リアルタイム性が必要である．特に，身の回りのインテリジェントな機器（アクチュエーター）を制御する部分には，より強いリアルタイム性が要求される．

（２）セキュアネットワーキング，セキュアコンピューティング

先に述べたユビキタスコンピューティング環境による夢，例えば，未来の ITS（Intelligent Transportation System）のイメージとして，自動車のナビゲーションシステムから，到着時刻に合わせて自宅の風呂の湯を沸かすといったシナリオが描かれてきた．実際に，このシナリオを実現するためには，悪意ある他者が自宅の風呂を操作することを防げなければならない．更に近年は，サイバーアタックやクラッキング，サイバーテロを想定した対策も求められている．ユビキタスコンピューティング環境を，ネットワーク経由のアタックから守るためには，セキュアな通信プロトコル，セキュアな通信システムの開発が不可欠である．ネットワーク基盤の遍在化（ユビキタス化）が急速に進んでいる現在，セキュリティーを向上させる技術開発は急務である．

（３）コンパクト性

ユビキタスコンピューティング環境では，非常に膨大な数の小さな機器にコンピュータや通信機能が埋め込まれる．従って，各ノードが小さくコンパクトであること重要である．計算機能や通信機能の偏在性（ユビキタス性）を高めるためには，各ノードがコンパクト化できることが不可欠である．

（４）エフォートレス（Effortless）

ユビキタスコンピューティングのためのネットワークプロトコルを実現する上で重要なことは，コンピュータに詳しくない一般利用者でさえも，自身が所有する機器の安全性，蓄積情報や通信の秘匿性等を手軽に確保できることである．現在でも多くのセキュアプロトコルがあるが，そのほとんどが，認証や暗号のための鍵の管理や認証局（CA 局）からの証明書の取得といった，専門知識を要する運用作業を伴うため，普通の人が手軽に使えるものになっていない．そこで，本研究課題では，耐タンパ性を有するハードウェアを利用することによって，より手軽で簡便なセキュア通信プロトコルを開発する．このプロトコ

ルによって,簡単に使えるパッケージ化された強固で安定した汎用セキュリティー基盤が実現され,コンピュータに詳しくない普通の人でも，セキュアな通信基盤の恩恵に浴することができる.

（５）ユーザとの親和性

ユビキタスコンピューティングは,人間の身の回りに計算機能や通信機能を埋め込むことで，人間生活をサポートする．そこで，重要な観点の一つは,どのように人間をサポートしていくかということである.ユビキタスコンピューティング環境においてこうした人間との境界部分，つまりヒューマンインタフェースをつかさどる分野は，**強化現実環境**（Augmented Reality）とか，**複合現実環境**（Mixed Reality）といった分野となる．本研究でも人間にとって，より自然でかつ利便性の高いサポートの手法の研究開発を進める.

（６）省リソース

ユビキタスコンピューティング環境では,ノード数が従来の分散環境やインターネット環境にもまして膨大な数になることから,一つのノードが使う電力量など,消費する各種リソースが小さくなるようなCalm Computing 技術を確立する．従来の情報通信技術は，性能や利便性を最大化するための研究開発に重きがおかれており,省資源を最大化するといった基準に則った研究開発は比重が低かった.本研究課題で実現するプロトコルやシステムでは，こうした省電力・省資源に取り組む.

（７）既存通信網との親和性

現在，IP による世界的ネットワークが構築されている．その他にも既存の通信網には,携帯電話網,通常の加入電話網をはじめとして,多様なものが存在する．これらは，性能，サービス，保守の容易性といった点で利害得失があり,これらが単一のプロトコルに統合されるとは考えづらい.ユビキタスコンピューティング環境を構成する通信プロトコルに関しても同様に,様々な利害得失をもった多様なプロトコルが混在する環境を前提とし,互いに補完的な関係になることが理想的である.

そこで，本研究開発課題では，こうした既存の多様なプロトコルとの相互接続によって,柔軟な情報交換や制御を行うメカニズムを確立する．具体的には，IP に基づくインターネット，携帯電話網上に構築されている情報通信網基盤と,ユビキタスネットワーキングプロト

コルとを相互接続する，ゲートウェイ技術を確立する．それは単に，ネットワーク層やトランスポート層によるゲートウェイだけではなく，セッション層やアプリケーション層にいたる，ほぼ全ての層において接続する必要があり，そのための統合的なゲートウェイ技術を構築する．

（８）高度な協調・調停動作による人間生活の支援機能の実現

ユビキタスコンピューティング環境では，単に多くのノードがあるだけでなく，それらが「協調」「調停」動作をすることで，人間生活を高度に支援する．例えば，部屋の温度が上ったら，窓を自動的に開け，もしも部屋の中でピアノをひきはじめ騒音が発生したら，窓を自動的に閉めて空調を入れるといった動作である．こうした協調・調停作業が，あちこちに埋め込まれた膨大な数のコンピュータの間で実施できなければならない．そのための通信システム，交換情報形式，超機能分散型ソフトウェアのためのプログラミングシステムなどの研究開発を行う．

2.1.3 研究実施計画の基本方針（概要）

ここでは，2.1，2.2 で示した本研究の目標と課題を達成する実施計画の方針の概要を示す．

（１）システム単位のサブプロジェクト構成

本研究開発課題における技術的ボトルネックは，いかに大量の小さいノードを，特定の目的に沿って協調動作させるかという，システム統合技術にあると考えている．そこで，要素技術毎に細分化したサブテーマわけをした場合，最後に統合化して相互接続環境を構築するあ困難になるため，次の通り「機器」による切り分けを中心としたサブテーマわけを行う．

1. セキュアハードウェア
2. 通信システム
3. ユーザ側のエンドノードシステム（モバイル端末／情報家電／PDA 等）
4. サーバ側のエンドノードシステム（認証サーバ／アプリケーションサーバ等）
5. システム統合技術
6. 超機能分散システム指向の開発環境

（２）システム工学的検証の重視

ユビキタスコンピューティング環境は,単に情報通信のメカニズムだけを研究開発するだけでは成功しない.ユビキタスコンピューティング環境は，人間・社会生活に無数に埋め込まれ，社会活動や生活を支援するものであるため，技術的に優れていても，例えば騒音や発熱の程度によっては人間生活にはなじまない.公共の場に設置するものは，多少の悪戯や荒い扱いにも耐えなければならない．膨大な数のノードが現実社会の中で容易にかつ安全に運用できるのか,無数に埋め込まれたインテリジェントオブジェクトのメンテナンスは可能なのか,ユビキタスコンピューティング環境のセキュリティーは厳密に運用できるのか，といった諸問題がある．

そこで，本研究開発課題では，こうした社会や生活と，ユビキタスコンピューティング環境の間の親和性を重視し,本研究開発成果を実用に耐えるシステムとして完成させるために,システム工学的見地からその検証を行う．検証項目としては，①システムの信頼性の検証，②運用評価，③ユーザビリティ，④スケールファクターのシミュレーション，⑤環境アセスメントを計画している．

2.1.4 研究実施計画の詳細（システム部分）

システム面の研究のサブテーマは，「2.3 （１）システム単位のサブプロジェクト構成」の部分で述べたとおり，機器種類毎に切り分けを中心とする．サブテーマとしては，以下を計画している．

【サブテーマ１】
セキュアコンピューティングの基盤となるセキュアハードウェア

【サブテーマ２】
基盤通信システムの研究開発

【サブテーマ３】
ユーザノードシステムの研究開発

【サブテーマ４】
サーバノードシステムの研究開発

【サブテーマ５】

ユビキタスコンピューティング環境を構成するシステム統合技術の研究開発

【サブテーマ６】
超機能分散システム指向の開発環境の研究開発

【サブテーマ１】
セキュアコンピューティングの基盤となるセキュアハードウェア

通信のセキュリティーは一般的にはソフトウェアだけで確保することはできない.何らかのハードウェアによる情報保護が不可欠である. 従来型の情報システムでは, 機材が設置されている建物や部屋に対する入退館管理等による物理的な保護が前提であった.ユビキタスコンピューティング環境では,公共の場に露出して設置されたものも対象であり,ユーザが常に携帯し頻繁に紛失や盗難が起きるものまで含まれる.しかも,前述したようにセキュリティーを確保するために,ユーザが暗号･認証の仕組みを理解することが必須であってはならない.

そこで, 本研究では, LSI 自体に不正アクセスが加えられないように加工を施した, いわゆる**「耐タンパー性（Tamper Resistance)」**を有するハードウェアを,ユビキタスコンピューティング向けチップとして新規に開発し,それをユビキタスコンピューティング環境のセキュアシステムの基盤パーツとする.

【サブテーマ２】
基盤通信システムの研究開発

(2-1) 基盤プロトコル概要

基盤通信システムのサブテーマでは,ユビキタスコンピューティング環境を構成する通信システム全般を扱い,本研究の核であるユビキタスネットワークプロトコルの研究,そのプロトコルスタックの開発,ルーターをはじめとした各種ネットワーキング装置を含む.ユビキタスコンピューティングシステムにおいては,その目的に最適化したネットワークアーキテクチャを導入する. 現時点では, 研究対象となるユビキタスネットワークのアーキテクチャと機能は以下の特徴をもつものを想定しているが, これは研究の進捗・進展や, 他の技術動向

に応じて変化する部分もある.

(2-2) データリンク層

データリンク層は既存の方式を用いる.例えば,2.5GHz および 5GHz 帯の無線 LAN, Bluetooth, ISO 14443, PHS, 第 3 世代の移動体通信ネットワークなどである. これらの方式としては,端末と端末が直接通信するアドホックモードの通信形態と,基地局およびバックボーンネットワークを介した通信形態の双方を検討する.

(2-3) ネットワーク層

ネットワーク層はユビキタスネットワークに適した新しい方式を考案して実現する. 通信形態は,ユニキャストとマルチキャストをサポートし, それぞれ帯域保証や優先制御を行うリアルタイム通信と,ベストエフォート通信を扱う.帯域保証等を行うリアルタイム通信の場合は, そのためのシグナリングを提供する.

ここでは, Layer 2 ARP (Address Resolution Protocol) が必要である. ユビキタスネットワーキングでは, IP で使用される ARP とは異なり,実世界上の位置や社会的なセマンティックスなどに基づいたノード指定に対応する (デ**バイス・ノードルックアップ機能**).

ネットワーク構成やノード間の帯域などのルーティング情報を交換するルーティングプロトコルを実現する. このプロトコルは,ユニキャストのためのプロトコルと,マルチキャストのためのプロトコルの双方, またダイナミックな変化に追従できる柔軟性が必要となる.

更に, ネットワークにおける輻輳や障害等を通知するための OAM (Operation Administration and Maintenance)情報交換プロトコルを備える. このプロトコルは, ネットワーク経路上の故障に加え, リアルタイム通信のための輻輳の検知やマルチキャストにおける障害情報の転送などを, 統合的にサポートする.

(2-4) トランスポート層

トランスポート層のプロトコルとしては,コネクション型とコネクションレス型のプロトコルを用意する必要がある.双方のプロトコルとも,遅延変動や誤り率などのサービス品質に関して,アプリケーションが要求する品質を最小限の機能で実現するサービス品質機能を有する. コネクション型のプロトコルはユニキャストを対象とし, コネクションレス型のプロトコルはユニキャストとマルチキャストの双方を対象とする. また, 通信相手の指定については, アドレスを指

定する方式のほかに,要求条件を指定する方式についてもサポートする．確認応答を有し，信頼性の高い通信を可能とする.

(2-5) セッション層

セキュリティーや認証のための機能を提供する.相互認証と同時に鍵交換を行い,セッション中は暗号通信が行なわれるセキュアセッション機能,またロールバック機能を備えたトランザクションセッション機能,リアルタイム応答が要求される場合のライトウェイトセッション機能を備える.

(2-6) アプリケーション層

アプリケーション層は,各種応用に対応するプロトコルを用意する.その他に,通信のサポートのために各種応用から共通に使用されるプロトコルを用意する必要がある．1 つは，サービスルックアップ機能で,サービス名からそれを提供するノードの物理アドレスを検索するなど，各種のネットワーク構成情報の検索プロトコルを構築する（**サービスルックアッププロトコル**）．また，ネットワーク機器の状態監視や構成変更などを遠隔で行うネットワーク管理用プロトコルも備える.

【サブテーマ３】
ユーザノードシステムの研究開発

ユビキタスコンピューティング環境を構成するノードの中で,ユーザと直接接することが想定される機器類の研究開発である.これをここではユーザノードと呼ぶが，想定されるユーザノードには，ユーザが携帯する移動ノードと，生活環境に設置される固定ノードがある.双方とも,**【サブテーマ２】基盤通信システムの研究開発**で開発された,ユビキタスネットワーキングプロトコルを搭載する.移動か固定かに応じて，利用可能な計算・通信資源や，物理通信路の環境，物理的な大きさ，それに基づくユーザインタフェース等が異なるために,それぞれ固有の実現技術が必要とされる.本サブテーマでは，こうした条件に適合した様々なユーザノードの構成方法･機構を中心に研究開発を進める.

(3-1) 移動ノード

移動ノードや,常にユーザが携帯してユーザとユビキタスコンピューティング環境との間のインタフェースの役割を担う端末である.具

体的には，PDA（Personal Digital Assistant），携帯電話スマートカードなどが想定される．固定ノードと比べた場合，移動ノードに関する研究課題として以下がある．

- 物理通信路は基本的に無線通信であり，ユーザの移動を考慮すると通信品質は安定しない，
- 紛失・盗難が起こる可能性が高いため，それに備えたセキュリティーメカニズムを備えること，
- 物理サイズが小さいため，それに適した洗練されたユーザインタフェース，
- 計算資源も限られているため，コンパクト性が求められる．
- バッテリー量の制約も大きいため，徹底した省電力機構．

移動ノードでは，こうした条件をクリアした中で，ユビキタスネットワーキングプロトコルの実現技術を確立する．

(3-2) 固定ノード

固定ノードは，ユビキタスコンピューティング環境に設置され，人間の振る舞いや，環境の変化に応じて柔軟かつ緻密に動作するインテリジェントオブジェクトである．具体的には，住宅内にある電子機器類，オフィスにあるコピー機やファックス，シュレッター―といった機器，また公共の場に設置された券売機，自動販売機，チケットゲートといった機器を想定している．移動ノードと比べた場合の固定ノードに関する研究開発項目の特徴は，以下の通りである．

- 物理的認証をうけない不特定多数によって利用されることが前提となり，そのための認証機能，セキュリティー機能が必要とされる．
- 特に公共の場に置かれた機器は，ネットワーク的にも公共的なセグメント上に置かれることになり，その場合にセキュア通信の確保が必要である．
- ユーザノードであると同時に，サーバ的な機能の提供も求められる．
- 回線や電源の状況は良い環境にある．

固定ノードでは，こうした特質を前提とした，ユビキタスネットワーキングプロトコルの実現技術を確立する．

【サブテーマ４】

サーバノードシステムの研究開発

サーバノードは，ユビキタスネットワーキングプロトコルを実現し，

特にユーザノードを対象としてネットワークサービスと機能を提供するノードである．具体的には，①セキュアセッションのための認証局，登録局，②分散トランザクションを支えるトランザクションサーバ，③サービスルックアップやデバイスルックアップのためのネットワーク環境サーバ，④ネットワーク管理のための管理サーバ，⑤各アプリケーションに依存したアプリケーションサーバなどが想定される．

従来の電子商取引分野の研究開発の経験により，サーバの運用者側の不正行為も問題とされており，サーバ側も耐タンパー性を持ったハードウェア構成が必要である．かえって移動型のユーザノードのように小型機器の方が，対タンパー性を実現することが容易であり，サーバノードの場合は，大きなノードにおいての耐タンパーハードウェアの実現技術が課題である．

また，ユビキタスコンピューティング環境においては，サーバにおいても，セキュア性を保ちながら，リアルタイム性を実現できるに十分な高応答性能を実現する必要がある．そこで，本研究開発課題では，サーバをセキュアに性能向上させるための，セキュアクラスタリング，暗号・認証処理機能を持ったデータキャッシュ・プロセスマイグレーションのメカニズムを確立する．特に，動的な情報更新が可能な高応答性を達成できるディレクトリサーバを実現する．

【サブテーマ５】

ユビキタスコンピューティング環境を構成するシステム統合技術の研究開発

ユビキタスコンピューティング環境は，様々なプロトコルを用いる様々な膨大な機器がヘテロジニアスに結合したシステムであり，それを統合し協調動作させるための技術を確立する．その統合技術を要素技術に分割すると，以下の項目が挙げられる．①通信環境を意識する必要のない確実なコネクティビティおよびサービスの実現，②ネットワーク運用状況，サービス実行状況に応じた，最適なリソース配分によるコンパクトネットワーク／サービス実行環境の構築，③通信環境に最適化したサービス実行メカニズム，④ネットワークへのノードの簡単な装着／脱着と移動時のサービス継続の開発．

上記の課題を実現するときに，本研究開発課題で開発するプロトコルと既存の IP や電話網を使った通信プロトコルの相互運用をスムーズに行うため，次の基本技術の開発を目指す．①アドレスを陽に指定

しないアドレス解決およびルーティングメカニズム,②プロトコル変換メカニズム,③複数の異なるネットワークをまたがったときの品質制御メカニズム，④端末と連携したサービス実行メカニズム.

【サブテーマ６】

超機能分散システム指向の開発環境の研究開発

ユビキタスコンピューティング環境を構築する上での技術的な課題として，システム開発効率がある. ユビキタスコンピューティング環境は，他の分散環境と比べると，膨大なノード数に特徴がある. 現在の計算機科学では,ここまで分散化された多数のノードを協調動作させるソフトウェアを効率よく開発する手法が存在しない. しかも，各ノードはリアルタイムプログラミングとセキュリティーという,単独でも困難なプログラミングを施さなければならない. 従って，ユビキタスコンピューティング環境のソフトウェアの開発環境は重要であり，本研究の成否を左右する大きな課題である.

現在計画している開発環境研究は，以下の 3 段階で進める.

1. ユビキタスコンピューティング標準開発環境
2. ユビキタスネットワーキングプロトコルを扱うミドルウェア
3. ユビキタスコンピューティング環境における情報処理モデルの確立

(6-1) ユビキタスコンピューティング標準開発環境

ユビキタスコンピューティング環境は膨大な数のノード数になる. まずハードウェアやオペレーティングシステムといった基盤ソフトウェア部分に対する標準化が必要である.膨大なノード数をまちまちの開発環境で構築しては,ソフトウェアの再利用性の観点から効率よくプロジェクトを運営できない. そこで，まず本プロジェクトの標準ハードウェア，標準基盤ソフトウェアの仕様を決め，その上でのユビキタスコンピューティング環境やユビキタスネットワーキングプロトコルのソフトウェアの再利用性，移植性の高い環境を構築する.

(6-2) ユビキタスネットワーキングプロトコルを扱うミドルウェア

ユビキタスネットワーキングのアプリケーション層のプログラミングを支援するためのミドルウェアを構築する.

(6-3) ユビキタスコンピューティング環境における情報処理モデルの確立

ユビキタスコンピューティング環境を実現する上で最も重要な分散協調動作を実現するソフトウェアの構築手法を研究する部分である．このテーマにおいても，次の２つのサブテーマを計画している．

1. 現実世界の記述方式
2. 超機能分散環境に適したプログラミングモデル（協調動作の記述）

ア．現実世界の記述方式

ユビキタスコンピューティング環境では,現実世界にコンテクストを取得し，協調動作の結果として，何らかの作用を現実世界にフィードバックする．こうした処理をコンピュータが扱うためには，現実世界をデジタル情報で表現し,それをノード間で交換できるための標準形式を構築する必要がある．しかもユビキタスコンピューティングが扱う事象は，単にオフィス空間といったものだけでなく，人間社会生活のあらゆる場面に及ぶため，まさに，現実世界あのあらゆる事象の標準デジタル表現形式を研究開発する．

イ．超機能分散システムのプログラミング技法

従来は,ネットワーク接続された複数のノード間における分散処理は,オブジェクトベースでモデル化したソフトウェア開発が主流になっている．しかしユビキタスコンピューティング環境のようにノード数が膨大である場合，扱うオブジェクト数も膨大になるため，その間の協調動作をノードオブジェクトレベルの peer-to-peer の協調関係をベースとした動作でプログラミングしては,抽象度が低すぎるということが問題となっている．従って，本研究では，ユビキタスコンピューティング環境全体に対して「計算場（Computing Field)」と呼ばれる仮想的なプログラミング抽象を提供し,各ノードとこの計算場の間の協調動作によってプログラミングする．

3.1.5　研究実施計画の詳細（システム工学的検証）

ユビキタスコンピューティング環境と人間社会･生活の間の親和性を重視し,本研究開発成果を実用に耐えるシステムとして完成させるために，システム工学的見地から，以下の検証を行う．

【サブテーマ７】
ユビキタスネットワーキングシステムのシステム工学的検証

(7-1) 信頼性の検証

本研究開発課題で構築されたシステム（以下，本システム）を，実際のユビキタスコンピューティング環境と同様の設置状況において運用し，そのシステム信頼性を検証する．ユビキタスコンピューティング環境は，生活のあらゆる面を支援するタイプのシステムであるため，誤作動などは致命的であり，この点に関する検証を行う．

(7-2) 運用評価

実際に本システムに想定される技術レベルの人員が，実験的に作られた本システムの環境を，一定期間オペレーションすることによって，①統合的視点によるセキュリティー強度の検証，②運用やメンテナンスの容易性を評価する．

(7-3) ユーザビリティ評価

本システムが利用者に提供するサービスのユーザインフェース手法について検証，評価する．特に，情報通信に関する技術に明るくない一般ユーザに対するユーザビリティ，また，身体障害者や子供，老人を含めたあらゆる人に対して使えるシステムになっているかという，ユニバーサルデザインの視点による評価を重要視する．

(7-4) スケールファクターのシミュレーション

本研究開発プロジェクトはあくまでも研究段階のものであるため，本研究成果が実際に世の中に大規模に普及した場合，どのような問題が起こっていくかを，本研究における実験のサンプルデータを使ったシミュレーションによって検証する．

(7-5) 環境アセスメント

本システムを生活環境に埋め込んだ場合の，放熱，騒音，電磁波などの影響を計測し，人体や他の機器，環境に対する影響を調査する．その結果をシステムの省資源部分にフィードバックしていく．

2.2 研究開発目標

2-2-1 最終目標

■全体を包括する最終目標（概要）

（1） 人間の振舞いや生活･社会を構成する事象に追従して応答するのに十分なリアルタイム性を持つ.

（2） 公開鍵暗号と PKI をベースとした暗号,認証のメカニズムを有し,社会のインフラを支えるユビキタス環境にふさわしい安全性と信頼性を実現できること.

（3） 情報家電やインターネットアプライアンスといった比較的乏しい計算機環境の上でも効率よく動作するように,実行性能がよくかつ規模が小さいシステムになっていること.

（4） システムを管理するための労力が小さいこと．具体的には，ユビキタスコンピューティング環境を構成する機器が設置されたら，たとえ停電等が起きても，無設定で復旧し，基本的に機器が故障するまで，メンテナンスする必要がない.

（5） 非専門家でも扱える簡便さを有すること．例えば，暗号･認証機構を知らない人でも,セキュアネットワーキングサービスを利用できること.

（6） 省リソース対応した回路技術を確立する．特に，省電力機能による電磁ノイズ発生の問題等を解決する.

（7） IP 網，デジタル方式の携帯電話網，PHS 網，固定加入電話網，ADSL 網といった，既存通信網との間のインターオペラビリティー機能を有すること.

（8） ユビキタスコンピューティング環境における典型的な協調･調停動作を複数実現することに成功し,その処理コードを分散透明な高い抽象度で記述できること.

■サブテーマ別の最終目標（詳細）

ア．基盤通信システムの研究開発

（1） ユビキタスネットワーキングプロトコルのセッション層部分までの基本プロトコルの仕様を開発し，その正当性，有効性を検証する.

（2） 上記のプロトコルを実現し，評価を行う.

（3） ユビキタスネットワーキングの物理層・データリンク層を担う，Bluetooth や ISO 14443，IEEE 802.11，ISO 7816，無線系電話プロトコルプロトコル等の中からネットワークシステ

ムの上で動作させるためのスタブ部分の仕様を開発すること.

（４） 既存のインターネット網である IP 網との間で相互運用と情報交換を可能にするゲートウェイ技術およびシステムを開発する.

（５） 基本機能として，認証機能，暗号機能を有すること.

（６） 認証・暗号機能の実現には，本研究のサブテーマ**「カ.」**で開発したセキュアハードウェアを十分に活用する.

（７） ソフトウェア規模は，十分小さい規模を想定する.

イ.ユビキタスコンピューティング環境を構成するシステム統合技術の研究開発

（１） IP 通信網や電話網などの既存の通信網との相互接続性を検証する.

ウ. 超機能分散システム指向の開発環境の研究開発（ハードウェア部分）

（１） ユビキタスコンピューティング環境の構築の用いる標準開発プラットフォームとしてのハードウェアを開発する.

（２） その上で，サブテーマ**「エ.」**で開発した標準 OS が動作する.

（３） サブテーマ**「カ.」**で開発したデュアル型のセキュアチップを搭載している.

（４） サブテーマ**「ア.（３）」**で挙げた各種ネットワークプロトコルを搭載する.

（５） 音声 CODEC を備える.

（６） グラフィックチップを備える.

エ. 超機能分散システム指向の開発環境の研究開発（ソフトウェア部分）

（１） 本研究サブテーマ**「ア.」**で開発するユビキタスネットワーキングプロトコルを標準機能で組み込み，それを本研究全体の標準プラットフォームとして利用する，標準リアルタイム OS を開発する.

(1-1) マルチタスク機能と，豊富なタスク間通信・同期機能を提供することができる.

(1-2) 省電力機能を有する.

(1-3) 本研究のサブテーマ**「カ.」**で開発したセキュアチップとの通信機能を有する.

（２） 本研究サブテーマ「ア.」で開発するユビキタスネットワーキングプロトコルに対して高抽象度のプログラミングインタフェースを提供するためのミドルウェアを開発する.

（３） 現実世界記述標準形式に関する研究開発

(3-1) ユビキタスコンピューティング環境が取り扱う実世界の各種環境情報情報の標準デジタル表現形式を策定する.

(3-2) ユビキタスコンピューティング環境が扱うあらゆるパラメータの表現を目指すため，その規模を例えると，「理科年表」のようなものになると考えている.

(3-3) 開発された標準記述形式は，ユビキタスネットワーキングプロトコルのプレゼンテーション層標準の一部として，全機器において使う.

(3-4) 表現の枠組みとしては，文字列形式である XML (eXtensible Markup Language)とバイナリ形式である TAD（TRON Application Databus）形式を構築する．特に後者は表現情報を効率よく表現できるための圧縮表現形式として，計算機資源が乏しいノードで利用する.

（４） 超機能分散プログラミングモデルに関する研究開発

(4-1) ユビキタスコンピューティング環境中に存在する莫大なノードの協調動作を高い抽象度でプログラミングできるプログラミングモデルおよび，そのプログラミング環境の開発を行う.

(4-2) ノード数は，数十から数万までを取り扱うことができ，ノードの分散性，動作の並列性をエンカプスレーションすることができる.

(4-3) あるユビキタスコンピューティング環境で動作していたソフトウェアをそのまま同じ機能をもった，他のユビキタスコンピューティング環境上でも稼動する移植性を有する.

オ. ユビキタスネットワーキングシステムのシステム工学的検証

（１） 本研究開発課題で構築したユビキタスコンピューティング環境の,一年程度の試験を行い,その期間の運用に耐えること.

（２） 現実社会における仕組みの中で運用しても，十分なセキュリ

ティー強度，運用の容易性が達成できること．

（3） 情報通信分野の素人である一般ユーザでも十分本システムを使いこなし，ユビキタスコンピューティング環境の機能の恩恵を受けられること．

（4） ユーザインタフェース部分には，ユニバーサルデザインが施されていること．

（5） 本研究開発課題で作成した実験レベルのシステムの運用データに基づき，それを都市レベルに拡大して普及させた場合の各種スケールファクターが確かめられたこと．

（6） 本システムを社会・生活の場に持ち込んでも，ユーザに不快感を与えたり，社会活動に悪影響を与えないこと．

カ. セキュアコンピューティングの基盤となるセキュアハードウェア

（1） コンタクトレス（無線）チャンネルのみを有するコンタクトレスチップと，コンタクトレス（無線）チャンネルとコンタクト（有線）チャンネルの双方を有するデュアルチップを開発する．

（2） コンタクトレス通信チャンネルの物理層・データリンク層のプロトコルは，ISO14443 Type-C 規格を満たす．

（3） コンタクト通信チャンネルの物理層・データリンク層のプロトコルは，ISO 7816 規格を満たす．

（4） 本課題で開発したユビキタスネットワーキングプロトコルで通信する機能を備える．

（5） PKI を使った公開鍵暗号技術に基いた暗号機能・認証機能を備える．

（6） 共通鍵暗号技術に基いた，実行効率のよい暗号機能・認証機能を備える．

（7） 耐タンパー性を有しており，悪意あるユーザからの不正操作から格納情報が守られる．

（8） ユビキタスコンピューティング環境を構成するノードに組み込むことで，そのノードの通信の安全性を向上できる．

キ. ユーザノードシステムの研究開発

（1） ユーザノードとは，ユビキタスコンピューティング環境の中で，利用者が直接接するユーザインタフェースをもった機器である．移動ノード，固定ノードとして，それぞれ複数種類のインテグレーションされたユーザノードを開発する．

（2） ユーザノードは，最終的にはサブテーマ「**ウ.**」で開発した標準ハードウェアを用い，サブテーマ「**エ.**」で開発した標準 OS，ミドルウェアなどを利用して開発する.
（3） サブテーマ「**ア.**」で開発したユビキタスネットワーキングプロトコルを実現する.

ク．サーバノードシステムの研究開発

（1） サーバノードとは，ユビキタスコンピューティング環境を裏で支える基盤サーバ群を含む.
（2）サーバノードは，以下の機能を提供する.
- CA 局や鍵配布サーバを含む PKI（公開鍵インフラストラクチャ）機能
- 電子マネーや電子チケットの決済機能
- 価値情報の発行機能
- デジタルコンテンツの発行機能

（3） サーバノードも悪意ある攻撃から守るためにハードウェアに一定の耐タンパー性を持たせる.

2-2-2 中間目標

■全体を包括する最終目標

（1） 公開鍵暗号と PKI をベースとした暗号,認証のメカニズムを有し,社会のインフラを支えるユビキタス環境にふさわしい安全性と信頼性を実現できること.
（2） 情報家電やインターネットアプライアンスといった比較的乏しい計算機環境の上でも効率よく動作するように,実行性能がよくかつ規模が小さいシステムになっていること.基盤プロトコル全体で，200～300KB 程度のソフトウェア規模を狙う.
（3） 非専門家でも扱える簡便さを有すること.
（4） システムを管理するための労力が小さいこと.

ア．基盤通信システムの研究開発

（1） ユビキタスネットワーキングプロトコルのセッション層部分までの基本プロトコルの仕様を開発する.
（2） 最終目標の記載欄で挙げ物理層・データリンク層プロトコルのうち，いくつかに関しては，そのスタブ部分の仕様開発を完了する.

（３） 基本機能として，認証機能，暗号機能を有すること．
（４） ソフトウェア規模は，十分小さいバイナリサイズを想定する．

イ．ユビキタスコンピューティング環境を構成するシステム統合技術の研究開発

（１） 相互接続性の検証が一部完了していること．

ウ．超機能分散システム指向の開発環境の研究開発（ハードウェア部分）

（１） ユビキタスコンピューティング環境の構築の用いる標準開発プラットフォームとしてのハードウェアを開発する．
（２） その上で，サブテーマ**「エ．」**で開発した標準 OS が動作する．
（３） サブテーマ**「オ．」**で開発したデュアル型のセキュアチップを搭載している．
（４） サブテーマ**「ア．（３）」**で挙げた各種 LAN，PAN のプロトコルを搭載可能である．
（５） 音声 CODEC を備える．
（６） グラフィックチップを備える．

エ．超機能分散システム指向の開発環境の研究開発（ソフトウェア部分）

（１） 本研究サブテーマ**「ア．」**で開発するユビキタスネットワーキングプロトコルを標準機能で組み込み，それを本研究全体の標準プラットフォームとして利用する，標準リアルタイム OS を開発する．
(1-1) マルチタスク機能と，豊富なタスク間通信・同期機能を提供することができる．
(1-2) 省電力機能を有する．
(1-3) 本研究のサブテーマ**「カ．」**で開発したセキュアチップとの通信機能を有する．

（２） 本研究サブテーマ**「ア．」**で開発するユビキタスネットワーキングプロトコルに対して高抽象度のプログラミングインタフェースを提供するためのミドルウェアを開発する．

（３） 現実世界記述標準形式に関する研究開発

(3-1) 内容的には，最終目標で記載したとおり．中間目標の時点では，標準形式の策定が完了している．それを実際のユビキタスコンピューティング環境に組み込んで実現するのは，これ以後の年度に行うものとする．

（4） 超機能分散プログラミングモデルに関する研究開発

(4-1) ユビキタスコンピューティング環境中に存在する莫大なノードの協調動作を高い抽象度でプログラミングできるプログラミングモデルおよび，そのプログラミング環境の開発を行う．中間目標の時点で，その基本モデルは確立する．その実現や検証はそれ以後の年度に行うものとする．

オ. セキュアコンピューティングの基盤となるセキュアハードウェア

（1） 最終目標の挙げた中で，コンタクトレス（無線）チャンネルのみを有するコンタクトレスチップの開発が完了している．

（2） コンタクトレス通信チャンネルの物理層・データリンク層のプロトコルは，ISO14443 Type-C 方式である．

（3） この時点で開発された版のユビキタスネットワーキングプロトコルで通信する能力を有する．

（4） 共通鍵暗号技術に基いた，実行効率のよい暗号機能・認証機能を有すること．

（5） 耐タンパー性を有しており，悪意あるユーザからの不正操作から格納情報が守られること．

カ．ユーザノードシステムの研究開発

（1） ユーザノードとは，ユビキタスコンピューティング環境の中で，利用者が直接接するユーザインタフェースをもった機器である．移動ノード，固定ノードとして，それぞれ１種類以上のインテグレーションされたユーザノードを開発する．

（2） ユーザノードは，最終的にはサブテーマ**「ウ.」**で開発した標準ハードウェアを用い，サブテーマ**「エ.」**で開発した標準 OS，ミドルウェアなどを利用して開発する．

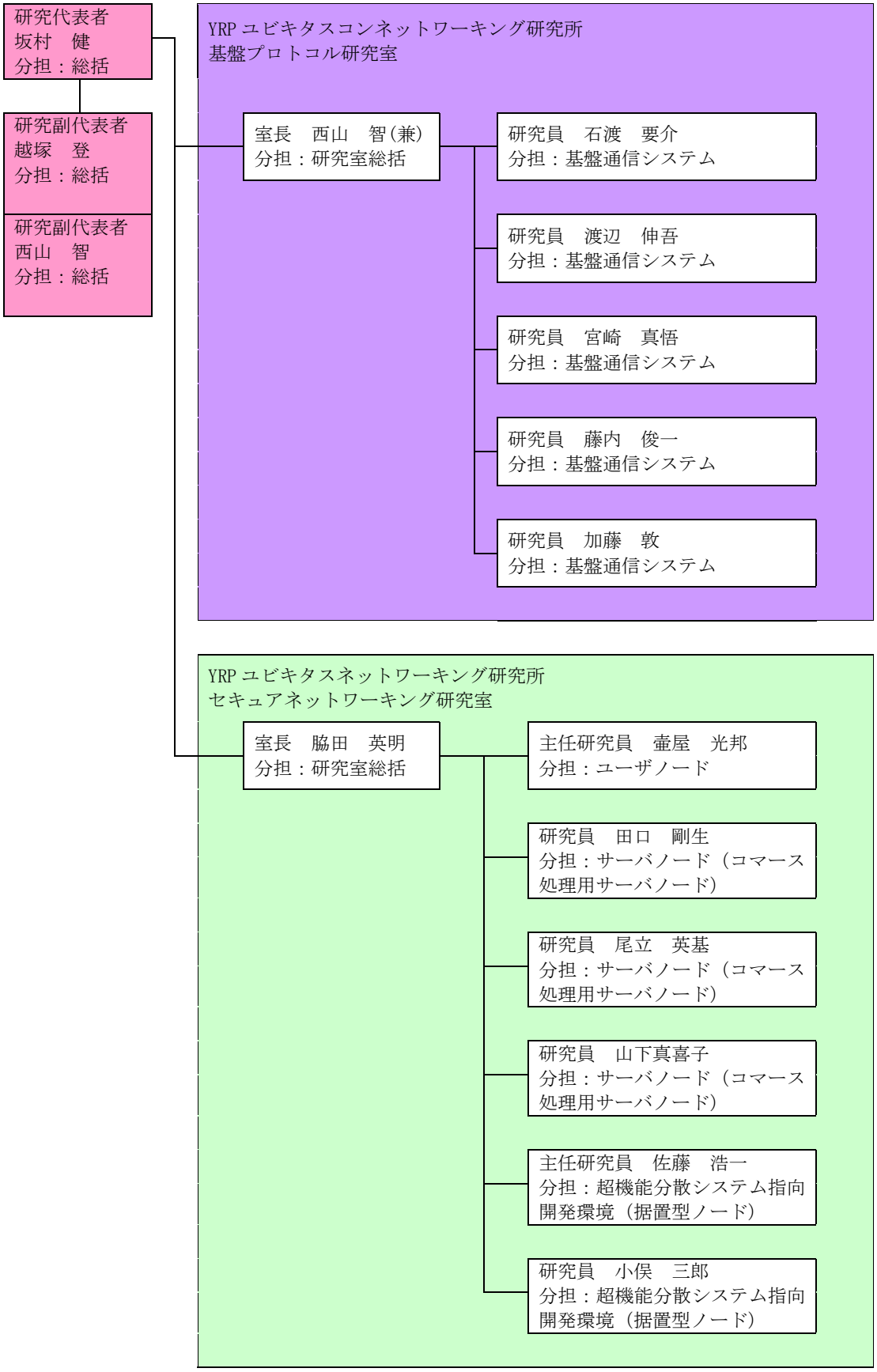
（3） サブテーマ**「ア.」**で開発したユビキタスネットワーキングプロトコルを備える．

キ．サーバノードシステムの研究開発

（1） サーバノードとは，ユビキタスコンピューティング環境を裏で支える基盤サーバ群を含む.

（2）中間目標の時点では，CA 局や鍵配布サーバを含む PKI（公開鍵インフラストラクチャ）機能の開発が完了している.

第三章
研究開発体制


研究代表者
坂村　健
分担：総括
研究副代表者
越塚　登
分担：総括
研究副代表者
西山　智
分担：総括
YRP ユビキタスコンネットワーキング研究所
基盤プロトコル研究室
室長　西山　智(兼)
分担：研究室総括
研究員　石渡　要介
分担：基盤通信システム
研究員　渡辺　伸吾
分担：基盤通信システム
研究員　宮崎　真悟
分担：基盤通信システム
研究員　藤内　俊一
分担：基盤通信システム
研究員　加藤　敦
分担：基盤通信システム
YRP ユビキタスネットワーキング研究所
セキュアネットワーキング研究室
室長　脇田　英明
分担：研究室総括
主任研究員　壷屋　光邦
分担：ユーザノード
研究員　田口　剛生
分担：サーバノード（コマース処理用サーバノード）
研究員　尾立　英基
分担：サーバノード（コマース処理用サーバノード）
研究員　山下真喜子
分担：サーバノード（コマース処理用サーバノード）
主任研究員　佐藤　浩一
分担：超機能分散システム指向開発環境（据置型ノード）
研究員　小俣　三郎
分担：超機能分散システム指向開発環境（据置型ノード）

第四章
研究開発の概要
（平成 14年度）

4.1 研究開発実施計画

4.1.1 研究開発の基本方針

平成 14 年度の研究開発の中心は、ユビキタスネットワーキング環境の基盤となる新しい基本方式の策定、およびその有効性を評価・検証するための、ハードウェア、ソフトウェアの実装である。その設計には、昨年度取り組んだ、ユビキタス型応用評価による知見が有効に活かされる。また、システムを試作する際には、やはり昨年度取り組んだ、研究開発基盤システムであるハードウェア、ソフトウェアを利用して構築する。

4.1.2 主要な研究題目と目標

本研究は、６つのサブテーマに分けて研究計画を立案している（全体計画の項目参照）。ここでは、その６つのサブテーマに沿って、今年度取り組むことを予定しているの主要な研究開発テーマについて述べる。

【サブテーマ１】
セキュアコンピューティングの基盤となるハードウェア

ユビキタスコンピューティング環境による夢を実現するためには、セキュリティーの確保が不可欠である。更に近年は、サイバーアタックやクラッキング、サイバーテロを想定した対策も求められている。ユビキタスコンピューティング環境を、ネットワーク経由のアタックから守るためには、セキュアな通信プロトコル、セキュアな通信システムの開発が不可欠である。そこで、本研究では、LSI 自体に不正アクセスが加えられないように加工を施した、いわゆる「耐タンパー性（Tamper Resistance）」を有するハードウェアを、ユビキタスコンピューティング向けチップとして新規に開発する。

その中でも本年度は、主にコンタクトレス（ノン・コンタクト）型の通信インタフェースを有するセキュアチップの研究を行う。現在の計画では、マルチメディア応用にも対応可能な（１）高性能なハイエンド型のセキュアチップと、身の回りのあらゆるものに埋込むことを想定した、安価で高信頼性の（２）超小型セキュアチップの研究を行う。

【サブテーマ２】
基盤通信システム

本研究ではユビキタス環境を構築するための、次世代通信プロトコルを研究開発する。特に本年度は、通信ノード・計算ノードの「偏在（ユビキタス）性」という特質部分を有効に扱うコア技術の研究開発を行う。まず第一にモバイル型の通信ノードが、様々な条件に応じて動的にネットワークを再構成する、（１）アドホックネットワーキングの研究である。第二に、ユビキタス環境を構成する据置型の通信ノードを対象とし、エンドユーザが自由に拡張することが可能な、（２）随意拡張型固定網通信プロトコルの研究である。また、通信プロトコルスタックの上位におけるプレゼンテーション層の研究として、ユビキタスネットワークプロトコルが扱う実世界情報のデジタル表現手法の研究として、（３）実世界データ研究・Everything ID 研究に取り組む。ユビキタス環境とは、実世界のあらゆる事象をコンピュータ上で処理し、デジタル通信で情報交換することと位置付けることができ、その際、実世界のあらゆる情報が、コンピュータの上で標準的な形で表現されることが不可欠であり、本年度は、その基幹部分のデータインフラの構築を行う。

【サブテーマ３】
ユーザノードシステム

ユビキタスコンピューティング環境を構成するノードの中で、ユーザと直接接することが想定される機器類の研究開発である。本サブテーマでは、ユビキタスコンピューティング環境での利用条件に適合したユーザノードの構成方法・機構を研究開発する。特に、本年度は、（１）カームネットワーキング／カームコンピューティング、（２）強化現実型ユーザインタフェース、（３）位置検出機構の３テーマを中心的に扱うことを計画している。

（１）カームネットワーキング／カームコンピューティング

ユビキタスコンピューティング環境では、あらゆる場所に膨大な数のネットワークノード・コンピューティングノードが設置される。これらは、電力などの資源を浪費することなく、最小限の資源利用で効率的に動作することが不可欠である。そこで、こおうした無数にちりばめられるコンピュータを想定し、ハードウェアおよび OS や BIOS レベルの基盤ソフトウェアにおける、新しい電源制御機構を含む省電

力機構の実現技術を研究開発する。

（２）強化現実型ユーザインタフェース

ユビキタス環境のユーザノードに対して、高い対ユーザ親和性（User Friendliness）を実現するためには、コンピュータの存在を意識することなく、自然な対話が行えることが重要である。そうしたユビキタス環境におけるユーザとのインタフェースをつかさどる技術として、本年度は、強化現実型ユーザインタフェース（Augmented Reality）の研究開発を進める。

（３）位置検出機構

上記の強化現実型ユーザインタフェースの特徴は、コンピュータやネットワークによって構成される電子的な仮想空間と現実空間を融合することによって、現実世界において、より高いユーザ親和性と IT サービスを提供する点にある。これを実現するために、最も重要な要素技術の一つで、現在まだ実用レベルに達していないものに、位置検出メカニズムがある。本年度は、特に屋内、つまり GPS などのグローバルな位置検出インフラが利用できない実環境、における高い精度のユーザ位置検出技術の研究開発を行う。

【サブテーマ４】

サーバノードシステム

サーバノードシステムは、ユビキタス環境をバックエンドで支える、大量の計算資源と通信資源を有するインフラシステムである。サーバノードシステムにはさまざまな役割があるが、今年度は特に、セキュリティー基盤としてのサーバーノードシステムの研究に注力する。最初のサブテーマにある、セキュアハードウェアの研究開発と連携し、そこで研究開発されるセキュアチップのセキュア基盤のバックエンドとして動作するものを研究開発する。

本年度のセキュア基盤の研究の主要な課題は、セキュリティーを実現する機構をコンパクトに実装すること（（１）ユビキタス PKI）と、公開鍵暗号系の認証処理を行うときの PKI の応答性を向上させる（２）リアルタイム PKI 技術の確立である。

【サブテーマ５】

システム統合技術

ユビキタス環境において、セキュリティーをエフォートレスに実現

する手法が求められている。そこで、本年度の研究では、ユーザの身体的・生理的特性を利用したバイオメトリクス型の認証機構を、ユビキタス環境に適用するために大規模化する研究に取り組む。そのためには、各ユーザ端末にバイオメトリクス検知用のセンサーを格納し、認証情報はサーバシステム上でセキュアに集中管理し、認証情報を通信回線で交換する、分散型バイオメトリクス認証処理である。この機構の実現には、セキュアハードウェアの研究、ユーザ端末の研究、サーバシステムの研究と連携をとった、統合的な技術として研究開発を進める。

【サブテーマ６】
超機能分散システム指向開発環境の整備

昨年度から継続し、ユビキタスコンピューティング環境の基盤システムとなる、ハードウェア（MPU、基本ボード、等）、ソフトウェア（オペレーティングシステム、各種ミドルウェア、等）の研究開発を継続する。特に本年度注力するのは、以下のテーマである。

（１）ワンボード型エンベディドアーキテクチャ

現在の PC に匹敵する計算資源や通信資源を有するハイエンドクラスのユビキタス環境向けアーキテクチャの研究である。実際に、本アーキテクチャに基づいたシステムを試作して、評価する。

（２）ワンチップ型エンベディドアーキテクチャ

最も小型で、安価なユビキタス環境向けの計算機ノードアーキテクチャを研究する。この場合、通信インタフェースをも含んだ、全システムが、ワンチップ上に実装されることを想定している。

（３）ミドルウェア研究

上記アーキテクチャシステムをターゲットとした、ミドルウェアシステムの研究開発を行う。主に、分散ネットワーキングシステムのミドルウェアや、暗号・認証などのセキュリティーシステムのミドルウェアの研究開発を行い、高い品質のユビキタス環境ソフトウェアを高い効率で開発するための手法を研究する。

4.2 研究開発の実施内容

本年度の委託業務の内容は，６つのサブテーマに分けて実施した．

【サブテーマ１】セキュアコンピューティングの基盤となるハードウェア

【サブテーマ２】基盤通信システム

【サブテーマ３】ユーザノードシステム

【サブテーマ４】サーバノードシステム

【サブテーマ５】システム統合技術

【サブテーマ６】超機能分散システム指向開発環境の整備

以下，それぞれのサブテーマについて，委託業務の実施内容について述べる．

【サブテーマ１】
セキュアコンピューティングの基盤となるハードウェア

（１） 高性能なハイエンド型のセキュアチップ

- ハイエンド型セキュアチップの最初のバージョンとして、8 ビット CPU を用いたセキュアデータキャリアチップの研究開発を行った。
- チップには、セキュリティーが強化されたセキュアプロトコル(eTP)を実装した。本実装では、暗号関数を５種類サポートし、従来技術と比較して強固なセキュリティレベルを実現した。
- 次に、更なるセキュリティー強化を狙って 16 ビット CPU を用いたセキュアデータキャリアチップの研究開発にも着手し、基本アーキテクチュアの設計を終えた。

（２）超小型セキュアチップ

チップサイズが 1㎜ 以下の超小型セキュアチップの技術調査および基本仕様の検討を実施した。

【サブテーマ２】基盤通信システム

まず第一にモバイル型の通信ノードが、様々な条件に応じて動的にネットワークを再構成する、（１）アドホックネットワーキングの研究として、屋内用センサネット用のネットワークプロトコルのモデルと詳細仕様を策定した。第二に、ユビキタス環境を構成する据置型の

通信ノードを対象とし、エンドユーザが自由に拡張することが可能な、（２）随意拡張型固定網通信プロトコルの研究として、データリンク層のプロトコルの詳細設計を行った。また、通信プロトコルスタックの上位におけるプレゼンテーション層の研究として、ユビキタスネットワークプロトコルが扱う実世界情報のデジタル表現手法の研究として、（３）実世界データ研究・Everything ID 研究に必要なデータ要素の洗出しを行った。またその交換に必要な基幹部分のインフラを開発した。

【サブテーマ３】ユーザノードシステム

（１）カームネットワーキング／カームコンピューティング

ハードウェアおよびOSやBIOSレベルの基盤ソフトウェアにおける、新しい電源制御機構を含む省電力機構のためのプロトコルと制御方式を開発した。

（２）強化現実型ユーザインタフェース

下記に述べる超音波センサと RFID を組み合わせた位置検出機構の研究及び、【サブテーマ２】で実施した屋内用センサネットプロトコルで得られる位置情報やユビキタス環境情報を実際の屋内モデルにマッピングしてユーザに提示する研究を実施した。

（３）位置検出機構

現実世界において、より高いユーザ親和性と IT サービスを提供するために、特に屋内、つまり GPS などのグローバルな位置検出インフラが利用できない実環境、における高い精度のユーザ位置検出技術として、超音波センサと RFID を組み合わせた位置検出と、画像認識による位置検出方式を開発した。また取得した位置を管理するためのサーバの開発を行った。

【サブテーマ４】サーバノードシステム

（１）ユビキタス PKI

従来の PKI 証明書(X.509)と比較し、リソースの少ないユビキタス機器で利用できる証明書の研究開発を実施した。具体的には、ユビキタス環境での情報セキュリティを保証する証明書の必要項目を抽出し、暗号方式の異なる複数種の暗号鍵を円滑に取り扱う、独自の形式を検討した。

（２）リアルタイム PKI

【サブテーマ１】で実装した eTP との組み合わせで、公開鍵暗号系の処理を IC カードでリアルタイムに実現できる認証方式を研究開発した。

上記のプロトコルをサーバノードシステムに実装し、ユビキタス PKI の基本アーキテクチャを構築した。

【サブテーマ５】システム統合技術

分散型バイオメトリクス認証処理に必要な要素技術の研究開発を実施した。

- 基本処理方式の検討を行った。
- 必要なハードウェアを試作した。
- デバイスドライバ等、基本的なソフトウェアの開発をおこなった。

【サブテーマ６】超機能分散システム指向開発環境の整備

（１）ワンボード型エンベディッドアーキテクチャ

試作・評価が完了した。ユビキタスコンピューティングに関連する各種デモンストレーションの開発に応用することで、その有用性を確認した。具体的には、RF-ID タグと組み合わせてグラフィックユーザインタフェースにモノの情報を有機的に表示するシステムや、電子的な価値情報を転々させるシステムを試作完成させ、評価した。

（２）ワンチップ型エンベディッドアーキテクチャ

５００円玉サイズのシステムとして、仕様策定および開発が完了した。セキュアチップを接続するインタフェースおよび各種通信機能を搭載したシステムとなっている。

（３）ミドルウェア研究

次のミドルウェアを開発した。

- 主なネットワークプロトコルを実現するミドルウェア
- 非接触 IC カードとセキュアな通信を行うための暗号通信ミドルウェア
- サーバと協調して認証を行うミドルウェア

4.3 研究開発の効果

【サブテーマ１】
セキュアコンピューティングの基盤となるハードウェア

（１） 高性能なハイエンド型のセキュアチップ

- 8 ビット CPU を用いたセキュアデータキャリアチップをいくつかのユビキタス応用（入退室システム等）に適用し、その有効性（安全性）を確認した。
- 16 ビット CPU を用いたセキュアデータキャリアチップの基本アーキテクチャは、来年度以降のチップの開発に利用し、更なるセキュリティ強化の検証を行う。

（２） 超小型セキュアチップ

チップサイズが 1mm 以下の超小型セキュアチップの技術調査においては、大きさは 0.5mm 以下のチップがいくつか存在するものの、我々の考えるユビキタスネットワーキング環境で要求されるセキュリティ強度に対しては、十分とは言えないものであった。設計した基本アーキテクチャを元に来年度以降も研究開発を継続する。

【サブテーマ２】基盤通信システム

（１） アドホックネットワーキングの研究

屋内用に特化したセンサネットのモデルを提案すると共に、そのモデルを効率的に実現するためのネットワークプロトコルの仕様を詳細化できた。この仕様に基づき実証システムを構築し、そのモデルの妥当性が評価により確認できた。これにより、IP 通信を提供できないような低処理能力のセンサを利用したセンサ情報の配信が可能となった。

（２） 随意拡張型固定網通信プロトコル

エンドユーザが自由に通信ノードを拡張できる随意拡張型固定通信網プロトコルのデータリンク層の詳細仕様化が完了し、平成 15 年度以降この仕様に基づく通信ハードウェアの作成を行う。これにより、専門的知識の無いエンドユーザがホームネットワーク等のユビキタスネットワークを自由に構築できるようになると考えられる。

（３） 実世界データ研究・Everything ID 研究

必要なデータの洗出しを行い、平成 15 年度以降に ID 割り当て方式を決定するための基礎ができた。

【サブテーマ３】ユーザノードシステム

（１） カームネットワーキング/カームコンピューティング

ハードウェアおよびOSやBIOS レベルの基盤ソフトウェアにおける、新しい電源制御機構を含む省電力機構のためのプロトコルと制御方式を開発し、これにより平成 15 年度以降に多数のユビキタス機器を制御するカームコンピューティングの応用を効率的に実現することが可能となった。

（２） 強化現実型ユーザインタフェース

屋内用センサネットプロトコルで得られる位置情報やユビキタス環境情報を実際の屋内モデルにマッピングしてユーザに提示する研究の成果は、平成 15 年度以降に強化現実型（AR）あるいは MR(Mixed Reality)型の応用に発展させることができる基盤として活用する予定である。

（３） 位置検出機構

GPS 等のグローバルな位置検出機構が利用できない屋内で、RFID と超音波センサを組み合わせて高精度かつ安価に屋内の多数のオブジェクトの位置を検出する方法や、画像認識により高精度に人間の位置を検出する方式を考案し、今後の位置依存型ユビキタス応用に適用することが可能となった。

【サブテーマ４】サーバノードシステム

（１）ユビキタス PKI

証明書全体のコンパクト化、証明書正当性を検査する処理時間の短縮を効果として得た。また、ユビキタス機器への組み込みを想定した試作チップへの実装により、実用に耐え得るデータサイズであることを物理的に検証できた。

（２）リアルタイム PKI

シミュレータや試作チップによる実装評価では、開発した即時応答型認証の有効性を確認できた。接触／非接触と複数の通信方式で動作

検証を試みる中では、非接触通信における電力供給等、いくつかの技術的課題も見つかったため、次年度以降は、こうした技術項目の研磨を継続し、チップを始めとしたサーバノードシステム全般の仕様や実装に反映させる。

【サブテーマ５】システム統合技術

基本処理方式を検討の結果、ISO 15408 規格をベースとした設計指針が知見として得られた。また、本技術を実現するベースとなる、指紋センサーのついた認証端末が試作されており、それを研究所内に埋め込み分散型バイオメトリクスの実験環境を整えることができた。また、それを動作させるために必要なソフトウエアの開発も行い、基本動作を確認することができた。次年度以降、他のサブテーマとの研究とも協調しながら、分散バイオメトリクス認証システムを確立する。

【サブテーマ６】超機能分散システム指向開発環境の整備

(1) ワンボード型エンベディッドアーキテクチャ

試作・評価が完了した。各種デモ・実験に応用して試した結果、ユビキタス環境における有用性を確認することができた。また、これにより、人とのコミュニケーション、電子的価値情報の転送といった当研究所の先進的な研究成果を、実機レベルで動作させて検証することが可能となった。

(2) ワンチップ型エンベディッドアーキテクチャ

設計・試作が完了した。次年度以降、この試作機を用いてセンサネットワークなどの研究を進めることが可能となった。

(3) ミドルウエア研究

暗号･認証などのセキュリティシステムを実現するためのミドルウエアを開発完了した。次年度以降、これを用いて、セキュリティを保った上での分散コンピューティングの研究を行っていく。また開発したミドルウエアをリアルタイムカーネル上に組み込む方式を検討終了した。この方式にもとづいてミドルウエア開発を行うことで、開発したミドルウエアを小さい移植コストで様々な機器開発に流用することが可能となった。

第五章
研究実施状況

5.1 セキュアハードウェアの研究開発

本章は、本研究開発テーマの中で、サブテーマ１、セキュアコンピューティングの基盤となるハードウェア、部分の研究成果を報告する。

まず第一の取り組みは、高性能なハイエンド型のセキュアチップである。ここで、ハイエンドのセキュアチップとは、暗号認証のためのハードウェア回路を有するデータキャリアチップのことである。我々は、ハイエンド型セキュアチップの最初のバージョンとして、８ビット CPU を用いたセキュアデータキャリアチップの研究開発を行ってきた。そして、その８ビット型セキュアチップをいくつかのユビキタス応用（入退室システム等）に適用し、その有効性（安全性）を確認した。更に、チップに対しては、セキュリティーが強化されたセキュアプロトコル(eTP)を実装した。また暗号認証に用いる、暗号関数を５種類サポートし、従来技術と比較して強固なセキュリティレベルを実現した。更に、セキュリティー強化を狙って 16 ビット CPU を用いたセキュアデータキャリアチップの研究開発にも着手し、基本アーキテクチュアの設計を行った。このセキュアデータキャリアチップの基本アーキテクチャは、来年度以降のチップの開発に利用し、更なるセキュリティ強化の検証を行う予定である。

また、第二の取り組みとして、チップサイズが 1mm 以下の超小型セキュアチップに関する技術調査および基本仕様の検討を実施した。その結果、大きさは 0.5mm 以下のチップがいくつか存在するものの、ユビキタスネットワーキング環境で要求されるセキュリティ強度に対しては、十分とは言えないものであり、適切な利用方法をとる必要があることが判明した。

以下、こうした取り組み成果について、順を追って説明する。

5.1.1 高性能なハイエンド型のセキュアチップ

（ア）セキュアデータキャリアチップのユビキタス応用

本節では、昨年度より本研究所で進めているセキュリティー機能を有するデータキャリアチップ（SDCC: Secure Data Carrier Chip）を使った、実際のセキュリティー応用実験システムについて述べる。本年は、（１）入場ゲートシステム、（２）入室認証システム、（３）部屋制御用認証システムを構築し、実際に本研究所で実用とした。以下、これらのシステムについて順に説明する。

（１）入場ゲートシステム

入場ゲートのを通過するためのトークン情報を保持した SDCC を所定のカード R/W にかざすと、U-Card によって実装されているゲート制御システムが SDCC の情報を取得し、その情報に応じてゲートの開閉を制御する（**図 5.1-1, 5.1-2 左**）。

正常な入場トークンを有する SDCC は、ゲートの開放と同時に制御システムがそのトークンの取り出し、SDCC からトークンはなくなる。トークンが消滅した SDCC をかざしても、エラーとなる（5.1-2 右）。

また、このトークンは、価値情報の転々流通機能によって、小型端末側に移動することもできる。この小型端末は、SDCC からトークンを読み出すＲ／Ｗの役割を果たすと同時に、入場ゲートに対しては、SDCC のエミュレーションを行う。つまり、ISO 14443 Type-C の対称通信を利用している。

図 5.1-1: ゲートシステム前景

図 5.1-2：ゲートが開放した例と（左）、トークンがなくて、エラーとなった例（右）

（２）入室認証システム

入場認証情報や入室認証情報を持った SDCC と連携させ、入場ゲートの入場開閉、ドアの開錠施錠を行い、更に入室入場の様子のカメラ画像を転送する、監視システムを開発した。ゲートやドアの制御に U-Card、カメラ画像の撮影と圧縮および転送にμU-Card を応用してシステムを構築した（図 5.1-3)。

図 5.1-3: セキュリティードアカード R/W 部とカメラ部

（３）部屋制御用認証システム

当研究所には、セミナーや会議を行うためのミーティングルームが

あり、外部からの来客と内部の研究員とが混在して、討議等を行っている。ここでは、コンピュータの画面をプレゼンテーションの時にスクリーンに映写したり、ビデオを上映したりする。比較的、頻繁にスクリーンに映写するコンピュータを切り替えたり、照明を制御する要求が出てくる。

そこで、我々は、会議卓の手元に制御用端末（U-Card で実装）を設置し、各会議参加者がそれぞれ管理できるシステムを構築した、ただし、ゲストも社内の研究員も、まったく同様に装置を制御できてしまっては、セキュリティー上の問題がある。そこで、社内研究員は、入場ゲートや入室認証システムで使っている SDCC で作られている社員省を個人認証カードとして利用し、ゲストは暫定的なゲスト専用カードを渡し、そのカードを用いて利用者認証することにした。

社員証カードをおいたときは、システム全体を操作することが可能である。ゲストがカードをおいた場合には、照明などの制御はできないが、前面のプロジェクタ画面を、テーブルの各自の手元のディスプレイ信号の入力へと切り替えることができる。

テーブルの上には、SDCC をおくポイントに[e]マーク

テーブルの手元を引き出すと、制御用のコンピュータと、各種入出力用のコネクタとケーブルが収容

社員カードをおき認証されると、システム制御パネルが現出する

これで部屋のプレゼン機器、照明機器を制御できる。
図 5.1-4: 部屋制御用認証システム

（イ）16 ビット CPU を用いたセキュアデータキャリアチップ（SDCC）の基本アーキテクチャ

（１）全体アーキテクチャ

SDCC は、eTRON アーキテクチャを採用しており、システム全体として、図 5.1-5 のアーキテクチャとなる。

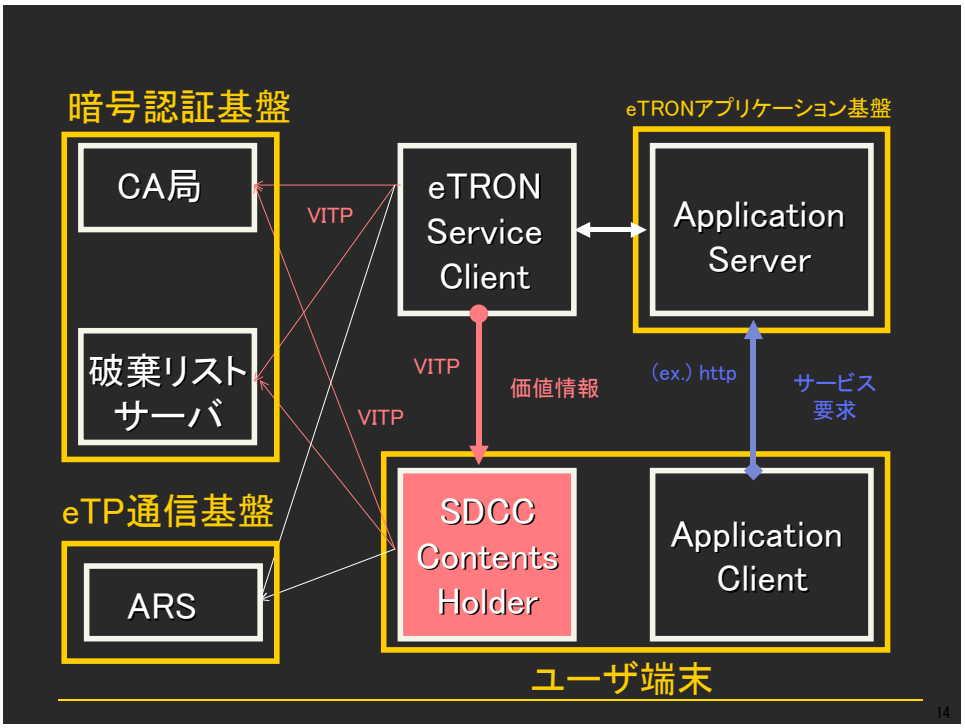


図 5.1-5: eTRON ベースの SDCC 全体システムアーキテクチャ

4. SDCC Contents Holder
   本アーキテクチャにおいて、価値情報を格納しているノードを Contents Holder と呼ぶ。SDCC はこの Contents Holder に相当する。カード型のものでなくとも、サーバ型でも、デスクトップ型のノードでも、eTRON インタフェースを有して、それによって、格納情報のインタフェースがとれる機器はすべて Contents Holder である。
5. eTRON Service Client
   一方、Contents Holder に格納されている情報を操作するノードが、Service Client である。たとえば、SDCC で構築された IC カードに電子チケットを送りこむサーバーシステムは、"Service Client"となる。
6. Applicatation Server, Application Client
   価値情報を交換する契機や、指示となる操作を行うアプリケーションである。たとえば、電子チケットシステムで言えば、どの電子チケットを買うかどうかと注文し、決済する部分のシステムがこの Applicatiaon Server と Application Client となる。そして、購入を決めた電子チケットは、Service Client から Contents Holder へ、SDCC がサポートする VITP: Valuable Information Transfe Protocol によって、安全にダウンロードされる。

7. **CA 局**
   Service Client や Contents Holder が公開鍵暗号や認証を行う時に必要となる電子証明書や公開鍵、秘密鍵のペアなどを発行する。
8. **破棄リストサーバー**
   無効になった証明書のリストを蓄積し、信頼性の高い暗号認証処理を行う前に、受け取った証明書が有効かどうかチェックするための問い合わせに利用される。
9. **ARS: Address Resolution Server**
   価値情報を転送する VITP はセッション層のプロトコルである。セッション層における通信の相手指定は、現在 eTRON ID で行うが、これを、経路制御可能なネットワーク層のアドレスとの対応を保持しているサーバが ARS である。

（２）VITP: Valuable Information Transfer Protocol

Contents Holder と Service Client 間で行われる価値情報を扱う通信プロトコルが VITP: Valuable Information Transfer Protocol である。VITP の論理的なパケット形式は、次のとおりである。

10. Address Headder
    - Destination Address
    - Source Address
11. Main Payload
    - Command identifier
    - Command parameters
      etc..

    ※ 暗号セッション中は暗号化される
12. MAC Tailer
    - MAC Class Identifier
    - MAC Body (CRC, Hash, Digital Signature, etc..)

    ※ パケットの途中の改竄を検出する

VITP はセッション層のプロトコルで、上記の論理パケット形式の部分で説明したとおり、相手を eTRON ID というセッション層ＩＤを使ってしている。そこで構築するセッションは、次のメッセージ交換によるプロトコルで構成される（**図 5.1-6**）。

13. セッション開始時に認証処理

- 通信相手が誰かを特定
- eTRON ID が特定される。

14. セッション通信中は、基本的に
    - 暗号通信による通信内容秘匿
    - 電子署名などの MAC による改竄防止
15. セッション終了時に、Commit/Abort
    - Abort されたときには、Roll-Back 処理

    ※　カードなどの場合はハードウェア機能に依存

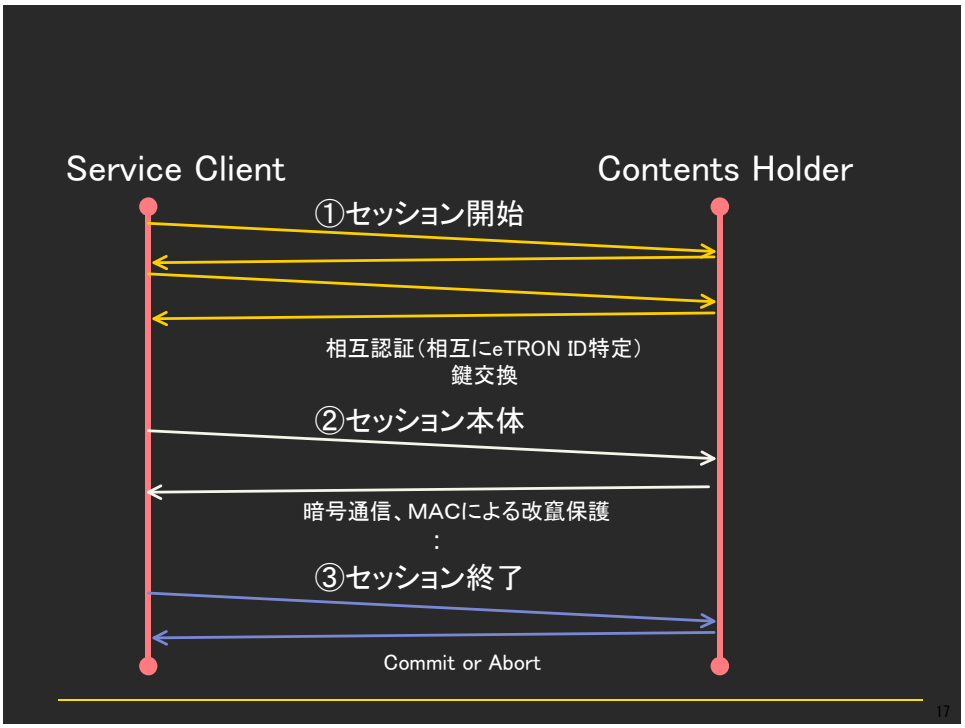


図 5.1-6: VITP のパケット交換

（３）16 bit SDCC コマンド体系

コマンド体系は、以下のとおり基本命令群と応用命令群から構成される。

16. 基本命令群（セッション層以下のプロトコルを実現する命令群）
    - (ア)　セッション／トランザクション命令群
    - (イ)　認証機能
    - (ウ)　暗号アルゴリズムネゴシエーション機能
    - (エ)　Commit/Abort 機能
    - (オ)　PKI 管理命令
    - (カ)　通信制御用命令

17. 応用命令群（アプリケーション命令を実現する）

（ア） ファイル I/O 命令群

① create 命令

② delete 命令

③ read 命令

④ update 命令

など

（イ） 価値情報の転々流通処理命令

（ウ） 暗号／復号＋加算／減算のアトミック命令

（エ） VPN 用鍵交換命令

5.1.2 超小型セキュアチップ

（ア）概要

現在超小型チップとして、1mm×1mm 以下のサイズでは、CPU を格納することができない。CPU や暗号認証のための暗号回路のためには、どうしても 1mm×1mm 以上のチップ面積が必要であることがわかった。従って、1mm×1mm 以下のチップでは、ハードウェアによる耐タンパー性や、情報を読み書きする物理プロトコルレベルのプロテクション以外のソフトウェアレベルのプロテクションを行うことは困難である。

（イ）μチップ（日立製作所）

たとえば、既存の超小型チップとしては、日立製作所のμチップが知られている（図 5.1-1）。これは、2.4GHz 帯の搬送波を用いる RFID で、中には 128 ビットの情報のみを格納することができる。この 128 ビットの情報は、μID と呼ぶもので、各チップをユニークに識別する識別子となるものである。工場出荷時に番号の重複が起きないように、ID のユニーク性を保証するように書き込まれ、以後ユーザがそれを変更することができないようになっている。

こうした超小型チップのセキュリティー応用について述べる。たとえば、μチップの重要の一つには、それが貼付されたものの偽造を発見するというものが含まれている。しかし、チップ面積も小さく、搭載できるロジック量も小さいことから、高度なセキュリティープロトコルを実装することのできないものである。

従って、格納されたデータを正当なユーザからのみ読めるというプライバシー保護といったセキュリティーに関しては、まったく無力である。一方、これだけの微小サイズで実装された RFID は他に例を見

ないこと、チップ毎に格納されているＩＤは異なっており、しかもそれをユーザが変更できないという性質から、目視によるチップの物理的形状確認と、電気的動作の確認の両方を行うことにより、チップの真贋判定を行うことが容易にできる。従って、μチップのセキュリティー応用としては、それを貼付したモノの偽造防止・偽造発見が有望であると考えられている。

図 5.1-7：μチップ［日立製作所］

5.2 基盤通信システムの研究開発

5.2.1 アドホックネットワーキングの研究

(ア)方式概要

（1） はじめに

ユビキタスサービス提供のためのユビキタスネットワークの一環として、多数のセンサがネットワークに接続されたセンサネットワークが研究されている。従来、センサネットワークは屋外などのアドホックな環境での利用が想定されてきた。ここでは屋内での利用を想定したセンサネットワークについて提案する。

（2） センサネットワークに関する従来研究

一般にはセンサ情報を運ぶネットワーク全体をセンサネットワー

クと呼ぶ場合もあるが、ここではセンサ情報専用のネットワークをセンサネットワークと呼ぶ。センサネットワークに接続するセンサには、IP 等の既存ネットワークプロトコルや Java 等の高水準な言語が利用できる比較的高機能なセンサに加えて、多数の小型かつ比較的小能力なセンサ[1][2]が接続されると考えられる。このような小型センサは、処理能力の制限から IP 等の高水準なネットワークプロトコルを提供できない場合がある。また電力等の制限から、要求に応じて応答を返すような通信形態をとることが困難なものがある。このため、専用の通信方式に基づくセンサネットワークの研究が行われてきた。まず、省電力な通信方式については、[3][4][5]等の省電力 MAC(Medium Access Control)プロトコルが研究されている。また一般に小型センサに付属するような低性能なアンテナでは利得は通信距離の 4 乗に比例して低下する[1]ため、小型通信セルによるマルチホップ通信が有効であり、かつ屋外等にセンサがばら撒かれた場合予めセンサ間の網構成を把握することは困難であるため、省電力なアドホックルーティングプロトコル[6][7][8]が提案されている。さらに、多数のセンサ群を集合的に扱うセンサデータベースの研究[9][10]も行われている。

（３）屋内用センサネットワークの提案

(3-1)　屋内用センサネットワークに関する前提

本稿で提案する屋内センサネットワークでは次のような前提を置いた。

超小型センサは、屋外等通常の場合と同様に電力線から電力が給電されるとは限らない。

センサ情報を受信する側は、電力が給電されある程度高出力で、かつ高性能なアンテナを持つ基地局を仮定することができる。このため、従来研究の小セルマルチホップ通信方式以外に、非対称な出力による通信方式も考慮できる。

屋内ではセンサの数や位置は比較的変化しない。

(3-2) 屋内用センサネットワークのモデル

上記の前提に基づき、ここでは以下のようなセンサネットワークを考えた。

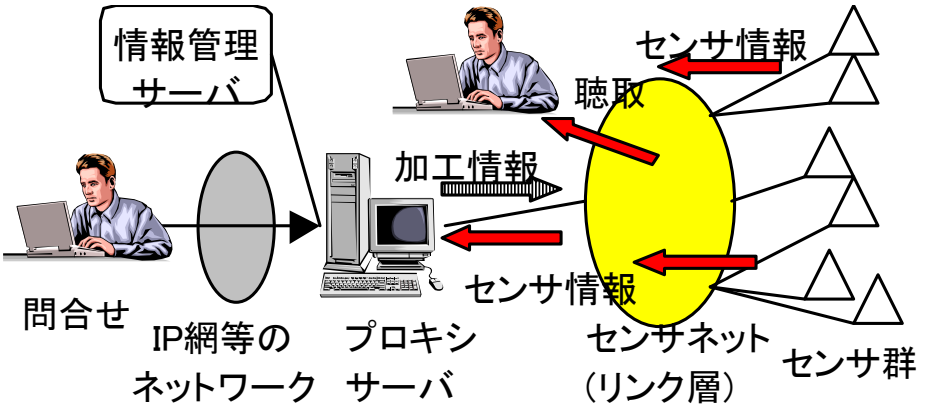


図 5.2-1　アプリケーションのモデル

§ アプリケーション層(図 5.2-1 参照)

センサは常にセンサ情報要求に対して応答できるとは限らない。そこでセンサは基本的にセンサ情報をネットワークに自律的に送出する。ユーザはネットワークを流れる情報を聴取することで、目的のセンサ情報を得る。

センサ情報はセンサの状況に応じて送出されるため、定期的に得られるとは限らない。また、センサ自身が自分の位置等をセンサ情報に付加するとは考えにくい。そこでネットワークには知的なノード(以下プロキシサーバと呼ぶ)が存在し、垂れ流し状態のセンサ情報を収集する。情報管理サーバからセンサ位置等を取得し、収集したセンサ情報を、時間軸あるいは空間的に補間して高次の情報(加工情報)を作成し、ユーザからの問い合わせに対して提供する。また加工情報を再びセンサネットワークに送出する。例えば、壁面の温度センサ群の情報を集約して、壁面温度として提供/送出する。

§ ネットワーク層(図 5.2-2 参照)

プロキシサーバが常に同一サブネットに存在することを仮定することは難しい。従って、ネットワーク層は必要である。不特定の相手に不定期に少量データを送出するアプリケーションが主であること、無線等バス型の下位層が利用されること、前提よりアドホックなルーティングは必要無いことを想定し、ブロードキャストアドレスによるフラッディングを主体に利用する。

TTL(Time-to-live)パラメタを小さく(例えば 2 から 3 程度)することで、ルータの中継により網全体にフラッディングされることを防ぐ。(図 5.2-2(a)) プロキシサーバが遠方にしかない場合は、IP 等

既存ネットワーク層上にトンネリングする。(図 5.2-2(b))

多数存在するセンサに IPv6 等のアドレスを付与することも検討されているが、ここでは MAC アドレスや応用レベルの ID 等多様なアドレス方式からネットワークアドレスとして選択して利用可能とし、センサへのアドレス割り当てを容易とする。

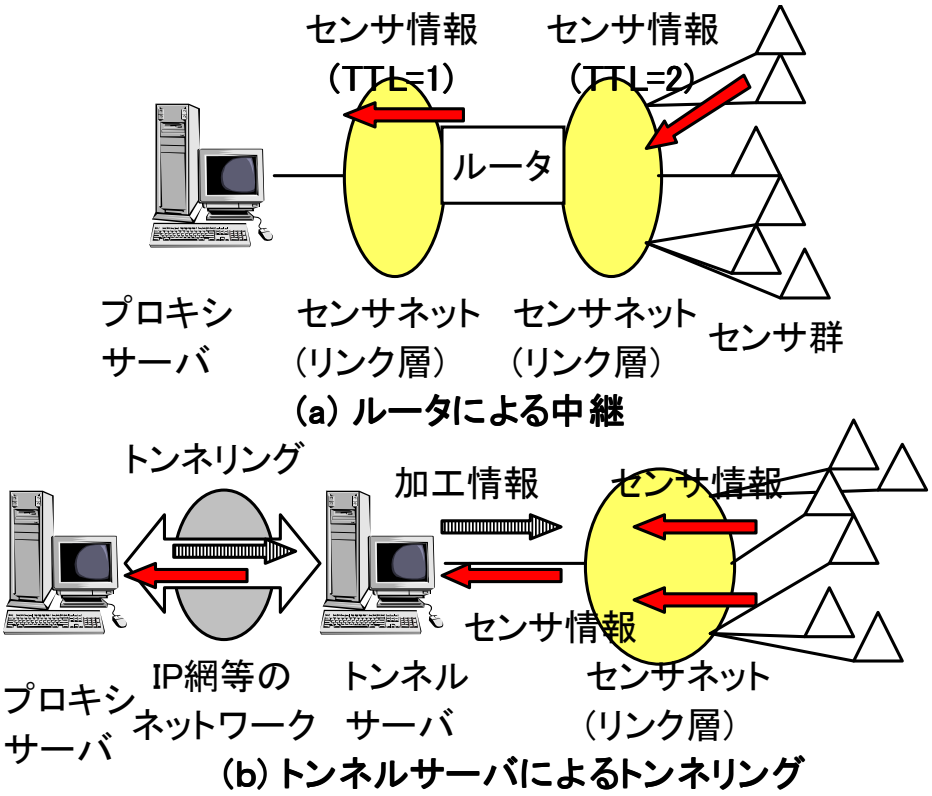


図 5.2-2　ルータによる中継とトンネリング

§ リンク層以下

小型アンテナを持つ超小型センサと高性能アンテナを持つ基地局との間の非対称な出力による無線リンクを想定する。

センサの省電力化には MAC 等のリンク層プロトコルが重要である。ここでは基地局が存在することと前提(3)により、[5]のような TDMA 等中央制御的な省電力 MAC プロトコルを利用することを想定する。MAC 方式の詳細については今後検討する。

(4) ネットワーク層プロトコルの設計

(3)で述べたモデルに基づきネットワーク層プロトコルの設計を行った。

(4-1) パケット形式

パケット形式は、以下の特徴を持つ。

複数のアドレス形式を同時に使用可能とするために､送受共にアドレス型とアドレス長を指定するパラメタを持つ。

センサ情報が主なデータであることを考慮し、パケット最大長は 4096 バイト程度とした。

設計したパケット形式を図 5.2-3 に示す。SAT と RAT がアドレス型を示すパラメタであり、省略（或いはブロードキャスト)、MAC アドレス、IP アドレス、eTRONID 等が指定できる。SAL 及び RAL がアドレス長を指定するパラメタであり､アドレス長が可変であるため全体としてヘッダ部は可変長となる。

version(2)	IHL(6)	Identification(8)	TTL(4)	Length (12)		
protocol(4)	flag(2)	Offset(10)	SAT(4)	RAT(4)	SAL(4)	RAL(4)
Source Address (0～128)						
Destination Address (0～128)						
Options if any						
User data follows						

