修　士　論　文

CCDM: 中央管理方式に基づくセンサおよびアクチュエータの管理アーキテクチャと管理スクリプト分割手法の提案

CCDM: Central Controller-based Device Management Architecture and Method to Split Management Scripts

指導教員　　江崎 浩 教授

東京大学大学院 情報理工学系研究科
電子情報学専攻

氏　名　　48-076418　**杉山 哲弘**

提　出　日　　2009年2月4日

梗概

近年, キャンパスネットワークやビル管理ネットワークにおいて, 周辺の環境情報を取得するセンサ機器や周辺の空間に影響を与えるアクチュエータがインターネット接続性を持つような状況が現れている. 増大するセンサおよびアクチュエータの統合的な管理基盤を提供することは重要かつ挑戦的な技術課題である. そこで, 本研究では, ネットワーク管理者にとってはこのような多様なデジタルデバイスを扱いやすく変更の手間が少なく, またアプリケーションの実行に当たっては耐障害性に優れた管理基盤を提供するシステムである CCDM を提案する. CCDM は中央管理方式によるセンサおよびアクチュエータの管理と管理スクリプトの分割による分散実行の環境を提供する. CCDM の中央管理アーキテクチャは分散設置される多数のデジタルデバイス間での協調動作に必要な管理情報や制御プログラム間での一貫性を保証し, 管理コストを低減することを大きな特徴としている. CCDM の管理インターフェースはサービススクリプト, リソーススクリプトと呼ぶ 2 種類の管理スクリプトから構成されている. サービススクリプトはセンサおよびアクチュエータの操作に必要十分な命令セットを提供する. また, リソーススクリプトはデジタルデバイスの機能セットや資源の利用状況あるいは相互接続の状況などの情報を収集することにより, 提案システムが通信プロトコルの差異やネットワークの状態変化に柔軟に対応するためのデータベース機能を提供する. サービススクリプトとリソーススクリプトの重ね合わせによって生成される実行スクリプトはネットワーク上の各ノードに分散配置され実行される. 本研究では CCDM のプロトタイプの実装と検証を行い, システムの管理コストが減少すること, および耐障害性が向上することを確認した.

目次

図目次

表目次

第 1 章

序論

1.1 はじめに

1969 年にアメリカ国防総省によって提案された Advanced Research Projects Agency Network(ARPANET) が構築されて以来, インターネットは力強く発展, 普及し, ビジネスや教育, 人々の社会・産業活動を世界規模で大きく変革してきた [1]. インターネットのユーザ数は現在も増大しており, その要請に応えるため通信インフラの高速, 大容量化が進行している [2]. また, インターネットは, 当初のコンピュータのみではなく, 各種センサ, 事務用製品, 家電といった多様なデジタル機器が接続されるようになった. インターネットは社会・産業基盤として進化・発展し, これからも社会の構造変革に大きな貢献と寄与を続けていくと考えられる.

インターネットは複数のネットワークを相互に接続した大規模分散ネットワークである [3]. インターネットは論理的なネットワークであり, あらゆる機器はインターネットプロトコル (IP)[4] に従うことにより, インターネット上のすべてのノードとの通信が可能となる. 半導体を初めとした通信機器に必要な部品の高密度化, 低価格化 [5] により, 様々な機器が IP をサポートできるようになり, インターネットに接続するようになった. 例えば,Live E!プロジェクトは環境センサをインターネットに接続し, 生きた環境情報を自由に流通し共有する電子情報基盤の構築を目指している [6]. グリーン東大工学部プロジェクトは建物内に複数のベンダにまたがる空調システム, センサ機器, 通信機器が存在する状況において, それらのシステムを統合し相互に協調動作することで省エネルギーを達成するシステムの検証を行っている [7].

多種多様な機器がインターネットに接続している現在, またその数が爆発的に増えて行くであろう将来においても, 適切に継続動作可能な管理アーキテクチャが確立されなければならない. ネットワークの管理は, 煩雑で, 設定ミスを起こしやすく, 労働集約的な領域であるために高価なものとなってしまう [8, 9]. 特に, ルータやスイッチ, サーバなどの統合的な管理手法については数多くの研究がなされてきたが, キャンパスネットワークや企業ネットワークなどの現在のネットワークは規模が大きく複雑で, 利用されるアプリケーション, プロトコルは多岐に渡ることが指摘されてきた [10]. このような複雑なネットワークの管理を簡素化・自動化する試みは挑戦的である [11]. 本研究では, 特に, 周辺の空間の情報を取得するような機器 (センサ) やその情報に従って周辺に影響を及ぼす機器 (アクチュエータ)(以下, これらをデバイスと総称する) がインターネットに接続している状況に焦点を当て, これらの管理の問題に

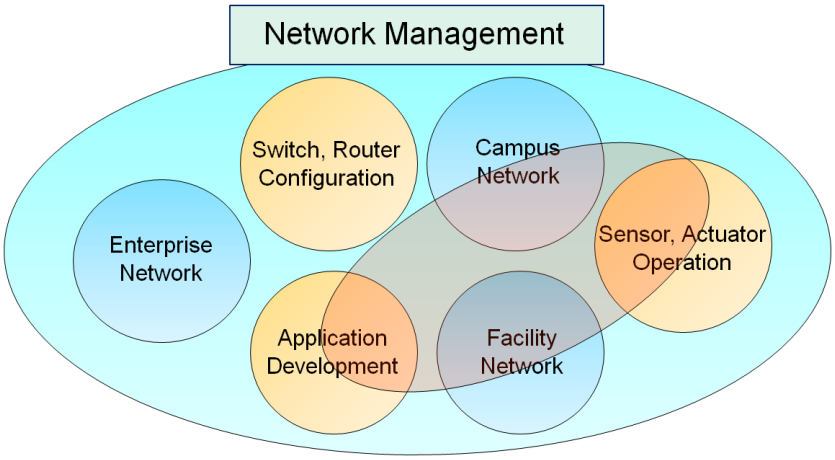

図 1.1　ネットワーク管理の対象と空間

挑戦する.

1.2　本研究で対象とするネットワーク環境

ネットワーク上で提供されるサービスはウェブ, メール, ファイル転送など様々であり, 管理の対象となる機器もサーバ, ルータ, スイッチ, アクセスポイント, センサなど多岐に渡る. インターネットに接続し管理が必要となる対象, 空間を図 1.1 のように分類した.

- ネットワーク管理の空間
  1. 企業ネットワーク
  2. キャンパスネットワーク
  3. その他のファシリティネットワーク
- ネットワーク管理, アプリケーションの対象
  1. ルータ, スイッチ, アクセスポイントなどのネットワーク到達性のために必要な機器
  2. ウェブやメールなどのアプリケーションの管理 (ユーザ管理や設定など)
  3. センサおよびアクチュエータなどの周辺の環境と相互に影響を及ぼし合う機器

本研究では, この分類のうち, 管理の空間としてキャンパスネットワークやその他のファシリティネットワークに焦点を当て, 管理の機器としてセンサやアクチュエータ, それらが利用するアプリケーションを対象とする.

1.3　本研究の特徴と貢献

センサによる周辺情報の取得を目的とした研究の分類として無線センサネットワークが存在する [12]. 無線センサネットワークでは, 周辺の情報を取得するセンサ同士がその場限りの無線ネットワークを構築する状況を想定しており, 本研究が想定している状況と電源環境やネットワークの安定性において異

なる (第 2.2.2 節). 一方, ファシリティネットワークやビルディングオートメーションを対象としている技術として Building Automation and Control Networking protocol(BACnet) や Lonworks などが存在する [13]. しかし,BACnet や Lonworks はベンダによって異なるインターフェースを統一した Application Program Interface(API) を提供する支配的な標準であるが, 管理アーキテクチャについての提案は目標としていない (第 2.1 節).

本研究では, キャンパスネットワークやファシリティネットワークにおけるセンサおよびアクチュエータの管理の問題を解決することを目指している. その手法として近年のネットワーク管理の動向に存在する中央管理方式 [10, 11, 14, 15] をアーキテクチャとして採択している. 中央集中型による解決策はインターネットの分散的で非権威的なアーキテクチャと真逆であるように見える. IP のベストエフォートのモデルはシンプルで不変であり, 分散アルゴリズムに適している. しかし一方で, ネットワークの管理はその対極にあると考えられる. ネットワーク管理に対する要求は複雑であり, かつ管理データの一貫性を必要とする. そのため, ネットワーク管理の領域において, 分散的な方法を適用することは, 不可能ではないがより大きな技術課題を解決しなければならないと考えられる. そこで本研究では, ネットワークの管理アーキテクチャとして中央管理方式による手法を採用することにした.

ネットワーク管理においては提供するインターフェースや命令セットの種類や粒度も重要な要素となる. 提案システムではネットワーク汎用 I/O 制御を目的として開発された高級言語 [16] に基づいたサービススクリプトとリソーススクリプトという 2 種類のスクリプト言語によって, デバイスの管理を規定する. サービススクリプトは四則演算, 条件判断, 分岐などのデバイスの操作に必要十分な命令セットを提供する. また,post/catch, INPUT/OUTPUT, chunk などの特徴的な命令を用いることによって, デバイス毎に異なる通信プロトコルの差異を隠蔽し, 適切なホストでサービスを提供することが可能となる. 管理者は下位レイヤを考慮することなくデバイスが実現するサービスを記述することができる. リソーススクリプトはネットワークの情報やデバイス毎の通信プロトコルが記述されている. サービスを記述したスクリプトとリソースを記述したスクリプトが分離されることにより, ネットワークの状態変更時やサービスの追加時の変更が容易なシステムを提供することが可能となる. サービススクリプトとリソーススクリプトを重ね合わせ, 各ホストに適した実行スクリプトを生成するアルゴリズムは本研究の重要な提案の 1 つである.

以下に, 本研究の用いる手法の特徴と既存研究に対する貢献をまとめる.

1. 中央コントローラによる管理アーキテクチャのデバイス管理への適用
2. デバイスの管理に適した命令セットの定義
3. (2) の命令セットを複数の実行環境で実行するための命令セット分割アルゴリズムの提案

1.4　本論文の構成

第 2 章では本研究に関係する技術や研究についての紹介を行う. 主に議論するのは, 本研究でリファレンスアーキテクチャとして参考にした中央管理方式によるネットワーク管理方式, ファシリティネットワークにおける技術動向, 近年の無線センサネットワークの管理と構築に関する研究動向である.

本研究の提案について述べているのが第 3 章と第 4 章である. 第 3 章では提案システムに対する要件

と提案するアーキテクチャの概要を述べる. 第 4 章では第 3 章で述べた提案アーキテクチャを実現するための具体的な方策と実装について述べる.

第 5 章と第 6 章では提案システムのプロトタイプ実装の検証と評価について述べる. 第 5 章では本研究における提案システムの検証と評価について述べる. 本研究では提案システムの検証のために 7 つのアプリケーションを実装し, 正常動作することを確認した. 第 6 章では, 提案システムで重要と考えるネットワーク管理コスト, システムの耐ネットワーク障害性, システムの導入の障害についての検証結果と評価を述べる.

第 7 章では本研究の結論と意義, 得られた結果について改めて述べる.

第 2 章

関連技術・研究

ネットワークの管理は煩雑で, 間違いを起こしやすく, コストがかかるものである. [8] は 500 以上のコンポーネントの障害と 3 のインターネットの大規模障害から,

1. サービスの障害の 1 番の原因の 3 分の 2 がオペレータの誤りによるものであること.
2. ネットワークの復旧に要する時間の最も大きな要因の 3 分の 2 がオペレータの誤りによるものであること.
3. オペレータの誤りに分類されるもののうち最も大きな要素が設定の間違いであること.
4. システムに特化して書かれたフロントエンドのソフトウェアの障害が特筆すべきものであること.
5. より拡張可能なオンラインでの検証や, 十分に公開され, 検証されたコンポーネントにより, 障害の発生率を減らすことができること.

を検証し, 報告している. ネットワーク管理に関しては近年でも多くの研究がなされ, 負担を軽減するための試みがなされている. [17] はボトムアップのアプローチにより, 複雑なルータ, スイッチ類の設定を自動的に生成する手法を提案している. [18] は新しいネットワークの設定を検証できるよう, 実際に動いているネットワークの裏で, 新しい設定を検証できるような仕組みを提案している. 第 2.3 節で紹介する研究では中央管理方式を用いたネットワーク機器の管理を行っている.

これらの研究は主にルータやスイッチのようなインターネットに接続するための機器の管理を対象としているが, 本研究ではセンサおよびアクチュエータの管理を対象としている. 機器管理のアーキテクチャや要求事項, 枠組みについてはこれらの研究を参考にできる部分は多い. 一方で, 管理者に提供すべき命令セットやインターフェースについては異なる提案が必要であろう. 本章ではまず第 2.1 節で本研究が対象とするビル管理ネットワークやファシリティネットワークにおける課題とそれを解決する技術について述べる. 次に第 2.2 節で既存のネットワーク管理システムにおいて用いられているデータ表現について考察する. 第 2.3 節では近年の研究動向して存在する中央管理方式によるネットワーク管理アーキテクチャを用いる論文を紹介する.

2.1 ファシリティネットワークに関する技術

2.1.1 BACnet

ファシリティネットワークでは, 様々なビルディングオートメーションシステム, 制御システム, 製品が通常異なるベンダにより提供されているが, それらを統合して使用することが要求される. ビルディングに存在する機器の情報を交換し利用することが, ビル管理システムの運用にとって重要である. プロプラエタリな通信方式は, システムインテグレータのシステム統合やファシリティネットワークの開発に多大な困難をもたらしてきた. BACnet は,Building Automation and Control Networking protocol の略語で, 空調、照明、アクセス制御、火気検出などの制御をベンダの仕様に依らない共通なインターフェースを提供することを目的としたプロトコルである. BACnet は, 最初の Standard Project Committee (SPC) のミーティングがアメリカ暖房冷凍空調学会 (American Society of Heating, Refrigerating and Air-Conditioning Engineers,ASHRAE)[19] の年次会合において開催された 1987 年以来, 活発な開発が行われている.

BACnet は, まず,hardware binary input and output value, hardware analog input and output values, software binary and analog values, schedule information, alarm and event information, files, control logic と定義されるデータを転送する方法を提供する. BACnet は, 内部の設定, データ構造, 制御方法を定義するのでなく, ネットワーク上で見えるべき情報が標準化されたオブジェクトにより個別の実装から抽象化する. 標準化されたオブジェクトと下位のデータ・処理の間のマッピングはベンダ次第となる.

また,BACnet は,Open Systems Interconnection(OSI) の基本参照モデルに基づいた階層化されたプロトコルアーキテクチャを持つ [20, 21, 22].

2.1.2 LONWORKS

LONWORKS のプラットフォームは, 高機能な機器間で, ピアツーピア, あるいはマスタスレーブの通信方式を可能とする分散的な制御システムである.LONWORKS の技術は全システムのアドレス要求をサポートする一方論理的なセグメンテーションを可能とするアーキテクチャである. セグメンテーションはノードのアプリケーションに透過なネットワークレイヤレベルのルータにより達成され, ネットワークに接続したあらゆるノードから導入, 診断, 監視が可能な方法を提供する. オープンで, 標準化された LonTalk プロトコルはネットワーク越しに機器がメッセージの送受信ができる通信サービスを取り決めている. オプションとして, メッセージの確認応答, 認証, 処理時間に応じた優先配信を提供する. LonTalk プロトコルは, 以下のような特徴を持つ.

- Layer1 物理層: ネットワーク制御アプリケーションは, 異なる導入シナリオに適応しシステムの拡張を容易にするため, 複数の物理媒体のサポートを必要とする.LONWORKS のプラットフォームは, 幅広い種類の物理媒体をサポートしている.
- Layer2 リンク層：物理媒体を使用する方法は, 制御ネットワークにとってとても重要であり, システムのパフォーマンスはその方式に依存する.Predictive ppersistent CSMA, collision

avaoidance, optional priority, collision detection access scheme がリンク層プロトコルとして用いられるが, アプリケーションがリンク層のプロトコルを指定することもできる.

- Layer3 ネットワーク層:LONWORKS におけるネットワーク層の特徴は, コネクションレス, ドメインでのブロードキャスト, セグメンテーションの概念がないこと, ループに縛られないトポロジ, ラーニングを可能とするルータ, である.
- Layer4 トランスポート層: LONWORKS のトランスポート層は, 完全なデータの転送を保証する. また, 確認応答, 確認応答を必要としないユニキャスト・マルチキャストメッセージ, 認証サーバまでを提供する. 効率的な実装により LonTalk のパケットはとても短い, そのためネットワークの帯域が節約されメッセージは速く到達する.
- Layer5 セッション層: セッション層は, セッションの操作と接続の調整を行う.
- Layer6 プレゼンテーション層: プレゼンテーション層は, 通常, オペレーティングシステムの一部であり, あるプレゼンテーションフォーマットからどのようにデータを変換するかを規定する.LonTalk プロトコルは非常に大きな柔軟性を提供し, ネットワークの変数, メッセージ, 別のフレームを構文として転送することが可能である.
- Layer7 アプリケーション層: 通信相手, 品質保証, データの構文は, この層で規定される. いくつかのアプリケーションはこの層をアプリケーション層として実行するが, この層はアプリケーションそのものではない.LonTalk プロトコルはアプリケーション開発を容易にし, 総合接続性のある製品を提供するため, 下位層の機能も提供することが可能である.

また,LONWORKS/IP の機能は,BACnet/IP のものに似ているが,1 点大きな違いがある. LONWORKS/IP の方法では, 通信パケットは,UDP/IP によりカプセル化される. この方法により, 長い距離のあるインターネットの通信ネットワークを越えることができる [23, 24].

2.1.3 BACnet と LON WORKS

BACnet は物理・データリンク層の 1 つとして, エシェロンの仕様である LonTalk を定義している. そのため, その他の物理・データリンク層の仕様であるイーサネット, ARCNET, MS/TP, PTP と同様に,BACnet のメッセージを伝送することができる. しかし一方で,LonTalk 自体も独自の制御言語を持っている.BACnet の言語とエシェロンの言語が根本的に異なるため, たとえ共通の LonTalk ネットワークに接続していても, いづれかの言語を使用している機器はお互いに相互接続性を持たない.

BACnet はビルディングオートメーションを特に意識して設計されているが,LON はより一般的なネットワークの問題解決のために設計されている. そのため,BACnet には,time-of-day scheduling, alarm generation, control prioritization のような制御が存在するが,LON には存在しない [25, 22].

まとめ

BACnet や LONWORKS はビル管理ネットワークやファシリティネットワークにおいて有力で支配的なプロトコルである. これらは製品やベンダによって異なる API を統一し相互運用を可能とすることを目的としている. これらは非常に優れたプロトコルであるが, 本研究が対象とする管理のアーキテクチャ, 手法については言及していない. また, 本研究では対象とするセンサおよびアクチュエータ機器は

ネットワークレイヤに IP を用いることを想定している.

2.2　ネットワーク管理のための命令セット/データ表現に関する研究

ネットワーク管理の技術やシステムについては, 大まかに以下の要素に分割できると言える.

1. 運用・管理を実現するアーキテクチャ (第 2.3 節)
2. ネットワークを構成する要素のデータ表現や命令セット
3. ネットワーク機器の状態取得, 監視, 制御を行うためのプロトコルについて

本節では (2) ネットワークを構成する要素の命令セットについて述べる. 第 2.2.1 節ではルータやスイッチの管理の標準的なプロトコルを目指す野心的な試みである NETCONF について紹介する. 第 2.2.2 節では, 無線センサネットワークにおけるネットワーク管理の問題について述べる.

2.2.1　NETCONF

NETCONF はネットワーク機器に設定を導入し, 操作し, 削除する方法を提供する. NETCONF ではプロトコルのメッセージだけでなく設定データの表現も Extensible Markup Language(XML)[26] によって行う. アプリケーションは NETCONF の API を使用し, 機器の設定のデータの全てや一部分を送ったり, 受け取ったりすることができる.

NETCONF の特徴

NETCONF のプロトコルは Remote Procedure Call (RPC) のパラダイムを用いる. クライアントは XML でエンコードされた RPC メッセージをサーバに送り, サーバも同様にメッセージを返す. RPC 上のリクエストもレスポンスも同じ XML のスキーマで記述され, 同じように構文を解析できるとする. NETCONF の重要な側面はネットワーク機器のネイティブな機能を充実に実行できるような機能を提供することである. これにより管理者側の実装の負担が軽減され, 適宜機器の新しい特徴を用いることができるようになる. 加えて, アプリケーションは機器のネイティブなユーザインターフェースの構文や文脈にアクセスできる. 例えば, ネットワークトポロジ, リンク, ポリシ, ユーザ, サービスのようなデータのうちの一部のデータセットを取得するようなクエリを出すことができる. このようなデータセットは記述はベンダ非依存なスキーマによってなされる. そして, ベンダや製品,OS に適した形式に変更され使用される [27, 28, 29, 30].

まとめ

NETCONF はデータ表現に XML を用いているという点で本研究に類似している. しかし, 本研究ではセンサおよびアクチュエータの管理を目的としているため提供する命令セットが大きく異なる.

2.2.2 無線センサネットワークにおけるソフトウェア管理に関する研究

無線センサネットワークはセンサを無線によって相互接続し実空間の情報を細かい粒度で取得することを目的としている．無線センサネットワークの研究の初期においては, 主として低電力消費化によるネットワークの長寿命化に関する研究が行われてきた. しかし最近では, そのような低消費が達成された上で, センサネットワークの保守のためにどのような管理基盤を提供するかが研究課題とされてきている [31].

Maté

数多くのセンサノードが存在する無線センサネットワークにおいては, ノードの離脱や追加, ネットワークの構成変更が行われる．そのため, 一度配布されたセンサのソフトウェアの再プログラムが必要となる場合がある．Maté はセンサネットワークのノードのための仮想マシンを提案している．Maté の提供するプログラムインターフェースによりプログラムサイズは軽量になり, プログラム送出の際の電力消費量が低減される．このソフトウェアの実行基盤はアドホックネットワークルーティングやデータのアグリゲーションのために用いられる．

Maté は無線センサネットワークを通してパケットで転送できるような 24 の命令セットを持ったバイトコードのインタプリタである. アドホックルーティングのような複雑な経産が 7 つのカプセルにより表現され, そのコード量は 100 バイト以下となる. そのため, 様々なプログラムを少ないトラフィックで素早くネットワークに導入することが可能となる [32].

本研究で対象とするようなセンサおよびアクチュエータがネットワークに接続しているような状況とは異なるが, センサ機器の展開に当たってはそれらを操作するプログラム言語や実行基盤が必要となることがわかる．

2.3 中央コントローラを用いたネットワーク管理手法

近年の研究動向として中央管理方式によるネットワーク管理手法が存在する．本研究でリファレンスアーキテクチャとして用いる中央管理方式によるネットワーク管理手法について本節では紹介する．

2.3.1 Ethane

Ethane は強力だがシンプルなネットワーク管理モデルを提供する新しいアーキテクチャを提案している．そのアーキテクチャは以下の 3 つの基本原則により構成される．

- ネットワークは高水準の名前によって宣言されるポリシによって決定される．
- ポリシがパケットが通るべき経路を決定する．
- ネットワークはパケットとそのソースの結び付けを保つ.

Ethane によりネットワーク管理者は 1 つのポリシの設定でネットワークの細部から全体までの動作を決定することができる. ポリシは Controller と呼ばれる制御サーバに記述され, 配下にある全てのルー

タ, スイッチ, アクセスポイントなどに自動で反映される. 例えば,「全てのゲストユーザはプロキシ越しの HTTP のみによる通信しか行わない」といったポリシを管理者に負担をかけない命名規則により定義することができる.

Ethane ではハード, ソフト両面での実装を行い, 約 300 のホストで 4 ヶ月の実験運用を行い, その結果と考察を述べている.

Ethane が考える問題点と解決策

Ethane は複雑化するネットワーク管理の問題を中央制御アーキテクチャにより解こうとしている. 中央集中型による解決策はネットワークの研究者にとって通常用いられない方式である.IP のベストエフォートのモデルはシンプルで不変であり, 分散アルゴリズムに適している. しかし, 一方で, ネットワークの管理はその対極にある. ネットワーク管理に対する要求は複雑であり, 強烈な一貫性を必要とする. そのため, 分散的な方法を採用することはネットワーク管理に適さない.

