システムソフトウェアと
オペレーティング・システム 71－14
マルチメディア通信と分散処理 73－14
（１９９５．１２．１）

移動計算環境におけるアプリケーション構築のための ソフトウエアアーキテクチャ

保木本晃弘, 中島達夫

北陸先端科学技術大学院大学 情報科学研究科

〒 923-12 石川県能美郡辰口町旭台 15

本稿では、移動計算環境における適応可能なアプリケーションの必要性とそれを安全に構築するためのアプローチについて述べる。

An Approach for Constructing Adaptive Applications in a Mobile Computing Environment

Akihiro Hokimoto, Tatsuo Nakajima

Japan Advanced Institute of Science and Technology

15 Asahidai, Tatsunokuchi, Ishikawa, 923-12, JAPAN

In this paper, we describe our approach for constructing adaptive applications in a mobile computing environment.

1 はじめに

近年、ネットワークのホスト移動透過性を提供する様々なプロトコルが提案されており、実験的に運用が行なわれるようになってきた。例えば[4, 7]などのプロトコルが提案されている。これらのプロトコルは、IPパケットの移動したホストへの配送を支援しており、IPプロトコルにこれらを使用することでTCPやUDPを用いたアプリケーションがそのまま移動ホストで利用可能である。しかしながら、低バンド幅のシリアル通信回線を利用したり、携帯電話回線や無線LANなどの無線通信を利用したネットワークに接続するといくつかの問題が発生する。

1つは、低バンド幅のシリアル通信回線を通るデータは、できる限り減らしたいにもかかわらず、SLIPを使用するとすべてのパケットにTCP/IPヘッダを付加するため、それが通信のオーバヘッドとなる問題である。現在、TCPに関しては、CSLIPやPPPを用いることでこのプロトコルヘッダのオーバヘッドを減らすことができるが、UDPヘッダを減らすための方法は提供されていない。また、経路途中のルータでのパケット紛失によるTCP/IPの制御情報などもバンド幅を浪費する原因となる。2つめは、TCP/IPの実装の問題である。現在のTCP/IPの実装は、物理層における遅延やエラーが発生することを仮定しておらず、大きな遅延変動やパケット紛失は経路途中のルータでの輻輳が原因として処理をしている。このため、物理層での遅延変動により再送タイマがタイムアウトすると、TCPパケットの送信元は、どこかのルータで輻輳が発生したものとして再送を行なう。この時、輻輳崩壊を起こさないように再送タイマのタイムアウトを2倍にしてウインドウを半分に減らしてしまう。このTCPの輻輳制御により、無線通信におけるフェージングや電波状態の悪化を原因とした物理層での誤り制御機構による大きな遅延変動やセルのハンドオフによる一時的な中断が生じると、すぐに反応時間が遅くなってしまう。

このような問題を解決する一つのアプローチとして、間接通信によるネットワークアーキテクチャが提案されている。例えば、I-TCP[1]は、移動ホストと固定ホスト間の通信を移動ホストとMSR[1]間、MSRと固定ホスト間の2つの通信に分割し、移動ホストとMSR間の実装とMSRと固定ホスト間の実装を各通信メディアの特性に合わせて提供することで上記の問題を解決している。また、[5, 10]では、I-TCPと同様のアーキテクチャで移動ホストとMSRの間にフィルタやプロキシを配置することでストリーム毎にバンド幅や遅延の変化などの変化を吸収することを可能としている。これによって、アプリケーションや通信メディアの特性に合わせたデータの提供が可能となる。例えば、[5]では、プロキシにキャッシュ機能を持たせることで、WWWサーバへのアクセスの最適化を行なっている。このような最適化は、ネットワークサービスだけではなく、アプリケーションにおいても必要となる。このため、様々な状況の変化に適応するアプリケーションの系統的な構築法が要求されている。

本稿では、適応可能なアプリケーションを構築するための系統的なアプローチについて述べる。第2節では、適応可能なアプリケーションの必要性とその実現のための問題点について述べる。第3節では、適応可能なアプリケーションを系統的に実現するためのアプローチについて述べる。第4節では、アプリケーションの実現について述べる。第5節では、まとめと今後の課題について述べる。

[1]Mobile Support Router

2 適応可能なアプリケーションの構築に向けて

2.1 適応可能なアプリケーションの必要性

従来のアプリケーションは、実行開始時に一度コンフィグレーションを行なうと、プログラムの実行中に再設定できなかった。このため、プログラムの実行中に実行環境が変化するとプログラムを再起動する必要があった。また、従来の計算機は、使用中にハードウエアデバイスが変更されることはなかった。このため、アプリケーションを予め利用可能なハードウエアデバイス合わせてアルゴリズムや機能などを調整することができた。これは、計算機の利用中に実行環境が変化しないという前提によって可能であった。しかしながら、携帯計算機を含む移動計算環境では、この前提がしばしば崩れてしまう。

