システムソフトウェアと
オペレーティング・システム 81－20
（１９９８．５．７）

Inforgent: モーバイルエージェントを用いた情報閲覧支援環境

牧野 聡[1] 大越 匡[1] 中澤 仁[1] 徳田 英幸[1,2]
[1]慶應義塾大学 大学院 政策・メディア研究科 [2]慶應義塾大学 環境情報学部

本研究では，モーバイルエージェントによる情報閲覧支援環境: *Inforgent* システムを構築し，その設計と実装を行った．ユーザの送出したエージェントが計算機間を移動し，移動先計算機上において情報の取得を行う．そして同時に，ユーザのコンピュータで表示再生が可能な形式へと，取得したデータに対して加工や変換の処理を行う．処理を経たデータをエージェントは持ち帰り，ユーザのコンピュータ上でそのデータを表示再生する．本システムの使用により，ユーザは自身の持つ計算機資源や，取得しようとしているデータの形式について意識する必要がなくなり，柔軟性の高い情報閲覧が可能となる．

Inforgent: Agent-Assisted Informaton Browsing System

Satoshi MAKINO[1], Tadashi OKOSHI[1], Hitoshi NAKAZAWA[1] and Hideyuki TOKUDA[1,2]
[1]Graduate School of Media and Governance, Keio University
[2]Department of Environmental Information, Keio University

This report describes an agent-assisted information browsing system, *Inforgent*. The agent dispatched by the user moves towards the host which holds the desired data, and retrieves the informaiton from the host. Then on the same host, the agent processes the data and converts the data suitable for output. After the agent comes back with the processed data, it shows the output in desired way. By using this system, users can be unconscious of their computer resources and the type of the desired data, and can browse information with more flexibility.

1 はじめに

まず，本研究に対する問題意識と目的について述べる．

1.1 問題意識

コンピュータネットワークの発展に伴い，テキスト，画像，音声など様々な種類の情報を蓄積・利用・共有することが容易になった．しかしその結果として，クライアントに依存したデータが多く見られるようになった．すなわち，ハードウェアの表示再生性能に依存したマルチメディアデータや，特定の閲覧ソフトウェアを要求する独自形式のデータが出現している．このようなデータを，携帯端末などのような制約を受けた計算機環境下で閲覧することは困難あるいは不可能である場合が多く，自由な情報の閲覧が妨げられている．

1.2 本研究の目的

本研究では，データに対してあらかじめ付加的処理を行い，処理を経たデータをクライアント側に提供するシステムの構築を目的とする．

ここでの「付加的処理」とは，主にデータからクライアント依存の部分を取り除くための処理を指すが，データの種類やユーザの指定によって様々な処理が考えられる．画像に対するスケーリング処理などの一般的な QOS (Quality Of Service) コントロールにとどまらず，データのサイズやフォーマット，表現形式に対する操作，出力デバイスの変更など多様な例が挙げられる．

また，これらの処理は組み合わせることもできる．たとえば，画像の中の文字情報を認識してその内容を要約してから，さらにこれを日本語に翻訳しクライアント側に送るなどの例が考えられる．

2 情報閲覧支援環境

情報閲覧支援環境として開発されたいくつかの関連研究について検討し，それらの持つ問題点を明らかにする．

2.1 従来環境の特徴

1.1節で述べた問題意識すなわち「制約を受けた環境下での情報閲覧」に基づいた研究はすでにいくつかなされている．具体例としては，KMSF-MCAP[2]，ProxiWeb[3]，DeleGate[4]，情報表現形式自動変換機構[5]などが挙げられる．

これらのシステムは，情報が置かれているサーバとユーザのクライアントとの間に，何らかの中継サーバを配置するというものである．計算機資源に対する制約の少ない環境に置かれた中継サーバにおいて，クライアント上での表示再生に適した形へとデータの加工が行われる．

2.2 従来環境の問題点

KMSF-MCAP や ProxiWeb は，対象とする情報取得プロトコルやサポートするクライアントプラットフォームが限定されている．そのため，新たなプロトコルやプラットフォームに対応にはプログラムの書き換えが必要となる．また，データに対する加工機構がサーバに内包されているため，新たな加工処理への対応にもプログラムの書き換えを必要とする．

DeleGate は汎用のプロトコル中継システムとして使用することが可能であり，任意のフィルタプログラムを加工処理に使用可能である．しかしこのフィルタプログラムはサーバ起動時にパラメータとして入力するため，フィルタ追加時にはサーバの再起動を必要とする．

情報表現形式自動変換機構は，加工処理のためのフィルタプログラムを動的に追加することが可能である．しかし，新規プロトコルに対する柔軟性の欠如という点では KMSF-MCAP や ProxiWeb と同様である．