弾力性やスケーラビリティの問題は中央集中方式には常に存在する. しかしながら,Ethane の論文では一般的な冗長化技術により, 十分なスケーラビリティが保証されることを述べている.

Ethane のアーキテクチャと構成要素

Ethane は明示的な許可の無いエンドホスト同士で通信を禁止することで, ネットワークを制御する. このネットワークを実現するために Ethane は 2 つの主なコンポーネントを用意している.

1 つは, すべてのパケットのフローを決定する全体のネットワークポリシを含む中央コントローラである. パケットがコントローラに到達すると, コントローラはそのパケットに相当するフローが許可されるべきか否かを決定する. コントローラは全体のネットワークトポロジを知っていて, 許可されたフローの経路の計算を行う. 明示的にスイッチ間のフローを許可することで, ユーザからのアクセスが選択された経路に沿ってなされることを保証している. コントローラは冗長化やパフォーマンスの向上のため, 複製することも可能である.

2 つめの要素は,Ethane に対応したスイッチである. 高機能でネットワークの全体情報を把握しているコントローラと対照的に, これらのスイッチは簡素で単純である.Ethane 対応のスイッチは通常のフローテーブルとコントローラへの安全な通信路で構成され, 単純にコントローラの指示通りにパケットを転送する. フローテーブルに存在しないパケットが到着した場合, パケットが到着したポートの情報とともに, スイッチはコントローラにパケットを転送する. Ethane ネットワークの全てのスイッチが Ethane スイッチである必要はなく, 対応スイッチが徐々に増えていくことが可能となっている.

Ethane の検証と考察

Ethane のプロトタイプは約 300 のホストで構成され, 平均で約 120 のホストが動作している. フロー要求の 1 秒当たりの数はピーク時で 750 で, 通常は 30-40 である. 一方,Controller が処理可能なフロー数は 1 秒当たり 11,000 である. これは, 実験のピーク時のフロー数よりも十分に小さい値であり,CPU の負荷は無視できるほどであった.

プロトタイプ環境での使用に十分な性能であるため, この論文では 8,000 のホストからなる LBL と 22,000 のホストからなるスタンフォード大学のデータセットを用いた制御可能なホスト数を算出して

いる．LBL のデータセットにおける 1 秒当たりのフロー数は 1,200 を越えない. また, スタンフォード大学のデータセットでは 9,000 のフロー数を越えない．そのため,Ethane のプロトタイプでは,1 つの Controller で 20,000 のホストを処理可能であるとしている．

実際の運用では, ルールセットが大きくなり, 物理的なオーバヘッドも大きくなることが予想される．しかし,Ethane システムも実装を改良していく余地がある．また,Controller は 1 つで運用されるのでなく複数で運用されることが想定されている．これらの点を考慮して,Ethane は十分スケーラブルであると言えよう．

Ethane のまとめ

Ethane は中央管理方式のアーキテクチャにより, ネットワークの管理に纏わる諸問題を解決することを提案している．中央管理方式は強烈な一貫性を必要とする, 複雑なネットワーク管理の問題を容易にするが, スケーラビリティ, 弾力性という点に問題があると考えられ, デプロイメントの障害になると考えられる．Ethane ではその問題を勘案し, プロトタイプ実装を行い, 約 300 のホストが動くネットワークで検証を行った. その結果, 企業ネットワークやキャンパスネットワークを考慮した規模では,Ethane の提案するアーキテクチャが十分スケーラブルであることを示している [11].

2.3.2　Open Flow

Open Flow は, 日々使用するネットワークに実験的なプロトコルを走らせる研究者に実際的な方法を提供し, ネットワークベンダが大学キャンパスのバックボーンやサーバ室に展開できるようベンダが自社製品に Open Flow を加えることを目的としている.Open Flow で使用されるイーサネットスイッチは, フローテーブルとフローのエントリを追加, 削除するための標準的なインターフェースを備えていれば良い.

Open Flow は実際的である一方, 不均一なスイッチが存在する状況において, 各スイッチがラインレートで動作し高度なポートの密度でも使用が可能とする. 一方で, ベンダはスイッチの内部構造を公開する必要はない.Open Flow はキャンパスにおける大規模なテストベッドを構成し得る．OpenFlow は現在, ネットワーク機器ベンダによる実装が進行中で, いくつかの大学では実際に試験運用が開始されている．

OpenFlow が考える問題点と解決策

OpenFlow は以下を問題として掲げている．

1. インターネットはビジネスや家庭, 教育において欠かせないインフラとなった. それは, ネットワークの研究者にとって喜ぶべきことだが, 同時に足枷にもなっている. 具体的には, 以下の理由による．
2. 既に設置されている莫大な数のネットワーク機器, プロトコルを新しいものに置き換えるのには非常に大きなコストがかかる．
3. 生活ネットワークとして用いられているネットワークで実験的な機器, プロトコルを動作させることは, ネットワークの安定運用の面から難しい．

4. 新しいプロトコルが大規模なネットワークで動作することを検証するために, 実運用されているネットワークを使用するが難しい.
5. これらのために, ネットワークの研究者からなされる新しい提案のほとんどが検証されずに終わっている.

そして, 以上の問題を

1. 商用のネットワーク機器やルータに頼らない (なぜなら, それらは通常, オープンなソフトウェアプラットフォームを提供しないし, インターフェースの提供の幅も大きくない. 内部の構造は, それが企業の競争力の源であるため, 公表しない.)
2. 十分なポート密度を持ち, ラインレートで動作すること
3. デプロイメントが高価にならないこと

という点を考慮して, 解こうとしている.

OpenFlow のアーキテクチャと構成要素

OpenFlow は, 最近のほとんどのイーサネットスイッチやルータがフローテーブルを持っている点に着目し, それを利用している. フローテーブルは, 一般には TCAM によって構成され, ファイヤウォールや NAT,QOS, 統計情報の取得のため, ラインレートで動作する. 各ベンダのフローテーブルは仕様が異なる一方で, 多くのスイッチやルータが共通して持つ機能が存在する.OpenFlow はそのような各種スイッチ, ルータに共通する部分を利用して, それらのフローテーブルを操作するオープンなプロトコルを提供する.

OpenFlow が提供する機能により, ネットワークの管理者はトラフィックを生活用と研究用に分けることができる. パケットが流れる経路と受け取る処理を選択することで, 研究者は自分のフローを制御することができる. この方法で, 研究者は新しいルーティングアルゴリズム, セキュリティモデル, アドレス方式, さらにはインターネットプロトコルに代わるプロトコルまで, 試すことができる. そしてその同じネットワークで, 生活用のトラフィックは分離され, 今までと同じように処理される.

以上のように,OpenFlow では商用スイッチのフローテーブルを操作することで, スイッチ内部のフォワーディングの操作を実現する. そして, このインターフェースを利用して OpenFlow のスイッチを操作するのが Controller と呼ばれる,OpenFlow ネットワークの制御装置である.Controller はオープンフロー対応のスイッチにフローテーブルのエントリの追加, 削除などの指示を出す. 例えば, 研究者が実験中の複数のコンピュータを繋げる場合, 実験コンピュータのトラフィックと生活ネットワークのトラフィックを分離するようなフローテーブル操作の指示が Controller から出される. あるいは, 進行中の実験のフローを動的に追加, 削除することも可能である. このように,OpenFlow は指令を出す Controller とフローテーブルを自在に変更できるスイッチにより構成される.

Open Flow における, ネットワークの実験の流れは, 以下のようになる. アミーという研究者が, 自分の考えた新しいプロトコルを, 多くの人によって使われるネットワークで試したいとする. この時,Open Flow のネットワークは以下の 2 つの性質を有する.

1. アミー以外のユーザに属するパケットは標準で検証済みのプロトコルが動く正式なスイッチや

ルータを通る.

2. アミーはネットワーク管理者の許可を受けた自分のトラフィックやその他トラフィックにフローエントリを追加する.

まとめ

OpenFlow の構成要素は大きく分けて以下の 2 つの要素である.

1. OpenFlow 対応スイッチ： Controller の指令を受け, 内部のフローテーブルを変更可能なスイッチ.
2. Controller：OpenFlow 対応スイッチにフローテーブルのエントリの追加・削除という形で指令を出す.

OpenFlow は, 実験を行いたい研究者が Controller を通して (キャンパス) ネットワークの状態を変更するという, 中央管理のアーキテクチャに基づいている. Ethane との技術的な優位点は,OpenFlow はスイッチのフローテーブルの機能に着目しており,Ethane のスイッチを一般的にした点にあると言える. また,OpenFlow は現在実装が進行中のアクティブなプロジェクトである.

2.3.3　SANE

SANE は Ethane 同様, 企業ネットワークにおけるネットワーク管理の問題を解こうとしている.SANE ではネットワーク内の全ての接続を司る 1 つの保護レイヤを定義する.Open Flow や Ethane 同様, 中央に配置したサーバがネットワーク内のルーティングやアクセスコントロールの決定を行う.SANE は段階的なスイッチの展開が不可能なため,Ethane が提案された [14].

2.3.4　Clean Slate 4D

Clean Slate 4D では, ネットワークの脆弱さや管理の難しさをネットワークの制御プレーンや管理プレーンの複雑さにあるとしている. インターネットでは, ソフトウェアやプロトコル, ネットワークの決定が分散的になされ, 複雑に結びついている. Clean Slate 4D ではネットワークレベルのオブジェクト, ネットワーク全体を見渡した決定, 直接的なコントロールの 3 つを原則とし, ネットワークアーキテクチャの再調整を行う.Clean Slate 4D の 4D は,decision, dissmeniation, discovery, data の 4 語の頭文字である. 4D のアーキテクチャは AS の決定ロジックによりネットワーク上でやり取りを支配するプロトコルから完全に分離する. AS レベルのオブジェクトは決定プレーンで全てが決まり, データプレーンでどのようにパケットを転送するかはルータ間で直接設定がやり取りされる. 4D のアーキテクチャではルータやスイッチは決定プレーンの命令に基づき単にパケットを転送するだけである. そして決定プレーンによるネットワークの制御を支援するため統計情報を収集する. 4D ではネットワークの管理や制御について大量の変化を要するが, パケットのデータ形式の変更は必要としない [15].

まとめ

中央集中型のアーキテクチャは一般にスケーラビリティに劣っていると考えられている．大規模に動作しているインターネットは分散型のアーキテクチャである．一方で, 一貫性に優れ簡易なシステムとなる中央集中方式は管理アーキテクチャとして適していると考えられ, この方式を採用している研究がいくつか存在する．特に Ethane では中央管理方式によるシステムのプロトタイプを実装し, 企業ネットワークやキャンパスネットワーク程度の規模であればスケーラビリティに問題が無いことを示している．これらの研究の対象領域は本研究の対象とは異なるが, 中央集中型の管理方式はセンサおよびアクチュエータの管理にも適していると考えられる．なぜならば, センサおよびアクチュエータの管理は, 通常のルータ・スイッチ機器の管理以上に多数の管理対象があり多様で複雑だからである．一貫性に優れシンプルな中央管理アーキテクチャを採用することはセンサおよびアクチュエータ機器の管理にとって自然である．

第 3 章

CCDM の中央コントローラに基づくアーキテクチャ

本論文では中央管理方式によりセンサおよびアクチュエータの統合的な管理を実現する Central Controller-based Device Management(CCDM) を提案する．CCDM は以下に挙げる 2 点において大きな特徴を持つ．

1. アーキテクチャ: CCDM はネットワーク管理に適した中央集中方式による情報管理と耐障害性に強い分散実行のアーキテクチャを実現する．
2. 命令セット: センサおよびアクチュエータの操作や管理に適した管理インターフェースと命令セットを提供する．

本章では, まず,CCDM が前提とする環境で中央集中型による管理と分散実行のアーキテクチャが妥当である理由 (第 3.1 節) を説明する．次に, 実現に当たって必要となる要件 (第 3.2 節) を説明し, システムの構成要素 (第 3.3 節) と続く．命令セットの詳細と分散実行の実現方法については第 4 章に説明を譲る．

3.1 提案の前提とアーキテクチャの検討

ネットワークの管理は, 煩雑で, 間違いを起こしやすく, 人手のミスによるものが大きい. 本論文では, キャンパスネットワークやファシリティネットワークにおいて, センサおよびアクチュエータ (デバイス) の管理に適切なシステムを提案することを目的としている．具体的には, 以下のような状況が前提である．

- キャンパスやオフィスビルのような生活空間に多数のデバイスが存在する．
- デバイスはインターネットへの接続性を持つ．
- センサが収集した情報を活かすためには, アクチュエータへの制御が必要である. センサの情報を集約, 解析しアクチュエータへの制御を行うようなサービスを提供するホスト (サービスノード) がネットワーク上に存在する．
- ルータやスイッチによって隔てられるネットワークセグメント毎 (「ネットワークセグメント」

は通常レイヤ 3 のネットワーク区分の意味で用いられるが, ここではレイヤ 2 での区分けの意味も含む) にサービスノードの導入を想定できる.

デバイスが多数存在する状況では, それらの管理, 維持が非常に煩雑で多大な手間を要する. すなわち, 多数のデバイスの関係が複雑であるため,1 つのデバイスの設定変更がネットワークの広範囲に影響を及ぼし, 多数のサービスの設定変更を迫る. 変更を行うべき設定の箇所が多ければ多いほど, 設定の誤りを犯す可能性は高くなるだろう.

そこで, 本研究では, 一貫性の提供に優れシンプルなシステムとなる中央集中方式による手法を管理アーキテクチャとする. 中央管理アーキテクチャでは, 論理的に中央に存在するサーバ (コントローラ) がネットワークのノードの動作を制御する. 一般に, 中央集中方式のアプローチがインターネットの研究において採られることは多くないが, 一貫性の保持に優れシンプルなシステムとなる中央集中方式は管理アーキテクチャとして優れている (第 2.3 節を参照). 膨大な数のデバイスが存在するネットワークにおいても, 管理が統一的になされているため 1 点の変更を容易に全体に反映できるようなシステムを構築することができる.

中央集中方式によるアーキテクチャは管理には優れている. 一方で, 集中的な実行はスケーラビリティに劣り, 常に同一地点に存在するサーバを利用するためネットワークの障害に弱い. 例えば, 温度センサとその温度データを出力するディスプレイがネットワーク的に近い位置にある場合, 遠くのサーバを介さず近くのサーバで通信が完結する方が良い.

そこで本研究では, 管理プレーンと実行プレーンの分離による中央管理分散実行のアーキテクチャを提供する. 管理を司るコントローラは論理的に 1 台だが, 実行を担当するノードは各ネットワークセグメントに存在しネットワーク的に近いデバイスへのサービス提供を行う. 実行を担当するノードは 1 度サービス提供の指令をコントローラから受け取れば, その指示通りに動き続ける. そのため, コントローラへの接続性がなくなっても, ローカルネットワークに存在するデバイスにはサービスの提供を継続することができる. CCDM では, 管理プレーンと実行プレーンの分離により, 管理者にとっては管理が容易でサービスの耐障害性に優れるアプリケーションの実行基盤を提供する.

3.2　中央管理分散実行のアーキテクチャの実現の要件

CCDM では中央管理分散実行のアーキテクチャを実現する. 中央管理分散実行のアーキテクチャに登場するのは, まず, 管理を司るコントローラとその管理を実現する実行ノードである. コントローラは論理的に 1 台が存在し, 実行ノードは各ネットワークセグメントに配置される. 実行ノードは, コントローラからの指令を解釈し実行するようなランタイムを導入している必要がある.

アプリケーションの実行に当たっては, コントローラに記述された管理スクリプトが解釈され, 各実行ノードで動作する必要がある. アーキテクチャの実現に当たって課題となるのは, 集中的に記述された管理スクリプトから分散的に動作する実行スクリプトを生成する部分である. CCDM では中央管理分散実行の実現のため, 管理スクリプトとして提供するサービスの情報を記述したサービススクリプトとネットワークの状態を記述したリソーススクリプトの 2 種類を用意する. サービススクリプトはリソーススクリプトによって適切な形に変形され, 各ノードに配置される必要がある. この機能はコントローラ

が持つべき機能である.

また, 管理者に提供する命令セットも重要な検討事項となる.

3.3　CCDM の構成要素

図 3.1 に CCDM のアーキテクチャの概要と構成要素を示す. CCDM の構成要素として以下の 8 要素を定義する.

1. CCDM コントローラ
2. CCDM ノード
3. デバイス
4. スクリプト生成機構
5. CCDM ランタイム
6. サービススクリプト
7. リソーススクリプト
8. 実行スクリプト

以下の第 3.3.1 節〜第 3.3.8 節では,CCDM の構成要素と用語を定義し, 解説する.

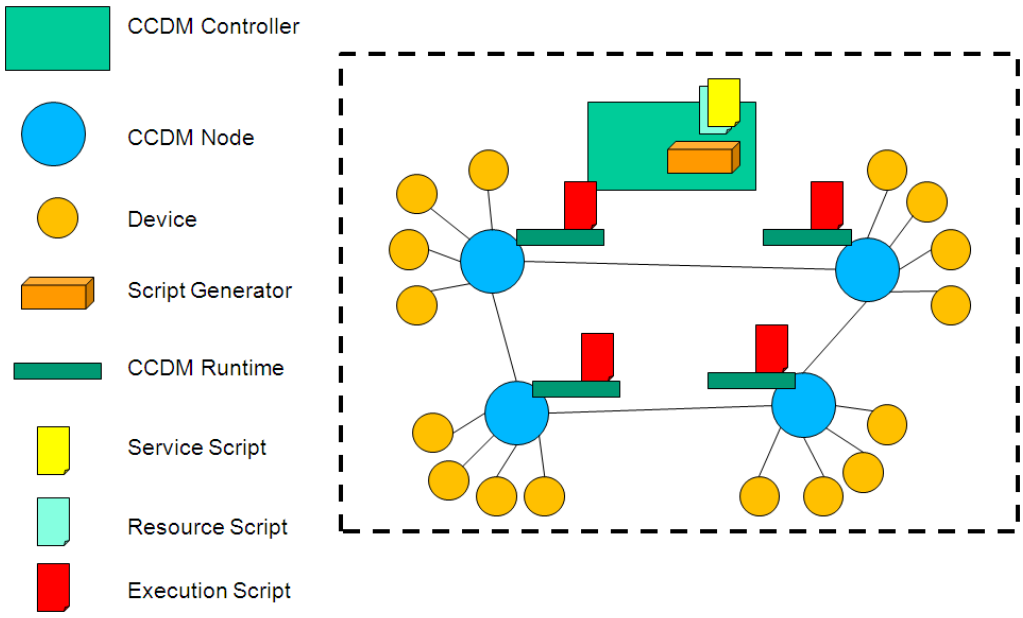

図 3.1　CCDM の概要と構成要素

3.3.1　CCDM コントローラ

CCDM コントローラは CCDM システムに論理的には 1 台のみ存在するコントローラである. CCDM コントローラは CCDM システムの管理を司り, 各 CCDM ノードにサービス実行の指令を出す.

CCDM コントローラはスクリプト生成機構を持ち,1 つ以上のサービススクリプトと 1 つのリソーススクリプトが記述される. スクリプト生成機構により生成された実行スクリプトの配布という形で, CCDM ノードへのサービス実行の指令を出す.

3.3.2　CCDM ノード

CCDM ノードは各ネットワークセグメントに設置される CCDM システムの実行ノードである. CCDM ノードは CCDM ランタイムを持つ. CCDM コントローラから実行スクリプトを受け取り, ランタイム上で解釈, 実行する. CCDM ノードは, あらかじめ CCDM コントローラのネットワークアドレスの情報を保持しているとする.

3.3.3　デバイス

デバイスは CCDM のサービスを利用するノードである. デバイスは周辺の環境情報を取得するセンサと周辺の環境に影響を与えるアクチュエータの集合である. 具体的には, センサは温度センサや照度センサ, スイッチであり, アクチュエータはディスプレイ, 照明, ブザーなどである. 温度センサならば CCDM システムに温度をアップロードし, ディスプレイならば CCDM から受け取ったデータを表示する. デバイスは特別なソフトウェアの導入を必要としない.

3.3.4　スクリプト生成機構

スクリプト生成機構は, サービススクリプトとリソーススクリプトから実行スクリプトを生成する. 生成の具体的な方法については第 4 章で述べる.

3.3.5　CCDM ランタイム

CCDM ランタイムは実行スクリプトを解釈し, 実行する. CCDM ランタイムは java によって記述されており,XML による高級言語の構文解析器と評価器である [16].

3.3.6　サービススクリプト

CCDM で提供されるサービスを中央集中的に記述したものがサービススクリプトである. 四則演算や条件分岐, 判断などのデバイスの管理に必要な命令セットで構成される. サービススクリプトの詳細ついては第 4 章で述べる.

3.3.7 リソーススクリプト

リソーススクリプトには CCDM のネットワークの状態が記述される．具体的には,CCDM ノードの IP アドレスや利用可能なポートデバイスの通信プロトコルなどである．リソーススクリプトの詳細については第 4 章で述べる．

3.3.8 実行スクリプト

実行スクリプトは CCDM ランタイム上で動作するスクリプトである．XML により記述される高級言語である [16].

第 4 章

CCDM の分散実行の実現

本章では, 実行スクリプトとランタイム (第 4.1 節), スクリプトの結合と分割を実現するアーキテクチャ (第 4.2 節), サービススクリプト (第 4.3 節), リソーススクリプト (第 4.4 節), サービススクリプトとリソーススクリプトの結合方法 (第 4.5 節) の順に説明する.

4.1 実行スクリプトとランタイム

CCDM ではコントローラに存在するスクリプト生成機構により, 各 CCDM ノードに適した実行スクリプトが生成される. 実行スクリプトの実行には, スクリプトの定義とランタイムが必要となる.

CCDM では実行スクリプトに [16] で提案されたネットワーク汎用 I/O 制御のための高級スクリプト言語を用いる. この高級言語は, ネットワーク接続された組込み機器との固有の通信を翻訳スクリプトとして記述する. スクリプトを制御ソフトウェアから隠蔽する翻訳機構を提供することを目的としており, 以下のような特徴がある.

- 本高級言語は, 汎用 I/O(GPIO) のような通信機能を持った組込み機器を対象としている. そのため, 本研究で対象とするデバイスの操作や制御に適した命令セットを備えている.
- 制御ソフトウェアの開発効率や再利用性が高い.
- XML により記述された柔軟性が高いオブジェクト志向のスクリプト言語である.
- 命令が XML のタグにより記述されているため, 置換が容易である.

CCDM ではネットワークに接続したデバイスへのサービス提供を行う. 実行スクリプトとして以上の特徴を持つ高級言語を採用することは理に適っているといえよう. CCDM では実行スクリプトとして [16] の高級言語を使用し, ランタイムも同研究で開発された java による構文解析器・評価器を使用する.

4.2 スクリプトの結合と生成

CCDM ではサービススクリプトとリソーススクリプトという 2 種類のスクリプトの内容を結合して, 新しい複数のスクリプトを生成する. その概念を図 4.1 に示す.

サービススクリプトもリソーススクリプトも Document Object Model(DOM) のツリーと見ることができよう. サービススクリプトのツリーは CCDM ノード毎に情報が記述されたリソーススクリプト

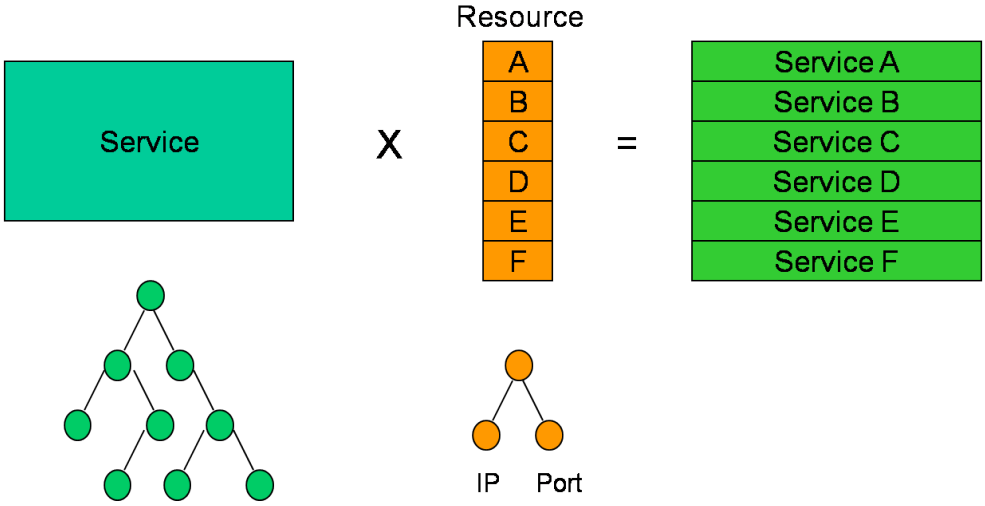

図 4.1　スクリプトの結合と生成

に従って CCDM ノードの数の分だけ実行スクリプトが生成される. この機能を提供するのがスクリプト生成機構である. 詳細については次節以降で述べる.