この前提が崩れる原因の1つは、PC-Cardインタフェースにある。携帯型計算機は、小さく軽くする要求から、アプリケーションを実行するために必要となるハードウエアデバイスのすべてを予め装備することは困難である。この制約を補うために、PC-Cardインタフェースが装備されている。ユーザは、必要に応じてこのインタフェースにEther、PHS、GPS、フラッシュメモリなどのPC-Cardを挿入することでこれらのデバイスを利用できる。例えば、Etherカードを挿入するとEthernetを介してネットワークと接続可能となる。また、ネット

ワークを必要としない場合には、メモリカードを挿入することで予め装備されている物理メモリを拡張することができる。このようにPC-Cardインタフェースは、携帯型計算機の利用性を向上する反面、上記の前提を崩してしまい、アプリケーションを予め利用可能なデバイスに合わせて静的に調整することを不可能とする。このため、デバイスの状況に合わせて動的に調整可能なアプリケーションが必要となる。

2つめは、計算資源の状況の変化による資源管理方式の変更である。携帯型計算機は、ソケットと内蔵バッテリの2つの電源駆動方式を提供しているが、これはOSの資源管理方式に大きな影響を与える。例えば、内蔵バッテリは、その容量に限りがあるために、消費電力を抑えるためにハードディスクをスピンダウンしたり、無線送信のデータをまとめ送りする資源管理方式を使用する。ところが、ソケットの場合には、容量を気にする必要がないために、消費電力が高くても高速な資源管理方式が使用できる。このため、アプリケーションは、各資源管理方式に適したアルゴリズムを動的に選択できる必要がある。

3つめは、外界の変化による処理や情報の変化である。移動計算環境では、様々な場所や状況で計算機を利用するために、これらの状況に適したサービスを提供した新しいタイプのアプリケーションが要求される[6]。例えば、Active BadgeやGPSによる位置情報を利用して、計算機の前から離れると自動的にスクリーンセイバを起動したり、現在いる場所によってメニューが変化したり、自分の端末の画面やウインドウを他の端末に一時的に表示するアプリケーションが開発されている。今後は、このようなアプリケーションへの要求がより一層増えてくるものと思われる。

2.2 適応可能なアプリケーション構築における問題点

2.1節で述べた状況に適応するアプリケーションは、(1) アルゴリズム、機能、構成などが状況に応じて変化する、(2) ユーザに提供する情報が状況によって変化する、の2つに分けて考えることができる。本稿における適応可能なアプリケーションは、前者の適応を対象としている。

図1は、Ethernet、PHS、Modemなどのネットワークデバイスに適応可能なWWWブラウザの状態遷移を示している。このWWWブラウザは、起動時のデバイスがEthernetならば、Ethernetに適した実装を選択して起動する。もし、実行中にEthernetからPHSに変更されると、PHSに適した実装へと変更される。同様に、EthernetからModemに変更されると、Modemに適した実装へと変更される。このような適応可能なアプリケーションを実現するためには、様々な問題を解決しなければならない。これらの問題は主に安全性と拡張性の問題に集約される。ここでは、これらの問題を仕様と実装の2つに分けて考える。

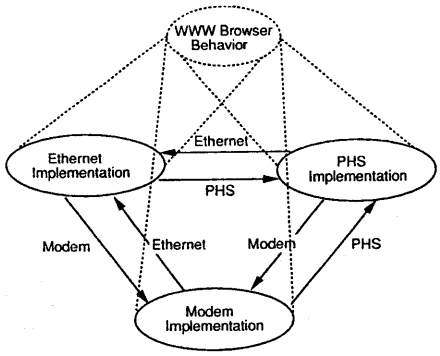

図1: WWWブラウザの状態遷移の例

まず、仕様の問題について考える。適応可能なアプリケーションでは、ハードウエアデバイスや資源管理方式などの変更い合わせてその振舞いを正しく変更することが重要となる。例えば、図1の例において、ネットワークデバイスがPHSに変更されたにもかかわらず、Modemに適したアルゴリズムを選択したり、実行不能に陥るようなことがあってはならない。このような状況に陥らないように仕様の段階で何らかの保証が必要となる。また、既に存在するアプリケーションを新しい状況に適応するように仕様を追加する場合がある。例えば、WWWブラウザにATMや赤外線などのデバイスを追加する必要があるかもしれない。このとき、アドホックな仕様追加や過剰仕様などによって実現が非常に困難となったり、実現不可能な仕様を作成してしまう可能性がある。このような状況を予め検証するための方法が必要となる。