図 5.2-3　ネットワークプロトコルのパケット形式

(4-2) ルータのアドレス解決

センサは複数の通信リンクを持たないことを想定し､センサはアドレス解決を行わない。

ユニキャストアドレスが受信アドレスに指定された場合､ルータは MAC アドレスへのアドレス解決が必要となる｡センサは不定期ではあるがセンサ情報を送出することから、特に ARP のような専用のアドレス解決プロトコルは設けず､ルータは受信パケットからアドレス解決表を維持する。

（５）まとめ

ここでは,屋内での利用を想定したセンサネットワークのモデル及び通信方式について提案した｡本モデルでは、センサが自律的センサ情報を送出しユーザあるいはプロキシサーバがその情報を利用する。

下位通信方式は、フラッディングによるネットワーク層と非対称出力を持つ無線リンクを想定した。

（イ） SL3 プロトコル

（１） 目的

各種センサからの情報を転送するためのネットワーク環境の構築を考えた場合、

ＩＰプロトコルではセンサに組み込む場合、処理が重いためセンサを小さくする事が困難となる。したがって、現在主流であるＩＰプロトコルよりも軽量（簡易）なプロトコルを作成し、軽量なプロトコルを用いてセンサネットワーク環境を構築する事によりセンサ部の小型化、小電力化を図る目的のため、ＳＬ３（Simple Layer 3）を作成する。

（２） 概要

ＳＬ３の主な機能概要を次に示す

- レイヤ３とレイヤ４を兼ねる位置づけとする
- 主にブロードキャスト転送を基本とする
- 再送制御は行わない
- ＴＴＬ（Time To Live）によりブロードキャストの範囲を制御する
- ＳＬ３のアドレスは多種な物を使用できるよう考慮する
- ＳＬ３のチェックサムは省略する

（３） 複数アドレス機能

ＳＬ３の独自機能として、複数のアドレスタイプをサポートする事が挙げられるが、複数のレイヤ３プロトコルをサポートする必要はない。

SAT、SAL、SourceAddress を一組として１つのアドレスとし論理的に１つのアドレスタイプとしてＳＬ３が処理する。また、１つの端末にて複数のアドレスを持つことを可能とする。

（４） マルチキャスト機能

アドレスにマルチキャストを指定してパケットを構成し、下位レイヤに渡す。

受信に関してはマルチキャストアドレスを判断しアプリケーションに渡す。

（５）データ分割・再構成機能

ＭＴＵに応じて送信データを分割しパケットを生成する。

ＭＴＵはインタフェース毎にレイヤ２から取得可能な場合は動的に取得する。

不可能な場合は、初期設定により固定値を割り当てる。

再構成の場合はflagをOffsetの値を用いてパケットの再構成を行う。

（６）ユニキャスト・マルチキャスト機能

ユニキャストとマルチキャストのパケットを送受信可能とする。

但し､マルチキャスト機能に条件文を追加する程度で用意に実装可能と考えられるため、マルチキャスト機能と同時に実装しても良い。

（７）　ping 機能

簡易管理機能として ping 機能を実装する。

ICMP と同等の基本的な機能および操作性を実現する。

ユーザデータ領域には、ping のシーケンス番号（４バイト程度)、要求、応答フラグ、任意のデータ長（コマンドオプションで長さを設定する、デフォルトは３２バイト程度）のデータをパディングする。

（８）　ＳＬ３パケットフォーマット

ＳＬ３プロトコルヘッダは最小で８バイト､最大で４０バイトとなる。

表 5.2-1 に SL3 パケットヘッダのフォーマットを示す

A)　　表 5.2-1　ＳＬ３パケットフォーマット

version(2)	IHL(6)	Identification(8)	TTL(4)		Length(12)	
Protocol(4)	flag(2)	Offset(10)	SAT(4)	RAT(4)	SAL(4)	RAL(4)
Source Address (0～128)						
Destination Address (0～128)						
Option						
User Data						

(8-1) 各フィールドの説明

(a) Version

２bit の領域によりＳＬ３プロトコルのバージョンを表す。

今回の開発時は固定値で１を設定する。

(b) IHL

６bit の領域でパケットヘッダ長を表す。

１バイト単位でプロトコル依存のオプションフィールドまでを含む長さを表す。

６bit で表せる最大値は 63 であるが､0 はありえないためこの場合の値を最大値「64４」と考える。

(c) Identification

８bit の領域によりパケットのシリアル番号を表す。

0xff まで使い切った場合は 0x00 に戻る（ラウンドする)。

(d) TTL (TimeToLive)

４ビットの領域で TTL を表す。

IP プロトコルの TTL と同意である。

(e) Length

１２bit の領域でパケット長を表す。

ヘッダを含む全てのパケットの長さで、１バイト単位で最大 4096 となる。(MTU は 4096 とする)

12 bit で表せる最大値は 4095 であるが、0 はありえないためこの場合の値を最大値「4096」と考える

(f) Protocol

４bit の領域でプロトコル種別を表す。

表 5.2-2 プロトコル種別

プロトコル種別	値
ＳＤＰ（SensorDataProtocol）	0x01
ＳＣＰ（SensorControlProtocol）	0x02
ＳＰＰ（SimplePingProtocol）	0x08
ＳＲＰ（SL3RoutingInformationProtocol）	0x0a

(g) flag

２bit の領域でフラグを表す。

１bit 目は分割禁止、２bit 目はモアビットを表す

Protocol				1	2	Offset	
－	－	－	－	(A)	(B)	－	－

(A) 分割禁止を表す bit　（don't fragmentation）
(B) モアビットを表す bit　（more fragmentation）

図 5.2-4 フラグビットの詳細

(h) Offset

１０bit の領域で分割のオフセットを表す。

本領域はパケット分割が発生した場合、4 バイト単位のオフセットを示す値を設定する。

(i) SAT (SendAddressType)

４bit の領域でアドレス型を表す。

本領域の値が０の場合は省略を表し、１の場合は MAC アドレス、２の場合は IP アドレス、３の場合は eTronID（128bit）をＳＬ３の送信アドレスとして使用している事を表す。

(j) RAT (ReceiveAddressType)

４bit の領域でアドレス型を表す。

本領域の値が０の場合はブロードキャストを表し、１の場合は MAC アドレス、２の場合は IP アドレス、３の場合は eTronID（128bit）をＳＬ３の受信アドレスとして使用している事を表す。

(k) SAL (SendAddressLength)

４bit の領域で送信アドレスの長さを２バイト単位で表す。

アドレスが省略の場合のアドレス長は０となる。

(l) RAL (ReceiveAdderssLength)

４bit の領域で受信アドレスの長さを２バイト単位で表す。

ブロードキャストの場合はアドレス長は０となる

(m) SourceAddress

0～128 bit の領域で送信アドレスを表す。

省略の場合、本領域は存在せず、SAT で指定したアドレスが設定される。

(n) DestnationAddress

0～128 bit の領域で送信アドレスを表す。

本領域はブロードキャストの場合は存在せず、RAT で指定したアドレスが設定される。

(o) Option

本領域はプロトコルに特化したパラメータを指定するための領域であるが、本開発段階では本領域は未定義である。

(p) UserData

コード（１バイト)、レングス（２バイト)、バリュー（ｎバイト）の形で設定する。

詳細は、（９）参照のこと。

（９） ユーザデータ

ＳＬ３プロトコルにより送受される各プロトコルのユーザデータフォーマットを図 5.2-5 に示します。

ユーザデータフォーマット領域のフォーマット

パラメータグループコード（1）	パラメータグループ長（2）	パラメータコード（1）	パラメータ長（2）	パラメータ値（n）	パラメータ	…
		パラメータ領域				

※ 括弧内はフィールド長を示す

※ パラメータグループ長は上記パラメータ領域の長さを表す

図 5.2-5 ユーザデータフォーマット

(9-1) パラメータグループコード

パラメータグループに設定するグループコードを以下に示します。

表 5.2-3 パラメータグループコード

0x01	センサデータ	温度センサ(0x?1)

0x81	センサデータ取得要求	温度センサ(0x?1)
0x91	センサインターバル時間設定要求	温度センサ(0x?1)
0xa0	SL3 Rip	－
0xf0	SL3 Ping	－

パラメータグループコード下位４bit はセンサ種別を表すコードと考え、 0x01、0x81、0x91 のパラメータグループコード内の????0001 は温度センサを表す。

(9-2) センサデータ

センサデータ送信時のパラメータグループ内パラメータの種類を次に示す。

表 5.2-4 センサデータパラメータ

パラメータ	コード	領域長	説　明	値　例
Version	0x01	1Byte	SDP のバージョン番号	0x00
SensorType	0x02	1Byte	使用しているセンサのタイプ	0x00 (LM92)
SensorInfoType	0x03	1Byte	センサ情報のタイプ	0x00 (温度データ)
SensorInfo	0x04	2Byte	センサ情報	0x0000
eTronID	0x0f	16Byte	eTron ID	16 バイトデータ

(9-3) センサデータ取得要求

センサデータ取得要求時のパラメータグループ内のパラメータの種類を次に示す。

表 5.2-5 センサデータ取得要求パラメータ

パラメータ	コード	説　明	値　例
なし	-	-	-

センサデータ取得要求時の、センサからの応答は通常のセンサデータで応答される。

(9-4) センサインターバル時間設定要求

センサインターバル時間設定要求時のパラメータグループ内のパラメータの種類を次に示す。

表 5.2-6 センサインターバル時間設定要求パラメータ

パラメータ	コード	領域長	説　明	値　例
IntervalTime	0x05	4Byte	インターバル時間（ms）	60000（６０秒）

(9-5) SL3Ping

SL3 PING 送信時のパラメータグループ内のパラメータの種類を次に示す。

表 5.2-7 SL3 PING 送信時パラメータ

パラメータ	コード	領域長	説　明	値　例
SequenceNumber	0x06	4Byte	シーケンス番号	0x00000000
Flag	0x07	1Byte	フラグ	0x01:要求/0x80:応答
Ping Data	0x08	nByte	詰め物データ	デフォルト３２バイト
eTronID	0x0f	16Byte	eTron ID	16 バイトデータ(※１)

※１　センサからの応答時のみ設定。

(9-6) SL3Rip

SL3 RIP 送信時のパラメータグループ内のパラメータの種類を次に示す。

表 5.2-8 SL3 RIP 送信時パラメータ

パラメータ	コード	領域長	説　明	値　例
RoutingInfo	0x09	24Byte	SL3 ルーティング情報	※２

※２　SL3 ルーティング情報領域のフォーマット

SL3 アドレスタイプ（１）	アドレス情報（１６）	MAC アドレス（６）	状態フラグ（１）

（１０）　ＳＬ３ルータ機能概要

(10-1) ブロードキャストパケット転送機能

ある一方のインタフェースから他のインタフェースにブロードキャストパケットを転送する機能である。

なお､パケットループの原因になるため同じインタフェースへは送出を行わない。

ブロードキャストする際にはパケットヘッダ内のＴＴＬから１減算しＴＴＬが０の場合は転送動作を行わない。

(10-2) パケットフラグメント機能

パケットサイズがＭＴＵを越えた場合、パケットの分割を行う。

ＭＴＵはインタフェース毎にレイヤ２から取得可能な場合は動的に取得する。

不可能な場合は初期設定にて固定値を割り当てる。

(10-3) ルーティングテーブル自動生成機能

SourceAddress を含むパケットを受信した場合に、自インタフェース毎にそのＳＬ３アドレスと、Next Hop となるレイヤ２のＭＡＣアドレスの対応表を作成する。

付加機能としてルーティングテーブルの一括クリアなどの管理操作機能を実装する。

(10-4) ユニキャスト・マルチキャストパケット転送機能

ルーティングテーブルを参照して､合致するインタフェースにパケットを送出する。

合致しない場合はパケットを破棄し､エラーログをローカルファイルに出力する。

(10-5) ルーティングテーブルの広報機能

作成したルーティングテーブルを単純にブロードキャストにて広報する。

使用するプロトコルとしてはＳＬ３にこだわらずＩＰを使用してもかまわない。

インタフェース毎に作成されたルーティング情報テーブルを､他のインタフェースにブロードキャストにて送信する｡ＴＴＬは０で送信する。

他のルータからのルーティングテーブルを受信した場合テーブル

に新規・更新・削除などのエントリがあれば、自分のルーティングテーブルのエントリを追加・更新・削除を行う。

ルーティング情報テーブルの各エントリには状態フラグ（正常・新規・更新・削除)、および最終更新日時を付加する。

状態フラグは一度ルーティングテーブルを広報すると、「新規」と「更新」は正常となり、「削除」は実際に情報テーブルから削除する。「削除」となるのは手動で設定した場合のみとする。

(10-6) ルーティング情報の設定

ルーティング情報の設定において上記した自動に生成、広報されるテーブルと、ユーザにより入力される静的なテーブルの２種類を用意する。

（１１） プロキシ機能概要

(11-1) センサデータ収集・蓄積機能

センサからのブロードキャストパケットを収集し、センサデータをデータベース(Postgres)に格納する。

(11-2) センサデータ検索機能

リモートからの SQL に応答する機能であるが、基本的にはデータベースの機能が満たしていれば問題ない。但し、プロキシローカルで検索し、結果を表示する簡易な検索用 UI を実装する。

(11-3) 送出間隔リモート設定機能(設定操作側)

センサ名(ID または種類)、属性値名(送出間隔)、設定値(時間)を指定し、センサにメッセージを送信する。センサの属性値が正常に変わったかどうかの応答は返らないとする。簡易な MIB を定義する必要がある。

(11-4) センサデータ要求・応答型通信(要求側)

センサ名(ID)、属性値名を指定してセンサデータを要求し、応答を受信する。さらに応答に含まれるセンサデータをデータベースに格納する。例えば SNMP の GET オペレーションのようなイメージ。簡易な MIB 定義が必要である。

(11-5) 加工データの SL3 ネットワークへの送出機能

収集したデータを解析・集約して新たな情報を生成する。例えばある部屋の温度の平均値を算出する、などを行う。また、生成した情報は、SL3 ネットワークにブロードキャストで送信する。テスト用 SL3 ネットワーク内に簡易な受信端末(受信して内容を表示するのみ)を作成して設置する。

（１２） 擬似センサデータ発生ツール

(12-1) 温度センサ模擬機能

温度センサの送出するパケットを模擬する。最大 1000 個のセンサを模擬することが可能とする。センサ ID、送出間隔、送出値(平均)、変動幅(一様分布)、データ発生開始時刻(時刻指定またはランダム)を一組としたリスト(シナリオ)を解析するパーサを作成する。シナリオをパーサにかけて、時刻毎にデータを送出するセンサ ID を列挙するテーブルを作成し、そのテーブルに従ってセンサデータを送出する。

(12-2) シナリオ自動生成機能

上記シナリオを自動生成する。センサの種類(温度センサのみ)、送出間隔(固定またはランダム)、データ発生開始時刻(固定またはランダム)、センサ数を設定すると、自動的にセンサ ID を割り当ててシナリオを作成する。

(12-3) 他のセンサの模擬機能

温度センサ以外のセンサを模擬する。センサ名、値の範囲、値の単位などを入力して新規センサを登録し、上記のシナリオ自動生成機能によりシナリオを作成する。

（１３） センサハードウェア

(13-1) 目的

環境情報をシステム内に取り込む目的で、μT-Engine にセンサを接続しプログラムによりセンサ情報を取り組み、ＳＬ３プロトコルを使用してネットワークへ送出する。

(13-2) 概要

本開発では温度センサを用い、本装置が設置される部屋の室温を取得し、ＳＬ３プロトコルを使用し温度情報を収集する。

(13-3) センサインタフェースについて

μT-Engine と 接続されるセンサH／W間のインタフェースは８pin(8bit)の単純な CPU マッピング領域が存在するのみである。この領域を使用し、プログラムとセンサH／Wとのインタフェースを行う方法を次に示す。

センサH／Wからのセンサデータの取得方法としては次に示す２通りの取得方法が考えられる。

センサH／Wによる垂れ流しデータをセンサＰＧ側にて定期的に拾い読みする

プログラムとセンサH／W間で簡易的なハンドシェークを行い、プログラム側からの要求時にH／W側からセンサデータを通知する

１の場合は今後のセンサ種別追加などの拡張性が失われるため、本開発では２の案にてインタフェースを取り決める事とする。

（１４） センサH／Wインタフェース

μT-Engine と 接続されるセンサH／W間のインタフェースは８pin(8bit)の単純な CPU マッピング領域が存在するのみである。この領域を使用し、プログラムとセンサH／Wとのインタフェースを行う方法を次に示す。

方法としては、プログラムとセンサH／W間で簡易的なハンドシェークを行い、プログラム側からの要求時にH／W側からセンサデータを通知する

(14-1) プログラムとH／W間のシーケンスについて

プログラム側から見てH／W側とのインタフェースには８bit の領域を使用し、Read、Write を行うこととなる。この領域を有効に使用し、多種に及ぶセンサにできるだけ対応できる形でインタフェースを定義する。また、センサからのセンサ値が８bit で間に合わない場合を考慮する必要があるため、センサデータを最大４バイトとし、最大４回のシーケンスによりH／Wよりセンサデータを受け取ることとする。

(14-2) 通信シーケンス

プログラムとH／W間の通信シーケンスを次に示す。

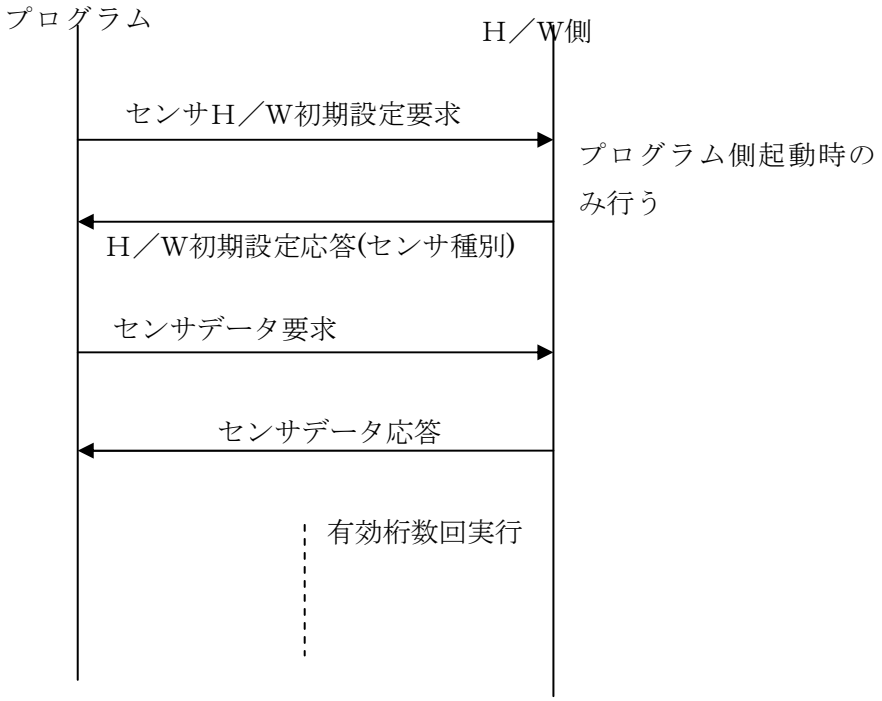


図 5.2-6 通信シーケンス

(14-3) 各シーケンス詳細

次に各シーケンスの詳細を示す。

プログラム側からの要求の書き込み終了後の、読み込みを行うまでの間、H／W処理時間として数ミリ秒プログラム側にて読み込みを待たせる必要がある。

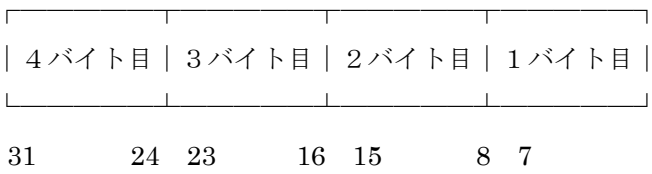
(a) プログラム側からの要求コマンド各 BIT 説明

数種類のセンサをサポートする場合、種別を表すＩＤが必要となる。

このＩＤをセンサデータ要求時にセンサHWへ通知し、対象センサの値を取得できるように考慮する。ただし、今後サポートするセンサの数を現状では最大４とし、センサからの値は３２bit 固定とする。上位２ＢＩＴにセンサからの値のバイト番目を識別する意味づけを行いＰＧ側から要求するバイトの番目を指定する。

bit 定義を次に示す。

・00：１バイト目
・01：２バイト目
・10：３バイト目
・11：４バイト目



※要求送信時は、各 bit 値の or 値で通知する。

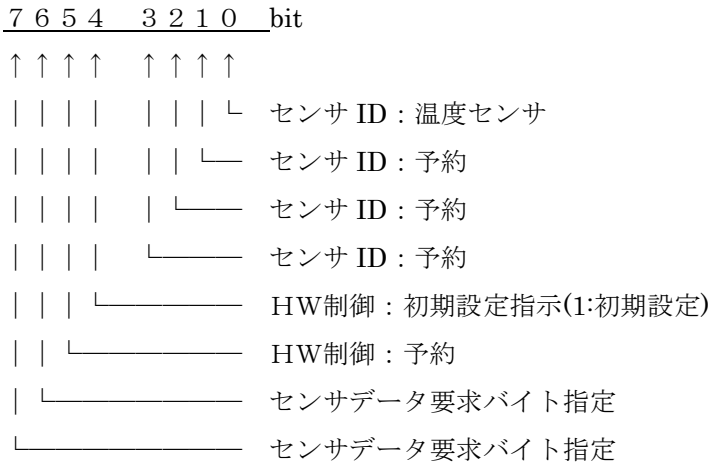


図 5.2-7 プログラム側からの要求コマンド各 BIT 説明

(b) 各要求時の値

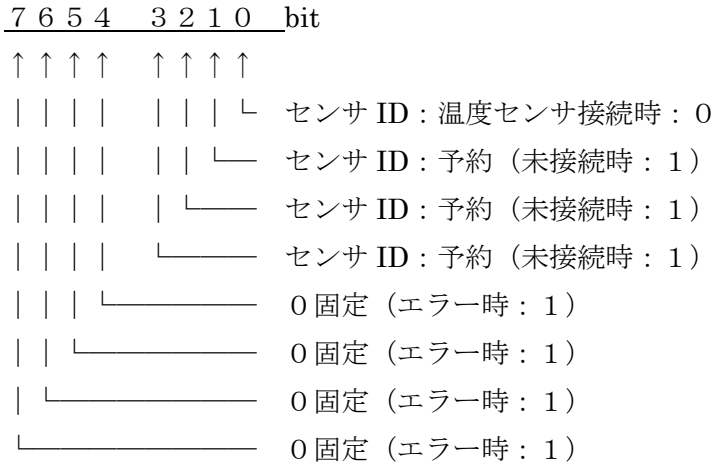
本開発（温度センサ）における各要求時の値を次に示す。

なお､データ要求時は有効桁数の高いバイト番目より要求することとする。

- 初期設定要求：０ｘ１０
- 温度センサデータ要求（２バイト目）：０ｘ４１
- 温度センサデータ要求（１バイト目）：０ｘ０１

(c) H／W側からの応答コマンド各 BIT 説明

初期設定応答



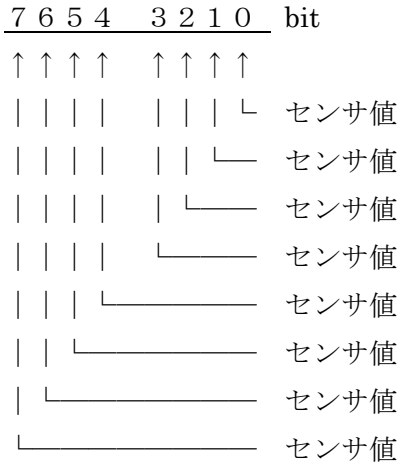
H／W初期設定の結果応答としてH／W側にて現在接続されているセンサを返答する。現状使用できるセンサは温度センサのみであるため、本開発における応答の正常値は０ｘ０ｅとなる。

※H／W初期設定に対するエラー時は値が０ｘＦＦとなる（HW未接続時も同様）

図 5.2-8　H／W側からの応答コマンド各 BIT 説明

データ応答

プログラム側からのデータ要求で指定するバイト目（１、２、３、４バイト目）のセンサから取得した値を各 bit へ設定しプログラム側へ通知する。この場合センサによっては未使用バイトが発生するがこの場合は、値を０ｘＦＦで返信する。（プログラム側にてセンサ毎の有効バイト数を把握している必要がある）



センサにより応答されるバイト数に差があるため、応答は必ず４バイトとする。
センサにより未使用なバイト領域は０ｘＦＦを応答する

図 5.2-9 データ応答ビットの説明

（１５）　Linux 側ＳＬ３インタフェース

Linux における SL3 プロトコルのインタフェースはライブラリとしてプロトコルの切り口を設け、そのライブラリ関数の呼び出しにより SL3 プロトコルを使用することを可能とする。

なお、本ライブラリはランタイムライブラリとしシンボリックリンクを設定することにより、バージョンアップ時などには、上位ＡＰのリンクなどを行わない形式とする。

(15-1) ＡＰＩ

上位ＡＰへ提供する関数の説明を以下に示す。

(15-2) SL3 アドレス情報格納テーブル

アドレス情報格納テーブルは、ＡＰＩ関数にて共通に使用するテーブルである。

(a) テーブル定義

アドレス情報格納テーブルの定義を以下に示す。

```
struct sl3addr_info {
```

```
        int     flag;
        char   info[512];
};
```

flag　　SL3 プロトコルまたは、アドレス種別
info　　コンフィグレーションファイルパスまたは､アドレスを設定するバッファ領域

(b) メンバの意味合い

アドレス情報格納テーブルは使用するインタフェース関数によりメンバの意味合いが変化する。

呼び出す関数後毎の意味合いを以下に示す。

SL3Open（）時

SL3Open 時は、本テーブルは使用するプロトコルの指定（flag）とコンフィグレーションファイルのパス（設定無変更時は長さ０）を指定する。

SL3Send（）時

送信相手の SL3 アドレスを指定する。

SL3Recv（）時

送信者のアドレスを格納する。

(15-3) 各ＡＰＩ関数の説明

ＳＬ３ライブラリにて提供する各関数の説明を以下に示す。

(a) SL3Open

プロトタイプ

```
int SL3Open( struct sl3addr_info *sl3addr );
```

関数説明

SL3 インタフェースをオープンします。

SL3Open 関数のコール時に使用するＳＬ３のプロトコルを指定し、指定箇所は「sl3addr_info」内の flag フィールドを使用します。

引数

表 5.2-9　SL3Open の引数

変数名	説　明
sl3addr	SL3 アドレス情報テーブルのアドレス

プロトコル種別　(flag)

１：SDP（センサデータプロトコル）
２：SCP（センサコントロールプロトコル）
８：SPP（センサＰＩＮＧプロトコル）

コンフィグレーションファイルパス　(info)

「sl3addr_info」内の info フィールドはＳＬ３用のコンフィグレーションファイルのフルパスを指定する。（このファイルにはＳＬ３の設定を行う各種パラメータが記述されているが、設定を特に変更しない場合は、本領域は文字列長０の配列として関数を呼び出す。）

戻り値

関数の戻り値はディスクリプタとして、SL3Close、SL3Send、SL3Recv で使用する。
エラー時は＜０の値が戻る。

(b) SL3Close

プロトタイプ

```
int SL3Close( int sl3_fd ) ;
```

関数説明

SL30pen でオープン済みのインタフェースをクローズする。

引数

表 5.2-10 SL3Close の引数

変数名	説　明
sl3_fd	SL3Open()の戻り値

戻り値

正常時は＞＝０の値が戻る。
エラー時は＜０の値が戻る。

(c) SL3Send

プロトタイプ

```
int SL3Send( int sl3_fd, struct sl3addr_info *sl3addr, char
*send_data, int size );
```

関数説明

SL3 プロトコルにて引数でしていするデータを送信する。
Sl3Send 関数にて指定する sl3addr 内の length 領域で、使用する相手アドレス種別を表す。
（実際の長さでは無い）
値 4：IP アドレス／6：MAC アドレス／16：eTron-ID

これらアドレスの文字列イメージを addr の領域へポインタ設定する。

- IP アドレスは ． 区切りの数字文字列（10 進イメージ）
- MAC アドレスは ： 区切りの数字文字列（16 進イメージ）
- eTronID は - 区切りの数字文字列（16 進イメージ）

引数

表 5.2-11 SL3Send の引数

変数名	説　明
sl3_fd	SL3Open()の戻り値
sl3addr	送信先アドレス指定
send_data	送信データのアドレス
size	送信データ長

戻り値

正常時は＞＝０の値が戻る。
エラー時は＜０の値が戻る。

(d) SL3Recv

プロトタイプ

```
int SL3Recv( int sl3_fd, struct sl3addr_info *sl3addr, char
```

```
*recv_data, int size );
```

関数説明

SL3 プロトコルのデータを受信する。
受信データが存在しない場合、本関数はブロックし制御は戻らない。
sl3addr の領域は、受信バッファ同様に上位で確保しライブラリへ渡す。
この領域「sl3addr」の意味合いは、SL3Send と同様である。

引数

表 5.2-12 SL3Recv の引数

変数名	説　明
sl3_fd	SL3Open()の戻り値
sl3addr	送信元アドレス
recv_data	受信データの格納バッファ
size	送信データの格納バッファ長

戻り値

正常時は＞＝０の値が戻る。
エラー時は＜０の値が戻る。

（１６）　μT-Engine 上のタスク構成

μT-engine 上で動作するプログラムはタスク構成をとり、複数個の処理をタスクと呼ばれる構成で作成する。以下にタスク構成図を示す。

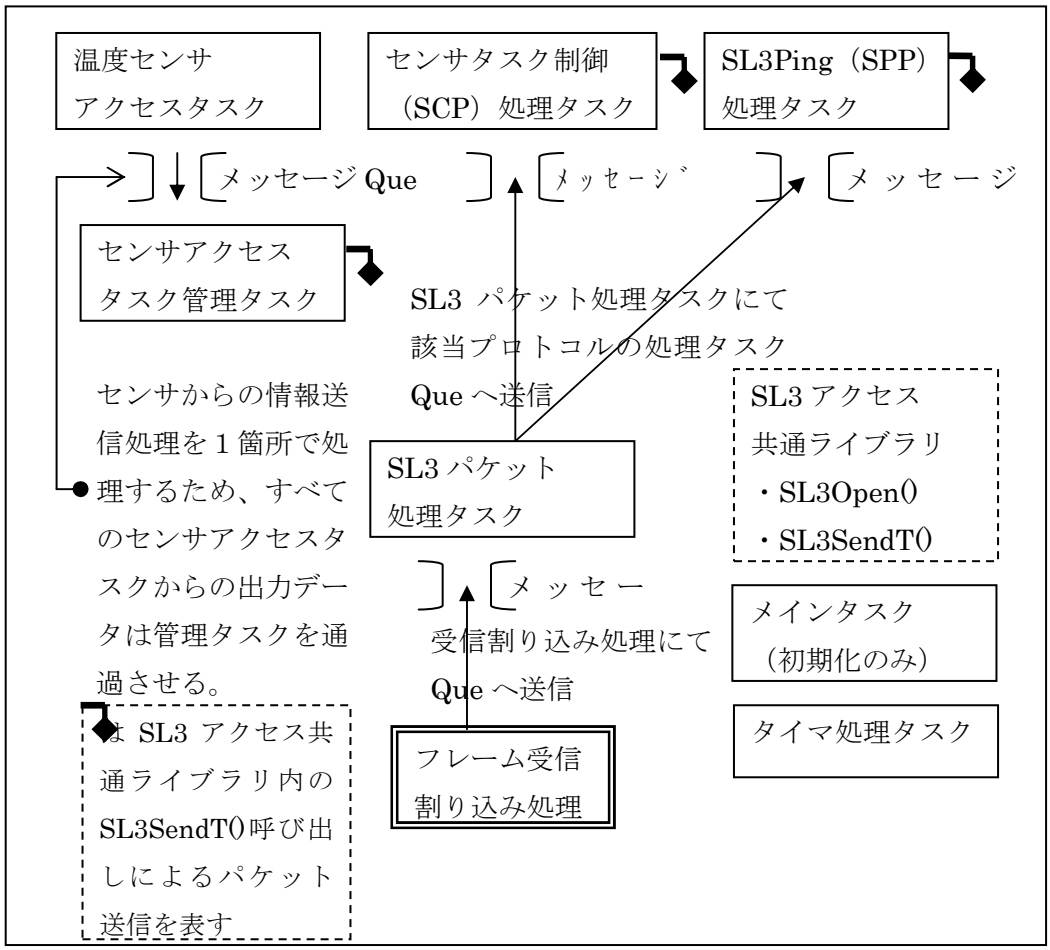


図 5.2-10　μT-engine 上のタスク構成図

(16-1) タスク

μT-engine 上で動作するタスクはタスク ID を付与し OS により管理される。

各タスクのタスク ID を表 5.2-13 に示す。

B)　　表 5.2-13　タスク ID 一覧

No	タスク名称	タスク ID
1	メインタスク	１０
2	SL3 パケット処理タスク	１１
3	タイマタスク	１２
4	センサアクセスタスク管理タスク	２０
5	センサタスク制御タスク	２１
6	温度センサアクセスタスク	２２
7	SL3Ping 処理タスク	３０

(16-2) タスク間Ｉ／Ｆ

本開発においては､各タスク間においてデータをやり取りする方法として外部変数を用いた方式と､メッセージキューを用いた方式を採用している。

メッセージキューは送信されたメッセージを受信する側がメッセージキューを作成し､送信タスクはそのキューに対して書き込みを行う。各タスクにて作成するメッセージキューの番号を表 5.2-14 に示す。

C)　　表5.2-14　メッセージキュー番号一覧

No	生成タスク	キュー番号
1	センサアクセスタスク管理タスク	２５
2	センサタスク制御タスク	２１
3	SL3Ping 処理タスク	３０
4	SL3 パケット処理タスク	１１

(16-3) 各タスク説明

各タスクの説明を以下に示す。

(a) センサタスク管理タスク

本タスクはセンサアクセスタスク用の初期処理および､センサハードウェアの初期処理を行うタスクである。

初期処理

センサハードウェアに対する初期処理（リセット要求）を行うとともに、コンフィグレーションファイル「utsmng.cfg」をタスク初期処理にて読み出し、自 eTronID の値、プロキシへのユニキャスト用 SL3 アドレスの値、を外部変数に格納する。

タスク処理

センサアクセスタスクから受け取ったデータを SL3 アクセス共通ライブラリ内の SL3SendT（）関数を用いて、送信する。

(b) 温度センサアクセスタスク

本タスクは実際にセンサハードウェアへアクセスし実際の温度情報を取得し、温度情報データを作成するタスクである。

初期処理

コンフィグレーションファイル「utsTemp.cfg」をタスク初期処理

にて読み出し、温度センサ情報を読み出すインターバル時間値を外部変数に格納する。

タスク処理

インターバル時間の監視を行い、時間経過後、センサハードウェアへアクセスし温度情報を読み出し、センサ情報データのパラメータ部を作成後、センサタスク管理タスクへのメッセージキューへデータを送信する。

(c) センサタスク制御タスク

本タスクは SL3 プロトコルで運ばれるセンサ制御用のプロトコル SCP の処理を行うタスクである。SCP の内容としては、センサ情報の取得（GET）および、センサ情報読み出しタイミングの設定（SET）の２つの機能がある。

初期処理

初期処理としての特別な処理は行わない。

タスク処理

センサ情報の取得を受信した場合は、温度センサアクセスタスクへ対しセンサハードウェアからの強制温度情報読み出し指示を行う。また、センサ情報読み出しタイミングの設定要求を受信した場合は、タイミング時間値を受信データより取り出し、外部変数へ格納する。

(d) SL3Ping 処理タスク

本タスクは SL3 プロトコルで運ばれる SL3Ping の応答処理を行う。

初期処理

初期処理としての特別な処理は行わない。

タスク処理

SL3Ping 要求を受け取った場合、要求内容を解析し要求を送信した相手に対して、SL3Ping の応答を、自 eTronID をパラメータとして追加し返信する。

（ウ） 実証実験

本研究で開発したセンサネット用ネットワークプロトコル（SL3）

を実証するために、センサ、ルータおよびプロキシサーバに本プロトコルを実装し、評価を行った。

（１）　機器の構成

図 5.2-11 に実証実験で使用したシステムの全体構成を示す。このうちセンサネットの実証実験では、図下半分が対応する。

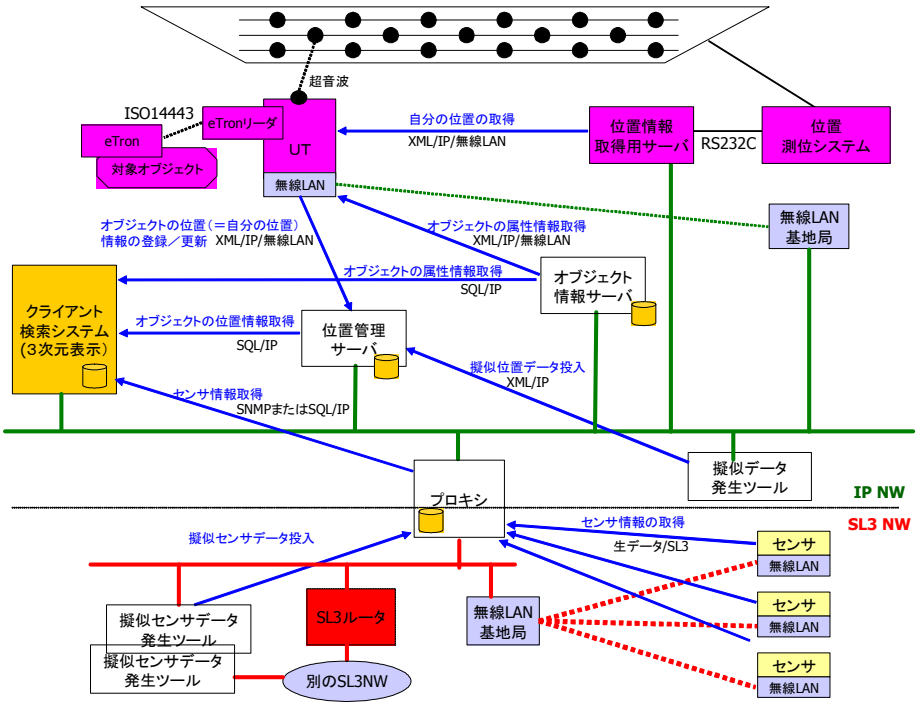


図 5.2-11　実証実験の構成

図 5.2-11 において、センサはμT-Engine と温度測定用 IC からなる温度センサであり、複数組存在する。センサは SL3 プロトコルにより測定データをネットワークに垂れ流している。プロキシサーバはネットワークを流れるセンサ情報を収集し、他のアプリケーションに提供する。（ここではクライアント検索システム）　このほかに、性能確認のための擬似センサデータ発生ツールと SL3 用のルータを実装した。

（２）　評価に使用した機器の状況

(2-1) センサ

センサは、μT-Engine を制御用コンピュータとし、温度測定用の

IC と組み合わせて、温度センサとした。

(a) センサ端末μT-Engine 部分

センサ側に使用したμT-Engine を図 5.2-12 に示す。

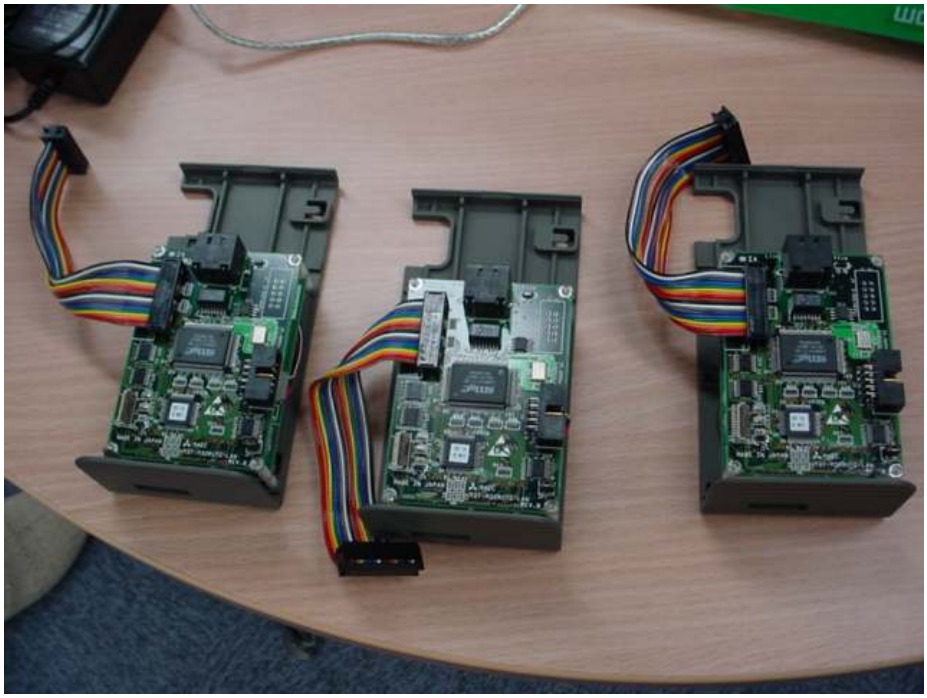


図 5.2-12 センサ側に使用したμT-Engine

(b) 温度センサ部

温度測定用 IC を利用した温度センサを図 5.2-13 に示す。

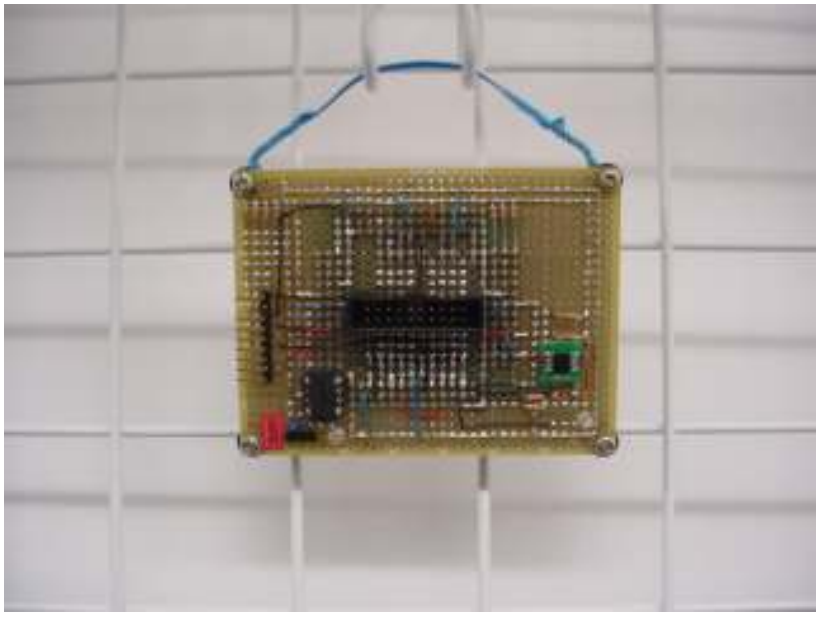


図 5.2-13 温度測定用 IC を用いた温度センサ部

(2-2) センサの設置状況

図 5.2-14 および図 5.2-15 にセンサの壁面への設置状況を示す。また図 5.2-16 にセンサ部の μT-Engine への接続状況を示す。

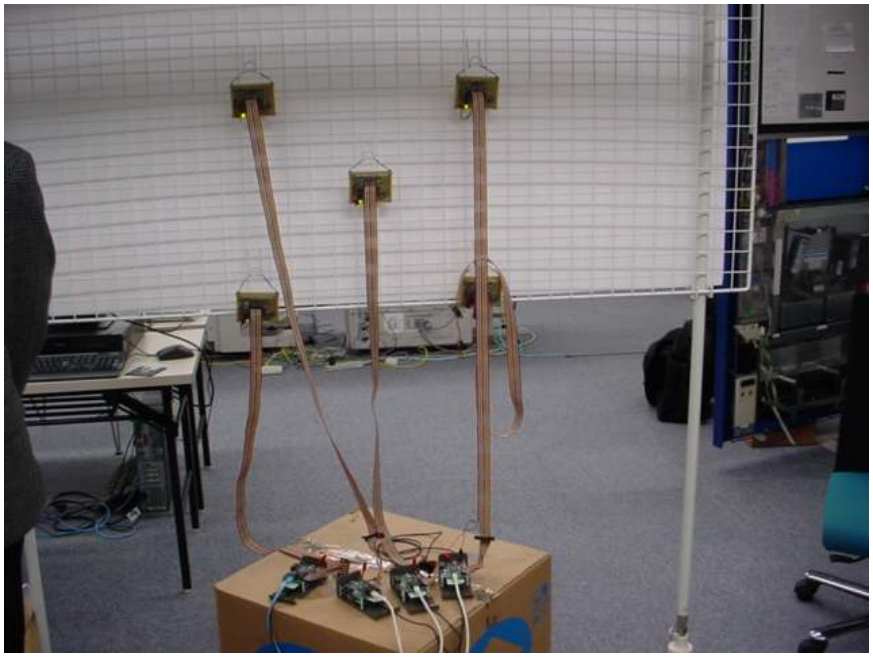


図 5.2-14 センサの設置状況

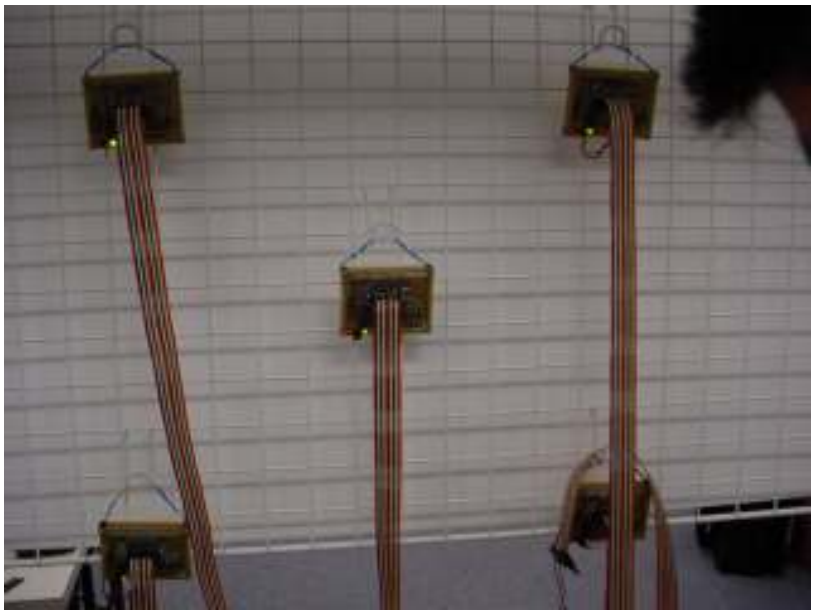


図 5.2-15 センサの設置状況の拡大図

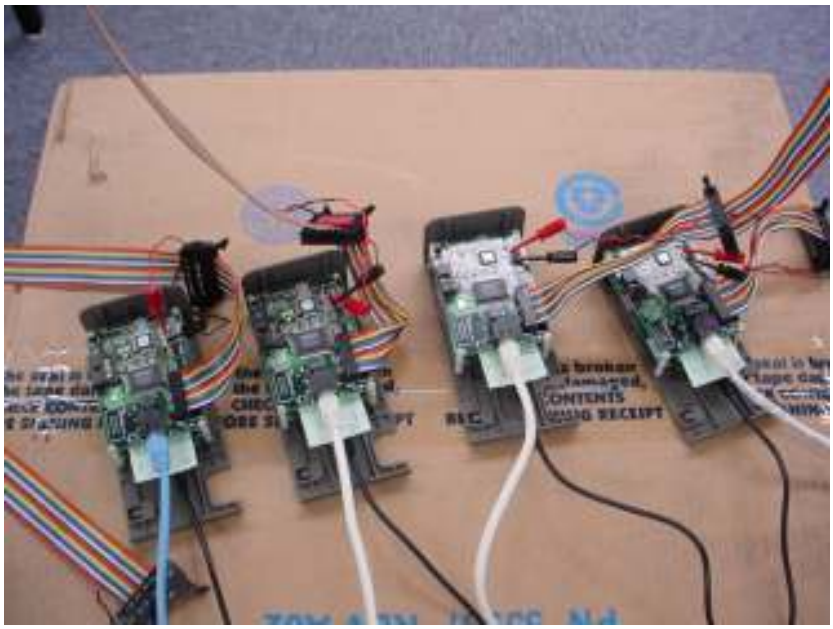


図 5.2-16 センサの μT-Engine への接続状況

参考文献

[1] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors", in Proc. of ACM Mobicom'00, 2000.

[2] J. M. Kahn et. al, "Next Century Challenges: Mobile Networking for Smart Dust", in Proc. of Mobicom'99, 1999.

[3] A. Woo and D. Culler, "A Transmission Control Scheme for Medium Access in Sensor Networks", in Proc of Mobicom'01, 2001.

[4] K. Sorabi et. al., "Protocols for Self-Organization of a Wireless Sensor Network", IEEE Pervasive Communication, Oct., 2000.

[5] E. Shih et. al., "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks", in Proc. of ACM Mobicom'01, 2001.

[6] W.R. Heizelman et. al., "Energy-Efficient Communication Protocol for Wireless Sensor Networks", in Proc. of IEEE Hawaii Int'l Conf. on Sys. Sci., 2000.

[7] C. Intanagonwiwat et. al., "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", in Proc. of ACM Mobicom'00, 2000.

[8] L. Li and J. Y. Halpern, "Minimum-Energy Mobile Wireless Networks Revisited", in Proc. of ICC'01, 2001.

[9] COUGAR Device Database Project, http://www.cs.cornell.edu/database/cougar/index.htm

[10] DataSpace Project, http://www.cs.Rutgers.edu/dataman/

5.2.2 随意拡張型固定網通信プロトコルの研究

ユビキタス社会になると、一般家庭にも通信機能を搭載した超小型チップが設置されるようになる。これらの超小型チップは照明や空調といった住宅の設備機器に搭載されるほか、家庭内のいたるところに設置されている温度センサや湿度センサ、人感センサなどにも搭載され、互いにネットワークに接続されて省エネルギーと快適性を実現できるようになる。しかしながら、一般家庭において利用されることを想定すると、このネットワークは従来の IP ネットワークでは実現されていない、いわゆる Zero Administration 性(随意拡張性)が必要となる。また、制御を行うことからリアルタイム性が必要である。そこで、平成 14 年度は、随意拡張型固定網通信プロトコルの研究の一環として、これらの設備機器を協調動作させリアルタイムで処理するためのネットワークプロトコルを研究した。

(ア) はじめに

制御系のプロトコルとして、情報家電の制御を目的とした ECHONET、工場やビルの設備管理を目的とした LonWorks や FL-net、車内ＬＡＮ向けの ARCNET とその改良型の EC-NET などがある。これらのプロトコルにはそれぞれ長所があり分野によっては標準的なプロトコルとなっているものもある。

一方、ユビキタス社会では、我々が日々生活を行う住宅そのものの設備機器（照明・空調・窓・壁など）自身が通信機能を持ち、設備機器自身が周囲の状況を判断して省エネルギー化を図ったり、あるいは我々の生活空間を快適にしてくれたりすることが求めらる。

(イ) プロトコル概要

(1) リアルタイム性とアクセス方式

住宅の設備機器（ノード）をネットワークに接続する場合、人間が操作をしてから機器が反応するまでの応答時間、つまり最大待ち時間の保証がネットワークの使い勝手に重要な影響を与える。このため、ネットワークにはリアルタイム性が求められる。リアルタイム性を確保するには、TDMA（時分割多重アクセス）が向いているが、TDMA はノード数の変動に弱くまだ無駄も多いため不採用とした。最大待ち時間を短縮するには LAN の伝送速度を上げれば良いが、その分消費電力が増加してしまう。ユビキタスな空間を実現するには膨大な数のノードが必要であるが、これでは省エネルギー化を図るという前提に反し

てしまう。このため、今回は伝送速度を必要最小限に抑えつつ最大待ち時間を保証するために、アクセス方式にはトークンパッシングバス方式を採用した。

トークンパッシングバス方式ではネットワーク中になんらかのパケットが常に流れており、それをネットワーク中の全てのノードが観測可能である。このことは、今回のように各ノードが自律的に周囲の状況を判断して処理する必要のあるネットワークに向いている方式でもある。

(2) 自律構築型 LAN

これまでのネットワークは大なり小なりそのネットワークに関する知識が必要であり、例えば新規にノードを追加する場合。ネットワーク担当者がそのネットワークの作法に則った手順を踏んで追加(接続)する必要があった。しかし、ユビキタス社会では、ネットワークは家庭内に引かれており管理者は居住人であるため、ネットワークの作法を意識させるわけにはいかない。すなわち、特別な知識や手順など不要で、ただコンセント（電源用とネットワーク用）に挿すだけでネットワークに接続できることが求められる(随意拡張性)。ここではこれを、自律構築型 LAN と呼ぶ。

自律構築を実現するために、本プロトコルは以下の機能を有している。

- ノード ID の自律生成機能
- プローブ機能
- ネットワークへの勧誘/応答/結果通知機能
- 自己紹介機能

以下、それぞれについて概要を説明する。

(2.1) ノード ID の自律生成機能

IP 網の場合 DHCP を用いてノード ID（IP アドレス）を動的に取得することが可能であるが、DHCP の場合サーバが必ず必要であるため、制約がある。そこで、サーバが無くてもネットワークを自律で構築できるよう、各ノードが既存ノードの ID を元に、自律的にノード ID を決定できる方式を考案した。

(2.2) プローブ機能

すでにネットワークが構築の場合はトークンなど何らかのパケットが流れているため、ある一定周期の間には必ず何らかのパケットを観測できるはずである。しかし、その周期の間に一度もパケットを観測できなかった場合は、トークン紛失やネットワーク切断など何らかの異常が発生していると考えられる。

このような場合は速やかにネットワークを回復させる必要があるため、ネットワークに対して強制的にパケットを送出して「誰かいませんか？」と探りを入れることにする。この機能がプローブ機能である。プローブパケットを受信できたノードは、応答パケットを流すことで互いを認識することが可能となり、速やかなネットワークの自律的回復を図ることが可能となる。

本機能は、ネットワーク構築の最初の段階でも機能する。

(2.3) ネットワークへの勧誘/応答/結果通知機能

トークンパッシング方式の場合、トークンを持っているノードしかパケットを送信する権利がない。このため、新規にノードを追加する場合には CSMA/CD 等とは違って何らかの処理が必要となる。トークンパッシング方式を採用している既存技術では追加されたノードがネットワーク上に強制的にジャミング信号を送出してネットワークを破壊し、その後に行われる再構築の過程でネットワークへの参加を果たすという方法をとっているが、これではその瞬間にネットワーク中を流れているパケットが破壊されてしまうし、再構築完了までの時間もかかるなどデメリットが多い。

本プロトコルではこれらのデメリットを回避するため、ネットワーク中にマスター機能を持つノードを決定し、そのノードが定期的に「ネットワークに参加したいノードはいないか？」という勧誘パケットをネットワーク中に送信する。追加されたノードは、この勧誘パケットを認識した後、自身のノード ID の申告値とともに応答を返す。応答を受け取ったマスターは、申告されたノード ID の重複検査を行った後、正式なノード ID を申告元へ通知する。

このように、ノード追加時にプロトコルレベルでシーケンス処理をいれることによって、自律的にネットワークを構築することができるようになる。

なお、マスター機能を持つノードは、現在ネットワーク中に存在しているノード ID を元に自動的に決定される。

(2.4) 自己紹介機能

自己紹介パケットは、ノードがネットワークに正常に接続された段階で送信する。送信先は、全ドメインへのブロードキャストとする。また、必要に応じて、一定周期で送信することもある。

自己紹介に必要な情報として、以下がある。

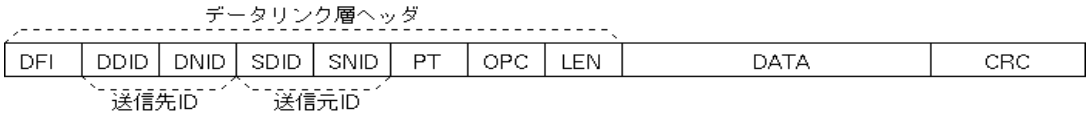
- 位置情報(座標、設置場所コード)
- 機能情報(機器コード)

（３） 論理 ID と物理 ID

一般的なネットワークプロトコルでは送信先の情報は論理的な値であり、実世界の情報とは無関係な場合がほとんどである。しかしユビキタス社会では、ユーザはノードの論理的な ID がどうであるかを気にすることなく、「この部屋の空調の温度を下げたい」とか「この部屋の照明を暗くしたい」というような要求をするのが自然である。

そこで、プロトコルが自律的に動作する部分は従来のような論理的で ID を扱い、ユーザからの指示がある場合や、ノード同士が互いの機能を協調して動作する場合には物理的な属性を表現した ID でも動作できるようにしておく。

本プロトコルで使用するパケットフォーマットを、以下に示す。



DFI　：送信先フィールド識別子
DDID：送信先ドメインID or 設置場所コード
DNID：送信先ノードID　or 機器コード
SDID：送信元ドメインID
SNID：送信元ノードID
PT　：パケットタイプ
OPC　：オペレーションコード
LEN　：DATA部の長さ
CRC　：DDIDからDATAまでのCRC

図　5.2-17　パケットフォーマット

そのための識別子がパケットの先頭に設けた DFI という識別子である。設置場所コードと機器コードを仕様として定義しておき、これらの値をヘッダに使用して送信先情報を「どこにある」+「どんな機器」という型式でも指定できるようにする。

DFI 値による送信先 ID の位置付けを以下に示す。

表 5.2-15　DFI 値の一覧

DFI 値	モード名	DDID の値	DNID の値	備考
0x00	通常モード	設置場所コード	機器コード	通常運用において、「あの部屋の照明を(窓を、空調を)操作したい」といった処理をプロトコルレベルで表現するために使用する。
0x01	網管理モード	送信先ドメイン	送信先ノード	

(4) 位置情報

本プロトコルにおいては、位置情報は座標と設置場所の 2 種類の表現方式にて管理する。

(4-1) 座標

3 次元座標を扱う。

ノードの位置情報は頻繁に変更されるものではないので、16bit にして余裕を持たせておく。分解能については、CAD とかとの兼ね合いもあるので、1cm とする。そうすると、最大値は約 655m となる。

座標原点は、xy 軸に関しては建築図面の平面図を、z 軸に関しては立面図を参考にする。

(4-2) 設置場所

位置情報の表現手段が上記の座標情報だけだと人間にはわけがわからないし、分散協調動作を行なう場合にも支障がでるので、設置場所の情報も別途定義する。この情報を設置場所コードと呼ぶ。基本的な考え方は、ECHONET を参考にした。

表 5.2-16 設置場所表現方法

項	設置場所	ユーザ定義	場所コード	場所番号

		b 7	b 6	b5	b 4	b 3	b 2	b 1	b 0
1	居間	0	0	0	0	1	000b～111b (000b は未設定扱い)		
2	食堂	0	0	0	1	0			
3	台所	0	0	0	1	1			
4	部屋	0	0	1	0	0			
5	浴室	0	0	1	0	1			
6	洗面所	0	0	1	1	0			
7	便所	0	0	1	1	1			
8	廊下	0	0	0	0	0			
9	階段	0	1	0	0	1			
10	玄関	0	1	0	1	0			
11	納戸	0	1	0	1	1			
12	倉庫	0	1	1	0	0			
13	庭	0	1	1	0	1			
14	車庫	0	1	1	1	0			
15	ベランダ	0	1	1	1	1			
16	自由定義	1	0000000b ～ 1111110b						
17	未設定	0	0	0	0	0	0	0	0
18	不定	1	1	1	1	1	1	1	1
19	予約 (未使用)	0x01 ～ 0x07							

以上より、8bit で設置場所を表現する。例えば、項 4 の「部屋」の場合は 0x21～0x27 までのコードを使用して 7 部屋までの識別を行なう。部屋の中での詳細な場所の把握については、座標情報を用いる。

（５） 拡張性

本プロトコルは住宅への適用を想定している。その場合のネットワーク構成を図 5.2-18 に示す。

図 5.2-18 のように、各部屋に対応するドメインと、それらを束ねた基幹ドメインで構成される。上記パケットフォーマットの通り、ドメイン内のノードは 8bit の空間内に収容され、ドメインも全体で

8bit の空間内に収容される。つまり、本プロトコルは 16bit でアドレッシングするが、本プロトコルをビルなどに適用する場合には 16bit では足りなくなる。そこで、拡張性に関しては以下のように対応した。

16bit で不足する可能性があるからアドレスの bit 数を増やすという方式は、単純ではあるが結局 bit 数が限界となってしまって拡張性を損ねるので、bit は現状の 16bit のまま拡張を行なえるようにする。

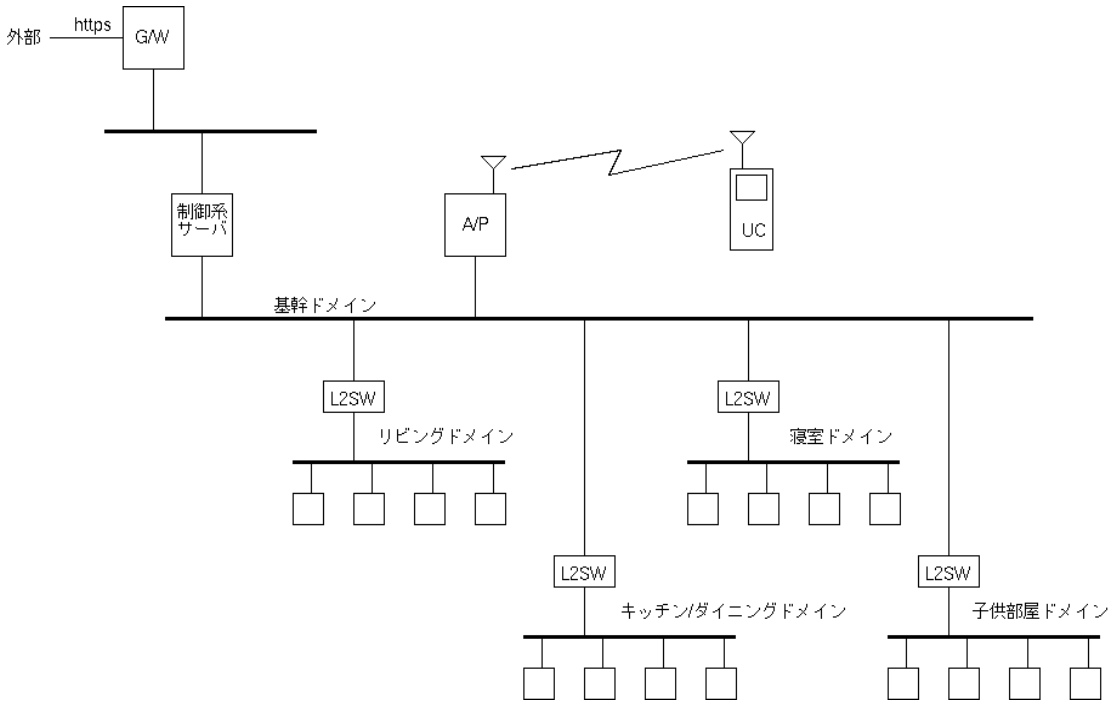
具体的には、IP におけるプライベートアドレスの概念を導入する。すなわち、基幹ドメインと各ローカルドメインを併せた空間を、一つのプライベート空間とする。

拡張する場合は、この制御系で代表されるプライベート空間を一つの仮想ノードと見做して、仮想ノードでネットワークを構築する。必要に応じてこのネットワークにルータを接続し、さらに大規模なネットワークへと拡張していく。

この「仮想ノード」の概念によって、論理的に無限台のノードを接続できるようになる。

図 5.2-18 ドメイン

（６） ネットワーク性能の予測



ユビキタスコミュニケータからコマンドを発行して、対応ノードが

反応するまでの時間を 100 ミリ秒前後になるような、伝送路速度の検討を行なった。

(6-1) 結論

消費電力および性能的な観点から、制御系ネットワークの伝送路の転送速度は、1～5Mbps の範囲とする。

(6-2) 検討結果

LAN コントローラ LSI の消費電力を抑えるには、以下の 2 通りの方法がある。

- 電源電圧を下げる
- 動作周波数を下げる

このうち、動作周波数については伝送路速度と比例関係にあり、ネットワークの性能に直接影響するので、以下の方針で検討を行なった。

制御系ネットワークを流れるトラフィックを、best/typical/worst の 3 種類想定し、伝送路の帯域を 1Mbps～10Mbps の範囲で振らせる

この時の検討結果を表および図に示す。

表 5.2-17 伝送路速度とトラフィックによる遅延

伝送路速度 (Mbps)	Best (msec)	Typical (msec)	Worst (msec)	評価
1.0	62.74	321.82	817.70	△
1.5	41.83	214.54	545.13	○
2.0	31.37	160.91	408.85	○
2.5	25.10	128.73	327.08	○
5.0	12.55	64.36	163.54	○
8.0	7.84	40.23	102.21	-
10.0	6.27	32.18	81.77	-

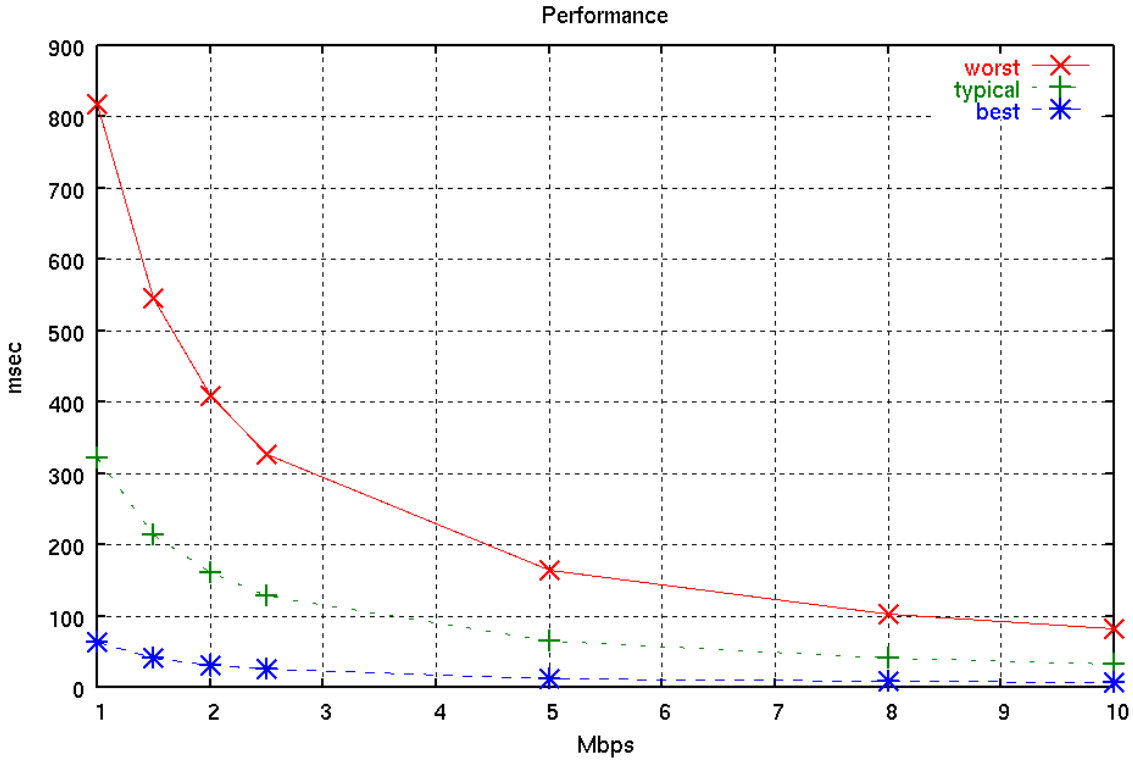


図 5-2.19　伝送路速度と遅延の関係

- best case　　：トークンのみが巡回しているケース
- typical case：各ノードが 3byte(データコード 1byte + 引数 1byte)のパケットを送信し、相手から応答が返って来るケース
- worst case　：各ノードが最大長(246byte)のパケットを送信し、相手から応答が返って来るケース

各シーンのまとめ

worst case は、最大構成(253 ノード)時に全てのノードが最大パケット長 (2+255+2)を送信して応答を受信するという流れになっており、現実的には起こり得ないケースであると言える。この worst case の値は、最悪値保証の目安として使用することができる。

typical case は各ノードがコマンドを投げてその応答を受信するというものであるが、これも最大構成時に全てのノードが同時にコマンドを投げることは確率的に低い。

best case はトークンだけが巡回しているケースであり、非常にありがちである。

以上より、実際の利用シーンでは、トラフィック量は best と typical の中間に多く分布することが予想される。

要求条件からみて

当初の目標性能として、UC から操作をしてノードが反応するまでの時間を 100～ 200msec 程度にしたいと考えていた。この値の根拠は、「多くの人にとって、100～ 200msec 程度で反応があればリアルタイムと言えそうだ」という、非常に感覚的なものだった。

UC からの操作の場合はルータを経由するため、2 倍まではいかないもののもう少し時間がかかることになる。しかし、クリティカルパスは UC からコマンドを発行して宛先ノードのソフトウェア処理が完了するまでであり、応答以降の時間は体感性能には影響しない。

以上の点から、伝送路速度は 1～5Mbps が妥当な値であることが判った。

（ウ）今後の方針

プロトコルの仕様はすでに固まっており、今後はその仕様を実証する段階に入る。具体的には、TCP/IP 上でアプリケーションとして本プロトコルをソフトウェア的に実装して機能評価を行う。機能評価完了後あるいは並行してハードウェア開発（FPGA）を行って性能評価も行う。

プロトコルの評価完了後、現在仕様検討作業を進めているノードの自律分散協調動作やパーソナライズのアプリケーションを作成する。最終的にはこれらのアプリケーションを今回開発したプロトコル上で動作させ評価する予定である。

（エ） まとめ

制御系向けネットワークプロトコルには、冒頭で述べたように ECHONET/LonWorks/ARCNET/EC-NET など既存技術が多数存在している。しかし、ユビキタス社会に適用するためには、リアルタイム性が欠如していたり、あるいは随意拡張性がないなどの問題点がある。そこで、随意拡張型固定網プロトコル研究の一環として、ユビキタスネットワーク用の制御系リアルタイムプロトコルを開発した。本プロトコルはトークンベースのリンク層プロトコルであり、リアルタイム性と随意拡張性を有する。机上検討により、要求される遅延時間と電力消費を考慮した伝送路速度は 1.5Mbps あたりが適切であることが判った。

5.2.2 実世界データ研究・Everything ID 研究

（ア）目的と概要

ユビキタスコンピューティングの基盤通信システムにおいて、あらゆる「モノ」を自動認識するための基盤技術を確立するため、それらの「モノ」に対して固有の ID を付与することが重要となる。

このユビキタス環境における ID の付与基準、体系を考案するにあたり、現在世の中に存在する各種コード体系の ID 付与基準、付与権限、管理機構などを調査し、それらの調査結果を ID 付与基準検討に資することとした。

また、これらの調査結果をもとに、ユビキタス環境におけるあらゆる「モノ」への ID の付与し、認識するための新たな方式として Everything ID を考案した。

（イ）既存 ID 調査

本研究を進めるにあたり、現在世の中に存在する各種コード体系の ID 付与基準、付与権限、管理機構などを調査し、ID に関する現状把握をすることとした。

（１）調査対象

調査対象コードは以下のとおりである。

表 5.2-18 調査対象コード

調査対象領域	調査対象コード
汎用	● ＥＡＮ－１２８ ● ａｕｔｏＩＤ（ｅＰＣ） ● ＱＲコード
流通	● ＪＡＮ ● ＩＴＦ ● ＮＷ－７ ● ＣＯＤＥ－３９ ● ＩＳＢＮ ● 共通取引先コード ● ＵＰＣ ● ＪＩＣＦＳ／ＩＦ－ＤＢ ● ＪＩＣＦＳ
ネットワーク	● ＩＰアドレス ● インターネット・ドメイン名
ディジタルコンテンツ	● ｃＩＤｆ（ｃｉｄ）
医療・福祉	● ＩＣＤ ９－ＣＭ（手術・処置） ● 病名交換用コード（病名コード） ● ＩＣＤ １０（病名） ● 臨床検査項目分類コード（ＪＬＡＣ１０） ● 医療用具の一般名称と分類（通称「厚生労働省赤本分類」） ● 薬価基準収載薬品コード（通称「厚生省コード」「薬価コード」） ● 個別医薬品コード ● レセプト電算処理システム基本マスター医薬品コード ● 薬事法承認番号 ● 標準検査項目マスター ● 標準手術処置マスター ● 標準病名マスター ● 医療材料マスター ● 標準医薬品マスター（通称「ＨＯＴ番号」）
金融	● ＪＩＳ－Ⅰ型（クレジットカード番号） ● ＪＩＳ－Ⅱ型（クレジットカード番号） ● ＪＩＣＳＡＰ仕様 ● ＥＭＶ仕様（ＥＭＶデータエレメント表） ● 全銀協ＩＣキャッシュカード標準仕様（全銀協データエレメント表）
食品	● 生鮮共通食品コード
公共	● 特許番号（登録番号） ● 自動車登録番号 ● 車体番号 ● 電話番号 ● 社会保険番号（基礎年金番号） ● 住民基本台帳番号（住民票）

（２）調査方法

具体的調査内容とその方法について、以下の通りとした。

- コード体系およびその他項目について公開情報をベースに調査を行う。
- 管理運用状況・システム化状況等の詳細についての実地訪問・ヒアリングは行わない。

表 5.2-19 調査方法

調査内容			調査方法		
項目		概要	公開情報を基に収集	メール・Tel による問合せ	訪問・ヒアリング
1	コード体系	コードの桁数、データ構造等に関する現状調査及び今後の動向調査を行う	○	△	×
2	利用状況	コードの利用方法、利用状況、適応業界等について、事例を交えて調査を行う	○	△	×
3	管理組織/体制	コード振出を担う管理運用組織の枠組み及び、実際の管理運用体制について調査を行う	○	△	×
4	運用状況/課題	管理運用組織におけるコード振出運用状況、運用上の課題等について調査を行う	●	×	×
5	システム化状況/課題	コード振出運用管理組織のシステム化状況、システム化課題、拡張性等について調査を行う	●	×	×

○：主な調査方法・情報ソースとする
●：情報ソースとするが、十分な情報を得るためには訪問・ヒアリングが必要と思われる項目
△：必要に応じて調査の方法・情報ソースとする
×：調査方法・情報ソースとして用いない

（３）調査結果

調査対象コードについての調査結果より、コードの性格、及びデータ長による分類を行った。

（ａ）コードの性格による５分類

調査の対象コードを、識別されるオブジェクトやコード体系の性格から以下の５つに分類した。

表 5.2-20 コードの分類

コード分類		特徴	記事
表現形式		・JIS や ISO 等の標準化団体で情報の媒体上の表現形式のみ規定 ・「識別子の体系」はアプリケーションの設計・運用者が独自に規定する（ex.NW-7 は宅急便・宅配便の配送伝票システム、書留郵便の管理用システム、各種会員カードシステム等に用いられているが、ID 体系はアプリ毎に異なる）	・実際の適用では個別アプリケーションが独自にコード体系を規定して利用している ・個別アプリケーションを調査する必要性がある
物理的 ID	複数対象なし	・物理的に識別する対象が存在し、対象を個別に識別する ID コード体系	－
	複数対象あり	・物理的に識別する対象が存在するが、対象の何らかの属性に着目し分類する分類コード体系	
論理的 ID	複数対象なし	・識別する対象は物理的な存在ではないが（概念等）、そのような論理的な対象を個別に識別する ID コード体系	対象そのものにはＩＣタグ等を付与することはできないが、容器等の物理的な対象にＩＤを付与することで様々なアプリケーションの展開が考えられる
	複数対象あり	・識別する対象は物理的な存在ではないが（概念等）、そのような論理的な対象の何らかの属性に着目し分類する分類コード体系	

データベースマスター	・コンピュータによる情報処理を容易にするために、データベース内で用いられることを目的として作成された識別子と内容を記したファイル	本来はコンピュータの内部処理用のマスターファイルであってユビキタスＩＤの対象外と考えられる
ファイル構造体	・ツリー状の複数のファイルからなるファイル構造体 ・ファイル構造やそこに格納されるべき情報が定義される	・８Kbit や３２Kbit の容量を持つファイル構造体なので、ユビキタスＩＤの対象外と考えられる ・カードの物理的ＩＤを付与することは考えられるが、採用はアプリケーション次第

（ｂ）対象コードの 5 分類による整理

表 5.2-21 コードの５分類による整理

コード分類		具体例	必要領域*	
			キャラクタベース (bytes)	バイナリベース (bits)
表現形式		ＥＡＮ－１２８	可変長	可変長
		ＱＲコード	4926 以下	34386 以下
		オートＩＤ(ePC)	12	96
		NW-7	可変長	可変長
		CODE-39	可変長	可変長
物理的 ID	複数対象なし	共通取引先コード	6	17
		cID f （cid）	可変長	可変長
		特許番号（登録番号）	10	34
		自動車登録番号	4 以下	14
		車体番号	各メーカーに依存（16 前後）	各メーカーに依存（128 前後）
		社会保険番号（基礎年金番号）	10	17
		住民基本台帳（住民票）コード	11	17
		JIS-Ⅰ型（クレジットカード）	60-80	480-640
		JIS-Ⅱ型（クレジットカード）	60-80	480-640
	複数対象あり	JAN	13	40

		ITF	14	44
		ISBN	10	31
		ＵＰＣ	12	38
		薬価基準収載薬品コード	12	43
		個別医薬品コード	12	43
		レセプト電算処理システム基本マスター　医薬品コード	9	32
		医療用具一般名称と分類	9	28
		薬事法承認番号	16	69
		生鮮共通商品コード	13	40
論理的ID	複数対象なし	ＩＰアドレス（v4/v6）	4/16	32/128
		インターネットドメイン名	255 以下	2,040 以下
		電話番号	9-11	32-37
	複数対象あり	ICD　9－CM(処置コード)	4	14
		病名交換用コード(病名コード)	4	32
		ICD　10(病名)	4	28
		臨床検査項目分類コード(JLAC10)	17	81
データベースマスター		ＪＩＣＦＳ／ＩＦ－ＤＢ	-	-
		ＪＩＣＦＳ	-	-
		標準検査項目マスター	-	-
		標準手術処置マスター	-	-
		標準病名マスター	-	-
		医療材料マスター	-	-
		標準医薬品マスター	-	-
ファイル構造体		JICSAP 仕様	200-16K	1,600-128K
		EMV 仕様	200-16K	1,600-128K
		全銀協 IC キャッシュカード仕様	200-16k	1,600-128K

*必要領域：コードの表現に必要なデータ領域を、コードの桁数・データ構造から算出した。

（ｃ）キャラクタベースマッピング表

対象コードをキャラクタベースでマッピングした場合の関係図を以下に示す。

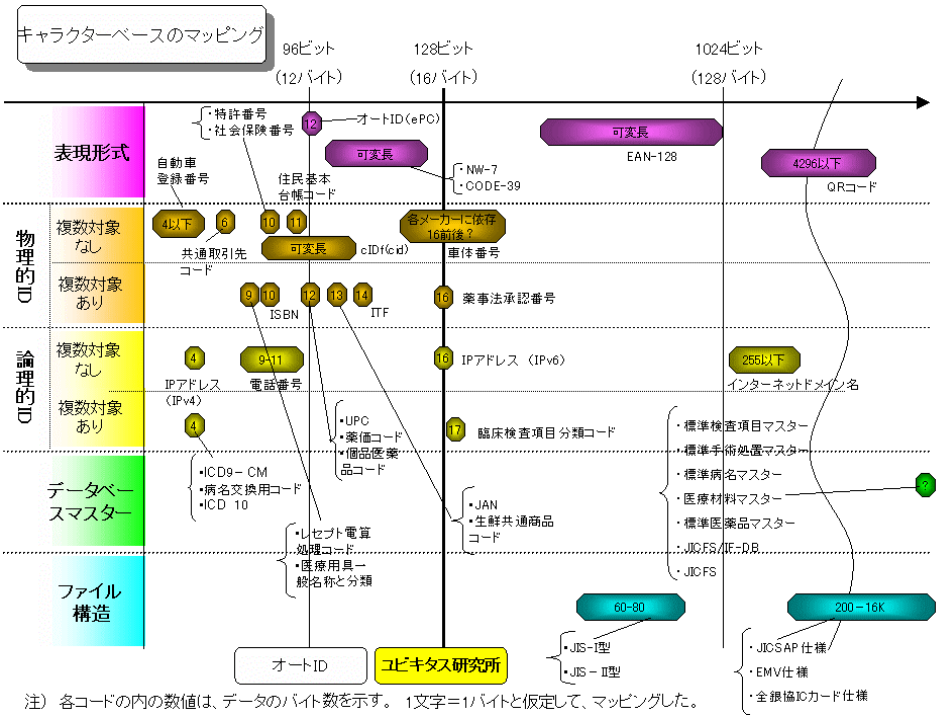
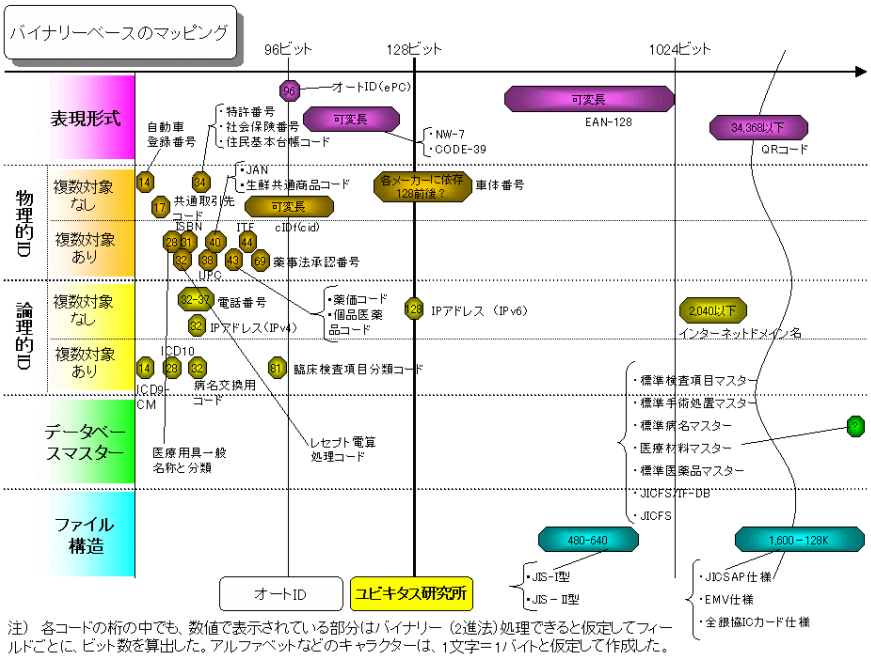