4.3　サービススクリプトの命令セット

サービススクリプトはネットワーク管理者への唯一のインターフェースである. ネットワーク管理者やサービスの提供者はサービススクリプトに自身が実現したい管理やアプリケーションの内容を記述する. サービススクリプトで提供する命令セットの種類や粒度は重要となる. サービススクリプトは実行スクリプトと同様に基本的には [16] による高級言語の命令セットを踏襲する.CCDM のサービススクリプトではデバイスの通信プロトコルの隠蔽や分散実行の実現のためにいくつかの命令セットの追加を行う.

サービススクリプトの命令セットを表 4.1 に示す. 命令セットとして, 基本的な四則演算, ビット演算, 論理演算, 条件分岐などが定義されている. 通信機能としては, データグラムソケットが定義されている. プロトタイプでの定義は,UDP によるデータグラム通信のみだが,TCP によるストリーム通信やシリアルインターフェースを介した通信のための命令セット拡張も可能である. 命令の分散実行の実現のための命令セットが,chunk(第 4.3.1 節), post/catch(第 4.3.2 節) であり, デバイス毎の通信プロトコルやインターフェースを隠蔽し, 抽象化した機能を提供するものが INPUT/OUPUT(第 4.3.3 節) である. また,post/catch 命令により, サービスの記述者はどの CCDM ノードでサービスが提供されるかを意識せずにスクリプトを記述することができる. これらの命令が XML 上の言語として規定され, メソッド集合として XML タグで表現される.

表 4.1　サービススクリプトの命令セット

Category	Command
Data Type	byte int double text null true false
Operation	plus minus multiply divide mod
Bit Operation	bitand bitor bitxor bitlshift bitrshift biturshift
Logical Operation	and or not xor
Test	eq neq lt gt lteq gteq
Type Cast	castbyte castint castdouble casttext
Branch	if
Loop	while
Procedure	progn
Variable	setq getq
Array	byteArray byteArrayAppend
	byteArrayGet byteArrayLength
Socket	dgSocket dgServerSocket dgSend dgRecv dgClose
Distributed Execution	chunk post catch
Device Communication Abstraction	INPUT OUTPUT

4.3.1　chunk

chunk 命令は, 命令セットのうち, 分割不可能な命令の集合である. 命令型プログラミングにおいてはプログラムの状態を持つ必要がある. 分割不可能な命令の集合とは, 同じ状態を共有する命令列である. つまり,chunk とは同一の名前で変数 (サービススクリプトでは,setq, getq) を扱うことができる命令のかたまりのことである.chunk 命令タグには progn 命令と同様に, 他の命令を入れ子構造にして挿入できる. 異なる chunk 同士で変数の交換を行いたい場合には第 4.3.2 節で述べるように post/catch 命令を用いることができる.

4.3.2　post/catch

post/catch 命令は第 4.3.1 節で述べた chunk 間のデータ転送を先入れ先出し方式 (First-In First-Out, FIFO) に基づき実現する. 図 4.2 に post/catch 命令の概要を示す. chunk 間のデータ転送に FIFO のメカニズムを用いるのは, 以下の理由による.

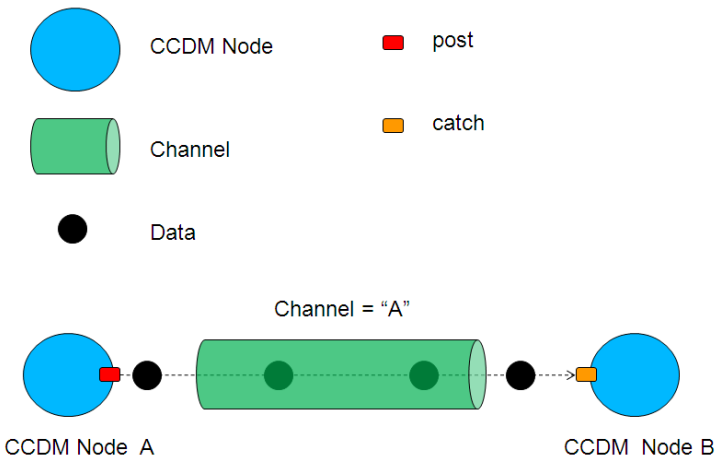

図 4.2　post/catch 命令

1. 最もシンプルなインターネットのパケット転送は受け取ったパケットの順に次の宛先に転送する.
2. 送信側, 受信側双方で同期や順序制御, 再送制御などの複雑な機能を実装する必要が無い.
3. (2) の理由のため,1 対多のデータ転送を実現し易い. 送る側は自由に送れば良く, 受け手もデータの受信順に処理を行えば良い.

post/catch の内部は UDP データグラムの転送により実装されている. そのため, 順序制御や再送制御, 確認応答などの処理は行わない.

post/catch 命令は 2 つの属性を持つ.1 つは name 属性,1 つは channel 属性である. post/catch 命令はある対の post,catch 間でデータの転送を行う. 転送されるデータを指定するのが name 属性である.post の際には name 属性の値と一致する変数の値が送られ,catch の際には name 属性で指定された名前の変数に受け取った値が代入される.

また,post/catch 命令は channel と呼ぶ post-catch 間の通信路の識別子を持つ. post 命令により送出されるデータは, 同一の channel の catch にデータを転送する. また,post/catch はマルチキャスト的なデータ配送が可能である (図 4.3). 例えば, channel が”A” の post 命令が 1 つあり, 同様に channel が”A” の catch 命令が 3 つある場合,post されたデータは 3 つの catch に配送される. また, 逆も同様で,channel が”B” の post 命令が 3 つあり,channel が”B” の catch 命令が 3 つある場合,catch は 3 地点からのデータを受け取ることができる (図 4.4). 受け取るデータの順序制御は行われず,FIFO の原則に基づき受け取ったデータ順に処理される.

4.3.3　INPUT/OUTPUT

INPUT/OUTPUT 命令は制御対象の機器との通信を行う際に発行される命令で, デバイス毎の通信プロトコルやインターフェースの差異を吸収した API をネットワーク管理者に提供する. INPUT

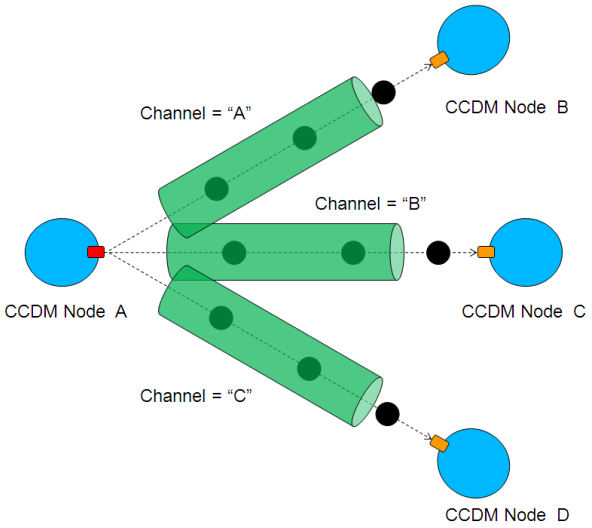

図 4.3　post/catch 命令 (2)

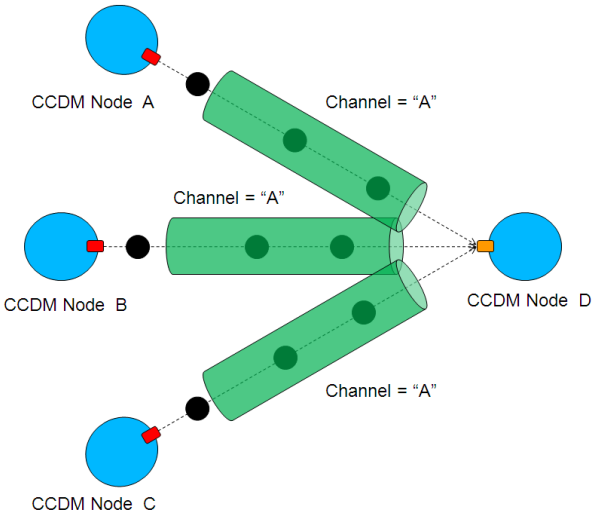

図 4.4　post/catch 命令 (3)

命令は from 属性を持ち,from で指定された機器からデバイス特有のプロトコルでデータを取得する. OUTPUT 命令は to 属性を持ち,ARG 命令で指定した命令を引数として渡すことができる. to で指定した機器には,INPUT 命令同様, デバイス特有のプロトコルでデータを送信する. 機器特有のプロトコルへの命令の変換は第 4.4 節で述べるリソーススクリプトを参照することにより行う.

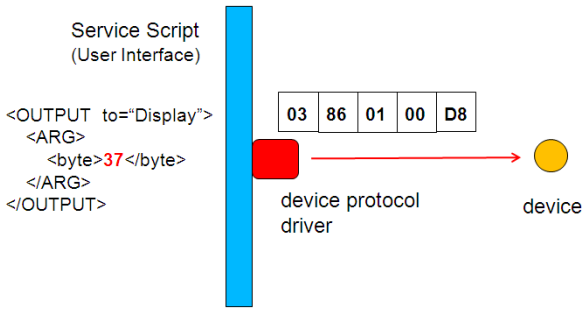

図 4.5　INPUT/OUTPUT 命令

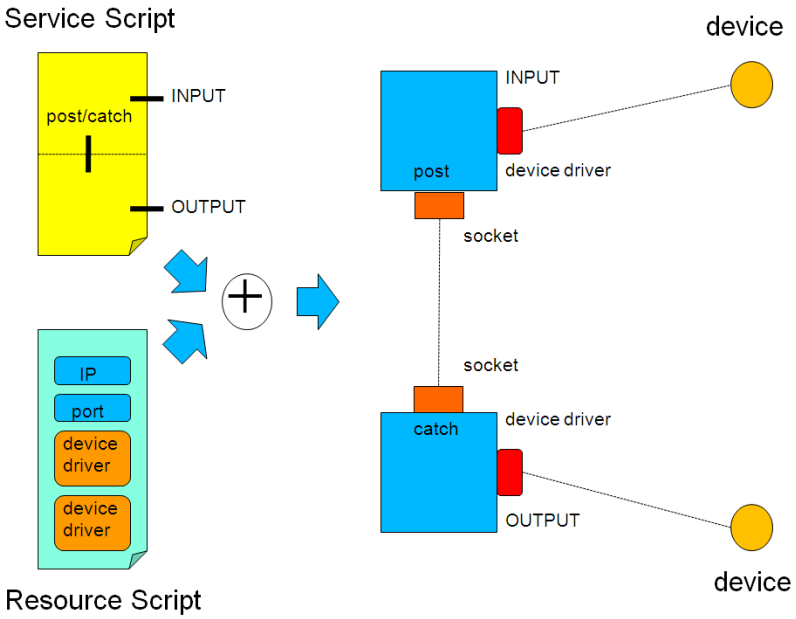

図 4.6　サービススクリプトの特徴

例えば, 図 4.5 では,ASCII 文字コードで表現された数字 7 を, ネットワーク接続した液晶ディスプレイに表示する際の例を載せている. ユーザは, サービススクリプトには,「ASCII 文字コードで 7 を意味する 37(16 進数) を Display という機器から出力する」と, 図のように XML タグで表現する. サービススクリプトの記述は, リソーススクリプトの参照の結果, 機器に応じた適切なデータ列 (03 86 01 00 D8) に変換され, 送信される.

4.3.4　サービススクリプトの特徴

サービススクリプトは,chunk,post/catch,INPUT/OUTPUT の特徴的な命令を持つ. これらの命令の特徴と分割後の状態を図 4.6 に示す. chunk はスクリプトの分割を可能にし,post/catch は分割された chunk 間のデータの交換方法を提供する.INPUT/OUTPUT はデバイス毎の通信プロトコルを抽象

化するインターフェースを提供する.

サービススクリプトの特徴を以下にまとめる.

1. 命令セットの記述言語は XML で規定され, 命令は XML タグで表現される.
2. 四則演算, 論理演算, 条件分岐などのデバイスの管理, 操作に必要な基本的な命令セットが定義されている.
3. ネットワーク管理者は, 記述したサービスがどの CCDM ノードで実行されるかを意識せずにスクリプトを記述することができる.
4. デバイス毎の通信プロトコルを抽象化して扱うことが可能である.

4.4　リソーススクリプトのスキーマ

CCDM においては,CCDM ネットワーク上で提供されるサービスの内容を記述したサービススクリプトとリソーススクリプトが存在する. このサービススクリプトとリソーススクリプトが重ね合わさることにより,CCDM ランタイムが解釈可能なスクリプトが生成される. 第 4.3 項で述べたように, サービススクリプトは, ネットワーク管理者により容易なサービス提供のインターフェースを提供するため,

1. サービスの記述者は, 記述しているサービスがどのホストで動作するかを意識せず, サービススクリプトを記述できる
2. サービスの記述者は, デバイス毎の通信プロトコルの差異を意識せず, サービススクリプトを記述できる.

というスクリプト構造になっている. CCDM ネットワークに参加している CCDM ノードやデバイスの情報, デバイス毎に異なる通信プロトコルの情報がリソーススクリプトに記述されることになる. リソーススクリプトはサービススクリプト同様に,XML 形式で記述される. 図 4.7 にリソーススクリプトの XML スキーマを示す.

リソーススクリプトのルートは resource である.resource は複数の host を持つ.host は CCDM ノードのことであり, 各 CCDM ノードの IP アドレス, 使用可能なポート, 接続しているネットワーク機器などの計算機資源の情報を持つ. 1 つの host には, 複数の device の情報を記述することが可能である. device は CCDM サービススクリプトと同様の命令セットを記述するようになっており, このタグにおいて機器特有の通信プロトコルなどが記述される.

プロトタイプ実装では, リソーススクリプトはネットワーク管理者が手動で作成している. しかし, ホストの IP アドレスやデバイスの通信プロトコルのようなリソーススクリプトに記述される情報は管理者に依存せず, デバイスやネットワークの状態により一意に定まる. そのため, リソーススクリプトに関しては,CCDM ノードやデバイスの情報の自動収集によって生成することが可能である. 例えば, リソース情報の自動収集には Simple Network Management Protocol(SNMP)[33, 34] のような既存のプロトコルを使用することも考えられる. また, 異なる手法によりネットワーク情報を取得することも考えられる.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="resource" type="type_resource"/>
  <xsd:complexType name="type_resource">
    <xsd:sequence>
      <xsd:element name="host" minOccurs="1" maxOccurs="unbounded"
        type="type_host"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="type_host">
    <xsd:sequence>
      <xsd:element name="ip" type="xsd:string"/>
      <xsd:element name="port" type="type_port"/>
      <xsd:element name="devicelist" type="type_devicelist"/>
      <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="type_port">
    <xsd:sequence>
      <xsd:element name="begin" type="xsd:positiveInteger"/>
      <xsd:element name="end" type="xsd:positiveInteger"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="type_devicelist">
    <xsd:element name="device" minOccurs="0" maxOccurs="unbounded"
      type="type_device"/>
  </xsd:complexType>
  <xsd:complexType name="type_device">
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="unbounded" type="the command
set of CCDM Service Script" />
      <xsd:attribute name="name" type="xsd:MNTOKEN" use="required"/>
      <xsd:attribute name="func" type="function" use="required"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="function">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="INPUT"/>
      <xsd:enumeration value="OUTPUT"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

図 4.7　リソーススクリプトのスキーマ

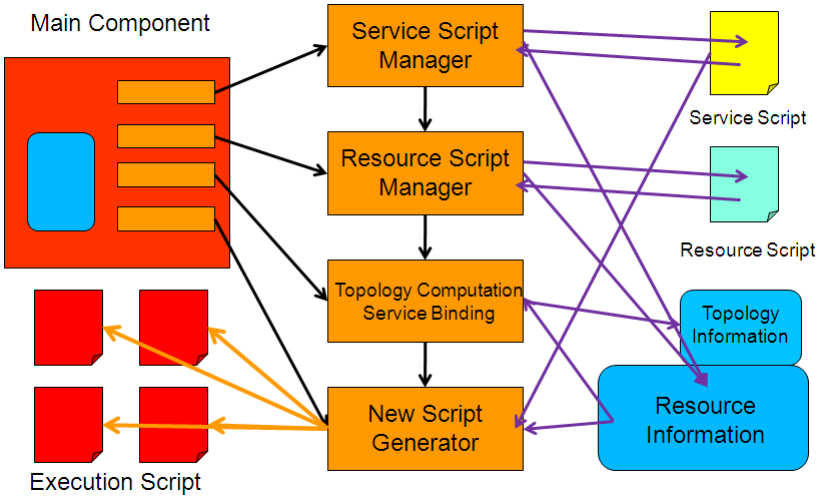

図 4.8　スクリプトの生成機構

4.5　リソーススクリプトによるサービススクリプトの置換

CCDM コントローラで記述されたサービススクリプトとリソーススクリプトは, 重ね合わせにより CCDM ノード毎に異なる実行スクリプトが生成される. 実行スクリプトの生成の機構の概要を図 4.8 に示す. スクリプトの生成は,

1. サービススクリプトの解釈を行う Service Script Manager
2. リソーススクリプトの解釈を行う Resource Script Manager
3. サービススクリプトとリソーススクリプトの情報に基づいたオーバレイネットワークトポロジの生成とセンサおよびアクチュエータの結びづけを行う Topology Computation and Service Binding
4. (3) の結果に基づき, 新しいスクリプトを生成する New Script Generator

の 4 つの段階で行われる.

(1),(2) によって解釈された情報は,Resource Information のインスタンスに保存される.Resource Information は内部で,chunk と device の関係, device と host の関係, 各 host の情報,post/channel の情報, 機器の通信プロトコルの情報,chunk の総数,host の総数を持つ. 実行スクリプトの生成の際には, 以上の情報を約 20 の関数により提供する.

4.5.1　Service Script Manager

Service Script Manager は, サービススクリプトに記述されている内容のうち, トポロジ計算とセンサおよびアクチュエータの結合のために必要な情報を抜き出し,Resource Information に格納する. 具体的には, サービススクリプト内の以下の情報が抽出される.

- chunk: chunk の総数とサービススクリプト内での chunk のインデックス.
- post/catch: 各 post/catch が使用している channel 名とスクリプト内での位置.
- INPUT/OUTPUT: 各 INPUT/OUTPUT が結び付いている機器の種類とスクリプト内での位置.

4.5.2　Resource Script Manager

Resource Script Manager はリソーススクリプトの情報を読み出し,Resource Information に格納する. Resource Script Manager により読み出される情報は以下である.

- host: host の総数とリソーススクリプト内での host のインデックス.
- ip: 各 host の IP アドレス.
- port: 各 host で利用可能な IP アドレスの幅. 開始のポート番号から最後のポート番号まで.
- device: 各 host に接続しているデバイスの種類, 通信プロトコル.

4.5.3　Topology Computation and Service Binding

Topology Computation and Service Binding は,Resource Information に基づき, 実行スクリプトの生成に必要な計算を行う. 具体的には,CCDM のルーティングの決定, デバイスの結びつけ, 使用可能なポートの割り当てを行う.

まず,CCDM のルーティングについての前提条件を以下に記す.

- CCDM におけるメッセージのルーティング:CCDM においては, デバイスがメッセージを送出し, ネットワーク上のノードである CCDM ノードでデータの計算やプロトコルの変換が行われる. その結果が出力先のデバイスに送出される.
- CCDM においてデバイスの制御を複数のホストで行うのは, 計算量を割り振ることではない. エンドノードに近い CCDM ノードでサービスを提供することで, ネットワークの障害に強い実行環境を提供し, センサ-アクチュエータ間のメッセージの遅延を小さくすることが目的である.
- サービススクリプトの 1 つの chunk は 1 つ以下のの INPUT/OUTPUT を持つ.

CCDM ではネットワークレイヤのプロトコルに IP の使用を前提としている.CCDM では余分な CCDM ノードを中継せず, デバイスに接続している CCDM ノードに直接データを転送した方が良いと考えられる. CCDM では, メッセージの INPUT を担当する CCDM ノードから OUTPUT を担当する CCDM ノードまで,1 ホップでデータを転送し,2 台で計算を分担することとする. 以上の方針に基づき,Topology Computation では,

1. chunk と紐付いている device に確認
2. device が紐付いている host から,chunk が実行されるべき host の決定
3. 紐付いていない chunk の場合は, スクリプト上で近い chunk と同一 host で実行する

というポリシにより, ルーティングの決定を行う.

Topology Computation and Service Binding では,post/channel コマンドの置き換えのために必要な post コマンドの目的 host の決定と port の割り当てを行う．Resource Information から同一の channel 名を持つ catch コマンドの chunk を確認し, 適切な目的 host を決定する．また, 同時に catch の待ち受けを行う port の決定も行う.port の決定は, リソーススクリプトの port の begin から順に使用する.port が end まで使用された場合は, エラーを返す.

4.5.4 New Script Generator

New Script Generator では, サービススクリプトを置換し, 実行スクリプトを生成する. 置換される 5 つの命令は,INPUT,OUTPUT,post,catch,chunk である.

- INPUT:INPUT 命令は, 対応する deviceprotocol に置換される.Script Generator では,from 属性を参照し適切な deviceprotocol を参照することを行う.
- OUTPUT:OUTPUT 命令は, 対応する deviceprotocol に置換される.Script Generator では,to 属性を参照し適切な deviceprotocol を参照することを行う.
- post: post コマンドは以下のコマンドに置き換えられる.

```
<setq name="client_socket_name">
  <dgSocket>
    <text>IP address</text>
    <int>port</int>
  </dgSocket>
</setq>
<dgSend>
  <getq name="client_socket_name"/>
  <getq name="INPUT"/>
</dgSend>
<dgClose>
  <getq name="client_socket_name"/>
</dgClose>
```

  すなわち,post コマンドはデータグラムソケットにより実現される,CCDM 上のネクストホップへのデータ転送となる.
- catch: catch コマンドは以下のコマンドに置き換えられる.

```
<setq name="server_socket_name">
  <dgServerSocket>
    <int>port</int>
  </dgServerSocket>
</setq>
<setq name="name">
```

```
    <dgRecv>
      <getq name="server_socket_name"/>
    </dgRecv>
  </setq>
  <dgClose>
    <getq name="server_socket_name"/>
  </dgClose>
```

すなわち,catch コマンドはデータグラムソケットによる post コマンドからのデータ取得となる.

- chunk:chunk 命令が発行されない host の場合,chunk 以下のコマンドは削除される.

