1J-7

Plan9とECMAScriptを用いた分散組込みシステムのプログラミングシステムの提案

盛合 智紀† 佐藤 未来子† 並木 美太郎‡
東京農工大学大学院工学府 †/東京農工大学大学院工学研究院 ‡

1 はじめに

近年の組み込み機器の高性能化，低価格化を背景にセンサーネットワーク等の組込み機器を対象とした分散システムが注目を浴びている．大規模システム向けの分散システムを計算機資源の少ない小型の組込み機器で利用するのは困難であり，新しいプログラミングモデルが求められている．

本研究では，分散システム中に大量に設置される組込み機器を対象として，ノードのプログラマとユーザの双方にとって透過性の高いプログラミングモデルを提供することを目的としている．ノードのプログラマにはシステム全体をプログラミング単位とし，通信プロトコルを隠蔽したアクセス透過性の高いプログラミングモデルを提供し，ノードのユーザにはノードの計算機資源をファイル入出力で利用するためのインタフェースを提供する．

2 システム概要

本システムでは同じネットワーク上に存在するノード，ノードを利用するクライアントにおいて，分散システムの通信プロトコルとしてTCP/IPを利用した9Pプロトコル[1]を用いる．ノード間の協調動作も9Pプロトコルを利用したファイル入出力で実現することで，端末とノードの通信プロトコルとを共通化し計算機資源を節約する．

図1に本システムの概念図を示す．システムの構築方式として，ノードにネットワークファイルプロトコルの9Pプロトコルをサポートした軽量なVM(Virtual Machine)を用いる．このVMが9Pプロトコルのプロトコルスタックを保持し，ノードの通信を管理する．また，ノードの処理内容の記述にはECMAScriptをベースに独自拡張を施した言語を利用する．

クライアントはOSが9Pプロトコルをサポートするシステムを想定している．ノードの計算機資源が仮想化されたファイルをクライアントのファイルツリーの一部にマウントすることでシステムに参加し，リモートマウントによるアクセス透過性の高い方法でノードの利用が可能である．

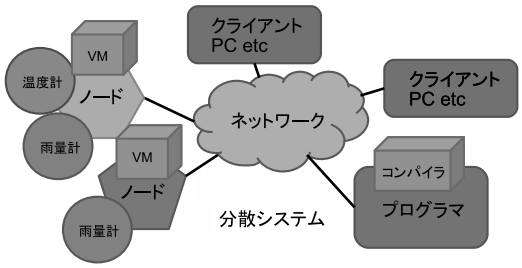

図 1: システム概念図

3 ECMAScriptのインタプリタを有するノードによる分散システムの構築法

ノードのプログラマに分散システム全体を一つのファイルツリーとみなしたプログラミングモデルを提供する．ノードの処理内容を記述する言語としてECMAScriptをベースにしたものを用い，ECMAScriptにはノードの設定やファイルへ仮想化するプロパティを明示する記述の為の拡張を加える．

システム内のノードをオブジェクトと定義し，ノードが行う処理はプロパティで定義する．ECMAScriptにおけるオブジェクトの参照とプロトタイプチェーンにノード間の通信を隠蔽し，ファイルツリーへの操作に対応付けることで，ノードを超えた計算機資源を暗黙的に利用でき，高いアクセス透過性が実現できる．

3.1 ECMAScriptのファイルツリーへの仮想化

表1の対応を元に，ECMAScriptで記述したオブジェクトの各要素をファイルツリーと対応させる．これにより，オブジェクト間の参照を9Pプロトコルで実現する．ファイルツリーがノードを超えて繋がることで一つの大きなファイルツリーとなり，ノードのプログラマは各要素へのアクセスが可能となる．

3.2 分散システムが構築するファイルツリー

プログラマのノードの見え方を図2に示す．プログラマはノードをリモートマウントして利用する．図2に示すようにプログラマがマウントするノードは親ノード一つである．親ノードの下には子ノードがぶら下がる形になり，親ノードのルートディレクトリを頂点としてシステム全体で一つのファイルツリーを形成する．プロパティが仮想化されたファイルに入出力を行うことで計算結果を取得したり値を与える．

オブジェクトディレクトリの下にディレクトリが存在する場合，オブジェクトの継承関係を意味しておりECMAScriptの場合プロトタイプチェーンの対象となる．ノードを超えた継承関係もありえるため，親オブ

表 1: ECMAScriptとファイルツリーの対応

ECMAScript	ファイルツリー
名前空間	ディレクトリ
オブジェクト	ディレクトリ
メソッド	ディレクトリとファイル
プロパティ	ファイル
インスタンス化	ファイルへの仮想化
継承関係	階層関係
プロトタイプチェーン	階層を遡る

Proposal of Programming System for Distributed Embedded System using Plan9 and ECMAScript

† Tomoki Moriai
† Mikiko Sato
Graduate school of Engineering, Tokyo University of Agriculture and Technology
‡ Mitaro Namiki
Graduate school of Engineering, Tokyo University of Agriculture and Technology