次に、実装の問題について考える。アプリケーションの振舞いを変更するためには、アルゴリズムや機能を実現している関数やプロシジャを変更する必要がある。例えば、EthernetからPHSにデバイスが変更されると、Ethernetのときに使用されていた高バンド幅や有線通信に対応した関数やプロシジャを低バンド幅や無線通信に対応したものに変更する必要がある。この実装は、仕様

における状態遷移に対応するように正しく実装される必要がある。また、アプリケーションは、予測していなかった新しい状況への対応をできるだけ容易に実現することが重要となる。例えば、図1にATMや赤外線などに適応するための実装を追加するためにWWWブラウザをはじめから作り直すような状況は避ける必要がある。しかしながら、従来のアドホックな実装では、アプリケーションの様々な部分の関数やプロシジャに各ネットワークデバイスに対応した処理が分散しているために、新しい状況に対応するための変更が非常に困難である。また、ATMや赤外線に関係ない関数も変更する可能性があるために新しいバグを含みやすくする原因となる。

現在、[2, 3, 9]のような様々な拡張や適応可能なOSサービスやアプリケーションの構築を目標に設計されたプログラミング言語システムやOSが提案されている。これらのシステムでは、拡張や適応のための機構を提供している。しかしながら、現実には、上記のような問題を解決しなければ適応可能なアプリケーションを実現することは困難である。

3 適応可能なアプリケーションを構築するための系統的アプローチ

本節では、適応可能なアプリケーションを安全に構築するためのアプローチについて述べる。

3.1 安全性の検証

我々は、適応可能なアプリケーションを安全に実現するために、少なくとも、(1) アプリケーションが適応可能な状況を示した状態遷移の正当性、(2) その状態遷移の各状態における振舞いの正当性、の2つを検証する必要があると考えている。

まず、状態遷移の正当性の検証について考える。我々は、状態遷移の正当性を保証するために少なくとも次の4つを保証する必要がある。1つは、非決定性の排除である。もし、状態遷移に非決定性の要素があると、デバイスなどに適した処理を選択不能とする。これは、状況に適していない処理を選択したり、実行不能な状況に陥る危険性を高める。2つめは、到達可能性である。状態遷移に現実には起こり得ない状況の組合せを含むことによって到達できない状態が存在するかもしれない。3つめは、他の状態への遷移可能性である。例えば、1度PHSが使用されると2度と他のデバイスを使用できない状況に陥る可能性がある。4つめは、直接遷移の保証である。例えば、EthernetからPHSに状態が遷移するときにModemを介して遷移することはオーバヘッドとなるため避ける必要がある。

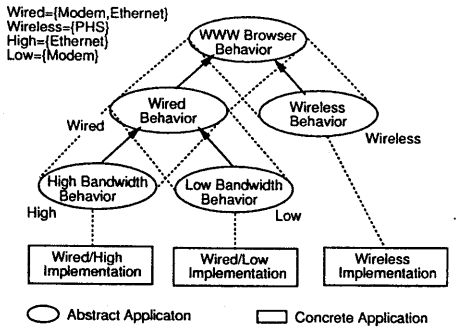

図2: 抽象アプリケーションと具象アプリケーション

次に、状態遷移の各状態における振舞いの正当性について考える。状態遷移の正当性の検証は、デバイスや資源管理方式などの変更が発生したときに正しく振舞いの選択が可能であることを保証するが、各状態の振舞いに関しては保証していない。我々は、適応可能なアプリケーションの振舞いの正当性の検証は、(1) 予測可能な振舞い、(2) 実現可能性などを保証する上で重要であると考えている。デバイスや資源管理方式に適したアプリケーションが正しく振舞っているかどうかの検証は、従来までほとんど議論されていない問題の一つである。しかしながら、我々は、1つのアプリケーションの状態遷移の各状態の振舞いにおいて、ある状態時の振舞いが本質的に他の振舞いと異なる場合には、それは他のアプリケーションと考え、状態遷移の中に追加できないものと考えている。例えば、図1において、Ethernet、PHS、Modemの振舞いは、WWWブラウザの振舞いとしては本質的に同じものであると考えている。このため、これらの振舞いは、同一のアプリケーションとして実現可能である。3.2節では、この振舞いの正当性の検証に対する我々のアプローチについて述べる。