第 5 章

CCDM の検証と実験環境

5.1　キャンパスネットワークを対象とした実験環境の構築

本研究ではネットワーク管理アーキテクチャと管理スクリプトの提案により, ネットワークの管理コストの減少や耐障害性を増すことを目的としている．章 1 章で述べたように, 対象としているネットワークは, ある建物内のネットワークやキャンパスネットワークである．CCDM のプロトタイプ実装の動作検証や評価を行うためには模擬的な実験環境が必要である．実験環境となるネットワークは小規模なものとなるが, 模倣するネットワークは規模を問わず現実に促しているほど良い．そこで実験環境の構築に当たっては東京大学工学部 2 号館のネットワークと江崎研究室を参考にした．東京大学工学部 2

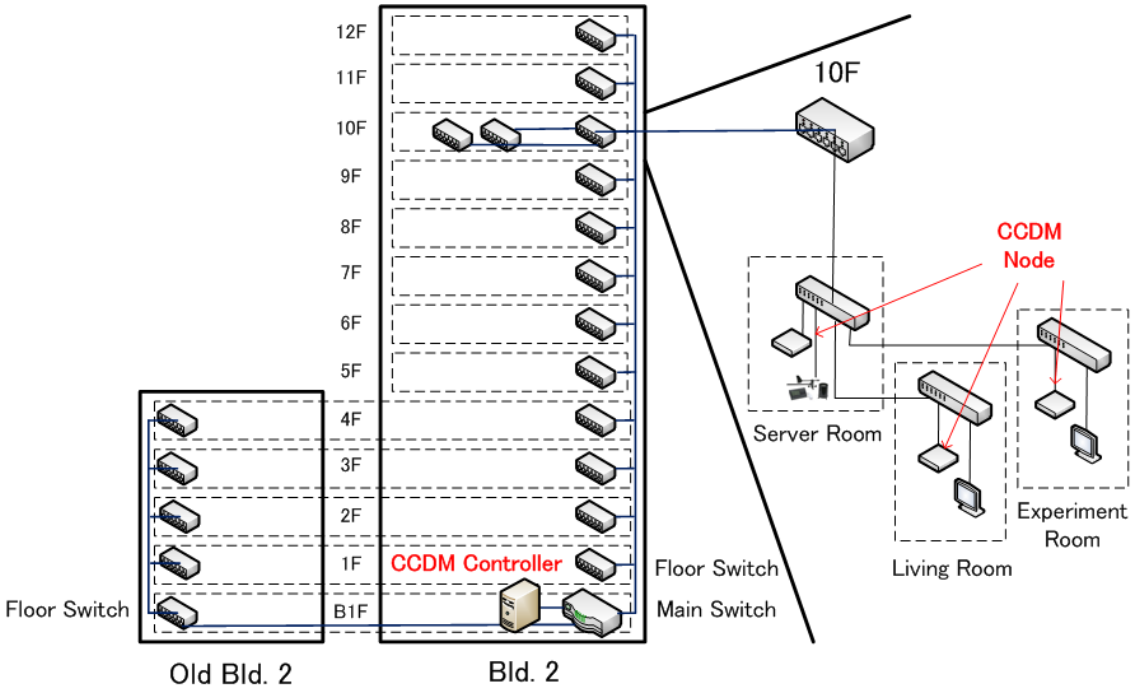

図 5.1　工学部 2 号館ネットワークの概要

号館のネットワークの概要と CCDM システムの導入イメージを示したものが図 5.1 である.

この図はレイヤ 2 のネットワーク図を考慮して作成している. 現在, 工学部 2 号館には基幹スイッチが地下 1 階に存在し, その基幹スイッチから各階のフロアスイッチに光ファイバが伸びている. 各フロアスイッチからはフロア内の各部屋の情報コンセントに光ファイバや Unshielded Twist Pair(UTP) ケーブルで接続されている.

導入される CCDM の構成要素は赤で示している. 建物の CCDM システムを統括する CCDM コントローラは基幹スイッチの存在する弱電室への設置を想定する. CCDM コントローラの指令を受けスクリプトを実行する CCDM ノードは, 各階の各部屋に設置される. 特に, 今回は 2 号館 10 階の江崎研究室内で CCDM のサービスが提供される状況を想定し, 江崎研究室のサーバ室, 居室, 実験室に CCDM ノードを配置する状況をシナリオとした.2 号館 10 階江崎研究室のレイヤー 2 の図も図 5.1 に拡大する形で示している. 江崎研の基幹スイッチや各種サーバが存在するサーバ室には CCDM ノードと気象センサが設置される. サーバ室の基幹スイッチの下流につながるネットワークは居室と実験室のネットワークである. 居室と実験室のスイッチはサーバ室の基幹スイッチの下流のスイッチとして直接接続されている. 居室と実験室にもサーバ室と同様に CCDM ノードとディスプレイ, 気象センサが設置されているとする.

以上で述べたネットワーク環境を模すよう作成した実験環境が図 5.2 になる. 今回の実験では,CCDM システムの導入によりネットワークの管理コストが減少すること, 耐障害性が増すこと, スクリプト実行によるオーバヘッドの測定を行い検証することを目的とした. そのため, 評価に影響の少ないレイヤ 3 のトポロジは無視することとし, 実験ネットワークのレイヤ 3 は 192.168.1.0/24 のフラットなネットワークとした. 耐障害性が増すことを示す必要があるためレイヤ 2 はより忠実に再現している. 地下 1 階の弱電室に設置された CCDM コントローラから 10 階の江崎研サーバ室のスイッチまで, 実験環境では 1 本の UTP で接続される. 地下 1 階と 10 階江崎研サーバ室間のネットワークは江崎研内のネットワークに比べ遅延があることが考えられる. 実験の結果と評価では遅延についても考慮する必要がある. サー

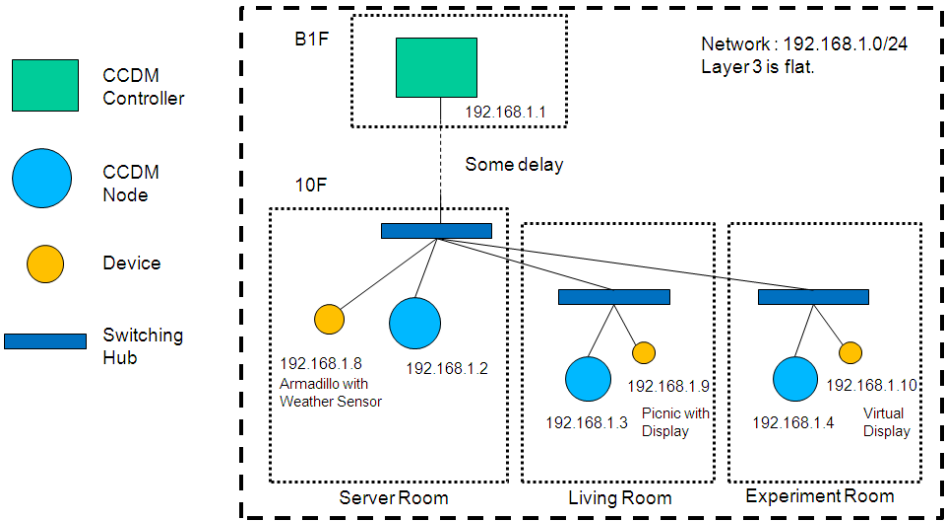

図 5.2　本研究で用意した実験環境

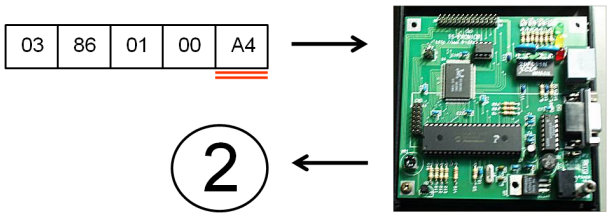

図 5.3　デバイス (PICNIC) でのディスプレイ表示命令の発行

バ室, 居室, 実験室の 3 つのレイヤ 2 のネットワークを再現するためスイッチングハブを 3 台用意し, 図 5.2 のように配置した. 3 台のスイッチングハブの配下に CCDM 実行環境をインストールした Linux PC3 台を配置した. サーバ室を再現しているスイッチの配下には気象センサ WM918[35] に接続した小型 CPU ボード Armadillo[36], 居室を再現しているスイッチの配下には液晶ディスプレイに接続した Picnic デバイス, 実験室を再現しているスイッチの配下にはノート PC を配置した.

以下は図 5.2 で描かれている要素の説明である.

- CCDM Controller(192.168.1.1):実験環境における CCDM コントローラ
- CCDM Node(192.168.1.2 192.168.1.3 192.168.1.4):実験環境における CCDM ノード. サーバ室, 居室, 実験室に計 3 台を設置.
- Armadillo with Weather Sensor(192.168.1.8):気象センサを接続した小型 CPU ボード Armadillo.UDP 上の 1 オクテット命令を発行することで, 対応する気象観測項目を文字列で応答する.
- PICNIC with Display(192.168.1.9):UDP により受け取ったデータを表示するディスプレイ.
- Virtual Display(192.168.1.10):ノード PC 上に実装した仮想ディスプレイデバイス.
- Switching Hub:レイヤ 2 のネットワークを再現する. 計 3 台を設置.

5.2　CCDM により実現する 7 つのアプリケーション

CCDM システムの動作検証と評価を行うために CCDM で動作する 7 つのアプリケーションを考え用意した. 本節では 7 つのアプリケーションの要素と動作概要について説明する.

5.2.1　Weather Display Service

Weather Display Service は気象センサに接続した Arimadillo with Weather Sensor (Armadillo WS) から温度データを取得し,PICNIC に接続した液晶ディスプレイに温度データを表示する. Armadillo WS は UDP 上の 1 オクテット命令を発行することで, 対応する気象観測項目を文字列で応答する. 具体的には, 温度 (0x10), 湿度 (0x11), 風向 (0x12), 風速 (0x13), 雨量 (0x14), 気圧 (0x16) となっている. 例えば, 湿度の場合 [0x11] を発行すると,[0x37 0x38 0x2E 0x34](=74.8) などと ASCII 表現での応答を返す.UDP の待ち受けポートは 9999 であり, 実験環境では 192.168.1.8 のアドレスがついている.

PICNIC では UDP 上のプロトコルによりデータを受け取り, 液晶ディスプレイに表示するインター

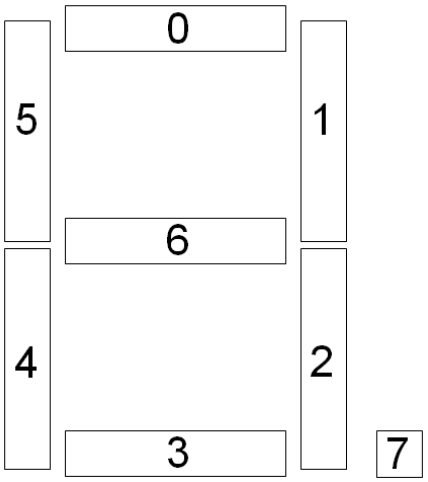

図 5.4　ディスプレイの bit と表示の対応

表 5.1　ディスプレイの bit の対応表

figure	bit	hexadecimal number
1	11111001	0xF9
2	10100100	0xA4
3	10110000	0xB0
4	10011001	0x99
5	10010010	0x92
6	10000011	0x83
7	11011000	0xD8
8	10000000	0x80
9	10011000	0x98
0	11000000	0xC0
.	01111111	0x7F

フェースが提供されている. 例えば, 図 5.3 のような [0x03 0x86 0x01 0x00 0xA4] というリクエストを送出すると, ディスプレイは「2」を表示する．ディスプレイに表示される数字に直接の関係があるのは 5byte 目で, 図 5.4 のように対応した bit が立っている線がディスプレイ上で光る. 表 5.1 には入力 bit 列に対して表示される数字の一覧を示す．UDP の待ち受けポートは 10001 であり, 実験環境では 192.168.1.9 のアドレスがついている.

Weather Display Service の動作概要を図 5.5 に示す．Weather Display Service の動作の流れは次の (1)〜(5) である．(1)CCDM Node A が Armadillo WS に温度データの取得リクエスト (UDP プロトコルのポート 9999. データグラムの値は [0x10]) を送出する．(2) データの取得リクエストを受

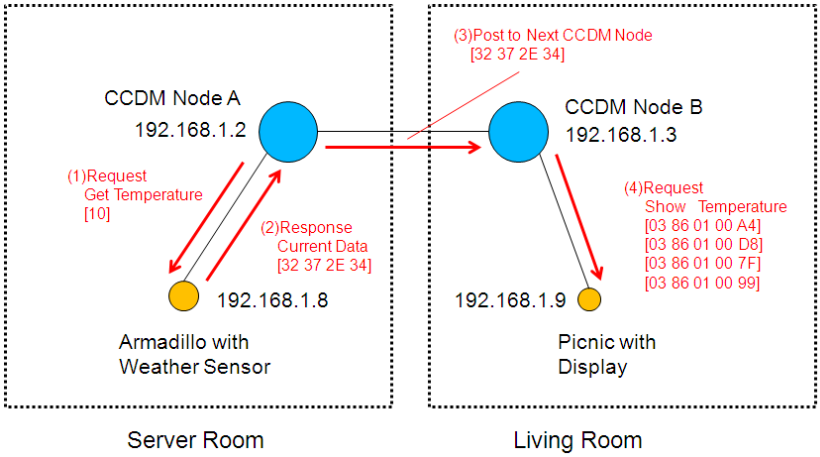

図 5.5　Weather Display Service の動作概要

け取った Armadillo WS はその時の温度データを返す. データ形式は ASCII 表現で [0x32 0x37 0x2E 0x34](=27.4) である.(3)Armadillo WS から温度データを受け取った CCDM Node A はそのデータを CCDM Node B に転送する.(4)CCDM Node A から温度データ (ASCII 表現) を受け取った CCDM Node B は Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの.27.4 は [0x03 0x86 0x01 0x00 0xA4] [0x03 0x86 0x01 0x00 0xD8] [0x03 0x86 0x01 0x00 0x7F] [0x03 0x86 0x01 0x00 ox99] に相当する.) に変換して, データを送出する.

以上の一連の流れにより, サーバ室で取得した温度データを居室のディスプレイに表示するサービスが Weather Display Service である. このサービスを記述したサービススクリプトを付録 B に添付する.

5.2.2　Multicast Data Delivery Service

Multicast Data Delivery Service は 1 つのセンサから取得したデータを post/catch 命令のマルチキャスト配送の仕組みを用いて, 複数のディスプレイで表示する. 温度データを取得するセンサは第 5.2.1 節で用いた Armadillo WS を用いる. 取得した温度データを表示するディスプレイとして, 第 5.2.1 節で用いた PICNIC with Display と Virtual Display を用いる. Virtual Display は JAVA で実装されたデータ表示プログラムである.UDP プロトコルのポート 30000 番でデータを待ち受け, 受け取ったデータを ASCII 表現の文字列で解釈し, ターミナルに出力する.

Multicast Data Delivery サービスの動作概要を図 5.6 に示す. Multicast Data Delivery の動作の流れは次の (1)〜(4) である. (1)CCDM Node A が Armadillo WS に温度データの取得リクエスト (UDP プロトコルのポート 9999. データグラムの値は [0x10]) を送出する. (2) データの取得リクエストを受け取った Armadillo WS はその時の温度データを返す. データ形式は ASCII 表現で [0x32 0x37 0x2E 0x34](=27.4) である.(3)(3)'Armadillo WS から温度データを受け取った CCDM Node A はそのデータを CCDM Node B と CCDM Node C に転送する.(4)CCDM Node A から温度データ (ASCII 表現) を受け取った CCDM Node B は Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの.27.4 は [0x03 0x86 0x01 0x00 0xA4] [0x03 0x86 0x01 0x00 0xD8] [0x03 0x86 0x01 0x00 0x7F]

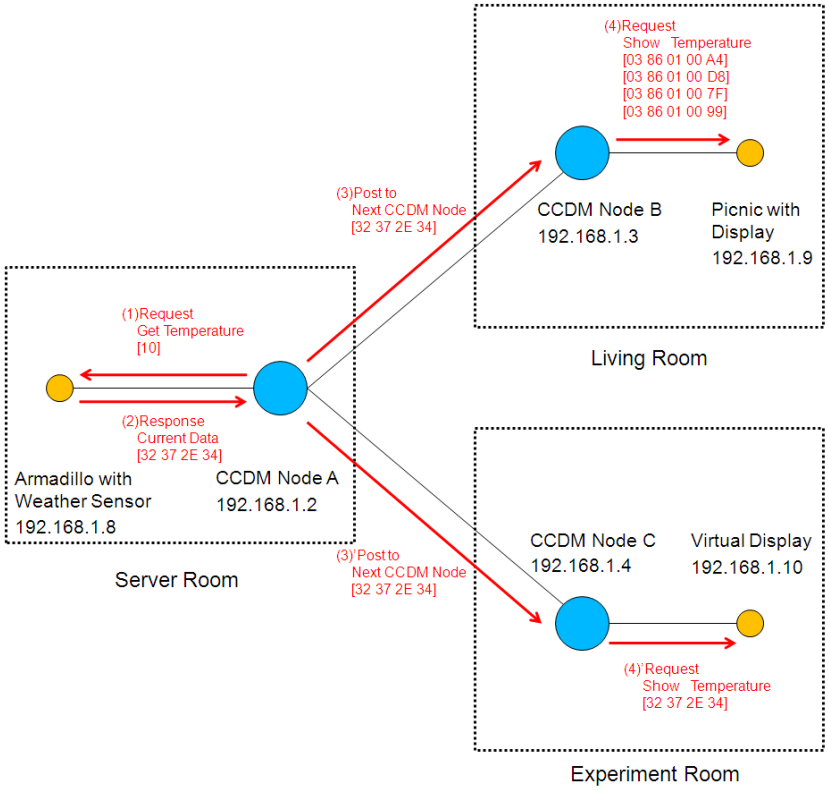

図 5.6　Multicast Data Delivery の動作概要

[0x03 0x86 0x01 0x00 ox99] に相当する.) に変換して, データを送出する. (4)'CCDM Node A から温度データ (ASCII 表現) を受け取った CCDM Node C は Virtual Display が解釈可能な型でデータを送出する. この場合, 受け取ったデータが ASCII 文字列なので, そのまま転送する.

このように,Multicast Data Delivery Service ではデータの表示先の受信プロトコルが異なる場合にもそのプロトコルの差を吸収して扱えることを示している. この例では,CCDM ノード B と CCDM ノード C がそれぞれ出力先のプロトコルに応じた形式でデータを出力している.Multicast Data Delivery を実行するサービススクリプトを付録 C に添付する.

5.2.3　Weather Threshold Notification Service

Weather Threshold Notification Service は観測している温度データがある閾値以上になった場合のみ, そのデータをディスプレイに表示するサービスである. 取得したデータの出力機器として第 5.2.1 節で用いた Virtual Display を用いる. 気象データを取得するセンサとして PICNIC に接続した温度センサ (Sensor with PICNIC) を用いる. このセンサは図 5.7 のようなリクエストを発行することで得られる応答結果の中から特定のオクテット列を取り出し, 計算式 X*128+Y/2 を適用することで温度 (°C) として取り出すことができる.

Weather Threshold Notification Service の動作概要を図 5.8 に示す. Weather Threshold Notifica-

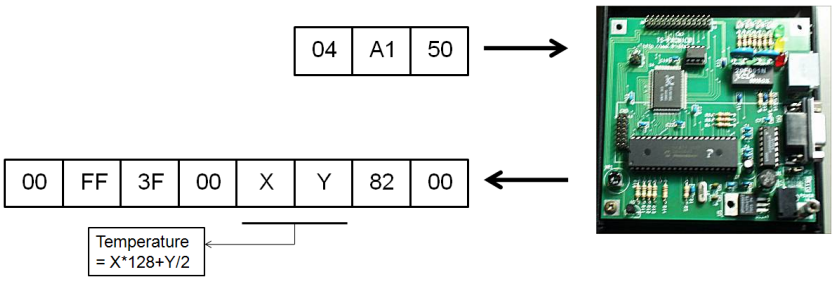

図 5.7　PICNIC からの温度取得操作

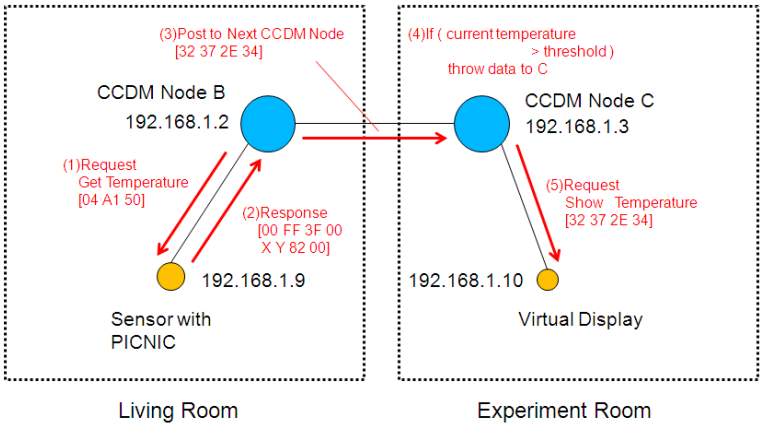

図 5.8　Weather Threshold Notification の動作概要

tion Service の動作の流れは次の (1)〜(5) である．(1)CCDM Node B が Sensor with PICNIC に温度データの取得リクエスト (UDP プロトコルのポート 10001. データグラムの値は [0x04 0xA1 0x50]) を送出する．(2) データの取得リクエストを受け取った Sensor with PICNIC はその時の温度データを返す. データ形式は PICNIC 特有のもので [0x00 0xFF 0x3F 0x00 0xX 0xY 0x82 0x00] である. このバイト列のうち,X*128+Y/2 を計算したものが温度データとなる．(3)PICNIC with Sensor から温度データを受け取った CCDM Node B はそのデータを CCDM Node C に転送する．(4) 取得したデータがあらかじめ設定された閾値よりも大きいかどうか判断する．真ならば (5) の操作を行い, 偽ならば何もしない．(5)CCDM Node B から温度データ (ASCII 表現) を受け取った CCDM Node C は可能な型でデータを送出する. この場合, 受け取ったデータが ASCII 文字列なので, そのまま転送する.

以上の一連の動作により,Weather Threshold Notification Service は閾値以上の温度データのみをディスプレイに表示する. このサービスを記述したスクリプトを付録 D に添付する.

5.2.4　Data Aggregation Service

Data Aggregation Service は単一のソースのセンサからのデータを集約 (3 回の平均を算出) して, 出力機器に転送する. この操作により CCDM ノード間でのデータ転送量を減少することが期待できる．取

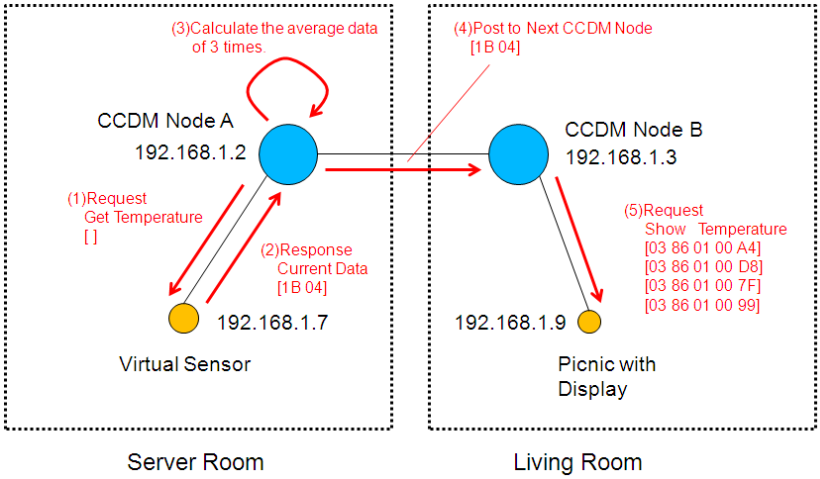

図 5.9　Data Aggregation Service の動作概要

得したデータを出力する機器として第 5.2.1 節で用いた PICNIC with Display を用いる．気象データを出力するセンサとして Virtual Sensor を用いる．Virtual Sensor は JAVA で実装された仮想的なセンサデータ出力プログラムである．データの生成はランダムウォークアルゴリズムにより行う．

$$T_n = T_0 + \sum_{j=0}^{n} Z_j$$

$$Z_j = \{-0.5, 0.5\}$$

ランダムウォークアルゴリズムにより,1000 ミリ秒に 1 回, 内部に保持しているデータを更新する．UDP プロトコルのポート 9998 番で待ち受けており, リクエストを受け付けるとその時の温度データを返す．応答するデータは 2byte で,1byte 目に温度の整数部の値, 2byte 目に温度の少数部の値を格納している．例えば,[0x1B 0x04] は 27.4 度を示す．

Data Aggregation Service の動作概要を図 5.9 に示す．Data Aggregation Service の動作の流れは次の (1)〜(5) である．(1)CCDM Node A が Virtual Sensor に温度データの取得リクエスト (UDP プロトコルのポート 9998) を送出する．(2) データの取得リクエストを受け取った Virtual Sensor はその時の温度データを返す.27.4 度の場合, 値は [0x1B 0x04] である．(3)「(1)(2)」を 3 回繰り返し,3 回の温度データの平均を算出する．(4)CCDM Node A は算出した平均の値を CCDM Node B に転送する．(5)CCDM Node A から温度データを受け取った CCDM Node B は Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの) に変換して, データを送出する．

以上の一連の動作により Data Aggregation Service はセンサデータを集約して出力装置に転送する．このサービスを記述したスクリプトを付録 E に添付する．

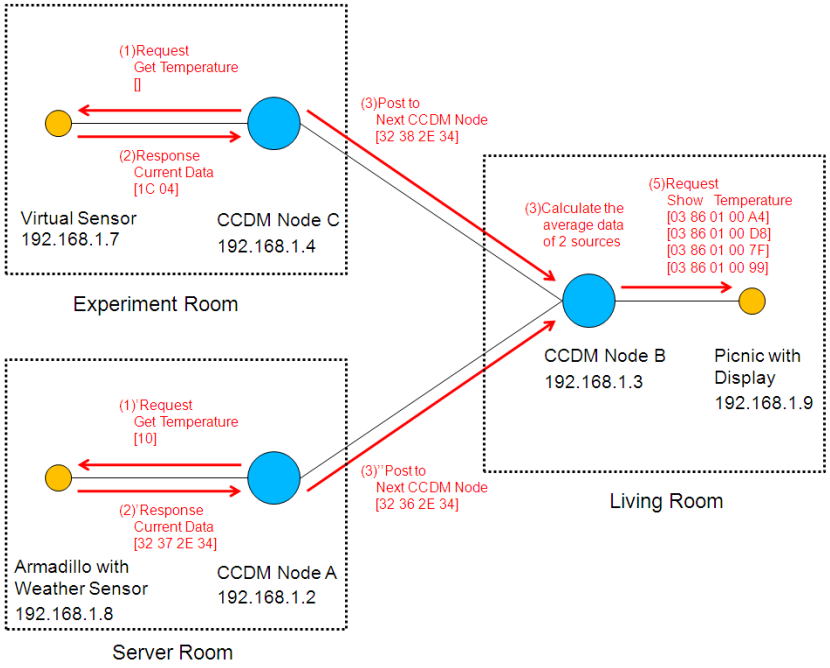

図 5.10　Several Source Data Aggregation Service の動作概要

5.2.5　Several Source Data Aggregation Service

Several Source Data Aggregation Service は 2 地点のセンサからのデータを集約 (平均を算出して) して, 出力機器に転送する. 温度データを出力するセンサとして第 5.2.1 節で用いた Armadillo WS と第 5.2.4 節で用いた Virtual Sensor を用いる. 取得したデータを出力する機器として第 5.2.1 節で用いた PICNIC with Display を用いる.