図 5.2-20 キャラクターベースマッピング図

（ｄ）バイナリベースマッピング表

対象コードをバイナリベースでマッピングした場合の関係図を以下に示す。



注）各コードの桁の中でも、数値で表示されている部分はバイナリー（2進法）処理できると仮定してフィールドごとに、ビット数を算出した。アルファベットなどのキャラクターは、1文字＝1バイトと仮定して作成した。

図 5.2-21 バイナリーベースマッピング図

（ｅ）各コードの調査結果詳細

各コードの調査結果詳細を以下に示す。

（ｅ）－１ 調査結果詳細提示対象

各コードの調査結果詳細については、全調査対象領域より特徴的な項目を選定し提示することとする。

具体的なコードは以下のとおりである。

表 5.2-22 調査結果詳細を提示するコード

調査対象領域	調査結果詳細を提示するコード
汎用	ＥＡＮ－１２８ ａｕｔｏＩＤ（ｅＰＣ）
流通	ＪＡＮ ＩＳＢＮ 共通取引先コード
ネットワーク	ＩＰアドレス
ディジタルコンテンツ	ｃＩＤｆ（ｃｉｄ）
医療・福祉	個別医薬品コード
金融	ＪＩＳ－Ｉ型（クレジットカード番号）
食品	生鮮共通食品コード
公共	電話番号 住民基本台帳番号（住民票）

（ｅ）－２ 凡例

調査結果詳細中の項目に関する凡例は以下のとおりである。

◆コード体系

◆◆コードタイプ

該当コードをクラスコード/個別コードのコードタイプに分類した。

表 5.2-23 コードタイプ

コードタイプ	定義	具体例
クラスコード	・コード振出機関は申請機関の識別を行うクラスコードを付与する。 ・クラス以下の各々の識別コードは申請機関が独自に自クラス内で重複がない様に定める。 ・クラスは複数の階層構造を持つこともある。	・JAN（メーカーコード） ・ISBN（出版者記号） ・IP アドレス ・電話番号
個別コード	コード振出機関がコードの最も細かい単位まで規定し、識別コードを付与する。	・共通取引先コード ・ｃID ｆ ・個別医薬品コード ・住民基本台帳コード

◆◆同一コードでの複数対象の有無

該当コード体系がオブジェクトのユニーク性（only one であるかどうか）を意味するか否かによって区別した。

表 5.2-24 同一コードでの複数対象の有無

複数対象	定義	具体例
有	・工業製品や本など同一コードを付与された複数の対象が存在する ・識別コードの意味するものは対象の何らかの属性や分類である。	・JAN ・ISBN ・個別医薬品コード
無	・識別コードの示す対象が世界中で唯一無二の存在であるもの。	・共通取引先コード ・IP アドレス ・ c IDf ・電話番号 ・住民基本台帳コード

◆◆桁数

該当コード体系の数字・英字を含めた桁数を記した。

◆◆データ構造

該当コード体系がどのようなデータ要素を含み､また個々のデータ要素が英字・数字を使ってどのように表されているかを記した。

【例１】

JAN メーカーコード 9 桁バージョン

a1a2a3a4a5a6a7a8a9 b1b2b3 c1

表 5.2-25 データ構造の記載方法 1

a1-a9	数字	JAN メーカーコード	・a,b,c 等→個々のデータ要素を示す。 ・a1a2…a_n→各データ要素の桁数を示す（原則的には 1 文字 1 バイトであるが、4 ビットや 1 ビットのものもある）。 ・数字→各データ要素がどんな表現方法でコーディングされているかを示す（数字、英字等）。
b1-b3	数字	商品アイテムコード	
c1	数字	チェックデジット	

【例２】

EAN-128 コード

a b c d1 d2 d3 d4 d5 d6 e f g h

表 5.2-26 データ構造の記載方法 2

データ構造の記載方法			
a	数字	識別子（2 桁から 4 桁）	a,b,c(アルファベット単独のもの)→各データ要素の必要桁数が可変長等で固定されていない場合
b	数字	メーカーコード、商品アイテムコード	
c	数字	識別子（2 桁から 4 桁）	
d1-d6	数字	品質保持期限日	
e	数字	識別子（2 桁から 4 桁）	
f	数字	バッチナンバー	
g	数字	識別子（2 桁から 4 桁）	
h	数字	連続番号	

◆◆コードの再利用性

個々のコードが一度利用されなくなった後にまったく同一のコードを他のものに付与するコードの再利用の有無を示している。

表 5.2-27 コードの再利用性

再利用性	定義	具体例
有	再利用される。	JAN、電話番号
無	再利用されない。	ISBN、住民基本台帳コード
対象外	表現形式やデータベースのため、利用性については判断できない。	QR コード、NW-7

◆利用状況

該当コードがどのような業種､業務アプリケーションで主に使用されているかを記した

◆管理／組織／体制

該当コードの振出機関を記した。

管理・運用体制は公知情報で明らかになる範囲で記載した。

◆解説

該当コードの概要や起源､最近の動向などコードの特徴の理解の参考になるような情報を記載した。

◆情報ソース

該当コードの調査にあたっての出典、参考資料、電話インタビュー先など情報ソースを記載した。

（ｅ）－３　調査結果詳細

（ｅ）－３－１　EAN-128 コード

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

有

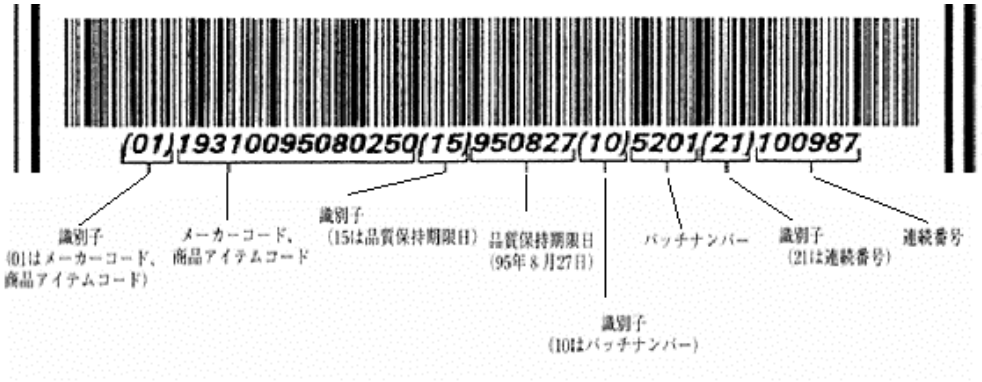
◆◆桁数

可変長

◆◆データ構造（例として下記に示す：他の体系もあり）

a b c d1 d2 d3 d4 d5 d6 e f g h

a	数字	識別子（2 桁から 4 桁）
b	数字	メーカーコード、商品アイテムコード
c	数字	識別子（2 桁から 4 桁）
d1-d6	数字	品質保持期限日
e	数字	識別子（2 桁から 4 桁）
f	数字	バッチナンバー
g	数字	識別子（2 桁から 4 桁）
h	数字	連続番号



◆◆コードの再利用性

有

◆利用状況

チェーンストアとサプライヤー間での ASN（Advanced Shipping Notice：事前出荷情報）/SCM（Shipping Carton Marking：混載商品用物流ラベル)物流管理システムやハム会社において下記のようなものに利用されている。

商品関連（製造年月日、品質保証期限、ロットナンバー等)

企業間取引データ（出荷先番号、出荷先企業コード、注文番号、梱包番号等)

◆管理／組織／体制

流通システム開発センター・流通コードセンター

◆解説

◆◆背景

バーコードの表示データ形式まで含めた標準化は共通商品コードの標準化として、UPC、EAN-13 など 25 年前から行われ POS（Point of Sales：販売時点情報管理)、EOS(Electronic Ordering System)、検品の業務で世界的に普及してきた。集合包装の商品コードのマーキングとして ITF が標準化されたのが 10 数年前で、これも物流管理、商品管理の分野で広く使われてきている。

この間、物流仕訳や商品管理、製造日や製造番号の管理などに個別企業のシステムとしていろいろなバーコードが使われてきたが、その標準化が遅れていた。国際 EAN 協会は、そのため標準バーコードとし

て、1992 年に EAN-128 を標準化し、その普及促進をはかってきた。日本でもコード 128 のバーコード・シンボルは 1996 年に JIS 化（JIS X-0504）されている。

1994 年に流通システム開発センターは商品関連情報、企業間取引情報などの業務システムを識別する「アプリケーション識別子とコード 128 利用マニュアル」を作成して、EAN-128 の利用研究にはいった。そして日本チェーン・ストア協会と共同で「EAN-128／連続梱包番号に基づく SCM ラベルによる新検品システム」を開発した。このシステムは（株）ダイエー、ジャスコ（株）、（株）伊勢丹を初め多くの企業で仕入れ先との間で使われている。

◆情報ソース

バーコード総合情報サイト

http://www.barcode.co.jp/

流通システム開発センター流通コードセンター　：　概説流通情報システム化 2002 年版　(2002)

（ｅ）－３－２ オート ID（ePC）

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

無

◆◆桁数

64-96 ビット

◆◆データ構造（例として 96 ビット TYPE1）

a1a2 b1b2b3b4b5b6b7 c1c2c3c4c5c6 d1d2d3d4d5d6d7d8d9

（1 文字は 4 ビットを意味する。）

a1-a2	数字、英字（8 ビット）	バージョン番号

b1-b7	数字、英字（28 ビット）	事業者コード
c1-c6	数字、英字（24 ビット）	商品アイテムコード
d1-d9	数字、英字（36 ビット）	シリアル番号

◆◆コードの再利用性

不明

◆利用状況

下記の利用が想定されている。

- ・流通経路の把握
- ・詳細な在庫管理

◆管理／組織／体制

オートＩＤセンター

◆解説

◆◆コードの特徴

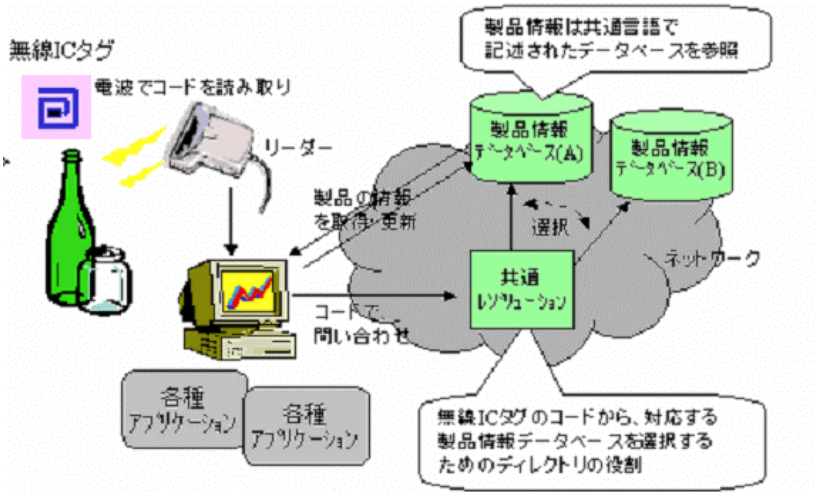
シリアル番号を含んだコードで、製品の種類だけでなく個別の物品の管理が可能である。

無線 IC タグにはコードのみを記録し製品の情報はネットワーク上のデータベースを参照するアーキテクチャにより、タグの超低コスト化と製品情報の柔軟な管理を実現できる。

無線 IC タグのコード体系、製品データベース記述言語、コードから製品データベースを参照する仕組み、の三つの標準化により、多くの事業者が共通に無線 IC タグを使用可能である。

◆◆規格化の組織

オート ID センターは、マサチューセッツ工科大学（MIT）に本部を置く国際的な非営利研究機関で、次世代のバーコードシステムを研究開発するという UCC （Uniform Code Council：バーコードの標準化団体）のビジョンの実現に向けて、1999 年に設立された。同セン



ターは、グローバルなサプライチェーン上での製品の識別とそのトラッキングを可能にする、国際的でオープンなネットワークシステムの開発に向けたインフラの構築と標準化を使命としている。

オートＩＤセンターは無線タグの情報をネットワークで利用する仕組み　「ePC(パレットやケース、または個々の商品に割り当てられる固有の ID)」(electronic product code),「Savant(ePC ネットワークを通じて情報を管理・運用するサーバー)」,「ONS(各 ePC とデータベースを関連付けるネットワークディレクトリ)」(object naming service) という仕組みを開発した。ePC は無線タグに搭載される 96 ビットの識別子である。ePC を収集した Savant は，ONS と呼ぶ DNS（domain name system）に似た機能を持つ ONS サーバーに，商品の販売元の IP アドレスを問い合わせる。Savant はそのアドレスに ePC を転送する。具体的には，無線タグを搭載する商品が，POS（販売時点情報管理）端末などに近付くと，読み取り機が無線タグの ID を自動的に読み取る。端末内のソフトウエア「Savant」が，インターネット上のサーバー「ONS」に，ID を基にどのメーカーの商品かを問い合わせる。こうしてメーカーの IP アドレスを知った「Savant」が，ID 情報をメーカーのサーバーに送信する。

◆◆今後の予定

ウォルマート社や米ホーム・デポ社などの大手小売企業は、オート ID センターの技術に多額の投資をし、サプライチェーンの効率を高めたり、倉庫から消費者の玄関先までの商品の流れを追跡しようとしている。

カミソリ・メーカーの米ジレットは，2003 年１月から米国でカミソリ製品５億個にオート ID 仕様の無線タグを搭載し，在庫管理など

で活用する実験を開始した。

日本でも大日本印刷が 2002 年秋に加工食品向けに実験を始めている。

凸版印刷も 2003 年内に同様の実験を始める。

◆情報ソース

オート ID センター

http://www.autoidcenter.org/index.asp

野沢哲生　：　あらゆるモノに極小無線タグを内蔵　ＩＤ仕様に日米 2 方式が名乗り，日経コミュニケーション(Feb. 2003)

（ｅ）－３－３ JAN コード（Japanese Article Number）

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

有

◆◆桁数

標準タイプ；13 桁

JAN メーカーコード 9 桁バージョン

JAN メーカーコード 7 桁バージョン

短縮タイプ；8 桁

◆◆データ構造

JAN メーカーコード 9 桁バージョン

a1a2a3a4a5a6a7a8a9 b1b2b3 c1

a1-a9	数字	JAN メーカーコード
b1-b3	数字	商品アイテムコード
c1	数字	チェックデジット

JAN メーカーコード 7 桁バージョン

a1a2a3a4a5a6a7 b1b2b3b4b5 c1

a1-a7	数字	JAN メーカーコード
b1-b5	数字	商品アイテムコード
c1	数字	チェックデジット

短縮タイプ 8 桁
a1a2a3a4a5a6 b1 c1

a1-a6	数字	JAN メーカーコード
b1	数字	商品アイテムコード
c1	数字	チェックデジット

◆◆コードの再利用性

有

◆利用状況

JAN コードは商品情報を表すコードとして幅広い業種・商品に使用され、以下のようなシステム内で使われている。

- POS システム
- 受発注システム
- 棚卸・在庫管理システム
- 公共料金等の支払いシステム

◆管理／組織／体制

流通システム開発センター・流通コードセンター

◆解説

◆◆JAN メーカーコード

ＪＡＮメーカーコードの最初の2桁はEANコードにおける国コード（日本は 45 と 49）である。2000 年 12 月までに登録した企業には 7 桁を付与していたが、2001 年 1 月以降、新規に登録した企業には原則的に 9 桁が付与されている。

◆◆コードの付与方法

JAN メーカーコードは流通システム開発センター・流通コードセンターが登録管理し、商品アイテムコードはメーカーが独自に決定する。

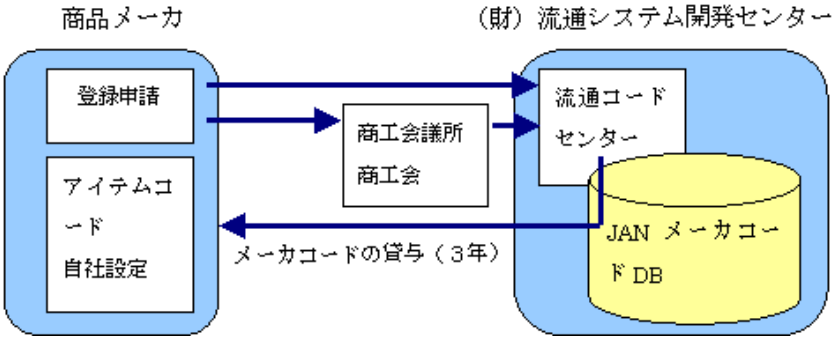
◆◆コードの特徴

JAN コードは、国際的には EAN コード(European Article Number)と呼称され、アメリカ、カナダにおける UPC(Universal Product Code)と互換性のある国際的な共通商品コードである。

◆◆規格化の組織

流通システム開発センター・流通コードセンター

ユーザーからの申請によって（財）流通システム開発センター・流通コードセンターが「JAN メーカコード」をユーザーに付番貸与する。ユーザは重複の無いよう商品アイテムコードを自社で設定し、商品バーコード等に利用する。各ユーザーが商品情報を登録する JICFS/IF-DB もある。



◆情報ソース

流通システム開発センター

http://www.dsri-dcc.jp/company/jan/

流通システム開発センター流通コードセンター　：　概説流通情報システム化 2002 年版　(2002)

（ｅ）－３－４　ISBN コード（International standard book numbering）

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

有

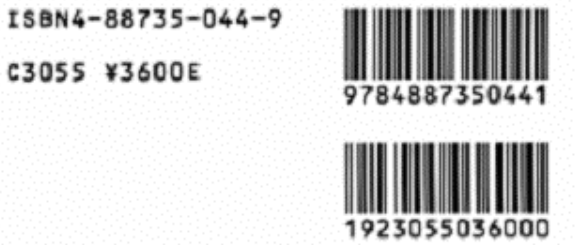
◆◆桁数

10 桁

◆◆データ構造

ISBN a1 b c d1

a1	数字	グループ記号
B（b+c=8 桁）	数字	出版者記号又は製作者記号
C（b+c=8 桁）	数字	刊行物記号
d1	数字	チェックデジット



◆◆コードの再利用性

無

◆利用状況

下記のようなものに利用されている。

- 図書及びその他の単行刊行物（印刷された図書・パンフレットなど）
- 複合媒体刊行物、その他複合でない同様の媒体の刊行物（点字刊行物・地図・電子刊行物など）

◆管理／組織／体制

日本図書コード管理センター

◆解説

◆◆背景

ISBN は、1969 年にイギリスの標準図書番号を元に、ISO 2108-1978 として制定され、1988 年には日本工業規格、JIS X 0305-1988 「国際標準図書番号(ISBN)」となった。日本では、ISBN の基本部分に加えて、「読者対象」「発行形態」「内容」を示す分類コードと本体価格を加えたコード体系を開発し「日本図書コード」として使用している。

◆◆コードの付与方法

グループ記号と出版者記号又は製作者記号は日本図書コード管理センターが登録管理し、刊行物記号は出版者又は製作者が独自に決定する。

グループ記号は、国、地域、言語又はその他の便宜的なグループを指す。国際 ISBN 機関が割り当てる。グループ記号の長さは、そのグループの出版点数によって異なる。

出版者記号又は製作者記号は、日本図書コード管理センターが割り当てる。出版者記号又は製作者記号の長さは、その出版者又は製作者の出版点数によって異なる。（例：岩波書店は 00、ピアソンエデュケーションは 89471）

刊行物記号は、出版者又は製作者が付与する。刊行物記号の長さは、その前に置かれたグループ記号及び出版者記号又は製作者記号の長さによって定まる。

チェックデジットは、重みとして 10 から 2 までを用いるモジュラス 11 によって算出する。なお、チェックデジットが 10 となった場合は 10 の代わりに X とする。

◆◆2005 年に 10 桁から 13 桁へ改訂

ISBN の改訂が、国際標準化機構第 46 専門委員会の第 9 分科会によって進められている。2003 年 1 月 30 日と 31 日に行われた作業ミーティングでは、ISBN の構成数字を現状の 10 桁から 13 桁へ拡張することなどが確認された。このミーティングを受けた委員会原案が、

2003 年 2 月 28 日に公表された。30 年以上に渡って図書の識別のために用いられてきた ISBN だが、近年、電子出版などによって出版点数が急増していることから、登録のための番号数の不足が指摘されていた。改訂された ISBN は，2005 年 1 月に ISO 2108 第 4 版として発行され、2 年間の移行期間の後に、13 桁の ISBN へと完全移行される予定である。

◆情報ソース

日本工業標準調査会
http://www.jisc.go.jp/
日本図書コード管理センター
http://www.isbn-center.jp/index.html

（ｅ）－３－５ 共通取引先コード

◆コード体系

◆◆コードタイプ

個別コード

◆◆同一コードでの複数対象の有無

無

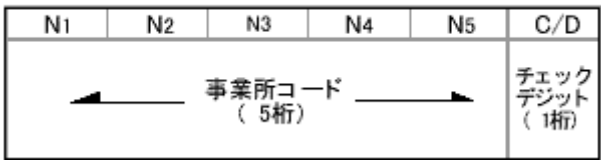
◆◆桁数

6 桁

◆◆データ構造

a1a2a3a4a5 b1

a1-a5	数字	事業所コード
b1	数字	チェックデジット



◆◆コードの再利用性

無

◆利用状況

百貨店やチェーンストアなどの小売業が、商品を卸売業やメーカーに発注するときにその発注先事業所を表すコードを統一したもので、下記の中で利用されている。

・企業間の受発注や納品
・代金決済処理における伝票上
・EOS や EDI におけるデータ

◆管理／組織／体制

流通システム開発センター・流通コードセンター

◆解説

◆◆背景

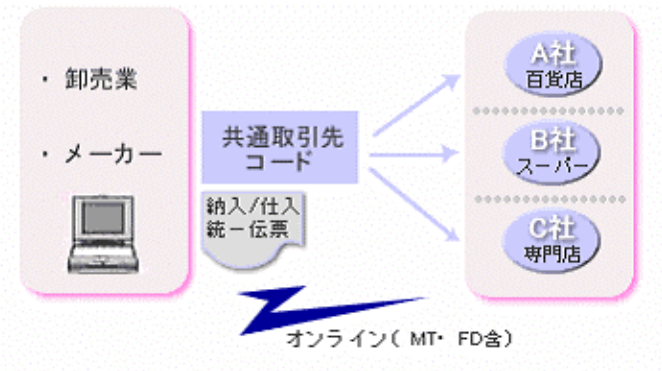
共通取引先コードは、1977 年に経済産業省が制定し、2003 年 3 月末現在、共通取引先コードの登録事業所数は、約 57，000 件となっている。

◆◆登録の対象となる事業所

共通取引先コードを利用している百貨店、チェーンストア、ボランタリーチェーン、ホームセンター、生協、専門店、共同仕入機構、卸売業等に商品やサービスを納入する卸売業または製造業

Ｊ手順（流通業の標準通信）でオンラインデータ交換を行っている企業

小売業、物流業、サービス業



◆◆付番方法

共通取引先コードは、流通コードセンターが登録申請のあった事業所に付番貸与する。一度、付番貸与された共通取引先コードは、原則として変わらない。

◆◆登録にかかる費用と日数

登録申請料として 1 コードにつき 5,250 円がかかる。(事前納付制)共通取引先コードの有効期間は 3 年間で、継続して登録する場合は、3 年ごとに更新料 5，250 円がかかる。番号通知を受けるまでの日数は、流通コードセンターへ登録申請書を提出されてから、3 日から 1 週間程度である。

◆情報ソース

流通システム開発センター

http://www.dsri-dcc.jp/company/01/kyoutsu_c.htm

流通システム開発センター流通コードセンター　：　概説流通情報システム化 2002 年版　(2002)

（ｅ）－３－６　ＩＰアドレス

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

無

◆◆桁数

IPv4 アドレスは 32 ビット
IPv6 アドレスは 128 ビット

◆◆データ構造

- IPv4 アドレス
  32 ビット（32 桁）の 2 進数
  （例：11000000101010000000000000001010）

8 ビット毎に区切り、それぞれ 10 進数で表記する。
(例：11000000.10101000.00000000.00001010)
192.168.0.10

◆◆コードの再利用性

有

◆利用状況

- IPv4 アドレス
  割り当てホスト数　179348　；　返却ホスト数　93052　(2002 年 12 月現在)

◆管理／組織／体制

国内の管理組織:社団法人日本ネットワークインフォメーションセンター

国際的な管理組織：ICANN (The Internet Corporation for Assigned Names and Numbers)

◆解説

◆◆概要

IP アドレスは TCP/IP ネットワーク上で、通信相手(ホスト)を識別するための番号である。現在、最も一般的に使用されているのが IPv4 アドレス(v4 はバージョン 4)。
IPv4 アドレスは 32 ビットのアドレスで、通常「123.45.67.89」のように、8 ビットごとに区切った 4 つの数字により表記する。また、IPv6（バージョン 6）は IPv4 に代わるプロトコルとして、IETF(Internet Engineering Task Force)の IPNG ワーキンググループで準備が進められてきたプロトコルである。IPv6 アドレスは 128 ビットのアドレスである。アドレス数はおよそ 10 の 38 乗という天文学的な数字になる。IPv6 では、パケットそのものを暗号化してセキュリティを強化する「IPsec」と呼ばれる機能や、ネットワークの自動設定機能が付加されている。

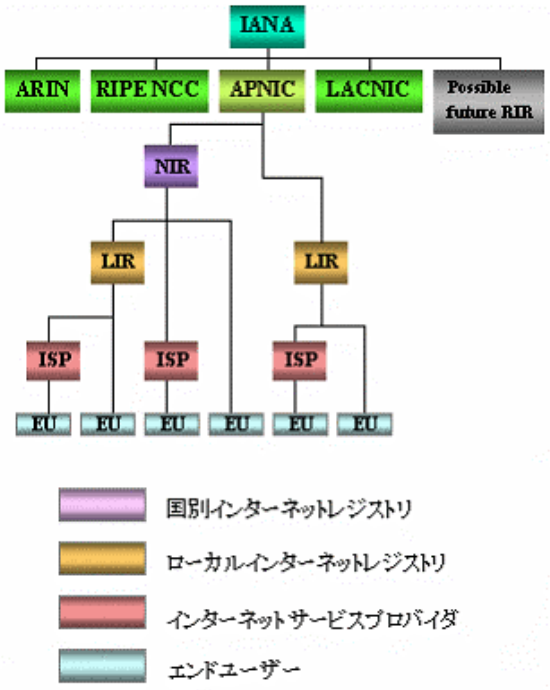
◆◆国際的な IP アドレス管理

IP アドレスは、現在 ICANN の一機能である IANA(Internet Assigned Numbers Authority)を頂点とする階層構造による割り振り・割り当てが行われている。IANA からは各 RIR（地域インターネ

ットレジストリ）へアドレスブロックが割り振られ、RIR は、割り振りを受けたアドレスブロックをさらに分割して LIR（ローカルインターネットレジストリ）に割り振られる。

◆◆国内における IP アドレス管理

（社）日本ネットワークインフォメーションセンター（JPNIC）配下の LIR は IP アドレス管理指定事業者として JPNIC から IP アドレス管理業務の委託を受ける。 IP 管理指定事業者は JPNIC からアドレスブロックの割り振りを受けエンドユーザや自組織のネットワークに対して IP アドレスの割り当てを行う。



◆情報ソース

（社）日本ネットワークインフォメーションセンター
http://www.nic.ad.jp/ja/index.html

（ｅ）－３－７ cIDf (cid)

◆コード体系

◆◆コードタイプ

個別コード

◆◆同一コードでの複数対象の有無

無

◆◆桁数

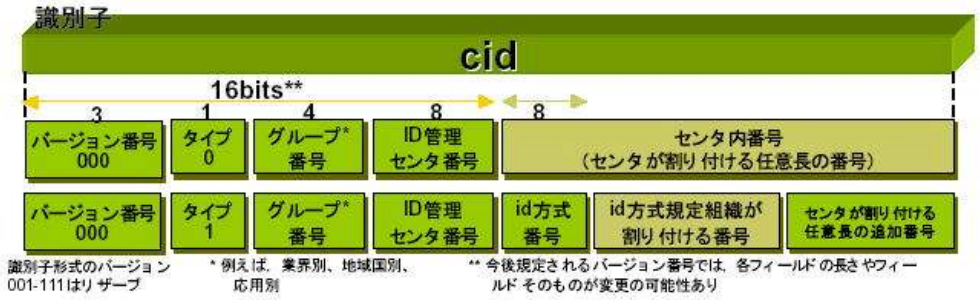
可変長

◆◆データ構造

a1a2a3 b1 c1c2c3c4 d1d2d3d4d5d6d7d8 e

（1 文字は 1 ビットを意味する。）

a1-a3	数字（3 ビット）	バージョン番号
b1	数字（1 ビット）	タイプ
c1-c4	数字、英字（4 ビット）	グループ番号
d1-d8	数字、英字（8 ビット）	ID 管理センタ番号
e	数字、英字（任意長）	センタ内番号



◆◆コードの再利用性

不明

◆利用状況

下記のデジタルコンテンツへの利用が想定されている。

・書籍

・映画

◆管理／組織／体制
コンテンツ ID フォーラム

◆解説
◆◆背景

コンテンツ ID フォーラム(cIDF)は各ディジタルコンテンツ毎にユニークなコード（ "コンテンツ ID"）を付与することにより、著作権を保護しながらのディジタルコンテンツのネットワーク流通を促進する枠組みの策定を目的とし，1999 年 8 月 4 日に、東京大学安田浩教授の提唱により発足。

◆◆コードの特徴

コンテンツ ID は、世界に唯一の RA（レジストレーション・オーソリティ）監督のもと、各コンテンツホルダやコンテンツ発行者自身、または第三者がそれぞれ運営するコンテンツ ID 管理センタが振り出す。

コンテンツごとに､ヘッダや電子透かしによりユニークなコンテンツ ID が埋め込まれる。また、属性情報の一部についても、DCD(流通記述子)として、コンテンツへ埋め込まれる。

フルセットのコンテンツ属性情報は、コンテンツ ID をキーに各コンテンツ ID 管理センタが運営するデータベース（IPR-DB）で管理。

◆◆規格化の組織

コンテンツ ID の発行と権利に関連する処理を表す処理フローは下記の通りである。

著作者や著作権利用者は事前に電子認証機関へユーザ登録する。

著作権者はコンテンツ ID 発行センタに権利情報を申請、コンテンツ ID の発行を要求する。

コンテンツ ID 発行センタは著作権者を認証する。

コンテンツ ID 発行センタはコンテンツ ID を発行し、権利情報を設定する。

コンテンツ ID 発行センタは IPR-DB センタへ権利情報を登録する。

コンテンツ ID 発行センタは発行したコンテンツ ID を著作権者に通知する。

著作権者と著作権利用者との間で著作権の利用許諾契約を交わす。

著作権利用者はコンテンツ ID 発行センタへ流通情報を申請、コンテンツ ID の発行を要求する。

コンテンツ ID 発行センタは著作権利用者を認証する。

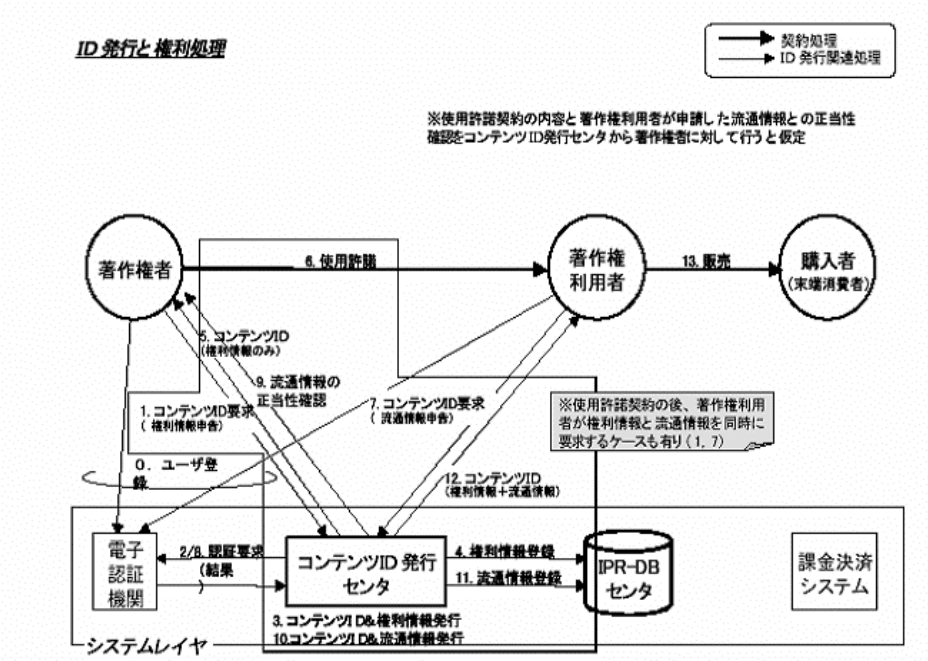
コンテンツ ID 発行センタは申請された流通情報が利用許諾契約に反しないか、著作権者へ確認する。

コンテンツ ID 発行センタはコンテンツ ID を発行し、流通情報を設定する。

コンテンツ ID 発行センタは IPR-DB センタへ流通情報を登録する。

コンテンツ ID 発行センタは発行したコンテンツ ID を著作権利用者に通知する。

著作権利用者は購入者へコンテンツを販売する。



◆◆コンテンツ ID の提供する機能

コピーライト機能

だれでもコンテンツの権利関係や作成年月日などの属性情報を知ることができるようになる。

コンテンツ属性検索機能

コンテンツを同一基準で検索したり、取り寄せたりすることができる。

バーコード機能

業者や業界にまたがるコンテンツの流通履歴、販売履歴などの収集が効率的に行える。

不正利用検出機能

コンテンツ ID をキーに、ネットワーク上の不正利用コンテンツを検

出できる。また、コンテンツの利用条件を参照し、利用条件遵守の仕組みを構築することもできる。

正当性検証機能

該当コンテンツが正式な著者から発行され、その後不正改竄が行われていないことの証明など、コンテンツ正当性をチェックすることができる。

編集履歴参照機能

IPR データベースに蓄積してあるコンテンツの再編集の履歴を見ることができる。

データベース共通キー機能

デジタルアーカイブ構築の際の共通識別コードとなり、検索、相互参照が容易となる。

◆情報ソース

コンテンツ ID フォーラム

http://www.cidf.org/japanese/index.html

（ｅ）－３－８ 個別医薬品コード

◆コード体系

◆◆コードタイプ

個別コード

◆◆同一コードでの複数対象の有無

有

◆◆桁数

12 桁

◆◆データ構造

a1a2a3a4 b1b2b3 c1 d1 e1e2 f1

A1-a4	数字	薬効分類
B1-b3	数字	投与経路及び成分
C1	アルファベット	剤形

D1	数字	同一分類内別規格薬効分類
E1-e2	数字	同一規格単位内の銘柄番号（個別の商品銘柄と対応する番号）
F1	数字	チェックデジット

◆◆コードの再利用性

不明

◆利用状況

病院が薬品管理する際に、薬価情報を迅速に取り込むためのキーコードとして広く利用されている。

◆管理／組織／体制

株式会社　医薬情報研究所

◆解説

◆◆特徴

基本的構成は、「薬価基準収載医薬品コード」と同じである。より細かく、銘柄（商品）別に対応している点が違う。具体的には、「同一規格単位内の銘柄番号」（以下、「銘柄番号」）が違う。
「薬価コード」の場合、厚生労働省が 2 年に 1 回告示する公報を元にしているが、その公報では、商品名称だけではなく一般名称（統一名収載品）も「銘柄番号」にしているため、個別の商品が特定できないデメリットがある。
たとえば、塩酸プロプラノロールという薬品に関しては、住友製薬が発売した「インデラル錠１０ｍｇ」は、その商品名で登録されているのに対して、その他の製品（東和薬品の「ソラシロール錠」など）は、「塩酸プロプラノロール」という「統一名収載品」としてまとめて登録されている。
全て商品名で登録されている点が、「個別医薬品コード」の特徴である。

◆情報ソース

佐々木哲明　：　用語コードの標準化
http://www1.fukui-med.ac.jp/KMI/doc23/doc23d.html

医療情報システム開発センターの HP
http://www.medis.or.jp/standard/index2.html
浜松医科大学の HP
http://www.mi.hama-med.ac.jp/jami-inventory/yakuzai.html

（ｅ）－３－９ JIS- 型（クレジットカード）

◆コード体系

◆◆コードタイプ
クラスコード
◆◆同一コードでの複数対象の有無
無

◆◆桁数
カード内に記録することのできる文字数は最大 72 文字（桁）
クレジットカードの会員番号の桁数(データ構造のｂの領域)：19 桁（最大）

◆◆データ構造
a1b c d e f1g1

A1	英字、数字（７ビット符号）	開始符号	
B	英字、数字（７ビット符号）	広義の会員番号*：最大 19 桁	データ部分：最大 69 文字
C	英字、数字（７ビット符号）	セパレータ	
D	英字、数字（７ビット符号）	有効期限	
E	英字、数字（７ビット符号）	自由領域	
F1	英字、数字（７ビット符号）	終了符号	
G1	英字、数字（７ビット符号）	LRC(水平冗長検査文字)	

*広義の会員番号（19 桁）には、個人（口座）を識別する番号だけでなく、発行会社を識別する部分（通常は上３桁ないしは６桁）を含む

◆◆コードの再利用性

不明

◆利用状況

クレジットカード（磁気ストライプ）

◆管理／組織／体制

（財）流通システム開発センター

◆解説

◆◆背景

ＪＩＳ規格に定められたカード規格は以下の通り。
①ＪＩＳ Ｘ６３０１：識別カード ――― 物理的特性
②ＪＩＳ Ｘ６３０２：識別カード ――― 記録技術
③ＪＩＳ Ｃ６２２０：情報交換用符号

ＪＩＳ Ｘ６３０２では、国際用としてのＪＩＳ－Ⅰ型、国内用としてＪＩＳ－Ⅱ型の記録様式が定められている。ただ、当初制定に当たっては、全銀協における標準化の検討が、ＩＳＯの検討・制定作業よりかなり先行していたことから、ＪＩＳ－Ⅱ型が日本における標準規格としてされた。この為、日本国内の各金融機関が設置したＣＤ・ＡＴＭは、国際間の相互互換性はなく、海外で発行された各種金融決済カードの取扱ができず、問題となっていた。しかし、1999 年７月、ＪＩＳ規格の改訂がなされ、国際用としてのＪＩＳ－Ⅰ型が正規の規定となり、従来のＪＩＳ－Ⅱ型（以下『旧ＪＩＳ－Ⅱ型』という）は付属定義書扱いとなっている。

◆◆コードの特徴

クレジットカードでは、第２トラック部分（ＡＢＡ規格：ＡＢＡ＝American Bankers Association ＝アメリカ銀行協会）を使用し、フォーマットはＩＳＯ規格による。（ＩＳＯ２８９４／３１６６／３５５４／４２１７）

◆◆データ構造

「会員番号」が、発行会社識別の為に用いられる唯一の判定キーとなる。通常は上３桁ないしは６桁であるが、最大 12 桁の番号帯にて判定される。「会員番号」のコードの一部として、ＩＳＯ規格で定め

られた「業態コード」(「会員番号」の頭 4 桁)、「企業コード」が含まれる。

◆◆利用状況

具体的には以下のケースで利用。

ISO 規格で定められた業態コード

ISO規格業態分類	主な利用会社(国際ブランド等)
0	For assignment by ISO/TC68 and for other future industry assignments
1	Airlines
2	Airlines and other future industry assignments
3	Travel and entertainment:DINERS、AMEX、JCB
4	Banking/Financial:VISA
5	Banking/Financial:Master
6	Merchandising and Financial:DISCOVER(SEARS)
7	Petroleum
8	Telecommunications and other future industry assignment
9	For assignment by national standards bodies

クレジットカード上にクレジット企業コードが電子的に記録(エンコード)され、クレジットカード発行企業を認識。

POS、CAT(クレジット信用照会端末)などの、クレジット共同利用端末における端末識別番号として用いられ,クレジットカードがどの端末で使用されているかを認識。

クレジットカード決済用ネットワーク(CAFIS など)の回線認識や、カード会社への接続会社コード。

◆情報ソース

ECOM の HP　(電子決済グループ)

http://www.ecom.jp/report/wg2-3/h10denshi-c-1.html

日本規格協会　:　ＪＩＳハンドブック 65　情報技術Ⅱ(2003)

（ｅ）－３－１０ 生鮮共通商品コード

◆コード体系

◆◆コードタイプ

個別コード

◆◆同一コードでの複数対象の有無

有

◆◆桁数

13 桁

◆◆データ構造

青果

a1a2a3a4 b1b2b3b4 c d e f

a1-a4	数字	生鮮フラグ
b1-b4	数字	標準品名コード
C	数字	栽培方法
D	数字	サイズ
E	数字	量目/入り目数
F	数字	チェックデジット

花き・食肉

a1a2a3a4 b1b2b3b4 c d

a1-a4	数字	生鮮フラグ
b1-b4	数字	標準品目コード
C	数字	ゼロ固定
D	数字	チェックデジット

水産物

a1a2a3a4 b1b2b3b4 c d1d2 e

A1-a4	数字	生鮮フラグ
B1-b4	数字	標準品目コード
C	数字	態様
D	数字	形状・部位
E	数字	チェックデジット

品目		コード体系				
青果		4922	□××××	P	S　V	C/D
			標準品名コード	栽培方法	サイズ　量目/入り数	
花き		4922	□×××××		00	C/D
			標準品目コード		ゼロ固定	
食肉		4922	□×××××		00	C/D
			標準品目コード		ゼロ固定	
水産物	生鮮品	4922	□××××	T	F_1F_2	C/D
			標準品目コード	態様	形状･部位	
	加工用	4922	□××××	T	P_1P_2	C/D
			標準品目コード	態様	形状･部位	

(注 1)「4922」は生鮮 4 品共通のプリフィックス。C/D はチェックデジット
(注 2)標準品名コードの先頭1桁(□部分)は 4 品の識別コード。
花き:1　野菜:3　果実:4　水産物:6　精肉:7　部分肉:8

◆◆コードの再利用性

無

◆利用状況

- 鮮品の取引に伴う情報交換（受発注等）
  果物の産地でのソースマーキング

◆管理／組織／体制

（財）食品流通構造改善促進機構

◆解説

◆◆背景

農林水産省が 1997 年度から 5 か年計画で実施した「食品流通情報化基盤開発事業」の一環として、(財)食品流通構造改善促進機構からの委託を受けて、(財）流通システム開発センターが、生鮮 4 品（青果物、水産物、食肉、花き）の標準商品コードと EDI 標準メッセージの

開発を行った。

ちなみに、小売業における既存の POS システムや EOS において、産地でのソースマーキングが多い青果物を小売店の販売レベルで単品指定ができる。この体系を他の 3 品の共通商品コードと区別するために「生鮮ＪＡＮコード」と呼んでいる。生鮮 JAN コードが出荷者でソースマーキングされることによって、小売業の POS システムや EOS で利用され、出荷者～卸売市場～小売業間の商流と物流の効率化に活用されることが期待されている。

また、食肉、花き、水産物については、物流梱包の識別を行う標準物流バーコードを、国際標準の UCC/EAN-128 体系に準拠して制定された。

◆情報ソース

（財）食品流通構造改善促進機構の HP
http://www.ofsi.or.jp/task_edi/index.html

（財）流通システム開発センターの HP
http://www.dsri-dcc.jp/company/03/seisen.htm

（ｅ）－３－１１ 電話番号

◆コード体系

◆◆コードタイプ

クラスコード

◆◆同一コードでの複数対象の有無

無

◆◆桁数

(1) 固定電話

9～10 桁

(2) 携帯電話・PHS

11 桁

(3) IP 電話

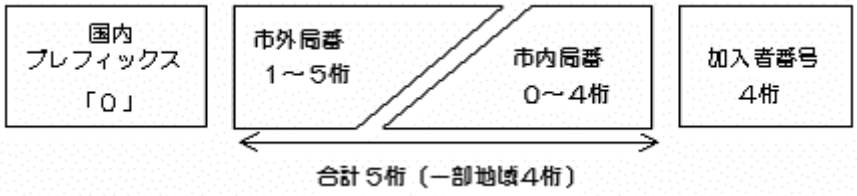
11 桁

◆◆データ構造

(1)固定電話

a1 b c d1d2d3d4

a1	数字	国内プレフィックスの 0
b（b+c=4 又は 5 桁）	数字	市外局番（地域ごとに総務省告示で規定）
c（b+c=4 又は 5 桁）	数字	市内局番（総務省が電気通信事業者ごとに指定）



d1-d4	数字	加入者番号（電気通信事業者が独自に決定）

(2)携帯電話

a1a2a3 b1b2b3 c1c2c3c4c5

a1-a3	数字	080 又は 090
b1-b3(b1=0 を除く)	数字	総務省が事業者(NTT ドコモやジェイフォン等）ごとに指定
c1-c5	数字	事業者が独自に決定

(3)PHS

a1a2a3 b1b2b3 c1c2c3c4c5

A1-a3	数字	070
B1-b3	数字	総務省が事業者(NTT ドコモや DDI ポケット等）ごとに指定
C1-c5	数字	事業者が独自に決定

(4) IP 電話

a1a2a3 b1b2b3b4 c1c2c3c4c5

a1-a3	数字	050
b1-b4(b1=0 を除く)	数字	総務省が第一種・第二種電気通信事業者の区別なく指定
c1-c4	数字	事業者が独自に決定

◆◆コードの再利用性

有

◆利用状況

家庭や企業などにおいて利用される。

◆管理／組織／体制

総務省

◆解説

◆◆固定電話

初めの「0」は、国内プレフィックスと呼んでおり、国内通話を示す「合図」である。これに続く市外局番は 1～5 桁であり、この部分については地域ごとに総務省告示で規定している。また、市内局番は 0～4 桁であり、この部分については総務省が電気通信事業者ごとに指定を行うこととしている。

◆情報ソース

総務省

http://www.soumu.go.jp/joho_tsusin/top/tel_number/index.html

総務省総合通信基盤局電気通信事業部電気通信技術システム課

http://www.soumu.go.jp/s-news/2002/020513_1.html

（ｅ）－３－１２ 住民基本台帳（住民票）コード

◆コード体系

◆◆コードタイプ
個別コード

◆◆同一コードでの複数対象の有無
無

◆◆桁数
11 桁

◆◆データ構造
a1a2a3a4a5a6a7a8a9a10 b1

a1-a10	数字	住民票コード
b1	数字	チェックデジット

◆◆コードの再利用性
無

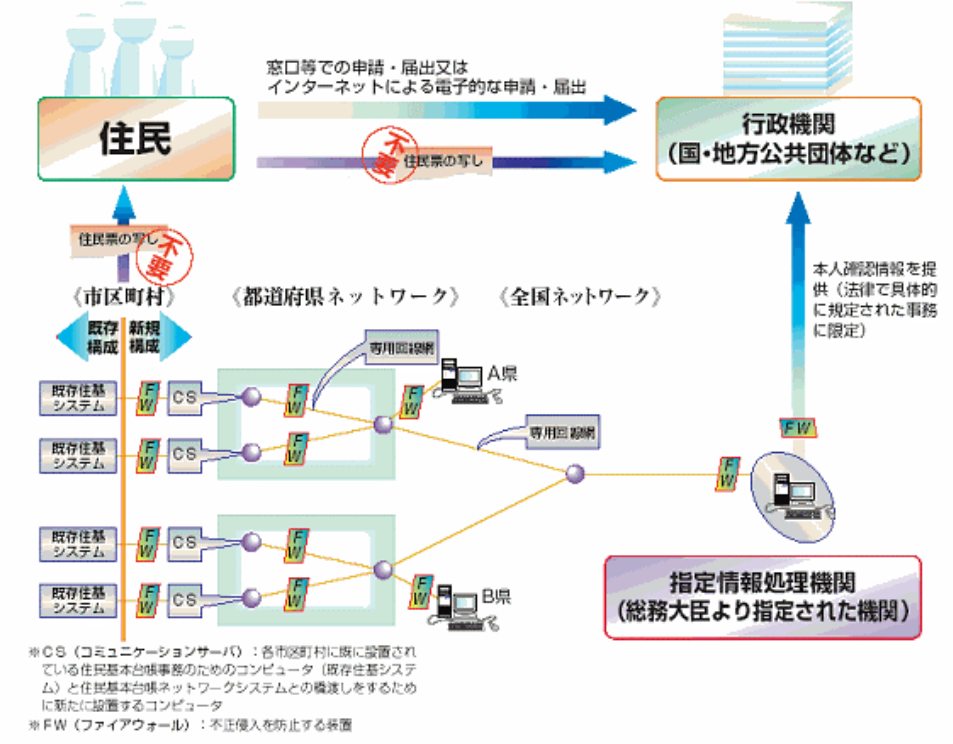
◆利用状況
行政機関へ申請・届出を行う際、住民票の写しの添付が省略できる。
「電子政府・電子自治体」の基盤となる。

◆管理／組織／体制
財団法人地方自治情報センター

◆解説
◆◆住民基本台帳ネットワークシステム

住民基本台帳ネットワークシステムは、４情報（氏名・生年月日・性別・住所）、住民票コードとこれらの変更情報により全国共通の本人確認を可能とする地方公共団体共同のシステムで、電子政府・電子自治体を実現するための基盤となるものである。
今後、行政機関（国・地方公共団体など）では各種の届出・申請などの際に、提出していた住民票の写しの代わりに、ネットワークシステ

ムから氏名、生年月日、性別、住所などの本人確認情報の提供を受けることが可能となる。



◆◆付与方法

財団法人地方自治情報センターがランダムな 10 桁とチェックデジット 1 桁の数字を生成する。

財団法人地方自治情報センターが各市町村の住民の数に合わせて番号を割り振る。

各市町村が割り振られた番号を 1 人 1 人の住民に付与する。

この番号はコンピュータを使って一人ずつ無作為に付けられるので、家族であっても全く関連のない番号が付けられている。

住民票コードは全国で重複して付けられることのない番号であるため、確実な本人確認が可能となる。

住民票コードは住所や氏名等に変更があっても変わることはないが、本人からの申し出により、理由を問わずいつでも変更することができる。ただし、番号を指定することはできない。

変更を希望する場合には、市役所市民課又は支所で変更申請が必要となる。変更申請の際には本人であることを確認する書類（運転免許証、パスポートなど）を提示する必要がある。

◆◆住民基本台帳ネットワークシステムのセキュリティ

地方公共団体共同のシステムであって、国が一元的に管理するシステムではないこと

都道府県や全国センターに保有される情報は、本人確認のための４情報（氏名・住所・性別・生年月日）、住民票コードとこれらの変更情報のみであること

行政機関へのデータ提供は、住民の居住関係の確認のための求めがあったときに限定し、個別の目的ごとに法律上の根拠が必要であり、かつ、目的外利用を禁止すること

上記の３つの条件によりさまざまな個人情報を一元的に収集･管理することを「法律上」認めない仕組みとなっている。

◆情報ソース

総務省

http://www.soumu.go.jp/c-gyousei/daityo/index.html

財団法人地方自治情報センター

http://www.lasdec.nippon-net.ne.jp/

総務省自治行政局市町村課に電話ヒアリング

（ウ）Everything ID の考案

前項までの調査結果をもとに、ユビキタス環境であらゆるものに付与する Everything ID を考案し、付与基準等や課題に関する考察を行った。

（１）Everything ID の目的

ユビキタス環境とは、実世界のあらゆる事象をコンピュータ上で処理し、デジタル通信で情報交換することと位置付けることができ、その際、実世界のあらゆる情報が、コンピュータの上で標準的な形で表現されることが不可欠である。

本考察では、このユビキタス環境におけるデータインフラの一環として、あらゆるものを認識するための基礎となる Everthing ID の最適な付与基準を考案することを目的とする。

（２）Everything ID 要件の整理

既存 ID 調査にて判明した各既存 ID の特徴等を考慮し、ユビキタス環境における Everything ID 実現に向けた要件の整理を行った。

（ａ）データ長

既存 ID 調査における各既存 ID のデータ長のマッピングより、実世界における ID 付与として概ね 128bit までのデータ長をとる表現形式が主流と考えられる。

これにならい、Everything ID においてもデータ長を 128bit とすることが最適であると考えられる。

なお、128bit 長に収まらない ID の主流化など、将来の拡張性も考慮する必要がある。

（ｂ）既存ＩＤコードの吸収

実世界の各種 ID は、既にそれぞれの分野で広く活用され、全く新規の ID 体系を持ち込むことは、利便性の向上が見込めるとしても、移行・設備投資・操作習熟等の大きな壁が存在し、普及の妨げとなることが考えられる。

Everything ID ではこの点を考慮し、既存の各種 ID を吸収し表現することが可能なメタコードの概念を採用することが重要である。

（ｃ）センタ割り当てを受けないローカル性

ID の付与を受ける際に、付与センターによる中央集権的なユニー

ク ID 付与は運用面での困難性をはらんでいる。

例えば位置情報と時間情報を組み合わせた時空間アドレスをもってユニークネスを確保することにより ID として活用できるとともに、センターからの割り当てを受けずに利用できるコードタイプも考えられる。

Everything ID ではこのようなタイプの IDについてもサポートし、ID 付与における運用性を確保する必要がある。

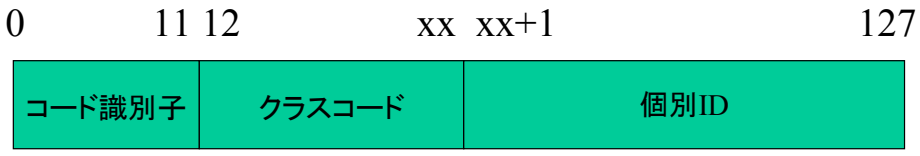
（３）Everything ID

128bit のコード長をもち、既存 ID を包含し表現することができるメタ IDとして、以下のような ID付与基準をもつ構造を Everything ID として考案した。

（ａ）Bit アサインメント（クラスコードの場合）

Everything ID の Bit アサインメントを以下に示す。

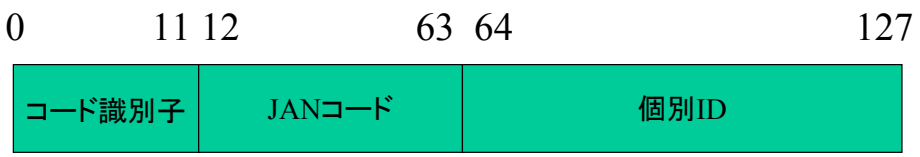
図 5.2-22 Bit アサインメント



00～11（12bit）・・・コード識別子
12～xx（可変）・・・クラスコード
xx+1～127（可変）・・・個別 ID

（例）JAN コードの場合

図 5.2-23 JAN コードの Bit アサインメント例



00～11（12bit）・・・コード識別子（JAN コードの識別子を設定）
12～63（52bit）・・・JAN コード領域（JAN コード自体が入る）
64～127（64bit）・・・個別 ID（製品種類の中の個別 ID）

このように、コード識別子を活用することにより、既存のコードを Everything ID に取り込み、互換性を確保することができる。

また、従来クラスコードとして製品種類までしか表現できなかったコードについて、同一種類内での製品個別 ID（シリアル番号等）を設定できるようになり、製品個体毎の管理など、きめ細かなサービスの実現が期待できる。

なお、クラスコードが存在しない個別コードについては、12bit 以降全てを個別 ID として活用することが可能である。

同様に、時空間アドレスによるセンタ割り当てを受けない ID についても、コード識別子＋個別 ID にて表現する。

（ｂ）拡張性

128bit 長の ID 構造で表現できない長大な ID については、さらに 128bit 単位で拡張可能な構造を提供する。

（４）今後の課題

（ａ）ID 解決手段と通信基盤の整備

Everything ID は、既存 ID を取り込んで表現可能な構造をもつが、その ID が何を(どの既存 ID を)指し示すのかを解決する手段として、Everything ID 解決サーバが必要となる。

常に最新かつ正確な情報をリアルタイムに提供する解決手段を用意することが望ましいが、そのためには、全ての既存 ID を一括集中管理する方式は困難である。

従って、解決手段を提供する Everything ID 解決サーバは、コード識別子によって、既存 ID の解決先を特定し、解決先サーバ等と協調しながら ID 解決する必要がある。

このような機能をもつ Everything ID 解決サーバを柱とした通信基盤の整備がユビキタス環境における重要なテーマとなるだろう。

（ｂ）普及推進に向けた取り組み

既存 ID 取り込みにあたっては、該当する業界団体等への協調を積極的に働きかけるための、運営組織の整備が不可欠となるだろう。

5.3 ユーザノードシステム技術の研究開発

5.3.1 カームネットワーキング／カームコンピューティング

ハードウェアおよび OS や BIOS レベルの基盤ソフトウェアにおける、新しい電源制御機構を含む省電力機構のためのプロトコルと制御方式を開発した。

（ア）はじめに

コンピュータが身の回りのいたる所に組み込まれるユビキタス環境では、システム全体の電力消費量の増加が予想される。そのようなマクロ的な電力消費量の増加を抑えて環境への負荷を低減させるには、機器自体がエネルギーを消費しないような省電力設計を実現することはもちろん、機器の電力消費量の監視やそれらの省電力機能をいかに緻密に制御するかということも重要である。また、ユビキタス環境では前述の通り膨大な数のコンピュータが偏在することになる。それらの機器を如何に効率的に管理するかも重要とある。これらに対する 1 つの解として、ネットワークを用いることが考えられる。ここでは、ネットワークを用いた機器の電力消費量の監視や機器の省電力機能の制御をネットワーク電源管理と呼ぶことにする。本研究の目的はユビキタス社会において、省エネルギー化に対するネットワーク電源管理の有効性を検証するものである。

来たるユビキタス社会において、ネットワーク電源管理を実現する為に以下の検討を行った。

18. 電源に関して機器が持つべき項目
19. ネットワーキング方法

1.についてはネットワークを用いた電源管理ということに主眼をおき、電源に関して監視するパラメータや制御すべき項目などを規定した電源プロファイルの策定を行った。また 2.については特殊なものではなく標準化されており、且つ機器の監視や制御に柔軟に対応可能な通信プロトコルということに主眼をおき、IP 網におけるネットワーク管理用のプロトコルである SNMP をネットワーキングの 1 方法と考えた。また、本研究ではユビキタス環境においてネットワーク電源管理の対象となる機器を、組み込み機器のプラットフォームを担う T-Engine と捉え、T-Engine をベースとして研究を行った。それに伴い、SNMP の実装対象は設備機器が応用対象である μT-Engine とした。

以上のことから、本ドキュメントでは電源管理プロファイル及び μT-Engine 用 SNMP の基本仕様及び実装確認の結果を示す。そして最後に、本研究の課題及び今後の方針について記述する。

（イ）ネットワーク電源管理プロファイル

表 5.3-1 に、電源に関してネットワーク経由で管理すべき項目を列挙する。全ての機器がこれらの項目全てを実装するわけではなく、機器シリーズ(標準/μ/ナノ/ピコ)などに応じてサブセットを規定してそのサブセットを実装することとする。

表 5.3-1 ネットワーク電源管理パラメーター一覧

管理対象	情報種別	説明	SYSTAX	R/W
システム	製造者	メーカ名	string(0..31)	read-only
	機器名	製品名 or 型番	string(0..63)	read-only
	機器種別	ルータ, PC, PDA, ケータイ, センサ …	string(0..31)	read-only
	設置名	ホスト名(識別名)	string(0..31)	read-write
	設置場所	mobile ｜ static(階-部屋-天井｜壁｜床｜ドア｜…)	string(0..63)	read-write
	管理者	name tel e-mail address URL	string(0..63) string(0..63) string(0..63) string(0..63)	read-write
	機器版数	software hardware	string(0..31) string(0..31)	read-only
	経過時間	uptime	integer	read-only
	製造年月	YYYY-MM	string(0..31)	read-only
	製造番号	シリアル番号	string(0..63)	read-only
	保守情報	マニュアルへの URL や サポートセンター電話番号等	string(0..63)	read-only

電源	搭載電源種別	機器が保有する電源の種別
		1: その他
		2: 外部電源
		3: バッテリ
		4: インライン給電
		5: 自家発電
		6: 無線送電
		
		複数設定可。インライン給電は LAN 経由の給電、自家発電は T-Engine 独力での発電を想定。無線送電はマイクロ波送電や RFID などの電磁誘導発電を想定。
			integer{
			other(1),
			external(2),
			battery(3),
			inline(4),
			private(5),
			wirelessPowerSupply(6)
			}
				read-only

	動作電源種別	現在電力を供給している電源種別
		1: 不明
		2: その他
		3: 外部電源
		4: バッテリ
		5: インライン給電
		6: 自家発電
		7: 無線送電
		
		インラインン給電はLAN経由の給電、自家発電はT-Engine独力での発電を想定。無線送電はマイクロ波送電や RFID などの電磁誘導発電を想定。
			integer{
			unknown(1),
			other(2),
			external(3),
			battery(4),
			inline(5),
			private(6),
			wirelessPowerSupply(7)
			}
				read-only

	バッテリユニット搭載数	バッテリユニットの搭載数	integer	read-only

	バッテリユニット接続状態※	バッテリがある場合のユニット接続状態
		1: 不明
		2: 未接続
		3: 接続
			integer{
			unknown(1),
			disconnected(2),
			connected(3)
			}
				read-only

	バッテリ残り時間※	現在の負荷状態において予測されるバッテリの利用可能時間	integer	read-only
	バッテリ容量閾値	「バッテリ状態」の「3:バッテリ容量低下」を宣言する為のバッテリ使用可能時間	integer	read-write

	バッテリ状態	バッテリの状態
		1: 不明
		2: バッテリ正常
		3: バッテリ容量低下
		
		バッテリ正常とは、「バッテリ容量閾値」よりも長い「バッテリ使用可能時間」がある状態をさす。バッテリ容量低下とは、「バッテリ容量閾値」よりも短い「バッテリ使用可能時間」になっている状態をさす。
			integer{
			unknown(1),
			normal(2),
			low(3)
			}
				read-only

	ネットワーク WakeUp 対応	ネットワーク WakeUp 機能の有無
		1: 不明
		2: なし
		3: あり
		
		ネットワーク WakeUp とはネットワーク経由での電源 ON を意味する。
			integer{
			unknown(1),
			none(2),
			supported(3)
			}
				read-only

	ネットワーク ShutDown 対応	ネットワーク ShutDown 機能の有無
		1: 不明
		2: なし
		3: あり
		
		ネットワーク ShutDown とは CPU 暴走時など異常時にネットワーク経由での電源 OFF を意味する。
			integer{
			unknown(1),
			none(2),
			supported(3)
			}
				read-only

	消費電力状態	消費電力状態
		1. その他
		2. 通常
		3. 省電力
		4. スタンバイ
		
		通常とは通常動作していることをさす。
		省電力とはプロセッサのクロックスピードを落とした状態で動作していることをさす。
		スタンバイとはプロセッサのクロックスピードを最低レベルに落とし、ディスプレイへの出力を停止した状態をさす。
			Integer{
			other(1),
			normal(2),
			powersave(3),
			standby(4)
			}
				Read-only

	電源制御対応	電源制御の有効化
		1: 有効
		2: 無効
		
		電源制御対応を有効にすることで、下段に記述する 5 つの電源制御が可能となる。
			integer{
			enabled(1),
			disabled(2)
			}
				read-write

	IDLE 時低消費電力モード以降制御	IDLE 時低消費電力モード移行許可/禁止
		
		1: 許可
		2: 禁止
			integer{
			permit(1),
			inhibit(2)
			}
				read-write

	サスペンド移行制御	自動サスペンド許可/禁止の参照及び設定
		1: 許可
		2: 禁止
			integer{
			permit(1),
			inhibit(2)
			}
				read-write

	サスペンド移行時間	サスペンドに移行するまでの時間の参照及び設定	integer	read-write
	ディスプレイ制御	ディスプレイが存在する場合にディスプレイ OFF までの時間の参照及び設定	integer	read-write

	電源操作	電源を操作する
		1: 電源断
		2: 再起動
			integer{
			halt(1),
			reboot(2)
			}
				read-write

※　バッテリ搭載数に応じてそれぞれの項目が設定される。

本研究において電源管理プロファイルの実装は、SNMP における拡張 MIB の概念を導入し MIB として実現することにした。

（ウ）SNMP over T-Engine

(1) はじめに

(1-1) 適用

以下は、三菱電機製μT-Engine「M3T-M32104UT」をターゲットとした IP ネットワークにおけるネットワーク管理プロトコルである SNMP に関する仕様である。

(1-2) 概要

SNMP over T-Engine の主な特徴について以下に示す。

SNMPv2 プロトコルに準拠(一部未サポート)

- MIB の自由な拡張
- 1 タスクで動作

(2) システム定義

(2-1)用語定義

図 5.3-1 に SNMP の基本アーキテクチャを示す。 SNMP を使用するシステムは以下で構成される。

- マネージャ
- エージェント
- MIB
- SNMP プロトコル

マネージャとは、管理アプリケーションからの要求に従って snmpget や snmpset などのメッセージをエージェントに送り、その結果をエージェントから受け取るコンポーネントである。つまり、T-Engine に対して要求する側の機能である。

SNMP エージェントとは、管理される機器に実装され、マネージャからの要求に従い、それに対して応答したり機器の制御を行うコンポーネントである。

MIB とは、端的に言うとインデックスの集合である。マネージャがエージェントに対してオブジェクト ID と呼ばれるインデックスを指定して要求を出すと、エージェントは指定されたインデックスに対応する処理を行う。

SNMP プロトコルとは、マネージャ - エージェント間で要求・応答をやり取りする通信プロトコルである。 SNMP プロトコルはアプリケーション層のプロトコルであり、UDP/IP を下位レイヤとしている。

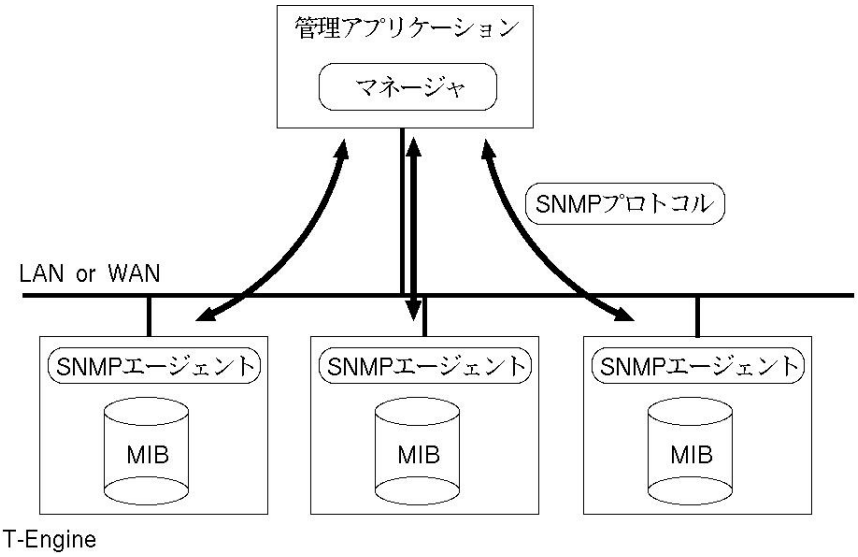


図5.3-1　SNMPの基本アーキテクチャ

また、SNMP エージェントは RFC2275 で規定される VACM をサポートしている。その為、コミュニティ名やアクセス制限などの設定情報をファイルとして保持している。コミュニティ名とは、マネージャがエージェントに要求を出す場合に使用するパスワードと言える。SNMP エージェントは起動時にそのファイルを読み込み、マネージャからの要求に対してコミュニティ名に基づいたセキュリティの確保やアクセス制限を行っている。このファイルをコンフィグレーションファイルと呼ぶことにする。

(2-2) システム構成

SNMP over T-Engine のシステム構成上の特徴を以下に示す。

- ユーザタスクとしてシステムに組み込まれ動作する
- SNMP タスクから他タスクの起動は行わず常に 1 タスクとして動作する

また、三菱電機製μT-Engine「M3T-M32104UT」の機器固有の特徴として以下も挙げられる。

- IP アドレスやネットマスクなどのネットワーク設定情報をファイルとして持つ

図 5.3-2 にシステム構成を示す。

SNMP over T-Engine の実体は、T-Engine 上の SNMP プロトコルを解釈する SNMP エージェントと MIB である。

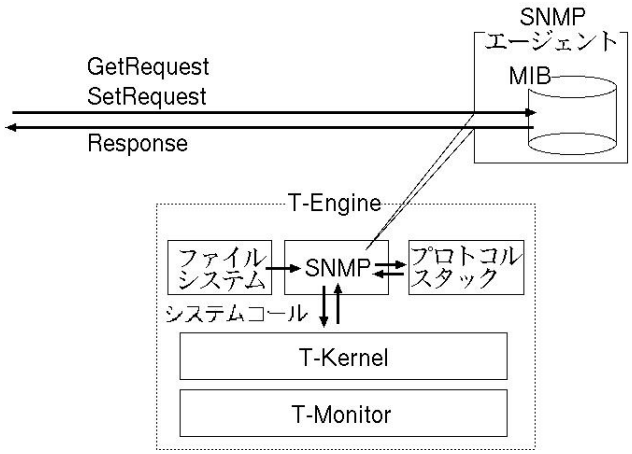


図5.3-2　システム構成

(2-3) システム用件

SNMP over T-Engine に必要なシステム上の条件について以下に示す。

- TCP/IP プロトコルスタック
- ファイルシステム

TCP/IP プロトコルスタックについては、SNMP プロトコルの下位レイヤが UDP/IP であるので、それらをサポートしている必要がある。また、ソケット関数として BSD ソケット API であることが望ましい。

ファイルシステムについては、起動時にコンフィグレーションファイルを読み込むのに必要となる。関数としては、UNIX 系の標準関数である fgets()と同等の機能を有しているか、もしくは fgets()をシミュレートできる API が提供されている必要がある。また、三菱電機製μT-Engine の特徴として、ネットワーク設定情報を取得する場合にもファイルシステムが使用される。

(3) 提供機能

SNMP over T-Engine は SNMPv2 プロトコルに準拠しているが、一部未サポート機能が存在する。本章では、SNMP over T-Engine が提供する機能、また未サポートとする機能について示す。

(3-1) サポート機能

(a) SNMP メッセージ

- GetRequest
- GetNextRequest
- GetBulkRequest
- SetRequest
- Response

(b) VACM

SNMP over T-Engine では、VACM に関する各設定情報(コミュニティ名、グループ名、アクセス制限、ビュー定義) をコンフィグレーションファイルとして保持している。

図 5.3-3 にコンフィグレーションファイルの設定例を示す。

```
#       # is a comment out.

#            sec.name        source          community
com2sec    unl_manager    192.168.1.2/32  YRPubinjp0
com2sec    unl_mynet      192.168.1.0/24  YRPubinjp1
com2sec    unl_o_net      default         public

#                          sec.model     sec.name
group      rwAccess        v1            unl_manager
group      rwAccess        v2c           unl_manager
group      roAccess1       v1            unl_mynet
group      roAccess1       v2c           unl_mynet
group      roAccess2       v1            public
group      roAccess2       v2c           public

#              incl/excl    subtree                  mask
view     all   included     .1                       80
view     part  included     .1.3.6.1.4.1.16500.11.1  ff.80

#         content          sec.model  sec.level  match  read   write  notif
access    rwAccess    ""     any        noauth     exact  all    all    none
access    roAccess1   ""     any        noauth     exact  all    none   none
access    roAccess2   ""     any        noauth     exact  part   none   none
```

図 5.3-3　コンフィグレーションファイルの設定例

com2sec

このディレクティブは source と community の組でセキュリティ名(sec.name)を指定するのに用いる。 source のパラメータとしてはサブネットまたは default が指定できる。サブネットは IP/MASK で指定する。

なお、図 3.1 ではコミュニティ名として public を指定しているが、実運用においては推奨されない。

group

このディレクティブはセキュリティモデル(sec.model)とセキュリティ名(sec.name)の組でグループ名を指定するのに用いる。セキュリティモデル(sec.model)は v1,v2c を指定する。

view

このディレクティブは指定した名前のビューを定義するのに用いる。ビューのタイプは included/excluded のいずれかである。 mask は 16 進数で表した 8 ビットを.又は:を区切ってリストにしたものである。mask が指定されていない場合のデフォルトは ff である。

access

このディレクティブはグループとそのセキュリティのレベルをビューにマッピングするのに用いる。""は空文字列を指定する。セキュリティモデル(sec.model)は v1,v2c, any が指定可能である。セキュリティレベル(sec.level)は noauth を指定する。read, write, notif はビューに対応するアクセス方法(view ディレクティブで定義したビュー又は none)を指定する。

(3-2) 未サポート機能

- Trap
- Notification

エージェントが自発的にマネージャにイベントを通知する場合のメッセージである Trap 及び Notification については現状未サポートとする。

(4) 動作環境

表 5.3-2 及び表 5.3-3 に三菱電機製μT-Engine「M3T-M32104UT」の動作環境について示す。

表 5.3-2 ハードウェア仕様

No.	項目	内容
1	CPU	M32104 ・入力クロック：27MHz ・ 動作クロック(最大)：216MHz ・ バスクロック(最大):54MHz
2	メモリ	以下のメモリを搭載 ・ フラッシュメモリ(ブート ROM 用)：4M バイト ・ SDRAM：16M バイト
3	LAN	・ SMSC 社製 LAN コントローラ(LAN91C111)

表 5.3-3　ソフトウェア仕様

No.	項目	内容
1	OS	リアルタイム OS ・M3T-MR32R
2	ネットワーク	TCP/IP プロトコルスタック ・M2S-AE32R04
3	ファイルシステム	DOS ファイルシステム ・M3S-AE32R00

(4-1) メモリ配置

図 5.3-4 に三菱電機製μT-Engine「M3T-M32104UT」のメモリ配置を示す。 SNMP を含むプログラムは起動時にフラッシュ ROM から SDRAM 上にコピーされ、 SDRAM 上で実行される。

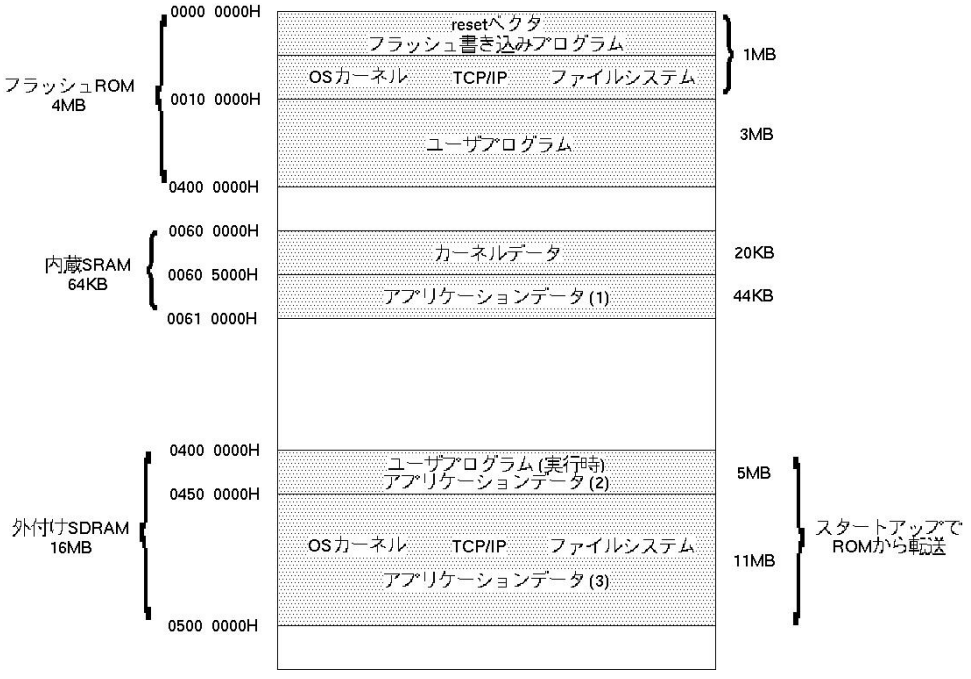


図 5.3-4　メモリ配置

(4-2) SNMP の起動シーケンス

図 5.3-5 に SNMP の起動シーケンスを示す。

電源を投入すると、初期化プログラムが起動されハードウェア及び OS の初期化が行われる。SNMP エージェントタスクは初期設定用タスクが実行する configuration 関数により起動される。

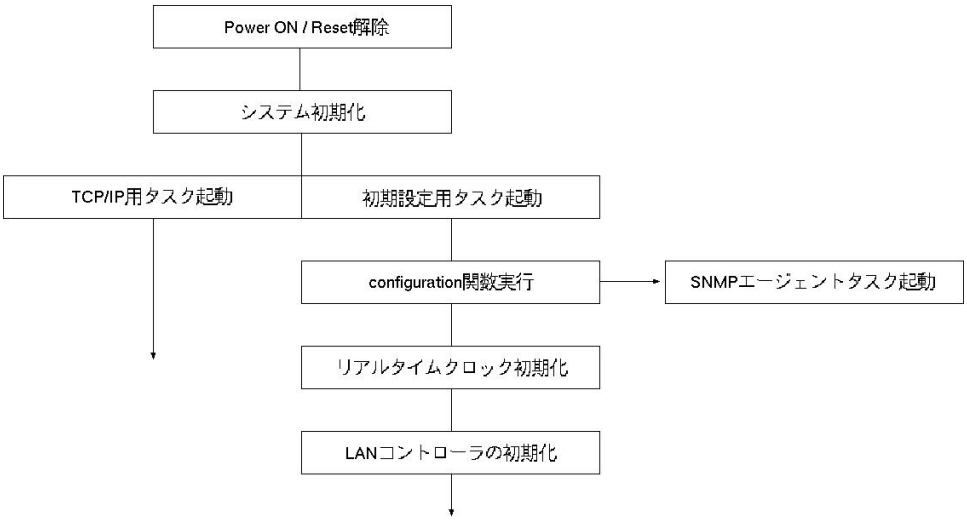


図 5.3-5　SNMPの起動シーケンス

(4-3) Configuration 関数

初期設定用タスクは configuration 関数を実行する。configuration 関数はユーザが作成する関数であり、必ず configuration という関数名にする必要がある。図 5.3-6 に記述例を示す。

```
#include <mr32r.h>

extern void main(int argc, char *argv[]);

void configuration(void){
        T_CTSK   ctask;

        ctask.tskatr = __MR_EXT;
        ctask.task = main;      /* SNMPタスクの指定 */
        ctask.itskpri = 5;
        ctask.stksz = 10240;
        cre_tsk(3, &ctask);
        sta_tsk(3, 0);
}
```

図 5.3-6　configration関数の記述例

(5) 開発環境

(5-1) クロス開発環境

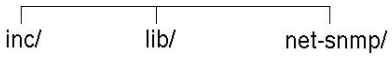
SNMP over T-Engine の実行モジュールを作成するには、ターゲットとなる T-Engine 専用のクロスコンパイラを使用して PC 上で開発を行う、いわゆるクロス開発を行う。表 5.3-4 にクロス開発環境について示す。なお、開発ホストとなる PC には特に制約はない。また、Linux のバージョンについては今回の開発で使用したカーネルのバージョンを示す。

表 5.3-4　クロス開発環境

No.	項目	内容
1	OS	Linux 2.4.18
2	開発ツール	Red Hat 社製 GNUPro ツールキット Ver. 2.93
3	開発言語	C 言語

(5-2) ディレクトリ構成

図 5.3-7 にクロス開発環境におけるディレクトリ構成を示す。図 5.3-7 の inc/及び lib/は三菱電機製μT-Engine「M3T-M32104UT」固有のシステムヘッダファイル及びライブラリが格納されている。 net-snmp/が SNMP over T-Engine 開発用のディレクトリである。



D)　　図 5.3-7　クロス開発環境のディレクトリ構成

(5-3) モジュール構成

SNMP over T-Engine の開発環境において、モジュールはディレクトリとして表現される。図 5.3-8 に SNMP over T-Engine のモジュール構成を示す。図 5.3-8 中の×印が記載されているディレクトリは SNMP over T-Engine では使用しない。