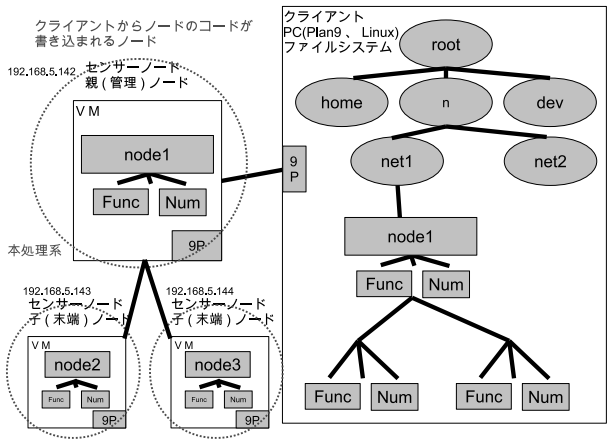

図 2: プログラマから見えるシステムのイメージ

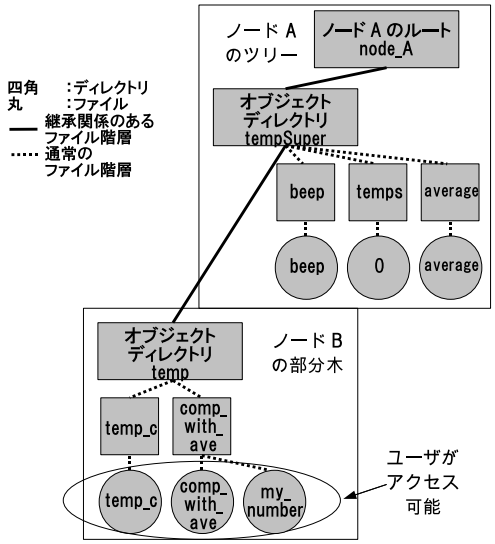

図 4: システムの提供するファイルツリー

ジェクトのプロパティやメソッドへのアクセスもノードを超えて可能とする．継承を利用することで親ノードが保持する計算機資源を透過的に利用し，計算結果だけを複数の子ノードから参照できるようにし，システム全体での計算機資源を節約する．

3.3 ECMAScript による分散システムの記述例

```
/*ファイルに仮想化されない親ノードAのオブジェクト*/
var TempSuper = function() {}
TempSuper.temps = new Array[num_of_nodes];
TempSuper.prototype.beep = function() {...}
TempSuper.prototype.average = function() {
    var temps_sum = 0;
    for (i=0; i<num_of_nodes; i++) {
        temps_sum += this.temps[i];
    }
    return temps_sum / num_of_nodes;
}

/*ファイルに仮想化される手続きを含む子ノードBのオブジェクト*/
var Temp = function() {}
Temp.prototype = new TempSuper();
Temp.temp_c = function () {
    return dev.ad1 * （変換式）;
}
/* ファイルに仮想化する変数を指定するvisible */
Temp.temp_c.visible = true;
/* プロトタイプチェーンの利用例 */
Temp.comp_with_average = function (my_number) {
    TempSuper.temps[my_number] = this.temp_c();
    if ( this.temp_c() > this.average() + 5) {
        this.beep();
    }
}
Temp.comp_with_average.visible = true;
/*ノードとオブジェクトの設置場所の対応もECMAScriptで記述*/
```

図 3: プロトタイプチェーンを利用した例

図 3 にプロトタイプチェーンとオブジェクトの参照を利用したノード間の協調動作を行うコード例を示す．図 3 において，子ノード (Temp) で温度データを取得し，親ノード (TempSuper) で平均温度を計算する．

異なるノード間のオブジェクトを参照する例として，Temp の comp_with_average で親ノードの temps に子ノードで取得した温度データを記録している．子ノードは average オブジェクトをプロトタイプチェーンで参照することで平均温度を利用できる．

Temp ではファイルに仮想化するプロパティを明示する為に拡張した visible が記述されている．visible により，修飾されているプロパティがクライアントにアクセスされる対象のファイルに仮想化される．

図 3 で示したコードにより生成されるシステムのファイルツリーは図 4 のようになる．プログラマは親ノードのルートディレクトリを頂点としたツリーをマウントして利用する．

本研究では，分散システム全体をプログラミングの対象にし，ECMAScript を用いてノードの処理内容をオブジェクトとして記述することで，ノード間の通信を伴う処理の記述をオブジェクトの参照やプロトタイプチェーンに隠蔽する．ノードのプログラマは通信プロトコルの記述や通信待ちなどネットワークプログラミングを考慮せずにノードの処理内容を記述できる．

4 実装

MCF52233 基板上で試作を行い，VM の動作と 9P プロトコルのサーバとしての機能を実装した．Linux からの発行される ls，cd，cat 等のコマンドに応答し，ファイルツリーの提供が可能である．

VM の処理性能は同じマイコン上に実装されている C 言語のインタプリタである SilentC とネイティブコードと動作性能を比較した．処理時間として，本研究の方式はネイティブコードに比べ 30 倍近い実行時間がかかるが，インタプリタである SilentC の 200 分の 1 程度の実行時間で処理できる．センサーネットワークの処理系として十分に高速な実行性能を得られた．

5 おわりに

本研究では，ECMAScript のオブジェクトでノードを定義することで，プログラマから通信プロトコルの記述を隠蔽し，アクセス透過性の高いプログラミングモデルによる分散システムの構築法を提案した．

今後の課題は，9P プロトコルのクライアントとしての機能の実装と，ノード間の連携動作の実装である．

参考文献

[1] Bell-labs: Plan 9 File Protocol, 9P http://plan9.bell-labs.com/sys/man/5/INDEX.html (2008)

[2] Yoshimasa Niwa, Satoru Tokuhisa, Masa Inakage: Talktic: a development environment for pervasive computing applications ACM International Conference Proceeding Series; Vol. 352 pp. 34-41 (2008)