また,DeleGate や情報表現形式自動変換機構はともに処理機能に関する拡張性を謳っているが，両システムでの処理機能を実現するプログラムはフィルタ (ストリームとして入力を受け取り，処理結果をストリームとして出力する) 形式に限定されている． したがって，この形式に当てはまらない処理 (例: 結果として複数のオブジェクトが生成されるような処理) には対応することができない．

以上の点からまとめられる，既存の情報閲覧支援環境における課題を示す．

プロトコルに対する汎用性

一つあるいは特定のプロトコルへ限定されず，将来的にどのようなプロトコルが出現した場合でも対応できるような機構が必要である．

プロトコルの動的追加

新たなプロトコルに対応する際には，ソースコードの書き換えやプログラムの再起動の必要がないことが要求される．

処理機能に関する拡張性

プロトコルに対する汎用性と同様に，未知の処理機能に関しても対応可能であるように設計されていなければならない．

処理機能の動的追加

これも「プロトコルの動的追加」の項と同様である．処理機能はサーバ内部の機能として提供されるのではなく，外部の機能として動的に追加できることが求められる．

処理形式の柔軟性

従来のフィルタ形式に当てはまらない任意の処理形式にも対応できるような設計がなされるべきである．

クロスプラットフォーム性

一度記述されコンパイルされたプログラムはそのまま他のコンピュータ上でも実行可能であることが望ましい．

よって，本研究では以上の問題点を解決することを目標とする．

3 モーバイルエージェント

本研究で使用するモーバイルエージェント[1]の定義と機能を解説し，本研究におけるエージェントを用いたアプローチの妥当性を明らかにする．

3.1 エージェント概説

エージェントとは，自律性(autonomy),社会性 (social ability), 反応性(reactivity), 自発性(pro-activeness)の 4 特性を備えたソフトウェアあるいはハードウェアに立脚したシステムであるとされている[6].

また,モーバイルエージェントとはエージェントの 1 機能分類であり,「人間の代行として自律的に計算機間を移動しながら，仕事の遂行をする計算主体」と定義することができる．本報告において「エージェント」と言及する際は，このモーバイルエージェントの定義に従う．

3.2 エージェント・アプローチ

2.2節において，従来の情報閲覧支援環境が持つ問題点を指摘した．本研究ではこれらの問題点に対する解決として，情報の取得・閲覧にエージェントを用いたアプローチを提案する．これによって，以下に示す様々な利点を得ることができる．

情報への適応的アクセス

エージェントがデータの形式やサイズについて自律的に判断を行い，適切な付加的処理を行う．したがって，ユーザはデータの形式やサイズについて意識することなく情報の閲覧を行うことが可能である．

クライアント非依存の拡張性

新たなデータ形式が出現した場合でもその違いはエージェント側で吸収されるため，クライアント側に対して変更を行うことなく対応することが可能である．

ネットワーク通信量の削減

通常のサーバ・クライアント間コミュニケーションは，ネットワークを経由してメッセージを交換することによって行われている．しかしエージェントがサーバの存在するホストに移動すれば，ネットワークを経由せずに直接サーバと通信を行うことができ，ネットワーク通信量を削減できる可能性が生じる．

計算機資源の有効活用

計算機資源を豊富に持つ計算機を選択してその計算機上で処理を行ったり，負荷の高い計算機上での処理を回避すること等によって，計算機資源の効率的な活用や負荷の分散を実現できる．

携帯端末に好適

情報取得や付加的処理はすべてサーバ側で行われるため，クライアント側のプログラムサイズを小さくすることができる．また，クライアントはエージェントを送出するときとそれを帰着させるときのみネットワークに接続されていればよい．

処理機能の動的配置

移動先ホストに，期待する処理機能がインストールされていない場合を考える．このような場合でも，エージェントがその機能に対応するプログラムを内包して移動し，移動先ホストでそのプログラムを実行することが可能である．

4 設計

本研究において構築されるシステムの概要，各構成要素の設計，システムの動作手順について述べる．

4.1 システム概要

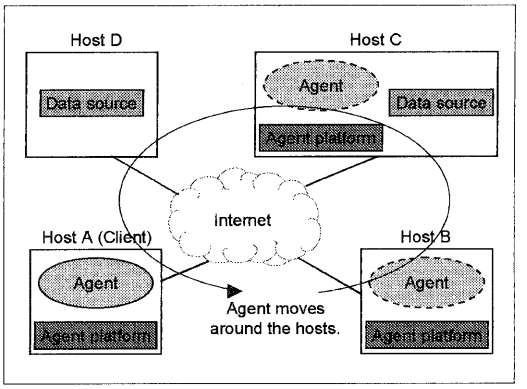

図 1: *Inforgent* システム概念図

本 *Inforgent* システムは，エージェント，エージェントプラットフォーム，データソースの 3 サブシステムから構成される．ユーザはエージェントを生成・送出し，エージェントはデータソースを参照してデータを取得する．エージェントプラットフォームはエージェントの状態