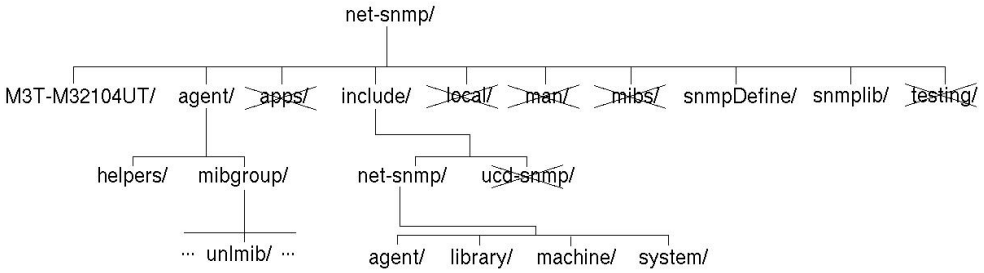


図 5.3-8　モジュール構成

(a) net-snmp/

SNMP over T-Engine の Top ディレクトリである。 SNMP の移植ソフトウェアである NET-SNMP のディレクトリ構造を色濃く残している。 M3T-M32104UT/及び snmpDefine/は SNMP over T-Engine 用に新規に作成したディレクトリである。また、apps/、local/、man/、mibs/、testing/については SNMP over T-Engine では使用しない。

以下に主なファイルの概要について示す。

表 5.3-5 net-snmpモジュール概要

No.	項目	内容
1	Define.h	SNMP over T-Engine でのネットワークやファイルシステム、OS の提供機能などシステム毎の差異を吸収するためのヘッダファイル。agent/、helpers/、mibgroup/、include/、snmpDefine/、snmplib/配下のすべてのヘッダファイル/ソースファイルは Define.h をインクルードする必要がある。
2	configure	make 環境構築用のスクリプト。本スクリプトを実行すると、net-snmp/Makefile、agent/Makefile、agent/helper/Makefile、agent/mibgroup/Makefile、snmplib/Makefile 及び include/net-snmp/net-snmp-config.h が作成される。net-snmp-config.h は configure スクリプトによる環境チェック結果を反映しているものだが、あくまでクロス開発ホストの環境を反映したものであることに注意。よって、configure スクリプトは、各ディレクトリの Makefile 作成の為に使用する。その場合に、各ディ

		レクトリの Makefile は configure 実行時のパラメータを反映したものとなる。
3	net-snmp-config.h.define	No.2 で説明した net-snmp-config.h はクロス開発ホストの環境を反映したものなので、実際のターゲットに即した net-snmp-config.h を作成する必要がある。net-snmp-config.h.define は三菱電機製μT-Engine「M3T-M32104UT」用の net-snmp-config.h で、include/net-snmp/ 配下に net-snmp-config.h というファイル名で配置する必要がある。
4	Makefile	Top の Makefile。configure スクリプトによって自動生成される。Makefile を実行すると agent/Makefile、agent/helper/Makefile、agent/mibgroup/Makefile、snmplib/Makefile を実行する。snmpDefine/の Makefile.define や M3T-M32104UT/の makeScript は実行しないので注意。また、この Makefile は各ディレクトリの Makefile を実行しコンパイルが正常終了すると、コンパイラ(のリンクモード)を使用してリンクを行おうとする。コンパイラとリンカが別ツールとなっている開発環境の場合はリンクエラーとなるので注意。その場合は、リンクディレクトリである agent/で別途リンクする必要がある。

(b) M3T-M32104UT/

このディレクトリは、三菱電機製μT-Engine の固有の処理に関するモジュールで、SNMP エージェントタスクの起動に関係する configuration 関数や低水準関数などが定義される。以下に主なファイルの概要について示す。

表 5.3-6　M3T-M32104UTモジュール概要

No.	項目	内容
1	configure.c	初期設定用タスクからコールされる configuration 関数を定義。
2	ioinit.c	SDRAM コントローラ初期化関数、デバッグシリアル入出力関数を定義。
3	lowlib.c	低水準関数モジュール。
4	ut_gloss.c	低水準関数モジュール。本システムにおいて malloc 関数で確保される領域は、本ファイル内で定義される _SBRKSIZE の値によって決定される。
5	ut_printf.S	表5.2No.4のMakefileより実行されるMakefile。agent/配下のソースファイルをコンパイルし、agent/.libs/にライブラリとして格納する。
6	makeScript	コンパイル用スクリプト。No.1～No.5 のソースファイルをコンパイルし、それらをリンクディレクトリである agent/へコピーする。このスクリプトは Top の Makefile からコールされないので注意。

(c) agent/

このディレクトリはSNMPエージェントの処理に関するモジュールである。このディレクトリの配下にあるhelpers/はSNMPエージェントの内部処理に関するモジュールである。mibgroup/はMIBに関するモジュールである。また、このディレクトリは helpers/、mibgroup/、snmpDefine/、snmplib/の各モジュールで作成されたライブラリをリンクするディレクトリでもある。

以下に主なファイル及びディレクトリの概要について示す。

表 5.3-7 agentモジュール概要

No.	項目	内容
1	snmpd.c	configuration 関数から起動される SNMP エージェントタスクの main 関数を定義。すべてはここから始まる。
2	link.ld	リンク時にリンカに指定するセクション設定ファイル。このファイルの先頭でオブジェクトファイル snmpd.o、configure.o、ioinit.o、snmpd.o、ut_printf.o、ut_gloss.o を指定する。
3	link_command	SNMP over T-Engine のリンカスクリプト。コンパイルにて作成されるライブラリ及びシステムライブラリと link.ld で指定したオブジェクトファイルをリンクする。
4	Makefile	表 5.2No.4 の Makefile より実行される Makefile。agent/配下のソースファイルをコンパイルし、agent/.libs/にライブラリとして格納する。
5	.libs/	No.4 の Makefile により作成される agent/モジュールのライブラリが格納される。自動作成される。
6	helpers/	SNMP エージェントの内部処理に関するモジュール。helpers/にも表 5.2No.4 の Makefile より実行される Makefile が存在し、作成したライブラリを helpers/.libs/に格納する。

(d) mibgroup/

SNMP over T-Engine が管理する MIB モジュールが格納される。1 つの MIB モジュールが 1 ディレクトリに対応している。新規に MIB を作成する場合はこのディレクトリ配下に配置する必要がある。

以下に主なファイル及びディレクトリの概要について示す。

表 5.3-8 mibgroupモジュール概要

No.	項目	内容
1	util_funcs.c	メッセージチェック関数など、各 MIB モジュールにおいて共通で使用される内部関数を定義。
2	mibII/	標準 MIB のモジュール。その大半が UNIX 系に特化した仕様となっているため、それ以外のシステムにそのまま移植するのは困難。
3	unlmib/	新規作成 MIB のサンプルモジュール。
4	Makefile	表 5.2No.4 の Makefile より実行される Makefile。どのモジュールをコンパイルするかは、configure 実行時に指定するパラメータ(組み込む MIB モジュールを指定するパラメータ)によって決まる。作成したライブラリは agent/配下に格納される。

(e) include/

include/net-snmp/agent/は agent/、include/net-snmp/library/には snmplib/のソースファイルに対応するヘッダファイルが格納される。

以下に主なファイル及びディレクトリの概要について示す。

表 5.3-9 includeモジュール概要

No.	項目	内容
1	net-snmp-config.h	本来は configure スクリプトにより自動生成されるファイル。configure スクリプトは実行時のパラメータやシステムの持つヘッダファイルの有無などをチェックして、チェック結果をフラグとして net-snmp-config.h に記述する。そのフラグが SNMP の構造に大きく影響する。但し、configure を実行して

		もスクリプトがチェックするのはクロス開発用の PC の環境なので、ターゲット用の net-snmp-config.h を自力で作成する必要がある。
2	include/net-snmp/agent/	agent/のソースに対応するヘッダファイルが格納。
3	include/net-snmp/library/	snmplib/のソースに対応するヘッダファイルが格納。

(f) snmpDefine/

Define.h にて定義される関数の実体を定義するモジュール。
以下にファイルの概要について示す。

表 5.3-10　snmpDefineモジュール概要

No.	項目	内容
1	Define.c	Define.h で宣言した関数の実体を定義するファイル。SNMP over T-Engine において、機能上、必要でない関数をダミー関数として定義したり、ターゲット固有の初期化関数である target_init()を定義。三菱電機製μT-Engine「M3T-M32104UT」ではネットワークの設定情報をファイルから読み出してネットワークの設定を行う。また、ファイルシステムのようなシステム毎に異なる関数を定義する。SNMP over T-Engine では以下を定義。 te_fs_open()：ファイルのオープン te_fs_close()：ファイルのクローズ te_fs_read()：低水準関数 read()相当 te_fs_write()：低水準関数 write()相当 te_fs_lseek()：低水準関数 lseek()に相当

		te_fs_fgets()：fgets()に相当 te_fs_getc()：getc()に相当 te_fs_ungetc()ungetc()に相当
2	Makefile.define/	Define.c からライブラリを作成するスクリプト。作成したライブラリは snmpDefine/.libs/に格納される。表 5.2No.4 の Makefile からは起動されないので注意。

(g) snmplib/

snmp プロトコルに関するモジュールである。以下に主なファイルの概要について示す。

表 5.3-11 snmplibモジュール概要

No.	項目	内容
1	read_config.c	コンフィグレーションファイルを読み込む関数などを定義。Define.h に定義される配列 path_config[]に指定されるファイルを読み込む。 また、read_config.c に定義される以下の関数はファイルシステムに大きく依存する仕様である。SNMP over T-Engine をファイルシステムの異なるシステムに移植する際には注意が必要である。 read_config() read_configs()
2	snmpUDPDomain.c	UDP に関する処理を定義。提供されるソケット関数はシステムによって異なるが、BSD ソケット API に準拠していれば Define.h において以下にように#define で読み換えるだけでよい。 例：

		#define sendto unix_sendto #define recvfrom unix_recvfrom ※ unix_sendto は M2S-AE32R04 TCP/IP プロトコルスタックにおける関数名。

（エ）実装確認

本研究では､電源管理プロファイルの実装方法として SNMP の拡張 MIB を導入し、またネットワーキングの方法として標準化がされている SNMP プロトコルを使用した。

以下に、SNMP の GetRequest を用いて、μT-Engine 上に構築したネットワーク電源管理用の MIB にアクセスした結果を示す(GetRequest コマンドの記述は省略)。実装した拡張 MIB は、その目的が閉じた領域でのネットワーク電源管理のトライアルにつき、暫定的な enterprise 番号“16500”を付与している。フィールドトライアル時には、IETF に対して所定の申請を行う必要がある。

以下は表 5.3-1 中のシステムの箇所を表現している。網掛けで記述している個所が GetRequest で取得したネットワーク電源管理プロファイルの管理項目となる。

製造者：
SNMPv2-SMI::enterprises.16500.1.1.1.0 = STRING: "MITSUBISHI ELECTRIC"

機器名：
SNMPv2-SMI::enterprises.16500.1.1.2.0 = STRING: "GZ-32"

機器種別：
SNMPv2-SMI::enterprises.16500.1.1.3.0 = STRING: "Web-Camera"

設置名：
SNMPv2-SMI::enterprises.16500.1.1.4.0 = STRING: "unknown"

設置場所：
SNMPv2-SMI::enterprises.16500.1.1.5.0 = STRING: "unknown"

管理者：
SNMPv2-SMI::enterprises.16500.1.1.6.1.0 = STRING: "Atsushi Kato"
SNMPv2-SMI::enterprises.16500.1.1.6.2.0 = STRING: "03-5437-2336(732)"
SNMPv2-SMI::enterprises.16500.1.1.6.3.0 = STRING: "atsushi@ubin.jp"
SNMPv2-SMI::enterprises.16500.1.1.6.4.0 = STRING: "http://10.129.0.1/"

機器版数

SNMPv2-SMI::enterprises.16500.1.1.7.1.0 = STRING: "1.B0.01"
SNMPv2-SMI::enterprises.16500.1.1.7.2.0 = STRING: "M3T-M32104UT"

経過時間：
SNMPv2-SMI::enterprises.16500.1.1.8.0 = Timeticks: (5700) 0:00:57.00

製造年月：
SNMPv2-SMI::enterprises.16500.1.1.9.0 = STRING: "2002-10"

製造番号：
SNMPv2-SMI::enterprises.16500.1.1.10.0 = STRING: "8G108554"

保守情報：
SNMPv2-SMI::enterprises.16500.1.1.11.0 = STRING: "http://10.129.0.1/support/manual.pdf"

以下は表 5.3-1 中の電源の箇所を表現している。網掛けで記述している個所が GetRequest で取得したネットワーク電源管理プロファイルの管理項目となる。

搭載電源種別：
SNMPv2-SMI::enterprises.16500.1.2.4.1.0 = INTEGER: 3

動作電源種別：
SNMPv2-SMI::enterprises.16500.1.2.4.2.0 = INTEGER: 2

バッテリユニット搭載数：
SNMPv2-SMI::enterprises.16500.1.2.4.3.1.0 = INTEGER: 2

バッテリユニット識別子：
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.1.1 = INTEGER: 1
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.1.2 = INTEGER: 2

バッテリユニット接続状態：
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.2.1 = INTEGER: 3
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.2.2 = INTEGER: 3

バッテリ残り時間：
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.3.1 = INTEGER: 180
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.3.2 = INTEGER: 180

バッテリ容量閾値：
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.4.1 = INTEGER: 30

```
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.4.2 = INTEGER: 30

バッテリ状態：
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.5.1 = INTEGER: 2
SNMPv2-SMI::enterprises.16500.1.2.4.3.2.1.5.2 = INTEGER: 2

ネットワーク WakeUp 対応：
SNMPv2-SMI::enterprises.16500.1.2.4.4.0 = INTEGER: 1

ネットワーク ShutDown 対応：
SNMPv2-SMI::enterprises.16500.1.2.4.5.0 = INTEGER: 1

消費電力状態：
SNMPv2-SMI::enterprises.16500.1.2.4.6.0 = INTEGER: 2

電源制御対応：
SNMPv2-SMI::enterprises.16500.1.11.1.0 = INTEGER: 2

IDLE 時低消費電力モード移行制御：
SNMPv2-SMI::enterprises.16500.1.11.2.0 = INTEGER: 1

サスペンド移行制御：
SNMPv2-SMI::enterprises.16500.1.11.3.0 = INTEGER: 2

サスペンド移行時間：
SNMPv2-SMI::enterprises.16500.1.11.4.0 = INTEGER: 0

ディスプレイ制御：
SNMPv2-SMI::enterprises.16500.1.11.5.0 = INTEGER: 0

電源操作：
SNMPv2-SMI::enterprises.16500.1.11.6.0 = INTEGER: 0
```

（オ）課題及び今後の方針

本研究の目的は、ユビキタス社会の省エネルギー化におけるネットワーク電源管理の有効性を検証するものである。今後は策定したネットワーク電源管理プロファイル及び実装した SNMP を用いて、その検証を行うことになる。その場合の課題として、規定した電源管理プロファイルの項目の中には現状の T-Engine の仕様では取得や制御が不可能な項目も存在する。更なる検討を続け省エネルギー化に有効と思われる項目については、T-Engine への機能追加及び改良を促す必要がある。また、今回はネットワーキングの 1 方法として SNMP を採用したが、SNMP に限らず設備機器に実装する場合にはプログラムサイズ及び性能についても十分考慮

する必要がある。また、潤沢な機能を有さない機器の管理についても検討を行う必要がある。それらを踏まえ継続して研究を行っていく。

5.3.2 強化現実型ユーザインタフェース

5.3.3 に述べる超音波センサと RFID を組み合わせた位置検出機構の研究及び、【サブテーマ２】で実施した屋内用センサネットプロトコルで得られる位置情報やユビキタス環境情報を実際の屋内モデルにマッピングしてユーザに提示する研究を実施した。

ユビキタス環境において、環境内のオブジェクトを管理してコンテクスト依存のユビキタスサービスを提供する場合、ユビキタスコンピュータが現在管理している状況をわかりやすい方法でユーザに提供することが重要となる。このひとつの手法が強化現実型ユーザインタフェースである。

今年度は、この強化現実型ユーザインタフェースの基盤となる、現実とほぼ類似した３次元表示によりユビキタス環境をユーザに提示する研究を行った。

本研究では、5.3.3 節　位置検出機構　で述べる、超音波センサと RFID を組み合わせた位置検出方式で取得した、ユビキタス環境内のオブジェクトを３次元表示するとともに、5.3.1 アドホックネットワーキングの研究　で述べた屋内用センサネットの実験システムにおいて壁面に設けられた温度センサから得られる温度情報をもとに、壁面の温度を推定し、実環境にマッピングしてユーザに提示する研究を行った。

以下にその概要を示す。

（ア）　クライアント検索システム

クライアント検索システムは、３次元表示するべきオブジェクトを選択するためのユーザインタフェースを提示する。この処理概要を以下の図に示す。

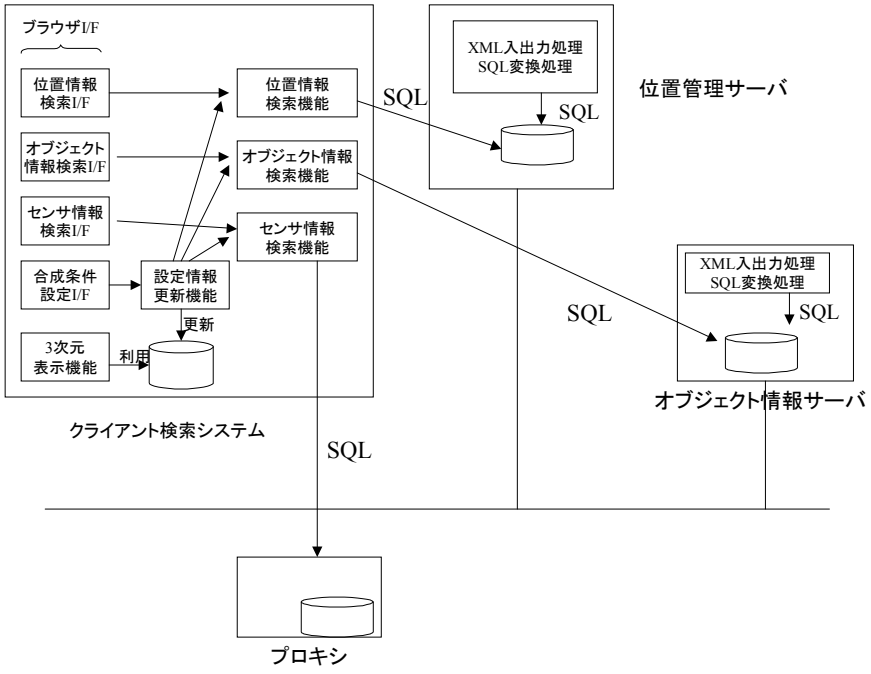


図 5.3-9　クライアント検索システムの処理概要

（１） 機能概要

本機能は、３次元表示させるオブジェクトおよびセンサをクライアントのブラウザ画面から選択し、それらの情報をデータベース登録する。

（２） 機能一覧

以下に、実装される機能を示す。

(2-1) オブジェクト情報検索

３次元表示させるオブジェクトをブラウザから選択し、選択したオブジェクト情報をデータベース登録する。本機能は３画面を有する。

(a) オブジェクト情報検索画面

ブラウザから検索条件を入力する。

(b) オブジェクト情報検索結果一覧画面

(a) で入力された検索条件に合致するオブジェクト情報の一部を画面表示し、さらに詳細情報を表示するオブジェクトを選択する。このとき、選択されたオブジェクト情報がデータベースに登録される。

(c) オブジェクト情報詳細画面

(b) で選択されたオブジェクトの詳細情報を画面表示する。

(2-2) センサ情報検索

３次元表示させるセンサをブラウザから選択し、選択したセンサ情報をデータベース登録する。本機能は３画面を有する。

(a) センサ情報検索画面

ブラウザから検索条件を入力する。

(b) オブジェクト情報検索結果一覧画面

(a) で入力された検索条件に合致するセンサ情報の一部を画面表示し、さらに詳細情報を表示するセンサを選択する。このとき、選択されたセンサ情報がデータベースに登録される。

(c) センサ情報詳細画面

(b) で選択されたセンサの詳細情報を画面表示する。

(2-3) 合成条件設定

３次元表示対象のオブジェクトまたはセンサを選択し、データベース登録する。本機能は１画面を有する。

（３）　画面遷移

以下に画面遷移を示す。

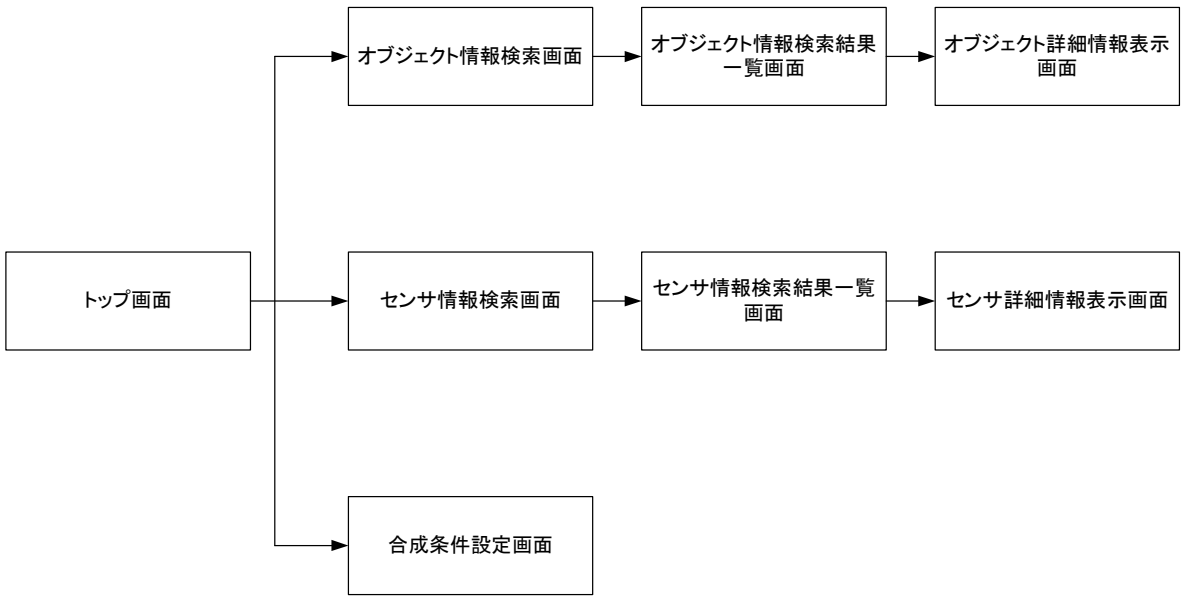


図 5.3-10　クライアント検索画面遷移図

トップ画面から、オブジェクト情報検索／センサ情報検索／合成条件設定を選択後、各画面へと遷移する。

（４）　機能詳細

(4-1) クライアント検索

(a) トップ画面

(a-1) 画面イメージ

オブジェクト情報検索／センサ情報検索／合成条件設定画面にそれぞれ遷移する。

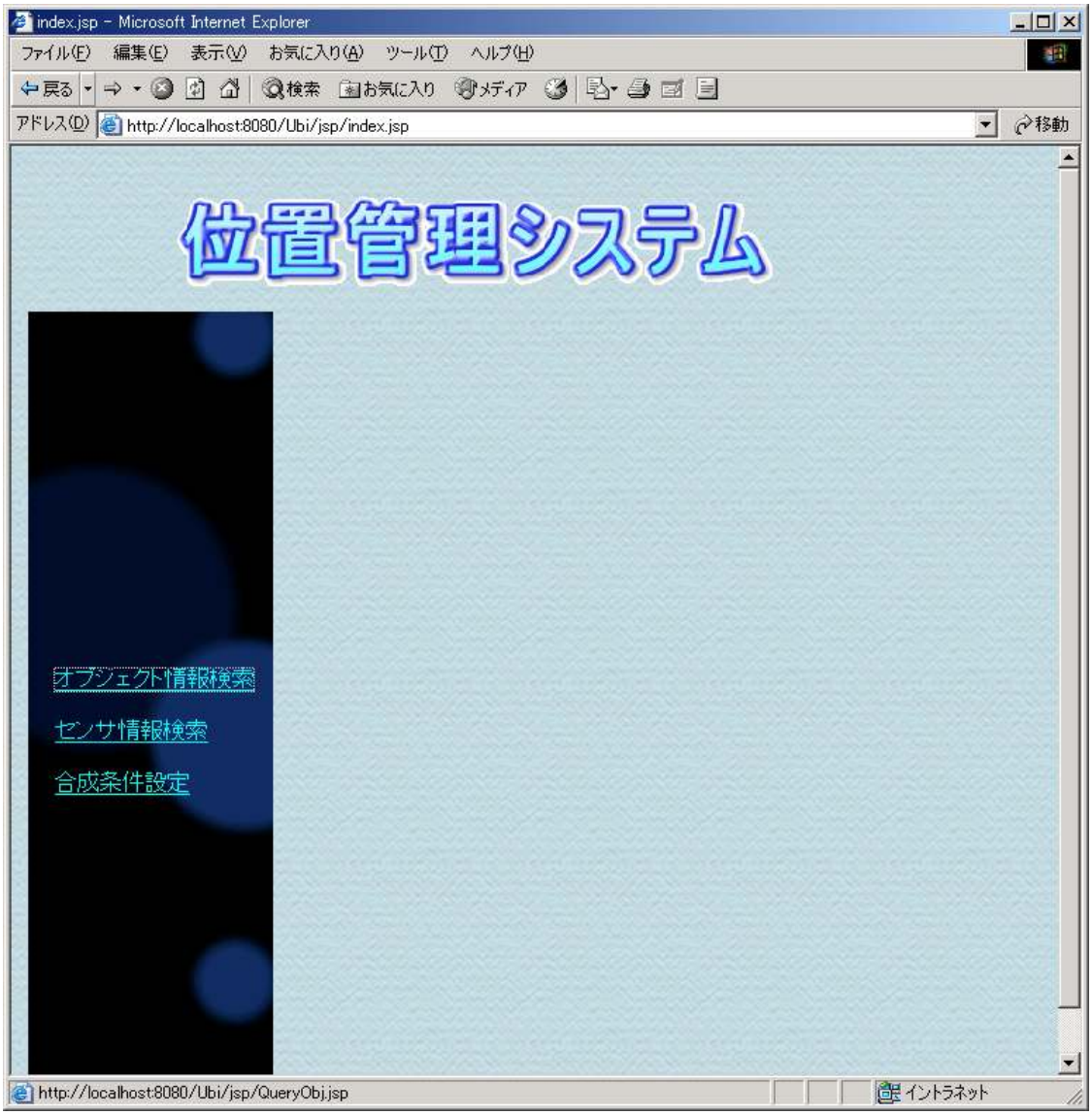

index.jsp - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り メディア
アドレス(D) http://localhost:8080/Ubi/jsp/index.jsp
移動
位置管理システム
オブジェクト情報検索
センサ情報検索
合成条件設定
http://localhost:8080/Ubi/jsp/QueryObj.jsp
イントラネット

図 5.3-11 トップ画面

(4-2) オブジェクト情報検索

オブジェクト情報検索画面

(a-1) 画面イメージ

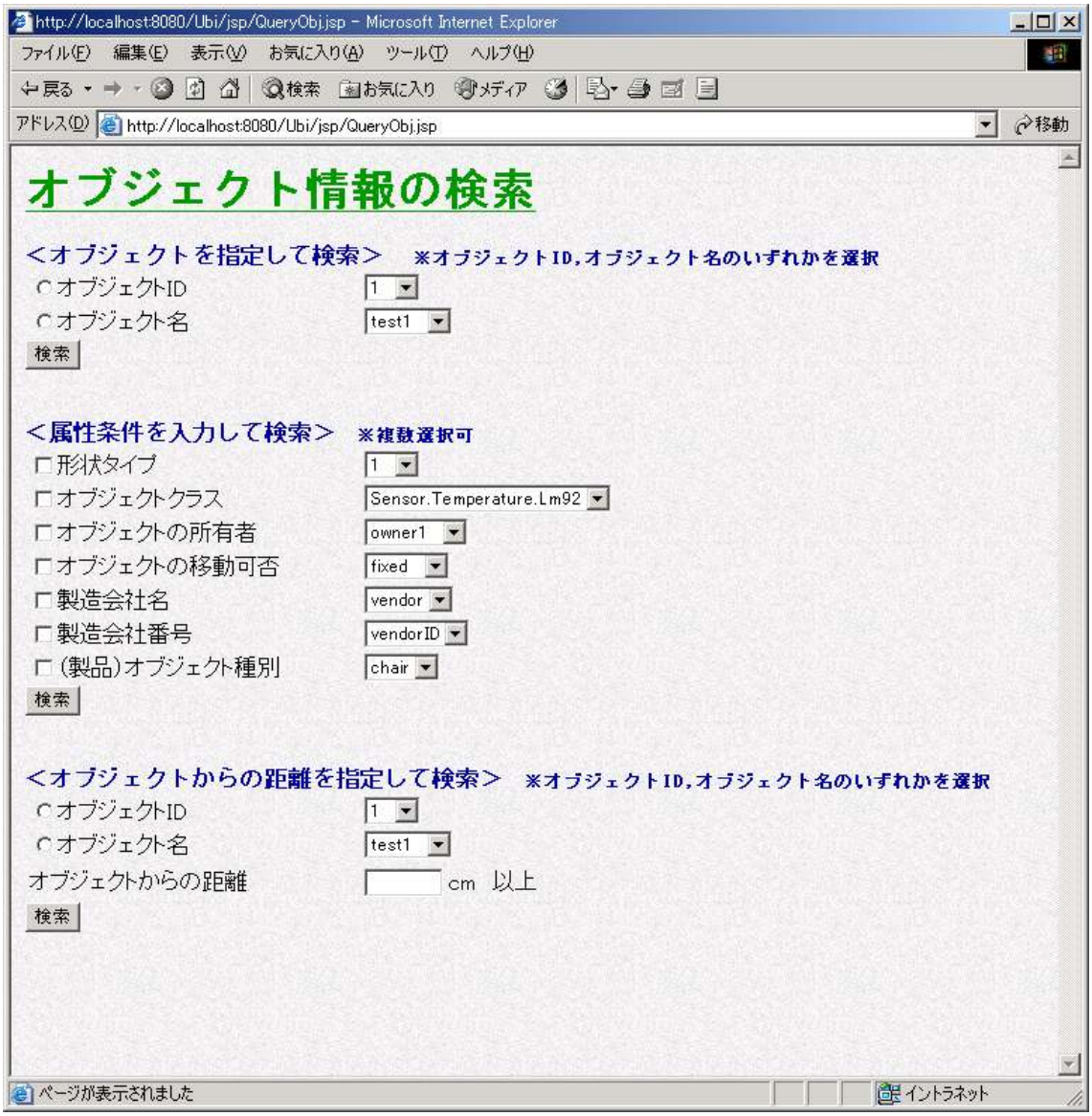


図 5.3-12 オブジェクト情報検索画面

(a-2) 画面操作

オブジェクトの選択方法は、オブジェクトを指定して検索／属性条件を入力して検索／オブジェクトからの距離を指定して検索の 3 とおりある。

<オブジェクトを指定して検索>

(1) 検索対象のオブジェクト ID またはオブジェクト名のいずれかのラジオボタンをチェックする。

(2) (1)でチェックした項目の値を、隣接するリストボックスから選択する。

(3) 検索ボタンを押すと、選択された値を元に位置管理サーバ・オブジェクト情報サーバのテーブルを参照し、オブジェクト情報を取得する。

(4) ＤＢ参照結果を検索結果一覧に表示する（オブジェクト情報検索結果一覧画面）。

●

＜属性条件を入力して検索＞

(1)　検索対象オブジェクトの形状タイプ／オブジェクトクラス／オブジェクトの所有者／オブジェクトの移動可否／製造会社名／製造会社番号／（製品）オブジェクト種別から検索条件をチェックする（複数選択可）。
(2) (1)でチェックした項目の値を、隣接するリストボックスから選択する。
(3) 検索ボタンを押すと、選択された値を元に位置管理サーバ・オブジェクト情報サーバのテーブルを参照し、オブジェクト情報を取得する。
(4) ＤＢ参照結果を検索結果一覧に表示する（オブジェクト情報検索結果一覧画面）。

＜オブジェクトからの距離を指定して検索＞

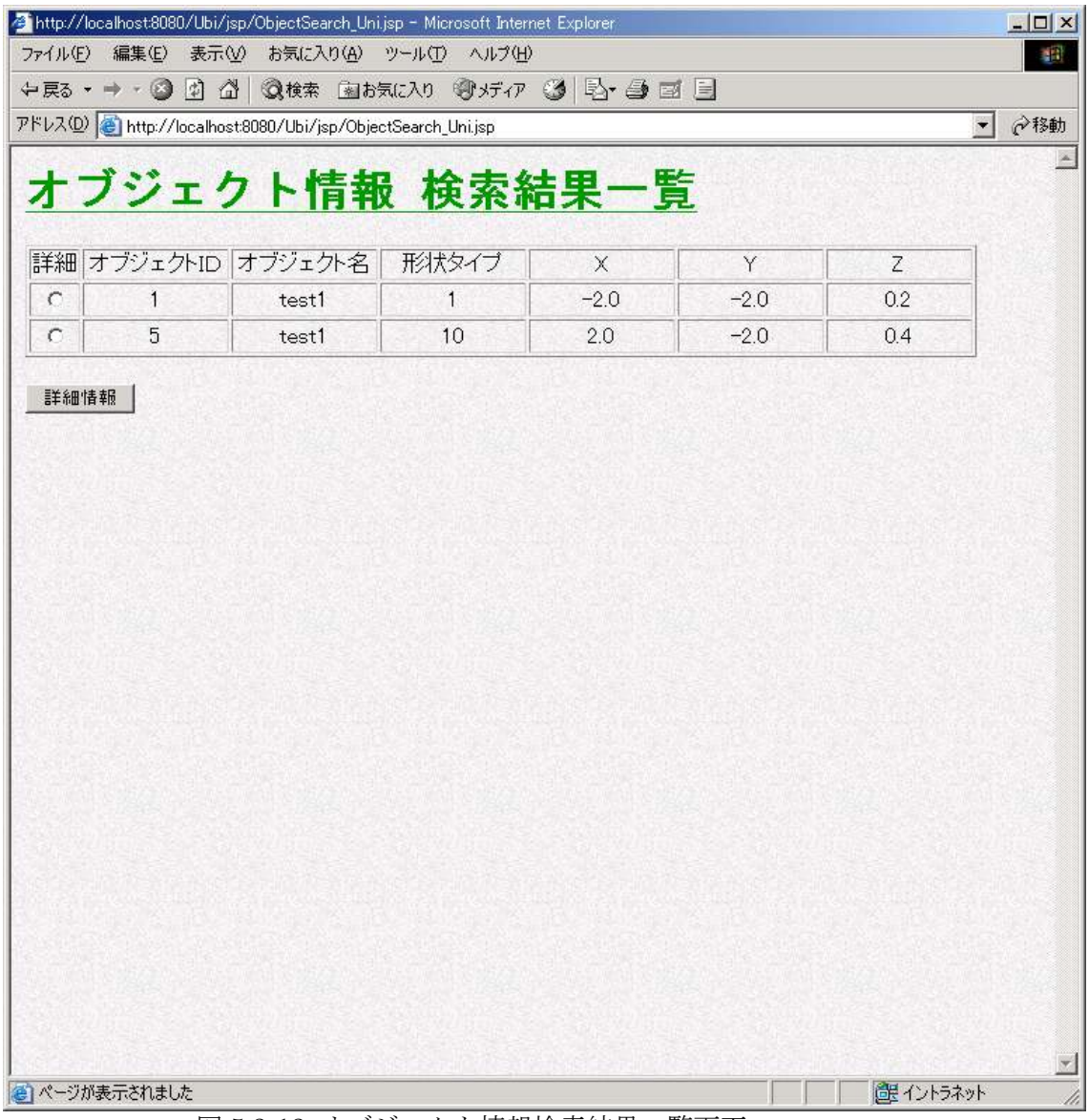
(1) 検索対象のオブジェクト ID またはオブジェクト名のいずれかのラジオボタンをチェックする。
(2) (1)でチェックした項目の値を、隣接するリストボックスから選択する。
(3)　オブジェクトからの距離を指定する。
(4) 検索ボタンを押すと、選択された値を元に位置管理サーバ・オブジェクト情報サーバのテーブルを参照し、オブジェクト情報を取得する。
(5) ＤＢ参照結果を検索結果一覧に表示する（オブジェクト情報検索結果一覧画面）。

●

(b) オブジェクト情報検索結果一覧画面

オブジェクト情報検索画面で指定された検索条件で DB 参照した結果を表示する。

(b-1) 画面イメージ



詳細	オブジェクトID	オブジェクト名	形状タイプ	X	Y	Z
○	1	test1	1	-2.0	-2.0	0.2
○	5	test1	10	2.0	-2.0	0.4

図 5.3-13 オブジェクト情報検索結果一覧画面

(b-2) 画面操作

(1) 詳細情報を表示したいオブジェクトを選択し（ラジオボタンを選択)、詳細情報ボタンを押す（複数選択不可)。

(2) 詳細情報ボタンを押すと、選択されたオブジェクトの詳細情報を位置管理サーバ・オブジェクト情報 サーバのテーブルより取得し、オブジェクト詳細情報表示画面に表示する。

(c) オブジェクト詳細情報表示画面

(c-1) 画面イメージ

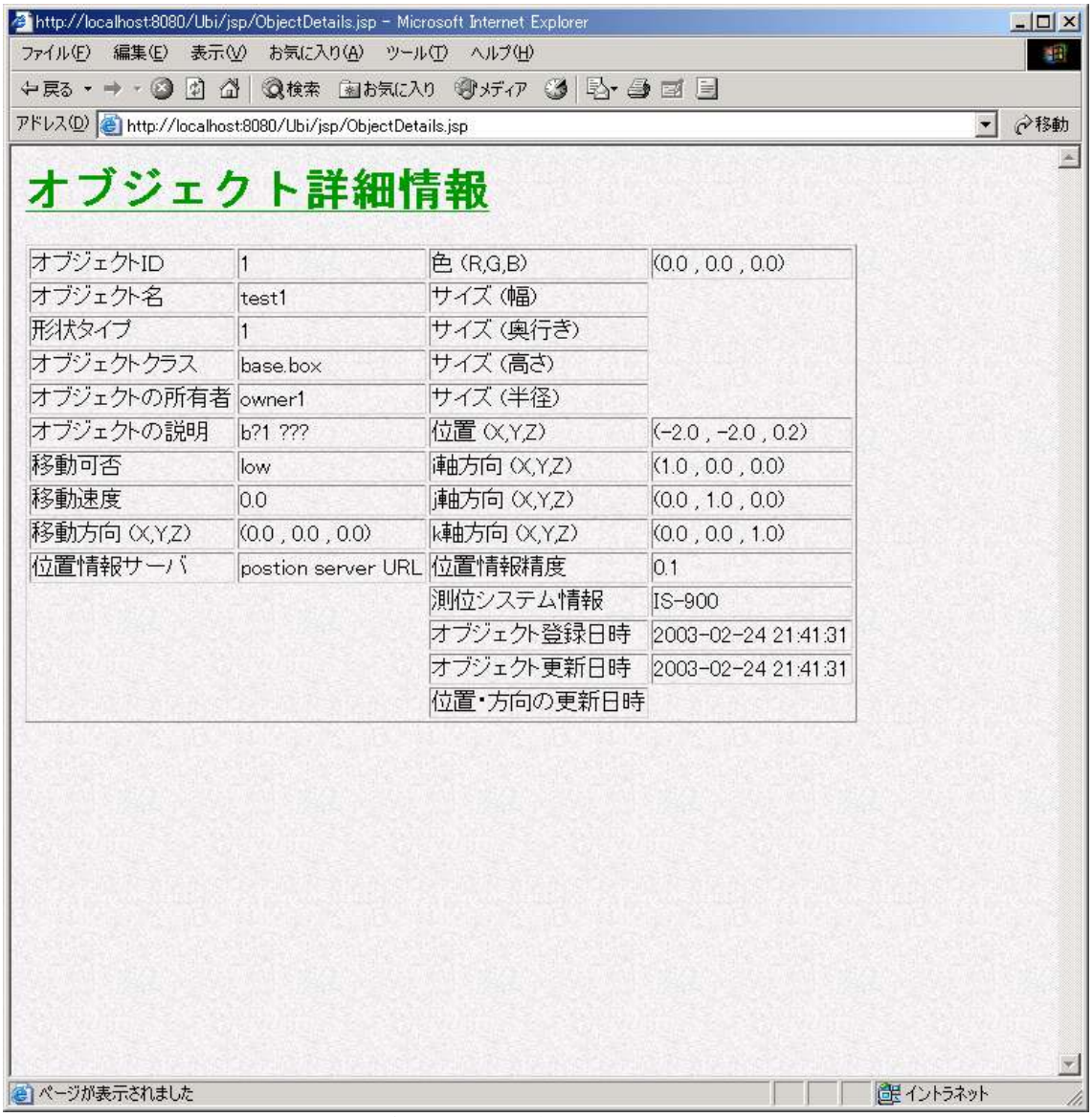


図 5.3-14 オブジェクト詳細情報表示画面

オブジェクト情報検索結果一覧画面で選択されたオブジェクトの詳細情報を表示する。

サイズについては、形状タイプに応じた値のみを表示し、設定されていない項目は空欄とする。

(4-3) センサ情報検索

センサ情報検索画面

(a-1) 画面イメージ

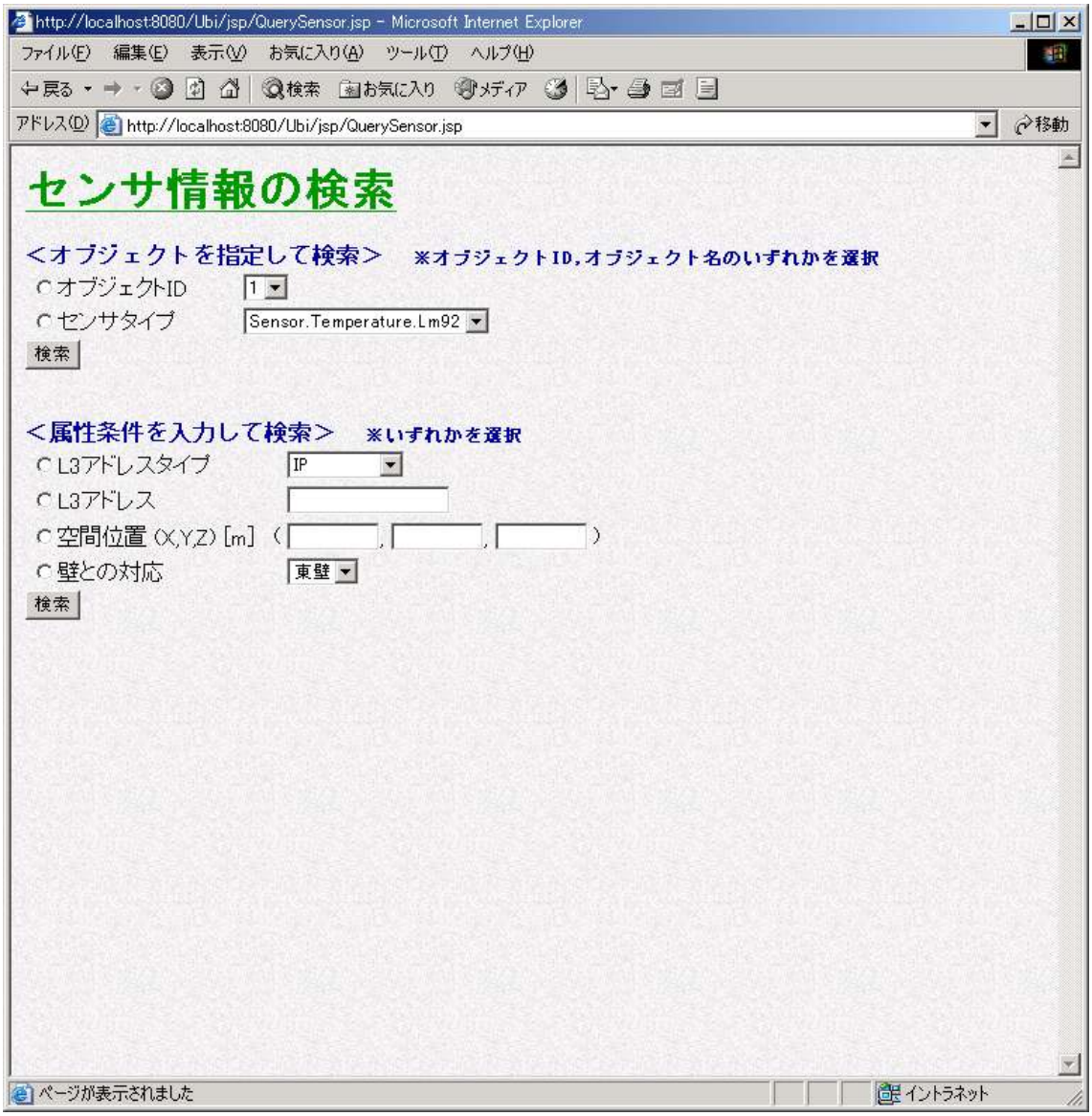


図 5.3-15 センサ情報検索画面

(a-2) 画面操作

オブジェクトの選択方法は、オブジェクトを指定して検索／属性条件を入力して検索の 2 とおりある。

<オブジェクトを指定して検索>

(1) 検索対象のオブジェクト ID またはセンサタイプのいずれかのラジオボタンをチェックする。

(2) (1)でチェックした項目の値を、隣接するリストボックスから選択する。

(3) 検索ボタンを押すと、選択された値を元に位置管理サーバのテーブルを参照し、センサ情報を取得する。

(4) ＤＢ参照結果を検索結果一覧に表示する（センサ情報検索結果一覧画面)。

●

＜属性条件を入力して検索＞

(1)　検索対象オブジェクトの L3 アドレスタイプ／Ｌ３アドレス／空間位置（X,Y,Z）／壁との対応のいずれかのラジオボタンをチェックする。

(2) (1)でチェックした項目の値を、隣接するリストボックスから選択または直接入力する。

(3) 検索ボタンを押すと、選択された値を元に位置管理サーバ・オブジェクト情報サーバのテーブルを参照し、センサ情報を取得する。

(4) ＤＢ参照結果を検索結果一覧に表示する（センサ情報検索結果一覧画面)。

●

(b) センサ情報検索結果一覧画面

センサ情報検索画面で指定された検索条件で DB 参照した結果を表示する。

(b-1) 画面イメージ

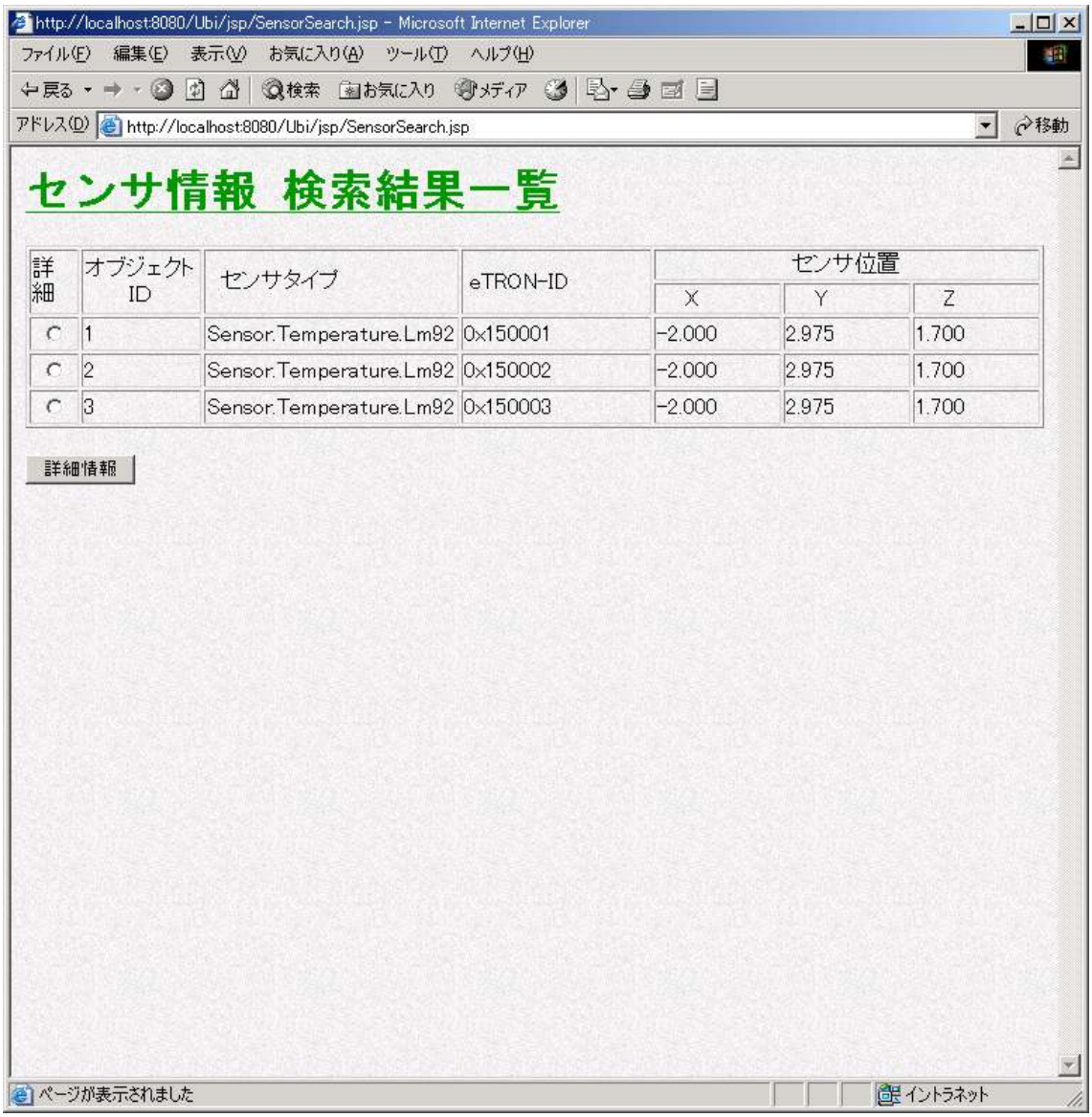


図 5.3-16 センサ情報検索結果一覧画面

(b-2) 画面操作

(１) 詳細情報を表示したいセンサを選択し（ラジオボタンを選択)、詳細情報ボタンを押す（複数選択不可)。

(2) 詳細情報ボタンを押すと、選択されたセンサの詳細情報をオブジェクト情報サーバのテーブルより取得し、センサ詳細情報表示画面に表示する。

(c) センサ詳細情報表示画面

(c-1) 画面イメージ

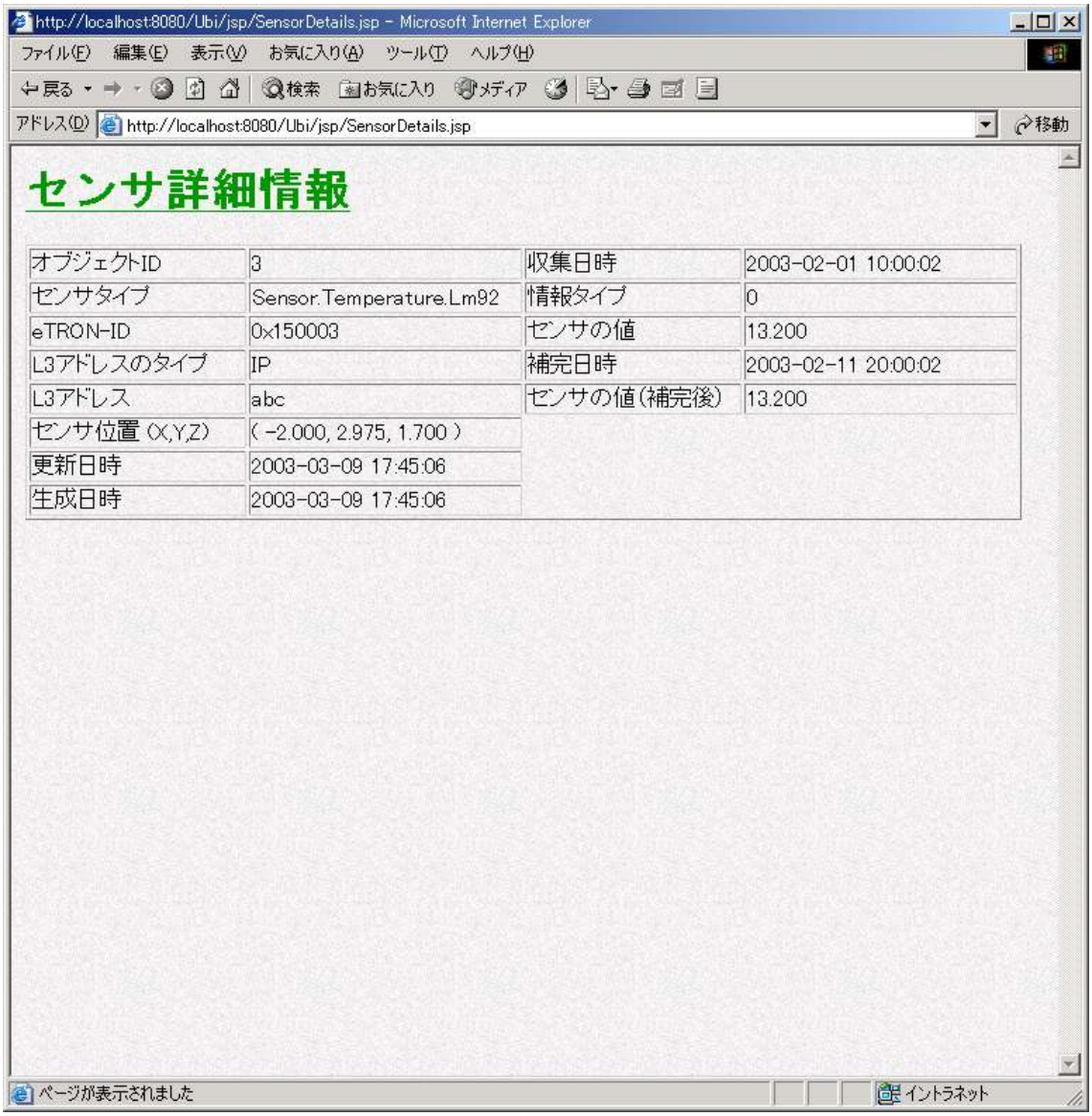


図 5.3-17 センサ詳細情報画面

(4-4) 合成条件設定

(a) 合成条件設定画面

オブジェクト／センサ情報検索結果一覧表示画面で選択されたオブジェクト ID が登録対象のオブジェクトリストに格納された状態で画面表示する。

(a-1) 画面イメージ

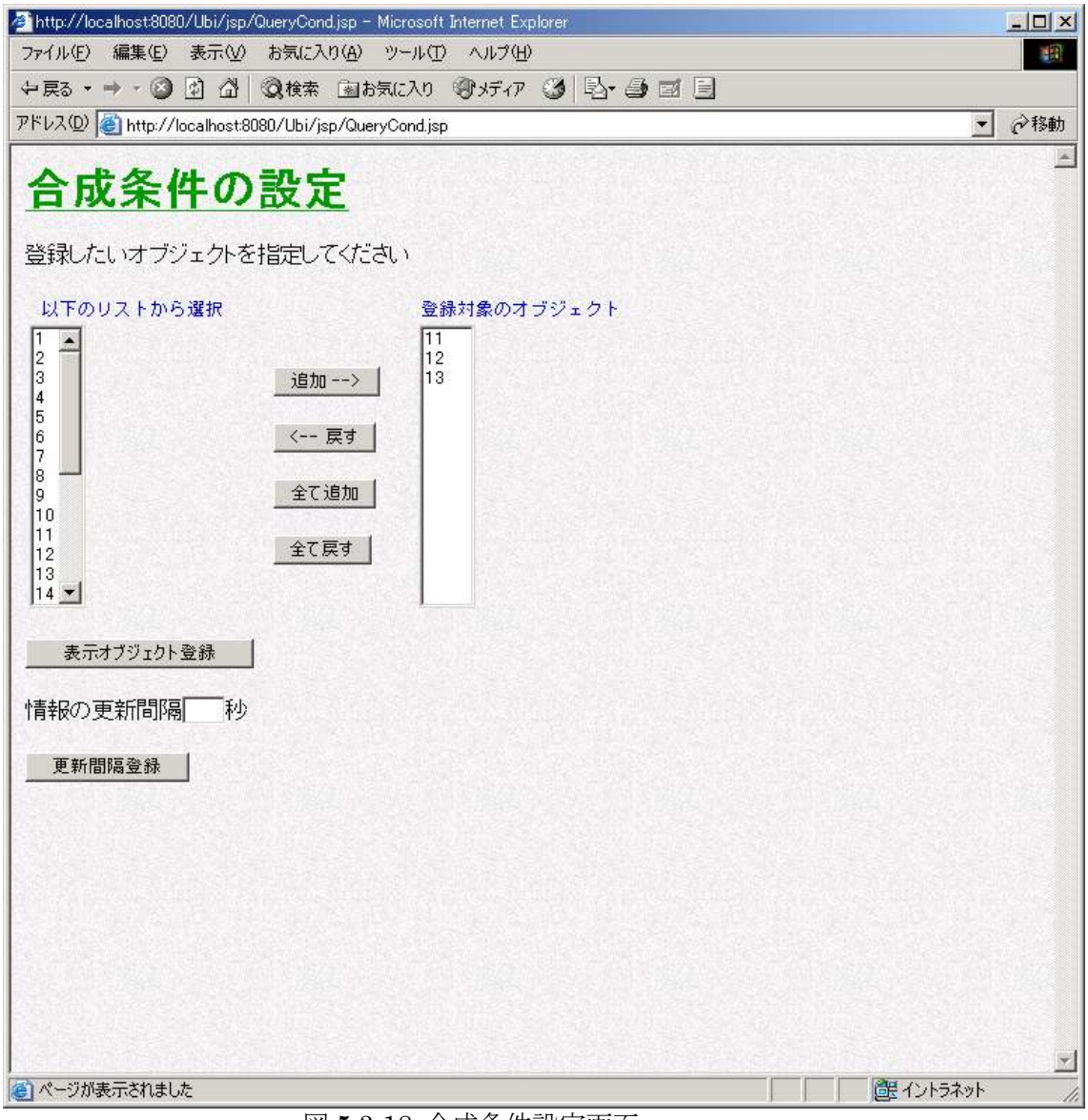


図 5.3-18 合成条件設定画面

(a-2) 画面操作

(1) 登録対象オブジェクトの選択

- 合成表示したいオブジェクトを画面左側のリストから選択し（複数選択可）、追加ボタンを押すと、登録対象のオブジェクトリストに追加される。
- 全て追加ボタンを押すと、画面左側の全てのオブジェクトが登録対象のオブジェクトリストに追加される。

(2) 登録対象オブジェクトの削除

- 登録対象から削除したいオブジェクトを画面右側のリストから選択し（複数選択可)、戻すボタンを押すと登録対象のオブジェクトリストから削除される。
- 全て戻すボタンを押すと、画面右側の全てのオブジェクトが登録対象のオブジェクトリストから削除される。

(3) 合成表示するオブジェクトの登録

- 表示オブジェクト登録ボタンを押すと、画面右側のリストに表示されているオブジェクトが DB に登録される。

(4) 情報の更新間隔設定

- 更新間隔登録ボタンを押すと、対象オブジェクトの位置情報・センサ属性情報を DB に更新する間隔を設定する。

- （イ）　実行例

図 5.3-19、図 5.3-20 に仮想空間での現実オブジェクトの表示例を示す。また、図 5.3-21 に図 5.3-20 に対応する実オブジェクトの写真を示す。図からわかるように、実際のオブジェクトの配置に一致する環境が仮想現実として実現されている。

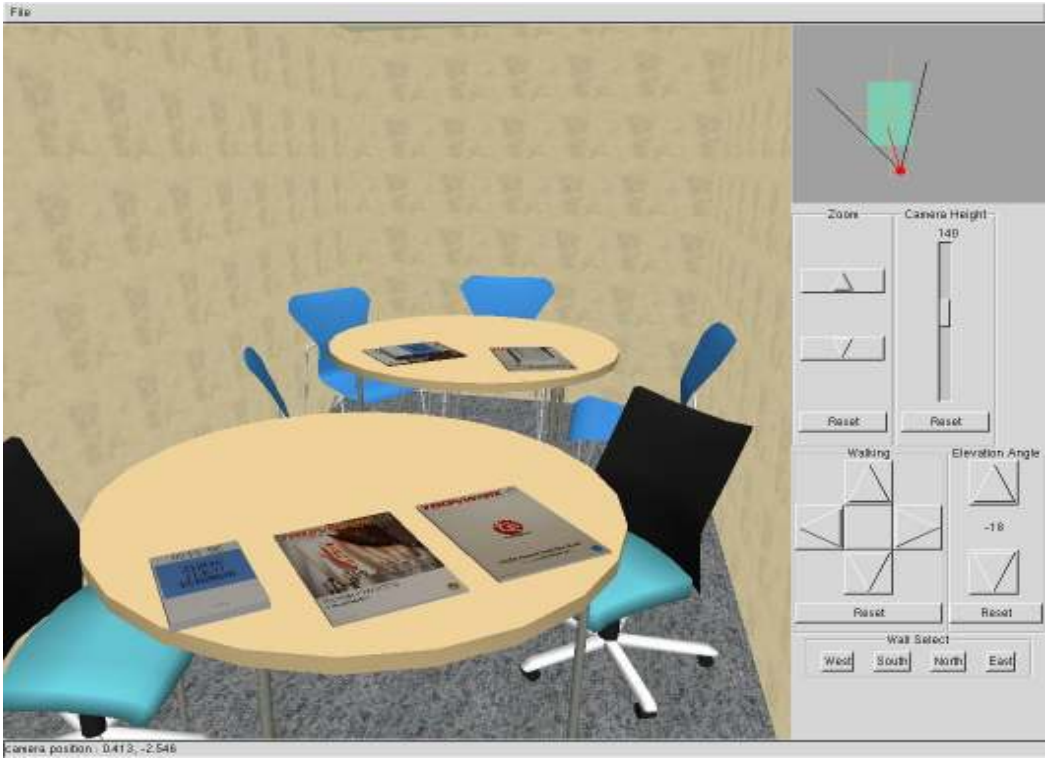


図 5.3-19 仮想空間上での表示例

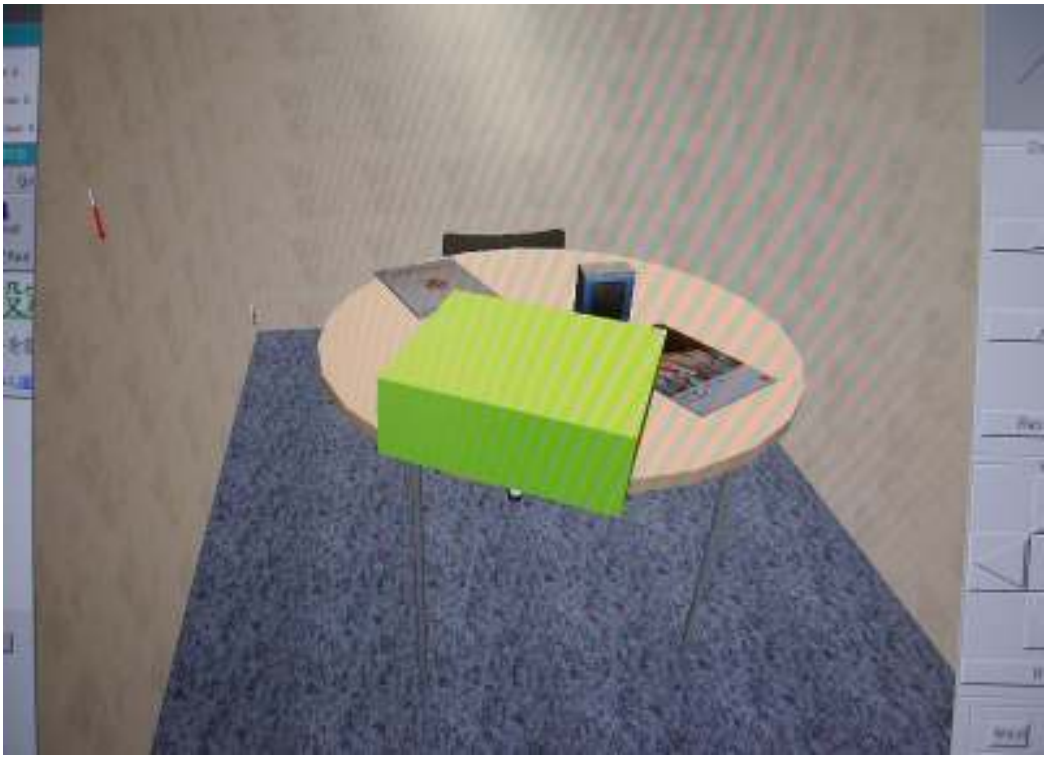


図 5.3-20 仮想空間での表示例　（机、椅子、本、および箱）

図 5.3-21 仮想空間に対応する実オブジェクト(写真)

5.3.3 位置検出機構

現実世界において、より高いユーザ親和性と IT サービスを提供するために、特に屋内､つまり GPS などのグローバルな位置検出インフラが利用できない実環境、における高い精度のユーザ位置検出技術として、超音波センサと RFID を組み合わせた位置検出と、画像認識による位置検出方式を開発した。また取得した位置を管理するためのサーバの開発を行った。

（ア） はじめに

身の回りのあらゆるものにコンピュータが組み込まれるユビキタス環境では、コンピュータ同士の通信のみならずコンピュータが現実世界のあらゆる情報を取得、識別し現実世界と連携した動作が行なえることが重要になってくる。その一つとして、位置情報を活用したコンテクスト依存のサービスを提供することが考えられる。位置に依存したサービスを提供するためには、コンピュータが自分の位置を知るだけではなく、環境に存在する物体の位置も得る事が必要になってくる。しかし、実際には位置を知るためにすべての物に位置情報が取得可能なセンサを取りつけることは難しい。そこでこの問題点を解決するために、今年度は、屋内、すなわち GPS などのグローバルな位置検出インフラが利用できない環境において、多数のセンサを使用しない条件で高精度に位置を検出する機構について、以下の 2 通りの方式を検討した。

- 超音波位置センサと非接触 IC カードを利用した位置検出機構
- 画像認識による位置検出方式

以下にそれらについて報告する。

（イ） 超音波位置センサと非接触 IC カードを使用した位置検出方式

（１） 従来方式

コンピュータで物の位置情報を取得するひとつの方法として、測定するためのセンサ部を対象物に取りつける方法がある[1]。このような方式を用いることで取りつけたセンサ部の位置を得ることが可能になる。しかし、従来方式には次のような問題点がある。

センサのコスト測定対象物に一つ以上(向きを取得するためには最低三つ)のセンサが必要となる。

センサへの電力供給電池駆動の場合には電池寿命が存在する。外部から供給する場合には配線が必要となる。

センサの取り付けセンサの貼付位置が測定点となるため、事前にセンサの取り付け位置を正確に計測しておく必要がある。また、対象物の大きさも測定する場合には、センサを隅へ取りつける必要があり、物理的な制約が大きい。

（２）提案方式

(2-1) 提案方式の概要

本稿ではセンサを取りつける代わりに非接触 IC カードを利用した位置検出方式を提案する。提案方式は屋内での利用を想定し、あまり動かない物を対象として、固有の ID 情報を持ち非接触で通信が可能な IC カード(非接触 IC カード)を位置情報の取得対象となるすべての物(オブジェクト)に貼付ける。位置センサを取りつけた小型の端末を利用し、貼付けたカードから固有の ID 情報などを読み出し対象物の識別を行なう。次に端末により対象物を測定することで対象物の位置と向きを取得する方式である。

提案方式を用いることで、端末のみに位置センサを搭載すればよく、位置情報の取得を安価に行なえる。また、非接触 IC カードは通信時にカードリーダライタから供給される電力により動作するので、電力供給の問題も解決される。さらに非接触 IC カードの貼付位置が測定点ではないため、カードの貼付はオブジェクトの任意の位置に行なえ柔軟性が高い。

(2-2) システム構成

本稿で提案する位置検出方式を利用した位置情報管理システムは図 5.3-22 の構成になる。本システムは、次の四つの要素から構成される。

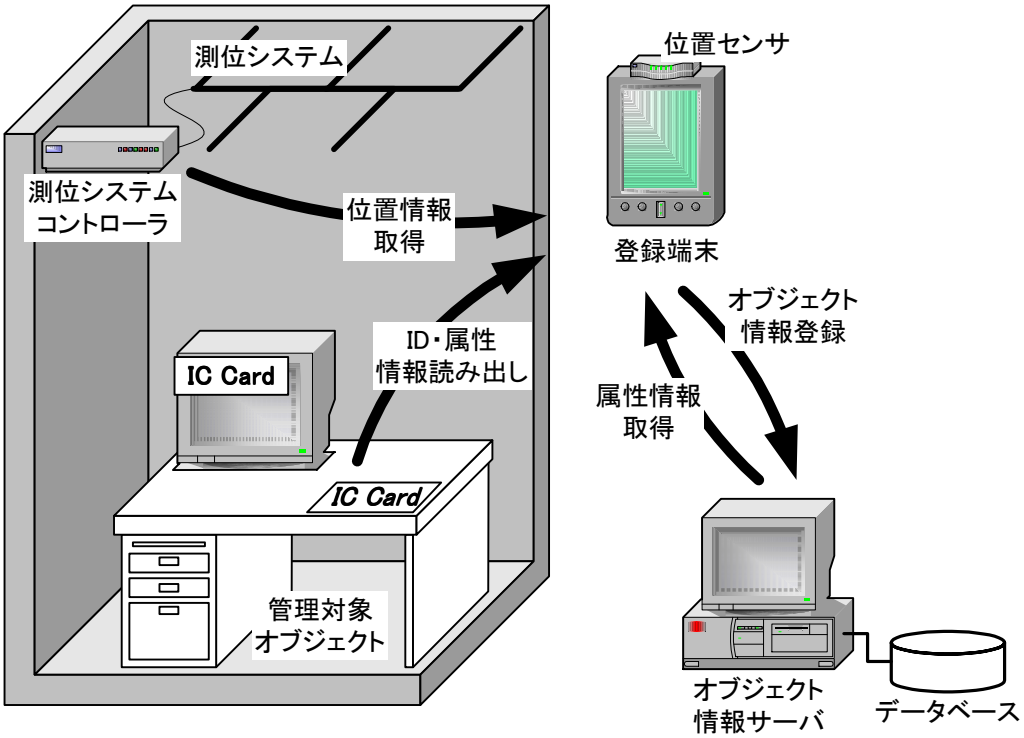


図 5.3-22 システム構成

- 非接触 IC カード:オブジェクトに貼付けた非接触 IC カードは固有の ID 番号を持つ。これに加えてオブジェクトの属性情報(大きさや色や素材など)を必要に応じてカード内に存在するメモリにあらかじめ格納しておくこともできる。一般的な非接触 IC カードは数キロバイト程度のメモリ(EEPROM)を持っており、このようなテキスト情報の格納には十分な容量が

ある。

- 測位システム:測位システムは、三次元空間の位置を測定し取得するための装置である。屋内での利用を想定した場合、超音波の伝搬時間を利用し、位置検出部(位置センサ取り付け部)と複数地点との距離を計測、計算し位置を割り出すシステムなどを利用する。位置を測定するためのセンサ部は登録端末に取りつける。
- 登録端末:オブジェクト情報(オブジェクトの位置情報および属性情報)を取得し、その情報をサーバへ登録するために小型の端末を用いる。この端末は非接触 IC カードのリーダライタを持ち、オブジェクトに貼付された非接触 IC カードから、内部に保持されている情報(固有 ID 、属性情報)を読み出すことが可能である。また前述の通り位置取得のためのセンサを取り付けてあり、本端末の位置情報が取得可能になっている。
- オブジェクト情報サーバ:オブジェクト情報サーバは、端末から送られるオブジェクト情報をデータベースを用いて保管・管理する。オブジェクトの属性情報に加えて、三次元の空間位置情報も扱う必要があるため、位置情報の管理には空間検索が可能なデータベースを使用する。登録端末およびアプリケーションはこのサーバにアクセスしオブジェクト情報を取得する。

なお登録端末とオブジェクト情報サーバ間の通信ではデータを XML (Extensible Markup Language)により符号化することとした。

(2-3) 基本動作

位置情報の取得、登録は次の手順で行なう。

ユーザはオブジェクトに貼付した非接触 IC カードに端末を近づける。この時、端末はカードリーダライタを通して非接触 IC カードと通信を行い、オブジェクトの ID と属性情報を読み出す。

次に端末を利用して測位システムから位置情報(三次元位置・向き情報)の取得を行なう。オブジェクトの位置情報の計測は、オブジェクトのあらかじめ定めた点の空間位置を測定することで行なう。位置情報の取得は次の三通りの方法がある。

(a) 大きさ取得可

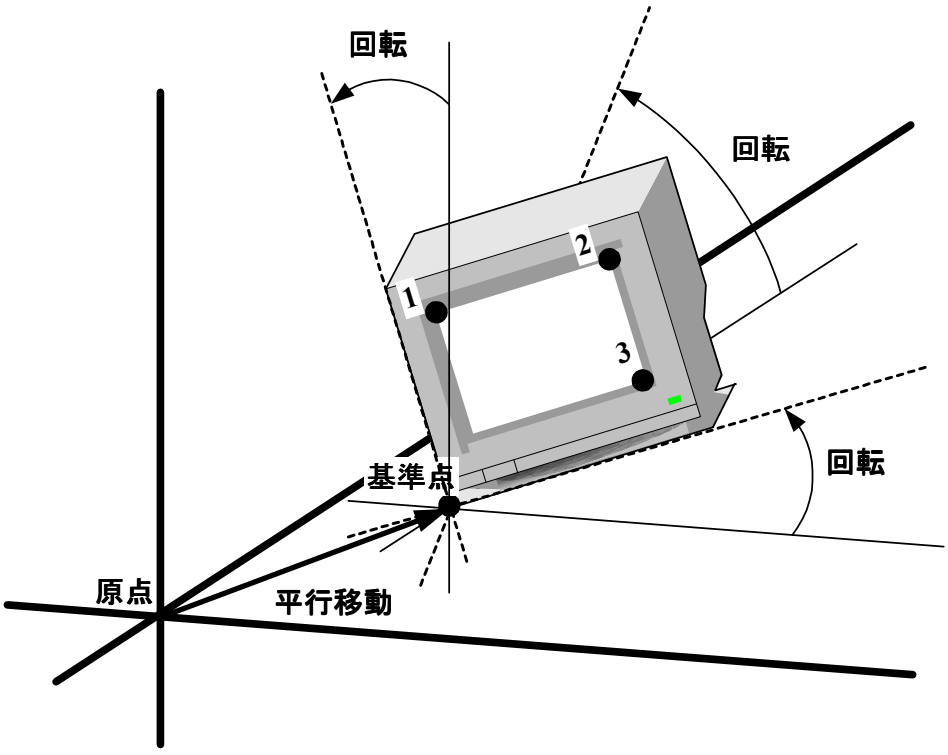
オブジェクトの大きさが取得可能(非接触 IC カードかオブジェクト情報サーバに登録済)の場合には、端末を使用し同一直線上に存在しないオブジェクト上の三点の位置を測定する。これらの三点は、オブジェクト内のある一点(基準点)からのオフセットが既知(あるいは取得可能)であれば良く、必ずしもオブジェクトの隅である必要はない。例えば図 2 のテレビの場合では点 1 から点 3 の三点を測定することで位置情報の取得を行なう。次に測定した空間上の三点の位置とオブジェクト内でのこれら三点の位置関係をもとに計算を行ないオブジェクトの向き(回転行列)を求め、測定点からオブジェクト基準点までのオフセット補正し基準点の座標を得る。このようにして位置を表わす基準点の座標、向きを表わす回転行列が得られることで、オブジェクトの位置は一意に定まる。

(b) 一点測定

前述の方法では、位置情報に複数点の測定が必要だが、オブジェクトの大きさが取得可能で、向きの情報も取得可能であるセンサを利用できる場合には一点のみの測定で済ませることが可能となる。この方法では、オブジェクトの基準点からのオフセットが既知である一点のみを事前に決めておいた方向に端末を向けて測定する。例えば図 5.3-23 のテレビの場合では点 2 のみを測定し、その際に端末を上方(点 3 から点 2 の向き)へ向けて測定する。測定により得られた端末の向きからオブジェクトの向きを計算し、基準点から測定点までのオフセットを補正し、基準点座標と回転行列を得る。

(c)大きさ取得不可

オブジェクトの大きさが IC カードあるいはサーバから取得不可能である場合には、位置情報の取得と合わせて大きさの取得も行なう。この場合は、同一平面上にないオブジェクトの隅四点の位置を測定する。四点の空間位置を測定することで、オブジェクトの位置、向きに加え、幅、高さ、奥行きの情報を得ることが可能となる。



E)　　図 5.3-23 複数点測位による位置情報取得

端末を使用して得られた位置、向き、大きさや属性情報などの全てのオブジェクト情報を XML で記述しオブジェクト情報サーバへ送る。

オブジェクト情報サーバは、端末から送られてきた XML データを解析してデータベースへ登録する。位置情報の格納には空間検索を利用できるデータベースを使用し、残りの情報にはリレーショナルデータベースを利用する。

(2-4) まとめ

本節では、位置情報を取得する対象となる物へ位置センサを取りつけることなく位置・向き情報を取得することが可能にし、物の位置を容易にコンピュータへ入力することを可能する方式を提案した。位置センサの代わりに安価で薄い非接触 IC カードを貼付することで、従来方式に比べてコストを抑え、電力供給の問題も解決した。さらに貼付位置が測定点ではなくなるため取り付けが容易に行なえる。

（３） ユビキタス通信プロトコル（HTTP on XML）

クライアント群（UT、クライアント検索システム、擬似データ生成ツール）と、サーバ群（位置管理サーバ・オブジェクト情報サーバ）間の通信プロトコルは、IP/HTTP/XML を利用することとし、位置管理サーバとオブジェクト情報サーバは HTTP サーバ(例えば Servlet)として動作する。

また、リクエスト（登録、更新、削除、検索）の表現形式、オブジェクトデータ表現形式はいずれも、XML を用いる。

(3-1) インスタンス例

以下に実際の HTTP 通信にて送信される XML 書式を示す。

(a) オブジェクトの登録(1)

(a-1) UT(ユビキタス端末)→オブジェクト情報サーバへの登録

他のサーバ(位置管理サーバ)に登録する情報(位置・向き情報)を除いた情報をオブジェクト情報サーバへ登録。登録する情報には外部サーバ(位置管理サーバ)へのポインタも含む。

オブジェクト情報サーバへの登録時は必ず IndexNum は 0 を指定することとし、サーバ側で番号を割り当てる。(IndexNum を指定して、該当するオブジェクト情報をすべて上書きは行なわない。)

```
<Request type="Register">
  <Object>
    <Ids>
      <ObjectId type=" IndexNum ">0</ObjectId>  サーバで割り当てを行なう際は 0
      <ObjectId type=" Name ">オブジェクトの名前</ObjectId>
      <ObjectId type=" eTRON-ID ">0x13…(中略)...fe</ObjectId>
    </Ids>
    <ObjectClass>Computer.NoteBook</ObjectClass>
    <Transform source=" http://location-server.com/path/to/cmd " />
    <Shape>
      <Box size=" 1.3 0.5 0.9 " />
      <Appearance>
```

```
        <Material diffuseColor=”0.8 0.8 0.8” />
      </Appearance>
    </Shape>
    <Speed level=”low” rate=”0.5” direction=”1.0 0.0 0.0” />
    <Product>
      <VendorName>会社名</VendorName>
      <VendorId>0x1234</VendorId>
      <ProductName>製品名</ProductName>
      <ProductId>0xfedc</ProductId>
      <ProductClass>製品種別</ProductClass>
    </Product>
    <Oinfo ontology=”pc.dtd” >
      <pc_type>NoteBook</pctype>
      <memory>128MB</memory>
    </Oinfo>
    <Oinfo                                    ontology=”TempSensor.dtd”
source=”http://proxy-srv.domain/path/to/cmd” />
source 属性が指定されている場合要素は空となる。
    <Owner>所有者</Owner>
    <Description>説明文</Description>
  </Object>
</Request>
```

(a-2) オブジェクト情報サーバ→UT への(登録に対する)応答

```
<Reply result=”Success” >
  <StatusMessage>応答文</StatusMessage>
  <ObjectId type=”IndexNum”>321</ObjectId>          サーバで割り当てた番号
</Reply>
```

(b) オブジェクトの登録(2)

(b-1) UT→位置管理サーバへの登録

ID 関連と位置、向きのみを位置管理サーバへ登録する。

IndexNum にはオブジェクト情報サーバで割り当てられた番号を使用する。

```
  <Request type="Register">
    <Object>
      <Ids>
        <ObjectId type="IndexNum">321</ObjectId>
        <ObjectId type="Name">オブジェクトの名前</ObjectId>
        <ObjectId type="eTRON-ID">eTRON の ID</ObjectId>
      </Ids>
      <Transform>
```

```
        <Translation>位置を表すパラメータ</Translation>
        <Rotation>向きを表す行列パラメータ</Rotation>
        <LocationSystem type="IS-900"/>
      </Transform>
    </Object>
  </Request>

  (b-2) 位置管理情報サーバ→UT への(登録に対する)応答
  <Reply result="Success">
    <StatusMessage>応答文</StatusMessage>
  </Reply>
```