3.2 抽象アプリケーションと具象アプリケーション

我々のアプローチでは、抽象アプリケーションと具象アプリケーションの概念を導入する。抽象アプリケーションは、アプリケーションの実装に関する詳細を省き、単純にアプリケーションの振舞いのみを与える。これに対して、具象アプリケーションは、データ構造、アルゴリズム、構成要素などが固定した実行可能なアプリケー

ションである。

抽象アプリケーションの振舞いは、状態遷移によって表現する。抽象アプリケーションは、状態遷移で表現された振舞いを元に、適応可能な状況に合わせて段階的に詳細化される。この段階的に詳細化された抽象アプリケーションの集まりは、抽象アプリケーション木と呼ばれる。抽象アプリケーション木の一番上にくる抽象アプリケーションはルートと呼ばれ、そこから派生したすべての抽象アプリケーションを代表した振舞いを表現する。ルート以外の抽象アプリケーションは、実行環境や外界の状況に対応した条件を持っている。この条件は、抽象アプリケーションの振舞いを詳細化する時の方針を示している。状況に対応した状態遷移は、抽象アプリケーション木の終端点にある抽象アプリケーションの集まりによって構成される。この状態遷移の条件は、ルートから終端点にある抽象アプリケーションまでの条件によって決定する。これによって構成された状態遷移は、前節の検証を行なうことで安全性を検証する。

振舞いの検証に関して、我々は、同一ルートを持った抽象アプリケーションは本質的に同一の振舞いをするものと考える。これは、木を構成している抽象アプリケーションの振舞いがすべてルートから派生した振舞いであることによる。また、同じルートを持たない抽象アプリケーションは、本質的に振舞いが異なると考え、他のアプリケーションとみなす。もし、新しい状況に対応するために振舞いを追加する場合、その振舞いがルートから派生可能ならばその振舞いを抽象アプリケーションに追加できるが、派生できない場合には、その振舞いをルートの変更なしに追加することができないと考える。このルートの変更は、アプリケーションを別のアプリケーションに変更することを意味し、大幅な変更が必要となる。

具象アプリケーションは、抽象アプリケーション木の終端点にある抽象アプリケーションのインスタンスである。具象アプリケーションは、状況に対応した状態遷移の各状態の実装に対応している。もし、実行環境や状況の変化が発生すると、状態遷移に従い具象アプリケーションが変化することでアプリケーションは状況に適応する。

図2は、抽象アプリケーションと具象アプリケーションの関係の例である。抽象アプリケーションはのルートであるWWW Browser Behaviorは、WWWブラウザを代表する振舞いを与えている。ルートの1つ下の階層にある抽象アプリケーションのWire、Wirelessの条件は、WWW Browser Behaviorを詳細化するための方針である。Wireは、有線通信に適した詳細化であり、Wirelssは、無線に適した詳細化を行なう。さらに、Wiredの下の階層にあるHigh、Lowの条件は、それぞれ高バンド幅、低バンド幅に適した詳細化である。具象アプリケーションは、各終端点の抽象アプリケーションのインスタンスであり、それぞれ、Wired/High、Wired/Low、Wirelessの実装となっている。また、具象アプリケーションの状態遷移の条件は、それぞれ、Wire/High、Wired/Low、Wireless となる。図2では、図1と同様の状態遷移を構成する。

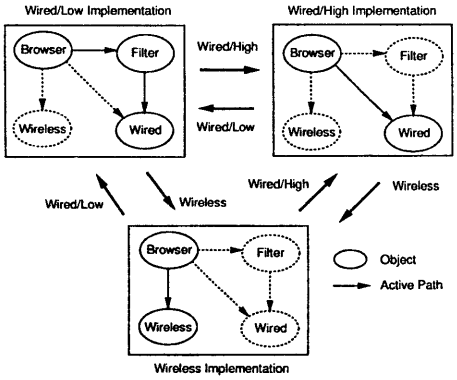

図 3: コンポジットオブジェクトの振舞いの変更

4 オブジェクトコンポジションによる実現

第3節で述べた抽象アプリケーションにより仕様が定義されると、具象アプリケーションを実装する必要がある。我々は、この実装をコンポジットオブジェクト(composite object)[11] によって行なう。コンポジットオブジェクトは、複数のオブジェクトの集まりによって構成されたオブジェクトである。これは、オブジェクトグラフと呼ぶグラフで表現される。オブジェクトグラフは、コンポジットオブジェクトを構成するオブジェクトの参照関係を表現したもので、オブジェクトを節、参照を辺と考えた有向グラフである。オブジェクト間のインタフェースは、予め検証して保証することができる。このアプローチでは、オブジェクトグラフ上のオブジェクトを置換したり、グラフの構成を変更することで振舞いを変更する。