を管理する一方，エージェントは複数のエージェントプラットフォーム間を移動する．

本システムの概念図をに示す．

4.2 システム構成

エージェント本体

エージェント本体は，以下の機能要件を満たすクライアントプログラムである．

1. ユーザから，取得するデータや加工及び表示形式に関する情報を受け取る機能
2. ユーザの端末上でデータを表示・再生する機能
3. 実行状態を保持したままネットワーク上を移動する機能
4. 加工・変換などの処理を行う機能
5. 処理を経たデータを表示・再生するための UI (User Interface) を作成する機能

エージェントプラットフォーム

エージェントプラットフォームは，エージェントの生成や移動ならびに消滅，エージェント間通信などを司るサーバプログラムである．すなわち，エージェントプラットフォームはユーザ側・データソース側のいずれにも存在する．

エージェントの到着を待ち受けるサーバ部分と，内部に存在する各エージェントの状態を管理するモニタ部分から構成される．

処理のためのモジュール

処理のためのモジュールはエージェントプラットフォーム内に存在し，エージェントによって動的にロードされ，処理を実行する．その機能によって以下の 3 通りに分類される．

1. Protocol module
   データソースに対して通信を行い，目的とするデータを取得する．データ取得プロトコルごとに用意される．
2. Process module
   取得したデータに対して加工や変換などの処理を行う．複数組み合わせて使用することが可能である．
3. Presentation module
   処理を経たデータを，ユーザ側端末上で表示再生できる形式へと変換する．

エージェントはこれらのモジュールの中から数個を適宜選択してロードし，その機能を実行する．モジュールの組み合わせによって，複雑な処理も柔軟に実現される．

データソース

データソースは，エージェントからのアクセスを受け付け，リクエストに応じてレスポンスを返すサーバプログラムである．具体例としては WWW や FTP 等のサーバが挙げられる．

データソースに対するアクセスは前述の Protocol module を介して行われるため，既存のサーバを改変なしにそのまま利用可能である．

4.3 動作手順

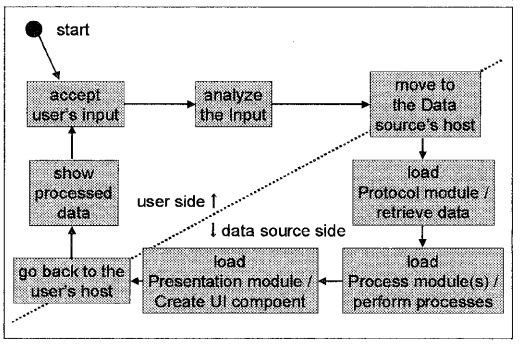

図 2: 動作手順概略図

まず，ユーザは自ホスト上でエージェントプラットフォームを起動し，これを通じてエージェントを生成する．そして，取得するデータや期待される加工・変換処理，表示形式を指定する．

エージェントは対象とするデータが存在するホストの，エージェントプラットフォーム上に移動する．そして，情報取得のプロトコルに対応した Protocol module をロードし，これを通じてデータソースからデータを取得する．次にエージェントは必要に応じて，指定された加工・変換処理に対応した Process module をロードし，その処理を行う．次に，エージェントはユーザの指定に応じて適切な Presentation

module をロードし，UI コンポーネントを生成する．

最後にエージェントは生成されたコンポーネントを保持してユーザ側ホストに戻り，表示・再生を行う．

動作手順の概要を図 2に示す．

5 実装

本システムの実装方針と，各構成要素に対する実装の概要について述べる．

5.1 実装方針

まず 1 点目に，エージェントにはクロスプラットフォーム性が求められる．エージェントはネットワーク上の様々なホスト上を移動するため，OS やハードウェアに依存した実装は望ましくない．したがって，実装には Java 言語を採用した．

2 点目には，クライアント側の改変を必要としない機能拡張性が重要である．すなわち，モジュールによる処理がサーバのみで完結するようにし，新たに処理機能を追加した場合でも既存のクライアント (エージェント) に対しての改変や追加を必要としないように実装した．

5.2 実装概要

エージェント本体

```
arrive() {
    if(現在のホスト名 == ユーザのホスト名) {
        <画面表示>
    } else {
        <情報取得・加工処理>
    }
}
```

図 3: arrive() メソッド (擬似コード)

エージェント本体は，AgentSpace システム [7]によって提供される Agent クラスを継承 (extends) したものである．

エージェントが移動先ホストに到着すると，コールバックメソッド arrive() が呼び出される．そこでこのメソッドをオーバーライドし，各ホストごとの処理内容を記述する．このメソッドの擬似コードによる記述を図 3に示す．