(c) オブジェクトの更新(1)

(c-1) UT→オブジェクト情報サーバへの更新

サーバでは IndexNum をキーとしてオブジェクトを探し、クライアントから送られてきたタグに対応する部分のみをデータベースに対して更新する。

<Oinfo>タグに関しては同じ属性を持つタグが複数存在することを許すため、更新時に一つでもこのタグが指定されていれば、データベース上の全ての<Oinfo>タグに相当する情報が書き換えられる(上書きされる)。つまり、データベース上にはあるが、更新時に存在しないものは削除される。

```
<Request type="Update">
  <Object>
    <Ids>
      <ObjectId type="IndexNum">321</ObjectId>
    </Ids>
    <Shape>
      <Box size="1.3 0.5 0.9"/>
      <Appeparance>
        <Material diffuseColor="0.0 0.0 0.7"/>          色の変更
      </Apperance>
    </Shape>
    <Oinfo ontology="pc.dtd">
      <pc_type>NoteBook</pc_type>
      <memory>256MB</memory>                            メモリ増設
    </Oinfo>

TempSensor.dtd の ontology をもつ温度情報は存在しないので自動的に削除される。

    <Oinfo ontology="hdd.dtd">                          新たな Oinfo 追加
      <Vendor>IBM</Vendor>
      <Product>IBM-HDD-XXX</Product>
```

```
      <Capacity>100GB</Capacity>
    </Oinfo>
    <Owner/>                                          所有者情報削除
    <Description>変更したい説明文</Description>          説明文変更
  </Object>
</Request>
```

注:<Oinfo>タグの情報を全て削除したい場合は、下記の様に<Oinfo/>だけを指定する。

```
  <Request type=”Update”>
    …
    </Shape>
    <Oinfo/>          全ての Oinfo 情報を削除
    <Owner/>          所有者情報削除
    …
  </Request>
```

(c-2) オブジェクト情報サーバ→UT への(更新に対する)応答

```
<Reply result=”Success”>
  <StatusMessage>更新成功の応答文</StatusMessage>
</Reply>
```

(d) オブジェクトの更新(2)

(d-1) UT→位置管理サーバへの更新

基本的にはオブジェクト情報サーバへの更新と同じ。但し送る情報は位置管理サーバで管理する情報に関するタグのみ。

```
<Request type=”Update”>
  <Object>
    <Ids>
      <ObjectId type=”IndexNum”>321</ObjectId>
    </Ids>
    <Transform>
      <Translation>移動後位置を表すパラメータ</Translation>
      <Rotation>移動後の向きを表す行列パラメータ</Rotation>
    </Transform>
  </Object>
</Request>
```

(d-2) 位置管理サーバ→UT への(更新に対する)応答

```
<Reply result=”Success”>
  <StatusMessage>更新成功の応答文</StatusMessage>
</Reply>
```

(e) オブジェクトの検索(1)

<TargetId>により検索に利用するキー(<ObjectId>タグ)を指定する。<Retrieve>タグで検索結果として得るタグの名前を指定する。

(e-1) case1: IndexNum を指定してすべての情報(<Object>タグの要素)を得る場合

(e-1-1) UT→オブジェクト情報サーバへの検索

```
<Request type=”Query”>
  <TargetId>
    <ObjectId type=”IndexNum”>321</ObjectId>
  </TargetId>
  <Retrieve>Object</Retrieve>
</Request>
```

(e-1-2) オブジェクト情報サーバ→UT への(検索に対する)応答

```
<Reply status=”Success”>
  <StatusMessage>検索成功の応答文</StatusMessage>
  <Object>
    <Ids>
      <ObjectId type=”IndexNum”>0</ObjectId>  サーバで割り当てを行なう際は 0
      <ObjectId type=”Name”>オブジェクトの名前</ObjectId>
      <ObjectId type=”eTRON-ID”>0x13…(中略)...fe</ObjectId>
    </Ids>
    <ObjectClass>Computer.NoteBook</ObjectClass>
    <TimeInfo>                    サーバからの応答時には時刻に関する情報も含まれる
      <CreateTime>2002-12-19T15:21:18</CreateTime>
      <ModifiedTime>2002-12-20T17:19:38</ModifiedTime>
    </TimeInfo>
    <Transform source=”http://location-server.com/path/to/cmd” />
    <Shape>
      <Box size=”1.3 0.5 0.9” />
      <Appearance>
        <Material diffuseColor=”0.8 0.8 0.8” />
      </Appearance>
    </Shape>
    <Speed level=”low”  rate=”0.5”  direction=”1.0 0.0 0.0” />
    <Product>
      <VendorName>会社名</VendorName>
      <VendorId>0x1234</VendorId>
      <ProductName>製品名</ProductName>
      <ProductId>0xfedc</ProductId>
      <ProductClass>製品種別</ProductClass>
```

```
    </Product>
    <Oinfo ontology=" pc.dtd" >
      <pc_type>NoteBook</pctype>
      <memory>128MB</memory>
    </Oinfo>
    <Oinfo                                    ontology=" TempSensor.dtd"
source=" http://proxy-srv.domain/path/to/cmd" />
    </Oinfo>
    <Owner>所有者</Owner>
    <Description>説明文</Description>
  </Object>
</Reply>
```

(e-2) case2: eTRON ID を指定して IndexNum を得る場合

(e-2-1) UT→オブジェクト情報サーバへの検索

```
<Request type=" Query" >
  <TargetId>
    <ObjectId type=" eTRON-ID" >0x23..(中略)..f1</ObjectId>
  </TargetId>
  <Retrieve>Ids</Retrieve>
</Request>
```

(e-2-2) オブジェクト情報サーバ→UT への(検索に対する)応答

```
<Reply status=" Success" >
  <StatusMessage>検索成功の応答文</StatusMessage>
  <Ids>
    <ObjectId type=" IndexNum" >32</ObjectId>
    <ObjectId type=" Name" >オブジェクトの名前</ObjectId>
    <ObjectId type=" eTRON-ID" >0x23..(中略)..f1</ObjectId>
  </Ids>
</Reply>
```

(f) オブジェクトの検索(2)

(f-1) UT→位置管理サーバへの検索

現時点での仕様では、ObjectId の IndexNum のみでの検索をサポートする。

```
<Request type=" Query" >
  <TargetId>
    <ObjectId type=" IndexNum" >321</ObjectId>
  </TargetId>
  <Retrieve>Object</Retrieve>
</Request>
```

(f-2) 位置管理サーバ→UT への(検索に対する)応答

```
<Reply status=" Success" >
  <StatusMessage>検索成功の応答文</StatusMessage>
  <Object>
    <Ids>
      <ObjectId type=" IndexNum" >321</ObjectId>
      <ObjectId type=" Name" >オブジェクトの名前</ObjectId>
      <ObjectId type=" eTRON-ID" >eTRON の ID</ObjectId>
    </Ids>
    <Transform>
      <Translation>位置を表すパラメータ</Translation>
      <Rotation>向きを表す行列パラメータ</Rotation>
      <LocationSystem type=" IS-900" />
    </Transform>
  </Object>
</Reply>
```

(g) オブジェクトの削除

(g-1) UT→オブジェクト情報サーバ(位置管理サーバ)への削除要求

ObjectId の IndexNum を指定して、各サーバで保持している情報のうち削除したいものを指定する。現仕様では、IndexNum による指定のみをサポートする。

```
<Request type=" Delete" >
  <Object>
    <Ids>
      <ObjectId type=" IndexNum" >321</ObjectId>
    </Ids>
  </Object>
</Request>
```

(g-2) オブジェクト情報サーバ(位置管理サーバ)→UT への(削除要求に対する)応答

```
<Reply result=" Success" >
  <StatusMessage>削除成功の応答文</StatusMessage>
</Reply>
```

(3-2) DTD に基づくタグ定義

<!ELEMENT Request (Object?, TargetId?, Retrieve?)>

属性: type(必須)

要素: <Object>(登録、更新、削除時は必須),
<TargetId>と<Retrieve>(検索時必須)

クライアントからサーバに送るリクエストに使用する。

<Request>タグの要素としては、検索時は<TargetId>と<Retrieve>を必ず含むが、それ以外では<Object>タグを必ず含む。

<!ATTLIST Request
type (Register | Update | Delete | Query) #REQUIRED>
type 属性では要求の種別を表す下記のいずれかを必ず指定する。
Register: 登録
Update: 更新
Delete: 削除
Query: 検索

<!ELEMENT Object (Ids, ObjectClass?, TimeInfo?, Transform?, Shape?, Speed?, Product?, Oinfo*, Owner?, Description?)>
属性: なし
要素: <Ids>(必須)
その他のタグは利用に応じて使用。
オブジェクトを表現するためのタグ。
<Ids>タグを必須とするが、他のタグはリクエストの種類などに応じて使用する。

<!ELEMENT Ids (ObjectId+)>
属性: なし
要素: 一つ以上の<ObjectId> (必須)
オブジェクト ID を表現するためのタグ。要素として一つ以上の<ObjectId>タグを持つ。このタグの属性として type を持ち、要素として ObjectID を構成する識別子を持つ。

<!ELEMENT ObjectId (#PCDATA)>
属性: type (必須)
要素: type に応じた識別子(ID)の値
オブジェクトの識別子を表すタグ。type 属性でどのような種類の識別子かを表し、実際の値は要素として表す。異なる type 属性を持つ<ObjectId>タグは複数存在してもよい。

<!ATTLIST ObjectId
type (IndexNum | Name | eTRON-ID) #REQUIRED>
type はオブジェクトの識別子(ID)の種別を指定するため属性。この属性は下記の値のいずれかを必ず持つ。
IndexNum: サーバが割り当てた固有の値。十進数で表記する。
Name: オブジェクトの名前。文字列。
eTRON-ID: オブジェクトに取り付けた eTRON カードに格納されている ID。”0x”で始まる 16 進数で表記し、英字は小文字で表す。
有効な IndexNum の値は 1 以上の整数とする。但し、クライアントからサーバへデータを登録する場合において、サーバにより IndexNum の値を新たに割り当てる事を要求する場合のみ IndexNum は 0 とする。
各サーバ(オブジェクト情報サーバと位置管理サーバ)は、情報の登録時に

eTRON-ID が指定されている場合には重複チェックを行い、重複している場合にはサーバからの応答時に<StatusMessage>タグにエラーとなった情報の IndexNum の値も表示する。

例:

```
<ObjectId type=”IndexNum”>12</ObjectId> サーバ割り振る識別子
<ObjectId type=”Name”>名前</ObjectId>オブジェクトにつけた名前
<ObjectId type=”eTRON-ID”>0x1234…def</ObjectId>eTRON ID
```

<!ELEMENT ObjectClass (#PCDATA)>
オブジェクトの種別(テーブルかイスかセンサーか…)を表現するためのタグ。要素としてオブジェクトの種別を表す文字列を持つ。データベースには文字列をそのまま格納する。
文字列は、下記の例のようにクラス、サブクラス… を「.」でつないだものを使用する。
例:
　　LM92 による気温センサ
<ObjectClass>Sensor.Temperature.LM92</ObjectClass>
円卓
<ObjectClass>Table.RoundTable</ObjectClass>

<!ELEMENT TimeInfo (CreateTime, ModifiedTime)>
属性: なし
要素: <CreateTime>(必須), <ModifiedTime>(必須)
オブジェクトの情報が作成や更新された時刻を表わすタグ。クライアントからこのタグが送られることはなく、サーバが検索に対する応答を行なうときのみ使用される。サーバでは、新規にオブジェクト情報を作成した際に<CreateTime>に対応する情報を、更新された際には<ModifiedTime>に対応する情報をデータベースに対して更新する。要素として<CreateTime>と<ModifiedTime>を持つ。

<!ELEMENT CreateTime (#PCDATA)>
属性: なし
要素: 時刻情報
オブジェクトの情報が作成された時刻を表すタグ。要素としてオブジェクトの情報が作成された時刻を保持する。形式は、CCYY-MM-DDThh:mm:ss の形とする。時刻情報にはタイムゾーンは含めずに UTC を利用する。
参考: http://www.w3.org/TR/xmlschema-2/#dateTime
例:
　　<CreateTime>2002-12-19T15:21:18</CreateTime>

<!ELEMENT ModifiedTime (#PCDATA)>
属性: なし
要素: 時刻情報

オブジェクトの情報が更新された時刻を表すタグ。要素としてオブジェクトの情報が更新された最終時刻を保持する。形式は<CreateTime>と同じ。
オブジェクトの情報が作成された際には、<ModifiedTime>は<CreateTime>と同じ値となる。

<!ELEMENT Transform (Translation?, Rotation?, LocationSystem?)>
属性: source
要素: <Translation>, <Rotation>
オブジェクトの位置や向きを表現するためのタグ。
要素として、<Translation>タグと<Rotation>タグをもつ事ができる。外部(別なサーバ)に位置・向き情報を格納している場合や、デフォルトの値の時には要素は空となる。

<!ATTLIST Transform
source CDATA #IMPLIED>
位置・向き情報が外部のサーバに格納されている場合には source 属性を使用して、格納しているサーバを表現する。source 属性を指定している場合には、<Transform>の要素は空となる。

<!ELEMENT Translation (#PCDATA)>
属性: accuracy
要素: 三次元位置情報
オブジェクトの位置を三次元で表現するためのタグ。要素にはオブジェクト座標系での基準点(原点)を、表示系のどこに位置するかを表わす三次元の座標情報(位置情報)を持つ。値は表示用座標系における基準点(原点)からの位置を(x, y, z)の順にメートル単位での数値(実数)で表わし、省略時は"0 0 0"とみなす。
オブジェクトの座標系、表示用の座標系の定義は別途定める。
例:
<Translation>1.3 2.3 0.9</Translation>

<!ATTLIST Translation
accuracy CDATA #IMPLIED>
accuracy 属性は位置情報の精度を表す。

<!ELEMENT Rotation (#PCDATA)>
属性: なし
要素: 回転情報
オブジェクトの回転を表現するためのタグ。要素にはオブジェクトの座標系(i, j, k)から、表示用の座標系(x, y, z)への変換に必要な行列の情報を表現する。行列の定義については別途定める。
例:
<Rotation>1 0 0 0 1 0 0 0 1</Rotation>
<Rotation>x_i x_j x_k y_i y_j y_k z_i z_j z_k</Rotation>

<!ELEMENT LocationSystem EMPTY>
属性: type(必須)
要素: なし
位置情報取得に用いた測位システムに関する情報を表現するタグ。
(現在は、type 属性で製品情報のみを格納するが、将来的には測位システムの属性情報なども含める為の検討が必要)

<!ATTLIST
type CDATA #IMPLIED>
type 属性は測位システムの種別を表わす。種別には測位システムの名前を文字列で格納する。現在とりうる値は”IS-900”のみ。
(将来的には、GPS や超音波などの大まかな種別と、使用製品などの細かい分類に分けるなどの検討が必要)
例:
<LocationSystem type=”IS-900”/>

<!ELEMENT Shape ((Box | Cylinder | Cone | Sphere | BBox | Chair | Table | Book | PDA | TrashBox | SensorNode)?,
Appearance?)>
属性: なし
要素:
形状をあらわすタグ<Box>, <Cylinder>, <Cone>, <Sphere>, <BBox>, <Chair>, <Table>, <Book>, <PDA>, <TrashBox>, <SensorNode> のうち、いずれかひとつ
<Appearance>
オブジェクトの見た目を表現するためのタグ。要素として次のいずれかのタグを持つ。
Box: 直方体
Cylinder: 円柱
Cone: 円錐
Sphere: 球
BBox: バウンディングボックスによる図形
Chair: 椅子
Table: テーブル
Book: 本
PDA: PDA(携帯型端末)
TrashBox: ごみ箱
SensorNode: センサノード
※現仕様では、複数の形状をグループ化した複合形状はサポートしない。
※椅子やテーブルなどのあらかじめモデルデータを作っておくオブジェクトは上記の複数の基本形状(直方体や円柱など)の組み合わせとしてオブジェクト情報サーバには登録は行なわず、モデルデータはユーザインタフェース PC が保持する。

また、オブジェクトの表面を表わすタグ<Appearance>タグを要素として持つ。
例:

```
<Shape>
  <Box size=" 1.0 0.5 0.5" />
  <Appearance>
    <Material diffuseColor=" 0.8 0.8 0.8" />
  </Appearance>
</Shape>
```

<!ELEMENT Box EMPTY>
属性: size
要素: なし
直方体を表示するためのタグ。

<!ATTLIST Box
size　　CDATA　　"1 1 1">
属性 size は直方体の大きさを表わす。属性値は順に、i, j, k 軸方向の大きさを各々表わし、原点は直方体の中心部となる。単位はメートルで正の実数を用いて表現する。
例:

```
<Box size=" 0.2 0.3 0.1" />
```

<Box size=" ***Size_i Size_j Size_k*** " /> (図 5.3-24 参照)

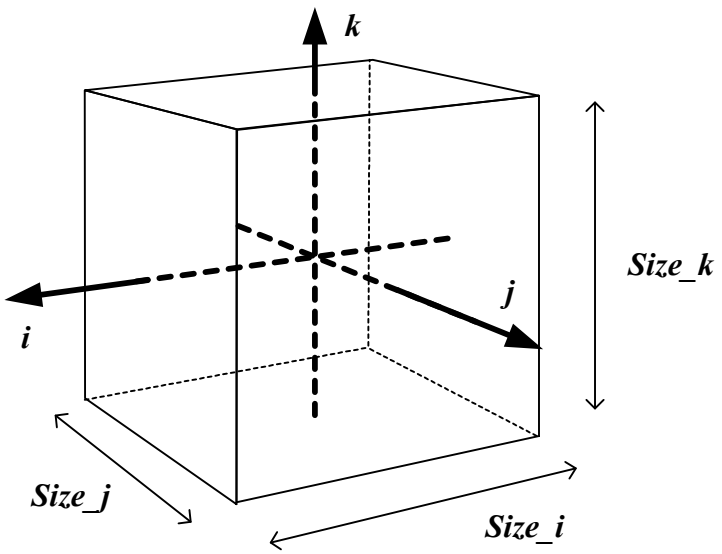


図 5.3-24　直方体

<!ELEMENT Cylinder EMPTY>
属性: height(正の実数), radius(正の実数)
要素: なし
円柱を表示するためのタグ。

<!ATTLIST Cylinder
height CDATA　"1"
radius CDATA　"1">
height 属性は高さを radius 属性は底面の円の半径を表す。省略時には 1 となる。height は円柱の k 軸方向の大きさ(高さ)を表わす。原点は円柱の中心(上下の円の中心を結ぶ線分を等分に分ける点)を原点とする。単位はメートルで正の実数を用いて表現する。
例:

<Cylinder height=”2”　radius=”3”/>
<Cylinder height=”*h*”　radius=”*r*”/>　(図 5.3-25 参照)

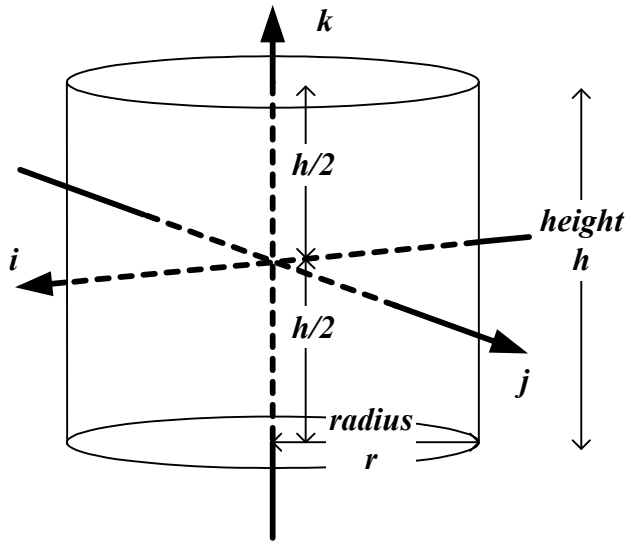


図 5.3-25　円柱

<!ELEMENT Cone EMPTY>
属性: bottomRadius(実数), height(実数)
要素: なし
円錐を表示するためのタグ。

<!ATTLIST Cone
bottomRadius　CDATA　"1"
height　　CDATA　"1">
height は高さを、bottomRadius は底面の円の半径を表す。省略時には 1 となる。height は円錐の k 軸方向の大きさ(高さ)を表す。原点は円柱の中心(底面の円の中心と頂点を結ぶ線分を等分に分ける点)を原点とする。単位はメートルで正の実数を用いて表す。
例:

<Cone height=”1.3”　bottomRadius=”0.5”/>
<Cone height=”*h*”　bottomRadius=”*r*”/>　(図 5.3-26 参照)

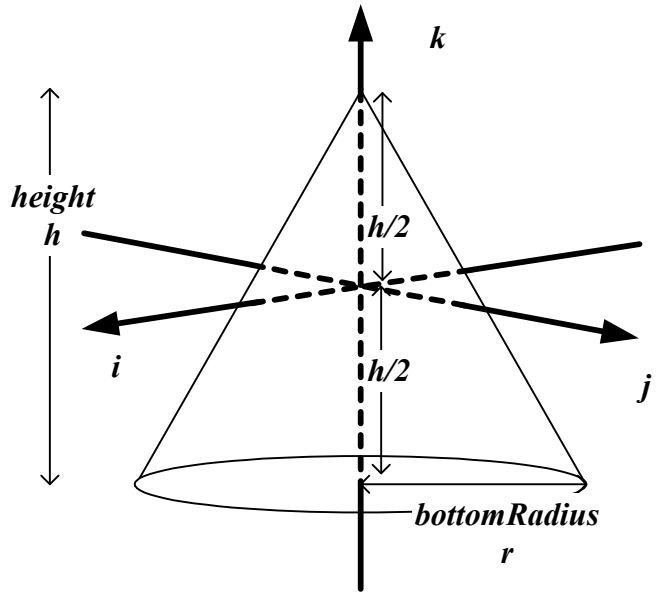


図 5.3-26 円錐

<!ELEMENT Sphere EMPTY>
属性: radius(実数)
要素: なし
球を表示するためのタグ。

<!ATTLIST Sphere
radius CDATA "1">
radius 属性は球の半径を表す。省略時には 1 となる。単位はメートルで正の実数を用いて表現する。このタグは原点から半径 radius の球を描く。
例:
<Sphere radius=”0.5”/>
<Sphere radius=”r”/”> (図 5.3-27 参照)

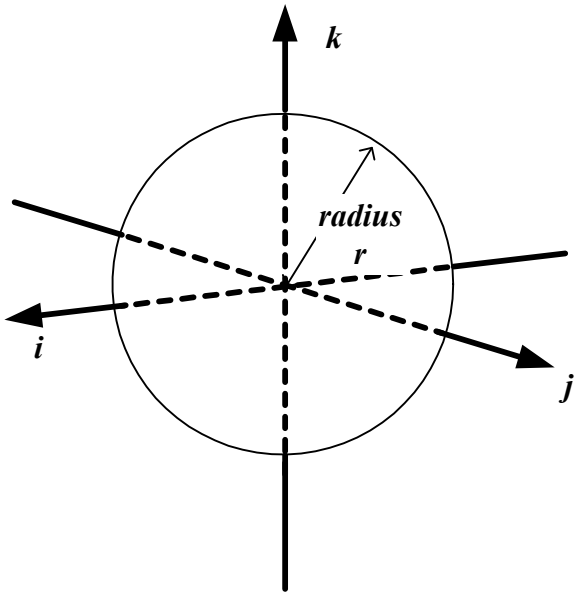


図 5.3-27　円

<!ELEMENT BBox EMPTY>
属性: size(実数), type
要素: なし
バウンディングボックスにより囲まれる図形を描画するためのタグ。サポートする図形としては、直方体、円柱、円錐、球。

<!ATTLIST BBox
size　　CDATA　　"1 1 1"
type　(Box | Cylinder | Cone | Sphere)　#REQUIRED>
size 属性は図形に外接する直方体(バウンディングボックス)の大きさを表わす。値のとり方は<Box>タグの size 属性と同様。また type 属性はバウンディングボックスに内接する図形の形状を表し、次のいずれかを取る。
Box:　直方体。<Box>タグでの描画と等しくなる。
Cylinder:　バウンディングボックスに内接する円柱。
Cone: バウンディングボックスに内接する円錐。
Sphere:　バウンディングボックスに内接する球。
例:
　　<BBox size=”0.5 0.8 0.7”　type=”Cylinder”/>
　　<BBox size=”***Size_i Size_j Size_k***”　type=”Cone”/>　(図 5.3-28 参照)

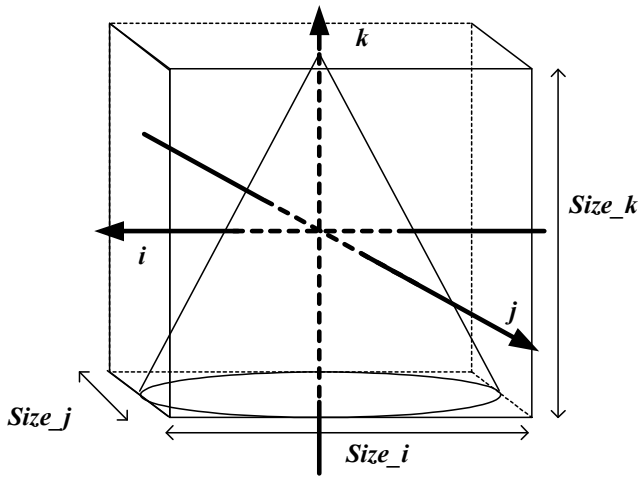


図 5.3-28 バウンディングボックスに内接する図形(円錐)

<!ELEMENT Chair EMPTY>
属性: type(文字列), size(実数)
要素: なし
椅子を表示するためのタグ。

<!ATTLIST Chair
type CDATA “1”
size CDATA #IMPLIED>
type 属性でイスの形状種類を表し、size 属性でイスが外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面は *j* 軸方向とする。
type の値と椅子の種類は下記の表に示すとおり。椅子の詳細形状については別途定める。
表 5.3-12

type	種類
1	座面、背もたれが木で出来ており、ひじ置きがない椅子。色は青。
2	布張りでひじ置きがなく、キャスターがついている椅子。座面が青緑色、背もたれが黒色。
その他	予約

例:
<Chair type=”1” size=”0.7 0.7 1.2”/>
<Chair type=”1” size=”*Size_i Size_j Size_k*”/> (図 5.3-29 参照)

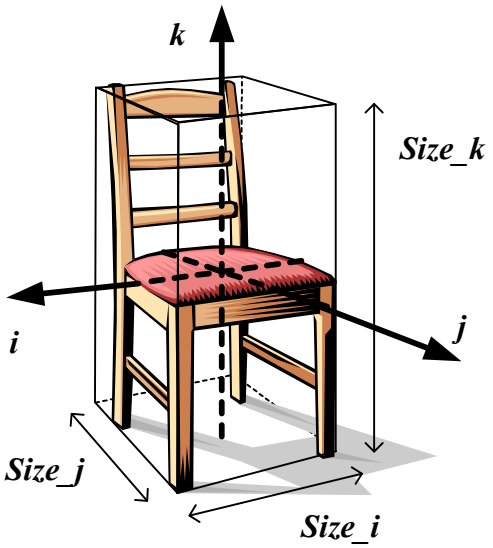


図 5.3-29　イス（注：実際のイスとは色や形状が異なる）

<!ELEMENT Table EMPTY>
属性: type(文字列), size(実数)
要素: なし
テーブルを表示するためのタグ。

<!ATTLIST Table
type　CDATA　　“1”
size　　CDATA　　#IMPLIED>
type 属性でテーブルの形状種類を表し、size 属性でテーブルが外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面は *j* 軸方向とする。
type の値とテーブルの種類は下記の表に示すとおり。テーブルの詳細形状については別途定める。
表 5.3-13

type	種類
1	天板が円状の机。天板の材質は木でできている。
2	長机。天板の材質は化粧板(白)で出来ている。
その他	予約

例:

<Desk size=”1.8 0.9 0.9” />
<Desk type=”1” size=”***Size_i Size_j Size_k***” /> （図 5.3-30 参照）

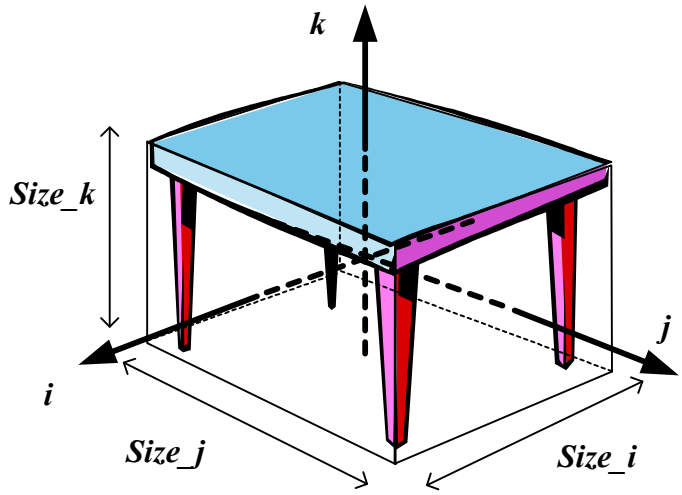


図 5.3-30　テーブル　(注：実際のテーブルとは色や形状は異なる)

<!ELEMENT Book EMPTY>
属性: type(その他), size(実数)
要素: なし
本の形状を表すためのタグ。

<!ATTLIST Book
type　CDATA　　"1"
size　　CDATA　　#IMPLIED>
type 属性で本の形状種類を表し、size 属性で本が外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面(表紙)は *j* 軸方向とする。
type としては現仕様では数種類を想定するが、その値と本の種類・詳細形状については別途定める。

例:
　　<Book size="0.18 0.01 0.26" />
　　<Book type="3"　size="*Size_i Size_j Size_k*" />　(図 5.3-31 参照)

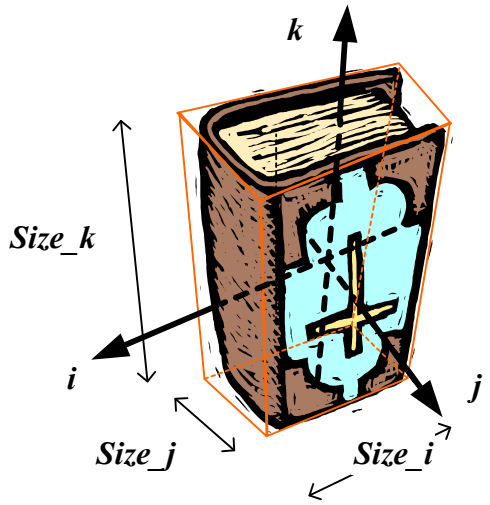


図 5.3-31　本　（注：実際の本とは色や形状は異なる）

<!ELEMENT PDA EMPTY>
属性: type(文字列), size(実数)
要素: なし
PDA 状の携帯型端末の形状を表すためのタグ。

<!ATTLIST PDA
type　CDATA　　“1”
size　　CDATA　　#IMPLIED>
type 属性で PDA の形状種類を表し、size 属性で PDA が外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面は *j* 軸方向とする。
現仕様では type は 1 のみを使用し、その他は予約とする。詳細形状については別途定める。

例:
　　<PDA size=” 0.08 0.015 0.13” />
　　<PDA type=” 1”　size=” ***Size_i Size_j Size_k***” />　（図 5.3-32 参照）

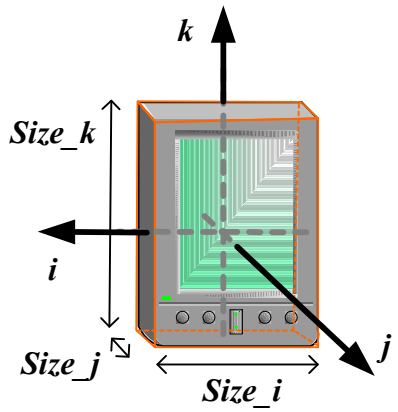


図 5.3-32　PDA　（注：実際の PDA とは色や形状は異なる）

<!ELEMENT TrashBox EMPTY>
属性: type(文字列), size(実数)
要素: なし
ごみ箱の形状を表すためのタグ。

<!ATTLIST TrushBox
type　CDATA　　“1”
size　　CDATA　　#IMPLIED>
type 属性でごみ箱の形状種類を表し、size 属性でごみ箱が外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面は *j* 軸方向とする。
現仕様では type は 1 のみを使用し、その他は予約とする。詳細形状については別途定める。

例:
　　<TrashBox size=”0.3 0.3 0.4” />
　　<TrashBox type=”1” size=”*Size_i Size_j Size_k*” /> （図 5.3-33 参照）

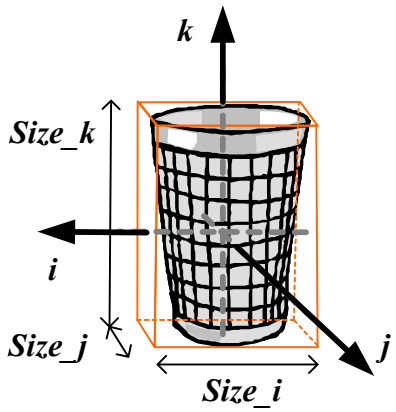


図 5.3-33　ごみ箱　（注：実際のごみ箱とは色や形状は異なる）

<!ELEMENT SensorNode EMPTY>
属性: type(文字列), size(実数)
要素: なし
センサノードの形状を表すためのタグ。

<!ATTLIST SensorNode
type　CDATA　　“1”
size　　CDATA　　#IMPLIED>
type 属性でセンサノードの形状種類を表し、size 属性でセンサノードが外接する直方体のサイズを表す。type 属性は文字列で表し省略された場合には、1 が指定されているとみなす。size 属性の表現形式は<Box>タグの size 属性と同じ形式を使用する。正面は *j* 軸方向とする。
現仕様では type は 1 のみを使用し、その他は予約とする。詳細形状については別途定める。
type の値とセンサノードの形状種類は下記の表に示すとおり。詳細形状については別途定める。
表 5.3-14

type	種類
1	温度センサノード。
その他	予約

例:

<PDA size=” 0.3 0.3 0.4” />
<PDA type=” 1”　size=” ***Size_i Size_j Size_k***” />

<!ELEMENT Appearance (Material?)>
属性: なし
要素: <Material>
オブジェクトの表面を表わすタグとして<Appearance>タグを使用する。

要素としては、色などを表わす<Material>タグをとり得る。

<!ELEMENT Material EMPTY>
属性: diffuseColor
要素: なし
オブジェクトの表面の色などを表わすタグ。

<!ATTLIST Material
diffuseColor CDATA "0.8 0.8 0.8">
diffuseColor 属性はオブジェクトの表面の色を表わす。値は R, G, B の順に表現し、各色は 0 から 1 の間の実数を使用し表現する。省略時の値は上記の通り。
例:
<Material diffuseColor="0.3 0.5 0.9" />
<Material diffuseColor="*R G B*" />

<!ELEMENT Speed EMPTY>
属性: level, rate(実数)
オブジェクトの移動可否、速度に関する情報を表わすタグ。

<!ATTLIST
level (high | middle | low | fixed) #IMPLIED
rate CDATA #IMPLIED
direction CDATA #IMPLIED>
level 属性はオブジェクトの大まかな移動速度を表わす。可能であれば rate 属性を使用して具体的な速度を direction 属性を使用して移動方向の単位ベクトルを表わす。level 属性の high は移動速度が速い物を表わし、low は遅い物、middle はその中間を表わす。また fixed は移動しない物を表現する。rate の値は実数を用い、単位は m/sec とする。direction の値は単位ベクトルを実数を用いて"*x y z*"軸方向の成分の順に表現する。

<!ELEMENT Product (VendorName, VendorID, ProductName, ProductID, ProductClass)>
属性: なし
要素: <VendorName>, <VendorID>, <ProductName>, <ProductID>, <ProductClass>
オブジェクトの製品元に関する情報を表わすタグ。要素として、製造会社(VendorName)、製品名(ProductName)を表わすタグと、それらを数値で管理した値(VendorID, ProductID)を持つ。また、オブジェクトの種別を表わすタグとして ProductClass を持つ。
例:
<Product>
<VendorName>株式会社 KDDI 研究所</VendorName>

```
    <VendorID>0x011d</VendorID>
    <ProductName>パソコン用机 PC-Desk-1</ProductName>
    <ProductID>0x05</ProductID>
    <ProductClass>Desk</ProductClass>
  </Product>
```

<!ELEMENT VendorName (#PCDATA)>
属性: なし
要素: 製造会社名(文字列)
オブジェクトの製造元を表わすタグ。要素として、製造会社を表わす文字列を持つ。

<!ELEMENT VendorID (#PCDATA)>
属性: なし
要素: 製造会社番号(16 進数の数値)
オブジェクトの製造元を表わすタグ。要素として、製造会社を表わす数値(16 進数)を持つ。値は 0x から始まる 16 進数で表記し、英字は小文字で表す。番号の割り当ては別途定義する。

<!ELEMENT ProductName (#PCDATA)>
属性: なし
要素: 製品名(文字列)
オブジェクトの製品名を表わすタグ。要素として、製品名を表わす文字列を持つ。

<!ELEMENT ProductID (#PCDATA)>
属性: なし
要素: 製品番号(16 進数数値)
オブジェクトの製品を番号で表わすタグ。要素として、製品名を表わす数値(16 進数)を持つ。値は 0x から始まる 16 進数で表記し、英字は小文字で表す。番号は製造会社ごとに固有の値となる。番号の割り当ては別途定義する。

<!ELEMENT ProductClass (#PCDATA)>
属性: なし
要素: オブジェクト種別(文字列)
オブジェクトの種別を表わすタグ。要素は文字列(ASCII)で表わす。
例:
<ProductClass>Chair</ProductClass> 椅子
<ProductClass>Desk</ProductClass> 机
<ProductClass>Book</ProductClass> 本

<!ELEMENT Oinfo (#PCDATA)>
属性: ontology(DTD を指定する文字列。必須), source(文字列)
要素: source 属性が指定されていない場合には ontology 属性で指定された形式をもつオブジェクト種別固有の属性情報。source 属性が存在する場合は空。
オブジェクト毎に固有の属性を表わすためのタグ。source 属性が指定されていない場合には要素としてオブジェクト毎に特有の属性を表現する XML 形式のデータを持つが、soruce 属性が指定されている場合は空となる。この要素を解釈するための DTD は ontology 属性に指定される。ontology 属性の異なる複数の<Oinfo>タグを持つことが可能。
このタグを扱う際に ontology 属性で未知の DTD を指定された場合には、XML で書かれている要素を単なる文字列として扱う。

<!ATTLIST Oinfo
ontology CDATA #REQUIRED
source CDATA #IMPLIED>
属性 ontology は<Oinfo>タグの要素の DTD を表わす URI。要素を外部のサーバに保持する場合には source 属性を使用してサーバの URI を表現する。source 属性が指定された場合には、タグの要素は空となる。
例:

```
<Oinfo ontology=” http://object-info.domain/DTDs/PC.dtd” >
  <PCType>NoteBook</PCType>
  <Memory>256MB</Memory>
</Oinfo>
<Oinfo ontology=” http://object-info.domain/DTDs/TempSensor.dtd”
       source=” http://proxy-server.domain/path/to/cmd” />
</Oinfo>
```

<!ELEMENT Owner (#PCDATA)>
属性: なし
要素: 所有者を表す文字列
オブジェクトの所有者を表わすタグ。要素として、所有者名を文字列で表わしたものを持つ。
<!ELEMENT Description (#PCDATA)>

属性: なし
要素: 説明文の文字列
オブジェクトの説明文章を表わすタグ。要素としてオブジェクトに関する説明の文字列を持つ。

<!ELEMENT TargetId (ObjectId)>
属性: なし
要素: <ObjectId>(必須)
検索時に、検索のキーとして使用するオブジェクト ID を指定するためのタグ。要素に検索対象のオブジェクト ID をもつ<ObjectId>タグを指定する。
例:

```
<TargetId>
  <ObjectId type=”eTRON-ID”>0x11…(中略)…ff</ObjectId>
</TargetId>
```

<!ELEMENT Retrieve (#PCDATA)>
属性: なし
要素: タグ名
検索時に、クライアントが検索結果として受け取りたいタグを指定するためのタグ。要素としては<Object>タグとその要素にあたるタグの名前が入る。指定できるのは一つのみで、サーバは指定されたタグ(と存在すればその子要素)を検索結果として返す。
例:
<Retrieve>Object</Retrieve> 全情報を欲しい場合には<Object>タグを指定
<Retrieve>Trnasform</Retrieve> 位置・向き情報だけ欲しい場合

<!ELEMENT Reply (StatusMessage, Object?,
Ids?, ObjectId?, TimeInfo?, CreateTime?, ModifiedTime?, Transform?, Translation?, Rotation?, LocationSystem?, Shape?, Box?, Cylinder?, Cone?, Shpere?, BBox?, Chair?, Table?, Appearance?, Material?, Speed?, Product?, VendorName?, VendorId?, ProductName?, ProductId?, ProductClass?, Oinfo*, Owner?, Description?)>
属性: result
要素: <StatusMessage>(必須), <Object>とそれに含まれるタグ
サーバがクライアントからの要求に対して応答する際に用いるタグ。要素として<StatusMessag>タグを必ず持つ。
データを登録する際にサーバが新規にインデックス番号(<ObjectId type=”IndexNum”>)を割り当てた場合には、<ObjectId>タグ(必ず type=”IndexNum”の属性)を用いてその番号を表す。
検索に対する応答には、検索要求に指定されたタグ(<Object>タグかそれに含まれうるタグ)を持つ。
更新と削除に対する応答では<StatusMessage>タグ以外のタグを持たない。

<!ATTLIST Reply
result (Success | Failed)　#REQUIRED>
result 属性として、処理結果を表わす値を持つ。Success は成功を、Failed が何らかの理由で失敗した事を表わす。

<!ELEMENT StatusMessage (#PCDATA)>
<StatusMessage>タグはサーバでの処理結果を表わす、付加的な文字列。主にクライアントを操作するユーザ(人間)に対してわかりやすく結果を伝えるために使用する。
失敗時にはその原因を人間に対してわかりやすい文字列で表現する。各サーバ(オブジェクト情報サーバと位置管理サーバ)は、情報の登録時に eTRON-ID が指定されている場合には重複チェックを行い、重複している場合にはサーバからの応答時に<StatusMessage>タグにエラーとなった情報の IndexNum の値も表示する。

(3-3) オブジェクトの回転 (<Rotation> タグの要素) について
各オブジェクトにおける座標系(***i, j, k***)からグローバル座標系(表示用座標系)(***x, y, z***)へ変換する際のオブジェクトの回転を表現する行列 R 次のように表される。

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{R}\begin{pmatrix} i \\ j \\ k \end{pmatrix} \quad (\text{但し } \det \mathbf{R} = 1)$$

$$= \begin{pmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{pmatrix}\begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

行列の各成分の意味は次のようになる。

x_i	***i*** 軸単位ベクトルのグローバル座標系における ***x*** 成分
x_j	***j*** 軸単位ベクトルのグローバル座標系における ***x*** 成分
x_k	***k*** 軸単位ベクトルのグローバル座標系における ***x*** 成分
y_i	***i*** 軸単位ベクトルのグローバル座標系における ***y*** 成分
y_j	***j*** 軸単位ベクトルのグローバル座標系における ***y*** 成分
y_k	***k*** 軸単位ベクトルのグローバル座標系における ***y*** 成分
z_i	***i*** 軸単位ベクトルのグローバル座標系における ***z*** 成分
z_j	***j*** 軸単位ベクトルのグローバル座標系における ***z*** 成分
z_k	***k*** 軸単位ベクトルのグローバル座標系における ***z*** 成分

この行列 R を Rotation タグで表現すると次のようになる。
<Rotation>x_i x_j x_k y_i y_j y_k z_i z_j z_k </Rotation>

位置情報データデータベースの位置情報テーブルとの対応は次のようになる。

表 5.3-15 位置情報データデータベースの位置情報テーブルとの対応

成分	位置情報テーブル	意味
x_i	i_axis_x	***i***軸単位ベクトルのグローバル座標系における ***x***成分
x_j	j_axis_x	***j***軸単位ベクトルのグローバル座標系における ***x***成分
x_k	k_axis_x	***k***軸単位ベクトルのグローバル座標系における ***x***成分
y_i	i_axis_y	***i***軸単位ベクトルのグローバル座標系における ***y***成分
y_j	j_axis_y	***j***軸単位ベクトルのグローバル座標系における ***y***成分
y_k	k_axis_y	***k***軸単位ベクトルのグローバル座標系における ***y***成分
z_i	i_axis_z	***i***軸単位ベクトルのグローバル座標系における ***z***成分
z_j	j_axis_z	***j***軸単位ベクトルのグローバル座標系における ***z***成分
z_k	k_axis_z	***k***軸単位ベクトルのグローバル座標系における ***z***成分

（４）　ＤＢ概要

ユビキタスシステムで動作する各種サーバにて使用するデータベースの種類を定義する。

表 5.3-16 データベースの一覧

オブジェクト情報データベース	オブジェクト情報サーバ上に置き，全てのオブジェクトの情報を格納するもの。
位置情報データベース	位置管理サーバ上に置き，全てのオブジェクトの位置および向きに関する情報を格納するもの
(温度)センサ情報データベース	プロキシ・サーバ上に置き，温度センサの温度データを格納するもの。
設定情報データベース	クライアント検索システム上に置き，上記データベースから適時取得したデータを格納し，3次元表示に用いるもの。
シナリオ情報データベース	オブジェクト情報サーバ上に置き，作成したシナリオの情報を格納するもの

オブジェクト情報データベース，位置情報データベース，およびセンサ情報データベースの概略 ER 図を示す。

(4-1) テーブル一覧

各テーブル図においては、カラム名ではなくデータ内容がわかるような名前を使用している。

テーブルの一覧を次表に示す。

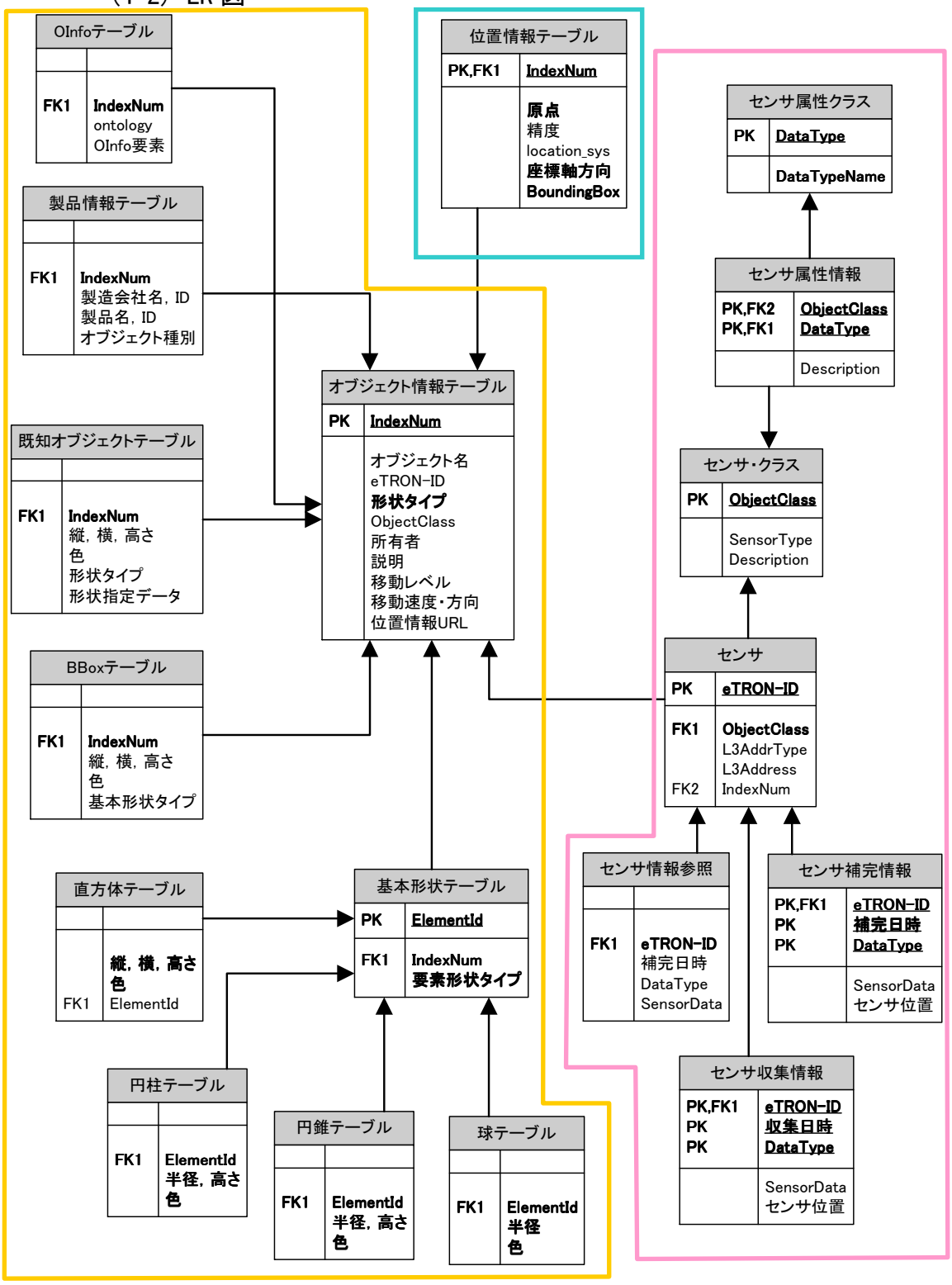
表 5.3.17 テーブル一覧

DB 名	テーブル名	概要説明	備考
オブジェクト情報 DB object_info_db			
	オブジェクト情報テーブル		
	製品情報テーブル		
	OInfo テーブル		

DB 名	テーブル名	概要説明	備考
	バウンディングボックス・オブジェクト・テーブル		
	既知オブジェクト・テーブル		
	基本形状テーブル		
	直方体テーブル		
	円柱テーブル		
	円錐テーブル		
	球テーブル		
位置情報 DB position_info_db			
	位置情報テーブル		
センサ情報 DB sensor_info_db			
	センサ属性クラス・テーブル	センサ情報の型（温度，湿度，など）を示すコードと名称の対応テーブル	
	センサクラス・テーブル	センサの種類を示すオブジェクトクラスとセンサタイプ・コードの対応表	
	センサ属性情報テーブル	センサタイプとその測定対象（センサ情報型＝温度，湿度，など）との対応表	
	センサ・テーブル	センサの実態に対応する表	
	センサ収集情報テーブル	センサから実際に収集されたセンサ情報を格納するテーブル	
	センサ補完情報テーブル	情報の欠損などを補完したセンサ情報を格納するテーブル	
	センサ情報参照テーブル	センサ補完情報のうちセンサごとにその最新データのみを格納するテーブル。	
クライアント検索システム DB objects_browser_db			
	設定情報テーブル	デモ空間サイズ，共通 DB 情報などクライアント検索システムとしての設定情報を格納するテーブル。	

DB 名	テーブル名	概要説明	備考
	合成条件マスターテーブル	三次元表示のための条件設定テーブル。	
	合成条件テーブル	三次元表示対象として選択されたオブジェクトのテーブル。	
シナリオ情報データベース			
	シナリオテーブル	生成したシナリオの情報を格納するテーブル	
	自動生成オブジェクト ID テーブル	自動生成したオブジェクトの情報を格納するテーブル	
	eTRON-ID 発番シーケンス	自動生成したオブジェクトに設定される eTRON-ID を発番するシーケンステーブル	

(4-2) ER 図



F)　　図 5.3-34　ER図

(4-3) データコード

本仕様で用いるデータコードを以下のように定義する。

(a) 形状タイプ

オブジェクト形状のタイプを示すコードで，１以上の 10 進数で表現する。オブジェクトのタイプについては，十の桁で以下のように分類する；

表 5.3.18 形状タイプ　十の桁の意味

0	： 基本形状。
1	： 基本的な既知オブジェクト
2 以上	： 今後出てくる既知オブジェクトに振り分ける

表 5.3-19 形状タイプ　コード一覧

コード	分類	説明	名称	推奨マクロ名
0	質点	属性のみのオブジェクト	Point	OBJTYPE_VIRTUAL
1	基本形状	直方体	Box	OBJTYPE_RECTANGLE
2		円柱	Cylinder	OBJTYPE_CYLINDER
3		円錐	Cone	OBJTYPE_CONE
4		球	Sphere	OBJTYPE_SPHERE
10	BoundingBox	BoundingBoxオブジェクト	BBox	OBJTYPE_BBOX
11	既知オブジェクト	床	Floor	OBJTYPE_FLOOR
12		壁	Wall	OBJTYPE_WALL
13		イス	Chair	OBJTYPE_CHAIR
14		テーブル	Table	OBJTYPE_TABLE
15		センサ	SensorNode	OBJTYPE_SENSOR
16		UT	PDA	OBJTYPE_UT
17		くずかご	TrashBox	OBJTYPE_TRASH
18		本	Book	OBJTYPE_BOOK

既知オブジェクトについては，例えばテーブルには円机，長机があり３D 表示上はそれぞれ別個の形状を用いなければならないことから，上記コードだけでは具体的な形状を規定することは難しい。これを解消するため，「サブ形状タイプ」というデータを設け，それによって３D 表示形状を特定することとする。なお，「サブ形状タイプ」はコードではなく文字列とする。詳細は「既知オブジェクトテーブル」を参照のこと。

(b) テーブル仕様の形式

ここでは各テーブルについて説明を試みる。各テーブルの形式は以下のような表で定義し，詳細の説明をその下に記述する。

テーブル英名： Sample

カラム	型	キー	必須	Idx	意味	備考
ndexnum	TEXT	PK1	○		IndexNum	
Bbox_x	BOX		○	rtree	バウンディングボックス	サンプル

図 5.3-35 テーブル説明用のサンプルテーブル

(b-1) カラム欄

各テーブルのカラム（フィールド）の名前を示す。カラム名で使用する文字は，英数字とアンダースコア"_"のみとする。本来，大文字小文字の区別はないが，混乱を避けるため小文字のみを使用する。

(b-2) 型欄

各カラム（フィールド）の型を以下のように表す；

表 5.3-20　フィールドの型

型		表記法	PostgreSQLでの型表現	説明
文字列	固定長	CHAR(n)	character(n) char(n)	長さn文字の空白でパッドされた固定長文字列
	可変長	VCHR(n)	character varying(n) varchar(n)	最大n文字の制限付き可変長文字列
	可変長	TEXT	text	長さ制限のない可変長文字列
整数	正負	INT2	smallint int2	符号付き2バイト整数
	正負	INT4	integer, int, int4	符号付き4バイト整数 -2147483648～2147483647
	正負	INT8	bigint	符号付き8バイト整数
	シリアル	SRL	serial	自動的に通し番号をつけるデータ型。範囲は1 ～ 2147483647。
	シリアル	BSRL	bigserial	自動的に通し番号をつけるデータ型。範囲は 1 ～ 9223372036854775807。
実数	単精度	FLT	real float4	単精度浮動小数点実数
	倍精度	DBL	double precision, float8	倍精度浮動小数点実数
ビット列	固定長	BIT(n)	bit(n)	長さnの固定長ビット列
	可変長	VBIT(n)	bit variability(n) varbit(n)	最大nビットの可変長ビット列

	可変長	VBIT(n)	VARBIT(n)	最大n文字の可変長ビット列
バイナリ	可変長	BYTEA	bytea	任意長のバイト列
日付	日時	TSTMP	timestamp(0) without time zone	日付と時刻両方を持つデータ。 有効桁数0を指定してミリ秒以下を省略する。
幾何	点	POINT	point	座標点 (x,y) ;座標値は単精度実数
	直線	LINE	line	無限直線。直線上の2点で表現。 ((x1,y1),(x2,y2));全て単精度実数
	線分	LSEG	lseg	線分。両端点で表現。 ((x1,y1),(x2,y2));全て単精度実数
	矩形	BOX	box	矩形。対角の2頂点で表現。 ((x1,y1),(x2,y2));全て単精度実数
	円	CIRCLE	circle	円。中心と半径で表現。 <(x,y), r>;全て単精度実数

その他 PostgreSQL で使用できる型には例えば以下のようなものもあるが，ユビキタスデータベースでは使用しない。

表 5.3-21 PostgreSQL で使用できるがここで使用しない型の例

numeric, decimal	固定小数点数	(n)または(n,m)の形で精度を指定できる数値型。ここでnは全体の桁数，mは小数点以下の桁数。
path	線分列	開いた，または閉じた線分列。
polygon	多角形	多角形

etc.

(b-3) キー欄

プライマリキーを"PK"で表す。
外部キー (foreign key)を"FK"で表す。

(b-4) 必須欄

NULL 値になってはならないカラムを"○"で示す。
ただし，NULL になってはならいだろうと思われるものには"○？"，NULL になってもいいかもしれないものには"？"を付ける。

(b-5) Idx 欄

インデックスを設定するカラムを示す。なお，プライマリキー・カラムには自動的にインデックスが付与される。

(4-4) データベースの管理

(a) PL/pgSQL の設定

標準インストール状態では、PL/pgSQL は使用できず create function ..... language 'plpgsql' ; で、"language 'plpgsql' does not exist"（または相当の）エラーが出る。pg_language を参照しても確認できる。ストアドファンクションの対象言語として PL/pgSQL を以下のようにして追加する。

```
$ su PostgreSQL 管理者
$ createlang　plpgsql　対象 DB 名）
```

pg_language は以下のようになる。

```
dbname=> select * from pg_language;
 lanname  | lanispl | lanpltrusted | lanplcallfoid | lanvalidator | lanacl
----------+---------+--------------+---------------+--------------+--------
 plpgsql  | t       | t            |         21083 |            0 |
 sql      | f       | t            |             0 |         2248 | {=U}
 internal | f       | f            |             0 |         2246 | {=}
 c        | f       | f            |             0 |         2247 | {=}
(4 rows)
```

(4-5) オブジェクト情報データベース

(a) 形状タイプクラス・テーブル

テーブル英名： shape_class

カラム	型	キー	必須	Idx	意味	備考
shape	INT2		○		オブジェクトの形状タイプ	
shapename	TEXT		○		形状タイプ名称	
modified_time	TSTMP		○		更新日時 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時	

shape

オブジェクトの形状タイプを指定するカラムで，「形状タイプ」で定義したコードを用いる。

shapename

オブジェクトの形状タイプに対応する名称で，「形状タイプ」で定義した名称を用いる。基本的に XML のタグ名に対応している。

本テーブルは AP が参照するためのもの（たとえば，クライアント検索システムの検索結果表示に使用する，など）である。

(b) オブジェクト情報テーブル

テーブル英名： object_info

カラム	型	キー	必須	Idx	意味	備考
indexnum	SRL	PK	○		IndexNum <ObjectId type="IndexNum">に対応	INT4
objectname	TEXT				オブジェクト名 <ObjectId type="Name">に対応	
etronid	TEXT				eTRON-ID <ObjectId type="eTRON-ID">に対応	UNIQUE 制約
shape	INT2		○		オブジェクトの形状タイプ 直方体，円柱，円錐，球，BBox，イス，テーブル，温度センサ，壁，床，etc.を表すもの。	
objectclass	TEXT				オブジェクト種別	
owner	TEXT				オブジェクトの所有者。	
description	TEXT				オブジェクトの説明文章。	
speed_level	TEXT				オブジェクトの移動可否，移動速度レベルを表す。 'high'|'middle'|'low'|'fixed'	
speed_rate	FLT				オブジェクトの移動速度。	[m/sec]
direction_x	FLT				オブジェクトの移動方向を表す単位ベクトルのグローバル座標系におけるX成分	
direction_y	FLT				オブジェクトの移動方向を表す単位ベクトルのグローバル座標系におけるY成分	
direction_z	FLT				オブジェクトの移動方向を表す単位ベクトルのグローバル座標系におけるZ成分	
transform	TEXT				位置・向きの情報を保持するサーバを示すURL。	

modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

indexnum
オブジェクトを特定する固有の識別子で，オブジェクトを登録する際にオブジェクト情報サーバが自動的に生成する。XML の IndexNum に相当する。

etronid
オブジェクトに取り付けられた eTRON カードに格納されている ID。カラムとしては TEXT 型だがその内容は"0x"で始まる 16 進数である。なお，etronid カラムは NULL 可だが，NULL 以外の値は一意であることとする。

objectclass
オブジェクトの種別を文字列で表したもの。DB 上では TEXT 型として扱うが，その内容はオブジェクトのクラス，サブクラスを"."でつないで種別を表現する。

shape
オブジェクトの形状タイプを指定するカラムで，「形状タイプ」で定義したコードを用いる。

(c) 基本形状テーブル
テーブル英名： primitive

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	FK	○		IndexNum	外部キー
elementid	SRL	PK	○		基本形状ID	INT4
shape	INT2		○		要素形状タイプ＝基本形状の種別。 ＝ 1 ： 直方体 ＝ 2 ： 円柱 ＝ 3 ： 円錐 ＝ 4 ： 球	
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

elementid
elementid はシリアルデータ型とし新たな基本形状オブジェクトを追加する（列を生成する）時に連番が自動的に振られるようにする。

shape
基本形状タイプを示すカラムで,「形状タイプ」で定義したコードを用いる。これにより当該オブジェクトの形状を現す基本形状の各テーブルを特定できる。

(d) 直方体テーブル
テーブル英名： box

カラム	型	キー	必須	Idx	意味	備考
elementid	INT4	FK	○		基本形状ID	外部キー
size_i	FLT		○		i方向の大きさ。幅。	[m]
size_j	FLT		○		j方向の大きさ。奥行き。	[m]
size_k	FLT		○		k方向の大きさ。高さ。	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

オブジェクトの色 diffcolor_r, _g, _b が null の場合，3 次元表示するときにデフォルト色（0.8 0.8 0.8）を用いる。

(e) 円柱テーブル
テーブル英名： cylinder

カラム	型	キー	必須	Idx	意味	備考
elementid	INT	FK	○		基本形状ID	外部キー
radius	FLT		○		底面の半径	[m]
height	FLT		○		k方向の大きさ。高さ。	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

(f) 円錐テーブル
テーブル英名： cone

カラム	型	キー	必須	Idx	意味	備考
elementid	INT4	FK	○		基本形状ID	外部キー

radius	FLT		○		底面の半径	[m]
height	FLT		○		k方向の大きさ。高さ。	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
modified_time	TSTMP		○		更新日時 [JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時 [JST]	

(g) 球テーブル

テーブル英名： sphere

カラム	型	キー	必須	Idx	意味	備考
elementid	INT4	FK	○		基本形状ID	外部キー
radius	FLT		○		底面の半径	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
modified_time	TSTMP		○		更新日時 [JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時 [JST]	

(h) BoundingBox オブジェクト・テーブル

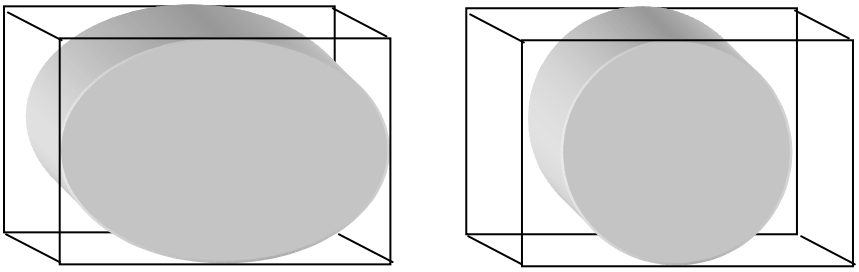
バウンディングボックスで囲まれる形状（基本形状のみ）を定義するオブジェクト。

テーブル英名： boundingbox

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	FK	○		IndexNum	外部キー
size_i	FLT		○		i方向の大きさ。幅。	[m]
size_j	FLT		○		j方向の大きさ。奥行き。	[m]
size_k	FLT		○		k方向の大きさ。高さ。	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
shape	INT2		○		内接する基本形状の種別。 = 1 ： 直方体 = 2 ： 円柱	

					= 3 : 円錐 = 4 : 球	
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

3 次元表示画面では，内接する基本形状はバウンディングボックスに合わせて変形することはなく，基本形状としての形を保ったままバウンディングボックスに内接するよう拡大・縮小して表示する。



このようにはならず　　このようになる

G)　　図 5.3-36 バウンディングボックス・オブジェクトの基本形状

オブジェクトの色 diffcolor_r，_g，_b が null の場合，3 次元表示するときにデフォルト色（0.8 0.8 0.8）を用いる

(i) 既知オブジェクト・テーブル

イス，テーブル，温度センサなど形状，大きさが決まっているオブジェクト。実際の形状（＝3D 表示画面に表示する形状）はここには持たず，クライアント検索システム内で保持する。

テーブル英名： predefined_obj

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	FK	○		IndexNum	外部キー
size_i	FLT		○		i方向の大きさ。幅。	[m]
size_j	FLT		○		j方向の大きさ。奥行き。	[m]
size_k	FLT		○		k方向の大きさ。高さ。	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	0～1.0
diffcolor_g	FLT				オブジェクト表面の色 G。	0～1.0
diffcolor_b	FLT				オブジェクト表面の色 B。	0～1.0
shape	INT		○		内接する既知オブジェクトの形状タイプ。	

					= 11 : 床 = 12 : 壁 = 13 : イス = 14 : テーブル = 15 : ~~温度~~センサ	
productshape	TEXT		○		既知形状指定データ。 既知オブジェクトの具体的な形状を示すもの。	
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

大きさ size_i, _j, _k

3 次元表示画面では，大きさ size_i，_j，_k のバウンディングボックスに内接する既知形状はバウンディングボックスに合わせて変形するか否かは既知オブジェクトの形状定義内容に依存する。既知形状としての形を保ったままバウンディングボックスに内接するよう拡大・縮小して表示する。

オブジェクトの色

オブジェクトの色 diffcolor_r，_g，_b が null の場合，3 次元表示するときにデフォルト色（0.8 0.8 0.8）を用いる（diffcolor_r，_g，_b についてはその色を適用するかどうかは不明。既知オブジェクトの形状定義内容に依存する。すなわち，色が指定されていたらそちらを採用する）。

productshape

イス，テーブルなど表示のために複数の形状があるオブジェクトに対し，既知オブジェクトの形状タイプ product だけでは形状を特定することができない。そのため，product と subtype の情報で形状を特定できるようにする；例えば product で決まるディレクトリの下に，subtype で指定される文字列で指定できる形状ファイルを置く。

(j) 製品情報テーブル

テーブル英名： product_info

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	FK	○		IndexNum	外部キー
vendor	TEXT				製造会社名	
vendorid	TEXT				製造会社番号 16進数だがDBでは文字列とする。	
productname	TEXT				製品名	
productid	TEXT				製品番号	

					16進数だがDBでは文字列とする。	
productclass	TEXT				オブジェクト種別	
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

(k) Oinfo テーブル

テーブル英名： oinfo

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	FK	○		IndexNum	外部キー
ontology	TEXT		○		<Oinfo>の ontology 属性の文字列。カラム elements の XMLに対応するDTDを表す。	
source	TEXT				<Oinfo>の source 属性の文字列。外部サーバにある要素を指定するURL。	
elements	TEXT				<Oinfo>配下のXML要素列。<Oinfo>と</Oinfo>の間を文字列として持つ。	
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

ontology

elements に格納される要素の DTD を表す。

source

Oinfo の要素が外部サーバに保持されている場合，その URL を指定するもの。

elements

Oinfo の要素。ontology で指定された DTD に則って XML 形式で記述された Oinfo の情報で，DB 上は文字列として保持する。
source カラムに値が設定されている場合（すなわち Oinfo の情報が外部サーバに存在する場合）はそちらが優先され，elements カラムの値は無意味である。これに関し DB 上は特に制約は設けないが，AP 上 source に値を設定する場合 elements を NULL にすることを推奨する。

(l) クライアント検索システム I/F テーブル

テーブル英名： if_objbrowser

カラム	型	キー	必須	Idx	意味	備考
client	TEXT				クライアント検索システム側のユーザを識別するコード	
indexnum	INT4				IndexNum	object_info
objectname	TEXT				オブジェクト名	object_info
etronid	TEXT				eTRON-ID	object_info
shape	INT2				オブジェクトの形状タイプ	object_info
objectclass	TEXT				オブジェクト種別	object_info
owner	TEXT				オブジェクトの所有者	object_info
description	TEXT				オブジェクトの説明文章	object_info
speed_level	TEXT				オブジェクトの移動可否	object_info
speed_rate	FLT				オブジェクトの移動速度	object_info
direction_x	FLT				オブジェクト移動方向X成分	object_info
direction_y	FLT				オブジェクト移動方向Y成分	object_info
direction_z	FLT				オブジェクト移動方向Z成分	object_info
transform	TEXT				位置・向きの情報を保持するサーバを示すURL。	object_info
modified_time	TSTMP				参照オブジェクトの更新日時[JST]	object_info
create_time	TSTMP				参照オブジェクトの生成日時[JST]	object_info
size_i	FLT				i方向の大きさ／半径	[m]
size_j	FLT				j方向の大きさ／高さ	[m]
size_k	FLT				k方向の大きさ	[m]
diffcolor_r	FLT				オブジェクト表面の色 R。	
diffcolor_g	FLT				オブジェクト表面の色 G。	

diffcolor_b	FLT				オブジェクト表面の色 B。	
primitiveshape	INT2				BBoxオブジェクトの基本形状タイプ。 = 1 : 直方体 = 2 : 円柱 = 3 : 円錐 = 4 : 球	boundingbox.shape
productshape	TEXT				既知形状指定データ。 既知オブジェクトの具体的な形状を示すもの。	predefined_obj.productshape

クライアント検索システムの検索画面（オブジェクトの詳細画面）および三次元表示機能から IndexNum をキーにしてオブジェクトを検索するときに使用するインタフェース用テーブルであり，PL/pgSQL のストアドファンクション select_for_objbrowser()によりデータが設定される。備考欄に参照するテーブルおよびカラムを示す。

select_for_objbrowser(client, id)

引数：　client [TEXT]　検索する機能を識別するユニークな名前。今回は検索機能と三次元表示機能の２つを識別できればよく，それぞれの機能を開発する時に互いに重複しないよう決定すること。

id [INT4]　検索対象オブジェクトの IndexNum

戻り値：［INT2］　0 ＝正常，　1＝異常

使 用 方法：

```
select select_for_objbrowser( 'XXXX', 123 );
select * from if_objbrowser;
```

client

検索する機能を識別するユニークな名前。今回は検索機能と三次元表示機能の２つを識別できればよく，それぞれの機能を開発する時に互いに重複しないよう決定すること。

size_i, size_j, size_k

オブジェクトのサイズを示す情報でオブジェクトの形状タイプごとにその意味が異なる。下表に各カラムに設定するデータの参照元を形状タイプごとに示す。なおこの表では boundingbox の省略形として"bbox"，predefined_obj の省略形として"pre"を用いていいる。

形状タイプ	形状の意味	size_i	size_j	size_k
1	直方体	box.size_i	box.size_j	box.size_k

2	円柱	cylinder.radius	cylinder.height	－
3	円錐	cone.radius	cone.height	－
4	球	sphera.radius	－	－
10	BBoxオブジェクト	bbox.size_i	bbox.size_j	bbox.size_k
13～18	既知オブジェクト	pre.size_i	pre.size_j	pre.size_k

diffcolor_r, diffcolor_g, diffcolor_b
オブジェクトの色を示す情報で，形状タイプごとに参照元のテーブルが異なる。

(4-6) 位置情報データベース
(a) 位置情報テーブル
テーブル英名： position_info

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	PK	○		IndexNum	
origin_x	FLT		○		グローバル座標系におけるローカル原点のX座標値	[m]
origin_y	FLT		○		グローバル座標系におけるローカル原点のY座標値	[m]
origin_z	FLT		○		グローバル座標系におけるローカル原点のZ座標値	[m]
accuracy	FLT				位置情報の精度	今回は使用しない
location_sys	TEXT				測位システムに関する情報	
i_axis_x	FLT		○		i軸単位ベクトルのグローバル座標系におけるX成分	
i_axis_y	FLT		○		i軸単位ベクトルのグローバル座標系におけるY成分	
i_axis_z	FLT		○		i軸単位ベクトルのグローバル座標系におけるZ成分	
j_axis_x	FLT		○		j軸単位ベクトルのグローバル座標系におけるX成分	
j_axis_y	FLT		○		j軸単位ベクトルのグローバル座標系におけるY成分	
j_axis_z	FLT		○		j軸単位ベクトルのグローバル座標系におけるZ成分	
k_axis_x	FLT		○		k軸単位ベクトルのグローバル座標系におけるX成分	
k_axis_y	FLT		○		k軸単位ベクトルのグローバル座標系におけるY成分	

k_axis_z	FLT		○		k軸単位ベクトルのグローバル座標系におけるZ成分	
bbox_xy	BOX		○	rtree	グローバル座標系における当該オブジェクトのバウンディングボックスのXY面 '((x, y),(x, y))'の形	[m]
bbox_yz	BOX		○	rtree	グローバル座標系における当該オブジェクトのバウンディングボックスのYZ面 '((y, z),(y, z))'の形	[m]
bbox_zx	BOX		○	rtree	グローバル座標系における当該オブジェクトのバウンディングボックスのZX面 '((z, x),(z, x))'の形	[m]
modified_time	TSTMP		○		更新日時［JST］ 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時［JST］	

本テーブルがオブジェクト情報テーブルと同じ DB にあれば，IndexNum は外部キーとなる。

(a-1) バウンディングボックスの計算方法

(a-1-1) 直方体，BoundingBox オブジェクト，既知オブジェクト

$\boldsymbol{O}$ ：オブジェクトの原点（origin_x, origin_y, origin_z）
$\boldsymbol{i}$ ：オブジェクトの i 軸方向単位ベクトル（i_axis_x, i_axis_y, i_axis_z）
$\boldsymbol{j}$ ：オブジェクトの j 軸方向単位ベクトル（j_axis_x, j_axis_y, j_axis_z）
$\boldsymbol{k}$ ：オブジェクトの k 軸方向単位ベクトル（k_axis_x, k_axis_y, k_axis_z）
S_i, S_j, S_k:オブジェクトの各軸方向サイズ（オブジェクト情報 DB から；size_i, size_j, size_k）

のとき，

$$\boldsymbol{I} = (S_i / 2) \cdot \boldsymbol{i}$$
$$\boldsymbol{J} = (S_j / 2) \cdot \boldsymbol{j}$$
$$\boldsymbol{K} = (S_k / 2) \cdot \boldsymbol{k}$$

とすると，バウンディングボックスの８頂点は以下のベクトル計算で求められる。

$$\boldsymbol{V1} = \boldsymbol{O} - \boldsymbol{I} - \boldsymbol{J} - \boldsymbol{K}$$
$$\boldsymbol{V2} = \boldsymbol{O} + \boldsymbol{I} - \boldsymbol{J} - \boldsymbol{K}$$
$$\boldsymbol{V3} = \boldsymbol{O} + \boldsymbol{I} + \boldsymbol{J} - \boldsymbol{K}$$
$$\boldsymbol{V4} = \boldsymbol{O} - \boldsymbol{I} + \boldsymbol{J} - \boldsymbol{K}$$

$$V5 = O - I - J + K$$
$$V6 = O + I - J + K$$
$$V7 = O + I + J + K$$
$$V8 = O - I + J + K$$

バウンディングボックス頂点のＸ座標値の最小値 Xmin 最大値 Xmax，Ｙ座標値の最小値 Ymin 最大値 Ymax，Ｚ座標値の最小値 Zmin 最大値 Zmax を求めれば，バウンディングボックスの各座標面のカラムは以下のように定義できる。

bbox_xy = ((Xmin,Ymin), (Xmax,Ymax))
bbox_yz = ((Ymin,Zmin), (Ymax,Zmax))
bbox_zx = ((Zmin,Xmin), (Zmax,Xmax))

(a-1-2) 円柱

O ：オブジェクトの原点（origin_x, origin_y, origin_z）
k ：オブジェクトの k 軸方向単位ベクトル（k_axis_x, k_axis_y, k_axis_z）
R, H ：円柱の半径と高さ（オブジェクト情報 DB 円柱テーブルから； radius, height）

3 次元空間上の円をＸ軸に投影したとき，その範囲は次式で求められる。

$$Xmin = 0x - R\cos\theta, \quad Xmax = 0x + R\cos\theta$$

ここで 0x は円の中心のＸ座標値，Ｒは円の半径，θ はＸ軸と円の面との成す角（小さい方）。

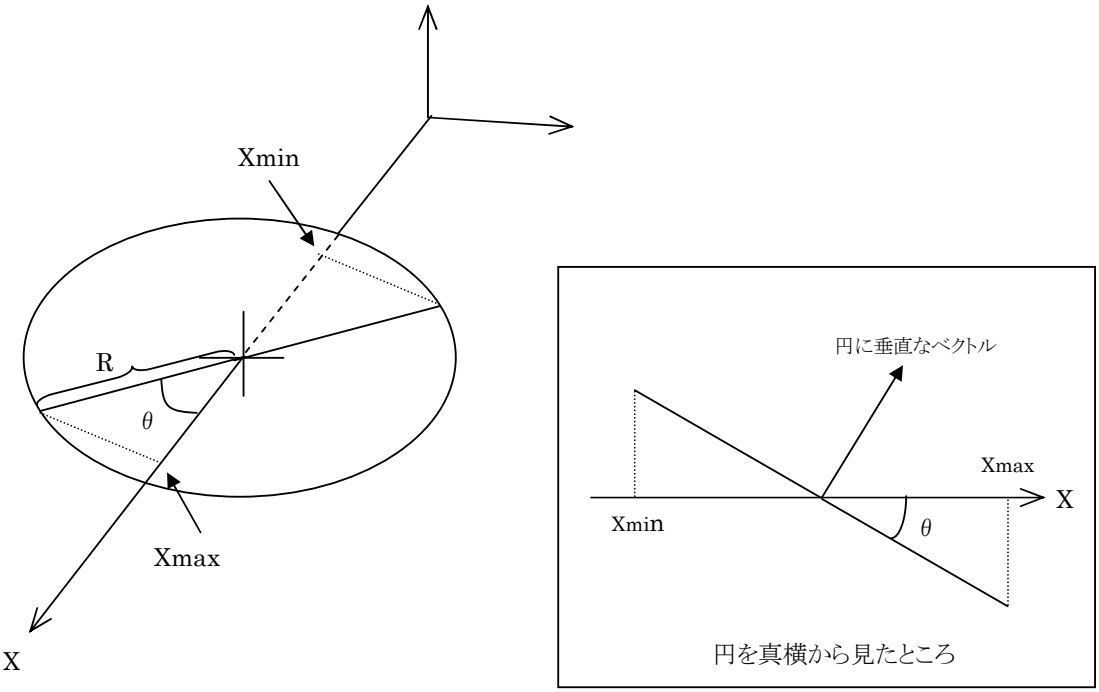


図 5.3-37 円柱の計算方法

円柱のバウンディングボックスは底面の円のバウンディングボックスから求められる。すなわち，2 つの底面のＸ座標軸上への投影範囲は $\cos\theta = C = \sqrt{(1 - k_x^2)}$ として

$$Xmin1 = (Ox - k_x \cdot H/2) - R \cdot C,$$
$$Xmax1 = (Ox - k_x \cdot H/2) + R \cdot C$$
$$Xmin2 = (Ox + k_x \cdot H/2) - R \cdot C,$$
$$Xmax2 = (Ox + k_x \cdot H/2) + R \cdot C$$

で計算でき，円柱のバウンディングボックスのＸ座標 Xmin，Xmax は

$$Xmin = Min\ (Xmin1,\ Xmin2)$$
$$Xmax = Max\ (Xmax1,\ Xmax2)$$

で求められる。Ｙ軸，Ｚ軸についても同様にしてもとめれば，以下バウンディングボックスのカラムの設定方法は直方体の場合と同様である。

(a-1-3) 円錐

$\boldsymbol{O}$ ：オブジェクトの原点（origin_x，origin_y，origin_z）
$\boldsymbol{k}$ ：オブジェクトのｋ軸方向単位ベクトル（k_axis_x，k_axis_y，k_axis_z）
R，H ：円錐の半径と高さ（オブジェクト情報 DB 円錐テーブルから； radius，height）

円錐のバウンディングボックスは底面の円のバウンディングボックスと円錐の頂点から求められる。すなわち，底面のＸ座標軸上への投影範囲は $\cos\theta = C = \sqrt{(1 - k_x^2)}$ として

$$Xmin1 = (Ox - k_x \cdot H/2) - R \cdot C,$$
$$Xmax1 = (Ox - k_x \cdot H/2) + R \cdot C$$

頂点のＸ座標値は

$$Xtop = Ox + k_x \cdot H/2$$

で計算できる。円錐のバウンディングボックスのＸ座標 Xmin，Xmax は

$$Xmin = Min\ (Xmin1,\ Xtop)$$
$$Xmax = Max\ (Xmax1,\ Xtop)$$

で求められる。Ｙ軸，Ｚ軸についても同様にしてもとめれば，以下バウンディングボックスのカラムの設定方法は直方体の場合と同様である。

(a-1-4) 球

O ：オブジェクトの原点（origin_x, origin_y, origin_z）
R ：半径 （オブジェクト情報ＤＢ 球テーブルから； radius）

各座標軸について最大値，最小値を以下のように計算する。

Xmin ＝ Ox － R
Xmax ＝ Ox ＋ R

（Ｙ軸，Ｚ軸についても同様）
以下バウンディングボックスのカラムの設定方法は直方体の場合と同様である。

(4-7) センサ情報データベース
(a) センサ属性クラス・テーブル
テーブル英名： sensor_attribute_class

カラム	型	キー	必須	Idx	意味	備考
datatype	INT2	PK	○		情報タイプ。	今回は 0固定？
datatypename	TEXT		○		情報タイプの名称。 XML の <SensorData> の type 属性に対応する。	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

datatype
センサから送信される「センサ情報のタイプ」に相当するコードで，「温度」，「湿度」，などセンサ情報の種別を示す。今回は，0＝「温度」のみ。
datatype の値はセンサプロトコル仕様で「SensorInfoType」として規定される。

datatypename
センサ情報タイプの名称で，datatype に対応する文字列を格納する。datatypename は XML の<SensorData>タグの属性 type に使用する。
″Temperature″　温度。
″Humidity″　湿度。
その他