図3は、第3節の図2の実現例である。アプリケーションは、Wired、Wireless、Filter、Browserの4つのオブ

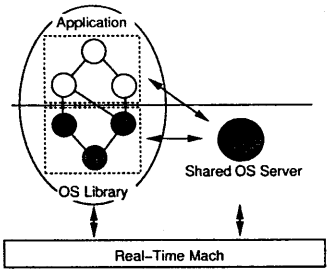

図 4: ソフトウエアアーキテクチャ

ジェクトから構成されている。アプリケーションを起動したときに、Wired/High の状態ならば、オブジェクトグラフ G=(O, R) は、O={Wired, Browser}, R={(Browser, Wired)} となる。もし、Wired/Low に状況が変化するとアプリケーションの状態は、Wired/High から Wired/Low へと変化する。このとき、O={Wired, Filter, Browser}, R={(Browser, Filter), (Filter, Wired)} となる。このようにオブジェクトグラフを変更することで、適応可能なアプリケーションを実現する。

この実装は、Real-Time Mach[8] を用いて図 4 のようなアーキテクチャによって実現している [12]。

5 まとめと今後の課題

本稿では、移動計算環境における適応可能なアプリケーションの必要性と適応可能なアプリケーションを構築するために安全性と拡張性が重要であることを述べた。また、適応可能なアプリケーションを安全に構築するための我々のアプローチについて述べた。この安全性の研究に関しては、現在はじまったばかりであり、非常に多くの課題が残されている。また、実際のアプリケーションに適応するに当たっても様々な課題が残されている。以下は、適応可能なアプリケーションを安全に構築するための課題である。

- アプリケーションの振舞いの仕様の形式化
- 振舞いの検証方法

また、以下は、我々が現在取り組んでいる課題である。

- 現実に使用可能なアプリケーションの構築
- 状況の変化をアプリケーションに通知するイベント配送機構
- アプリケーション毎の OS サービスの最適化
- それぞれの状況における最適なポリシ
- 外界の状況の変化に対する情報の適応

参考文献

[1] A.Bakre,B.R.Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", 15th International Conference on Distributed Computing Systems, May, 1995.

[2] B.N.Bershad, S.Savage, P.Pardyak, E.G.Sirer, M.Fiuczynski,D.Becker, S.Eggers, C.Chambers, "Extensibility, Safety and Performance in the SPIN Operating System", 15th Symposium on Operating Systems Principles, December, 1995.

[3] D.R.Engler,M.Frans Kaashoek and J.O'Toole Jr., "Exokernel: An Operating System Architecture for Application-Level Resource Management", 15th Symposium on Operating Systems Principles, December, 1995.

[4] J.Ioannidis,D.Duchamp,G.Q.Maguire Jr., "IP-based Protocols for Mobile Internetworking", In Proceedings of ACM SIGCOMM, September, 1991.

[5] M.Liljeberg, T.Alanko,M.Kojo,H.Laamanen and K.Raatikainen, "Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach", In Proceedings of SDNE'95, June, 1995.

[6] B.N.Schilit,N.Adams, R.Want, "Context-Aware Computing Applications", In Proceedings of WMCSA'1995, December, 1994.

[7] F.Teraoka, M.Tokoro, "Host Migration Transparency in IP Networks: The VIP Approach", Technical Report SCSL-TR-93-001, January, 1993.

[8] H.Tokuda,T.Nakajima and P.Rao, "Real-Time Mach: Towards a Predicatable Real Time System", In Proceedings of 1st USENIX Mach Symposium, 1990.

[9] Y.Yokote, "The Apertos Reflective Operating System: The Concept and Its Implementation", In Proceedings of OOPSLA'92, October, 1992.

[10] B.Zenel,D.Duchamp, "Intelligent Communication Filtering for Limited Bandwidth Environments", Fifth Workshop on Hot Topics in Operating Systems, May, 1994.

[11] 保木本晃弘, 中島達夫, "状況に依存した計算を実現するオブジェクトモデル", オブジェクト指向計算ワークショップ'95, 日本ソフトウエア科学会, 3 月, 1995.

[12] 保木本晃弘, 中島達夫, "適応可能なアプリケーションを構築するためのソフトウエアアーキテクチャ", 日本ソフトウエア科学会第 12 回大会, 日本ソフトウエア科学会, 3 月, 1995.