Several Source Data Aggregation Service の動作概要を図 5.10 に示す. Several Source Data Aggregation Service の動作の流れは次の (1)〜(5) である. (1)CCDM Node C が Virtual Sensor に温度データの取得リクエスト (UDP プロトコルのポート 9998) を送出する. (2) データの取得リクエストを受け取った Virtual Sensor はその時の温度データを返す.27.4 度の場合, 値は [0x1B 0x04] である. (3)CCDM Node C は CCDM Node B にデータを転送する. (1)'CCDM Node A が Armadillo WS に温度データの取得リクエスト (UDP プロトコルのポート 9999. データグラムの値は [0x10]) を送出する. (2)' データの取得リクエストを受け取った Armadillo WS はその時の温度データを返す. データ形式は ASCII 表現で [0x32 0x37 0x2E 0x34](=27.4) である. (3)'CCDM Node A は CCDM Node B にデータを転送する. (4)CCDM Node A,C から温度データを受け取った CCDM Node B は 2 つの値の平均を計算する. (5)CCDM Node B は Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの) に変換して, データを送出する.

以上の一連の動作により Several Source Data Aggregation Service はセンサデータを集約して出力装置に転送する. このサービスを記述したスクリプトを付録 F に添付する.

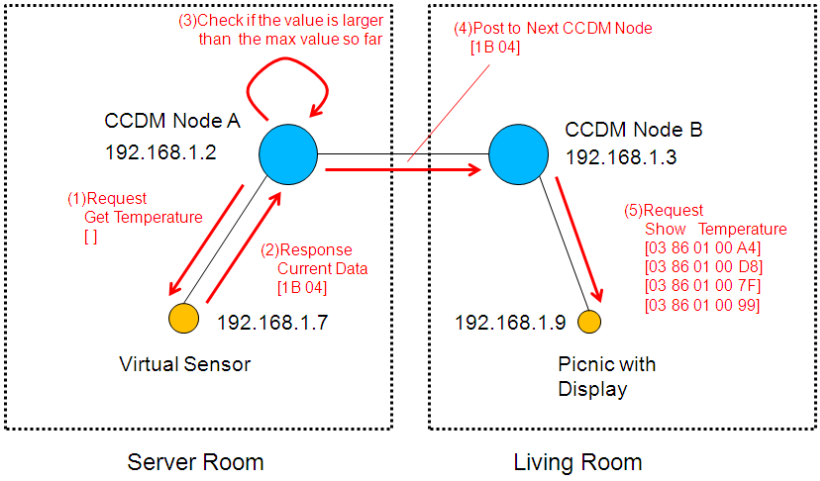

図 5.11　Largest Value Update Notification Service の動作概要

5.2.6　Largest Value Update Notification Service

Largest Value Update Notification Service はセンサ機器が出力する値が最大値を更新した場合にのみ出力機器にデータを転送する. この操作により CCDM ノード間でのデータ転送量を減少することが期待できる. 気象データを出力するセンサとして第 5.2.4 節で用いた Virtual Sensor を用いる. 取得したデータを出力する機器として第 5.2.1 節で用いた PICNIC with Display を用いる.

Largest Value Update Notification Service の動作概要を図 5.11 に示す. Largest Value Update Notification Service の動作の流れは次の (1)〜(5) である. (1)CCDM Node A が Virtual Sensor に温度データの取得リクエスト (UDP プロトコル, ポート 9998) を送出する. (2) データの取得リクエストを受け取った Virtual Sensor はその時の温度データを返す.27.4 度の場合, 値は [0x1B 0x04] である. (3) 取得した温度データが最大値かどうかを判断する. 前回までの値よりも大きければ (4) に進む. 同じであれば (1) と (2) を再び行う. (4)CCDM Node A は温度データを CCDM Node B に転送する. (5)CCDM Node A から温度データを受け取った CCDM Node B は,Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの) に変換して, データを送出する.

以上の一連の動作により Largest Value Update Notification Service は更新があった値のみを出力装置に転送する. このサービスを記述したスクリプトを付録 G に添付する.

5.2.7　Updated Data Delivery Service

Updated Data Delivery Service はセンサ機器が出力する値に変更があった場合にのみ出力機器にデータを転送する. この操作により CCDM ノード間でのデータ転送量を減少することが期待できる. 温度データを出力するセンサとして第 5.2.4 節で用いた Virtual Sensor を用いる. 取得したデータを出力する機器として第 5.2.1 節で用いた PICNIC with Display を用いる.

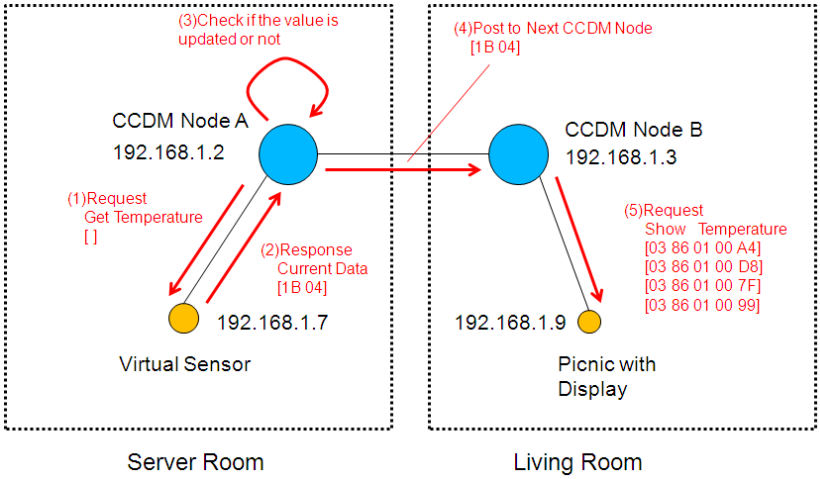

図 5.12　Updated Data Delivery Service の動作概要

Largest Value Update Notification Service の動作概要を図 5.12 に示す．Largest Value Update Notification Service の動作の流れは次の (1)〜(5) である．(1)CCDM Node A が Virtual Sensor に温度データの取得リクエスト (UDP プロトコル, ポート 9998) を送出する．(2) データの取得リクエストを受け取った Virtual Sensor はその時の温度データを返す.27.4 度の場合, 値は [0x1B 0x04] である．(3) 取得した温度データが前回の値と異なるかどうかを判断する. 異なっていれば (4) に進む. 同じであれば (1) と (2) を再び行う．(4)CCDM Node A は温度データを CCDM Node B に転送する．(5)CCDM Node A から温度データを受け取った CCDM Node B は Picnic with Display が解釈可能なデータ型 (表 5.1 に示したもの) に変換して, データを送出する．

以上の一連の動作により Updated Data Delivery Service は更新があった値のみを出力装置に転送する. このサービスを記述したスクリプトを付録 H に添付する．

第 6 章

CCDM のシステム評価

6.1 管理コストの減少に関する考察

ネットワークの管理コストを減少することが本研究の大きな目標の 1 つである．しかし, ネットワーク管理者にとっての管理コストの評価を客観的に行うことは難しい．いくつかのネットワーク管理に関する研究では, 提案システムの導入によるオーバヘッドやスケーラビリティに関して実験を行い, それを評価としている [11, 14, 18]．本研究でも同様に, 第 6.3 節で CCDM システムの導入によるオーバヘッドを観測し, 評価を行う．しかし, この方法だけでは管理者にとってのコストを評価することができない．本研究では管理コストの評価のため,

1. 管理者がサービスを提供するために行う命令 (スクリプト) の記述量 (節 6.1.1)
2. ネットワークやデバイスの情報に変更があった場合に必要となる命令 (スクリプト) の変更箇所 (節 6.1.2)

による評価を行う．一般に, コンピュータ言語の比較, 評価は難しい [37] が, その中でプログラムの記述量によりプログラミング言語を評価する方法は多く取られている [38, 39]．管理コストの評価に当たっては, このようなプログラミング言語の比較手法を参考にした．

6.1.1 サービス提供のためのスクリプト記述量の比較

管理コストの評価のために, サービスの提供の際に必要となるサービススクリプトの記述量の比較を行った. 図 6.1 に結果を示す．横軸の項目は比較を行ったサービススクリプトの名称, 縦軸はスクリプトに記述された命令の行数を示す. 青い棒グラフが CCDM システムを用いた場合のスクリプト記述量, 赤い棒グラフが CCDM システムを用いず,CCDM ノードで実行されるスクリプトを全て管理者が書いた場合の記述量である．表 6.1 に特徴的な結果をまとめた．

7 つのアプリケーション例の CCDM システムを用いた場合の総記述量は 1074(リソーススクリプトも含めると 1476) 行,CCDM システムを用いない場合の総記述量は 4539 行である．CCDM システムを用いた場合, 従来の 23.7% の記述量でサービスを提供できることになる．

また, サンプルスクリプトのうち従来の記述法と最も差が少ないサービスである Weather Threshold Notification Service と最も差が大きいサービスである Weather Display Service について考察す

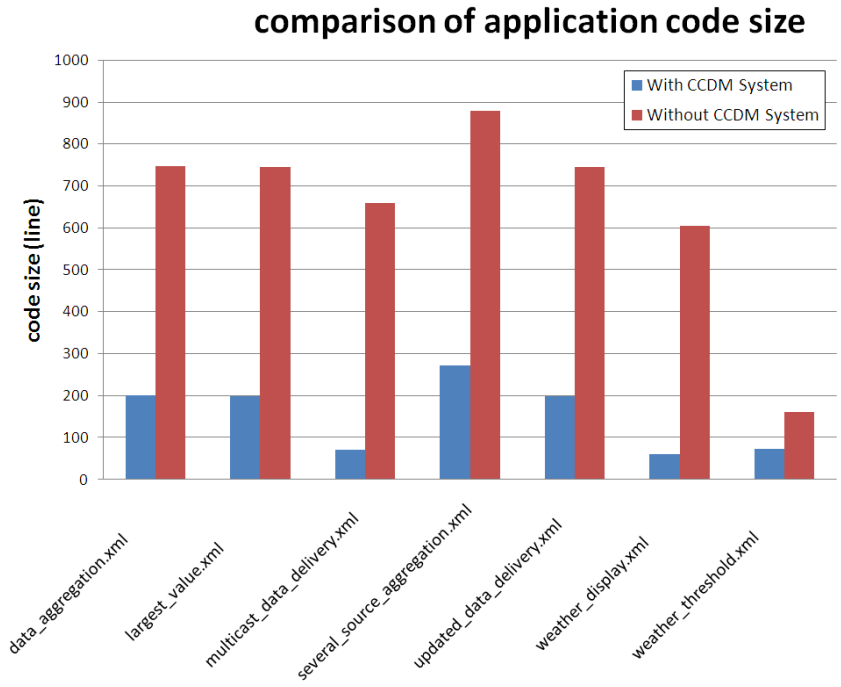

図 6.1　サービス提供のためのスクリプト記述量

表 6.1　サービス提供のためのスクリプト記述量の比較

	with CCDM	without CCDM
総記述量	1074(1476)	4539
記述量の平均	153	648
weather threshold notification	74	162
weather display	60	604

る．CCDM システムを用いた場合のスクリプト記述量は従来の方法に比べ Weather Threshold Notification Service で 45.7%, Weather Display Service で 9.9% である．サービス自体の記述が少なくデバイスとの通信プロトコルに関する記述部の占める割合が大きい Weather Display Service は,CCDM システム導入によるスクリプト記述量の削減幅が大きく, サービス自体の記述が多くデバイスとの通信プロトコルに関する記述部の占める割合の小さい Weather Threshold Notification Service は CCDM システム導入による記述量削減幅が小さい．すなわち, サービスにおけるデバイスとの通信プロトコルの割合が大きいシステムであればあるほど,CCDM システムの導入の効果が大きいことがいえる．

また, 図 6.2 にスクリプトの数と記述量の比較を行ったグラフを示す. 横軸はサービスを記述したスクリプトの数, 縦軸はスクリプトの記述量である．CCDM システムを用いた場合にスクリプト数 0 で

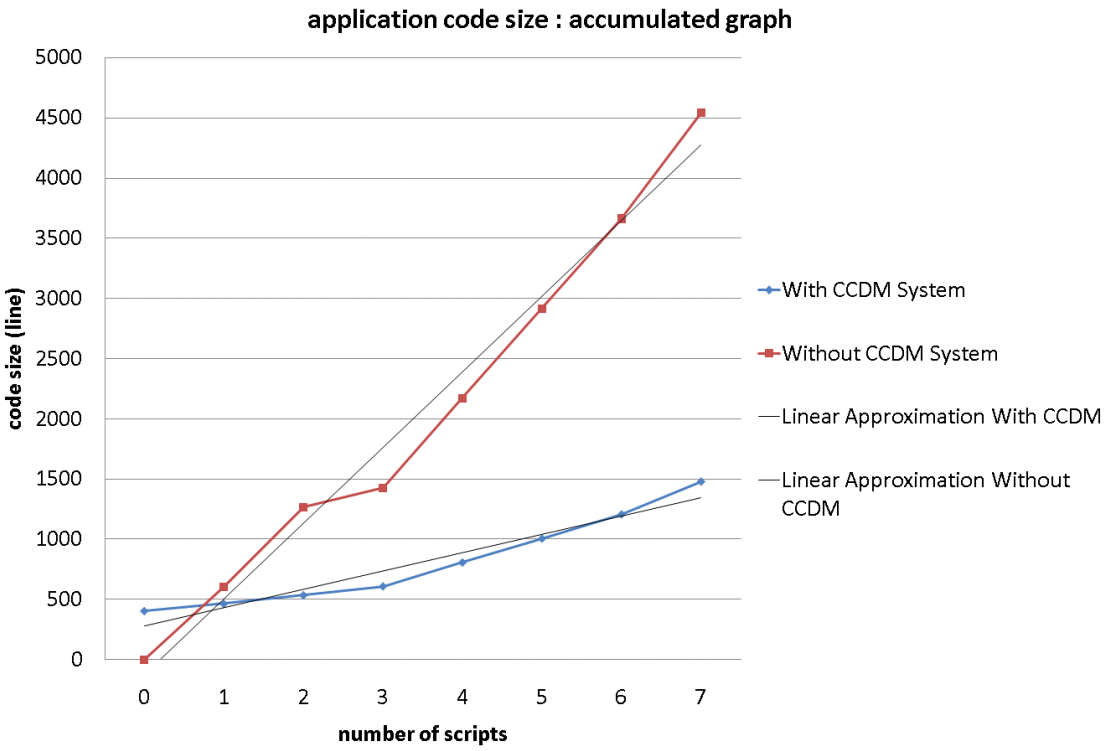

図 6.2　サービス数によるスクリプト記述量の比較

記述量が約 400 あるのは, リソーススクリプトの記述が必要なためである. 線形近似した結果の傾きは CCDM システムを用いない場合が用いる場合に比べて約 1.8 倍大きい. これは,CCDM システムを用いた場合, デバイスのプロトコルやホストの情報についての再利用性があるためと考えられる. このように,CCDM システムで動くサービスの数が多ければ多いほど,CCDM システムを使用しない場合に比べて優位であることがわかる.

6.1.2　ネットワーク状態の変更による, 変更箇所数の比較

ネットワーク管理においては,1 つのネットワークの情報の変更にために複数の機器の設定の変更が必要となる場合がある. この作業はネットワーク管理者の手間を要するものであり, 間違いを引き起こし易い. この問題の解決のために, 例えば ARUBA Networks では, 同一の設定で動作する複数の無線アクセスポイントの設定変更を統合的に行うシステムを提供している [40]. ネットワーク状態の変更による設定変更が CCDM システムの導入により容易になること, すなわち変更すべきコンフィギュレーションの箇所が減ることを検証した.

図 6.3 は, ネットワークの状態により必要となるスクリプトの変更箇所について検証したものである. 検証は Host A,B,C の IP アドレスが変更になった場合,Weather Sensor Device, Display Device のデバイス特有のプロトコルが変更になった場合の計 5 個のネットワーク状態の変更 (イベント) を想定し

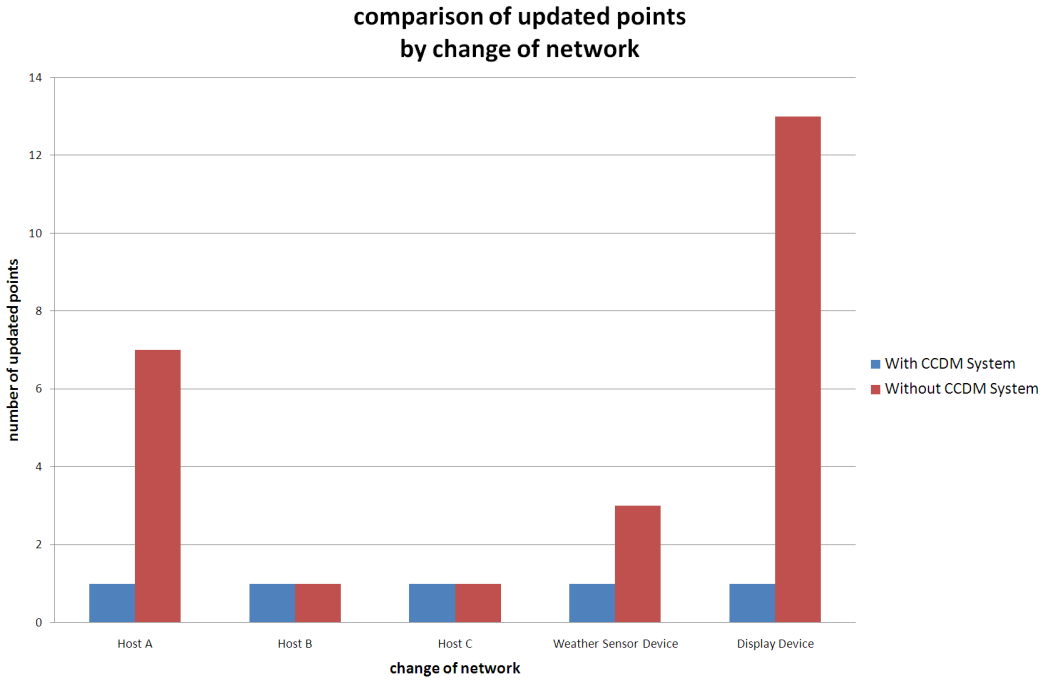

図 6.3　ネットワーク (リソース) 状態の変更によるスクリプトの変更箇所

て行った. 図の横軸はイベントの名称を, 縦軸はそのイベントで必要となった変更箇所数を示す. 変更箇所数の定義は行数ではない. 例えば, ホストの IP アドレスが変わった場合であれば, その IP アドレスが使用されている回数であり, デバイスのプロトコルで 2 バイト目と 3 バイト目を足すという作業が 2 バイト目のみ参照するという変更であれば, その命令が行われている回数である. 青い棒グラフが CCDM システムを用いた場合の変更箇所数で赤い棒グラフが CCDM システムを用いない場合の変更箇所数である.

単純に 5 つのイベント全てが起きたとした場合,CCDM システムを用いた場合の変更箇所数は 5 であり,CCDM システムを用いない場合の変更箇所数は 16 である. つまり, 変更箇所数は 31% に削減できている.

5 つのイベントの比較結果にはばらつきが大きい. 例えば,Host B や C の変更の場合には修正が必要となるスクリプトの箇所は 1 つだが,Display Device の変更の場合は 13 箇所である. これは,7 つのアプリケーション例で使用されているそれぞれの情報の使用頻度に違いがあるためである. 例えば,Display Device は 7 つのアプリケーションで多く使用されているため, そのプロトコルに変更があった場合には多くの実行スクリプトの変更が必要になる. 例えばこの場合の変更箇所数は 7.7% に削減されている. 一方,Host B や C の IP アドレスの情報は,catch 命令が記述されている場合だけなので少なくなる.

図 6.4 にサービススクリプト数の違いによる変更箇所数の変化を示した. 横軸は CCDM システムで使用されているとしたサービススクリプトの数を示す. 縦軸は変更箇所数である. 変更箇所数はそれぞれのサービス提供中に Display Device のプロトコルに変更があった場合とした. 青い線のグラフが CCDM

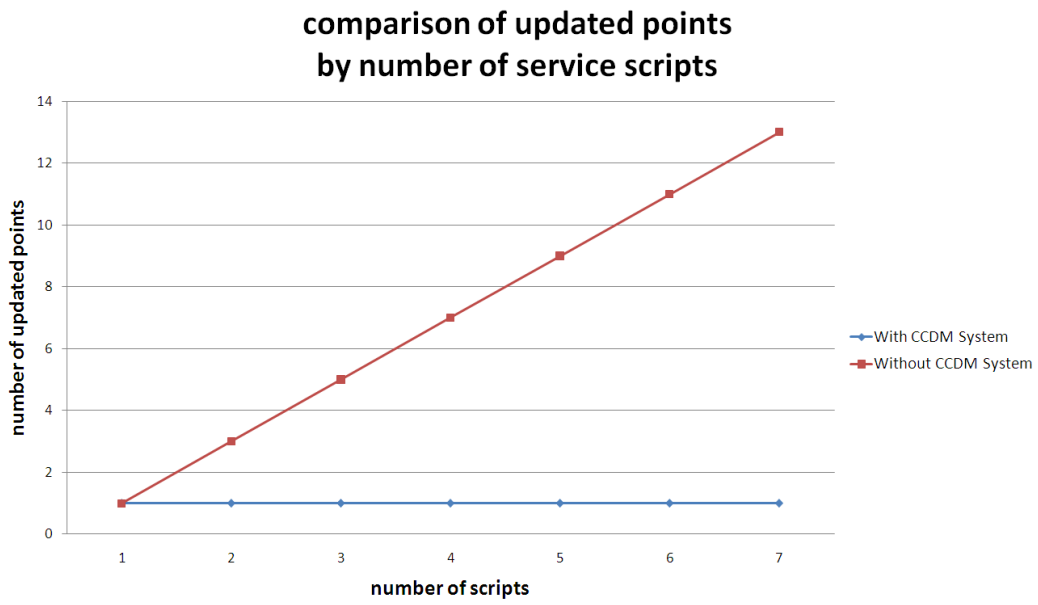

図 6.4　サービススクリプトの数によるスクリプトの変更箇所の違い

システムを用いた場合の変更箇所数で, 赤い線のグラフが CCDM システムを用いない場合の変更箇所数である.