なお，今回の版で本テーブルに実際に格納されるのは温度のみである；
datatype　＝ 0
datatypename　＝ ″Temperature″

(b) センサクラス・テーブル
テーブル英名： sensor_class

カラム	型	キー	必須	Idx	意味	備考
objectclass	TEXT	PK	○		センサから送信される「センサTYPE」に対応するオブジェクトクラス	
sensortype	INT2		○		センサTYPE。 センサから送信される「センサTYPE」値。	
description	TEXT				説明	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

objectclass

Y-0130-2 の<SensorClassID>に相当する。オブジェクト情報テーブルの objectclass と同じものなのでカラム名も同じにした。

センサの種別を表す文字列で，文字列は、下例のようにクラス，サブクラスを「.」でつないだものである。

ex)　LM92 による気温センサ → ″Sensor.Temperature.Lm92″

sensortype

センサから送信される SensorType パラメータの値。具体的な値とオブジェクトクラスとの対応関係はセンサプロトコル仕様で決められる。

なお，今回の版で本テーブルに実際に格納されるのは上例の LM92 のみである；

objectclass　　＝ ″Sensor.Temperature.Lm92″

sensortype　　＝ 0

(c) センサ属性情報テーブル

テーブル英名： sensor_attribute

カラム	型	キー	必須	Idx	意味	備考
objectclass	TEXT	PK ， FK	○		センサから送信される「センサTYPE」に対応するオブジェクトクラス	
datatype	INT2	PK FK	○		情報タイプ	
description	TEXT				温度センサの属性値の実体	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

objectclass

センサ種別（温度センサ，湿度センサ，など）を表すもので，外部キーとしてセ

ンサクラス・テーブルに関連付けられている。

datatype
外部キーとしてセンサ属性クラス・テーブルに関連付けられている。

description
温度センサの場合，センサデータ用ＸＭＬ定義の temperature タグに設定する属性を格納する。

(d) センサ・テーブル
テーブル英名： sensor

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4	(FK)			IndexNum	
objectclass	TEXT	FK	○		センサから送信される「センサTYPE」に対応するオブジェクトクラス	
etronid	TEXT	PK	○		eTRON-ID	
l3addrtype	TEXT				L3アドレスのタイプ "eTRON-ID"|"MAC"|"IP"	
l3address	TEXT				L3アドレス	
x	FLT				センサ位置 X座標	
y	FLT				センサ位置 Y座標	
z	FLT				センサ位置 Z座標	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

indexnum
オブジェクト情報テーブル，位置情報テーブルに登録されているセンサの IndexNum。

objectclass
センサ種別（温度センサ，湿度センサ，など）を表すもので，センサクラス・テーブルへの外部キー。

etronid
センサに取り付けられた eTRON カードに格納されている ID。"0x"で始まる 16 進数を文字列で格納する。

l3address

eTRON-ID 固定？
L3 アドレスは l3addrtype の値の別によって下記が格納される。

eTRON-ID	オブジェクトに取り付けたeTRONカードに格納されているID。”0x”で始まる16進数で表記する。
MAC	MACアドレスが入る。”0x”で始まる16進数で表記する。
IP	IPアドレスが入る。”0x”で始まる16進数で表記する。

(e) センサ収集情報テーブル
テーブル英名： sensor_rawdata

カラム	型	キー	必須	Idx	意味	備考
etronid	TEXT	PK FK	○		eTRON-ID	
collectiontime	TSTMP	PK	○		センサ情報が収集された日時[UTC]。	
datatype	~~TEXT~~ INT2	PK	○		情報タイプ。どの型(例えば温度、湿度)の情報であるかを示すデータ。 ~~″Temperature″~~ ~~｜″Humidity″｜TBD~~	
sensordata	FLT				センサの値	
x	FLT				センサ位置 X座標	
y	FLT				センサ位置 Y座標	
z	FLT				センサ位置 Z座標	

collectiontime, datatype, および sensordata は必須データだが DB への投入時のコストを軽減するため（ごく微量だが), NOT NULL 指定にはしない。

collectiontime
センサからの情報は CCYY-MM-DDThh:mm:ss 形式でタイムゾーンは UTC。collectiontime には″CCYY-MM-DD△hh:mm:ss″（△：空白）にして設定する必要がある。(それとも″CCYY-MM-DDThh:mm:ss″形式の文字列とする？)

sensordata
センサから得られる情報で, datatype の値の別によって下記が格納される。

Temperature	温度。本当は、セ氏、華氏等の単位がつくべきだが、今回はセ氏を仮定。値は文字列での実数値(固定小数点形式)
Humidity	湿度。値は文字列での実数値(固定小数点形式)

(f) センサ補完情報テーブル
テーブル英名： sensor_cookeddata

カラム	型	キー	必須	Idx	意味	備考
etronid	TEXT	PK FK	○		eTRON-ID	
cookedtime	TSTMP	PK	○		センサデータが補完された日時。	
datatype	~~TEXT~~ INT2	PK	○		情報タイプ。どの型(例えば温度、湿度)の情報であるかを示すデータ。 ~~"Temperature"~~ ~~|"Humidity"|TBD~~	
sensordata	FLT				センサの値	
x	FLT				センサ位置 X座標	
y	FLT				センサ位置 Y座標	
z	FLT				センサ位置 Z座標	

cookedtime, datatype, および sensordata は必須データだが DB への投入時のコストを軽減するため（ごく微量だが), NOT NULL 指定にはしない。

cookedtime

センサデータを補完した日時で Y-0130-2「プロキシでのセンサデータ用の DTD 定義」では CCYY-MM-DDThh:mm:ss 形式でタイムゾーンは UTC。cookedtime には"CCYY-MM-DD△hh:mm:ss"（△：空白）にして設定する必要がある。（それとも"CCYY-MM-DDThh:mm:ss"形式の文字列とする？）

sensordata

センサから得られる情報で, datatype の値の別によって下記が格納される。

0（＝"Temperature"）　温度。本当は、セ氏、華氏等の単位がつくべきだが、今回はセ氏を仮定。値は文字列での実数値(固定小数点形式)

1（＝"Humidity"）　湿度。値は文字列での実数値(固定小数点形式)

(g) センサ情報参照テーブル

テーブル英名： sensor_ref

カラム	型	キー	必須	Idx	意味	備考
etronid	TEXT	PK FK	○		eTRON-ID	
cookedtime	TSTMP				センサデータが補完された日時。	
datatype	INT2	PK	○		情報タイプ。どの型(例えば温度、湿度)の情報であるかを示すデータ。	
sensordata	FLT				センサの値	

各センサの補完情報の最新値のみを格納するテーブルで, クライアント検索システムから参照される。センサ補完情報テーブルにデータを投入するとき, 同時に

etronid＋datatype で決まるレコードを挿入，または更新すること。

(4-8) クライアント検索システムＤＢ

(a) 設定データ・テーブル

テーブル英名： conf

カラム	型	キー	必須	Idx	意味	備考
space_size_x	FLT		○		デモ空間のXサイズ［m］	横方向 東西方向
space_size_y	FLT		○		デモ空間のYサイズ［m］	縦方向 南北方向
wall_height	FLT		○		壁面の高さ［m］	
sensor_dist	FLT				センサ検索距離［m］	
objinfo_host	TEXT		○		オブジェクト情報DB HOST	
objinfo_port	TEXT				同 ポート番号	
objinfo_dbname	TEXT		○		同 DB名	
objinfo_user	TEXT				同 ユーザ名	
objinfo_passwd	TEXT				同 パスワード	
posinfo_host	TEXT		○		位置情報DB HOST	
posinfo_port	TEXT				同 ポート番号	
posinfo_dbname	TEXT		○		同 DB名	
posinfo_user	TEXT				同 ユーザ名	
posinfo_passwd	TEXT				同 パスワード	
snsinfo_host	TEXT		○		センサ情報DB HOST	
snsinfo_port	TEXT				同 ポート番号	
snsinfo_dbname	TEXT		○		同 DB名	
snsinfo_user	TEXT				同 ユーザ名	
snsinfo_passwd	TEXT				同 パスワード	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

クライアント検索システムで共通に使用するパラメタを格納するテーブル。

センサ検索距離

壁面上のセンサを検索する時に使用する検索範囲を決定するための距離。下図のように壁面の前後に検索範囲を広げる時に用いる。

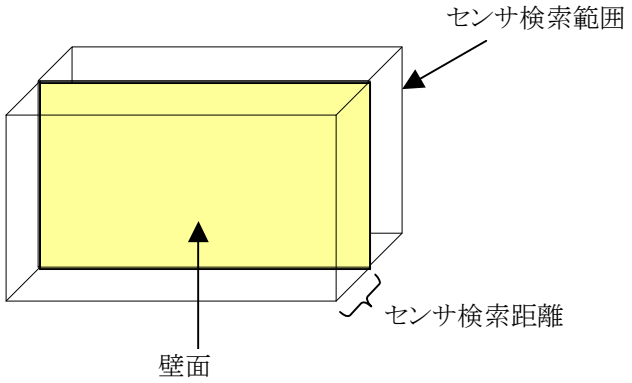


図 5.3-38 センサ検索距離の定義

HOST
各データベースが置かれているサーバのホスト名／IP アドレスを設定する。

ポート番号
各データベースの置かれる PostgreSQL サーバの TCP ポート番号を設定する。カラムが NULL の場合，プログラム上でポート番号の設定は行わない（結果としてデフォルト値 5432 が使用される）。

ユーザ名，パスワード
各データベースのユーザ名，パスワードを設定する。

JDBC の場合
JDBC でデータベースに接続するときは以下の形式の URL を使用する。

```
jdbc:postgresql://HOST:PORT/DBNAME
```

各データベースに接続する際は，XXX_host，XXX_port，XXX_dbname の各カラムの値を用いて URL を生成すること。

(b) 合成条件マスターテーブル
テーブル英名： viewing_cond

カラム	型	キー	必須	Idx	意味	備考
upd_period	INT4		○		三次元表示情報の更新時間間隔［sec］	
objs_upd_time	TSTMP		○		合成条件テーブル更新日時［JST］。	
modified_time	TSTMP		○		更新日時[JST] 初期値は生成日時と等しい	
create_time	TSTMP		○		生成日時[JST]	

クライアント検索システム・検索機能で指定された合成条件（自動更新時間，など）を格納するテーブル。検索機能と３次元表示機能のインタフェースとなる。

(c) 合成条件テーブル

テーブル英名： viewing_objects

カラム	型	キー	必須	Idx	意味	備考
indexnum	INT4		○		IndexNum	
objectname	TEXT				オブジェクト名	
etronid	TEXT				eTRON-ID	
shape	INT2		○		オブジェクトの形状タイプ	デフォルト 9999
objectclass	TEXT				オブジェクト種別	
speed_level	TEXT				オブジェクトの移動可否，移動速度レベルを表す。 'high'|'middle'|'low'| 'fixed'	

クライアント検索システム・検索機能で指定された合成条件のうち表示オブジェクトのＩｎｄｅｘＮｕｍを格納するテーブル。検索機能と３次元表示機能のインタフェースとなる。それ以外のカラムは予備であり，データの設定は不要。

shape

カラムの制約として NOT NULL だが，DEFAULT で 9999 が設定されるので設定時に考慮する必要はない。

(4-9) シナリオ情報データベース

(a) シナリオテーブル

テーブル英名： Scenario

カラム	型	キー	必須	Idx	意味	備考
ScenarioNo	SRL	PK	○		ユニークなキー	
ObjectClass	INT		○		1:Sensor.Temp 2:Computer.NoteBook 3:Computer.UT	
num	INT		○		個数	
create_time	TSTMP		○		作成日時	
speed_level	TEXT		○		オブジェクトの移動速度レベル	
note	TEXT				備考	
location	TEXT				場所	

ScenarioNo

シナリオを特定するコードでシナリオ作成の時に自動生成される。

(b) 自動生成オブジェクト ID テーブル

テーブル英名： AutoObjectID

カラム	型	キー	必須	Idx	意味	備考
seq	SRL	PK	○		ユニークなキー	
ScenarioNo	INT4		○		シナリオとのリンクキー	
ObjectClass	INT		○		1:Sensor.Temp 2:Computer.NoteBook 3:Computer.UT	
IndexNum	INT4		○		自動生成したオブジェクトの indexnum	
etronid	TEXT				自動生成したオブジェクトの etronid	

seq
自動生成したオブジェクトを特定するコードでオブジェクト自動生成時に自動生成される。
ScenarioNo
自動生成されたオブジェクトがどのシナリオにより生成されたかを識別するための識別コードである。

(c) eTRON-ID 発番シーケンス
シーケンス英名： eTRONID_Seq
自動生成されるオブジェクトに設定される etromid を自動生成するのに使用する。

（４）　評価

実装したシステムの評価を行なった。提案手法ではオブジェクトの位置測定に一つ以上の点の位置測定を行なう。測定する点は、大きさを測定する場合を除き、比較的自由に選ぶことができる。もし、測定した空間位置に誤差が大きいと、オブジェクトが空中に浮いたり、テーブルにめり込むなどの状況が生じることになってしまう。そこで、位置測定の誤差が少なく抑えることが可能な測定点を選択するために、条件の異なる点の位置検出における誤差を測定した。

(4-1) 評価条件

測定した点は、図 5.3-39 のイスにおける条件の異なる三点(点 A から C)である。各点において 20 回の試行を実施した。また、比較のために位置センサを静止させた状態(点 D)で同回数の位置を測定したした。

各点の条件は下記の通りである。

A 周囲に障害となる物がなく天井への見通しの良い場所。イスの背もたれの端。

B 天井への見通しは良いが対象物と接していない空中。イスを直方体に近似し、イスに向かって手前右上の直方体の隅。

C 対象物の側面にあり比較的天井への見通しが悪い場所。イスの脚の床面に接している地点。

D 位置センサを動かさない状態での測位システムの誤差を確認するための比較点である。

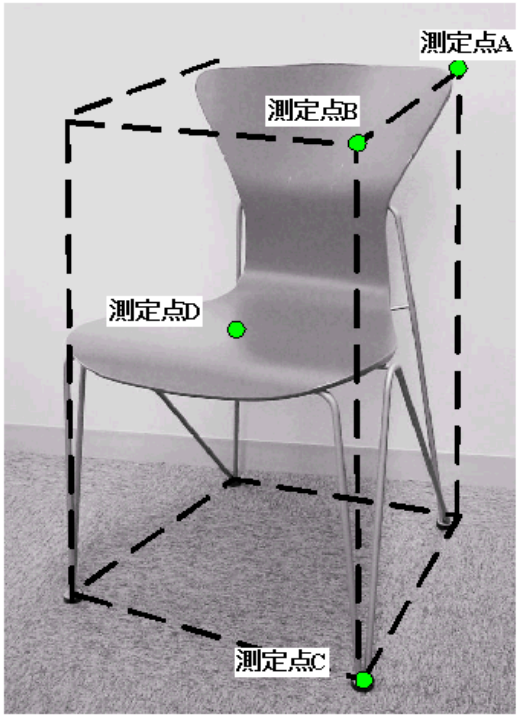


図 5.3-39 測定対象点

(4-2) 結果と考察

前項に示した条件で、各点での三次元位置を測定した。測定により得られた点の重心からの距離の最大、平均、分散の値は表 5.3-22 になった。

表 5.3-22 重心からの距離の最大、平均、分散値

	最大距離(cm)	平均距離(cm)	距離の分散
点 A	1.69	1.00	0.13
点 B	5.04	2.51	1.23
点 C	3.87	2.18	0.44
点 D	1.25	0.49	0.09

点 D の測定結果から、測位システム設置後の静止状態では、このセンサが 1cm 程度の測定精度を持つことが確認された。また、各測定を比較すると、測定誤差は点 A, 点 C, 点 B の順に大きくなった。

条件の良い測定点 A では、オブジェクトに接して測定を行なうことが出来るため、静止状態の点 D と比較しても、測定におけるばらつきも少なく、1cm 前後から大きくても 2cm 未満の誤差にとどまっている。オブジェクトの位置測定に使用し

ても、十分な精度を得られると思われる。

同様にオブジェクトに接して測定を行なえる測定点 C では、センサ部を対象物に密着する事が出来るのだが、測定対象物自体が超音波による測定の障害物となり若干精度が落ちている。誤差としては、およそ 2cm 程度の値に落ち着いている。

一方、オブジェクトと接することのない測定点 B では、目測により対象物が外接する直方体の形状を推測し、空中に存在する直方体の隅を測定している。このため、測定した値も大きくばらつき、安定した値を得ることが出来ず、最大で 5cm 程度のばらつきを生じている。この値はイスの大きさから考えると十分に大きいものであり、点 B は対象物の位置や大きさの測定には適さない事がわかった。

これらの結果より、測定点を適切に選ぶことで、1 から 2cm 程度の誤差で位置を測定できる事がわかった。位置情報を利用するアプリケーションにも依存するが、あまり大きくないオブジェクト（一辺が 10cm 程度）にも適用が可能と考えられる。

また、本方式では大きさの情報を非接触 IC カード、あるいはオブジェクト情報サーバから取得した。この大きさは事前に測定した値を IC カードあるいはサーバに登録しておくか、測位システムを利用して測定をする必要があり手間を必要とする。また、実際には対象となるオブジェクトの形状も直方体や円柱などの簡単なものだけではなく、それらの組合わせや複雑な形をしており、これらの情報を一つずつ入力することは多くの手間を要する。

しかし、屋内に存在する物の多くは、既製品であることが多い。そこで、オート ID センタやユビキタス ID センタのように身の回りの物にあらかじめ ID 情報が付与されるようになれば、ID 番号から大きさや形状などの製品の情報を知ることが可能であり、この問題は解消される。

(4-3) 実際のシステム稼動例

実験システムの全景を図 5.3-40 に示す。また図 5.3-41 および図 5.3-42 に位置測位システムとして使用した超音波による位置センサの固定側と、端末側センサを示す。さらに、図 5.3-43 から図 5.3-48 に実際に箱を測定する際の、測定前の状況、測定状況および測定後の結果を示す。

図 5.3-40 システム全景

図 5.3-41 超音波による屋内高精度測位装置　（天井取り付け側）

図 5.3-42 位置測定装置。　右下についているのが超音波による屋内高精度測位装置のセンサ部分

図 5.3-43 測定前の現実世界

図 5.3-44 測定前の位置管理システムでの位置管理状況

図 5.3-45 測定手順(1)　ID 取得

図 5.3-46 測定手順(2) 3 隅のうちの 1 点の測定

図 5.3-47 測定後の現実世界

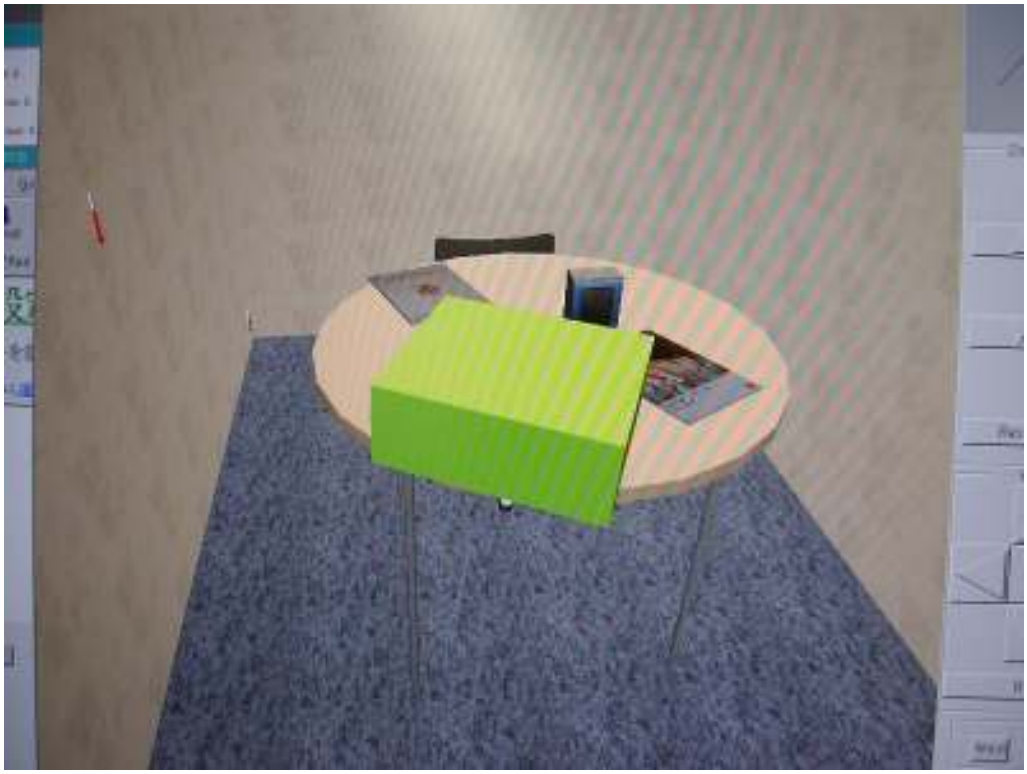


図 5.3-48 測定後の位置管理システムでの位置管理状況

（ウ） 分散協調センサーによる人の位置追跡の研究

(1) はじめに

プロセッサ及び周辺機器は日々進化し、現在では小型で高性能な安価なマシンが容易に入手できるようになった。また、インターネットの普及により、今では生活空間のあらゆるところにネットワークインフラが整備されている。それに従い、生活空間のあらゆるところにネットワークで接続された（小さな）コンピュータが、周囲のコンピュータと協調動作を行うことにより、人間の行動を様々な面からサポートすることを目指すユビキタスコンピューティングの基盤は整いつつある。

ユビキタスコンピューティングにおいて重要な概念のひとつとして、context-awareness がある。これは、ユーザや周囲の状態（context）によってアプリケーションの動作を変化させるということである。context には様々なものが考えられるが、特に「ユーザがどこにいるか」はサービスを提供する際、重要な情報となる。

今回、この「ユーザがどこにいるのか」をコンピューティング環境が把握する方式を検討することを目的として、カメラセンサを主体とする複数のセンサノードを協調動作させてユーザの位置追跡をするシステムの構築を行った。

(2) 関連研究

人間の位置追跡システムは既にいくつか存在するが、使われている技術や手法はそれぞれ異なり、それに応じて利点や欠点が存在する。位置追跡には、赤外線、超音波、ラジオ電波、カメラ画像などいろいろな技術を使われているが、従来の位置追跡システムは、大きくユーザに位置検出用のデバイスを所持させるタイプのものとビデオカメラの画像を解析して位置追跡をするタイプのものに、分類できる。

(2-1) 位置検出用のデバイス使用型

ユーザに位置検出用のデバイスを持たせるタイプの位置検出システムでは、そのデバイスの位置をユーザの位置として検出する。位置検出の方法としては、デバイスから発せられる信号の衰弱の程度を複数点で観測することにより位置を計算するもの、デバイスと受信機の間の信号の往復時間を複数点で観測することにより位置を計算するもの、デバイスからの信号を受信した近傍にある受信機の位置をそのデバイスの位置として近似するものがある。デバイスからの信号を乗せる媒体としては、赤外線、超音波、ラジオ電波などがあるが、それぞれ利用環境に制限がある。赤外線を使うものとしては、Xerox で開発された PARCTAB system[3] や Olivetti Research Laboratory で開発された Active Badge System[5] などがある。これらのシステムでは、デバイスから信号を受信した受信機の位置をそのデバイスの近似的な位置として取得する。求められる位置の精度はそれほど高くなく、ユーザがどの部屋にいるかということがわかるレベルである。赤外線は壁に遮られてしまうため、デバイスと受信機の間に障害物がある環境では使えない。また、直射日光など赤外線が含まれる強い光源がある場所でも使えない[1]。

超音波を使うものとしては、Active Badge System の後継システムである Bat System[4] や、ラジオ電波と超音波を併用する Cricket System[6][7] などがある。Bat System は天井に格子状に配置された複数の受信機により、デバイスからの信号の往復時間を測定することにより位置を計算する。Cricket System では、部屋に設置されたビーコンから発せられる信号をユーザの所持するデバイスが受信して、ユーザ側で位置が計算される。ビーコンからはラジオ電波と超音波の信号が発せされるがそれらの到達時間の差から距離を計算する。複数のビーコンからの位置が計算出来るときにはラテレーションにより位置が計算できるが、一つのビーコンからの距離しかわからないときには、そのビーコンの位置でユーザの位置を近似する。超音波を使ったシステムでは、一般的に赤外線に比べて高い精度で位置計算ができる（Bat System では 95% の確率で 3cm 以内の誤差）が、複数の受信機（もしくは送信機）を正確な位置に設置しなければならないのが欠点である。

ラジオ電波を使うものとしては、カーナビに使われている GPS システムや Microsoft Research が開発した RADAR[8] などがある。RADAR は無線 LAN の信号を利用する。フロアの複数点に受信機を設置し、フロア各所における無線 LAN の信号のＳＮ比をあらかじめ測定しデータベースを作っておくことにより、ユーザの位置を調べることができる。誤差は、25% の確率で 1.92m、50% の確率で 2.94m、75%の確率で 4.69m である[8]。このシステムの欠点は、決して小さくはない無線 LAN デバイスをユーザが持ち運ばなければならないことと、物の配置など環境が

変わるごとにデータベースを再構築しなければならないことである。

位置検出用のデバイスを人間に装備させるタイプの位置追跡システムに共通の欠点は、ユーザがこのようなデバイスを装着しないと位置が検出できないことである。位置検出の対象となるユーザ全てに位置検出用のデバイスを用意しなければならない。また、赤外線、超音波などを使った位置検出用のデバイスは研究室レベルで開発されたものであり、一般的に入手することはできない。モーションキャプチャシステムなどは入手可能であるが、一般的に非常に高価である。一方、各ユーザに固有ＩＤを持ったデバイスを持たせることにより、完全に人間を識別することができることが利点である。見た目で非常に似通った人物もこのようなシステムにおいては簡単に識別することができる。

(2-2) カメラ画像利用型

カメラ画像を利用した位置追跡システムはたくさんある。また、カメラ画像による監視システムを含めるとさらに多くの研究例がある。ここではそのうちいくつかの例を挙げるにとどめる。

EasyLiving は Microsoft Research が開発した位置追跡システムで、２台の３眼カメラを使って得られる奥行きの情報などを利用して人間の位置を求める。あらかじめ登録しておいた背景の奥行きと色情報を元に、背景から人間の各部位を抽出・再構成した後、カメラの相対的位置情報や人間の動きなどの情報を使って、最終的に人間の識別と位置の算出を行う。また、各人間に対して色ヒストグラムを記録していて、人間の識別に利用している[21][22]。ステレオ画像を使うことにより、近接している複数の人間を比較的簡単に分離することができるが、普通の単眼カメラで色情報だけを使って分離するのは難しい。ステレオ画像を利用したシステムには、他に TYZX system[23] などがある。特殊なカメラを使ったものとしては、全方位カメラを使った N-Ocular ステレオによる位置追跡システムがある[18][19][20]。

単眼のカメラを使った位置追跡システムとしては、分散協調視覚プロジェクト Active-Vision Agent(AVA) system[15] がある。AVA システムでは、カメラとＰＣからなる観測ステーションが基本構成単位となっていて、これらがネットワークで結ばれていて互いに情報を交換し合う構成になっている。各観測ステーションには「知覚」「行動」「通信」の機能を備えたエージェントプログラムが動いている。エージェントは人間を検出すると、その人間を追跡するためのエージェンシーというグループを周囲のエージェントとともに構成する。観測ステーション上のエージェントが互いに人間に関する情報を交換しあうことで、観測ステーション間での人間の安定したマッチングを行っている[17]。またこのシステムの拡張として、人間の安定した顔識別のための分散エージェントプログラムによる人間の顔の協調登録などが可能になっている[16]。

カメラ画像を利用した方法の利点は、位置検出用の専用デバイスをユーザが装着する必要がないので、比較的簡単にシステムを構成できることである。一方、ステレオカメラなど一般的でない特殊カメラを使わないと、画像の中で複数の人間がオーバラップしてしまった場合に、個々の人間を分離して識別するのが難しかったり、物陰に隠れてしまった人間の検出が困難であることが欠点である。また、画像だけから得られる情報は限られていることから、似通った人間がいる場

合、システム側で入れ違えてしまうということも考えられる。また、たとえ相対的に人間を区別できたとしても、デバイス使用型と違って絶対ＩＤで管理することができない。二人の人間がいるとして、あるシステム系ではＡとＢと識別され、別のシステム系ではＣとＤと識別されたとする。この場合、ＡはＣと対応するのかＤと対応するのかはわからない。このように、カメラ画像だけを利用するシステムで、ユーザの絶対識別を行うことは難しい。

(3) アーキテクチャ

ここでは本研究で検討した人間の位置追跡システムの概要と基本設計を説明する。

(3-1) システムの概要

関連研究で紹介したように、既にいくつもの位置追跡システムが存在するが、それぞれ長所と短所が存在する。モーションキャプチャのような特別なデバイスを使えば、誤差が数センチ以内に収まるような高い精度で位置追跡を行うことも可能であるが、そのような装置は一般的に大変高価である。一方、カメラ画像を利用した位置追跡システムは、やはりいくつかの欠点はあるものの安価に構成できる。そこで、我々はカメラ画像をベースにした位置追跡システムを構成する。位置追跡用のデバイスを人間に持たせないタイプの欠点の一つは、被追跡者のＩＤ取得ができないということである。カメラ画像だけだと、着ている服の色などから被追跡者を相対的に区別することができるが、見た目がそっくりな場合には区別できなかったり、追跡の途中で取り違えることも考えられる。そこで、被追跡者が個人識別のための何かしらのデバイスを所持している場合にはその識別情報を読みとり、以後その個人を保持しながら位置追跡を行い、個人の識別情報が得られない場合にも識別情報なしで追跡を行うシステムを構築する。個人識別のためのデバイスは、必ずしも高価なものである必要はない。識別カードや RFID といった安価なものでも構わない。

(3.2) 基本アーキテクチャ

(a) センサーノード

本システムにおいて、センサーと演算能力と通信機能をもったものをセンサーノードと呼び、このセンサーノードがシステムの基本構成単位になる。ここで、センサーとはビデオカメラや認証カードの読み取り装置などである。演算能力というのは、センサーからの情報を処理したり人間の位置を計算したりするために必要な機能を指しており、具体的にはＣＰＵとメモリを積んでいることを要求するものである。通信機能というのは、周囲のセンサーノードと通信を行うための機能のことであり、具体的には特に TCP/IP が使えることを要求している。

各センサーノードは、周囲のセンサーノードと情報を交換しながら協調して人間の位置追跡を行うが、システム全体を統括するようなサーバは存在しない。位置追跡の対象エリア内の各箇所に設置されたセンサーノードが、それぞれの領域にいる人間の位置追跡を行う。人間が別のセンサーノードがカバーする領域に移っていく場合には識別情報のハンドオフを行う。このような分散システムアーキテクチャを採用することで、位置追跡領域の大きさに対するスケーラビリティが

生まれ、原理的にはいくらでも対象領域を広げることができる。
本システムはカメラベースのシステムであるため、センサーノードには主としてカメラが付いているものを想定しているが、他の位置追跡システムをセンサーノードとして取り込むことも可能である。

(b) ノード上のタスク

各センサーノード上には、主として「センサータスク」「ネットワークタスク」「計算タスク」の３つのタスクが動いている。

センサータスク

センサータスクは、自ノードのセンサー情報を処理するタスクである。カメラの場合には観測された人間へのカメラからの角度や識別に利用する色情報などを処理し、RFID タグリーダーの場合には RFID の読みとりといったようにセンサーの種類ごとに処理の仕方は異なるが、抽出される情報は同じであり、位置に関する情報と識別に関する情報である。RFID タグリーダーの場合には、人間の位置を正確に捉えることはできないが、その代わりにその RFID タグリーダーの位置（もしくは想定される人間の位置）を返す。このような接触型の識別デバイスの場合には十分な近似的位置が求まる。
センサー情報の処理が終わると、検出された各人間に対して「タグ」（後述）という識別子をつける。センサータスクが付与するのはローカルタグというものであり、このタグの値はそのセンサーノード上でのみユニークな値になっている。センサータスクは自ノードのセンサーが得た特徴値を使って人間を識別してローカルタグを付与するので、このタグ付けの仕方はセンサーの種類ごとに異なる。カメラの場合には、色ヒストグラムを利用したり、人間の顔を識別できる場合にはその情報を使ってローカルタグを決定する。RFID タグリーダーの場合には、RFID タグの値をそのままローカルタグに使うことができる。センサータスクは、特徴値とローカルタグの対応表（ローカルタグテーブル）を保持する。得られた特徴値がテーブル内のどのエントリーにもマッチしない場合には、新しいローカルタグとともにテーブルに登録される。一つのローカルタグが毎回、同一人物に付与されることが理想であるが、カメラ画像において色ヒストグラムだけを使って、似通った服を着た人間には同じローカルタグを付与してしまう可能性があり、この場合、システム内で人間の取り違いが起きてしまう。カメラ画像だけを使っている場合には、こういう事態を完全に避けるのは難しいが、後述するローカルタグの組み合わせを行うことにより、この被追跡者の取り違いを起こりにくくすることができる。また、厳しく人物の識別を行った場合には、同じ人物であっても、毎回異なるローカルタグを付与してしまうことが考えられる。これも望ましいことではないものの、異なる人物に同じローカルタグを付与してしまうよりはよい。
ローカルタグ付与の処理をした後、最後に検出された人間すべてに対する「位置情報＋ローカルタグ」を周囲のセンサーノードに送信する。

ネットワークタスク

ネットワークタスクは、周囲のセンサーノードからの送られてきたパケットを

処理するためのタスクである。センサーノードにやってくるパケットには、周囲のセンサーノードから送られてくる被追跡者の情報が含まれているパケットの他にも、動作をコントロールしたり状態を参照したりするためのパケットがある。ネットワークタスクは、そういったデータパケット以外のリクエストを受け付けて処理する一種のサーバタスクである。周囲のセンサーノード上のセンサータスクが送信してくるパケットについては、受け付けてパースして計算タスクに渡す。実際の処理は計算タスクが行う。

計算タスク

計算タスクは、自ノード上のセンサータスクからの情報の他に、ネットワークタスクを介して他ノード上のセンサータスクから送られてくる情報を取得し、それら情報を統合・処理する。

それぞれのセンサータスクが送信してきた各被追跡者に対するローカルタグの処理はここで行われる。まず、位置情報をもとに被追跡者ごとに情報が整理される。次に被追跡者ごとにローカルタグの組み合わせが作られる。このローカルタグは前述したようにそれを付けたセンサーノード上でのみユニークな値である。したがって、各ローカルタグの組み合わせは、あらかじめ決められたセンサーノードＩＤの順番に並び替えられる。そうすることによって、この組み合わせはシステム全体でユニークなものとなる。計算タスクは、このローカルタグの組み合わせを使って個人識別を行い、グローバルタグ（後述）情報が存在する場合にはグローバルタグを決定する。計算タスクは、このローカルタグの組み合わせを保持するテーブル（コンビネーションテーブル）を持っていて、得られたローカルタグの組み合わせと最も近い組み合わせをこのテーブル内から探し出す。どのエントリーともマッチしない場合には、新しいエントリーとしてコンビネーションテーブルに追加する。マッチした場合には、そのエントリーに関連づけられている情報（もし存在すれば）を、その被追跡者の位置情報にして追加して、位置情報を利用するクライアントに結果を送信する。

(c) タグによる人物の識別

ローカルタグの組み合わせによる訂正

本システムでは、各被追跡者に対して各センサーノードがローカルタグという識別子を付けて、更にそのローカルタグの組み合わせを取り、その組み合わせを利用して被追跡者の識別を行う。こうすることにより、あるセンサーノードが間違ったローカルタグを付与した場合にも、それ以外のセンサーノードが正しいローカルタグを付与していれば間違いを訂正することができる。

sensor node 1	sensor node 2	sensor node 3	global ID
12 / 4	11 / 2	29 / 0	25
13 / 3	14 / 3	13 / 1	42
14 / 2	**12 / 1**	**15 / 2**	**16**
11 / 4	17 / 2	14 / 0	90
16 / 1	20 / 1	17 / 0	0

Table 5.3.1: コンビネーションテーブルの例１

sensor node 1	sensor node 2	sensor node 3	global ID
12 / 4	11 / 2	29 / 0	25
13 / 3	14 / 3	13 / 1	42
14 / 3	**12 / 2**	**15 / 1**	**16**
11 / 4	17 / 2	14 / 0	90
16 / 1	20 / 1	17 / 0	0

Table 5.3.2: コンビネーションテーブルの例２

Table 5.3.1 は、コンビネーションテーブルの例である。一番上のエントリーは、センサーノード１、２、３がそれぞれ１２、１１，２９というローカルタグ（/で区切られた左側の数字）を付与した人物に対応している。エントリー中で各ローカルタグの横にある数字（/で区切られた右側の数字）は、それぞれのローカルタグがこの組み合わせにおいて何回出現したかを表している。ここで、ある被追跡者に対してセンサーノード１、２、３がそれぞれ１４、１２、１６というローカルタグを付与したとする。計算タスクはコンビネーションテーブルを調べ、もっともこの組み合わせに近いエントリーを探し出す（上から３番目のエントリー)。センサーノード３ のローカルタグは一致しないものの、センサーノード１、２のローカルタグは一致したために、センサーノード３は間違ったローカルタグを付与したと判断することができる。また、コンビネーションテーブルは Table 5.3.2 のように更新される。

グローバルタグの伝搬

計算タスクはローカルタグの組み合わせに対して、システム全体でユニークなグローバルタグを付与する。ここで、グローバルタグは必ずしも存在しない。被追跡者が RFID などどのリーダーで読みとっても間違いなく同じ値を返すものを身につけていてセンサーがそれを読みとることができる場合にのみ、センサーノードはグローバルタグを発行できる。センサータスクは、検出した被追跡者に常にローカルタグを付与する一方で、グローバルタグは取得できるセンサータスクのみがローカルタグに加えて付与する。

計算タスクが得た情報の中にグローバルタグの情報が含まれている場合には、

そのグローバルを付与された被追跡者に対応するローカルタグの組み合わせに関連づけられて、コンビネーションテーブルに保持される。このローカルタグの組み合わせの中に自ノードのセンサータスクが付与したローカルタグが含まれている場合には、ローカルタグテーブルの該当するエントリーが、そのローカルタグの組み合わせに対応するコンビネーションテーブルのエントリーに関連付けられる。このようにすることにより、次回センサータスクがそのローカルタグを付与する時にグローバルタグも一緒に付与することができるようになり、被追跡者が移動するにともなって、その被追跡者に付与されたグローバルタグも移動経路に沿って配置されているセンサーノードに伝搬していく。

Fig.5.3.3 では、コンビネーションテーブルの一つのエントリーに対して複数のローカルタグが関連づけられている。前述したが、違う人間に対して同じローカルタグを付与することはエラーであるが、一人の人間に対して複数のローカルタグを付与することは望ましくはないもののエラーではない。複数のローカルタグが同じグローバルタグと関連づけられることも問題ない。

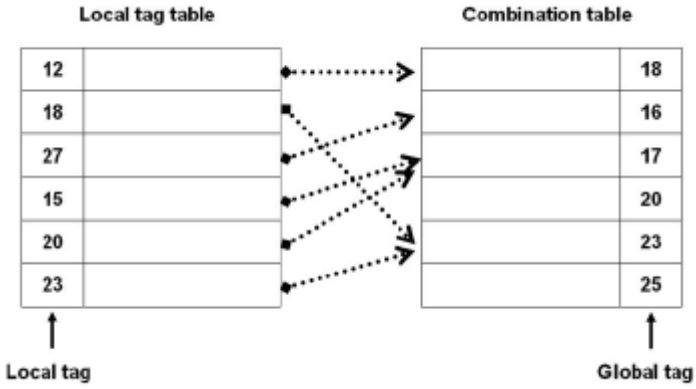


Figure 5.3.3 ローカルタグテーブルからコンビネーションテーブルへのリンク

(4) 実装

今回構築したシステムについて説明する。

(4.1) パイロットシステム環境

今回、Table 5.3.4 に示すマシンを使ってパイロットシステムを構築した。カメラセンサ(Fig5.3.5)は昨年度の研究で開発した据え置き型ノードコンピュータ(μU-Card)を利用して開発した。また識別情報を得るためのセンサノードは、RFID タグリーダーの代用として、モーショントラッカー(Fig.5.3.6)を用いた。代用した場合でもシステムアーキテクチャ自体は変わらない。なお本システムでは、安価に入手できるカメラをベースとした位置追跡システムを想定しているが、他の位置追跡システムからの情報も利用することが可能な設計となっている。

各センサーノードの配置図を Fig 5.3.7 に示す。四角で示しているのは、モーショントラッカーが人物識別及び位置追跡をすることができる領域である。もちろん、位置検出用のデバイスを所持していなければ位置は検出されない。一方、丸で示したのはセンサーノードＩＤが 13、14、15、16 のカメラノードであり矢印

の方向を向いている。

センサーノード	カメラ	モーショントラッカ
CPU	M32104(32bi/216MHz)	Pentium4 (1.5GHz)
メモリ	8M バイト	640M バイト
OS	ITRON	Turbo Linux 7.0
ノード個数	4	1

Table 5.3.4 パイロットシステムに使用したマシン

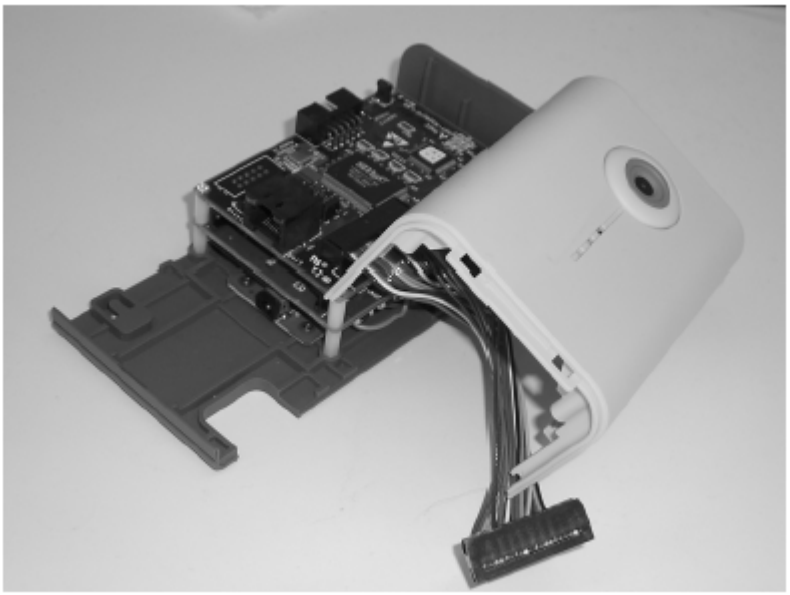


Figure 5.3.5 μU-Card

Figure 5.3.6 Motion Tracker

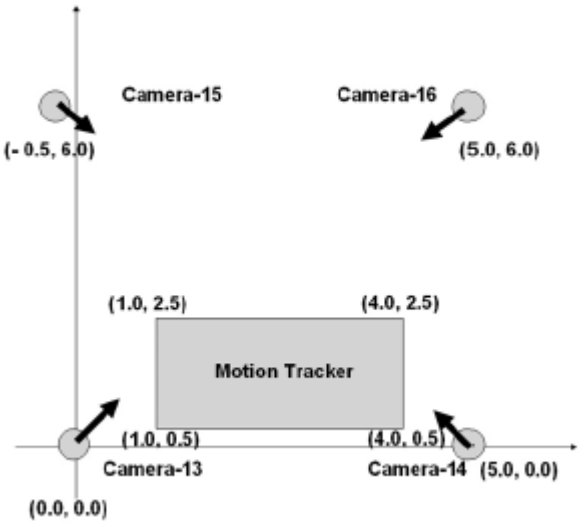


Figure 5.3.7 センサーノードの配置

５つあるセンサーノードは同一の LAN 上にあるが、今回はカメラセンサーノードは全てのカメラセンサーノードと通信するが、モーショントラッカーノードは、図のセンサーノードＩＤが１３と１４のカメラノードとしか通信を行わないようにした。これは、モーショントラッカーノードが発行したグローバルタグが、カメラノード１３及び１４を通して、カメラノード１５と１６に伝わるのを確認するためである。

(4.2) センサーノード上のタスクの実装

今回構築したパイロットシステムでは、前述したようにカメラとモーショントラッカーの２種類のセンサーノードが存在する。４台のカメラセンサーノードには同じプログラムが動いている。前述したように、本アーキテクチャにおいてはセンサーノード上に、センサータスク、ネットワークタスク、計算タスクの３つのタスクが基本的に動いているが、モーショントラッカーについては、もともと RFID リーダーといったシンプルなデバイスの代わりとして使っているので、センサータスクのみを実装した。

本来、センサーノードは他のセンサーノードからの情報がなくても自ノードのセンサー情報（カメラノードの場合にはカメラ画像）だけから、位置及び人物の識別が行えるのが理想である。しかし、今回のパイロットシステムにおいては、実装を簡単にするために、カメラノードは周囲のカメラノードと情報を交換することによって人物の位置算出をする形をとった。具体的には、複数の観測点（カメラノード）からの角度情報を使って人物の位置を算出している。

(a) センサータスク

モーショントラッカーノード上

モーショントラッカーノード上のセンサータスクは非常にシンプルなもので、モーショントラッカー自身が位置検出用のデバイスの識別ＩＤと位置情報を取得してくれるので、センサータスクはそれらを受け取り識別ＩＤからローカルタグとグローバルタグを生成して位置情報とともに、カメラセンサーノードに投げる。

カメラセンサーノード上

カメラ画像を処理して、写っているそれぞれの人間のカメラノードから見た観測角度を算出する。また、それぞれの人間について色情報も取得し、この情報をもとにローカルタグを決定する。具体的には、RGB 色情報とローカルタグの組をエントリーとするローカルタグテーブルを用意し、色情報を獲得するとこのテーブルを参照することによりローカルタグを決定している。取得した画像中に検出された各人間について、角度情報とローカルタグを求めた後、それらの情報を周囲のセンサーノードに送信する。

画像中の人間の角度算出は背景差分法による（Fig 5.3.8）。人間の位置追跡を始める前に人間がいない状態の背景を撮影しておき、以後この背景画像と取得した画像の差分をとることによって、人間の写っている部分を切り抜くことができる。次にこの部分を垂直方向に射影する。すると、人間が写っている部分は”山”として浮かび上がる。この山の中心に対する角度を、その人間のカメラから見た角度として算出する。画像中で複数の人物がオーバラップしている場合には、この方法では一人の人物の角度としてしか算出できないが、他の観測点（カメラセンサーノード）からの画像に分離して写っている場合には、それぞれの人物の位置を計算できる可能性がある。

Figure 5.3.8 背景差分方

(b) ネットワークタスク

アーキテクチャの説明で述べたが、このタスクは周囲のセンサーノードが送信してくるデータパケットをパースして計算タスクが利用できる形にする。また、センサーノードの動作を変えたりするための命令パケットの処理も行うサーバタスクになっている。

(c) 計算タスク

本パイロットシステムでは、人物の位置計算はカメラノード上の計算タスクが行う。カメラノード１３と１４は、すべてのカメラノードからの情報に加えてモーショントラッカーからの情報も受け取ることができる。カメラノード１５と１６は、モーショントラッカーからの情報を受け取ることができない。

カメラノード上の計算タスクの処理は、大きく３つのステップに分けることができる。各人物の位置計算、各人間の識別、結果の送信である。まず、ネットワークタスクに介して集められた、周囲のカメラセンサーノードからの各人物への観測角度を使って位置計算を行う。カメラノードからの人間の観測角度が得られると、そのカメラノードと人間を通る直線が決まる。複数のカメラノードからの角度情報をもとにこのような直線が複数決まると、人物の位置はこの直線の交わる点として求めることができる。しかし、複数の人物がいる場合には、Fig 5.3.9 に示すように位置が一意に決定できないことがある。本方式で位置計算をする限り、根本的にこの問題を解決することは難しいが、観測点（カメラノード）が増やすことである程度はこの問題に対処可能である。本パイロットシステムでは、３つ以上の直線が交わる点に人物が存在すると判定している。

各人物の位置が確定すると、次にそれらの人物の識別処理が行われる。交点として位置を求められた各人間について、関連づけられたローカルタグの組み合わせ作り、コンビネーションテーブルからこの組み合わせに最もマッチするエントリーを探す。ここでローカルタグの２つの組み合わせを（ある決まったセンサーノードＩＤ の順番に並んでいるとする。また、ローカルタグの組み合わせにおいて全てのセンサーノードが対応する人物を検出しローカルタグを含む情報を送ってくるとは限らないので、組み合わせにおけるそのセンサーノードのローカルタグ要素は「空」とする。）

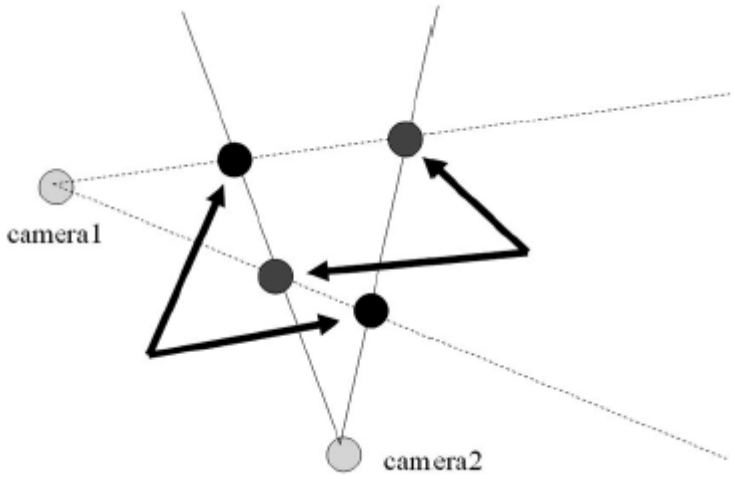


図 5.3.9 二つの可能性

$$Comb_A = (a_1, a_2, ..., a_n) \quad (4.1)$$

$$Comb_B = (b_1, b_2, ..., b_n) \quad (4.2)$$

と表現すると、以下の式の値の大きさでそれらの一致度を測っている。

$$\sum_{k=1}^{n} \Psi_{a_k, b_k}$$

ただし、

$$\Psi_{i,j} = \begin{cases} 0 & \text{if } i = \phi \text{ or } j = \phi \\ +1 & \text{if } i = j \\ -1 & \text{if } i \neq j \end{cases}$$

マッチしたエントリーがコンビネーションテーブルに見つかった場合には、そのエントリーを更新し、ローカルタグの組み合わせにおける自ノードが付与したローカルタグに対応するローカルタグテーブルのエントリーから、そのマッチしたコンビネーションテーブルのエントリーへのリンクを張る。こうすることで、（もしコンビネーションテーブルのエントリーに対応付けされているグローバルタグが存在すれば）ローカルタグからグローバルタグへの対応付けが行われる。次回、センサータスクがこのローカルタグテーブルのエントリーにアクセスした時には、（もし存在すれば）グローバルタグを獲得することができる。

ローカルタグの組み合わせに対してグローバルタグ情報がある場合には、リングバッファ（コンビネーションテーブルの各エントリーに存在する）に格納して、そのリングバッファ内でもっとも多く現れるグローバルタグをそのエントリーのグローバルタグとして採用する。このようにするのは、センサーノードの一つが人物の識別を誤った場合でも、他のセンサーノードが正しい対応付けを行っていれば（正しいグローバルタグを付与していれば)、正しい対応付けができるよ

うにするためである。また、グローバルタグ付けの信頼性を向上させるため、リングバッファ内においてあるグローバルタグが最も多く現れる場合であっても、その出現回数が閾値を越えない場合には、そのローカルタグの組み合わせに対するグローバルタグへの対応付けは見送るようにした。

最後にクライアントノードへ各被追跡者の識別情報（あればグローバルタグ）及び位置情報をを送信する。

(5) 考察

(5.1) 位置追跡の様子

パイロットシステムで、実際に人間が歩いた時の軌跡を Fig 5.3.10 と Fig 5.3.11 に示す。Fig 5.3.10 は、カメラノードの位置も含めたものであり、

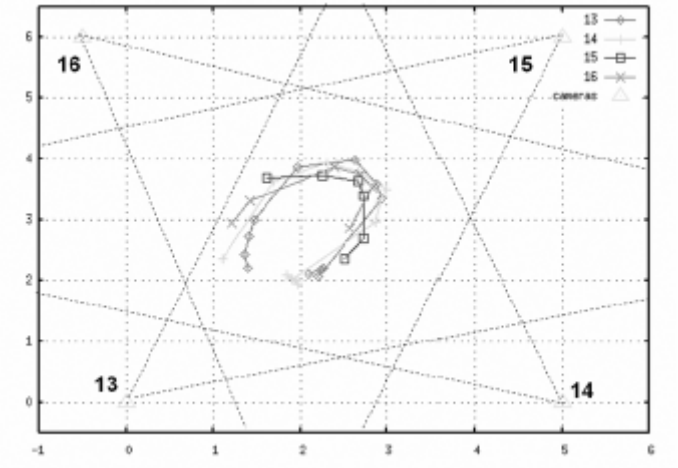


Figure 5.3.10 人間の軌跡

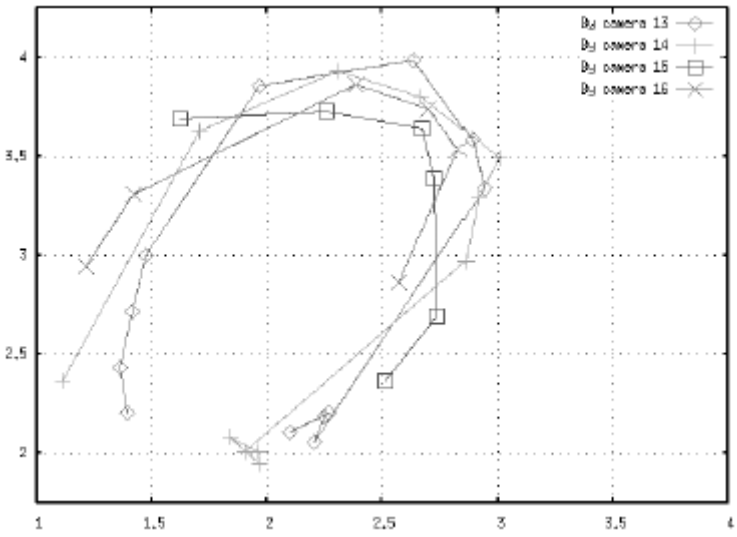


Figure 5.3.11 人間の軌跡(拡大図)

Fig 5.3.11 はその拡大図である。図中には描かれていないが実際の人間の位置

に対して、いづれのカメラノードもおよそ誤差 30cm くらいで位置を検出できた。アプリケーションが人間の位置情報をどんな形で、利用するかにもよるが、多くの場合十分な精度であると考えられる。

本位置追跡システムのアーキテクチャは複数人物の追跡に対応しているが、今回の構築したパイロットシステムでは、残念ながら複数人間に追跡については満足の行く結果は得られなかった。人間の立つ位置にもよるが、多くの場合において複数台のカメラの画像に重なって写ってしまい、各人間について正しい観測角度が得られないためである。これを克服するためにカメラの数を増やすことも考えられるが、この場合には各カメラノードにおいて交点の算出などの処理が重くなる。また、カメラノードと人間を通る直線の数が増えることにより、実際には人間が存在しないところにも交点がたまたま密集してしまい、そこに人間がいると判断してしまうことも起きる。更に、人間の数が増えるに従って、交点の数も指数関数的に増加してしまい、やはり実際には人間が存在しない位置に交点が密集することも起きてくる。

(5.2) 測定誤差による人物識別への影響

今回のパイロットシステムにおいて、カメラノードは色を使って人間を識別している。しかし、各人間に対していつも同じ色を取得できるわけではなく、そこには誤差が絶えず生じている。同じ部屋であっても場所によって、照明のあたり具合が違ったり、また複数の色が使われていて見る角度を変えると違った色が見える服を着用している場合には、特にこの傾向が顕著に現れる。そこで、色の測定誤差がどれくらい結果に影響を与えるのかを調べるため、追跡する人間２人（それぞれがフィールド上をランダムに歩く）の場合のシミュレーションを行った。

各カメラノードとモーショントラッカーノードの配置はパイロットシステムと同じであり、人間は直径 50cm の２次元の円として近似した。カメラノードは、毎回、各人間の（固定の）RGB の各色成分に対してランダムに§1～§4 の値を変化させた値を取得させた。また、２つの RGB データに対して、各成分の差の絶対値の合計が 10 を越えた時、違う人物であると判定するようにした。２人の人間の RGB 色は（15,15,15）と（20,20,20）とした。

Table 5.3.12 は、カメラノードの色の測定誤差が小さい場合（各 RGB 成分について誤差が§1）と大きい場合（各 RGB 成分について誤差が§1)、さらにそれらが混じった場合について、カメラノード１６におけるローカルタグからグローバルタグへの対応付けについて、５０回目の位置検出をした際のローカルタグテーブルの様子についてのシミュレーション結果（１０回行った平均値）を示したものである。「小３ 大１」というのは、誤差が小さいカメラが３台、大きいカメラが１台の場合の結果を表す。また、計算タスクの実装の所で述べたが、ローカルタグの組み合わせに対するグローバルタグの対応付けは、そのグローバルタグがリングバッファ内である一定以上の回数だけ出現しないと行われない。グローバルタグが伝搬していない場合にも、ローカルタグからグローバルタグへの対応付けはない。図中の「対応付け無」はこの２つの状況を含んでいる。２人の人間に対して、ローカルタグの合計が２つ以上存在するのは、一人の人間に対して複数のローカルタグが付与されていることを示している。しかし、前述したように一人の人間に複数のローカルタグが付与されること自体に問題はないが、一人の人間

に複数のグローバルタグが対応付けされるのは、識別がうまくいっていないことを示している。

この結果から、識別率が高い場合（「小４大０」）にはカメラノードに正しくグローバルタグが伝播し、識別率が低い場合（「小０大４」）には誤った対応付けがされることがあることが読みとれる。一方、識別率が低いカメラが一台ある場合（「小３ 大１」）には、全てのカメラノードの識別率が低い状況に比べて誤った対応付けが少ないことがわかる。れは、人間の識別に関して、一台のカメラノードのみの情報（ローカルタグ）を使って人間を識別するのではなく、周囲のカメラノードの情報と組み合わせること（ローカルタグの組み合わせ）により、より高い確率で正しく人物の識別が行われることを示していると考えられる。

	正	誤	対応付無し
小４大０	2.0	0.0	0.0
小０大４	3.5	1.8	1.3
小３大１	3.3	0.7	2.0

Table 5.3.12 ローカルタグからグローバルタグへの対応付け

(6) まとめ

カメラベースの分散協調型の位置追跡システムを提案した。位置追跡用のデバイスを人間に所持させるタイプの位置追跡システムでは、被追跡者全員にデバイスを用意する必要があり、またそのような機材は一般的に入手困難であったり大変高価であるという欠点があり、カメラ画像だけを利用した位置追跡システムでは相対的な区別は可能でも、絶対的な識別は難しいという欠点があった。我々は、RFID タグなどといった安価に入手できるものを利用して個人の識別子を取得し、それをカメラベースの位置追跡システムへハンドオフすることができる位置追跡システムを提案した。

今回構築したパイロットシステムでは、残念ながら複数の人間の位置追跡ができなかった。これは、カメラ画像において複数の人物が重なってしまった場合、分離してそれぞれに対する観測角度を測ることが出来なかったことに起因する。また、カメラノードと人間を通る直線群の交点として人間の位置を求める方法は、センサーノードの独立性を損なうことになり、依存関係にあるカメラノードが停止してしまった場合には、自ノードにおいて人物の位置計算ができなくなる。また、カメラノードにおいて色情報をつかって人物の識別を行っているが、更に人物の移動速度や前の位置などといった特徴値を利用すれば、さらに安定した識別を行うことができると考えられる。

参考文献

[1] Jeffrey Hightower and Gaetano Borriello, "A Survey and Taxonomy of Location Sensing Systems for Ubiquitous Computing," UW CSE 01-08-03, University of Washington, Department of Computer Science and Engineering, Seattle, WA, Aug. 2001.

[2] Want R., Hopper A., Falcao A. and Gibbons J. "The Active Badge Location

System". ACM Transactions on Information Systems, vol. 10, no. 1, pp. 91-102, anuary 1992.
[3] http://www.ubiq.com/parctab/parctab.html
[4] Addlesee M., Curwen R., Hodges S., Newman J., Steggles P., Ward A. and Hopper A. "Implementing a Sentient Computing System・ IEEE Computer, vol. 34, no. 8, pp. 50-56, August 2001.
[5] http://www.uk.research.att.com/ab.html
[6] Pryantha N.B., Chakraborty A. and Balakrishnan H. "The Cricket locationsupport system" Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM00), Boston, MA, August 2000.
[7] Nissanka Bodhi Priyantha, Allen K. L. Miu, Hari Balakrishnan, Seth Teller "The Cricket Compass for Context-Aware Mobile Applications・ Proc. of the 6th ACM MOBICOM Conf., Rome, Italy, July 2001.
[8] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System" Proceedings of IEEE Infocom 2000, Tel-Aviv, Israel, March 2000
[9] Orr R. and Abowd G. D. "The Smart Floor: A mechanism for natural user identification and tracking". Proceedings of the 2000 Conference on Human Factors in Computing Systems(CHI 2000), The Hague, Netherlands, April 2000.
[10] Sanjay Sarma, "Towards the 5 cents Tag" http://www.autoidcenter.org/
[11] Schilit, B., Adams, N., Want, R. "Context-Aware Computing Applications" Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. pp.85-90
[12] G. D. Abowd, A. K. Dey, R. Orr, and J. A. Brotherton. "Context-awareness in wearable and ubiquitous computing" ISWC, pages 179-180, 1997.
[13] Dey, A.K., et al. "The Conference Assistant: Combining Context-Awareness with Wearable Computing," in Proceedings of the 3rd International Symposium on Wearable Computers. 1999. San Francisco, CA: pp.21-28
[14] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. "Cyberguide: A mobile context-aware tour guide." ACM Wireless Networks, 3:421-433, 1997.
[15] 松山隆司, 和田俊和, 丸山昌之"能動視覚エージェントによる異動対象の協調的追跡" MIRU'98 pp.I-365-370 ,1998.
[16] 加藤丈和, 向川康博, 尺長健"安定な顔認識のための分散強調登録" 電子情報通信学会論文誌 vol-J84-D-II, no.3, pp.500-508, 2001.
[17] 中澤篤志, 日浦慎作, 加藤博一, 井口征士"分散視覚エージェントを用いた複数人物追跡システム" 情報処理学会論文誌第 42 巻第 11 号, 2001.
[18] Takushi Sogo, Hiroshi Ishiguro, Mohan M.Trivedi "Real-Time Target Localization and Tracking by N-Ocular Stereo" IEEE Workshop on

Omnidirectional Vision (OMNIVIS' 00)
[19] Takushi Sogo, Hiroshi Ishiguro, Mohan M.Trivedi "N-Ocular Stereo for Real-Time Human Tracking", Panoramic Vision: Sensors, Theory, and Applications (R. Benosman and S.B.Kang, Editors), Springer Verlag, 2000.
[20] Kim C. Ng, H. Ishiguro, Mohan M. Trivedi , and T. Sogo, "Monitoring Dynamically Changing Environments by Ubiquitous Vision System," IEEE Workshop on Visual Surveillance , Fort Collins, Colorado, June 1999.
[21] Krumm J., Harris S., Meyers B., Brumitt B., Hale M. and Shafer S. Multi-Camera Multi-Person Tracking for EasyLiving・Proceedings of 3rd International Conference on Visual Surveillance, Dublin, Ireland, pp. 3-10, July 2000.
[22] Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. "EasyLiving:Technologies for Intelligent Environments・ Proceedings of the International Conference on Handheld and Ubiquitous Computing 2000, pp. 12-27, September 2000.
[23] http://www.tyzx.com/
[24] Ismail Haritaoglu, David Harwood and Larry S. Davis "W4: Real-Time Surveillance of People and Their Activities"", IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 22, No. 8, August 2000 pp. 809-830

5.4 サーバーノードシステム技術

5.4.1 ユビキタス PKI

ユビキタス環境における PKI の実現のために、今年度は従来の PKI 証明書(X.509) と比較してリソースの少ないユビキタス機器で利用できる証明書の研究を行った。本節ではその研究内容について述べる。

（ア） 証明書の形式

ここで検討すべき証明書には X.509 ASN.1 Basic Encoding Rule, Packed Encoding Rule, XML および SPKI などがある。

（イ） 証明書プロファイルにおける基本要件

リソースが少ないユビキタス機器で利用するためには証明書はコンパクトなものである必要がある。そのためにまず証明書が用いられる条件（証明書プロファイル）を定める。ここではそのための基本要件を以下のとおりとした。

20. 認証の対象が何であるか（デバイス認証、個人認証）
    デバイス認証を基本とする(IC カード相当の耐タンパーデバイスを想定)。但し、個人認証も別途検討する。
21. 1 つの CA が管理する証明書の枚数:シリアル番号の番号空間を決定する要因となる。
    32bit 長の integer とする。
22. 発行者、公開鍵所有者の名前規則
    発行者や所有者のフィールド領域の大きさを決定する要因となる。また、代替名を使用する場合、証明書拡張領域のサイズを決定する要因となる。Issuer（発行者）や subject（公開鍵所有者）の名前表現は、証明書サイズに大きく影響を与えるため、できるだけ簡便な名前が望ましい。e-mail や URL など個人との紐づけが弱い証明書については、本属性の代わりに SubjectAltName を用いる場合がある。この場合は、subject を null にし、extensions の属性を critical にする必要がある。

issuer、subject のデータ構造

```
Name ::= CHOICE { -- only one possibility for now --
 rdnSequence  RDNSequence }
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
```

```
DistinguishedName ::=   RDNSequence
RelativeDistinguishedName  ::=
     SET SIZE (1 .. MAX) OF AttributeTypeAndValue
AttributeTypeAndValue           ::=     SEQUENCE {
     type    AttributeType,
     value   AttributeValue }
 AttributeType           ::=  OBJECT IDENTIFIER
 AttributeValue          ::=  ANY
```

ここでは公開鍵所有者の構造は、CN,OU,O,C とする。以下に例を示す。
CN=Toshiaki Tanaka, (MAX32bytes)
OU=Network Security Laboratory, (MAX32bytes)
O=KDD R & D Labs Inc., (MAX32bytes)
C=JP

発行者の構造は OU,O,C とする。以下に例を示す。
OU=Network Security Laboratory, (MAX32bytes)
O=KDD R & D Labs Inc., (MAX32bytes)
C=JP

23. 証明書利用アプリケーション：証明書の key 利用の目的を決定する要因となる（証明書サイズには影響ない)。
    key-encipherment を default とする。
24. ドメイン構成：証明書の検証パスを決定する要因となる。すなわち、ユビキタス端末側での処理に関わる。
    5 章での考察を参照とする。
25. 署名アルゴリズムおよびその鍵長：証明書サイズおよび、証明書検証処理量を決定する要因となる。但し、鍵長は一般的に安全と言われている値にする必要がある。また、電子署名に用いる場合は、証拠情報となるため長期保存に関する考慮が必要となる。
    SHA-1 with RSA（RSA 鍵長 1024bit）あるいは ECDSA（160bit）とする。
26. 証明書失効管理手法：証明書の失効管理の有無、および失効管理手法についてである。CRL を端末側で管理するかどうかにより、端末側の記憶容量の制限を考慮する必要がある。
    ユビキタス端末内に CRL を保有することは困難であるため、問い合わせ型の有効確認技術を用いる。具体的なプロトコルとしては、

OCSP（RFC 2560）がある。OCSP は証明書の有効性確認だけだが、認証パスの構築・検証および必要に応じて認証パスを構成する証明書（複数）の有効性確認をサーバサイドで行うためのプロトコルとして、DPV （Delegated Path Validation）と DPD（Delegated Path Discovery）機能の OCSP への拡張や、DPS+DPD を実現する新たなプロトコルとして CVP（Certificate Validation Protocol）が提案されている。

27. 認証局における鍵更新
    認証局における署名鍵更新の有無およびその方式についてである。証明書の拡張領域のサイズを決定する要因となる。安全性を考慮して鍵の更新を定期的に行うことを想定する。
28. 証明書ポリシーの扱い：各ドメインに証明書の発行、運用ポリシーが定められる。
    各ドメインに唯一のポリシーが存在し、オブジェクト ID で定義する。
29. PMI（Privilege Management Infrastructure）、AC（Attribute Certificate）を用いたアクセス制御の可能性
    今回の調査の対象外とする。

これらの前提をもとに、証明書サイズの評価を行う。証明書サイズは"variable"が理想である。

以下に証明書の実例を載せる（但し、データサイズの概算を行うことを目的としており、完全なデータ形式ではない）。

- Extension については、Basic Constraints のみ包含
- 全体サイズ　612 バイト
- エンコーディング方法　ASN.1 BER（Basic Encoding Rule）

図 5.4-1: ASN.1 BER による証明書の例（通常表示）

（ウ） 証明書基本プロファイル

以下に証明書および CRL のプロファイル案を示す。表中で最右カラムは、本プロファイル案での使用（Y）／未使用（N）を表す。

表 5.4-1: 証明書データ構造（extensions の詳細は別）

	ITU-T X.509 4th	Description	
Certificate			
version	必須	Extension を用いる場合は v.3	
serialNumber	必須	認証局が管理する証明書発行番号	
signature	必須	署名のアルゴリズム ID を指定	
issuer	必須	発行者名：表記は運用により決定	
validity	必須	有効期間	
subject	必須	ユーザ名：表記は運用により決定	
subjectPublicKeyInfo	必須	公開鍵情報	
issuerUniqueIdentifier	オプション	発行者名を再利用する際に使用	N
subjectUniqueIdentifier	オプション	ユーザ名を再利用する際に使用	N
extensions	オプション	拡張機能の利用時に使用	Y

表 5.4-2: CRL のデータ構造

	ITU-T X.509 4th	Description	
CertificateList			
version	必須	Extension のいずれかが critical の場合 v.2	
signature	必須	署名のアルゴリズム ID を指定	
issuer	必須	発行者名：表記は運用により決定	
thisUpdate	必須	失効リストが発行された日時	
nextUpdate	オプション	次の失効リストが発行される日時	N
revokedCertificates	オプション		
serialNumber	必須	証明書のシリアル番号	
revocationDate	必須	証明書が失効した日時	
crlEntryExtensions	オプション	拡張機能の利用時に使用	N
crlExtensions	オプション	拡張機能の利用時に使用	N

表 5. 4-3: 公開鍵証明書の拡張 1

		Description	
Key and policy information extension	Authority key identifier	証明書や CRL の検証用の鍵を識別する情報。例えば、鍵の更新が行われて、同一の CA から複数の鍵が利用されている場合に区別するための ID を表す。	Y
	Subject key identifier	証明すべき鍵を識別する情報。例えば、鍵の更新が行われて、同一の証明者が複数の鍵を利用している場合に区別するための ID を表す。	N
	Key usage	鍵の利用目的を定め、それ以外の利用を制限するための情報。例えば、署名鍵や、暗号化鍵など。	Y
	Extended key usage	上記以外の個別の目的で鍵の利用を制限する場合の情報。例えば、Microsoft は、クライアント認証用の識別子などを追加している。	N
	Private key usage period	公開鍵に対応する秘密鍵の有効期間。電子署名用の鍵のみに利用される。	N
	Certificate Policies	証明書の発行ポリシーや利用ポリシーを記述する情報。ポリシー識別子と修飾子のリストで構成される。目的は証明書の Validation Path を検証するために利用される。RFC249 においては、具体的な利用方法として、ポリシーを識別する OID と CPS（Certificate Practice Statement）をポイントする URL 情報および、利用者に表示するためのテキスト情報が含まれる。	Y
	Policy mappings	CA 証明書のみに利用される。発行者の証明書ポリシーのいくつかが、認証者の CA ドメインの証明書ポリシーと等価であることを、証明書発行者に示す。	Y

表 5. 4-4: 公開鍵証明書の拡張 2

		Description	
Subject and	Subject alternative name	公開鍵に紐づくユーザの代替名。e-mail や URL など厳格な個人との紐づけが弱い証明書については、本属性の代わりに SubjectAltName を用いる場合がある。	N

	Issuer alternative name	証明書や CRL 発行者の代替名。	N
	Subject directory attributes	証明書のユーザに対するディレクトリ属性を含む。	N

表 5.4-5：公開鍵証明書の拡張 3

		Description	
Certification path	Basic constraints	証明書のパス検証において、適用するパス長の制約（最大パス長）を記載する。	Y
	Name constraints	CA 証明書に利用される。証明書パスにある連続する証明書群のすべてのユーザ名に適用される名前空間。インターネットのドメイン名と同様の名前管理。	N
	Policy constraints	特定の証明書ポリシーを必要とする。あるいは、ポリシーマッピングを禁止するための情報。	N
	Inhibit any policy	特殊なポリシーである anypolicy の禁止。	N

表 5.4-6：CRL の拡張 1

			Description	
CRL and CRL entry extensions	CRL number	CRL entry extension	Authority ディレクトリ属性や CRL 配布点から CRL 発行者が発行する CRL のシリアル番号。	N
	Reason code	CRL entry extension	CRL の失効理由。	N
	Hold instruction code	CRL entry extension	証明書を保持する際に、行うべきアクションが記載された指示情報の OID を表す。	N
	Invalidity date	CRL entry extension	公開鍵証明書の公開鍵に対応する秘密鍵が露呈した、あるいは証明書が不正であることが判明した期日。	N
	CRL scope		CRL を利用する際の指定されたパラメタ範囲を規定する。	N
	Status referral		証明書の利用者に通知する失効情報。	N
	CRL stream identifier		CRL 発行局ごとに与えられるユニークな ID、CRL シリアル番号と結合して利用する。	N
	Order list		CRL のリストを失効期日およびシリアル番号のどちらの昇順で並べるかを指定。	N

	Delta information		関連するΔCRL の存在する位置と、次回ΔCRL が発行される時間を規定。	N

表 5.4-7: CRL の拡張 2

			Description	
CRL distribution points and delta-CRL	CRL Distribution Point	Certificate extension	証明書の失効情報 CRL を確認するポイントを指定。	Y
	Issuing distribution point	CRL extension	CRL 自身に配布点を記載するための領域。	N
	Certificate issuer	CRL entry extension	エントリに対応する証明書の発行者を識別する情報。	N
	Delta CRL indicator	CRL entry extension	ΔCRL の更新対象である CRL を識別する情報。	N
	Base update	ΔCRL	ΔCRL が失効状態を更新する日付時刻。	N
	Freshest CRL	Certificate extension	証明書利用者が参照すべき最新の CRL を識別する情報を規定。	N

（エ）コンパクション

ユビキタス端末･デバイスの利用ケースによって証明書サイズは異なる。そのため、証明書サイズは variable が理想である。しかし、コンパクションのみを追求すると、安全性が損なわれる。そこで、証明書のどの要素を使用／未使用にするかの検討を行い、実際の証明書を用いて証明書サイズを計算する。また、その際 X.509 ASN.1 Basic Encoding Rule でコーディングされた証明書（以下、X.509 証明書と表示）のどの要素がどの部分に対応しているかを明確にするために、バイナリエディタ表示された ASN.1 BER による証明書とその構文解析図を示す。

（１）証明書の要素の使用／未使用

①証明書データ構造

表 5.4-8 は、証明書データ構造のうち、証明書として必須である要素の使用（Y）／未使用（N）とユビキタス端末・デバイスに依存して必要となる要素の使用／未使用を示している。以下特に断りがない限り、必須性の common はユビキタス端末・デバイスすべてに共通する場合について、必要性の high、middle、low、non はそれぞれ high-CPU、

middle-CPU、low-CPU、non-CPU の各場合についての要素の使用／未使用を表しているとする。

表 5.4-8: 証明書データ構造（extensions の詳細は別）

	ITU-T X.509 4th	必須性 common	必要性 high	必要性 middle	必要性 low	必要性 non
Certificate	—	—	—	—	—	—
version	必須	N	Y	Y	N	N
serialNumber	必須	Y	Y	Y	Y	Y
signature	必須	N	N	N	N	N
issuer	必須	Y	Y	Y	Y	Y
validity	必須	N	Y	Y	N	N
subject	必須	N	Y	Y	Y	N
subjectPublicKeyInfo	必須	N	Y	Y	Y	N
issuerUniqueIdentifier	オプション	N	N	N	N	N
subjectUniqueIdentifier	オプション	N	N	N	N	N
extensions	オプション	N	Y	Y	N	N

証明書として必須である要素は、認証局ごとに一意であるシリアル番号（serialNumber）と記載していない要素を参照するために使用するその認証局の発行者名（issuer）である。証明書としての必須要素は、記載されていない要素を他から参照するために使用する要素のみである。つまり、どこの認証局に対するどの証明書について参照するのかが認識できれば良いので、発行者名とシリアル番号だけが必須要素である。ユーザ公開鍵情報でさえ参照することが可能であるため、記載しなくても良い。また、この場合署名も不要である。

また、high-CPU と middle-CPU の端末・デバイスに対し、証明書として必要である要素は、メモリ容量が十分にあることと認証局への参照に通信を必要とすることから、X.509 証明書において必須であった要素をほとんど使用した方が、処理の煩雑さを軽減できる。唯一必要がないのは、署名部分にも記載されているために無意味である署名アルゴリズム識別子（signature）だけである。一方、low-CPU の端末・デバイスについては、メモリ容量が high-CPU や middle-CPU の端末・デバイスと比べてそれほど多くないため、X.509 証明書において必須

であった要素をすべて使用することは望ましくない。そこで、証明書の中でも参照することが比較的多いユーザ名（subject）とユーザ公開鍵情報（subjectPublicKeyInfo）を証明書の要素として使用し、それ以外の要素は使用しない。有効期限（validity）については、本来なら純粋な期限切れなどを CRL でチェックすべきであり、参照されることは少ないため、証明書に記載する要素としては必要でないと判断した。最後に、non-CPU についてはメモリ容量が小さいことを考慮すると、証明書として必要である要素は上記の必須要素のみが良い。

②公開鍵証明書の拡張

表 5.4-9～11 は公開鍵証明書の拡張のうち、証明書として必須である要素の使用／未使用とユビキタス端末・デバイスに依存して必要となる要素の使用／未使用を表している。但し、①から必須である証明書の要素はないとする。

表 5.4-9: 公開鍵証明書の拡張 1

		必須性 common	必要性 high	必要性 middle	必要性 low	必要性 non
Key and policy	Authority key identifier	N	Y	Y	N	N
	Subject key identifier	N	N	N	N	N
	Key usage	N	Y	Y	N	N
	Extended key usage	N	N	N	N	N
	Private key usage period	N	N	N	N	N
	Certificate Policies	N	Y	Y	N	N
	Policy mappings	N	Y	Y	N	N

表 5.4-10: 公開鍵証明書の拡張 2

		必須性 common	必要性 high	必要性 middle	必要性 low	必要性 non

Subject and issuer	Subject alternative name	N	N	N	N	N
	Issuer alternative name	N	N	N	N	N
	Subject directory attributes	N	N	N	N	N

表 5.4-11: 公開鍵証明書の拡張 3

		必須性 common	必要性 high	必要性 middle	必要性 low	必要性 non
Certification path constraint extensions	Basic constraints	N	Y	Y	N	N
	Name constraints	N	N	N	N	N
	Policy constraints	N	N	N	N	N
	Inhibit any policy	N	N	N	N	N

証明書の拡張要素については、上記のメモリ容量の関係から、high-CPU と middle-CPU の端末・デバイスのみに使用し、low-CPU や non-CPU の端末・デバイスには使用しない。

まず、鍵とポリシー情報の拡張に関しては、使用鍵識別子（Authority key identifier)、鍵の使用法（Key usage)、証明書方針（Certificate Policies)、方針の対応（Policy mappings）の 4 種類を使用し、それ以外の要素は使用しない。使用鍵識別子については、鍵の更新が行われた後、同一の認証局に発行された複数の鍵の識別をするために必要である。鍵の使用法は、署名鍵や暗号化鍵などの公開鍵に利用目的を定め、それ以外の利用を制限するために必要である。証明書方針については、証明書は複数の方針を持つ環境で使用される可能性があることから、その証明書にはどの方針が有効であるかを示すために必要である。方針の対応については、認証局によって発行された証明書のみに利用され、発行者の証明書ポリシーの持つ複数の方針が、証明書の発行対象認証局のどの方針と等価であるかを指定するために必要である。

次に、ユーザと発行者情報の拡張に関しては、どの要素も使用しな

い。

最後に、認証パス制限に関しては、基本的制限(Basic constraints)のみ使用し、それ以外の要素は使用しない。基本的制限については、対象者が認証局として振る舞って良いかを示し、良ければ適用するパス長の制約（最大パス長）を指定するために必要である。