エージェントプラットフォーム

エージェントプラットフォームとしては，AgentSpace システムによって提供される AgentServer クラスをそのまま使用した．

処理のためのモジュール

3 種のモジュールのそれぞれに，Java 言語インターフェースを設定し，すべてのモジュールは該当するインターフェースを実装 (implements) するように定めた．エージェント本体は，これらのインターフェースに対してメソッド呼び出しを行うため，個々のモジュール名をプログラム中にハードコーディングする必要はない．

6 評価

本システムに対する評価を，定量的評価と定性的評価に分けて述べる．

6.1 定量的評価

本システムに対する定量的評価として，エージェントを使用せずに，ユーザ側ホストで情報取得や加工の処理を行った場合との所要時間比較を行った．具体的には，HTTP プロトコルを使用して取得したテキストデータに対して，特定の条件を満たす行のみを抽出するという処理を行った．結果を表 1に示す．

表 1: 性能測定 (単位: ミリ秒)

取得データ量	10KB	100KB	1MB
Inforgent 使用	2135	2349	4469
Inforgent 不使用	374	2421	23321

この結果から，本システムを使用しない場合の所要時間は取得データ量にほぼ正比例して増大するが，使用した場合の増加率は非常に小さくなることが読み取れる．この理由としては，サーバ側の豊富な計算機資源を利用して加工処理が行われることによる処理時間の短縮，サーバ側であらかじめ加工処理が行われることによるネットワーク通信量の削減などが考えられ

る.

したがって *Inforgent* システムを使用した場合, その処理内容によっては大幅な処理時間の短縮が可能となることが実証された. 一方データ量が小さい場合は本システムを使用しない方が所要時間は短くなるが, 3.2節で述べたエージェントによる利点とのトレードオフとなる.

6.2 定性的評価

Inforgent システムは, 従来の情報閲覧支援環境に見られた問題点(2.2節参照)参照を解決した.

本システムにおいて, データの取得や加工・変換などの処理機能はすべてモジュールとして, エージェント本体から切り離されている. したがって, 新たな処理機能を実現するためには, 対応するモジュールを作成してサーバ側に追加するだけでよく, 柔軟かつ動的な拡張性が実現される

また実装に Java 言語を使用することによって, 加工処理形式の柔軟性 (任意のオブジェクトを使用可能), クロスプラットフォーム性も実現される.

以上の点について, 関連研究との比較とともにまとめたのが表 2である. なお,K は KMSF-MCAP, P は ProxiWeb, D は DeleGate, J は情報表現形式自動変換機構, I は本研究をそれぞれ表す.

表 2: 関連研究との比較

↓比較項目 / 対象システム→	K	P	D	J	I
プロトコルの汎用性	×	×	○	×	○
プロトコルの動的追加	×	×	×	×	○
処理機能の拡張性	×	×	○	○	○
処理機能の動的追加	×	×	×	○	○
処理機能の柔軟性	×	×	×	×	○
クロスプラットフォーム	○	×	×	×	○

一方本システムを使用することによるデメリットとしては, エージェント本体がネットワーク上を移動することによるオーバーヘッド, エージェントの移動先ホスト上でエージェントプラットフォームが動作している必要があることなどが挙げられる.

7 おわりに

本研究では, ネットワーク上の情報を閲覧するための支援環境として *Inforgent* システムを提案し, 設計と実装ならびに評価を行った, 本システムによってユーザは, 自らの計算機資源や対象とする情報のデータ形式による制約について意識することなく, 柔軟な情報閲覧を行うことができる. また同時にモジュールの実装者も, 実装の柔軟性と動的な拡張性を得ることができる.

今後の課題としては, セキュリティの強化, 耐故障性の向上などが挙げられる.

参考文献

[1] 岩井俊弥, "Java モバイル・エージェント", ソフト・リサーチ・センター,(1998).

[2] 牧野,由良,原嶋,金平,大越,中沢,徳田, "Keio Media Space Family-MCAP システム", 情報処理学会コンピュータシステムシンポジウム論文集,pp.83-88,(1997).

[3] ProxiNet Inc., *ProxiWeb*, http://www.proxinet.com/proxiweb/, 1998.

[4] 佐藤豊, "プロトコル中継システム Dele-Gate の開発", 電総研研究速報,TR-94-17,(1994).

[5] 林周志, "インターネット上での情報表現形式自動変換機構の実現", 慶應義塾大学政策・メディア研究科修士論文,(1996).

[6] M. Wooldridge and N. R. Jennings, Agent Theories, Architectures and Languages: A Survey, In *Proc. of the ECAI-94 Workshop on Agent Theories, Architectures and Languages*, 1994.

[7] 佐藤一郎, "モーバイルエージェントシステム AgentSpace", http://www. islab.is.ocha.ac.jp/agent/,(1998).