CCDM システムを用いた場合, デバイスの情報の変更はリソーススクリプトのみでサービススクリプトの数によらず 1 箇所である. 一方,CCDM システムを用いない場合, デバイスの変更を逐一実行スクリプトに反映する必要があるので提供するサービスに比例して変更箇所も増える. このような性質から CCDM はサービスの提供数について非常に規模拡張性があることがいえる. サービスの数が少ない場合よりもサービスの数が多い場合の方が CCDM システムが有利となる. これはスクリプト生成のメカニズムをサービスを記述するサービススクリプトとネットワークの情報やデバイスのプロトコルについて記述したリソーススクリプトの結合による方式を採ったことに起因する.

6.2　耐障害性に対する考察

本節では,CCDM システムのネットワークの耐障害性について述べる. 本研究でのネットワーク障害はノード間でネットワークレイヤでの到達性がなくなることと定義する. ネットワークレイヤでの到達性が失われる原因は通常,

1. ネットワークレイヤでの問題. すなわちルータのミスコンフィギュレーションなどによりパケットのルーティングに問題が生じる場合.
2. スイッチのポート不良やルータモジュールの不良,UTP の物理的な断絶などによって生じる物理, リンクレイヤに生じる問題.

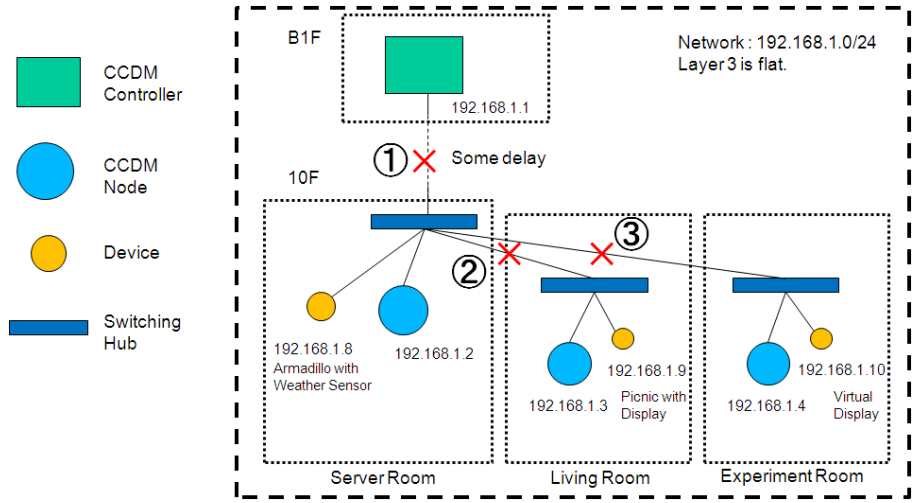

図 6.5　想定するネットワーク障害

表 6.2　ネットワーク障害時のサービス継続性：Weather Display Service

	①	②	③
With CCDM System	○	×	○
Without CCDM System	×	×	○

表 6.3　ネットワーク障害時のサービス継続性：Multicast Data Delivery Service

	①	②	③
With CCDM System	○	△	△
Without CCDM System	×	△	△

の 2 種類に大別できるが, 本研究ではこれらの差を特別には考慮しない. 本研究の障害としては, 主に (2) 物理, リンクレイヤの障害による到達性の喪失を考慮する. これは, 本研究の想定する環境, 実験環境がレイヤ 3 のフラットな小規模なネットワークのためネットワークレイヤの障害を想定しづらいためである.

図 6.5 に本研究で想定するネットワーク障害を示す. ネットワーク障害は①地下 1 階とサーバ室 ②サーバ室と居室 ③サーバ室と実験室 の 3 つのネットワーク間で起きる状況を想定し, 実験環境で検証を行った. 検証に使用したアプリケーションは,Weather Display Service, Multicast Data Delivery Service, Weather Threshold Notification Service の 3 つである.

検証は,3 つのアプリケーションについて,

1. CCDM システムを用いない場合. すなわち,CCDM コントローラが存在する B1F に同様にサービスを提供できるサーバを 1 台設置して, そのサーバ 1 台で全てのサービスの実行を行う状況.

表 6.4　ネットワーク障害時のサービス継続性：Weather Threshold Notification Service

	①	②	③
With CCDM System	◯	×	×
Without CCDM System	×	×	×

2. CCDM システムを用いる場合.

の 2 通りでそれぞれ行った.

検証の結果を表 6.2,6.3,6.4 に示す. 例えば,Weather Display Service は CCDM Node A が温度データを取得して CCDM Node B に転送し表示する. そのため,②CCDM Node A と CCDM Node B の間のネットワーク到達性が無い場合は,CCDM システムを用いる場合でもサービスを提供することはできなくなる. しかし,①のように B1F への到達性がなくなっても CCDM Node A と CCDM Node B の間での到達性は存在するのでサービスを提供することができる. 一方, 中央サーバでサービスの実行も行う場合には, 中央サーバへの到達性がなくなればサービスも継続できなくなる. 同様の結果は Multicast Data Delivery Service, Weather Threshold Notification Service でも見ることができた.

6.3　オーバヘッドに関する考察

6.3.1　スクリプトの生成に必要とする時間

CCDM システムでのサービス提供に必要となるサービススクリプトの結合, 生成にかかる時間を計測する. 以下が計測のコマンドである.

```
$time java org.livee.script.DivideScript weather_display.xml resource.xml
 > /dev/null
```

スクリプトの生成プログラムは以下のように第 1 引数にサービススクリプト, 第 2 引数にリソーススクリプトをとる.

```
$java org.livee.script.DivideScript SERVICE_SCRIPT RESOURCE_SCRIPT
```

図 6.6 にスクリプトの生成時間の計測結果を示す. 横軸は計測に用いたサンプルスクリプトの項目を, 縦軸はスクリプトの生成時間を示す. 単位はミリ秒である. 赤い棒グラフはスクリプトを生成してから各 CCDM ノードに実行スクリプトを配信し終えるまでの時間の計測結果である. 青い棒グラフはスクリプト生成プログラムのうち, スクリプトの配信部分を除いたもの, すなわち純粋にサービススクリプトとリソーススクリプトの解析, 結合から実行スクリプトの生成を行う部分のみの所要時間を計測したものである. 所要時間の計測をそれぞれ 100 回行い, 結果の平均を求めたものをグラフに載せている. 誤差範囲は各計測の標準偏差である.

配信部分を除いたスクリプトの生成時間について, 最も所要時間の少なかったスクリプトは weather_threshold.xml で, 最も所要時間の大きかった several_source_data_aggregation.xml の約

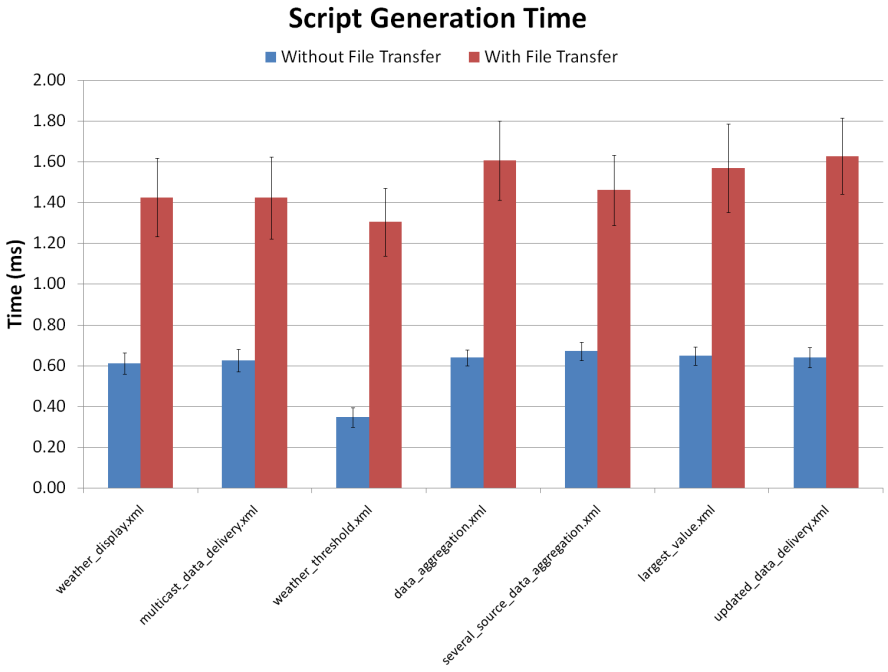

図 6.6　スクリプトの生成に要する時間

52% である．weather_threshold.xml はリソーススクリプトにより置き換わる部分が少ないので, このような結果になったと考えられる．また, 配信部分を除いた場合のスクリプトの生成時間の平均は,0.60 ミリ秒で, 配信部分を考慮に入れた場合も 1.49 ミリ秒である．ネットワークの管理者はスクリプト生成の時間については特に考慮することなく, システムを快適に利用することができると言えよう．

6.3.2　システム導入によるサービス処理の遅延

ネットワークの管理手法の評価としては, 提案システムの導入によるオーバヘッドやスケーラビリティに関する実験を行い, それを評価とすることが多い [11, 14, 18]．本研究でも同様に,CCDM システムの導入によるアプリケーションの実行の遅延を計測し, それが問題になるかどうかを評価する．

CCDM システムの導入によるアプリケーションの処理の実行時間の評価を行うために,(1)CCDM システムを用いた場合のアプリケーション処理実行時間の計測 (2)CCDM システムを用いない場合, すなわちコントローラサーバの存在する中央サーバのみでアプリケーションを実行する場合の処理実行時間の計測 の 2 つの計測を行った．アプリケーションは Armadillo WS からの温度データ取得と PICNIC with Display へのデータ表示である．図 6.7 に CCDM システムを用いた場合の実験環境を示す．CCDM Node A と CCDM Node B にそれぞれスクリプトが配置されている．CCDM Node A は Armadillo WS に温度データのリクエストを送出し, 取得する. 取得したデータは CCDM Node B に転送され,CCDM Node B はデータを Picnic with Display に送出する. この後,CCDM Node B はディスプレイへのデータ送出を行ったことを知らせるデータを CCDM Node A に送出する. この一連の動作

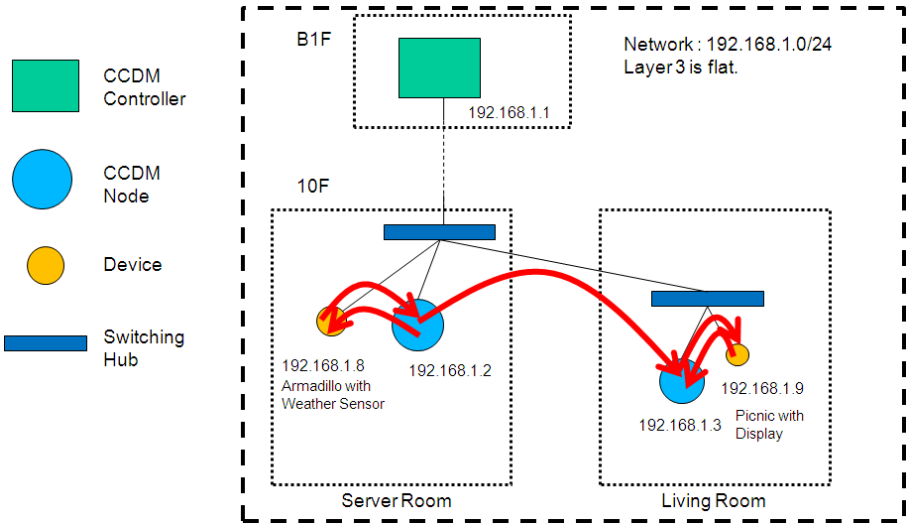

図 6.7　CCDM システムを用いた場合の処理実行時間計測実験

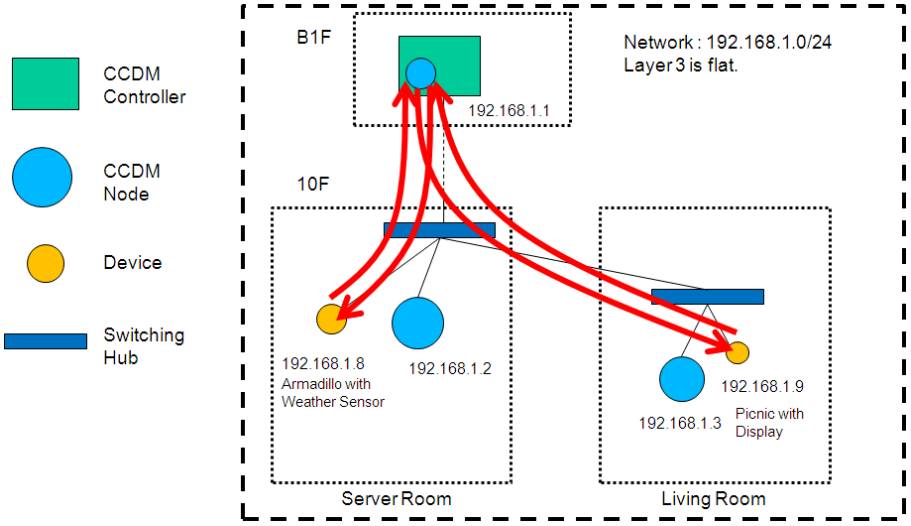

図 6.8　CCDM システムを用いない場合の処理実行時間計測実験

をアプリケーションの処理実行時間として計測した.

図 6.9 は CCDM システムを用いない場合の実験環境である. この環境では CCDM コントローラが CCDM ランタイムを備えスクリプトを実行することとする. 全ての処理は唯一存在するこのサーバによって行われる. CCDM Server は温度データ取得要求を Armadillo WS に送出し, 取得したデータを Picnic with Display に送出する. この動作をアプリケーションの処理実行時間として計測した.

今回の実験環境では,Controller と各 CCDM ノード, デバイスとの間に遅延があることを想定している. 実験ではより現実に近い状況を実現するため, 実際の工学部 2 号館における江崎研から地下 1 階弱電室へのネットワークの距離, すなわち Round Trip Time(RTT) を計測し,Linux の QOS ツール Traffic

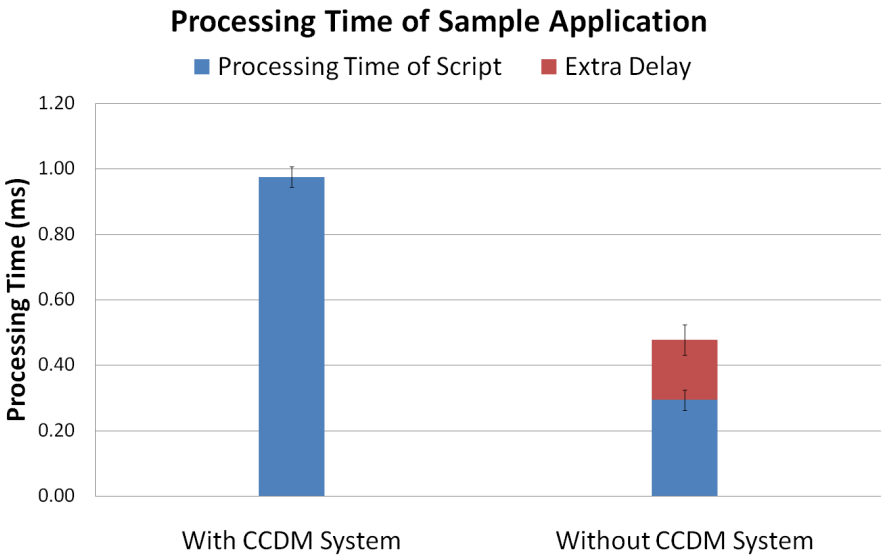

図 6.9　処理実行時間の計測結果

Control(tc)[41] によりネットワークの遅延を再現することを試みた.

まず, 江崎研サーバ室に設置してあるホスト elab-gw.ic.i.u-tokyo.ac.jp から地下 1 階弱電室に設置してあるホスト bld2-mgmt.t.u-tokyo.ac.jp に ping を打つことで Round Trip Time(RTT) を計測した. 試行回数は 300 回である.300 回の計測の平均は 0.417 ミリ秒で, 標準偏差は 0.0461 ミリ秒であった. この値を A とする. 同様に, 実験環境において Controller(192.168.1.1) から CCDM Node A(192.168.1.2) に ping コマンドを 300 回試行し,RTT を計測した.300 回の計測の平均は 0.184 ミリ秒で標準偏差は 0.00828 ミリ秒であった. この値を B とする.

実験環境の Controller-各 CCDM ノード間で追加すべき遅延の値は A-B を 2 で割ったもので 0.117 ミリ秒になるべきである. そこで,tc の以下のコマンドにより Controller の 192.168.1.1 のインターフェースに 0.117 ミリ秒の遅延を加えることを試みたが失敗に終わった.

```
$ tc qdisc add dev eth1 root netem delay 0.117ms
```

なぜなら,tc のネットワークエミュレータは, その使用によって約 2 ミリ秒の遅延が生じるからである. よって以下の実験結果では,Controller-各 CCDM ノード間を通過する場合には 0.117 ミリ秒の遅延 (往復であれば 0.234 ミリ秒の遅延) を追加することとする.

図 6.9 は処理実行時間の計測結果である. 処理実行時間の計測は time アプリケーションを用いて行い, それぞれ 10 回の試行を行った. 左の棒グラフは CCDM システムを用いた場合の結果で, 右の棒グラフが CCDM システムを用いない場合の結果である. 縦軸は処理実行時間で単位はミリ秒である. 青い系列がアプリケーションの処理実行時間である.CCDM システムを用いない場合に付加する必要がある 0.234 ミリ秒の遅延は赤い系列で示してある. 図中の誤差範囲は各計測の標準偏差により示している.

この結果は CCDM システムを用いることにより 0.448 ミリ秒の処理時間の増加があることを示して

いる. しかし, 通常のアプリケーションの使用であれば, この時間は無視できると考えて良いだろう.

6.3.3 システムの展開

CCDM システムではネットワークセグメント毎 (空間的には各部屋) に CCDM ランタイムを実装したホストが存在することを前提としている. 完全なシステムの導入には困難を伴うかもしれないが,CCDM ノードの漸次的な導入も可能である. 新規 CCDM ノードの導入の際には, リソーススクリプトに CCDM ノードの追加を記述するだけで, 全ての実行スクリプトに追加を反映することができる.

第 7 章

まとめ

本研究では, 中央管理方式によりインターネット上のセンサおよびアクチュエータの管理を行う CCDM を提案した. 中央管理のアーキテクチャは, 一般に, スケーラビリティや弾力性という点で不利な面があると考えられている. 一方で, システムの単純化や統一性, メンテナンスという面では優れており, 制御・管理データに関して厳密な一貫性の提供と設定に関する人的と煩雑性の軽減を要求するネットワーク管理の領域に関しては, 適合しているアーキテクチャであると考える.

CCDM における管理インターフェースは, 提供するアプリケーションについて記述するサービススクリプトとネットワークの情報を記述するリソーススクリプトの 2 種類のスクリプトを用いることが特徴である. サービススクリプトはセンサおよびアクチュエータ機器の操作に必要な四則演算, 条件判断, 分岐などの命令セットを提供する. リソーススクリプトは CCDM ノードの IP アドレス, 利用可能なポートの情報やセンサおよびアクチュエータのアクセスに必要なプロトコルを記述している. その結果, サービススクリプトの記述においては, デバイス毎のプロトコルの差異が隠蔽され, サービス提供者の負担が軽減される. サービススクリプトはリソーススクリプトに記述されたネットワーク情報の通りに分割され, 各 CCDM ノードに配置される.

本研究では CCDM システムの検証, 評価を行うための実験環境を構築し, 実験を行った. 動作検証に当たっては,CCDM システムでの利用が想定される 7 つのアプリケーションを作成し, 実験環境で動作することを確認した.CCDM システムがネットワーク管理において達成することを目的としているのは, ネットワーク管理が容易になることと同時に, 耐障害性があるサービスの実行環境を提供することである. システムの評価においては, 管理コスト, 耐障害性, 導入によるオーバヘッドを計測し考察した. 実験により CCDM システムはネットワークの状態変更や新しいサービスの提供時の管理コストが少なく, ネットワークの障害にも強いという結果を得た. 一方で,CCDM の導入により, システム全体でのサービスの処理速度が低下する問題やネットワーク毎に CCDM ノードを設置しなければならないという問題が考えられる. しかし, 実験結果によりサービスの処理速度の低下は, 実際の使用に当たっては問題の無い程度であることがわかった. また,CCDM ノードは漸次的な追加が可能であるため, 導入のオーバヘッドも大きくはない.

本研究は, 増大するセンサおよびアクチュエータの管理に挑戦し, 解決することを目的とした.CCDM システムはネットワーク管理者にとって扱うことが容易で, ネットワークの障害に強いアプリケーション実行基盤を提供する. 本研究では,CCDM のプロトタイプの実装と実験環境の構築により,CCDM シ

ステムが動作し十分に実用的であることを検証し評価した.

謝辞

修士の研究活動では, 本論文の執筆以外にも多くの事を学び多くの人からご指導とご支援を頂きました. ここに 2 年間の研究生活の感謝の意を表します.

まず, 研究を進めるに当たり, 大局的な視野と深い知識により終始的確なご指導, 助言を下さった江崎浩教授に深く感謝致します. 江崎浩教授からは学業の知識のみならず, 研究に対する心構え, 人生の生き方, 周囲への深い思いやりを学びました.

研究だけでなくネットワーク全般についての深い知識でご指導下さった山本成一博士, 豊富な経験と知識に基づき的確な助言を下さった岡部宣夫博士, 金海好彦氏, 土井裕介氏, 土本康生博士に感謝致します. 研究に対する真摯な姿勢を持ち厳しくご指導して下さった吉田薫氏, 研究そのものだけでなく研究に必要な余暇の過ごし方についてもご指導下さった藤田祥氏に感謝致します.

本論文の執筆に当たって, 研究の方向性, デザイン, 詳細についてご指導下さり, また議論や質問に長い時間お付き合い下さった落合秀也氏に深く感謝を申し上げます. 落合氏は論文の指導だけでなく, 数々のデモ展示-APNG ワークショップ,UNS,JGN,C40 など-に連れて行って下さり, そこで多くを経験させて頂きました. Live E! Project の活動を引っ張る姿からはリーダシップを学びました.

先輩として面倒見が良く同期として飾らず接して下さった安本直史氏, 論文執筆の苦楽を共にした真面目な姜鵬氏, 常に明るくいろいろな話題で話をしてくれる Sergio Carriho 氏に感謝致します. 同期として常に研究室に在室し, 私では至らない部分を補って研究室を盛り上げてくれた阪本裕介氏に感謝致します. 阪本氏の研究に取り組む姿勢は真摯であり, 同期であると同時に良き模範でした.

運用が出来研究室のネットワークを支えてくれる浅井大史君, 非常に研究が出来いろいろと助言をくれる肥村洋輔君, 趙越君, 川上雄也君, 杉田毅博君, 橋本紘希君, 平山陽彦君,Sathita Kaveevivitchai さん, Leela-Amornsin Lertluck 君,David Jageberg 君に感謝致します. 学生が研究し易い環境を提供すべく, 日々細やかな心配りをなさって下さる高橋富美氏, 田坂佳苗氏に感謝致します.

広く学生を受け入れ成長する場を提供して下さる WIDE Project の皆様, Live E! Project の松浦知史博士, 山内正人氏, 皆様に感謝致します. また, 電気系計算機管理 TA としてお世話を頂いた斎藤秀雄氏を始めとした TA メンバの皆様, 電気系事務室の皆様に感謝致します.

「我思う, ゆえに我あり.」の命題を残したルネ・デカルト氏に感謝致します.

研究生活や学生生活においていろいろなご指導, ご支援を下さった全ての皆様に感謝致します.