証明書の拡張要素については、証明書基本プロファイル案と同じ要素を使用することにした。なぜなら、ユビキタス環境において特別に必要となる要素がなかったからである。

（２）証明書サイズ

①ユビキタス端末・デバイスに必要な証明書サイズ

表 5.4-12 は、(1)において検討した証明書として必須である要素のみを使用した場合と各ユビキタス端末･デバイスに依存して必要となる要素を使用した場合の X.509 証明書サイズを示している。サンプル数は、RSA 1024 Bits、RSA 2048 Bits と鍵長の異なる 2 種の証明書を各 5 通ずつの計 10 通としている。

証明書として必須である要素のみを使用した場合は、発行者名とシリアル番号のみで署名もつけていないため、この証明書サイズは X.509 証明書のもとのサイズに対して 2 割程度になっている。non-CPU の端末･デバイスに対して必要である要素も必須要素と同じであるから、証明書サイズも当然等しくなっている。ここで、non-CPU の端末・デバイスの場合、参照する証明書の要素が多いことから、証明書サイズは小さくなっているが、認証局の処理の煩雑さを考えると、全体的な処理量はそれほど変わらない。

それに対して、high-CPU と middle-CPU の端末・デバイスに対し、証明書として必要である要素は、署名アルゴリズム識別子を除いただけであるから、ほとんど差がない。

表 5.4-12: 各ユビキタス端末・デバイスに必要な証明書サイズ

鍵長	ファイル名	X.509 証明書 (byte)	必須性		必要性						備考
			バイト数 (byte)	倍率 (倍)	high･middle (byte)	倍率 (倍)	low (byte)	倍率 (倍)	non (byte)	倍率 (倍)	
RSA 1024 Bits	CAC	622	127	0.20	607	0.98	552	0.89	127	0.20	3+拡張
	Class1	577	124	0.21	562	0.97	530	0.92	124	0.21	3
	FESTE	669	162	0.24	654	0.98	622	0.93	162	0.24	6
	GTE	606	131	0.22	591	0.98	559	0.92	131	0.22	4
	VeriSign	774	222	0.29	759	0.98	727	0.94	222	0.29	5
	各平均			0.23		0.98		0.92		0.23	
RSA 2048 Bits	ABA	953	167	0.18	938	0.98	877	0.92	167	0.18	6+拡張
	C&W	749	64	0.09	734	0.98	689	0.92	64	0.09	2
	DST	1030	220	0.21	1015	0.99	983	0.95	220	0.21	7
	SecureNet	733	57	0.08	718	0.98	673	0.92	57	0.08	2
	SERVICIOS	1023	197	0.19	1008	0.99	950	0.93	197	0.19	5+拡張
	各平均			0.15		0.98		0.93		0.15	
	全体平均			0.19		0.98		0.92		0.19	

注）各倍率は、X.509 証明書のもとのサイズに対する倍率を表している。
備考欄の数字は発行者やユーザにおける項目数、拡張は extensions があることを表す

また、low-CPU の端末・デバイスに対し、証明書として必要である要素は、high-CPU と middle-CPU の端末・デバイスの場合から、さらにヴァージョン(version)と有効期間(validity)と拡張(extensions)を除いたが、それぞれはサイズとしてそれほど大きくないため、あまり減っていない。

②ASN.1 BER による証明書（バイナリエディタ表示）と構文解析

①の X.509 証明書に関し、どの要素がどの部分に対応しているかを明確にするために、バイナリエディタ表示された ASN.1 BER による証明書を示す。図 5.4-2 は RSA 1024 Bits のファイル CAC の場合である。

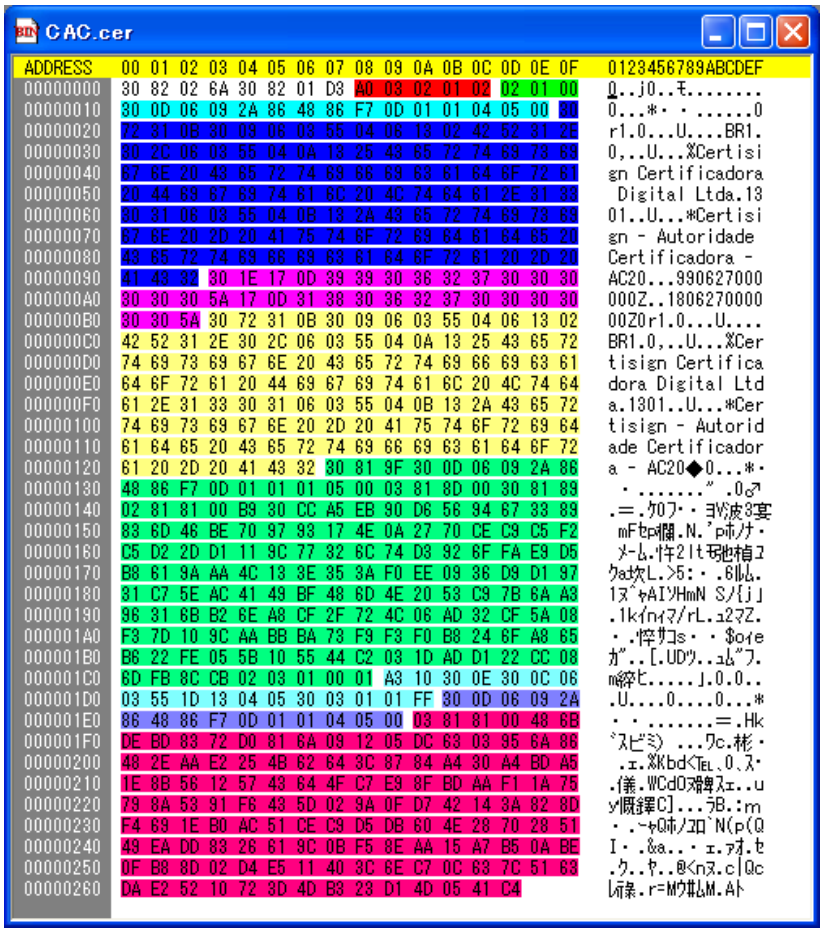


図 5..4-2: ASN.1 BER による証明書（CAC）

②各ユビキタス端末・デバイスによる可能な証明書サイズ

一般的に想定される各ユビキタス端末・デバイスの CPU（クロック数）とメモリ容量から、（1)でも検討したように、high-CPU と middle-CPU の端末・デバイスは、X.509 証明書をそのまま載せても十分であるが、low-CPU の端末・デバイスになると、そのままの X.509 証明書では不足する可能性が出てきて、non-CPU の端末・デバイスに至っては、X.509 証明書の要素を削らざるを得ないことがわかる。

（オ） 評価

現在、一般的に使われている X.509 証明書に対して、他方式の XML 言語で表示された証明書（以下、XML 証明書と表記)、S-expressions 形式で表示された SPKI の証明書（同 SPKI 証明書）との証明書サイズの比較を行う。さらに、コンパクションを考慮した場合について、X.509 証明書と XML 証明書､SPKI 証明書との証明書サイズの比較を行う。また、認証プロトコルをモデル化し、そのコストによる評価を行う。一方、ユビキタス環境において、CPU による性能だけではなく、通信ネックとなる部分の通信速度も検討すべき事項である。そこで、非接触型 IC カードの国際規格 ISO-14443 の制約条件を前提として、通信帯域を考慮した場合の証明書サイズ別の処理時間について計算・比較を行う。そして、モデル化した認証プロトコルに関し、通信帯域を考慮した場合の共通鍵処理と公開鍵処理の処理時間について計算・比較を行う。

X.509 証明書と XML 証明書、SPKI 証明書との証明書サイズの比較

まず、比較調査の対象は、X.509 証明書と XML 証明書、SPKI 証明書である。さらに、XML 証明書については、タグの要素名のつけ方によって証明書サイズが異なるため､次の三つの場合における証明書サイズの調査も行った。

- タグの要素名を X.509 証明書の定義通りにつけた（上限の）場合（以下、ALL と表示）
- タグの要素名を理解できる程度に短くした場合（同 SHORT）
- タグの要素名を 1 文字にした（下限の）場合（同 ONE）

サンプル数は、RSA 1024 Bits、RSA 2048 Bits と鍵長の異なる 2 種の証明書を各 5 通ずつの計 10 通とし、X.509 証明書サイズに対する各証明書サイズの倍率によって、比較を行った。

次の図 5.4-3 は X.509 証明書の例（以下、例に用いたのはすべて CAC のファイル）である。

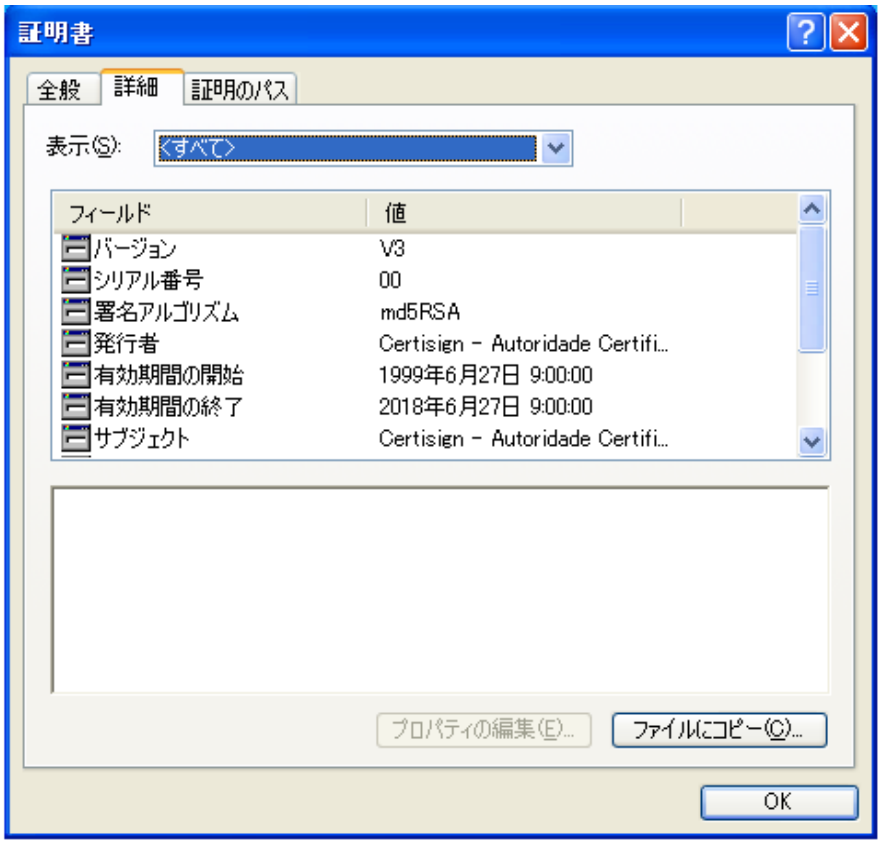


図 5.4-3: X.509 証明書の例

次の図 5.4-4 は ALL の場合についての XML 証明書の例である。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Certificate (View Source for full doctype...)>
- <Certificate>
    <version>V3</version>
    <serialNumber>eC</serialNumber>
  - <signature>
      <algorithm>md5RSA</algorithm>
    </signature>
    <issuer>OU=Certisign - Autoridade Certificadora -AC2 O=Certisign Certificadora Digital Ltda. C=BR</issuer>
  - <validity>
      <notBefore>1999/06/27 9:00:00</notBefore>
      <notAfter>2018/06/27 9:00:00</notAfter>
    </validity>
    <subject>OU=Certisign - Autoridade Certificadora -AC2 O=Certisign Certificadora Digital Ltda. C=BR</subject>
  - <subjectPublicKeyInfo>
      <algorithm>RSA (1024Bits)</algorithm>
      <subjectPublicKey>Y1aWArK6PKnpFErHhKMJa4Hicr34WSKTANHCG85JXFGmueMGOPj53FYyS
        iTNJpNuiz/yuVZVHPD0lyi17cPRrrAnzMQ8SJRekxQczeOZRznnELD2D7lWIQo3dDw8PSXKLU
        TlVDzJ1DdiIgapd3410Kro3uW1UXhd6VFmixyb/Jyme3D0Az0x0c3prKe</subjectPublicKey>
    </subjectPublicKeyInfo>
  - <extensions>
      <basicConstraints>Subject Type=CA, Path Length Constraint=None</basicConstraints>
    - <hash>
        <algorithmIdentifier>sha1</algorithmIdentifier>
        <hashValue>rrAnzMQANHCG85JXFGmueMGOPj53FYySWSKTQo3diTNJpNuiz/yuVZVHPD0lyi1
          7cPXhd6VFmi2D7lWIW1U3prKSJRekxQhKMJa4Hicr34Dw8PSXKLUTlVDzJ1DAz0x0cczediIg
          apd3410Kxyb/Jyme3D08eOro3uY1aWArK6PKnpFErHR</hashValue>
      </hash>
    </extensions>
</Certificate>
```

図 5.4-4: XML 証明書の例 (ALL の場合)

次の図 5.4-5 は SPKI 証明書の例である。

```
(cert
(version V3)
(display |eC|)
(issuer
(hash sha1 |rrAnzMQANHCG85JXFGmueMGOPj53FYySWSKTQo3diTNJpNuiz/yuVZVHPD0Iyi17cPXhd6VFmi2D7IWIW1U3p
rKSJRekxQhKMJa4Hicr34Dw8PSXKLUTIVDzJ1DAz0x0cczediIgapd3410Kxyb/Jyme3D08eOro3uY1aWArK6PKnpFErHR|))
(issuer-info OU=Certisign - Autoridade Certificadora -AC2 O=Certisign Certificadora Digital Ltda. C=BR)
(valid
(not-before 1999-06-27_09:00:00)
(not-after 2018-06-27_09:00:00))
(subject
(public-key
(rsa-pkcs1-md5 |Y1aWArK6PKnpFErHhKMJa4Hicr34WSKTANHCG85JXFGmueMGOPj53FYySiTNJpNuiz/yuVZVHPD0Iyi17cPRrrA
nzMQ8SJRekxQczeOZRznnELD2D7IWIQo3dDw8PSXKLUTIVDzJ1DdiIgapd3410Kro3uW1UXhd6VFmixyb/Jyme3D0Az0x0c3prKe|)))
(subject-info OU=Certisign - Autoridade Certificadora -AC2 O=Certisign Certificadora Digital Ltda. C=BR)
(extensions
(basic-constraint Subject Type=CA, Path Length Constraint=None))
(tag (*)))
```

図 5.4-5: SPKI 証明書の例

これらの例のようにして、RSA 1024 Bits 5 通、RSA 2048 Bits 5 通の計 10 通の X.509 証明書に対して、各要素とそのデータを X.509 証明書の場合と同等にした XML 証明書（ALL、SHORT、ONE の各場合）と SPKI 証明書を作成し、各証明書サイズの調査を行った。

表 5.4-13 は、X.509 証明書と XML 証明書（ALL、SHORT、ONE の各場合）、SPKI 証明書の各サイズと X.509 証明書サイズに対する各証明書サイズの倍率を示している。

表 5.4-13: 各証明書サイズと X.509 証明書に対する各証明書の倍率

鍵長	ファイル名	X.509 証明書 (byte)	XML証明書						SPKI証明書		備考
			ALL (byte)	倍率 (倍)	SHORT (byte)	倍率 (倍)	ONE (byte)	倍率 (倍)	バイト数 (byte)	倍率 (倍)	
RSA 1024 Bits	CAC	622	1239	1.99	1010	1.62	873	1.40	862	1.39	3+拡張
	Class1	577	1141	1.98	924	1.60	805	1.40	768	1.33	3
	FESTE	669	1166	1.74	951	1.42	830	1.24	793	1.19	6
	GTE	606	1155	1.91	938	1.55	819	1.35	782	1.29	4
	VeriSign	774	1314	1.70	1098	1.42	978	1.26	942	1.22	5
	各平均			1.86		1.52		1.33		1.28	
RSA 2048 Bits	ABA	953	1608	1.69	1381	1.45	1244	1.31	1233	1.29	6+拡張
	C&W	749	1410	1.88	1195	1.60	1076	1.44	1037	1.38	2
	DST	1030	1612	1.57	1396	1.36	1277	1.24	1241	1.20	7
	SecureNet	733	1386	1.89	1171	1.60	1052	1.44	1013	1.38	2
	SERVICIOS	1023	1735	1.70	1508	1.47	1371	1.34	1360	1.33	5+拡張
	各平均			1.75		1.50		1.35		1.32	
	全体平均			1.81		1.51		1.34		1.30	

注）備考欄の数字は発行者やユーザにおける項目数、拡張は extensions があることを表している。

＜考察＞

- X.509 証明書に比べて、平均的に XML 証明書サイズ、SPKI 証明書サイズはともに大きくなっている。XML 証明書はデータを挟んでいる前後のタグが大きく、SPKI 証明書はデータを囲んでいる括弧と要素名が大きさを持っているためである。
- XML の要素名が長い場合は鍵長の長い RSA 2048 Bits の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっているのに対し、XML の要素名が短い場合や SPKI 証明書の場合は鍵長の短い RSA 1024 Bits の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。要素名が短くなるにつれて、証明書サイズに対する鍵や署名の占める割合が大きくなるためである。
- XML 証明書より SPKI 証明書の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。XML 証明書がデータの前後をタグで挟んでいるのに対し、SPKI 証明書の方はデータの前のみに要素名をつけ、データの後ろは括弧で閉じているだけだからである。
- XML 証明書を使用する場合、タグの要素名は SHORT 程度の長さにするのが最適である。タグの要素名を ALL まで長くすると、証明書サイズが大きくなりすぎ、ONE まで短くすると、要素の表示として不明確になるためである。
- RSA 1024 Bits、RSA 2048 Bits はともに、同じ長さの鍵長同士でも X.509 証明書サイズに対する証明書サイズの倍率が非一様である。証明書要素のデータの大きさが一定である部分（ヴァージョン情報や有効期間、公開鍵など）以外の、データの大きさが不定である部分（発行者やユーザ、拡張など）に差があると考えられる。発行者やユーザの項目数が多い場合と証明書の要素に拡張がない場合、X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。
  - 発行者やユーザの項目数について
  - X.509 証明書が発行者やユーザを各項目ごとにコーディングしているのに対し、XML 証明書や SPKI 証明書は発行者やユーザを一項目として羅列しているためである。
  - 証明書の要素の拡張について

- 証明書の要素に拡張がある場合は、要素数が増えるためである。特に、ONE の場合の XML 証明書や SPKI 証明書のようにタグの部分が小さいとき、証明書サイズに対する拡張部分の占める割合が大きくなるため、影響が大きくなっている。

＜結論＞

X.509 証明書の代わりに用いるならば、現状では SPKI 証明書の方が良い。鍵長が長くなった場合や発行者やユーザが複雑な構成であった場合、証明書の要素を今より減らした場合に、特に威力を発揮する。また、XML 証明書についても、パースが楽である、タグごとに暗号化できるなどのメリットから、まだ考慮する余地は残っている。

（２）コンパクションを考慮した場合の証明書サイズの比較

まず、比較調査の対象は、(1)と同様に X.509 証明書と XML 証明書、SPKI 証明書である。そして今回は、XML 証明書について、タグの要素名を最適な長さである SHORT の場合のみとする。

サンプル数は、これまで通り、RSA 1024 Bits、RSA 2048 Bits と鍵長の異なる 2 種の証明書を各 5 通ずつの計 10 通とし、X.509 証明書サイズに対する各証明書サイズの倍率によって、比較を行った。

表 5.4-14～16 は、X.509 証明書と XML 証明書、SPKI 証明書の各サイズと X.509 証明書サイズに対する各証明書サイズの倍率を示している。表 5.4-14 は証明書として必須である（non-CPU の端末・デバイスに対する証明書として必要である）要素を使用した場合であり、表 5.4-15 は high-CPU と middle-CPU の端末・デバイスに対する証明書として必要である要素を使用した場合であり、表 5.4-16 は low-CPU の端末・デバイスに対する証明書として必要である要素を使用した場合である。

表 5.4-14: 各証明書サイズと X.509 証明書に対する各証明書の倍率（証明書の必須要素）

鍵長	ファイル名	X.509 証明書 (byte)	XML証明書 バイト数 (byte)	XML証明書 倍率 (倍)	SPKI証明書 バイト数 (byte)	SPKI証明書 倍率 (倍)	備考
RSA 1024 Bits	CAC	127	220	1.73	138	1.09	3+拡張
	Class1	124	223	1.80	141	1.14	3
	FESTE	162	225	1.39	143	0.88	6
	GTE	131	220	1.68	138	1.05	4
	VeriSign	222	308	1.39	226	1.02	5
	各平均			1.60		1.04	
RSA 2048 Bits	ABA	167	240	1.44	158	0.95	6+拡張
	C&W	64	170	2.66	88	1.38	2
	DST	220	282	1.28	200	0.91	7
	SecureNet	57	158	2.77	76	1.33	2
	SERVICIOS	197	293	1.49	211	1.07	5+拡張
	各平均			1.93		1.13	
	全体平均			1.76		1.08	

注）各倍率は、X.509 証明書のもとのサイズに対する倍率を表している。備考欄の数字は発行者やユーザにおける項目数、拡張は extensions があることを表している。

表 5.4-15: 各証明書サイズと X.509 証明書に対する各証明書の倍率（high-CPU・middle-CPU）

鍵長	ファイル名	X.509 証明書 (byte)	XML証明書 バイト数 (byte)	XML証明書 倍率（倍）	SPKI証明書 バイト数 (byte)	SPKI証明書 倍率（倍）	備考
RSA 1024 Bits	CAC	607	978	1.61	862	1.42	3+拡張
	Class1	562	892	1.59	768	1.37	3
	FESTE	654	919	1.41	793	1.21	6
	GTE	591	906	1.53	782	1.32	4
	VeriSign	759	1065	1.40	942	1.24	5
	各平均			1.51		1.31	
RSA 2048 Bits	ABA	938	1348	1.44	1233	1.31	6+拡張
	C&W	734	1163	1.58	1037	1.41	2
	DST	1015	1363	1.34	1241	1.22	7
	SecureNet	718	1139	1.59	1013	1.41	2
	SERVICIOS	1008	1475	1.46	1360	1.35	5+拡張
	各平均			1.48		1.34	
	全体平均			1.50		1.33	

表 5.4-16: 各証明書サイズと X.509 証明書に対する各証明書の倍率（low-CPU）

鍵長	ファイル名	X.509 証明書 (byte)	XML証明書 バイト数 (byte)	XML証明書 倍率（倍）	SPKI証明書 バイト数 (byte)	SPKI証明書 倍率（倍）	備考
RSA 1024 Bits	CAC	552	815	1.48	693	1.26	3+拡張
	Class1	530	800	1.51	678	1.28	3
	FESTE	622	825	1.33	703	1.13	6
	GTE	559	814	1.46	692	1.24	4
	VeriSign	727	973	1.34	852	1.17	5
	各平均			1.42		1.21	
RSA 2048 Bits	ABA	877	1188	1.35	1067	1.22	6+拡張
	C&W	689	1069	1.55	947	1.37	2
	DST	983	1271	1.29	1151	1.17	7
	SecureNet	673	1045	1.55	923	1.37	2
	SERVICIOS	950	1312	1.38	1191	1.25	5+拡張
	各平均			1.43		1.28	
	全体平均			1.42		1.25	

注）各倍率は、X.509 証明書のもとのサイズに対する倍率を表している。備考欄の数字は発行者やユーザにおける項目数、拡張は extensions があることを表している。

＜考察＞

- どの表も、X.509 証明書に比べて、平均的に XML 証明書サイズ、SPKI 証明書サイズはともに大きくなっている（SPKI 証明書に関しては、一部小さくなっているものもある）。XML 証明書に関しては、厳選して減らす要素を決めた low-CPU の端

末・デバイスの場合の X.509 証明書サイズに対する証明書サイズの倍率が低くなっており、最も要素を減らして必須要素のみ使用した場合の X.509 証明書サイズに対する証明書サイズの倍率が高くなっている。それに対し、SPKI 証明書に関しては、最も要素を減らして必須要素のみ使用した場合の X.509 証明書サイズに対する証明書サイズの倍率が低くなっており、逆に最も要素を残した high-CPU と middle-CPU の場合の端末・デバイスの場合の X.509 証明書サイズに対する証明書サイズの倍率が高くなっている。

- XML 証明書に関しては、最も要素を残した high-CPU と middle-CPU の場合、鍵長の長い RSA 2048 Bits の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっているのに対し、最も要素を減らして必須要素のみ使用した場合と厳選して減らす要素を決めた low-CPU の端末・デバイスの場合は、鍵長の短い RSA 1024 Bits の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。一方、SPKI 証明書に関しては、どの表も鍵長の短い RSA 1024 Bits の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。但し、最も要素を減らして必須要素のみ使用した場合が、X.509 証明書サイズに対する証明書サイズの倍率の低くなる率が大きくなっている。
- どの表も、XML 証明書より SPKI 証明書の方が X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。
- どの表も、RSA 1024 Bits、RSA 2048 Bits はともに、同じ長さの鍵長同士でも X.509 証明書サイズに対する証明書サイズの倍率が非一様である。証明書要素のデータの大きさが一定である部分（ヴァージョン情報や有効期間、公開鍵など）以外の、データの大きさが不定である部分（発行者やユーザ、拡張など）に差があると考えられる。発行者やユーザの項目数が多い場合、X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。証明書の要素に拡張がある場合、拡張を残してある high-CPU と middle-CPU の場合は、X.509 証明書サイズに対する証明書サイズの倍率が高くなっているが、拡張をなくした必須要素のみ使用した場合と low-CPU の端末・デバイスの場合は、X.509 証明書サイズに対する証明書サイズの倍率が低くなっている。

＜結論＞

(1) の場合と同様、X.509 証明書の代わりに用いるならば、現状では SPKI 証明書の方が良い。その中でも、最も要素を減らして必須要素のみ使用した場合で、かつ鍵長が短くなった場合や発行者やユーザが複雑な構成であった場合、特に威力を発揮する。

また、X.509 証明書のコーディングルールである X.509 ASN.1 Basic Encoding Rule を定義し直す場合より、XML 証明書を記述する XML 言語の表示形式を定義した DTD (Document Type Definition) や SPKI 証明書を記述する S-expressions の表示形式を定義したスクリプト言語を定義し直す場合の方が、開発の容易性の観点から言えば、メリットが大きい。

（３）相互認証の認証プロトコルのモデル化とコスト評価

相互認証の認証プロトコルは、各エンティティが所有する情報に従って以下の 4 つに区分される。

30. エンティティが秘密情報を共有している。⇒ローカル認証 (Single Sign On)
31. 両エンティティがそれぞれ相手の公開情報を所有しており、その情報を信頼している。⇒ローカル認証(Pretty Good Privacy: PGP)
32. 両エンティティのうち、一方が相手の公開情報を所有しており、その情報を信頼している。もう一方は、信頼された公開情報（ルート鍵等）のみを所有している。⇒クライアント - サーバ認証
33. 両エンティティが、信頼された公開情報（ルート鍵等）のみを所有している。⇒グローバル認証

①～④の場合の相互認証の認証プロトコルを以下にモデル化し、その通信及びエンティティでの計算コストをシミュレートする。なお、リプレイ攻撃を防止するための乱数要素の交換等は本コスト計算に含めないものとする。また、通信路の改竄等も考慮しない。それから、公開鍵の有効性確認の検証コストも考慮しないこととする。

各記号の意味は以下の通りとする。

SharedSecret： 共有秘密情報。⇒通信コスト：10
Auth_X：エンティティ X の認証情報。⇒通信コスト：10
Public_X：エンティティ X の公開情報。⇒通信コスト：100
Public_M：すべてのエンティティに信頼されている公開情報。⇒通信コスト：100

E(X)：X を暗号化したもの。⇒通信コスト：100
M：Auth_X の作成。⇒計算コスト：10
V：Auth_X の検証。⇒計算コスト：11（公開鍵処理の場合：10001）
C：情報の照合。⇒計算コスト：1
E：Public_X での暗号化。⇒計算コスト：10000
D：Public_X での復号化。⇒計算コスト：10000

①ローカル認証（Single Sign On）

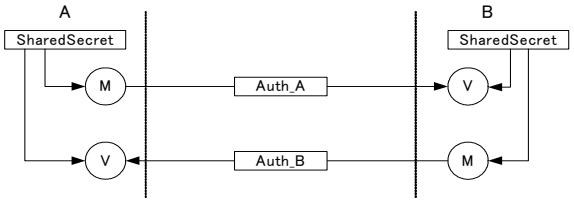


図 5.4-6: ローカル認証（Single Sign On）

②ローカル認証（PGP）

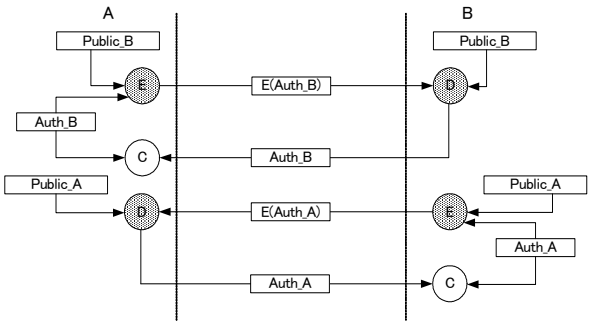


図 5.4-7: ローカル認証（PGP）

③クライアント - サーバ認証

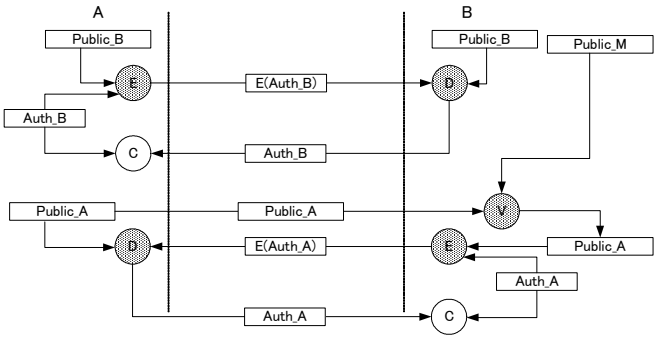


図 5.4-8: クライアント - サーバ認証

④グローバル認証

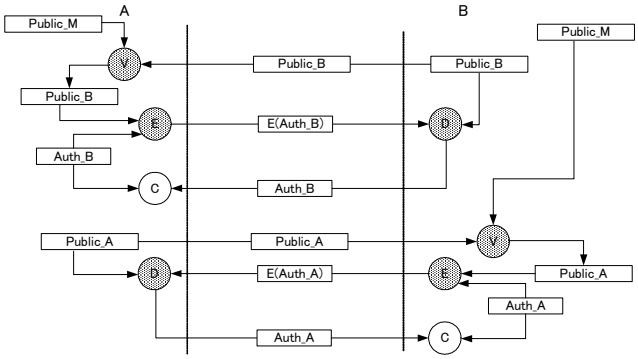


図 5.4-9: グローバル認証

以上より各プロトコルの計算コスト、通信コストを計算すると以下のようになる。

表 5.4-17: 各プロトコルの計算コスト・通信コストの比較

	計算コスト（A）	計算コスト（B）	計算コスト（合計）	通信コスト
①	21	21	42	20
②	20001	20001	40002	220
③	20001	30002	50003	320
④	30002	30002	60004	420

＜考察＞

計算コストを比較してみると、④のグローバル認証は①の Single Sign On を許可するローカル認証に比べて、1500 倍程度のコストを必要としていることがわかる。また、通信コストを比較してみると、④のグローバル認証は①の Single Sign On を許可するローカル認証に比べて、やはり 21 倍のコストを必要としていることがわかる。そのため、認証プロトコルからはグローバル認証をできる限り減らし、その分をローカル認証で代用することができる、すなわち公開鍵の処理を少なくすることがユビキタス環境には望ましい。例えば、図 5.4-10 のような、一度グローバル認証をしたら、有効期間中は Single Sign On を許可するドメイン管理方式などである。

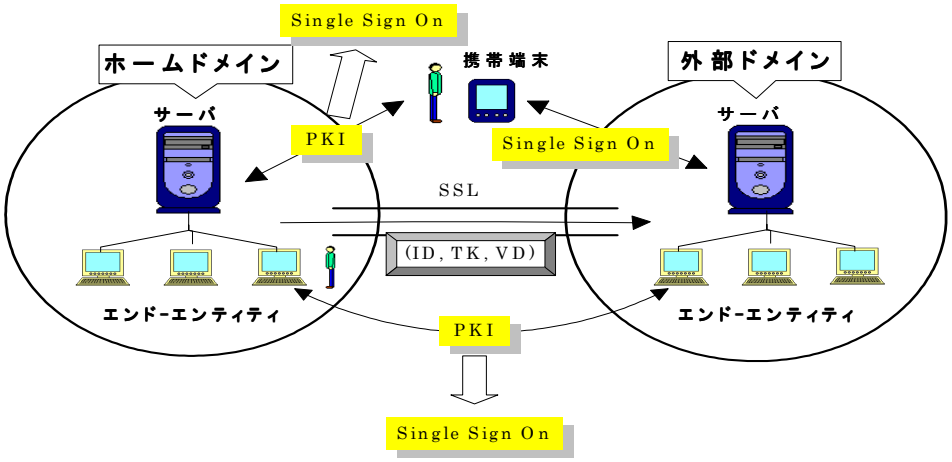


図 5.4-10: 相互認証の認証プロトコルを考慮した場合の望ましいドメイン管理方式

(1)や(2)のような証明書のコンパクションをしても、通信コストや計算コストはほとんど変化がないため、証明書サイズを減らすことは通信コストや計算コストという点では、それほど意味のあることではない。証明書サイズをただ減らすのではなく、公開鍵を半分にすれば、通信コスト・計算コストをともに 4 分の 1 程度にすることができる。しかし、公開鍵を短くすると、安全性も減少する。そこで、図 5.4-11 は数体ふるい法を用いた場合の RSA の鍵長に対する素因数分解の計算量を表したグラフである[16]。

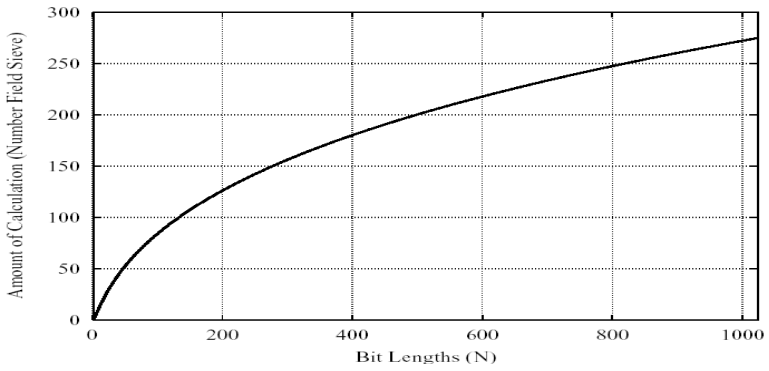


図 5.4-11: 数体ふるい法を用いた場合の RSA の鍵長に対する素因数分解の計算量

6 章のユビキタス環境におけるドメイン管理方式の(7)で記述した有効期間の短い証明書に関して、三年間有効期間のある証明書に対し、一日だけの証明書の RSA の鍵長を 100 Bits 程度にしても、図 5.4-11 のグラフから計算量は 0.3 倍弱しか減少しない。そのことから、One Day Ticket の RSA の鍵長を 100 Bits にまで減らしても、安全性が保

証されると言える。

表 5.4-18 は、RSA 1024 Bits の X.509 証明書の鍵長を 100 Bits にした場合の証明書サイズともとの証明書サイズに対する証明書サイズの倍率を示している。サンプルは、(1)などで用いた RSA 1024Bits の 5 通の X.509 証明書である。この表より、鍵長を 1024 Bits から 100 Bits に減らすだけで、もとの X.509 証明書に対し、証明書サイズは 6～7 割にまで減少することがわかる。

表 5.4-18: 鍵長 100Bits の場合の X.509 証明書サイズ

	CAC	Class1	FESTE	GTE	VeriSign
1024Bits(byte)	622	577	669	606	774
100Bits(byte)	390	345	473	374	542
倍率(倍)	0.63	0.60	0.71	0.62	0.70

（４）通信帯域を考慮した処理時間の比較

①非接触型 IC カードにおける証明書サイズ別の処理時間の比較

ユビキタス環境において、通信ネックとなることが考えられる非接触型 IC カードの国際規格 ISO-14443 の制約条件を前提として、通信帯域を考慮した場合の証明書サイズ別の処理時間について比較を行う。

国際規格 ISO-14443 の制約条件として、フレームサイズとフレームの待ち時間（リーダライタの送信フレーム終了後、カードが応答を始めるまでの最大時間）がある。まず、フレームサイズについては、カードとカードリーダライタとの間のネゴシエーションで決まり、最小で 16 バイト、最大で 256 バイトまで可能である。また、各フレームには、情報フィールドの他にプロローグフィールド（ヘッダ）部とエピローグフィールド（フッタ、誤り検出）部が存在する。プロローグフィールド部は最小 1 バイト、最大 3 バイト、エピローグフィールド部は 2 バイトをそれぞれ必要とし、残りが情報フィールドとなる。これ以上のサイズのデータは、ネゴシエーションで決まるフレームサイズごとに分割して送り、フレームごとに ACK を送ることになる。また、フレームの待ち時間については、最小で 302（us）、最大で 4949（ms）となり、これもフレームサイズ同様にカードとカードリーダライタ間の通信で決定する。ここでは、フレームサイズを最大の 256 バイト、情報フィールド以外のフィールドを 5 バイト、フレームの待ち時間を 302（us）から 4949（ms）までの範囲として、計算を行った。

以上の制約条件の下、図 5.4-12 のように三種類の証明書サイズに

場合分けして、処理時間の比較を行った。まず、サンプル数は、これまで通り、RSA 1024 Bits、RSA 2048 Bits と鍵長の異なる 2 種の証明書を各 5 通ずつの計 10 通とし、それぞれについて、そのままのサイズで用いた X.509 証明書とサイズをコンパクト化した証明書(以下、コンパクト化証明書と表示)とサイズを最もコンパクト化した証明書（同最コンパクト化証明書)、とサイズの異なる三種類の証明書を使用した。コンパクト化証明書、最コンパクト化証明書については、コンパクションで使用した low-CPU の場合の証明書、non-CPU の場合の証明書とそれぞれ同じサイズを用いている。また、コンパクト化証明書、最コンパクト化証明書は、記載していない要素を参照するようにし、その場合はディレクトリサーバから参照することになる。そのため、コンパクト化証明書、最コンパクト化証明書はそれぞれ近似的に Reference 証明書、X.509 証明書相当のサイズを参照することになる。ここで、Reference 証明書は、low-CPU の場合の証明書に対して、参照する要素に署名をつけたときの証明書サイズとした。

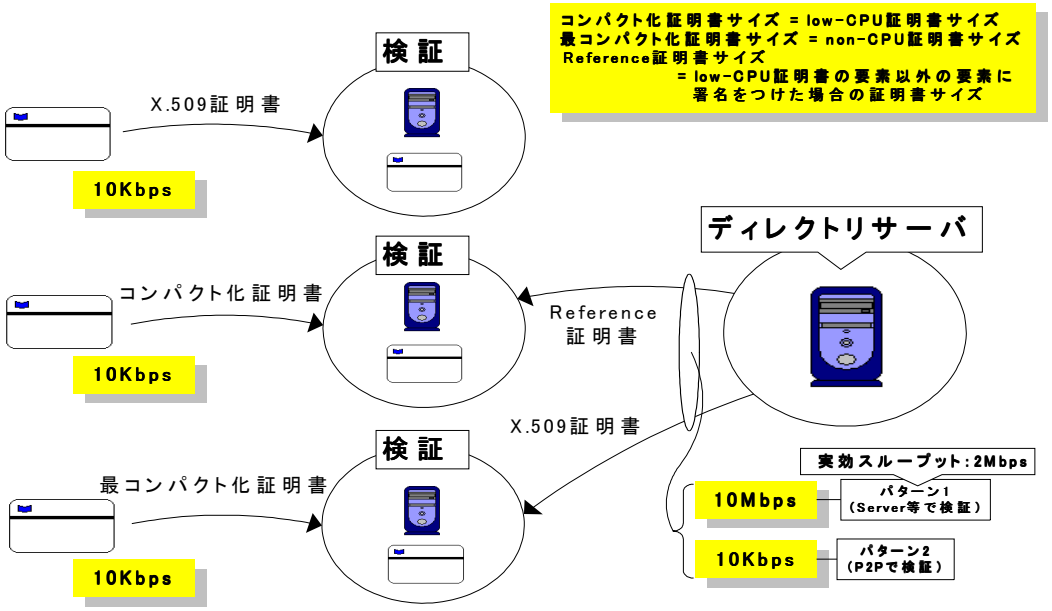


図 5.4-12: 証明書検証モデル

一方、通信速度については、非接触型 IC カードレベルからの通信速度を 10（Kbps）とし、検証する側がサーバ等であるときと P2P で非接触型 IC カードのときとで、ディレクトリサーバからの通信速度を場合分けした。パターン 1 のサーバレベルでの検証の場合は 10（Mbps)、パターン 2 の非接触型 IC カードレベルでの検証の場合は

10（Kbps）として通信時間を計算した。尚、パターン 1 の場合の実効スループットは、2（Mbps）とする。

表 5.4-19 は、検証する場合の各証明書サイズにおけるフレームの送信回数と通信時間を示している。

表 5.4-19: 検証時の各証明書サイズにおけるフレームの送信回数と通信時間

鍵長	ファイル名	サイズ (byte)	パターン1		パターン2	
			送信回数（回）	通信時間（ms）	送信回数（回）	通信時間（ms）
RSA 1024 Bits	CAC	210	1	0.840	1	168.0
	Class1	187	1	0.748	1	149.6
	FESTE	187	1	0.748	1	149.6
	GTE	187	1	0.748	1	149.6
	VeriSign	187	1	0.748	1	149.6
RSA 2048 Bits	ABA	345	1	1.380	2	481.1～5429.8
	C&W	329	1	1.316	2	468.3～5417.0
	DST	316	1	1.264	2	457.9～5406.6
	SecureNet	329	1	1.316	2	468.3～5417.0
	SERVICIOS	342	1	1.368	2	478.7～5427.4

表 5.4-20、21 は各々参照する場合のコンパクト化証明書（Reference 証明書を使用）、最コンパクト化証明書（X.509 証明書を使用）におけるフレームの送信回数と通信時間を示す。

表 5.4-20: 参照時のコンパクト化証明書（Reference 証明書を使用）サイズにおけるフレームの送信回数と通信時間

鍵長	ファイル名	X.509証明書			コンパクト化証明書			最コンパクト化証明書		
		サイズ (byte)	送信回数 (回)	通信時間 (ms)	サイズ (byte)	送信回数 (回)	通信時間 (ms)	サイズ (byte)	送信回数 (回)	通信時間 (ms)
RSA 1024 Bits	CAC	622	3	510.2～10407.6	552	3	454.2～10351.6	127	1	105.6
	Class1	577	3	474.2～10371.6	530	3	436.6～10334.0	124	1	103.2
	FESTE	669	3	547.8～10445.2	622	3	510.2～10407.6	162	1	133.6
	GTE	606	3	497.4～10394.8	559	3	459.8～10357.2	131	1	108.8
	VeriSign	774	4	636.1～15482.2	727	3	594.2～10491.6	222	1	181.6
RSA 2048 Bits	ABA	953	4	779.3～15625.4	877	4	718.5～15564.6	167	1	137.6
	C&W	749	3	611.8～10509.2	689	3	563.8～10461.2	64	1	55.2
	DST	1030	5	845.2～20640.0	983	4	803.3～15649.4	220	1	180.0
	SecureNet	733	3	599.0～10496.4	673	3	551.0～10448.4	57	1	49.6
	SERVICIOS	1023	5	839.6～20634.4	950	4	776.9～15623.0	197	1	161.6

表 5.4-21: 参照時の最コンパクト化証明書（X.509 証明書）サイズにおける

フレームの送信回数と通信時間

鍵長	ファイル名	X.509証明書 通信時間(ms)	コンパクト化証明書 通信時間(ms)		最コンパクト化証明書 通信時間(ms)	
		ー	パターン1	パターン2	パターン1	パターン2
RSA 1024 Bits	CAC	510.2～10407.6	455.0～10352.4	622.2～10519.6	108.1	615.8～10513.2
	Class1	474.2～10371.6	437.4～10334.7	586.2～10483.6	105.5	577.4～10474.8
	FESTE	547.8～10445.2	511.0～10408.3	659.8～10557.2	136.3	681.4～10578.8
	GTE	497.4～10394.8	460.6～10357.9	609.4～10506.8	111.2	606.2～10503.6
	VeriSign	636.1～15482.2	595.0～10492.3	743.8～10641.2	184.7	817.7～15663.8
RSA 2048 Bits	ABA	779.3～15625.4	719.9～15566.0	1199.6～20994.4	141.4	916.9～15763.0
	C&W	611.8～10509.2	565.1～10462.5	1032.1～15878.2	58.2	667.0～10564.4
	DST	845.2～20640.0	804.6～15650.7	1261.2～21056.0	184.1	1025.2～20820.0
	SecureNet	599.0～10496.4	552.3～10449.7	1019.3～15865.4	52.5	648.6～10546.0
	SERVICIOS	839.6～20634.4	778.3～15624.4	1255.6～21050.4	165.7	1001.2～20796.0

表 5.4-22 は、検証と参照を両方行った場合の各証明書サイズにおけるフレームの送信回数と通信時間を示している。

表 5.4-22: 検証、参照全体の各証明書サイズにおけるフレームの送信回数と通信時間

鍵長	ファイル名	サイズ (byte)	パターン1 送信回数 (回)	パターン1 通信時間 (ms)	パターン2 送信回数 (回)	パターン2 通信時間 (ms)
RSA 1024 Bits	CAC	622	1	2.488	3	510.2～10407.6
	Class1	577	1	2.308	3	474.2～10371.6
	FESTE	669	1	2.676	3	547.8～10445.2
	GTE	606	1	2.424	3	497.4～10394.8
	VeriSign	774	1	3.096	4	636.1～15482.2
RSA 2048 Bits	ABA	953	1	3.812	4	779.3～15625.4
	C&W	749	1	2.996	3	611.8～10509.2
	DST	1030	1	4.120	5	845.2～20640.0
	SecureNet	733	1	2.932	3	599.0～10496.4
	SERVICIOS	1023	1	4.092	5	839.6～20634.4

＜考察＞

- 10（Mbps）の通信速度としたパターン 1 のサーバレベルでの検証の場合は、参照に時間がかからないため、検証時に最小の証明書サイズを持つ最コンパクト化証明書の通信時間が最短であり、検証時に最大の証明書サイズを持つ X.509 証明書の通信時間が最長である。
- 10（Kbps）の通信速度としたパターン 2 の非接触型 IC カードレベルでの検証の場合は、参照にも時間がかかってしまうため、参照を行わない X.509 証明書の通信時間が最短であり、両方に署名をつけている分、もとの X.509 証明書より証明書

サイズの和が大きくなるコンパクト化証明書と Reference 証明書のペアの通信時間が最長である。

- 公開鍵のサイズが大きい分、RSA 1024 Bits の場合と RSA 2048 Bits の場合では、一般的に後者の通信時間の方がかかる。しかし、最コンパクト化証明書の場合のみ、公開鍵自身が証明書に記載されていなく、証明書サイズが小さくなることがあるため、それ以外の要素によっては後者の通信時間が短くなることもある。

＜結論＞

参照に時間のかかる P2P の場合は、参照を行わない X.509 証明書が最適であり、参照に時間のかからない検証サーバを用いた場合は、検証時に最小の証明書サイズを持つ最コンパクト化証明書と参照時に最大の証明書サイズを持つ X.509 証明書のペアが最適である。

②共通鍵処理と公開鍵処理の処理時間の比較

(3)の相互認証の認証プロトコルのモデル化で示した、共通鍵処理のローカル認証（Single Sign On）と公開鍵処理のグローバル認証に関して、①の制約条件を考慮した処理時間の比較を行う。

共通鍵処理の hash 関数を SHA-1(Secure Hash Algorithm 1)160 Bits とする。また、マシンの性能を PentiumⅢ 1.2GHz 程度として、hash 処理、公開鍵の暗号化処理、復号化処理、署名の検証処理の各処理時間を表 27 のようにする。但し、ここでの hash 処理は、MD5（Message Digest 5）128 Bits における処理時間を使用しているが、SHA-1 における処理時間とほとんど変わらないため、この処理時間を使用することとする。

表 5.4-23: 各処理時間の仮定表

	共通鍵・公開鍵処理共通	公開鍵処理		
	hash処理	暗号化処理	復号化処理	署名の検証処理
処理時間(ms)	0.00352	3.84	65.52	3.85

図 5.4-13 は共通鍵処理と公開鍵処理の証明書検証モデルを表している。以上の設定から、共通鍵処理と公開鍵処理の計算処理時間を比較し、さらに①の通信時間を用いて、共通鍵処理と公開鍵処理の全体の処理時間を比較する。ここで、公開鍵処理で使用した証明書は、RSA 1024 Bits の CAC ファイルとする。

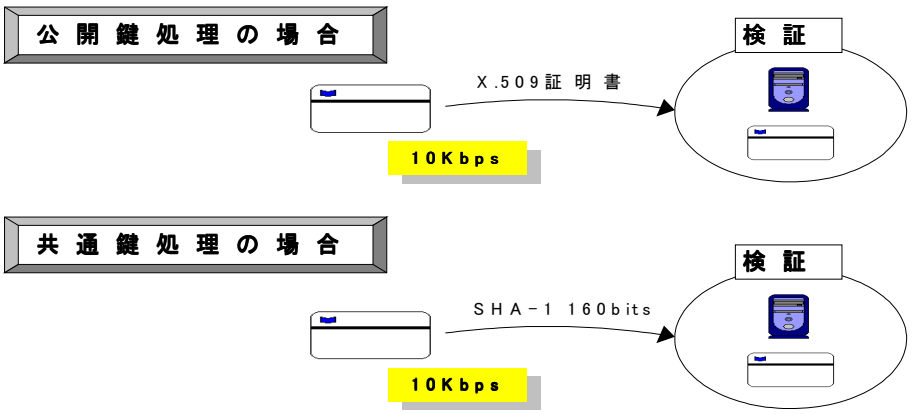


図 5.4-13: 共通鍵処理と公開鍵処理の証明書検証モデル

表 5.4-24 は共通鍵処理と公開鍵処理の計算処理時間を表し、表 5.4-25 は共通鍵処理と公開鍵処理の計算処理と通信処理を合わせた全体処理時間を表している。ここで、(3)相互認証の認証プロトコルのモデル化において、④のグローバル認証における Auth_A と Auth_B のサイズを 1024 Bits、それらを RSA 1024 Bits で暗号化した E(Auth_A)と E(Auth_B)のサイズも 1024 Bits とそれぞれ仮定する。

	hash (回)	時間 (ms)	符号化 (回)	時間 (ms)	復号化 (回)	時間 (ms)	検証 (回)	時間 (ms)	合計時間 (ms)
共通鍵計算	4	0.0141	0	0	0	0	0	0	0.0141
公開鍵計算	0	0	2	7.68	2	131.04	2	7.70	146.4200

表 5.4-24: 共通鍵処理と公開鍵処理の計算処理時間

表 5.4-25: 共通鍵処理と公開鍵処理の全体処理時間

	計算処理	通信処理						処理全体
	時間 (ms)	共通鍵 (回)	時間 (ms)	CAC (回)	時間 (ms)	1024 Bits (回)	時間 (ms)	合計時間 (ms)
共通鍵処理	0.0141	2	32.0	0	0	0	0	32.0141
公開鍵処理	146.4200	0	0	2	1020.4～20815.2	4	3291.6320～82470.8000	4458.4520～103432.4200

注）CAC は、RSA 1024 Bits の CAC ファイルの通信回数を表している。
1024 Bits は、Auth_A、Auth_B、E(Auth_A)、E(Auth_B)の各通信回数を表す。

＜考察＞

- 表 5.4-24 から、公開鍵処理の計算処理時間は共通鍵処理の計

算処理時間の 10000 倍以上かかっていることがわかる。

- 表 5.4.25 から、公開鍵処理の全体処理時間は共通鍵処理の全体処理時間の 140 倍程度から 3200 倍以上かかっていることがわかる。

＜結論＞

共通鍵処理と公開鍵処理の比較から、公開鍵処理を行うグローバル認証をできる限り少なくし、その分を共通鍵処理で済むローカル認証で代用する方式（図 5.4-10 のような Single Sign On を許可するドメイン管理方式）がユビキタス環境には望ましい。

5.4.2 リアルタイム PKI

（ア）リアルタイム PKI に向けた要素研究

（１）リアルタイム PKI とは

近年、インターネットのようなオープンな情報流通基盤の利用が普及するに伴い、様々なユーザ層に対して簡単な操作のみで安心して使えるような安全な情報流通技術が要望されてきている。またこれらのコンピュータやコンピュータネットワークは、日常生活に浸透しつつあり、身のまわりの様々な物にコンピュータが埋め込まれ、コンピュータネットワークを形成している。

このようなユビキタスネットワーク環境では、膨大な数のノードが分散・強調動作して人間生活を支援するが、身の回りのありとあらゆるものにコンピュータが埋め込まれているため、プライバシ情報を含め、それらのコンピュータが扱う情報は高度な安全性を要求される。

現在インターネットなどで広く使われている情報セキュリティのインフラ技術として PKI（Public Key Infrastructure）があるが、インターネットにおける PKI の標準（IETF PKIX）においては、その処理の複雑さなどから、比較的リソースの豊富なコンピュータにおいても必ずしも高速なパフォーマンスを得られていないのが現状である。

したがってインターネットでの PKI の標準技術を、比較的リソースの小さなノードが多く接続されているユビキタスネットワークに、そのまま適用したのでは高速な PKI 処理は期待できない。

上記を踏まえ、今回研究対象として以下の要素技術を挙げて、ユビキタスネットワークに適したリアルタイム PKI の基本アーキテクチャを研究開発した。

- 公開鍵証明書のフォーマット

認証時など、PKI で何らかの処理を行う場合には、まず相手の公開鍵証明書を解析し、検証する必要がある。その際に小型 IC チップなどのようなマシンパワーの低いコンピュータにおいても、そのような処理を高速に行わなければならない場合がある。

本研究では、下記の認証方式や VPN 構築方式、共通鍵交換方式に利用できるコンパクトな証明書のフォーマットを研究開発した。

- 小型 IC チップに適した軽い相互認証プロトコル

IC カードの利用シーンなどにおいては、IC カードの所有者を確実に識別し、認証する必要がある。今回そのための小型 IC チップに適

した処理の軽い認証方式の研究開発を実施した。

● 小型 IC チップを用いた VPN

IC チップが外部のコンピュータと通信する際、通信対象の情報を暗号化し外部への情報漏洩を防止する目的で、IC チップに VPN（Virtual Private Network）機能の実装を行った。この際、VPN で利用する共通鍵暗号のための、IC チップ向け鍵秘密共有アルゴリズムを研究開発した。

（２）公開鍵暗号に基づく軽量な相互認証プロトコルの研究

ユビキタスネットワーク環境では、多くの見知らぬノードと相互に認証を行う必要がある。こうした状況では、共通鍵暗号に比べ、公開鍵暗号に基づく認証方式が有効である。

そこで、本研究では極力特定の公開鍵暗号方式に限定しない、汎用的認証プロトコルを複数考え、処理速度や通信データサイズといった実用的観点から軽量な認証方式について検討した。

(2-1) チャレンジレスポンス型の相互認証

信頼の高い、強固な相手認証を提供する方式として、チャレンジレスポンス（Challenge-Response）型の認証方式が広く普及している。同認証方式では、本人性を証明する証明者が検証者からの質問データ（チャレンジ）に対して、本人たる証を以って応答（証明）を行う。

具体的に、公開鍵暗号に基づくチャレンジレスポンス型認証を簡単に示すと、次のような流れとなる：

34. 検証者は、予測不可能（ランダム）なデータ列（チャレンジ）を証明者へ送信する。
35. 証明者は、受信したデータ列と自己の秘密鍵を用いて、自分のみが計算可能なデータ列（レスポンス）を検証者へ返信する。
36. 検証者は、証明者の公開鍵を用いて、証明者からの返信データの正当性を検証する。正当な場合に限り、本人性を認める。

A. J. Menezes らの暗号専門書(″Handbook of Applied Cryptography″, CRC Press) では、公開鍵暗号に基づくチャレンジレスポンス型認証として、さらに「復号処理」に基づく方法と「デジタル署名」に基づく方法とに分類している。

それぞれの方法を定式化して、以下に示す。今、ノード A とノード

B が相互認証を行うとする。簡単のため、互いに相手公開鍵の正当性は保証の上、それぞれ保持しているとする。

(i)　復号処理に基づく方法

A→B：　$P_B(R_A, ID_A)$

ノード A は、乱数 R_A を生成し、自分の ID_A と共に、ノード B の公開鍵で暗号化する。この暗号化データ $P_B(R_A, ID_A)$ をノード B へ送信する。

A←B：　$P_A(R_A, R_B)$

ノード B は、自己の秘密鍵を用いて、ノード A からの暗号化データ $P_B(R_A, ID_A)$ を復号する。乱数 R_B を生成し、復号した R_B と共に、ノード A の公開鍵で暗号化する。この暗号化データ $P_A(R_A, R_B)$ をノード A へ送信する。

ノード A は、自己の秘密鍵を用いて、受信した暗号化データを復号する。復号した乱数 R_A' が、生成した乱数 R_A と一致するかを検証する。正当な場合に限り、ノード B の本人性を認め、次のステップへ進む。

A→B：　R_B

ノード A は、復号した乱数 R_B をノード B へ送信する。

ノード B は、受信したデータと生成した R_B とが一致するかを検証する。正当な場合に限り、ノード A の本人性を認める。

(ii)　デジタル署名に基づく方法

A→B：　R_A

ノード A は、乱数 R_A を生成し、ノード B へ送信する。

A←B：　R_B, ID_A, $S_B(R_B, R_A, ID_A)$

ノード B は、乱数 R_B を生成する。その乱数 R_B、受信した R_A 及びノード A の ID_A に対し、自己の秘密鍵を用いてデジタル署名 $S_B(R_B, R_A, ID_A)$ を計算する。ノード B は、R_B, ID_A, $S_B(R_B, R_A, ID_A)$ をノード A へ送信する。

ノード A は、ノード B の公開鍵を用いて、受信したデジタル署名 $S_B(R_B, R_A, ID_A)$ の正当性を検証する。
正当な場合に限り、ノード B の本人性を認める。

A→B：　ID_B, $S_A(R_A, R_B, ID_B)$

ノードAは、生成乱数R_A、受信乱数R_B、及びノードBのID_Bに対して、自己の秘密鍵を用いてデジタル署名$S_A(R_A, R_B, ID_B)$を計算する。ノードAは、ID_B, $S_A(R_A, R_B, ID_B)$をノードBへ送信する。

ノードBは、ノードAの公開鍵を用いて、受信したデジタル署名の正当性を検証する。正当な場合に限り、ノードAの本人性を認める。

【考察】

公開鍵暗号では、同じ方式に基づくものでも、暗号化／復号処理と署名／検証処理とでアルゴリズム自体が異なるものがある。それ故に、処理速度や通信データサイズに差異が生じてくる。公開鍵暗号方式によって、その上記特性を考慮し、環境に最適な認証手法を選択する実装も考えられる。

素因数分解の困難性に基づく RSA 暗号では、秘密冪数だけでなく、異なる二つの秘密素因数を保持し、活用することで処理速度が向上する。DSA (Digital Signature Algorithm)では署名データサイズが小さい、という実用上のメリットがある。一般的に、署名／検証では偽造への耐性を高めるため、少なくともハッシュアルゴリズムを組み合わせる必要がある。

こうした暗号方式や処理の特性を踏まえ、暗号処理を担うセキュリティチップの回路規模、計算処理速度、または通信データサイズといった観点から環境に応じて最適な手法を定めることが重要である。

ここで、本研究の想定モデルに対し、上に掲げた二つの相互認証モデルで満たされていない要件がある。

本研究が対象とするモデルでは、相互認証の中で相互が発生したランダムな情報を両ノード間で共有したい。つまり、鍵共有機能も備えた相互認証を行いたい。具体的には、相互認証のやり取りの中で双方が生成する二つの乱数を、認証後の暗号化通信に用いる暗号鍵の糧とする。

この双方で生成した乱数を秘密に共有する機能は、上記二つの認証方法にはない。よって、次節では上記二つの認証方法を踏まえ、本研究にて考案検討した鍵秘密共有機能を携えた、チャレンジレスポンス型の相互認証方式を紹介する。また鍵秘密共有の必要性については同じく次節で触れる。

(2-2) 鍵秘密共有を携えた相互認証プロトコル

本研究では、以下の要件を備えた相互認証プロトコルの設計を行った。

［要件 1］チャレンジレスポンス型相互認証
［要件 2］公開鍵証明書による鍵認証
［要件 3］秘密共有鍵材料の提供
［要件 4］極力特定の暗号方式に限らない汎用的なプロトコル

ここで、要件 3 の必要性を以下に記す。

本研究では、相互認証が正常に完了した後、認証通信中のデータを利用して、暗号化通信にてデータをやり取りするモデルを想定している。理由は、ユビキタスネットワーキング環境では、認証に加え、相互に制御コマンドや電子価値といった各種データの送受信が重要なことである。また、高速性を重視するため、認証後の暗号化通信では公開鍵暗号よりも共通鍵暗号を利用する一般的な仕組みが望ましい。

以上より、相互認証を行う過程で、当該ノード間で暗号化通信用の暗号鍵材料を整えるプロトコルを検討した。

本節では、3 つの考案プロトコルを紹介する。モデルとしては、ノード A とノード B が相互認証を行い、その後の暗号化通信に用いる鍵情報を秘密共有することである。すべてに共通した前提と表記を以下に記述する。

（前提）

- ノード A は、A の秘密鍵と対応する公開鍵証明書 $cert_A$、及び必要な暗号モジュールを所持する。
- ノード B は、B の秘密鍵と対応する公開鍵証明書 $cert_B$、及び必要な暗号モジュールを所持する。

（表記）

$E_X[]$	ノードＸの公開鍵によるを暗号化関数
$D_X[]$	ノードＸの秘密鍵による復号関数
$Sig_X[]$	ノードＸによるデジタル署名関数
Veri[]	デジタル署名の検証関数

5.4.3 リアルタイム PKI を用いたセキュア基盤ネットワークの研究開発

（１）目的

セキュア基盤ネットワークは、ユビキタスネットワークの基盤プロトコルとなり得る価値情報転送プロトコルを実装し、本研究所で研究開発されたセキュア IC チップに対してセキュアに電子的価値情報（本研究では、電子チケットを想定）を発行・流通することができるネットワークである。本節では、チケットを販売する既存（IP）ネットワーク上の価値情報プロバイダと連携したセキュア基盤ネットワークシステムを構築することにより、既存ネットワークとセキュア基盤ネットワークの連携方式及び価値情報の発行・流通に係る技術的検証結果と、研究成果として得られたさらなる課題について報告する。

（２）概要

本研究開発では、電子的価値情報をセキュアに流通させるアプリケーションとして、インターネット上に存在するチケット販売サイト（価値情報プロバイダ）から購入したチケットを、セキュア基盤ネットワークを経由し、指定するセキュア IC チップへ直接または間接的にダウンロードさせる仕組みを実現する。

また、ダウンロードされた電子的価値情報をそのまま権利行使（改札）させるためのゲーティングシステムについても併せて開発する。

セキュア基盤ネットワーク上のサーバノードでは、セキュア IC チップが取り扱う電子的価値情報の汎用フォーマットを検討した上で、それを利用した発行・流通方式を実現する。また、インターネットとセキュア基盤ネットワークの連携のための技術的な検証を行う。

セキュア基盤ネットワークには、リアルタイム PKI を具現化するためのセキュアな価値情報転送プロトコルを実装する。セキュア基盤ネットワークシステムの概要図を以下に示す。

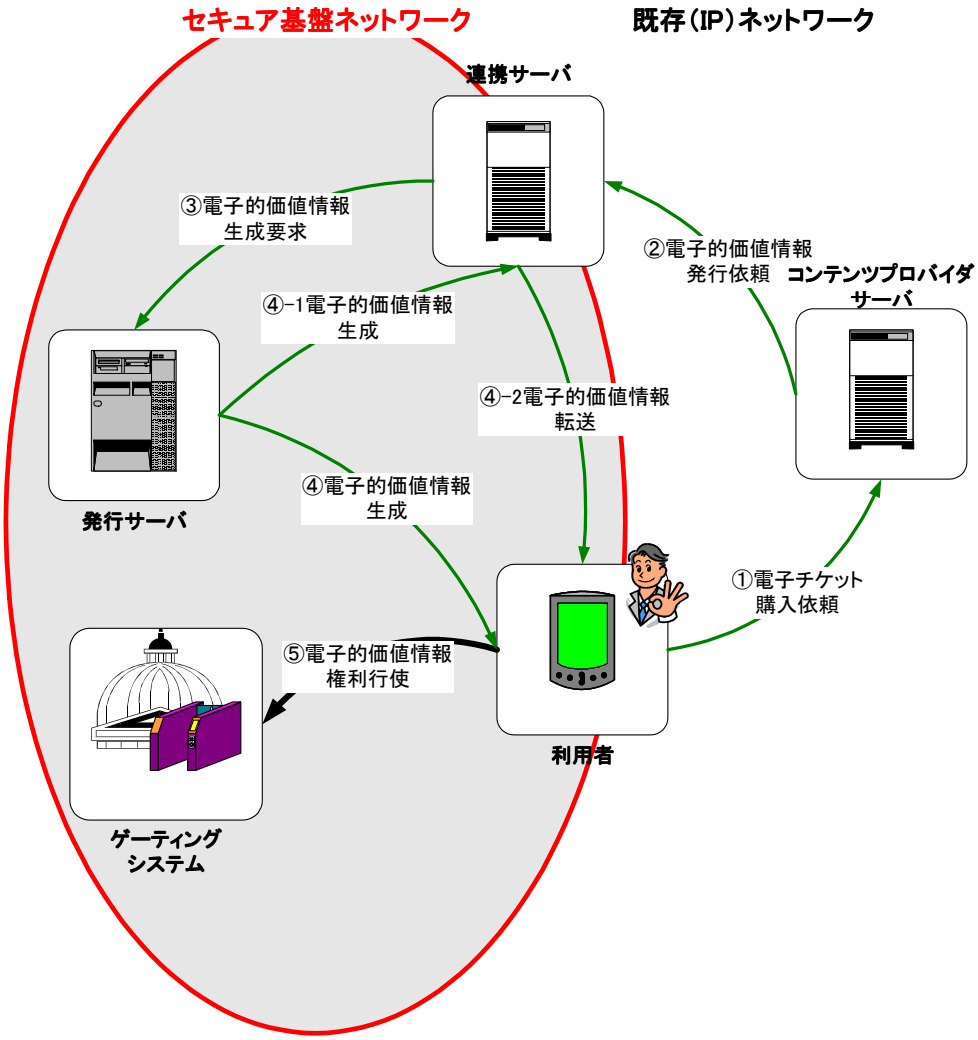


図 5.4-20: セキュア基盤ネットワークシステムの概要図

（３）システム構成

以下、セキュア基盤ネットワークシステムのハードウェア構成、及びソフトウェア構成について説明する。

まず、セキュア基盤ネットワークシステムのハードウェア構成は下図の通りである。

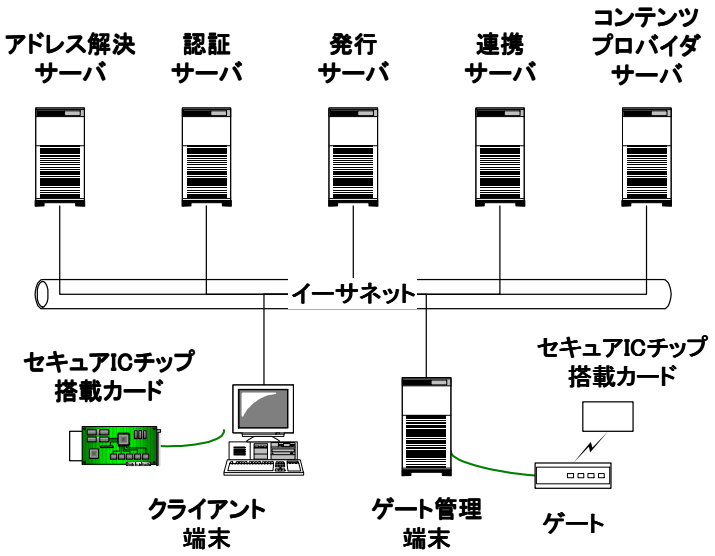


図 5.4-21: セキュア基盤ネットワークシステムのハードウェア構成

- コンテンツプロバイダサーバ
  - 利用者に対して電子チケット販売サービスを提供する
  - 利用者に対して公演検索、販売、ダウンロード状況照会等のサービスを提供し、電子的価値情報発行依頼の生成と連携サーバへの送信の機能を持つ。また、その他の機能として、発行したチケットに付随する詳細データ（約款等を含む）の管理・提供等の機能を有する
  - コンテンツプロバイダサーバは、利用者に対して、チケットの購入、配送を保証する
- 連携サーバ
  - コンテンツプロバイダサーバと連携して電子的価値情報の流通の仕組みを担う
  - コンテンツプロバイダサーバから送信される電子的価値情報発行依頼の受信機能、標準化された電子的価値情報生成要求の生成と発行サーバへの送信機能、発行サーバから送信される電子的価値情報受信機能、クライアント端末への電子的価値情報送信機能、電子的価値情報発行状況照会機能を有する
  - 連携サーバは､コンテンツプロバイダからの委託を受けて､セキュアな配送を保証する
- 発行サーバ
  - セキュア基盤ネットワークで流通する電子的価値情報を発行する

- 発行サーバは、連携サーバ及びクライアントに対して、電子的価値情報の発行を行う
- 発行サーバは、正当な電子的価値情報を生成することを保証する

- アドレス解決サーバ
  - セキュア IC チップに付与される ID によるアドレス解決を実現するため、セキュア IC チップに付与される ID と IP アドレスの組を管理する
- 認証サーバ
  - PKI に基づく公開鍵証明書の作成、及び CRL（公開鍵証明書破棄リスト）の登録、削除、検索サービスを提供する
- クライアント端末
  - コンテンツプロバイダサーバのチケット販売サービスを利用して、電子的価値情報を購入する
  - コンテンツプロバイダサーバに対するチケット購入要求、及び発行サーバまたは連携サーバから送信される電子的価値情報の受信機能を持つ。また、電子的価値情報の内容表示機能、電子的価値情報転送機能を有する
  - クライアント形態は、利用者が所有する PC、PC にシリアル接続されたセキュア IC チップ R/W、及びセキュア IC チップ搭載カードの一式とし、インターネット及びセキュア基盤ネットワーク双方への接続が可能なクライアント端末として開発
- ゲート管理端末
  - ゲートの稼動制御を行う
  - 利用者の入場可否を判定する際に利用されるデータの登録・管理、入場履歴取得等の機能を提供する
  - ゲート管理端末及びゲートは、利用者の保有する正しい電子的価値情報の権利行使を保証する
- ゲート
  - 電子的価値情報の権利行使者の入場可否判定を行う
  - 電子的価値情報の正当性判定機能、セキュア IC チップの有効性判定機能を有する

次にセキュア基盤ネットワークシステムのソフトウェア構成は、下図の通りである。

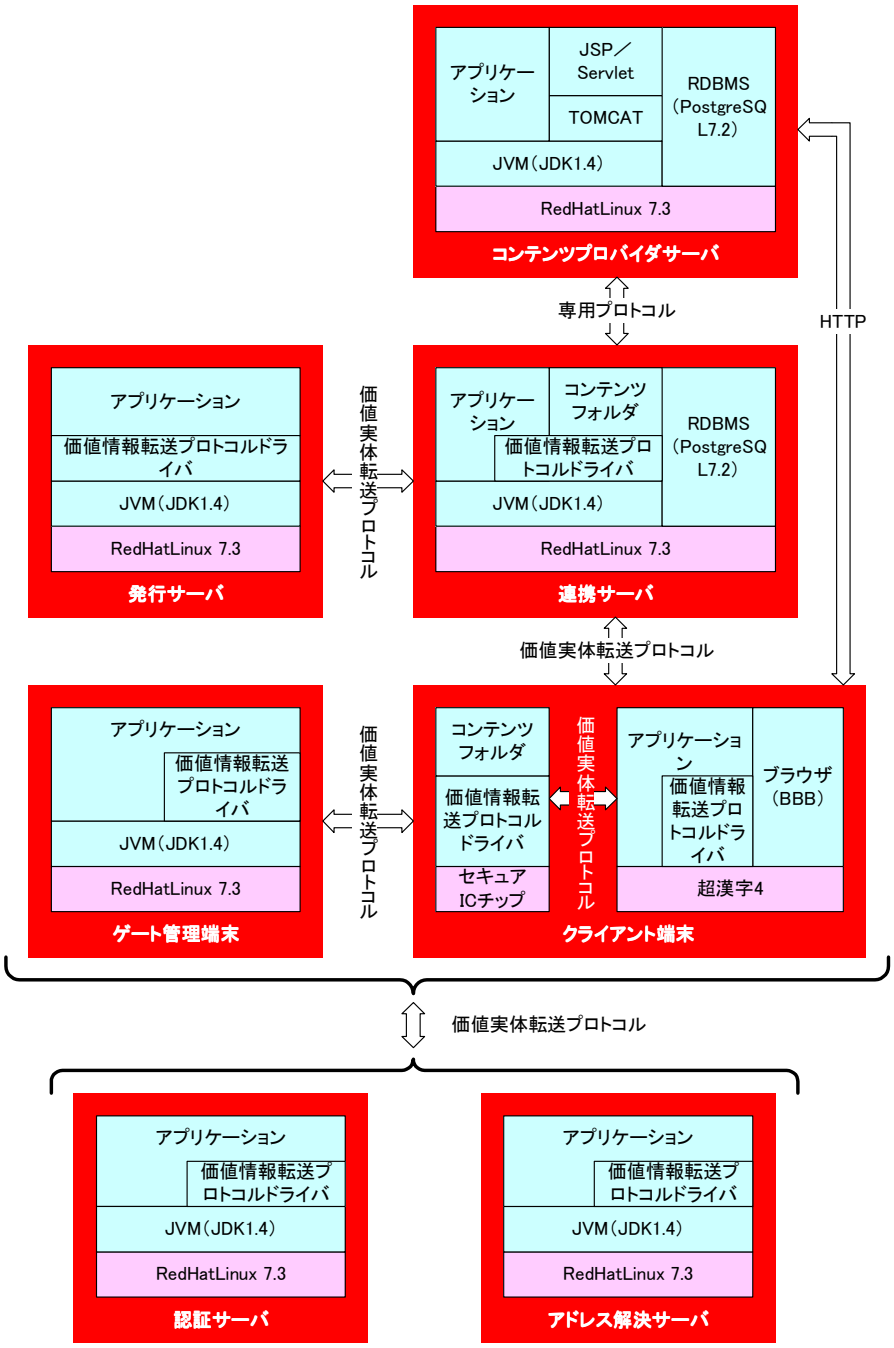


図 5.4-22: セキュア基盤ネットワークシステムのソフトウェア構成

（４）実施結果

リアルタイム PKI を具現化したセキュアな価値情報転送プロトコルを実装したセキュア基盤ネットワークは、前項までに示したシステム機能として実現され、セキュア IC チップに対してセキュアに電子的価値情報を発行・流通させることができるネットワークであることが確認された。

（５）技術的成果

セキュア基盤ネットワークシステムの構築・検証により、以下の技術的成果が確認された。

- 電子的価値情報の標準化
  - 電子的価値情報フォーマットの標準化を目指し、現在一般に流通するイベントチケットを権利行使条件・回数などの観点で分析した上で、以下の 4 種別のチケットコンテンツを表現することが可能な汎用電子的価値情報フォーマットを策定した
    - ✧ 日時指定型／1 回有効チケット
    - ✧ 期間指定型／1 回・n 回・無制限回有効チケット
    - ✧ 複合型／縦列チケット
    - ✧ 複合型／並列チケット
  - 電子的価値情報フォーマットでは、電子的価値情報を格納するセキュア IC チップの容量に制限があることを考慮し、権利行使に必要な情報のみを電子的価値情報フォーマット上に定義し、利用者がチケット（イベント）の詳細を確認するための情報はコンテンツプロバイダサーバに分散管理する方針が適切であることが確認された
  - 上記でフォーマットの定義を行った 4 種別のチケットコンテンツのうち、下記 2 種別について電子的価値情報の実装を行い、セキュア基盤ネットワークにおいてダウンロード／転々流通／権利行使が可能であることを確認した
    - ✧ 日時指定型／１回有効チケット
    - ✧ 期間指定型／１回・n 回・無制限回有効チケット

- 電子的価値情報発行の仕組み
  - コンテンツプロバイダに依存する電子的価値情報発行依頼のフォーマットと、電子的価値情報として発行する汎用電子的価値情報フォーマットを分離して定義・実装することにより、独立性の高い電子的価値情報発行処理を確立した
  - 電子的価値発行情報のダウンロードにおいて連携サーバを経由した転送方式を採用することにより、サーバノードにおける電子的価値情報の保管機能、及びダウンロード失敗による再ダウンロード機能の効率的な負荷分散設計を検証した

- 電子的価値情報の権利行使
  - 多種多様なチケットの権利行使方式に関してその要件を分析した上で、利用者の保持する電子的価値情報の権利行使価値を更新する（価値を消滅させる）方式が適切であることを確認し、それを実現した
  - 電子的価値情報の発行時にその価値情報実体へ権利行使に関する有効期間条件を設定することにより、電子的価値情報の権利行使有効期間の制御を可能とした

- 既存の価値情報プロバイダとの連携容易性確保
  - コンテンツプロバイダサーバでは、既存サービスと新規連携サービスを受けるためのセキュア基盤ネットワーク用のアプリケーションを分離設計することより、セキュア IC チップに付与される ID 等のセキュア基盤ネットワーク固有の情報を管理することなく連携させることで、既存コンテンツプロバイダサーバへの改造範囲を可能な限り限定する方式を確立した
  - 連携サーバでは、コンテンツプロバイダを識別する管理情報を保持するなど、コンテンツプロバイダの追加を考慮した拡張性を持たせる方式を確立した
  - 連携サーバの電子的価値情報発行処理性能として、コンテンツプロバイダサーバから同時接続 10 件、1 件あたり 4 枚までのチケット発行処理が可能であるレベルを確保した

- セキュア基盤ネットワークのセキュリティ
  - 不正アクセス防止のためのネットワーク構成に関する考慮について、研究所内のクローズドネットワークを前提としたため、今回は実装の対象外とした
  - ただし、無制限偽造のリスクを有するデータ及びモジュールに関して、独立性を保持した設計とすることにより、将来の物理的なセキュリティ施策への対応を考慮した

- ゲーティングシステムの機能
  - ゲーティングシステムの機能は、実際のイベント会場での運用要件を考慮し、1 イベントにおける複数ゲート改札に対応し、かつ各ゲートにはイベントチケットの座席情報等

を考慮したゲート個別の改札条件の設定を可能とした

（６）課題

今年度構築されたセキュア基盤ネットワークシステムを検証したことにより、電子的価値情報流通のためのセキュア基盤ネットワークとして求められるさらなる技術的課題についても確認された。

- 電子的価値情報の標準化に関する課題
  - 今回検討した汎用電子的価値情報フォーマットが対応し、かつダウンロード／転々流通／権利行使の仕組みについて未実装である以下のチケットへの対応
    - ✧ 複合型／縦列チケット
    - ✧ 複合型／並列チケット
  - 汎用電子的価値情報フォーマットのさらなる拡張性確保
  - 電子的価値情報の流通及び権利行使に関する利用条件等の制御

- 電子的価値情報の発行及び更新に関する課題
  - チケットが電子的価値情報となることにより紙媒体のチケットでは実施不可能であった機能として、電子的価値情報発行後に情報更新を行うなど、電子化することによる新たな付加価値の検討及び実装

- 電子的価値情報の保管に関する課題
  - セキュア基盤ネットワーク上に電子的価値情報保管サーバを構築して保管するなど、クライアントのセキュア IC チップ内に保持可能な電子的価値情報の数に関する制約を持たない方式の検討

- 権利行使に関する課題
  - 電子的価値情報の転々流通に関する制約等の制御
  - 電子的価値情報（権利）のトレーサビリティーの検討
  - 電子的価値情報の分割、統合の検討

- クライアント端末に関する課題
  - 今回対象としたクライアント端末形態だけでなく、インターネット及びセキュア基盤ネットワークへの接続が可能な

全てのクライアント端末の参加できる方式の確立。例えば、セキュア IC チップを内蔵したモバイル端末への拡張検討など

- セキュア基盤ネットワークノードのセキュリティに関する課題
  - サーバノードの耐タンパを実現するため、サーバ記憶領域に対し物理的なセキュリティ施策が必要
  - セキュリティ機能が強化された Trusted OS をサーバ OS として導入するなどの対策の実施

- 既存の価値情報プロバイダとの連携容易性確保に関する課題
  - 既存コンテンツプロバイダの現行業務運用の継続利用を確保するため、紙チケットとして販売されるチケットと電子的価値情報として販売されるチケットの管理業務の統合化
  - イベント中止及び延期時に発生するチケット払戻しに対応するための、発行した電子的価値情報の回収及び払戻し方式に関する検討

- ゲーティングシステム機能に関する課題
  - 会場やイベントの特性に合わせたゲーティング機能の拡張
  - ゲーティング時に電子的価値情報を所有したユーザの個人属性を取得して、サービスの向上やマーケティングに活用するなど、新たな付加価値を提供可能な方式の検討
  - 販売されたチケットの改札が即時に実施できる仕組みの検討

（7）まとめ

今年度構築したセキュア基盤ネットワークシステムにより、ユビキタスネットワークにおけるセキュアサーバノードの基礎を作ることができた。ただし、項番（6）で明らかにしたような、新たな課題も見つかったため、来年度以降も引き続き、課題解決に向けて、本セキュア基盤ネットワークシステムの研究を進めたい。

5.5 バイオメトリクス型認証（システム統合技術）

バイオメトリクス認証技術は、ユビキタスコンピューティングにお

ける情報セキュリティ技術の一要素として重要である。今回、バイオメトリクス技術の脅威と脆弱性の分析、およびバイオメトリクス技術に関する標準化動向の調査を行った。

5.5.1 バイオメトリクス技術の概要

バイオメトリクス認証技術は、従来、アクセス制御におけるパスワード代替形態が主であったが、現在は多くの情報がネットワークを介して共有化され、またサービスが提供されている。このため、アクセス制御における許認可を確認するための手段から、非対面の状況でサービスに対する利用者課金などを行なう際の本人を同定するための本人認証手段としての位置付けとなっている[1]。

表 1 に代表的なバイオメトリクスとその特徴を示す。バイオメトリクスには身体的な特徴と行動的な特徴の二種類があり、前者は指紋／掌形／顔／虹彩などが相当し、後者は声紋／署名などが相当する。発生や筆記は随意的な要素があるために声紋、署名は上記の身体的特徴の生体計測的なバイオメトリクスと異なり行動計測的な特徴と呼ばれる[2][3][4][5]。

表１　生体認証技術の比較

	生体情報	一般性	携帯性	永続性	収集性	精度	受容性	脅威耐性
身体的特徴	指紋	Medium	High	High	Medium	High	Medium	High
	顔	High	Low	Medium	High	Low	High	Low
	顔の赤外画像	High	High	Low	High	Medium	High	High
	虹彩	High	High	High	Medium	High	Low	High
	網膜	High	High	Medium	Low	High	Low	High
	静脈	Medium	Medium	Medium	Medium	Medium	Medium	High
	掌形	Medium	Medium	Medium	High	Medium	Medium	Medium
	耳	Medium	Medium	High	Medium	Medium	High	Medium
	匂い	High	High	High	Low	Low	Medium	Low
	ＤＮＡ	High	High	High	Low	High	Low	Low
行動的特徴	声紋	Medium	Low	Low	Medium	Low	High	Low
	動的署名	Low	Low	Low	High	Low	High	Low
	ｷｰｽﾄﾛｰｸ	Low	Low	Low	Medium	Low	Medium	Medium
	歩行	Medium	Low	Low	High	Low	High	Medium

5.5.2 バイオメトリクス技術の脆弱性分析

（ア）脅威と脆弱性の関係

本項目では､バイオメトリクスシステムの脅威と脆弱性の関係を分析することで、重点的に対策すべき脆弱性を明確化した。

まずバイオメトリクスの脆弱性を分析するにあたって､以下のように､バイオメトリクスシステム特有の脆弱性と、一般的な情報システムに共通する脆弱性の二つに分類した（表２）。

表２　バイオメトリクスシステムにおける脆弱性の分類

脆弱性の分類	内容
バイオメトリクスシステム特有の脆弱性	主に生体情報の特性に起因する脆弱性。脆弱性を利用するための難しさが生体情報の性質に依存するものを含む
一般的な情報システムに共通する脆弱性	他の個人認証・識別手段にも共通する脆弱性。主にバイオメトリクス認証･識別システム内のデータの漏洩や改ざんなど

次に脅威を分析するにあたり、バイオメトリクスシステムの脅威を、バイオメトリクスシステムに特有の脆弱性を利用する脅威と、一般的な脆弱性を利用した脅威に分類した（表３）。

表３　バイオメトリクスシステムにおける脅威の分類

脅威	内容
バイオメトリクスシステムに特有の脅威	バイオメトリクスシステム特有の脆弱性を利用する脅威
一般的な情報システムに共通する脅威	一般的な情報システムに共通する脆弱性のみを利用する脅威

これらを踏まえ、バイオメトリクスシステム特有の脆弱性の一覧を表４に、一般的な脆弱性の一覧を表５に示す。表はそれぞれ、脆弱性の名称と定義を示している。生体情報に特有の性質に起因する脆弱性、および脆弱性の程度が生体情報の性質に依存するものについては、バイオメトリクスシステム特有の脆弱性に分類した。生体情報の性質に特に関連がなく、他の個人認証・識別にも同様に存在する脆弱性は、一般的な脆弱性に分類した。

表４　バイオメトリクスシステム特有の脆弱性

脆弱性名称	定義
習熟	バイオメトリクスシステムの使用にあたって習熟が必要である脆弱性
受容性	バイオメトリクスシステムの使用に抵抗感を感じる利用者が存在する脆弱性
他人受入	他人受入が偶発的に発生する脆弱性
本人拒否	本人拒否が偶発的に発生する脆弱性
未対応	生体認証を使用できない人または生体情報が存在する脆弱性
利用者状態	利用者の状態変化により精度が変動する脆弱性
入力環境	生体情報の入力環境の変化により精度が変動する脆弱性
Wolf	Wolf により高確率な他人受入が発生する脆弱性
Lamb	Lamb により高確率な他人受入が発生する脆弱性
Goat	Goat により高確率な本人拒否が発生する脆弱性
認証パラメータ	認証パラメータの設定によって不適当な精度になる脆弱性
偽生体情報	物理的に偽の生体情報を取得できる脆弱性
公開	他人が利用者の生体情報を取得できる脆弱性
推定	テンプレートや照合結果から生体情報が推定できる脆弱性
限度	利用者ごとの生体情報の数に限界がある脆弱性
類似性	生体情報の類似した利用者が存在する脆弱性

表５　他の個人認証・識別にも共通する脆弱性

脆弱性名称	定義
登録	登録時の本人確認における脆弱性
単独	生体情報を単独で使用した場合、他人の ID に対して道具なしで攻撃可能な脆弱性
代替手段	生体認証を使用できない人または生体情報が存在する脆弱性
提供	本人の意思で生体情報を第三者に提供できる脆弱性
動機	バイオメトリクスシステムの利用者が認証あるいは識別される意志をもって入力を行なわなければならない脆弱性
センサ露出	生体情報を採取するセンサがシステム外部にさらされている脆弱性
データ漏洩	バイオメトリクス認証・識別システム内部のデータが外部に漏洩する脆弱性
サイドチャネル	バイオメトリクス認証・識別システムにおける何らかの情報が外部に漏洩する脆弱性
データ改ざん	バイオメトリクス認証・識別システム内部のデータを改ざんできる脆弱性
構成管理	システムを構成する要素の整合性が変化することにより、正常な動作や望んだ精度が得られない脆弱性
不活性化	ある条件を満たした場合、認証が一時的に受け付けられない脆弱性

バイオメトリクスの利用には、登録および運用（認証・識別）のフェーズがある。各フェーズにおける脅威は、同じ脆弱性を利用していても、異なった攻撃方法をとる場合が多い。そこで脅威の分析にあたっては、バイオメトリクスを登録フェーズと運用フェーズの二つに分け、各フェーズにおける脅威を分析した（表６、表７）。

表６　登録フェーズにおける脅威

名称	内容
Lamb によるバックドアの生成	偶発的あるいは意図的に Lamb（誰とでも一致するテンプレートの持ち主）を正当な利用者として登録することでバックドアを成し、なりすましを行う脅威。
偽生体情報によるバックドアの生成	偽生体情報を登録することでバックドアを生成し、なりすましを行う脅威。
登録時のなりすましによるバックドアの生成	攻撃者が正当な利用者になりすまして、生体情報を登録することで、攻撃者が認証を得られるバックドアを生成することで、なりすましを行う。
認証パラメータの不適当な設定によるバックドアの生成	偶発的あるいは意図的に、誰でも認証を許可するような認証パラメータを設定することで、バックドアを生成し、なりすましを行う脅威。
データ改ざんによるバックドアの生成	バイオメトリクス認証・識別システムの内部データ（生体情報やテンプレート）を不正に改ざんすることで、バックドアを生成し、なりすましを行う脅威。
登録者と生体情報提供者の不一致による可用性の阻害	意図的あるいは偶発的に登録者と生体情報提供者が不一致となることで可用性を阻害する脅威。
未対応による登録不可	利用者が（登録）未対応（FTE）のために可用性を阻害する脅威。
認証パラメータの不適当な設定による可用性の阻害	正当な利用者が生体情報を提示しても拒否するように、認証パラメータを設定することで、可用性を阻害する脅威。
使用拒否による可用性の阻害	ユーザが生体情報の提供に抵抗感を感じて、システムへの登録を拒否したために可用性の阻害が生じる。
データの改ざんによる可用性の阻害	バイオメトリクス認証・識別システム内部の生体情報やテンプレートを改ざんすることで可用性を阻害する脅威。

表７　運用（認証・識別）フェーズにおける脅威

名称	内容
他人受入によるなりすまし	偶発的な他人受入が発生するまで入力を繰り返すことでなりすましを行う脅威。
Wolf によるなりすまし	Wolf である攻撃者の生体情報を提示することで、なりすましを行う。
Lamb によるなりすまし	Lamb の ID に対して他人受入が発生するまで入力を繰り返すことでなりすましを行う脅威。
脅迫によるなりすまし	脅し等により正当な利用者に生体情報を提示させることで、なりすます。
偽生体情報によるなりすまし	正当な利用者の生体情報を何らかの方法で入手し、偽生体情報を作成するこでなりすましを行う。
認証パラメータの不正変更によるなりすまし	誰が生体情報を提示しても認証されるよう、認証パラメータを設定することで、だれでもなりすましができる脅威。
代替手段によるなりすまし	攻撃者が、意図的な本人拒否を引き起こしたり、未対応者を装うことで、弱い代替手段を利用し、なりすます脅威。
生体情報の推定によるなりすまし	何らかの方法で生体情報を推定することで、なりすましを行う脅威。
類似した生体情報をもつ利用者へのなりすまし	親や兄弟など、類似した生体情報をもつ利用者へのなりまし。
（入力）未対応による可用性の阻害	利用者が（入力）未対応（FTA）のために可用性の阻害が生じる。
Goat による可用性の阻害	Goat の利用者が本人拒否を頻発し、可用性を阻害する。
不活性化による可用性の阻害	本人認証・識別システムの不活性化の条件が意図的あるいは偶発的に満たされることで、正当な利用者が認証を受けられなくなり、可用性が阻害される脅威。
センサの破壊による可用性の阻害	生体情報を採取するセンサ自体が意図的あるいは偶発的に破壊されることで可用性がそがいされる。一般的な情報システムにも共通する脅威である。
希薄な動機によ	利用者の認証に対する動機が希薄なため、正しく生体

る可用性の阻害	情報を入力しないことによる可用性の阻害。
データの改ざんによるなりすまし	バイオメトリクス認証・識別システムの内部データ（生体情報、テンプレート、認証パラメータ）を不正に改ざんすることで、なりすましを行う。一般的な情報システムにも共通する脅威である。
構成変更	意図的あるいは偶発的にバイオメトリクス認証・識別システムの構成が変更されることで、異常な高頻度で本人拒否率を引き起こし、可用性が阻害される。一般的な情報システムにも共通する脅威である。
利用者状態の変化によるなりすまし	利用者状態が意図的あるいは偶発的に変化することで、なりすましが発生する脅威。
利用者状態の変化による可用性の阻害	利用者状態が意図的あるいは偶発的に変化することで、異常に高い確率で本人拒否を引き起こし、可用性が阻害される脅威。
サイドチャネルを利用した可用性の阻害	照合あるいは登録処理負荷の大きい生体情報を入力することで、バイオメトリクス認証・識別システムの可用性を阻害する脅威。
サイドチャネルを利用したなりすまし	ログデータなどから他人受入率の大きい利用者を特定して、なりすましを行う脅威。
本人拒否による可用性の阻害	偶発的に生じる本人拒否により可用性が阻害される脅威。

これらの分析結果により、ある脅威を防ぐためにどの脆弱性に対策を施すべきか、あるいはどの脆弱性に対策を施すことで効果的に脅威を防ぐことができるか、などを明らかにすることができる。

（イ）考察

（ア）ではバイオメトリクスシステムにおける脅威を洗い出し、登録フェーズおよび運用フェーズに分類して、具体的な実現方法と関連する脆弱性を示した。これによりある脅威についてどの脆弱性に対応すべきかが明確になった。

しかし、バイオメトリクスシステムにおいて、すべての脅威に対策するのはコストの面から現実的ではない。実際には、リスクマネージメントの考え方から、リスクの大きい脅威に対して適切な対策を施し

ていく必要がある。そのため、おのおのの脅威と脆弱性に関して、リスクを明確化すること（リスクアセスメント）が必要である。

リスクは、脅威と、それに対応する脆弱性の程度から次式のように定義できる。

リスク　＝　脅威　×　脆弱性

したがって、リスクアセスメントにあたっては、まず脅威と脆弱性の程度を明確化する必要がある。現状、脅威や脆弱性の程度を評価する共通的な基準は存在しないため、以下が課題となる。

37. 脅威と脆弱性の程度に関する評価基準の策定
38. バイオメトリクスシステムおよび生体情報ごとに細分化したリスク評価
39. 生体情報の脆弱性の程度を評価するための実験

5.5.3 バイオメトリクスシステムのセキュリティ要件と評価の枠組み

情報システムのセキュリティとは、想定されるリスクに対して効果的な安全対策を漏れなく行うことである。ファイヤーウォールや暗号装置などのセキュリティ技術においては、それぞれの装置やシステムの安全対策を保証するための国際的な評価標準である ISO15408 に準拠した製品開発が進んでいる。一方、バイオメトリクスは画像処理の応用技術として開発されたこともあり、照合精度の仕様のみが着目される傾向があり、情報セキュリティの観点での評価がされてこなかった。しかし、欧米では、ISO15408 に準拠した評価手引書（要件仕様書）プロテクションプロファイル（Protection Profile）の開発が進んでおり、セキュリティ技術の観点でバイオメトリクスを評価する取り組みが始まっている。

生体情報を用いた本人認証技術は、原子力関連施設や金融機関など高い安全性が要求される設備などへの入退室管理に応用されてきた。このような物理的アクセスコントロールでは、バイオメトリクス認証装置を壁に埋め込めるなどのセキュリティ対策により装置の改ざんや解析が困難であり、バイオメトリクス認証装置自身の安全性を確保できる。これに対して、インターネットや携帯網などを介したマスユーザに対する情報サービスを想定すると、バイオメトリクス認証装置

の管理はユーザに委ねられる場合もあり、バイオメトリクス認証装置自身の安全性を確保することが求められる。また、組織のセキュリティ対策として他のセキュリティ技術と組み合わせて生体認証技術を導入する場合、安全性のレベルを他のセキュリティ技術と同じレベルであることを保証できないと生体認証技術がセキュリティホールになりかねない。他のセキュリティ技術との安全保証上の相互接続性を保つ意味でも、ISO15408 への準拠は重要となる。

以下に、英国 CESG の BWG が開発した BDPP のドラフト、および米国国防総省の BMO(:Biometrics Management Office)が開発した DoD BSPP について概要を示す。

表８　プロテクションプロファイルの例

名称	概要
BDPP（英国 CESG）	保護の対象となる資産が格納されている（物理的あるいは論理的な）場所への入口からの入場に際して、入場者が誰であるか、あるいは既に登録済みであるかを認証するために使用される商用のバイオメトリクス認証装置に適用できる機能要件、および、保証要件を定めることを目的に開発された。 また EAL1 に追加を施した EAL1+から EAL4 までの４段階の安全性保証レベルを明らかにしている。
BSPP（米国 DoD）	国防システムなどへの適用を目的としており、より高度のセキュリティ保証レベル EAL5+の仕様策定を視野に入れている。 評価対象は、大きく登録処理と照合処理に機能が分割されている。登録処理は、生体情報を入力する機能と、特徴抽出する機能と、格納部への通信と保管の情報秘匿を目的とした暗号機能と、格納部へ登録する機能とで構成されている。照合処理は、生体情報を入力する機能と、特徴抽出する機能と、照合部への通信情報の秘匿を目的とした暗号機能と、復号と照合する機能と、入口の開閉などアプリケーションとのインターフェース部で構成されている。

また、バイオメトリクスシステムの評価方法としては、ISO15408 の評価における標準評価方法(Common Evaluation Methodology:CEM)をバイオメトリクスに適用する際に考慮すべき追加要件をまとめたものとして、The Common Criteria Biometric Evaluation Methodology Working Group/CESG による BEM(Biometrics Evaluation Methodology)がある。BEM は以下の二点を、バイオメトリクスシステムの評価において特に注意すべき点として指摘している。

（ｉ）バイオメトリクス特有の脅威と脆弱性

例えば、生体情報の偽造など、評価において、これらの脅威を検討し、TOE（Target of Evaluation）の Penetration Testing で明確化する必要がある。

（ii）統計的な性能試験

例えば、他人受入率（False Accept Rate:FAR）などが重要であり、EAL1 にセキュリティ機能強度（Strength of Function:SOF）として精度評価を追加すべきである。

5.5.4 バイオメトリクス技術の標準化の動向

従来、バイオメトリクスの互換性、精度評価、安全性保証に関する標準化は主に各国で独自に進められていた。例えば、互換性に関しては、バイオメトリクスデータフォーマットの米国 NIST 規格"CBEFF"（Common Biometrics Exchange File Format）[6]およびアプリケーションプログラムインターフェースの米国 ANSI 規格"BioAPI"[7][8]、精度評価方法に関しては、英国 CESG（Communication-Electronics Security Group）の"Best Practicein testing and reporting performance of biometric devices"[9]、日本規格協会情報技術標準化センター(JSA/INSTAC)の「指紋認証システムの精度評価基準 TR X0053」[10]、安全性保証に関しては、英国 CESG の"BDPP"[11]、米国国防省（DoD）の"BSPP"[12]などがある。

これらのバイオメトリクスに関する規格を国際的な標準にする目的で、ISO/JTC1 SC37 "Biometrics"が設立された。今後バイオメトリクスに関する標準化は SC37 を中心に行なわれる。SC37 以外の国際的な標準化組織としては、IC カードに関する標準を策定する ISO/JTC1 SC17、国際民間航空機関（ICAO:International Civil Aviation Organization）があり、IC カードにおけるバイオメトリクスの標準インタフェース"ISO7816-11"や、空港でのバイオメトリク

ス利用に関する要求仕様やデータフォーマットなどの策定を行なっている。

参考文献

[1] 瀬戸洋一:バイオメトリクスを用いた本人認証技術、計測と制御、Vol,37No,5,pp,395-401(1998)

[2] David D. Zhang:Automated Biometrics Technologies and Systems, Kluwer Academic Publishers(2000)

[3] 日本自動認識システム協会編:これでわかったバイオメトリクス,オーム社(2001)

[4] 赤松茂:コンピュータによる顔の認識サーベイ,信学論 A, Vol.J80-D Ⅱ,No.8,pp.1215-1230

[5] 金子正秀:顔による個人認証の最前線,映像情報メディア学会誌,Vol.55,No.2,pp.180-184

[6] NISTIR6529 ”Common Biometric Exchange File Format(CBEFF)”, NIST, 2001/1/3, http://www.itl.nist.gov/div895/isis/cbeff/CBEFF010301web.PDF

[7] ”BioAPI Specification version 1.1”, The BioAPI Consortium, 2001/3/16, http://www.bioapi.org/BIOAPI1.1.pdf

[8] Catherine Tilton, ”The BioAPI Specification: Recipe for Interoperability & Interchangeability”, Biometric Consortium Conference, 2002/2

[9] Best Practices in Testing and Reporting Performance of Biometric Devices Version 1.0,Jan.2000, Biometrics Working Group, Communications-Electronics Security Group, UK

[10] 「指紋認証システムの精度評価方法」TR X 0053, 日本規格協会, 2002 年

[11] ”Biometrics Device Protection Profile (Draft Issue 0.81)”, CESG, 2001/1/31

[12] ”Draft Department of Defense & Federal Biometric System Protection Profile for Medium Robustness Environments (ver 0.02)”, NIST, 2003/3/3

	ノードA			ノードB
		(1) $cert_A$ →		
			(2)	**Veri**[$cert_A$] = 1?
			(3)	$R_B \in_R Z$
			(4)	r_B = **E_A**[R_B]
		← (5) $cert_B$, r_B		
(6)	**Veri**[$cert_B$] = 1?			
(7)	R'_B = **D_A**[r_B]			
(8)	s_A = **Sig_A**[R'_B]			
(9)	$R_A \in_R Z$			
(10)	x_A = **E_B**[s_A]			
(11)	r_A = **E_B**[R_A]			
		(12) x_A, r_A →		
			(13)	s_A' = **D_B**[x_A]
			(14)	R_A' = **D_B**[r_A]
			(15)	**Veri**[R_B, s_A'] = 1?
			(16)	s_B = **Sig_B**[R_A']
			(17)	x_B = **E_A**[s_B]
		← (18) x_B		
(19)	s_B' = **D_A**[x_B]			
(20)	**Veri**[R_A, s_B'] = 1?			

図 5.4-14: デジタル署名＆復号処理に基づく認証プロトコル～方式 I

●方式 I：デジタル署名＆復号処理に基づく認証プロトコル

公開鍵暗号に基づくデジタル署名及び復号処理を組み合わせたプロトコル。処理の流れを以下に記す。

A → B：

(1) ノード A は自分の公開鍵証明書 $cert_A$ をノード B に送信。

B の処理：

(2) 受信した A の公開証明書の正当性を検証。

(3) 乱数 R_B を生成。

(4) A の公開鍵を用いて、生成した乱数 R_B を暗号化。

A ← B：

(5) ノード B は自分の公開鍵証明書 $cert_B$ と暗号化乱数 r_B をノー

ドA に送信。

A の処理：

(6) 受信したB の公開鍵証明書の正当性を検証。

(7) 暗号化乱数 r_B を復号。

(8) A の秘密鍵を用いて、復号乱数 R_B' へ署名。

(9) 乱数 R_A を生成。

(10) B の公開鍵を用いて、署名 s_A を暗号化。

(11) B の公開鍵を用いて、生成した乱数 R_A を暗号化。

A → B：

(12) ノードA は暗号化署名 x_A と暗号化乱数 r_A をノードB に送信。

B の処理：

(13) B の秘密鍵を用いて、暗号化署名 x_A を復号。

(14) B の秘密鍵を用いて、暗号化乱数 r_A を復号。

(15) 生成乱数 R_B と A の公開鍵を用いて、復号した署名 s_A' の正当性を検証。

(16) B の秘密鍵を用いて、復号乱数 R'_A へ署名。

(17) A の公開鍵を用いて、署名 s_B を暗号化。

A ← B：

(18) ノードB は暗号化署名 x_B をノードA に送信。

A の処理：

(19) A の秘密鍵を用いて、暗号化署名 x_B を復号。

(20) 生成乱数 R_A と B の公開鍵を用いて、復号した署名 s_B' の正当性を検証。

□

【方式Ⅰにおける問題と考察】

デジタル署名のデータサイズや数値としてみた場合の大きさと、暗号化に用いる公開鍵との関係次第では、暗号化データを正常に復号できない場合が生じる。

その具体例として、RSA 署名とRSA 暗号化を適用した場合で論じる。簡単のため、署名と暗号化に用いる公開鍵と秘密鍵のペアを同一のものとする。また、ノードA のRSA 公開鍵のうち、法（二つの秘密素因数の合成数）を N_A、同じようにノードB の法を N_B とする。

問題となるのは、以下の処理。

(a) ノードB が、暗号化されたノードA の署名を検証する。

（関連する処理番号： 8, 10, 13, 15）

(b) ノードAが、暗号化されたノードBの署名を検証する。
(関連する処理番号： 16, 17, 19, 20)

ここで、(a)において「N_A>N_B」の場合、ノードAのデジタル署名 s_A とノードBの公開鍵 N_B を整数としてみた時、前者が後者よりも大きくなる場合がある。この時、ノードBの処理(13)で復号された値が、ノードAのデジタル署名 s_A と等しくならない。結果、処理(14)でノードAの本人性を認めることができず、処理がエラー終了する。
逆に、「N_A<N_B」の場合、(b)において同様のエラーが発生する。

同問題の解決法としては、以下のような方法が考えられる。

- デジタル署名の最大値が暗号化用の相手公開鍵より大きくならないような署名方式を採用する。例えば、1024 ビットの RSA 暗号を暗号化方式に選ぶ場合、法が 1024 ビットの DSA や 160 ビットの楕円 DSA を選定する。
- 全ノードに共通の規則として、暗号化用と署名用とで異なる公開鍵暗号鍵ペアを設定する。その上で、RSA 暗号の場合では、暗号化用の法ビットサイズが署名用の法ビットサイズを上回るようにシステム全体で予め決めておく。例えば、暗号化用を 1024 ビット、署名用を 1023 ビットに設定する。これに伴い、公開鍵証明書にて両方の鍵を同時に承認する、あるいは独立の公開鍵証明書を検証するという具合に実装を合わせていく必要がある。

	ノードA			ノードB
		(1) $cert_A$		
			(2)	**Veri**[$cert_A$] = 1?
			(3)	$R_B \in_R Z$
			(4)	r_B = **E_A**[R_B]
		(5) $cert_B$, r_B		
(6)	**Veri**[$cert_B$] = 1?			
(7)	R'_B = **D_A**[r_B]			
(8)	h_A = **Hash** [R_B']			
(9)	$R_A \in_R Z$			
(10)	x_A = **E_B**[R_A | h_A]			
		(11) x_A		
			(12)	R_A' | h_A' = **D_B**[x_A]
			(13)	**Hash** [R_B] = h_A' ?
			(14)	h_B = **Hash** [R_A']
			(15)	x_B = **E_A**[h_B]
		(16) x_B		
(17)	h_B' = **D_A**[x_B]			
(18)	**Hash** [R_A] = h_B' ?			

図 5.4-15: ハッシュ関数を用いた認証プロトコル～方式 II

●方式 II：一方向性ハッシュ関数を組み合わせた認証プロトコル

暗号化／復号処理とハッシュ関数を組み合わせ、処理の高速化を狙ったプロトコル。また、認証時の署名処理を無くすことで、方式 I にあった復号エラー問題を回避する。

処理の流れを以下に記す。

A → B：

(1) ノード A は自分の公開鍵証明書 $cert_A$ をノード B に送信。

B の処理：

(2) 受信した A の公開鍵証明書の正当性を検証。

(3) 乱数 R_B を生成。

(4) A の公開鍵を用いて、生成した乱数 R_B を暗号化。

A ← B：

(5) ノード B は自分の公開鍵証明書 $cert_B$ と暗号化乱数 r_B をノー

ドAに送信。

Aの処理：

(6) 受信したBの公開鍵証明書の正当性を検証。

(7) 暗号化乱数 r_B を復号。

(8) 復号した乱数 R_B' のハッシュ値 h_A を計算。

(9) 乱数 R_A を生成。

(10) Bの公開鍵を用いて、生成乱数 R_A とハッシュ値 h_A を暗号化。

A → B：

(11) ノードAは暗号化データ x_A をノードBに送信。

Bの処理：

(12) Bの秘密鍵を用いて、暗号化データ x_A を復号。

(13) 生成乱数RBのハッシュ値が復号したハッシュ値hAに一致するか検証。

(14) 復号した乱数 R_A' のハッシュ値 h_B を計算。

(15) Aの公開鍵を用いて、ハッシュ値 h_B を暗号化。

A ← B：

(16) ノードBは暗号化データ x_B をノードAに送信。

Aの処理：

(17) Aの秘密鍵を用いて、暗号化データ x_B を復号。

(18) 生成乱数 R_A のハッシュ値が復号したハッシュ値 h_B に一致するか検証。

□

	ノード A			ノード B
		(1) $cert_A$ →		
			(2)	**Veri**[$cert_A$] = 1?
			(3)	$R_B \in_R Z$
			(4)	r_B = **E_A**[R_B]
		← (5) $cert_B$, r_B		
(6)	**Veri**[$cert_B$] = 1?			
(7)	R'_B = **D_A**[r_B]			
(8)	s_A = **Sig_A**[R'_B]			
(9)	$R_A \in_R Z$			
(10)	sa_1 = **msb_t**[s_A]			
(11)	sa_2 = **lsb_{n-t}**[s_A]			
(12)	xa_1 = **E_B**[R_A | sa_1]			
(13)	xa_2 = **E_B**[sa_2]			
		(14) xa_1, xa_2 →		
			(15)	R_A' | sa_1' = **D_B**[xa_1]
			(16)	sa_2' = **D_B**[xa_2]
			(17)	**Veri**[R_B, sa_1' | sa_2'] = 1?
			(18)	s_B = **Sig_B**[R_A']n
			(19)	sb_1 = **$msb_{t'}$**[s_B]
			(20)	sb_2 = **$lsb_{n-t'}$**[s_B]
			(21)	xb_1 = **E_A**[sb_1]
			(22)	xb_2 = **E_A**[sb_2]
		← (23) xb_1, xb_2		
(24)	sb_1' = **D_A**[xb_1]			
(25)	sb_2' = **D_A**[xb_2]			
(26)	**Veri**[R_A, sb_1' | sb_2'] = 1?			

図 5.4-16：復号エラーを解決する認証プロトコル～方式 III

●方式 III：方式 I の復号エラーを解決する認証プロトコル

方式 I における復号エラーを無くすため、デジタル署名を二分割して、送信する乱数と合わせて、相手公開鍵を超えないようサイズのバランスを取った 2 つのデータを計算する。その上で、各データに各暗号化処理を施すプロトコル。

処理の流れを以下に示す。

A → B：

(1) ノード A は自分の公開鍵証明書 $cert_A$ をノード B に送信。

B の処理：

(2) 受信した A の公開鍵証明書の正当性を検証。

(3) 乱数 R_B を生成。

(4) A の公開鍵を用いて、生成した乱数 R_B を暗号化。

A ← B：

(5) ノード B は自分の公開鍵証明書 $cert_B$ と暗号化乱数 r_B をノード A に送信。

A の処理：

(6) 受信した B の公開鍵証明書の正当性を検証。

(7) 暗号化乱数 r_B を復号。

(8) A の秘密鍵を用いて、復号乱数 R_B' へ署名。

(9) 乱数 R_A を生成。

(10) 署名 s_A の上位 t(bit)を sa_1 に格納。

(11) 署名 s_A の下位 n-t(bit)を sa_2 に格納。

(12) B の公開鍵を用いて、生成乱数 R_A と部分署名 s_A のビット連結データを暗号化。

(13) B の公開鍵を用いて、部分署名 sa_2 を暗号化。

A → B：

(14) ノード A は暗号化データ xa_1 及び xa_2 をノード B へ送信。

B の処理：

(15) B の秘密鍵を用いて、暗号化データ xa_1 を復号。

(16) B の秘密鍵を用いて、暗号化データ xa_2 を復号。

(17) A の公開鍵を用いて、部分署名 sa_1 と sa_2 をビット連結させた署名 s_A' の正当性を検証。

(18) B の秘密鍵を用いて、復号乱数 R_A ’へ署名。

(19) 署名 s_B の上位 t′(bit)を sb_1 に格納。

(20) 署名 s_B の下位 n-t′(bit)を sb_2 に格納。

(21) A の公開鍵を用いて、部分署名 sb_1 を暗号化。

(22) A の公開鍵を用いて、部分署名 sb_2 を暗号化。

A ← B：

(23) ノード B は暗号化データ xb_1 及び xb_2 をノード A に送信。

A の処理：

(24) A の秘密鍵を用いて、暗号化データ xb_1 を復号。

(25) A の秘密鍵を用いて、暗号化データ xb_2 を復号。

(26) B の公開鍵を用いて、部分署名 sb_1 と sb_2 をビット連結させ

た署名 s_A' の正当性を検証。

□

(2-3) 各方式の比較分析と考察

各方式 I、II、III において、各ノードで必要な PKI 処理 (暗号化、復号、署名、検証)、ハッシュ処理数を以下に示す。

表 5.4-26: 各方式における片側ノードの処理コスト

方式	PKI 処理				ハッシュ処理※2
	暗号化	復号	署名	検証※1	
方式 I	2	2	1	2	-
方式 II	1(A) 2(B)	2(A) 1(B)	-	1	1
方式 III	2(A) 3(B)	3(A) 2(B)	1	2	-

注 1) 公開鍵証明書の検証も含む。

注 2) デジタル署名処理内部のハッシュ処理は含まない。

方式 II や方式 III では、ノード A とノード B とで各暗号処理の必要性が異なる。例えば、方式 II の暗号化はノード A で 1 回、ノード B で 2 回必要なことを示している。こうした処理の種類と必要数の違いを、秘密鍵を用いた計算より公開鍵を用いた計算の方が処理が軽いといった各公開暗号方式の特徴と照らし合わせることも重要である。その上で計算力の劣ったノードが処理コストの小さい方を担い、より計算量の高いサーバ側でその逆側で認証を行う実装も有効である。

（3）小型セキュアチップによる VPN 機能の提供

耐タンパ性を備えた小型セキュアチップを駆使して、ネットワークを介して相手ノード（ネットワーク）とセキュリティが確保された VPN (Virtual Private Network)を構築したい。小型セキュアチップは例えば、IC カードといった携帯できる媒体で、非接触／接触チャネルを通じて利用者端末に接続する環境を想定する。

このとき、理想的には、耐タンパ性を備えたセキュアチップ内部で、以下に示すような暗号処理全てを請け負うことが望ましい。

- 相手ノードとの相互認証

- 認証後の暗号化データ通信用の暗号鍵秘密共有
- データ暗号化／復号
- データ認証（MAC 生成／検証、署名／検証）

暗号化や認証子（MAC、署名）を付与する相手ノードへのデータには、利用者端末内部の情報（アプリケーションデータやコンテンツデータ等）に限らず、セキュアチップ内のデータも含まれる。セキュアチップ内で管理する情報には、セキュリティ情報に加え、金銭や権利、資格といった価値ある電子情報が考えられる。

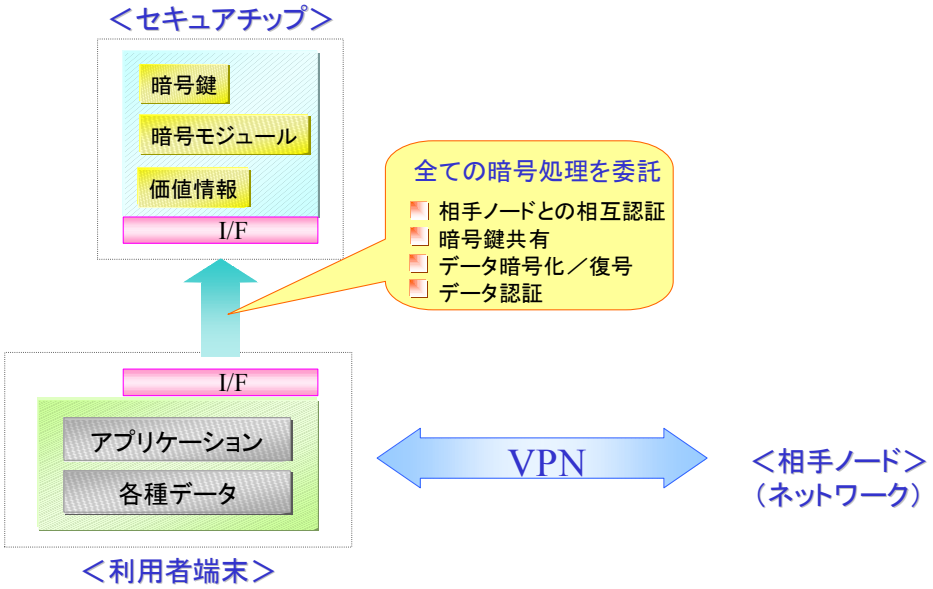


図 5.4-17: 理想的な VPN モデル

この理想形での利点の一つとして、利用者端末側が VPN 構築または構築後の暗号処理、暗号鍵情報の厳重管理を全てセキュアチップに任せてしまえる点である。特に暗号鍵情報については、あらゆる脅威から当該データを守るための強固な耐タンパ性を利用者端末に課す必要がない。これにより、ハードウェア設計時のコストを削減できる。

逆に利用者からすると、セキュアチップを IC カードといった媒体で所持すれば、暗号モジュールを持たない様々な端末からも相手ノードとセキュアな暗号化通信路を構築することができる。

しかし、チップの処理速度や外部とのインターフェースの大域幅を考慮すると、チップ内で大きなデータをリアルタイムに実用レベルで暗号処理するのは現状で厳しい。

そこで、セキュアチップだけでなく、接続する機器も含めた全体的な即時応答性を考え、今回、次のようなアーキテクチャを検討し、チ

ップ実装への研究開発を行った。ただし、ここではセキュアチップに比べ、利用者端末の方が計算処理速度、メモリ領域等において優れていることを想定している。

- セキュアチップは、利用者端末を通じて相手ノードとの相互認証を行う。
- セキュアチップは、相互認証の中で鍵共有を行い、以上の処理が正常に行われた場合に限り、この共有秘密情報を利用者端末へ提供する。
- 実際の VPN 通信処理は、セキュアチップより提供された VPN 用の暗号鍵情報を用いて、利用者端末内部で行う。

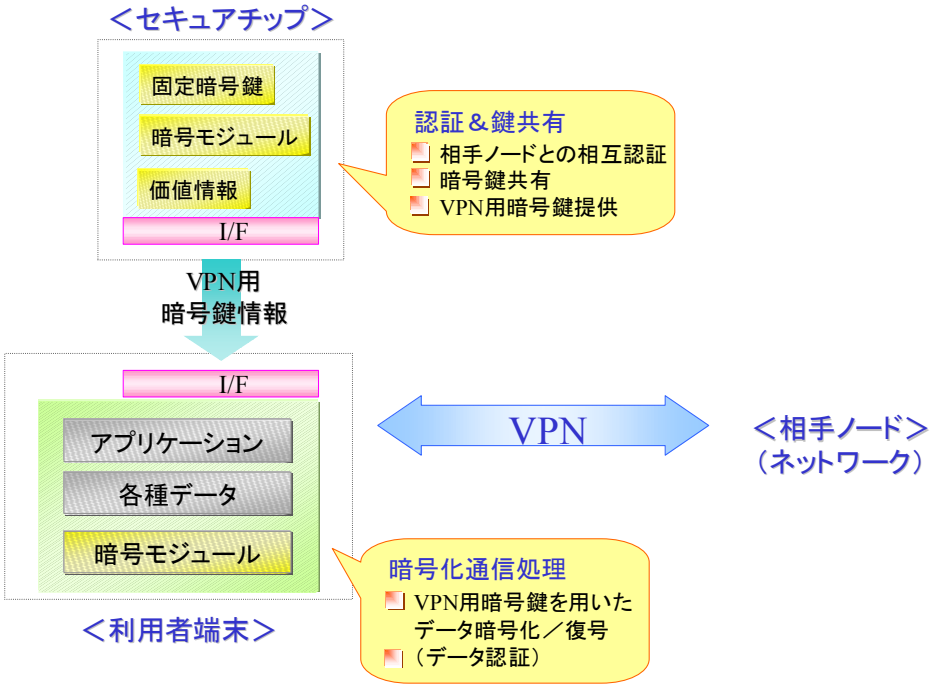


図 5.4-18: チップ実装に向けた VPN モデル

通信相手先は同じくセキュアチップを搭載した端末や、大きな計算力とデータ蓄積を供えたサーバ端末が考えられる。特に前者の場合だと、異なる利用者端末間で互いのセキュリティチップを用いて、気軽に“私的な”暗号化通信を行うことができる。セキュリティチップを所持し、暗号モジュールを搭載している任意の端末で VPN 通信を利用できる。

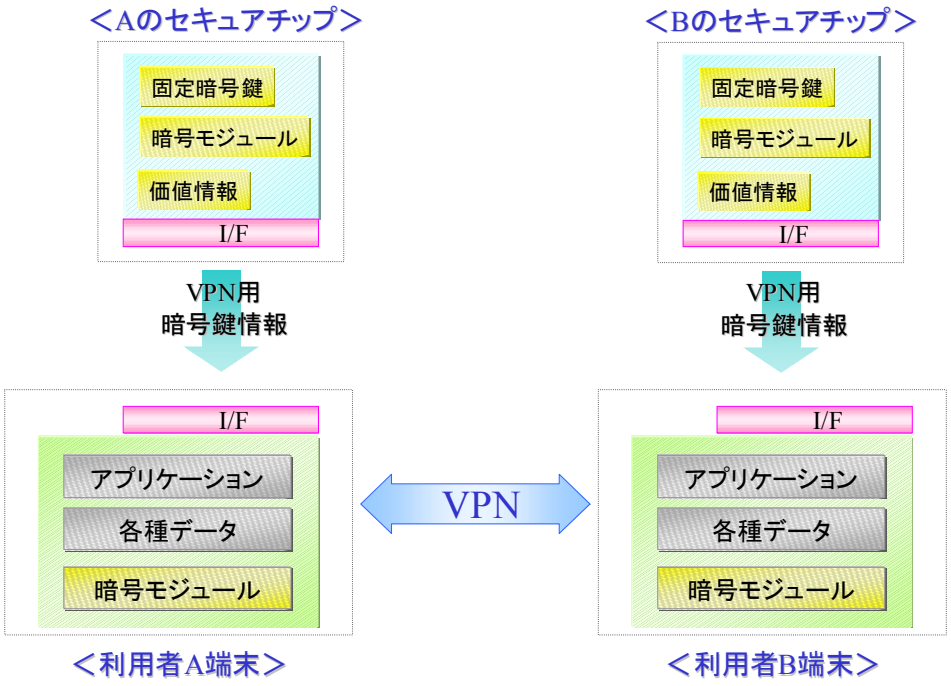


図 5.4-19: 異なる利用者端末間の VPN モデル

理想形モデルと実装に向けた VPN モデルとの大きな違いは、暗号モジュールを利用者端末に搭載し、内部で暗号処理を行うことである。認証や鍵共有のための暗号鍵情報はセキュアチップで堅牢に保護される。しかし、暗号化通信用の鍵が端末の管理下に置かれる以上、これを外部からの脅威から、ある規定レベルで防御する仕組みが必要となる。逆にいくらセキュアチップが堅牢であろうと、この利用者端末側管理・設計が杜撰であると、VPN を構築する本来の意図が失われてしまう。

アプリケーションレベルでのセキュリティホールへの対策、暗号モジュールの耐タンパ性パッケージ化といった様々なセキュリティ技術要素を組み込んだセキュア端末を想定している。その上で、認証と鍵共有までをセキュアチップ、その後の VPN 通信処理を計算処理能力の高い利用者端末と役割を明確に分けた実用モデルである。

（４）コンパクトな公開鍵証明書の設計

リアルタイム PKI を考える上で、認証の基盤となる公開鍵証明書をどう設計するかは重要な研究項目である。ユビキタスネットワークに遍在する処理能力及び記憶容量の限られた小型チップ、その外部インタフェースの大域幅、さらにリアルタイム性を追求すると、公開鍵証明書の小型化が一つ検討すべき項目として浮上してくる。

ユビキタス環境に応じた、従来の標準的な証明書形式（X.509）の小型化については、5.4.1 節に記述した。そこでは、特に証明書のデータ構成要素単位で必要性の度合いを評価した。本節では、オフラインにて鍵の有効性を検証する上で必須の公開鍵情報について、各暗号方式まで掘り下げた証明書小型化の検討を行う。

公開鍵暗号において、“公開可能”な情報として次の二種類がある。一つは、ノード間で共通的に定義できるパラメタ、もう一方は各ノードの秘密鍵と前記の共通化可能パラメタを用いて導出された各ノードにユニークな公開鍵。前者については、例えば、RSA 暗号の公開冪乗指数 e や、Diffie-Hellman 鍵交換方式での原始元 g や法 p（素数）といったパラメタである。

本研究での基本的な考え方は、こうしたノード間に共通化可能なパラメタについては公開鍵証明書内部に含めず、EEPROM といった各ノードの記憶領域に共通パラメタとして格納することである。これにより、公開鍵証明書本体を軽量化し、公開鍵暗号に基づく認証プロトコル時の通信データサイズを削減する。

ここでは、RSA 暗号、Diffie-Hellman 鍵交換、DSA を例に挙げ、上記考えを反映した場合の効果を以下に示す。

■ RSA 暗号の公開鍵情報（数値は例）

表記	内容	サイズ（Bytes）	本研究での対応
N	合成数	128	128（Bytes）
e	公開冪数	128	EEPROM 格納
合計サイズ		256	128（Bytes）

RSA 暗号での公開冪数”e”を整数値の 3 や 65537（$=2^{16}+1$）と共通的に定める実装がある（前者の使用に関しては、セキュリティ評価がいくつかなされており、採用アルゴリズムに応じて注意も必要）。この方針に因んで、各ノード毎に異なる合成数 N のみを公開鍵として証明書内部に含める。公開冪数が小さい場合、他方式と比べ、適用効果は小さくなるが、本節で扱う暗号方式間で公開鍵のサイズ／性質を一元化する実装上の別効果もある。

■ Diffie-Hellman 鍵交換の公開鍵情報（数値は例）

表記	内容	サイズ（Bytes）	本研究での対応
p	素数	128	EEPROM 格納
g	Z_p^*の生成元	128	EEPROM 格納
y	公開鍵	128	128（Bytes）
合計サイズ		394	128（Bytes）

Diffie-Hellman 鍵交換では、通信ノード間で正しく共有鍵を計算できるように、素数 p や Z_p^*の生成元 g をノード間で共通値とする必要がある。よってこれらを共通パラメタとして EEPROM に格納する。このサイズ削減効果は大きい。

■ DSA の公開鍵情報（数値は例）

表記	内容	サイズ（Bytes）	本研究での対応
p	素数（p=n*q+1、n：整数）	128	EEPROM 格納
q	素数	20	EEPROM 格納
g	部分巡回群の生成元（位数 q）	128	EEPROM 格納
y	公開鍵	128	128（Bytes）
合計サイズ		404	128（Bytes）

DSA では、離散対数問題の困難性に基づく上記 Diffie-Hellman 鍵交換の場合と同じく、大きな効果を得ることができる。署名のサイズをコンパクトにする目的から、素数 p と関係（p=n*q+1）をもつ素数 q が共通パラメタとして加わる。ここで、n と q さえ既知であれば、p を算出することができる。計算速度より記憶領域の確保を優先する場合には、p をその場で計算してしまうという実装も考えられる。本節では、即時応答性という速度を重視する意向から、p も EEPROM へ格納する表記とした。

この他、ECDSA のように、実質的な公開鍵を含めた公開可能な鍵情報の中で、共通化可能なデータの占めるサイズ割合が高いほど、その適用効果も大きくなる。

ユビキタスネットワーク環境では、高密度で遍在するノード群で絶え間ない相互認証通信が行われる。故に、公開鍵証明書のコンパクト化は各認証通信の省リソース化に貢献できる。この一つ当たりの認証通信における効果は、ネットワーク全体としての通信トラフィックを抑制する意味で大きな意味を持つ。その中で、本節で論じた鍵情報の

整理は、証明書の小型化に大きく貢献できることが分かった。

上記研究によりコンパクト化した公開鍵証明書を別途研究項目である相互認証プロトコルや小型セキュアチップを用いた VPN 構築にフィードバックし、試作チップへの実装を行った。

5.6 超機能分散システム指向開発環境の整備

5.6.1 ワンボード型エンベディッドアーキテクチャの研究

(1) はじめに

本研究では、当研究所の研究開発における各種実験そして実用化に向けた標準開発プラットフォームとして、携帯型ノード開発システム(U-Card)および据え置き型ノード開発システム(μU-Card)の開発を進めている。昨年度に仕様策定およびハードウエア（ボード）の開発を行ったのに引き続き、今年度はそれを応用したシステムの試作と評価を行い、その実用性を検証した。

今年度試作した様々な応用システムのうち、ここでは代表的なものとして、

40. U-Card を応用したユビキタス情報端末
41. IC カードと連携したセキュリティシステム
42. IC カードと連携した電子実体流通システム

を説明する。

(2) U-Card を応用したユビキタス情報端末

(2-1) ユビキタス情報端末について

ユビキタス環境では、個人が携帯する情報端末を使ってあらゆるところに組み込まれたチップと通信を行い、各種情報へのアクセス、電子的価値情報の転送やサービス利用を行う。今回、様々な日常シーンにおける情報端末の利用を想定して、U-Card を応用した情報端末の実験・デモシステムを開発した。以下、今年度開発した主な実験システムを紹介する。

(2-2) 医薬品のチェックシステム

本実験システムでは、医薬品の容器にチップが埋め込まれ、薬品の種類や製造時刻が記録されている状況を想定している。各人は携帯している情報端末を医薬品に近づける事により、薬品の種類や用法、使用期限などを知る事ができる。また、複数の薬品を使用しようとする場合に、薬の飲み合わせをチェックすることにより、薬品による事故を防ぐ事ができる。

以下に本実験システムの動作を説明する。

1) 医薬品の選択

使用者は U-Card 上で本実験システムのソフトを起動後、チェックしたい医薬品に U-Card をかざす。

2) 医薬品の説明表示

U-Card のディスプレイ上にかざした医薬品の説明が表示される（図 5.6.1-1)。

図 5.6.1-1 医薬品の説明画面

この時、かざした医薬品の使用期限が切れていた場合は、その旨も通知される（図 5.6.1-2)。

図 5.6.1-2　期限切れの警告画面

3) 飲み合わせのチェック

引き続き、他の医薬品を U-Card にかざすと、その医薬品の説明

を表示すると同時に、先にチェックした医薬品との飲み合わせ情報が表示される。一緒に服用しても安全な医薬品はその旨が表示される（図 5.6.1-3）。

図 5.6.1-3 飲み合わせが安全な医薬品の表示画面

一緒に服用すると危険な医薬品は警告画面が表示される（図 5.6.1-4）。

図 5.6.1-4 飲み合わせが危険な医薬品の表示画面

(2-3) 電子ブック購入システム

本実験システムは、実際の本に埋め込まれたチップの情報を情報端末で読み取り、その情報を元に情報端末が自動的に電子ブックサーバ

にアクセスし、その本の電子データをダウンロードする（図 5.6.1-5）。ユーザは携帯している情報端末を本にかざすだけで、その本の電子ブックデータをダウンロードし、情報端末上で閲覧することができる。

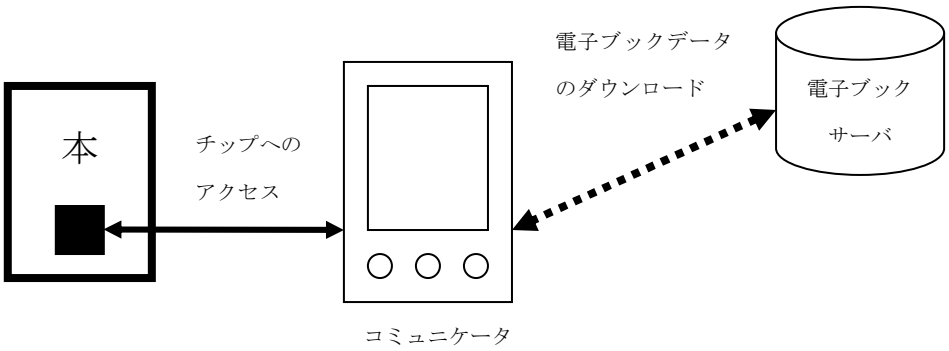


図 5.6.1-5 電子ブック購入システムの概要

以下に本実験システムの動作を説明する。

1）本の選択

使用者は U-Card 上で本実験システムのソフトを起動後、購読したい本に U-Card をかざす。

2）電子ブックデータのダウンロード

U-Card は、本に埋め込まれたチップより読みとった情報に基づき、電子ブックサーバに無線 LAN にてアクセスし、データをダウンロードする（図 5.6.1-6）。

図 5.6.1-6 電子ブックデータのダウンロード画面

3) ダウンロードした電子ブックデータの閲覧

使用者は U-Card にダウンロードした電子ブックデータを自由に閲覧することが可能である（図 5.6.1-7、図 5.6.1-8）。

図 5.6.1-7 ダウンロードした電子ブックデータ（表紙）

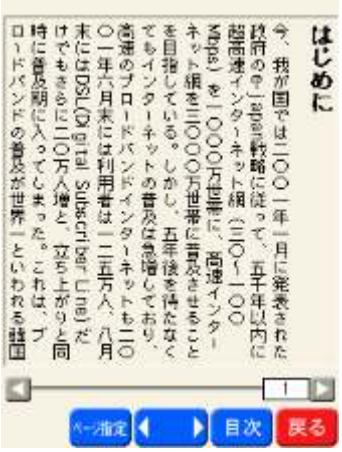


図 5.6.1-8 ダウンロードした電子ブックデータ（本文）

本実験システムの応用として、電子ブックサーバからのデータのダウンロードの際に、電子ブックの料金の決済も行う電子ブック購入システムに発展させることも考えられる。

(3) IC カードと連携したセキュリティシステム

入場認証情報や入室認証情報を持った IC カードと連携させて、セキュリティーゲートの入場開閉、セキュリティードアの開錠施錠を行い、更に入室入場の様子のカメラ画像を転送する、監視システムを開発した。ゲートやドアの制御に U-Card、カメラ画像の撮影と圧縮および転送にμU-Card を応用してシステムを構築した。

システムの動作は次の通りである。入場セキュリティーゲートと入退出ドアに開発した制御機器を取り付け、所定の位置に設置し制御機器に接続した IC カード R/W（リーダライタ）に予め認証情報を書き込んだ IC カードをかざすとゲートの解放を行ったり、ドアの開錠を行うようになっている。更にカードが R/W にかざされた時、カードの ID や機器の状態を表示端末にネットワークを通じて送信し、ゲートやドアの所定の位置に設置したカメラが撮影した画像を表示端末に表示する。IC カードを介して、コンピューティング環境が個人を認識し、人とコミュニケーションを行い、サービスを提供するという、ユビキタスコンピューティングの応用例にもなっている。

次に各部の構成と機能を説明する。

セキュリティーゲート

構成

セキュリティーゲート本体
制御部
IC カード R/W

機能

セキュリティーゲートの通過許可情報を持った IC カードを所定の位置に設置した IC カード R/W にかざすとカードの情報を制御部にて取得しゲートを開く。また、通過許可情報を持たないカードをかざした場合は、ゲートを開かない。写真 5.6.1-9 に全体を示す。

セキュリティードア

構成

電子錠付きドア
電子錠制御部

ICカードR/W

機能

セキュリティードアの通過許可情報を持った IC カードを所定の位置に設置した IC カード R/W にかざすとカードの情報を制御部にて取得し電子錠を開錠する。また、通過許可情報を持たないカードをかざした場合は、電子錠の開錠を行わない。写真 5.6.1-10 にドア付近に設置したR/W を示す。

カメラ

構成

カメラ

カメラ制御部

機能

セキュリティーゲートまたは、セキュリティードアを IC カードをかざして通過する際権利の施行者の画像データをネットワークを介して表示端末に送信し施行者の画像データを表示部に表示する。写真 5.6.1-10 上方にある小さな丸い穴にカメラが埋め込まれている。

写真 5.6.1-9 セキュリティーゲート

写真 5.6.1-10 セキュリティードアカード R/W

(4) IC カードと連携した電子実体流通システム

IC カードを使用した電子実体の発行、販売および権利行使の実証実験を行うためカード発券機、自動販売機およびセキュリティーゲートを開発した。カード発券機の制御にμU-Card を、自動販売機およびセキュリティゲートの制御に U-Card を利用している。

システムの動作は次の通りである。IC カード発券機は個人属性や入金情報を IC カードに書き込んだ上でカードを発券する。そしてカードに格納された金銭情報を使って、自動販売機で飲料製品の購入、チケットの購入および、アルコール製品の購入が行える。また、自動販売機によりチケットの購入を行うとその情報が IC カードに記録され、チケットの権利の行使してゲートを通過することができる。

なお本システムはユビキタスネットワーキング環境における価値情報の流通を行う実証実験に用いられる。

次に各部の構成と機能を説明する。

IC カード発券機

構成

IC カード発券機本体
カード発券機制御部
GUI(グラフィカルユーザインタフェース)部
IC カード R/W

機能

ユーザが入力した個人情報（大人/子供、日本語/英語など)およびユーザが入金した金額情報を IC カードに書き込み発券する。また、ユーザがカードの金額を清算する際、カードの回収および残金情報に相当する金額をユーザに返済する。(写真 5.6.1-11)

自動販売機

構成

自動販売機本体（飲料水製品の販売部）

自動販売機制御部
GUI 部
IC カード R/W

機能

飲料水製品、映画チケット等のチケット販売、アルコール製品の選択を行えそれぞれの販売を IC カードの金銭情報より行い販売相当額をカード内の情報より差し引く。また、チケットを購入の際には、チケットの権利情報をカードに書き込む。(写真 5.6.1-12)

セキュリティーゲート

構成

電子錠付きドア
電子錠制御部
IC カード R/W

機能

自動販売機により購入されたチケット情報が記入された IC カードが R/W 部かざされた時ゲートを開く。

写真 5.6.1-11 カード発券機

写真 5.6.1-12 自動販売機

(5) まとめ

標準開発環境として研究を進めているワンボード型エンベディッドアーキテクチャの有用性を検証するため、ユビキタスコンピューティングに関連する様々な応用システムを試作し、その有用性を評価した。その結果、様々なシステムに適用可能なことが判った。なお今回開発した応用システムは、本研究所の研究に今後活用される。

5.6.2 ミドルウエアの研究

(ア)ミドルウエア組み込み方式に関する研究

(1) はじめに

あらゆるところにチップが組み込まれて動作するユビキタスコンピューティング社会を実現するためには、開発するソフトウエアも膨大な量となる。そのため、一度開発したソフトウエアを流用し、新しくシステムを作る場合でもすぐ再利用できる機構が必要となる。

本サブテーマである超機能分散システム指向開発環境の整備では、ハードウエアアーキテクチャとともにリアルタイムカーネルの研究開発を行っているが、そのリアルタイムカーネルの機能としてミドルウエアを流用する機構を研究し開発した。これにより、今後の研究や実製品開発時に、様々なシステムに対してミドルウエアを流用することが可能となった。

(2) ミドルウエア再利用のための機構

ミドルウエア再利用に関連して、今回開発したリアルタイムカーネルが備える主な機構は以下である。

[1] オブジェクト ID の自動割り当て機能
[2] メモリ管理機能
[3] サブシステム管理機能
[4] デバイス管理機能

以下、これらについて説明する。

オブジェクト ID の自動割り当て機能

リアルタイムカーネルは様々なオブジェクト操作機能を提供する。その機能を利用して、マルチタスク・リアルタイムアプリケーション(ミドルウエアも含む)が開発される。オブジェクトとして、タスク、セマフォ、イベントフラグなどがあり、システムにはそれぞれ複数個存在することになる。各ミドルウエアやアプリケーションは、リアルタイムカーネルが提供する操作関数を呼んで、必要なオブジェクトを生成し利用する。

オブジェクトは ID 番号が振られ、その番号により操作オブジェクトを指定する。そしてオブジェクトを生成する際、従来の多くのリアルタイムカーネルでは、予めシステム全体で定めた一意の ID 番号を指定するようになっていた。この場合、システムのソフトウエア構成が変わる毎に各ミドルウエアで使うオブジェクトの ID 番号が変わることになり、ミドルウエアを変更したり再コンパイルして使用する

必要があった。

今回開発したリアルタイムカーネルでは、それを解決するため、オブジェクト生成時に ID 番号を自動的に割り当てることにした。例えばタスクを生成する操作関数は次のようになる。

```
ID tskid = tk_cre_tsk( T_CTSK *pk_ctsk );
```

このようにタスク生成操作関数が、ID 番号を自動的に割り当ててリターン値として返すようになっている。各ミドルウエアは変数にこの ID 番号を保存して利用すれば良い。ちなみに T_CTSK 構造体はタスク属性などを指定するパラメータである。

メモリ管理機能

今回開発したリアルタイムカーネルは CPU の MMU(Memory Management Unit)機能に対応している。具体的には、タスク固有空間の概念、メモリのアクセス保護機能、論理アドレスから物理アドレスの取得機能、プログラムの常駐/非常駐管理機能などである。

これらの機能により、アプリケーションタスクの暴走があってもシステムに組み込まれたミドルウエアに影響しない強固な開発環境を実現できる。またミドルウエアをダイナミックにロードして実行したり、更には仮想記憶を構築して実メモリ容量より大きな仮想メモリ空間を実現してシステムコストを削減すること等が、可能となっている。

サブシステム管理機能

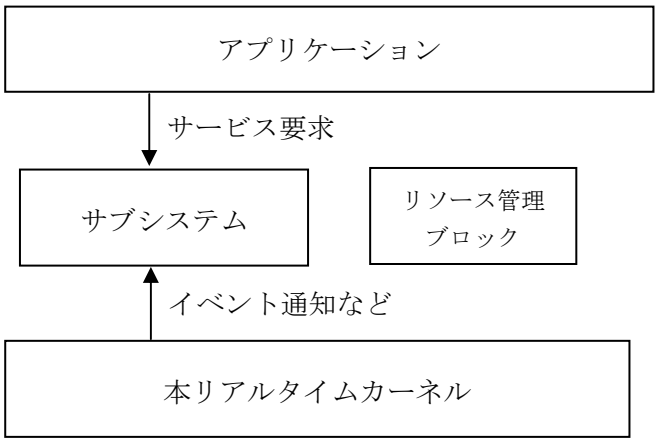


図 5.6.1 サブシステムの位置づけ

今回開発したリアルタイムカーネルでは、ミドルウエアをサブシステムとして組み込む機構をサポートしている。図 5.6.1 にサブシステムの位置づけを示す。(図にはサブシステムは一つしか記載していないが、通常複数のサブシステムが同時に動作している)

サブシステムはリアルタイムカーネル上で動作し、アプリケーションにサービスを提供する。また例外やイベントが発生した時にリアルタイムカーネルからサブシステムの機能が呼び出される。本リアルタイムカーネルでは、これらの機能の定義方法および呼び出し方法を規定している。これにより、様々なサブシステムを様々なシステムで使うことになっても、いつも統一された方法で組み込むことができる。また、例外やイベント発生時の処理呼び出し手順も決まるので、例えばシステム終了時の処理を、新しくシステムを開発する毎に調べなおして作成する等の手間が不要になる。以下に、例として、サブシステムの定義関数(tk_def_ssy 関数)を示す。

関数インタフェース

```
ercd = tk_def_ssy( ID ssid, T_DSSY *pk_dssy);
```

機能説明

ssid の ID 番号を持つサブシステムを定義する。サブシステムの内容は T_DSSY 構造体で定義される。

T_DSSY 構造体は次のメンバから成る。

ssyatr	サブシステム属性
ssypri	サブシステム優先度
svchdr	サービスコール関数
breakfn	タスク例外発生処理関数
startupfn	サブシステム起動処理関数
cleanupfn	クリーンアップ処理関数
eventfn	イベント処理関数
resblksz	リソース管理ブロックサイズ

リソース管理ブロックは、各々のサブシステムについて準備されるメモリ領域で、一つのサブシステムにつき一つあるいは複数個の管理ブロックを作成することができる。各ミドルウエアはリソース管理ブロックを使用して提供サービスの管理を行う。

通常、リソース管理ブロックは、一つのサブシステムにつきタスク固有空間毎に生成する。こうすれば、各タスク固有空間毎にサービス

提供状態を管理ブロックに格納でき、異なる空間へのサービスで干渉を防止したり、タスク固有空間削除時の終了処理を容易化することが可能になる。

デバイス管理機能

今回開発したリアルタイムカーネルでは、デバイスドライバを管理する機構をサポートしている。デバイスドライバとは、各種のデバイスを制御するソフトウエアであり、上位アプリケーションに対して論理的あるいは抽象的な制御関数インタフェースを提供するものである。

図 5.6.2 に本リアルタイムカーネルにおけるデバイス管理機能の位置付けを示す。

本リアルタイムカーネルでは、アプリケーションインタフェースとデバイスドライバインタフェースを標準化している。これにより内部仕様が理解しやすくなるとともに、各ドライバに必要な関数が定義されているため、システム全体におけるイベント処理やアボート処理の実現が容易になる。

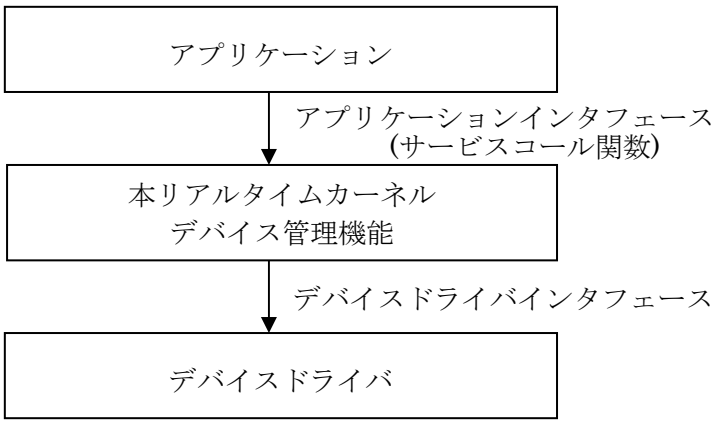


図 5.6.2 デバイス管理機能

デバイスを利用するときは事前に登録を行う。登録は tk_def_dev 関数を用いる。

tk_def_dev(デバイス名、T_DDEV 登録情報、初期情報);

T_DDEV 構造体は次のメンバから成る。この中の各関数は図 5.6.2

のデバイスドライバインタフェース関数に対応する。

drvatr;	ドライバ属性
devatr;	デバイス属性
blksz;	固有データのブロックサイズ
openfn;	オープン関数
closefn;	クローズ関数
execfn;	処理開始関数
waitfn;	完了待ち関数
abortfn;	中止処理関数
eventfn;	イベント関数

主なアプリケーションインタフェース関数を下記に示す。これらはサービスコール関数としてアプリケーションから呼ばれる。リアルタイムカーネルは一度パラメータ内容等を確認した後、デバイス登録時に指定された処理関数を実行する。

tk_opn_dev(デバイス名,　オープンモード);
　文字列"デバイス名"で指定されるデバイスをオープンする。
tk_cls_dev(デバイスディスクリプタ, クローズオプション);
　デバイスをクローズする。
tk_rea_dev(デバイスディスクリプタ, ...);
　デバイスから読み込み開始する。
tk_sus_dev(モード);
　モードに従った一時停止処理を行う。
tk_evt_dev(イベント送信先デバイス ID、...);
　デバイスドライバに要求イベントを送信する。

主なデバイスドライバインタフェース関数を以下に示す。これらはアプリケーションからの処理要求を受けた本リアルタイムカーネルが、登録されたデバイス情報に基づいて呼び出す関数となる。

openfn(デバイス ID、モード、拡張情報);
closefn(デバイス ID、オプション、拡張情報);
execfn(処理要求内容、タイムアウト時間、拡張情報);
waitfn(要求リスト、要求個数、タイムアウト時間、...);
abortfn(実行タスク ID、要求リスト、要求個数、...);
eventfn(要求イベントタイプ、イベント情報、...);

(3) まとめ

今回、ミドルウエアの流用とシステムにおける使い勝手を向上させるために、ミドルウエア組み込み方式の研究と、それを実現するリアルタイムカーネルの開発を行った。この成果をベースに、今後様々なミドルウエアやデバイスドライバを開発し、研究資産としていく。

（イ）ミドルウエア記述形式に関する研究

(1) はじめに

ミドルウェアの再利用性を高めることが、ソフトウェアの生産性を向上させるために重要である。ミドルウェアは、同一の CPU であればオブジェクト形式で再利用が可能、また、異なった CPU でも再コンパイルのみで再利用できることが望ましい。これを実現するため、ミドルウェアの記述形式に関して研究を行い、これを規定した。

(2) ミドルウェアモデル

超機能分散システム指向開発環境におけるミドルウェアのアーキテクチャモデルとそれに基づく分類に規定する。

●実現方式による分類

ミドルウェアは実現方式により以下の 4 種類に分類される。

[1] サブシステム型ミドルウェア
リアルタイムカーネルが提供するサブシステム管理機能により実現されるミドルウェア。

[2] デバイスドライバ型ミドルウェア
リアルタイムカーネルが提供するデバイスドライバ管理機能により実現されるミドルウェア。

[3] ライブラリ型ミドルウェア
ユーザライブラリとして実現されるミドルウェア。ユーザアプリケーションのコード生成時にリンクされる。

[4] ユーザタスク型ミドルウェア
ユーザタスクによるサーバにより実現されるミドルウェア。

●提供サービスによる分類

ミドルウェアは提供サービスにより以下の 4 種類に分類される。

[1] デバイスドライバ型ミドルウェア
単一のデバイスに対して、一定の抽象度を持った、ソフトウェアインタフェースを提供するミドルウェア。
（例）RS232C ドライバ、PCMCIA ドライバ、Ethernet ドライバ

[2] 抽象デバイス型ミドルウェア
単一または複数のデバイスに対して、高い抽象度を持った、ソフト

ウェアインタフェースを提供するミドルウェア。
（例）TCP/IP、FAT ファイルシステム、GUI パッケージ

[3] カーネル拡張型ミドルウェア
カーネルが管理するハードウェア資源やソフトウェア資源に対して、カーネルの機能やミドルウェアの機能を利用して、より高い抽象度のサービスを提供するためのミドルウェア。
（例）プロセス管理機能、仮想記憶機能、ローダ機能

[4] 一般機能提供型ミドルウェア
特にカーネルが管理する資源や外部デバイスと関連性が薄い、一般的な計算機能を提供するミドルウェア。
（例）ソート機能、サーチ機能、暗号/認証機能

(3) グローバルシンボル

超機能分散システム指向開発環境において、複数のミドルウェアを相互に組み合わせて利用することが可能であることを保証するために、グローバルシンボルについて規定を定める。主に、異なるベンダーが提供するミドルウェアを組み合わせて、コンパイル・リンクの際の不具合を防ぐ事を目的とする。

[1] ベンダーコード
それぞれのミドルウェアベンダーは、固有のベンダーコードが割り当てられる。異なるベンダーが同じグローバルシンボルを利用する事のないように、グローバルシンボルにこのベンダーコードを利用する。

（例） ベンダー名：YRP ユビキタスネットワーク研究所 ベンダーコード：unl

[2] ミドルウェアファイル名
ミドルウェアを構成するファイル(ライブラリなどのバイナリファイルとヘッダファイル）には、超機能分散システム指向開発環境において唯一性が保証される命名規則を以下の通り規定する。

〈ファイル名〉　　::= 〈ミドルウェア名〉“.”〈拡張子〉

〈ミドルウェア名〉　::=〈ベンダーコード〉“_”〈機能名〉[“_”〈詳細機能名〉]

● 機能名

機能名は、そのミドルウェアのファイルが実現する機能を表現する名前とする。各ベンダー内で、名前が衝突しないようにしなくてはならない。

機能名は、”a”～”z”、”0”～”9”の 2 文字以上の文字列とする。

（例） 機能：　MPEG2 のコーディック 機能名：　mpeg2

● 詳細機能名

一つのミドルウェアが、複数のバイナリファイルやヘッダファイルから構成される場合に、それらを区別するための詳細な名称。

詳細機能名は、機能名と同様”a”～”z”、”0”～”9”の 2 文字以上の文字列とする。

以上の規定にのっとり、例えばベンダーコード「unl」を持つベンダーが作成した MPEG2 コーディックのミドルウェアのファイル名は以下のようになる。

unl_mpeg2.a unl_mpeg2_basic.h

[3] export される関数名

あるミドルウェアのライブラリファイルに含まれており、アプリケーションに Export される関数名の命名規則を次のように定める。

〈関数名〉　::=〈ミドルウェア名〉”_”〈関数識別子〉

関数識別子は、”A”～”Z”、”a”～”z”、”0”～”9”の 2 文字以上の文字列とする。

（例） MPEG2 エンコード関数：　void　unl_mpeg2_encode(…);

MPEG2 デコード関数：　　void　unl_mpeg2_decode(…);

[4] export される大域変数名

あるミドルウェアのライブラリファイルに含まれており、アプリケーションに Export される大域変数名の命名規則を次のように定める。

〈変数名〉　　::= 〈ミドルウェア名〉”_”〈変数識別子〉

変数識別子は、”a”～”z”、”0”～”9”の 2 文字以上の文字列とする。

[5] 関数、大域変数で用いられるデータ型名

export される関数の引数や戻り値で用いられるデータ構造や、大域変数で使われるデータ構造の命名規則を次のように定める。

〈データ型名〉 ::= 〈ミドルウェア名〉”_”〈データ型識別子〉

データ型識別子は、”a”～”z”、”0”～”9”の 2 文字以上の文字列とする。

[6] ヘッダファイルに含まれる定数マクロ名

export される関数の引数や戻り値で用いられる Special Value や、大域変数で使われる Spacial Value の命名規則を次のように定める。

〈マクロ名〉　::=〈ミドルウェア名（大文字）〉”_”〈マクロ識別子〉

マクロ識別子は、”A”～”Z”、”0”～”9”の 2 文字以上の文字列とする。

(4) ミドルウェアパッケージ名

超機能分散システム指向開発環境におけるミドルウェアは、ミドルウェアパッケージ名を付与する事ができる。ミドルウェアパッケージ名は、そのミドルウェアの性質や特徴を表す名前である。主に、ミドルウェアを扱うソフトウェア開発環境におけるパッケージの検索処理などで用いる事を想定している。ミドルウェアパッケージ名の命名規則を次のように定める

```
〈パッケージ名〉　　::= 〈ベンダーコード〉 “_”〈機能名〉
　　[ “_”〈詳細機能名〉] “_” 〈バージョン表現〉
```

ここでいう〈機能名〉、〈詳細機能名〉は、ミドルウェアファイル名でつけたものと同じものとする。
バージョン表現については次のように定める。

```
xyyzz
```

x、yy、zz それぞれの形式と意味を以下のように定める。

x: 1～9　メジャーバージョン番号
ミドルウェアにバックワードコンパチビリティがなくなる変化をもたらすバージョン変更をした場合、インクリメントされる。

yy: a0～z0、00～99　マイナーバージョン番号
ミドルウェアの API に変更があるが、バックワードコンパチビリティが確保されている場合は、メジャーバージョン番号はそのままに、このマイナーバージョン番号がインクリメントされる。
a0～z0 はプレリリースに用いられ、00～99 は正式リリースに用いる。

zz: 00～99　マイナーリリース番号
ミドルウェアの API や、外部から見た論理的な動作にコンパチビリティを保ったバージョン変更の場合は、メジャーバージョン番号、マイナーバージョン番号はそのままに、このマイナーリリース番号がインクリメントされる。たとえば、バグフィックスや、より効率の良いアルゴリズムの導入などがこれに相当する。

なお、最初にリリースするミドルウェアのバージョン番号は、1.00.00 とすることを基本とする。ただし、既存のミドルウェアのポーティングなどの場合には、必ずしも最初のバージョンが 1.00.00 でなくてもよい。

（ウ）ミドルウエアの開発

ユビキタスコンピューティング環境の研究を進めるためには、あるいは実現するためには、基盤となるハードウエアおよび基本ソフトウエアの他に、セキュリティやネットワークプロトコルを実行処理するミドルウエアが必要になる。今年度は既にいくつものミドルウエアを開発した。下記に主なものを紹介する。

(1) SNMP

SNMP とは、Simple Network Management Protocol の略であり、IP 網における機器管理/ネットワーク管理用のプロトコルである。SNMP は 1990 年に IAB(Internet Architecture Board)にて標準化され、現在でもネットワーク管理領域におけるデファクト的なプロトコルである。

SNMP はそのアーキテクチャとして、『マネージャ』『エージェント』『MIB』『SNMP プロトコル』の 4 つのコンポーネントから構成される。『マネージャ』とはサーバなどに存在し機器やネットワークを管理するアプリケーションプログラムである。『エージェント』とは実際に管理する機器に実装され、マネージャからの指示に従い管理情報の送信や機器の制御を行う。『MIB』とは管理する機器に実装される管理項目のデータベースである。エージェントは取得要求に該当するデータを MIB から取得してマネージャへ送信する。『SNMP プロトコル』はマネージャ～エージェント間の通信に利用される通信プロトコルである。俗にいう SNMP 対応機器というのは、『エージェント』及び『MIB』が実装されている機器を指す。また、SNMP は機器の情報取得、機器の遠隔制御という 2 つの側面を持つ。

我々が実現を目指しているユビキタス社会では、身の回りのものやこれまでスタンドアロンで機能していた機器など大小様々なものがネットワークに接続される。その数はこれまでのインターネットにおけるルータや PC などの数をはるかに凌駕する。そのような分散コンピューティング環境では、如何にしてそれらを効率的に管理/制御

するかが重要な課題である。SNMP はその解決策の一つと考えられ、その研究を進めるため、今年度開発を実施し完了した。

(2)セキュア IC チップ用クライアントライブラリ

本ミドルウェアは、セキュア IC チップを利用するアプリケーションに対して、そのチップと確実かつ安全にコマンド送受信を行なうための機能を提供するものであり、以下の機能をもっている。

- セッション／トランザクション構築コマンドの送受信
- ファイルの生成／削除／転送／暗号化／復号などのコマンドの送
- 受信
- レコードの更新／読出しコマンドの送受信
- 課金情報を伴う暗号化／復号用の共通鍵の生成／削除／更新／読出しなどのコマンドの送受信
- 公開鍵証明書の更新／読出しコマンドの送受信
- 公開鍵の更新／読出し、および鍵ペアの生成／更新コマンドの送受信
- チップ情報の読出し（ポーリング）コマンドの送受信
- チップ情報、初期公開鍵証明書の書込みなどのコマンドの送受信
- VPN 用公開鍵証明書の更新／読出しコマンドの送受信
- VPN 用公開鍵の更新／読出し、および VPN 用鍵ペアの生成／更新コマンドの送受信
- チップ内部のパスワード／DES 鍵などの更新コマンドの送受信

(3) 暗号・認証ライブラリ

本ライブラリは、暗号・認証に関連した処理を行なうための機能を提供し、上位アプリケーションに代わりそれらの処理をおこなうものである。なお、前述のセキュア IC チップ用クライアントライブラリでも本ライブラリを利用している。本ライブラリは、主に以下の機能を提供する。

- 複数のハッシュアルゴリズム（MD5、SHA-1）によるハッシュ値の計算
- 公開鍵暗号（RSA）による暗号化／復号演算
- 複数の共通鍵暗号アルゴリズム（Camellia、Hierocrypt、Rijndael、DES、Triple DES）による暗号化／復号演算

(4) 非接触型セキュア IC カード・アクセスライブラリ

本ミドルウェアは、非接触型セキュア IC カード（以下、IC カード）を利用するアプリケーションに対して、IC カードをアクセスするための各種機能を提供する。

本ライブラリは、IC カードリーダライタ操作部、IC カード操作部、通信インタフェース部の三つに大別される。それぞれはモジュール化されて別々に追加・交換が可能である。

IC カードリーダライタ操作部の主な機能は以下の通りである。

- リーダライタの各種設定
- IC カードパケットの送受信

IC カードリーダライタは、機種により制御方法が異なる。使用する IC カードリーダライタに応じて、IC カードリーダライタ操作部を選択することにより、これに対応することができる。

IC カード制御部の主な機能は以下の通りである。

- IC カードのポーリング
- セッションの構築
- ファイルの生成、削除
- レコードの更新・読み出し

IC カードの機能が変更された場合は IC カード操作部を変更することにより、これに対応することができる。

通信インタフェース部は、ハードウェアに依存するリーダライタとの通信制御の標準化を行う。ハードウェアが異なる場合、このインタフェース部を変更する事により対応する。これにより、パソコンから組込み機器まで、ハードウェアの異なるプラットフォームにおいて、本ライブラリを容易に使用する事ができる。

添付資料
研究発表一覧

学会発表および論文発表

（査読無し）

43. 太田陽基，中尾康二，田中俊昭，西山智，越塚登，坂村健：「非接触 IC カードにおける X.509 証明書のコンパクト化に関する一考察」，電子情報通信学会　平成 15 年総合大会，A-7-29.
44. 西山智，渡辺伸吾，服部元，小野智弘，越塚登，坂村健：「ユビキタスサービスのための屋内センサーネットワークの提案」，情報処理学会第 65 回全国大会，3H-2，pp. 3-247～248.
45. 渡辺伸吾，西山智，服部元，小野智弘，越塚登，坂村健：「ユビキタス環境のための非接触 IC カードを使用した位置検出方式の提案」，情報処理学会第 65 回全国大会，3H-1，pp. 3-245～246.

（査読有り）

46. Katsunori Shindo, Noboru Koshizuka, and Ken Sakamura: "Ubiquitous Information System for Digital Museum using Smart Cards", in Proceedings of the SSGRR, Jan., 2003.
47. Katsunori Shindo, Noboru Koshizuka, and Ken Sakamura: "Large-scale Ubiquitous Information System for Digital Museum", in Proceedings of the 21st IASTED, Feb., 2003.

標準化活動への貢献および報道発表等

標準化活動への貢献

ユビキタスネットワーキングフォーラム（H14 年 6 月～H15 年 3 月）

- 技術部会
  - ネットワークアーキテクチャ専門委員会
  - どこでもネットワーク専門委員会
  - 何でもマイ端末専門委員会
  - 超小型チップネットワーキング専門委員会
- 企画部会

報道発表等

テレビ・ラジオ

- ＮＨＫ 他　５件

新聞・雑誌

- 日経新聞 他　１７件