最後に, 長い学生生活の間 1 つも不満を言わずに支えてくれた妻由里さんと笑顔とだじゃれとときどき涙で闘いを挑んでくる我が子大基に感謝致します.

2009 年 2 月 4 日

杉山 哲弘

参考文献

[1] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts and S. Wolff: “A Brief History of the Internet”, CONTRIBUTIONS IN LIBRARIANSHIP AND INFORMATION SCIENCE, **96**, pp. 3–24 (2001).

[2] 総務省:“情報通信白書 平成 20 年版” (2008). http://www.johotsusintokei.soumu.go.jp/whitepaper/ja/cover/index.htm.

[3] 小林浩, 江崎浩：“インターネット総論”, 共立出版株式会社 (2002).

[4] J. Postel: “Internet Protocol”, RFC 791 (Standard) (1981). Updated by RFC 1349.

[5] G. Moore: “Cramming More Components Onto Integrated Circuits”, Proceedings of the IEEE, **86**, 1, pp. 82–85 (1998).

[6] 落合秀也, 松浦知史, 砂原秀樹, 中山雅哉, 江崎浩：“広域センサネットワークの運用構造と多属性検索”, 電子情報通信学会論文誌 B, **J91-B**, 10, pp. 1160–1170 (2008).

[7] 江崎浩：“ICT を用いたグリーンキャンパスに向けた取り組み”, 信学技報, 第 108 巻 of IA2008-1, 東京, pp. 1–6 (2008). 2008 年 5 月 30 日 (金)　機械振興会館 (IA).

[8] D. Oppenheimer, A. Ganapathi and D. Patterson: “Why Do Internet Services Fail, and What Can Be Done About It?”, USENIX USITS ’03: 4th USENIX Sympodium on Internet Technologies and Systems (2003).

[9] Z. Kerravala: “Configuration Management Delivers Business Resiliency”, The Yankee Group (2002).

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner: “OpenFlow: Enabling Innovation in Campus Networks”, SIGCOMM Comput Commun. Rev, **38**, 2, pp. 69–74 (2008).

[11] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown and S. Shenker: “Ethane: Taking Control of the Enterprise”, SIGCOMM ’07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, Kyoto, Japan, ACM, pp. 1–12 (2007).

[12] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci: “Wireless sensor networks: a survey”, Computer Networks, **38**, 4, pp. 393–422 (2002).

[13] W. Kastner, G. Neugschwandtner, S. Soucek and H. Newman: “Communication Systems for Building Automation and Control”, Proceedings of the IEEE, **93**, 6, pp. 1178–1203 (2005).

[14] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown and S. Shenker:

"SANE: A Protection Architecture for Enterprise Networks", USENIX Security Symposium (2006).

[15] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan and H. Zhang: "A Clean Slate 4D Approach to Network Control and Management", ACM SIGCOMM Computer Communication Review, **35**, 5, pp. 41–54 (2005).

[16] 落合秀也, 江崎浩："高級言語によるネットワーク汎用 I/O 制御とプロトコル翻訳機構", 情報処理学会論文誌, **49**, 10, pp. 3451–3461 (2008).

[17] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson and J. Rexford: "The cutting EDGE of IP router configuration", ACM SIGCOMM Computer Communication Review, **34**, 1, p. 21 (2004).

[18] R. Alimi, Y. Wang and Y. R. Yang: "Shadow Configuration as a Network Nanagement Primitive", SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication, Seattle, WA, USA, ACM, pp. 111–122 (2008).

[19] ASHRAE: "ASHRAE: Official Website of American Society of Heating, Refrigerating and Air-Conditioning Engineers". http://www.ashrae.org/.

[20] S. Bushby: "BACnet A Standard Communication Infrastructure for Intelligent Buildings", National Institute of Standards and Technology, Published in Automation in Construction, **6**, 5–6 (1997).

[21] Standing Standard Project Commitee: "BACnet Official Website of ASHRAE SSPC 135", http://www.bacnet.org/.

[22] H. Newman: "BACnet - The New Standard Protocol", Electrical Contractor, pp. 119–122 (1997).

[23] E. Corporation: "Building Automation Technology Review", Technical report, ECHELON, http://www.echelon.com/solutions/building/papers/buildautomationreview.pdf (2003).

[24] A. Chervert: "Considerations for Using XML Web Services for Device-todevice Communication", Technical report, ECHELON, http://www.echelon.com/support/documentation/papers/XML-Considerations.pdf.

[25] D. Fisher: "BACnet and LonWorks: a white paper", AIRAH JOURNAL, **54**, 2, pp. 16–21 (2000).

[26] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler and F. Yergeau: "Extensible Markup Language (XML) 1.0", W3C Recommendation, **6**, (2000).

[27] R. Enns: "NETCONF Configuration Protocol", RFC 4741 (Proposed Standard) (2006).

[28] M. Wasserman and T. Goddard: "Using the NETCONF Configuration Protocol over Secure SHell (SSH)", RFC 4742 (Proposed Standard) (2006).

[29] T. Goddard: "Using NETCONF over the Simple Object Access Protocol (SOAP)", RFC 4743 (Proposed Standard) (2006).

[30] E. Lear and K. Crozier: "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)", RFC 4744 (Proposed Standard) (2006).

[31] 鈴木誠, 猿渡俊介, 南正輝, 森川博之：“無線センサノードのためのハードリアルタイム保証が可能な仮想マシン”, 電子情報通信学会論文誌, **J92-B**, 1, pp. 130–139 (2009).

[32] P. Levis and D. Culler: “Maté: A Tiny Virtual Machine for Sensor Networks”, ACM SIGOPS Operating Systems Review, **36**, 5, pp. 85–95 (2002).

[33] J. Case, M. Fedor, M. Schoffstall and J. Davin: “Simple Network Management Protocol (SNMP)”, RFC 1157 (Historic) (1990).

[34] J. Case, K. McCloghrie, M. Rose and S. Waldbusser: “Introduction to version 2 of the Internet-standard Network Management Framework”, RFC 1441 (Historic) (1993).

[35] Ambient Weather Co., Ltd.: “Oregon Scientific WM918 Weather Station”. http://www.ambientweather.com/wm918.html.

[36] Atmark Techno Co., Ltd.: “Armadillo-210”. http://armadillo.atmark-techno.com/armadillo-210.

[37] M. Felleisen: “On the Expressive Power of Programming Languages”, Science of Computer Programming, **17**, 1-3, pp. 35–75 (1991).

[38] J. Ousterhout: “Scripting: Higher-Level Programming for the 21st Century”, COMPUTER, pp. 23–30 (1998).

[39] K. Hatori and K. Hiraki: “A network programming language based on concurrent processes and regular expressions”, SE’07: Proceedings of the 25th conference on IASTED International Multi-Conference, Anaheim, CA, USA, ACTA Press, pp. 105–110 (2007).

[40] K. Melkote: “Scaling Enterprise Wireless LAN Deployments”, ARUBA Networks (2007).

[41] Milan P. Stanic: “tc: traffic control (Linux Qos Control tool)”. http://www.rns-nis.co.yu/ mps/linux-tc.html.

発表文献

[1] 杉山哲弘, 落合秀也, 江崎浩：”センサーネットワークに於けるサーバの負荷容量を考慮したデータ集約の実験と提案”, 信学技報, IA2007-36, 107, 280, pp.33-38 (2007).

[2] 落合秀也, 杉山哲弘, 阪本裕介, 山内正人, 石塚宏, 松浦知史, 砂原秀樹, 中山雅哉, 江崎浩：”Live E! 広域センサネットワークの運用状況分析 — 2008 年 5 月の運用状況 —”, 信学技報 , IA2008-3, 108, 74, pp.11-16 (2008).

[3] 杉山哲弘, 落合秀也, 江崎浩：”CCDM:中央管理手法によるセンサ・アクチュエータ機器管理アーキテクチャの提案”, 信学総合大会 (2009). [発表予定]

付録 A

リソーススクリプトの例

ソースコード A.1　実験で用いたリソーススクリプト

```
<?xml version="1.0" encoding="UTF-8"?>
<resource>
  <host name="A">
    <ip>192.168.1.2</ip>
    <port>
      <begin>10000</begin>
      <end>20000</end>
    </port>
    <devicelist>
      <device name="Display1" func="OUTPUT">
        <progn>
          <setq name="client">
            <dgSocket>
              <text>192.168.1.9</text>
              <int>10001</int>
            </dgSocket>
          </setq>
          <setq name="out">
            <byteArrayAppend>
              <byteArrayAppend>
                <byteArrayAppend>
                  <byteArrayAppend>
                    <byteArrayAppend>
                      <byteArray />
                      <byte>03</byte>
                    </byteArrayAppend>
                    <byte>134</byte>
                  </byteArrayAppend>
                  <byte>01</byte>
                </byteArrayAppend>
                <byte>00</byte>
              </byteArrayAppend>
              <byte>00</byte>
            </byteArrayAppend>
          </setq>
          <dgSend>
            <getq name="client" />
            <getq name="out"/>
          </dgSend>

          <setq name="out">
            <byteArrayAppend>
              <byteArrayAppend>
                <byteArrayAppend>
                  <byteArrayAppend>
                    <byteArray />
                    <byte>03</byte>
                  </byteArrayAppend>
                  <byte>06</byte>
                </byteArrayAppend>
                <byte>01</byte>
              </byteArrayAppend>
```

```
53              <byte>00</byte>
54            </byteArrayAppend>
55          </setq>
56          <dump><text>the following is ASCII number.</text></dump>
57          <dump><arg/></dump>
58          <if>
59            <eq>
60              <arg/>
61              <byte>46</byte>
62            </eq>
63            <progn>
64              <setq name="out">
65                <byteArrayAppend>
66                  <getq name="out"/>
67                  <byte>127</byte>
68                </byteArrayAppend>
69              </setq>
70            </progn>
71          </if>
72          <if>
73            <eq>
74              <arg/>
75              <byte>47</byte>
76            </eq>
77            <progn>
78              <setq name="out">
79                <byteArrayAppend>
80                  <getq name="out"/>
81                  <byte>255</byte>
82                </byteArrayAppend>
83              </setq>
84            </progn>
85          </if>
86          <if>
87            <eq>
88              <arg/>
89              <byte>48</byte>
90            </eq>
91            <progn>
92              <setq name="out">
93                <byteArrayAppend>
94                  <getq name="out"/>
95                  <byte>192</byte>
96                </byteArrayAppend>
97              </setq>
98            </progn>
99          </if>
100         <if>
101           <eq>
102             <arg/>
103             <byte>49</byte>
104           </eq>
105           <progn>
106             <setq name="out">
107               <byteArrayAppend>
108                 <getq name="out"/>
109                 <byte>249</byte>
110               </byteArrayAppend>
111             </setq>
112           </progn>
113         </if>
114         <if>
115           <eq>
116             <arg/>
117             <byte>50</byte>
118           </eq>
119           <progn>
120             <setq name="out">
121               <byteArrayAppend>
122                 <getq name="out"/>
123                 <byte>164</byte>
124               </byteArrayAppend>
125             </setq>
126           </progn>
127         </if>
128         <if>
```

```
129             <eq>
130               <arg/>
131               <byte>51</byte>
132             </eq>
133             <progn>
134               <setq name="out">
135                 <byteArrayAppend>
136                   <getq name="out"/>
137                   <byte>176</byte>
138                 </byteArrayAppend>
139               </setq>
140             </progn>
141           </if>
142           <if>
143             <eq>
144               <arg/>
145               <byte>52</byte>
146             </eq>
147             <progn>
148               <setq name="out">
149                 <byteArrayAppend>
150                   <getq name="out"/>
151                   <byte>153</byte>
152                 </byteArrayAppend>
153               </setq>
154             </progn>
155           </if>
156           <if>
157             <eq>
158               <arg/>
159               <byte>53</byte>
160             </eq>
161             <progn>
162               <setq name="out">
163                 <byteArrayAppend>
164                   <getq name="out"/>
165                   <byte>146</byte>
166                 </byteArrayAppend>
167               </setq>
168             </progn>
169           </if>
170           <if>
171             <eq>
172               <arg/>
173               <byte>54</byte>
174             </eq>
175             <progn>
176               <setq name="out">
177                 <byteArrayAppend>
178                   <getq name="out"/>
179                   <byte>131</byte>
180                 </byteArrayAppend>
181               </setq>
182             </progn>
183           </if>
184           <if>
185             <eq>
186               <arg/>
187               <byte>55</byte>
188             </eq>
189             <progn>
190               <setq name="out">
191                 <byteArrayAppend>
192                   <getq name="out"/>
193                   <byte>216</byte>
194                 </byteArrayAppend>
195               </setq>
196             </progn>
197           </if>
198           <if>
199             <eq>
200               <arg/>
201               <byte>56</byte>
202             </eq>
203             <progn>
204               <setq name="out">
```

```
205                 <byteArrayAppend>
206                   <getq name="out"/>
207                   <byte>128</byte>
208                 </byteArrayAppend>
209               </setq>
210             </progn>
211           </if>
212           <if>
213             <eq>
214               <arg/>
215               <byte>57</byte>
216             </eq>
217             <progn>
218               <setq name="out">
219                 <byteArrayAppend>
220                   <getq name="out"/>
221                   <byte>152</byte>
222                 </byteArrayAppend>
223               </setq>
224             </progn>
225           </if>
226           <if>
227             <eq>
228               <arg/>
229               <byte>58</byte>
230             </eq>
231             <progn>
232               <setq name="out">
233                 <byteArrayAppend>
234                   <getq name="out"/>
235                   <byte>0</byte>
236                 </byteArrayAppend>
237               </setq>
238             </progn>
239           </if>
240
241           <dgSend>
242             <getq name="client" />
243             <getq name="out"/>
244           </dgSend>
245           <dgClose>
246             <getq name="client" />
247           </dgClose>
248         </progn>
249       </device>
250       <device name="PICNIC" func="INPUT">
251         <progn>
252           <setq name="request">
253             <byteArrayAppend>
254               <byteArrayAppend>
255                 <byteArrayAppend>
256                   <byteArray />
257                   <byte>04</byte>
258                 </byteArrayAppend>
259                 <byte>161</byte>
260               </byteArrayAppend>
261               <byte>80</byte>
262             </byteArrayAppend>
263           </setq>
264           <setq name="INPUT">
265             <dgRecv>
266               <dgSend>
267                 <dgSocket>
268                   <text>192.168.1.9</text>
269                   <int>10001</int>
270                 </dgSocket>
271                 <getq name="request" />
272               </dgSend>
273             </dgRecv>
274           </setq>
275         </progn>
276       </device>
277       </devicelist>
278   </host>
279   <host name="B">
280     <ip>192.168.1.3</ip>
```

```
281     <port>
282       <begin>10000</begin>
283       <end>20000</end>
284     </port>
285     <devicelist>
286       <device name="CLIENT" func="OUTPUT">
287         <progn>
288           <setq name="client">
289             <dgSocket>
290               <text>192.168.1.1</text>
291               <int>30000</int>
292             </dgSocket>
293           </setq>
294           <dgSend>
295             <getq name="client" />
296             <arg/>
297           </dgSend>
298           <dgClose>
299             <getq name="client" />
300           </dgClose>
301         </progn>
302       </device>
303       <device name="ArmadilloTemp" func="INPUT">
304         <progn>
305           <setq name="request">
306             <byteArrayAppend>
307               <byteArray />
308               <byte>16</byte>
309             </byteArrayAppend>
310           </setq>
311           <setq name="INPUT">
312             <dgRecv>
313               <dgSend>
314                 <dgSocket>
315                   <text>192.168.1.8</text>
316                   <int>9999</int>
317                 </dgSocket>
318                 <getq name="request"/>
319               </dgSend>
320             </dgRecv>
321           </setq>
322         </progn>
323       </device>
324       <device name="VirtualSensor" func="INPUT">
325         <progn>
326           <setq name="request">
327             <byteArrayAppend>
328               <byteArray />
329               <byte>0</byte>
330             </byteArrayAppend>
331           </setq>
332           <setq name="INPUT">
333             <dgRecv>
334               <dgSend>
335                 <dgSocket>
336                   <text>192.168.1.1</text>
337                   <int>9998</int>
338                 </dgSocket>
339                 <getq name="request"/>
340               </dgSend>
341             </dgRecv>
342           </setq>
343         </progn>
344       </device>
345     </devicelist>
346   </host>
347   <host name="C">
348     <ip>192.168.1.4</ip>
349     <port>
350       <begin>10000</begin>
351       <end>20000</end>
352     </port>
353     <devicelist>
354       <device name="Display2" func="OUTPUT">
355         <progn>
356           <setq name="client">
```

```
357             <dgSocket>
358               <text>192.168.1.1</text>
359               <int>30000</int>
360             </dgSocket>
361           </setq>
362           <dgSend>
363             <getq name="client" />
364             <arg/>
365           </dgSend>
366           <dgClose>
367             <getq name="client" />
368           </dgClose>
369         </progn>
370       </device>
371       <device name="ArmadilloTemp2" func="INPUT">
372         <progn>
373           <setq name="request">
374             <byteArrayAppend>
375               <byteArray />
376               <byte>16</byte>
377             </byteArrayAppend>
378           </setq>
379           <setq name="INPUT">
380             <dgRecv>
381               <dgSend>
382                 <dgSocket>
383                   <text>192.168.1.8</text>
384                   <int>9999</int>
385                 </dgSocket>
386                 <getq name="request"/>
387               </dgSend>
388             </dgRecv>
389           </setq>
390         </progn>
391       </device>
392     </devicelist>
393   </host>
394 </resource>
```

付録 B

Weather Display Service

ソースコード B.1　Weather Display Service

```
<!-- Weather Display Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Weather Display Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="ArmadilloTemp"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>
    <catch channel="A" name="temp"/>

    <setq name="i">
      <int>0</int>
    </setq>

    <while>
      <lt>
        <getq name="i"/>
        <byteArrayLength>
          <getq name="temp"/>
        </byteArrayLength>
      </lt>
      <progn>
        <setq name="number">
          <byteArrayGet>
            <getq name="temp"/>
            <getq name="i"/>
          </byteArrayGet>
        </setq>
        <OUTPUT to="Display1">
          <ARG>
            <getq name="number"/>
          </ARG>
        </OUTPUT>
        <setq name="i">
          <plus>
            <getq name="i"/>
            <int>1</int>
          </plus>
        </setq>
        <sleep>
          <int>1000</int>
        </sleep>
```

```
53        </progn>
54      </while>
55      <OUTPUT to="Display1">
56        <ARG>
57          <byte>47</byte>
58        </ARG>
59      </OUTPUT>
60    </chunk>
61  </progn>
```

付録 C

Multicast Data Delivery Service

ソースコード C.1　Multicast Data Delivery Service

```
<!-- Multicast Data Delivery Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Multicast Data Delivery Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="ArmadilloTemp"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>
    <catch channel="A" name="temp"/>

    <setq name="i">
      <int>0</int>
    </setq>

    <while>
      <lt>
        <getq name="i"/>
        <byteArrayLength>
          <getq name="temp"/>
        </byteArrayLength>
      </lt>
      <progn>
        <setq name="number">
          <byteArrayGet>
            <getq name="temp"/>
            <getq name="i"/>
          </byteArrayGet>
        </setq>
        <OUTPUT to="Display1">
          <ARG>
            <getq name="number"/>
          </ARG>
        </OUTPUT>
        <setq name="i">
          <plus>
            <getq name="i"/>
            <int>1</int>
          </plus>
        </setq>
        <sleep>
          <int>1000</int>
        </sleep>
```

```
53        </progn>
54      </while>
55      <OUTPUT to="Display1">
56        <ARG>
57          <byte>47</byte>
58        </ARG>
59      </OUTPUT>
60    </chunk>
61
62    <chunk>
63      <catch channel="A" name="temp"/>
64
65      <OUTPUT to="Display2">
66        <ARG>
67          <getq name="temp"/>
68        </ARG>
69      </OUTPUT>
70    </chunk>
71
72  </progn>
```

付録 D

Weather Threshold Notification Service

ソースコード D.1　Weather Threshold Notification Service

```
<!-- Weather Threshold Notification Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Weather Threshold Notification Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="PICNIC"/>
    <post channel="A" name="INPUT"/>
  </chunk>


  <chunk>
    <catch channel="A" name="response"/>

    <setq name="temperature">
      <int>0</int>
    </setq>

    <if>
      <gteq>
        <byteArrayLength>
          <getq name="response" />
        </byteArrayLength>
        <int>8</int>
      </gteq>
      <progn>
        <setq name="temperature">
          <plus>
            <multiply>
              <byteArrayGet>
                <getq name="response"/>
                <int>4</int>
              </byteArrayGet>
              <int>128</int>
            </multiply>
            <multiply>
              <byteArrayGet>
                <getq name="response"/>
                <int>5</int>
              </byteArrayGet>
              <double>0.5</double>
            </multiply>
          </plus>
        </setq>
        <dump>
```

```
53            <getq name="temperature"/>
54          </dump>
55        </progn>
56      </if>
57
58      <if>
59        <gt>
60          <getq name="temperature"/>
61          <int>20</int>
62        </gt>
63        <OUTPUT to="CLIENT">
64          <ARG>
65            <byteArrayAppend>
66              <byteArray/>
67              <castbyte>
68                <getq name="temperature"/>
69              </castbyte>
70            </byteArrayAppend>
71          </ARG>
72        </OUTPUT>
73      </if>
74    </chunk>
75  </progn>
```

付録 E

Data Aggregation Service

ソースコード E.1　Data Aggregation Service

```
<!-- Data Aggregation Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Data Aggregation Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="VirtualSensor"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>

    <setq name="count_num">
      <int>3</int>
    </setq>

    <setq name="j">
      <int>0</int>
    </setq>

    <setq name="sum">
      <double>0</double>
    </setq>

    <while>
      <lt>
        <getq name="j"/>
        <getq name="count_num"/>
      </lt>
      <progn>
        <catch channel="A" name="temp"/>

        <setq name="t">
          <plus>
            <multiply>
              <castint>
                <byteArrayGet>
                  <getq name="temp"/>
                  <int>0</int>
                </byteArrayGet>
              </castint>
              <int>10</int>
            </multiply>
            <castint>
              <byteArrayGet>
```

```
53                <getq name="temp"/>
54                <int>1</int>
55              </byteArrayGet>
56            </castint>
57          </plus>
58        </setq>
59
60        <setq name="sum">
61          <plus>
62            <getq name="sum"/>
63            <divide>
64              <getq name="t"/>
65              <int>10</int>
66            </divide>
67          </plus>
68        </setq>
69
70        <dump>
71          <text>the following is sum.</text>
72        </dump>
73        <dump>
74          <getq name="sum"/>
75        </dump>
76
77        <setq name="j">
78          <plus>
79            <getq name="j"/>
80            <int>1</int>
81          </plus>
82        </setq>
83      </progn>
84    </while>
85
86    <setq name="avg">
87      <divide>
88        <getq name="sum"/>
89        <getq name="count_num"/>
90      </divide>
91    </setq>
92
93    <setq name="data">
94      <byteArrayAppend>
95        <byteArray/>
96        <byte>47</byte>
97      </byteArrayAppend>
98    </setq>
99
100   <if>
101     <gteq>
102       <getq name="avg"/>
103       <int>10</int>
104     </gteq>
105     <progn>
106       <setq name="data">
107         <byteArrayAppend>
108           <getq name="data"/>
109           <castbyte>
110             <plus>
111               <divide>
112                 <getq name="avg"/>
113                 <int>10</int>
114               </divide>
115               <int>48</int>
116             </plus>
117           </castbyte>
118         </byteArrayAppend>
119       </setq>
120     </progn>
121   </if>
122
123   <byteArrayAppend>
124     <getq name="data"/>
125     <castbyte>
126       <plus>
127         <mod>
128           <castint>
```

```
129                 <getq name="avg"/>
130               </castint>
131               <int>10</int>
132             </mod>
133             <int>48</int>
134           </plus>
135         </castbyte>
136       </byteArrayAppend>
137
138       <byteArrayAppend>
139         <getq name="data"/>
140         <byte>46</byte>
141       </byteArrayAppend>
142
143       <byteArrayAppend>
144         <getq name="data"/>
145         <castbyte>
146           <plus>
147             <mod>
148               <castint>
149                 <multiply>
150                   <getq name="avg"/>
151                   <int>10</int>
152                 </multiply>
153               </castint>
154               <int>10</int>
155             </mod>
156             <int>48</int>
157           </plus>
158         </castbyte>
159       </byteArrayAppend>
160
161       <setq name="i">
162         <int>0</int>
163       </setq>
164
165       <while>
166         <lt>
167           <getq name="i"/>
168           <byteArrayLength>
169             <getq name="data"/>
170           </byteArrayLength>
171         </lt>
172         <progn>
173           <setq name="number">
174             <byteArrayGet>
175               <getq name="data"/>
176               <getq name="i"/>
177             </byteArrayGet>
178           </setq>
179           <OUTPUT to="Display1">
180             <ARG>
181               <getq name="number"/>
182             </ARG>
183           </OUTPUT>
184           <setq name="i">
185             <plus>
186               <getq name="i"/>
187               <int>1</int>
188             </plus>
189           </setq>
190           <sleep>
191             <int>1000</int>
192           </sleep>
193         </progn>
194       </while>
195       <OUTPUT to="Display1">
196         <ARG>
197           <byte>47</byte>
198         </ARG>
199       </OUTPUT>
200     </chunk>
201
202   </progn>
```

付録 F

Several Source Data Aggregation Service

ソースコード F.1　Several Source Data Aggregation Service

```
<!-- Several Source Data Aggregation Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Several Source Data Aggregation Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="VirtualSensor"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>
    <sleep>
      <int>5000</int>
    </sleep>
    <INPUT from="ArmadilloTemp2"/>
    <post channel="B" name="INPUT"/>
  </chunk>

  <chunk>
    <catch channel="A" name="temp1"/>

    <setq name="t1">
      <divide>
        <plus>
          <multiply>
            <castint>
              <byteArrayGet>
                <getq name="temp1"/>
                <int>0</int>
              </byteArrayGet>
            </castint>
            <int>10</int>
          </multiply>
          <castint>
            <byteArrayGet>
              <getq name="temp1"/>
              <int>1</int>
            </byteArrayGet>
          </castint>
        </plus>
        <double>10</double>
      </divide>
    </setq>

    <catch channel="B" name="temp2"/>
```

```
53
54      <setq name="i">
55        <int>0</int>
56      </setq>
57
58      <setq name="t2">
59        <double>0</double>
60      </setq>
61
62      <setq name="integer">
63        <int>1</int>
64      </setq>
65
66      <while>
67        <lt>
68          <getq name="i"/>
69          <byteArrayLength>
70            <getq name="temp2"/>
71          </byteArrayLength>
72        </lt>
73        <progn>
74          <dump><text>while begin</text></dump>
75          <setq name="cur_byte">
76            <byteArrayGet>
77              <getq name="temp2"/>
78              <getq name="i"/>
79            </byteArrayGet>
80          </setq>
81          <if>
82            <eq>
83              <getq name="integer"/>
84              <int>1</int>
85            </eq>
86            <progn>
87              <if>
88                <eq>
89                  <getq name="cur_byte"/>
90                  <byte>46</byte>
91                </eq>
92                <setq name="integer">
93                  <int>0</int>
94                </setq>
95                <progn>
96                  <setq name="t2">
97                    <plus>
98                      <multiply>
99                        <getq name="t2"/>
100                       <int>10</int>
101                     </multiply>
102                     <castint>
103                       <minus>
104                         <getq name="cur_byte"/>
105                         <byte>48</byte>
106                       </minus>
107                     </castint>
108                   </plus>
109                 </setq>
110               </progn>
111             </if>
112           </progn>
113           <setq name="t2">
114             <plus>
115               <getq name="t2"/>
116               <divide>
117                 <castint>
118                   <minus>
119                     <getq name="cur_byte"/>
120                     <byte>48</byte>
121                   </minus>
122                 </castint>
123                 <double>10</double>
124               </divide>
125             </plus>
126           </setq>
127         </if>
128         <setq name="i">
```

```
          <plus>
            <getq name="i"/>
            <int>1</int>
          </plus>
        </setq>
      </progn>
    </while>

    <setq name="out_temp">
      <divide>
        <plus>
          <getq name="t1"/>
          <getq name="t2"/>
        </plus>
        <double>2</double>
      </divide>
    </setq>
    <setq name="data">
      <byteArrayAppend>
        <byteArray/>
        <byte>47</byte>
      </byteArrayAppend>
    </setq>

    <if>
      <gteq>
        <getq name="out_temp"/>
        <int>10</int>
      </gteq>
      <progn>
        <setq name="data">
          <byteArrayAppend>
            <getq name="data"/>
            <castbyte>
              <plus>
                <divide>
                  <getq name="out_temp"/>
                  <int>10</int>
                </divide>
                <int>48</int>
              </plus>
            </castbyte>
          </byteArrayAppend>
        </setq>
      </progn>
    </if>

    <byteArrayAppend>
      <getq name="data"/>
      <castbyte>
        <plus>
          <mod>
            <castint>
              <getq name="out_temp"/>
            </castint>
            <int>10</int>
          </mod>
          <int>48</int>
        </plus>
      </castbyte>
    </byteArrayAppend>

    <byteArrayAppend>
      <getq name="data"/>
      <byte>46</byte>
    </byteArrayAppend>

    <byteArrayAppend>
      <getq name="data"/>
      <castbyte>
        <plus>
          <mod>
            <castint>
              <multiply>
                <getq name="out_temp"/>
                <int>10</int>
```

```
              </multiply>
            </castint>
            <int>10</int>
          </mod>
          <int>48</int>
        </plus>
      </castbyte>
    </byteArrayAppend>

    <setq name="i">
      <int>0</int>
    </setq>

    <while>
      <lt>
        <getq name="i"/>
        <byteArrayLength>
          <getq name="data"/>
        </byteArrayLength>
      </lt>
      <progn>
        <setq name="number">
          <byteArrayGet>
            <getq name="data"/>
            <getq name="i"/>
          </byteArrayGet>
        </setq>
        <OUTPUT to="Display1">
          <ARG>
            <getq name="number"/>
          </ARG>
        </OUTPUT>
        <setq name="i">
          <plus>
            <getq name="i"/>
            <int>1</int>
          </plus>
        </setq>
        <sleep>
          <int>1000</int>
        </sleep>
      </progn>
    </while>
    <OUTPUT to="Display1">
      <ARG>
        <byte>47</byte>
      </ARG>
    </OUTPUT>
  </chunk>
</progn>
```

付録 G

Largest Value Notification Service

ソースコード G.1　Largest Value Notification Service Service

```
<!-- Largest Value Notification Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Largest Value Notification Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>3000</int>
    </sleep>
    <INPUT from="VirtualSensor"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>

    <setq name="count_num">
      <int>100</int>
    </setq>

    <setq name="j">
      <int>0</int>
    </setq>

    <setq name="max_temp">
      <int>0</int>
    </setq>

    <while>
      <lt>
        <getq name="j"/>
        <getq name="count_num"/>
      </lt>
      <progn>
        <catch channel="A" name="temp"/>

        <setq name="t">
          <plus>
            <multiply>
              <castint>
                <byteArrayGet>
                  <getq name="temp"/>
                  <int>0</int>
                </byteArrayGet>
              </castint>
              <int>10</int>
            </multiply>
            <castint>
               <byteArrayGet>
```

```
                <getq name="temp"/>
                <int>1</int>
              </byteArrayGet>
            </castint>
          </plus>
        </setq>

        <if>
          <gt>
            <getq name="t"/>
            <getq name="max_temp"/>
          </gt>
          <progn>
            <setq name="max_temp">
              <getq name="t"/>
            </setq>

            <setq name="out_temp">
              <divide>
                <getq name="max_temp"/>
                <double>10</double>
              </divide>
            </setq>
            <setq name="data">
              <byteArrayAppend>
                <byteArray/>
                <byte>47</byte>
              </byteArrayAppend>
            </setq>

            <if>
              <gteq>
                <getq name="out_temp"/>
                <int>10</int>
              </gteq>
              <progn>
                <setq name="data">
                  <byteArrayAppend>
                    <getq name="data"/>
                    <castbyte>
                      <plus>
                        <divide>
                          <getq name="out_temp"/>
                          <int>10</int>
                        </divide>
                        <int>48</int>
                      </plus>
                    </castbyte>
                  </byteArrayAppend>
                </setq>
              </progn>
            </if>

            <byteArrayAppend>
              <getq name="data"/>
              <castbyte>
                <plus>
                  <mod>
                    <castint>
                      <getq name="out_temp"/>
                    </castint>
                    <int>10</int>
                  </mod>
                  <int>48</int>
                </plus>
              </castbyte>
            </byteArrayAppend>

            <byteArrayAppend>
              <getq name="data"/>
              <byte>46</byte>
            </byteArrayAppend>

            <byteArrayAppend>
              <getq name="data"/>
              <castbyte>
```

```
                <plus>
                  <mod>
                    <castint>
                      <multiply>
                        <getq name="out_temp"/>
                        <int>10</int>
                      </multiply>
                    </castint>
                    <int>10</int>
                  </mod>
                  <int>48</int>
                </plus>
              </castbyte>
            </byteArrayAppend>

            <setq name="i">
              <int>0</int>
            </setq>

            <while>
              <lt>
                <getq name="i"/>
                <byteArrayLength>
                  <getq name="data"/>
                </byteArrayLength>
              </lt>
              <progn>
                <setq name="number">
                  <byteArrayGet>
                    <getq name="data"/>
                    <getq name="i"/>
                  </byteArrayGet>
                </setq>
                <OUTPUT to="Display1">
                  <ARG>
                    <getq name="number"/>
                  </ARG>
                </OUTPUT>
                <setq name="i">
                  <plus>
                    <getq name="i"/>
                    <int>1</int>
                  </plus>
                </setq>
                <sleep>
                  <int>1000</int>
                </sleep>
              </progn>
            </while>

            <OUTPUT to="Display1">
              <ARG>
                <byte>47</byte>
              </ARG>
            </OUTPUT>
          </progn>
        </if>

        <setq name="j">
          <plus>
            <getq name="j"/>
            <int>1</int>
          </plus>
        </setq>
      </progn>
    </while>

  </chunk>

</progn>
```

付録 H

Updated Data Delivery Service

ソースコード H.1　Updated Data Delivery Service

```
<!-- Updated Data Delivery Service by Akihiro Sugiyama -->

<progn>
  <initialize>
    <dump>
      <text>It's Updated Data Delivery Service</text>
    </dump>
  </initialize>

  <chunk>
    <sleep>
      <int>9000</int>
    </sleep>
    <INPUT from="VirtualSensor"/>
    <post channel="A" name="INPUT"/>
  </chunk>

  <chunk>

    <setq name="count_num">
      <int>100</int>
    </setq>

    <setq name="j">
      <int>0</int>
    </setq>

    <setq name="prev_temp">
      <int>0</int>
    </setq>

    <while>
      <lt>
        <getq name="j"/>
        <getq name="count_num"/>
      </lt>
      <progn>
        <catch channel="A" name="temp"/>

        <setq name="t">
          <plus>
            <multiply>
              <castint>
                <byteArrayGet>
                  <getq name="temp"/>
                  <int>0</int>
                </byteArrayGet>
              </castint>
              <int>10</int>
            </multiply>
            <castint>
              <byteArrayGet>
```

```
53                <getq name="temp"/>
54                <int>1</int>
55              </byteArrayGet>
56            </castint>
57          </plus>
58        </setq>
59
60        <if>
61          <neq>
62            <getq name="t"/>
63            <getq name="prev_temp"/>
64          </neq>
65          <progn>
66            <setq name="prev_temp">
67              <getq name="t"/>
68            </setq>
69
70            <setq name="out_temp">
71              <divide>
72                <getq name="prev_temp"/>
73                <double>10</double>
74              </divide>
75            </setq>
76            <setq name="data">
77              <byteArrayAppend>
78                <byteArray/>
79                <byte>47</byte>
80              </byteArrayAppend>
81            </setq>
82
83            <if>
84              <gteq>
85                <getq name="out_temp"/>
86                <int>10</int>
87              </gteq>
88              <progn>
89                <setq name="data">
90                  <byteArrayAppend>
91                    <getq name="data"/>
92                    <castbyte>
93                      <plus>
94                        <divide>
95                          <getq name="out_temp"/>
96                          <int>10</int>
97                        </divide>
98                        <int>48</int>
99                      </plus>
100                   </castbyte>
101                 </byteArrayAppend>
102               </setq>
103             </progn>
104           </if>
105
106           <byteArrayAppend>
107             <getq name="data"/>
108             <castbyte>
109               <plus>
110                 <mod>
111                   <castint>
112                     <getq name="out_temp"/>
113                   </castint>
114                   <int>10</int>
115                 </mod>
116                 <int>48</int>
117               </plus>
118             </castbyte>
119           </byteArrayAppend>
120
121           <byteArrayAppend>
122             <getq name="data"/>
123             <byte>46</byte>
124           </byteArrayAppend>
125
126           <byteArrayAppend>
127             <getq name="data"/>
128             <castbyte>
```

```
                <plus>
                  <mod>
                    <castint>
                      <multiply>
                        <getq name="out_temp"/>
                        <int>10</int>
                      </multiply>
                    </castint>
                    <int>10</int>
                  </mod>
                  <int>48</int>
                </plus>
              </castbyte>
            </byteArrayAppend>

            <setq name="i">
              <int>0</int>
            </setq>

            <while>
              <lt>
                <getq name="i"/>
                <byteArrayLength>
                  <getq name="data"/>
                </byteArrayLength>
              </lt>
              <progn>
                <setq name="number">
                  <byteArrayGet>
                    <getq name="data"/>
                    <getq name="i"/>
                  </byteArrayGet>
                </setq>
                <OUTPUT to="Display1">
                  <ARG>
                    <getq name="number"/>
                  </ARG>
                </OUTPUT>
                <setq name="i">
                  <plus>
                    <getq name="i"/>
                    <int>1</int>
                  </plus>
                </setq>
                <sleep>
                  <int>1000</int>
                </sleep>
              </progn>
            </while>

            <OUTPUT to="Display1">
              <ARG>
                <byte>47</byte>
              </ARG>
            </OUTPUT>

          </progn>
        </if>

        <setq name="j">
          <plus>
            <getq name="j"/>
            <int>1</int>
          </plus>
        </setq>
      </progn>
    </while>

  </chunk>

</progn>
```

付録 I

センサおよびアクチュエータの例

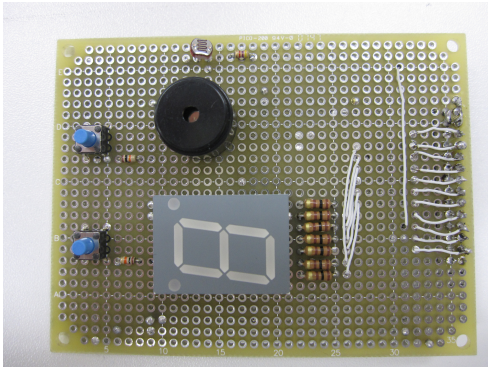

図 I.1　センサおよびアクチュエータ A

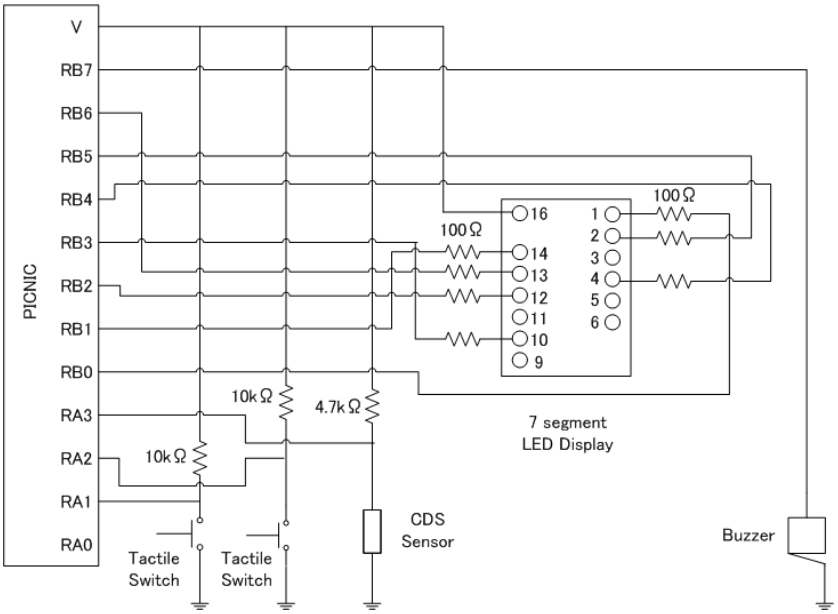

図 I.2　センサおよびアクチュエータ A の回路図

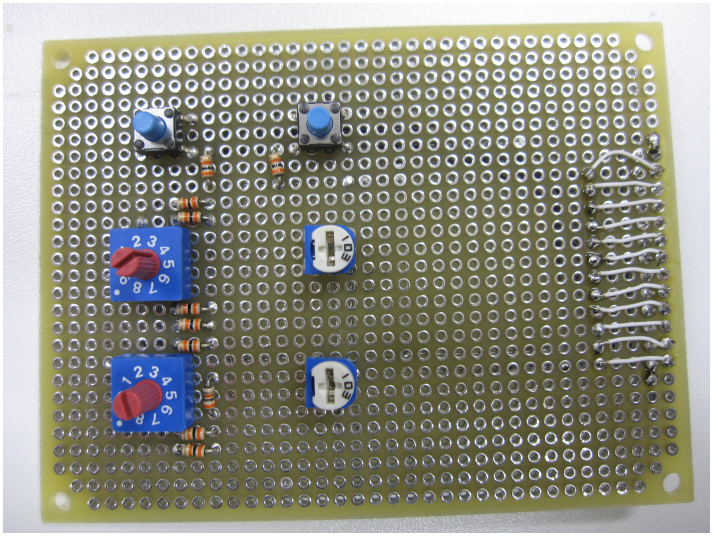

図 I.3　センサおよびアクチュエータ B

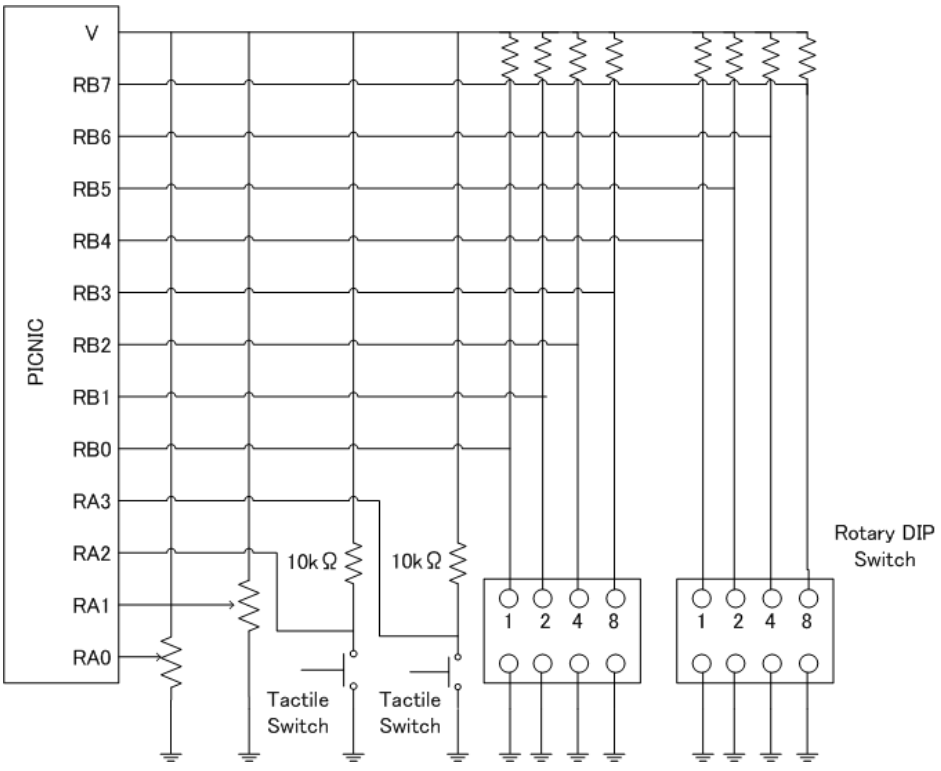

図 I.4　センサおよびアクチュエータ B の回路図

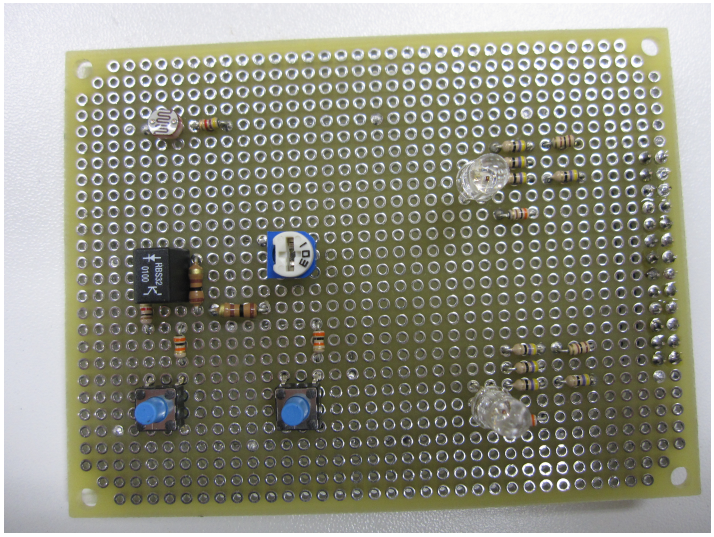

図 I.5　センサおよびアクチュエータ C

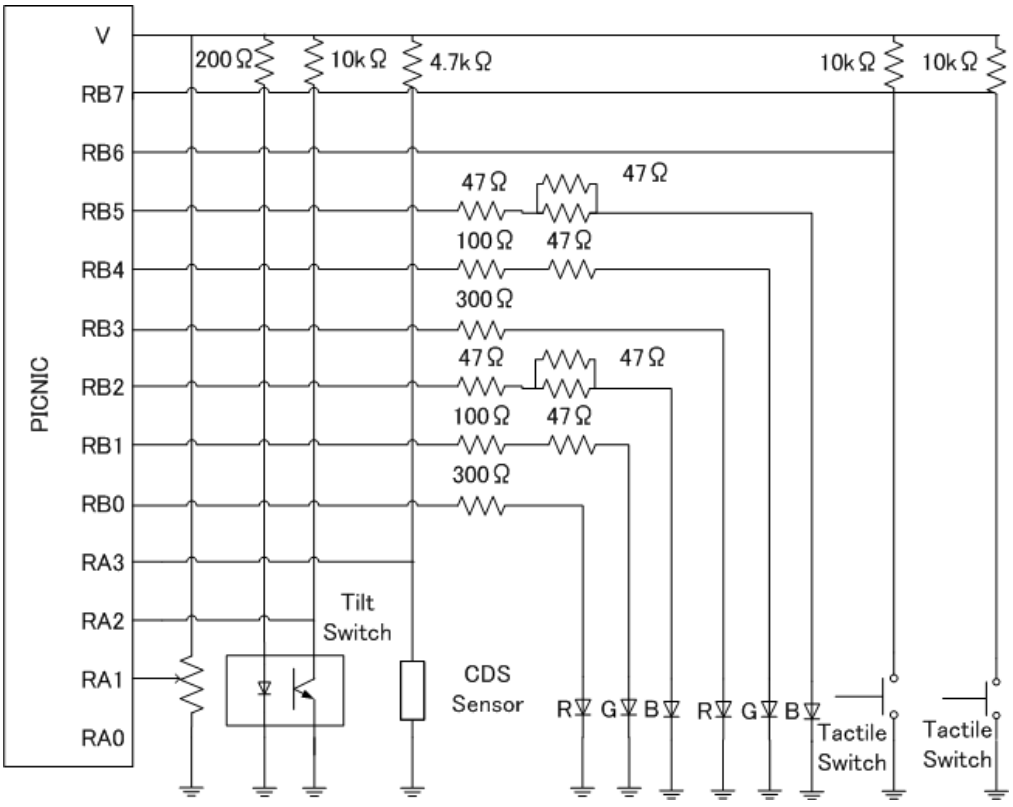

図 I.6　センサおよびアクチュエータ C の回路図