

解説

ネットワーク社会を支援する新しい知能メディア技術

2. モバイルエージェントとネットワーク

Mobile Agents and Network Applications by Ichiro IIDA and Takashi NISHIGAYA (NetMedia Laboratory, FUJITSU LABORATORIES LTD.).

飯田 一朗¹ 西ヶ谷 岳¹

¹ (株)富士通研究所 ネットメディア研究センター

1. はじめに

インターネットの急速な普及と携帯電話に代表される無線通信インフラの拡大にともない、ユーザのネットワーク環境は多様化し、サービスの可能性はますます広がっている。しかしこのような状況は、情報の洪水や通信手段の多様化を招き、ユーザにとって本当に使いやすいネットワークに必ずしもなっていないという問題が一方では発生しつつある。理想としては、場所によらず同じやり方でだれでも簡単にアクセスでき、しかもユーザの特性に応じて情報の選択を行ってくれたり、あいまいな指定でも間違えずに相手とつないでくれたりしてくれるようなネットワークが望ましい。このためには、個々のネットワークと各ユーザの間に立って個別にめんどうをみってくれる個人秘書のような機能が必要であり、エージェント技術に代表される知能情報処理技術が今後ますます重要になると考えられる。

本稿では、こういったネットワークの使い勝手を向上するエージェント技術、なかでもモバイルエージェント技術に焦点をあて、技術動向と今後の展望を議論している。

2. エージェント技術

2.1 エージェントとは

エージェントは、現在非常に広い意味で使用されており、明確な定義が難しい状況にある。エージェントとは、代理人という意味で使われる言葉であり、人間になり代わって自律的に仕事をしてくれるソフトウェアはみなエージェントと呼ぶことが可能である。ここでは、『独立性の高いファンクションごとにカプセル化したソフトウェアブ

ロセスで、コミュニケーションのためのインタフェースが規定されているもの』という広い定義で捉える。エージェントには、専門分野に応じて大きく分けて以下の3つのタイプがある。

(1) 人工知能分野

従来より分散 AI の分野で使われている概念であり、思考、意思決定、学習など人間の知能を代行してくれるものをここに分類する。インテリジェントエージェントと呼ばれることもある。情報検索などのアプリケーションがこの代表といえる。

(2) ヒューマンインタフェース分野

ヒューマンインタフェースを高度化し、コンピュータを擬人化した高度なインタフェースをもつものが考えられるようになっている。表情をもったコンピュータや人工生物のような感情をもつコンピュータを目指した分野である。

(3) 分散処理システム分野

ネットワークでつながった独立性の高いプロセス同士が、平等な関係で協調しあうシステム(マルチエージェントシステムと呼ばれる)。特定の機能を代行するものでインテリジェンスは前二者に比べ高くないのが普通である。

以下ではネットワークへの応用という観点から、とくに(3)のカテゴリに分類される技術に焦点をあてる。なかでも、どこでも実行できるコード(モバイルコード)を利用して、負荷分散やダウンロードなどを実現するモバイルエージェント技術について、概要、記述言語、ネットワーク応用について順に述べていきたい。

2.2 エージェントの利点

エージェントを考えるにあたって、従来のパラダイムであるクライアントサーバモデルと比較し

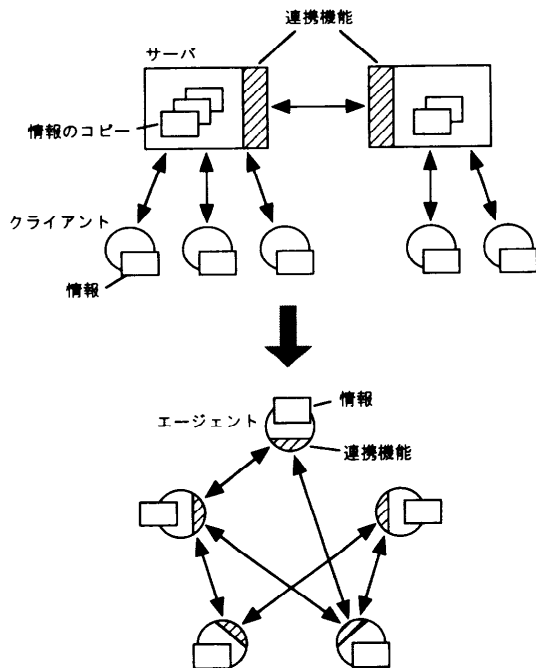


図-1 クライアントサーバモデルとマルチエージェントモデル

て考えるのがわかりやすい。図-1に、クライアントサーバモデルとマルチエージェントモデルの違いを模式的に示す。

現状のクライアントサーバモデルは、サーバの物理構成が固定化されている点に大きな制約がある。大規模構成になった場合にはサーバの分散化が必須であるが、特定のサーバに負荷が集中したとき、これを効率よくサーバ間あるいはクライアント間で分散させることが困難である。しかも機能がサーバに集中しているため、サービスの追加変更、クライアントの追加削除をシステムを止めずに行うことが難しい。

これに対し、マルチエージェントシステムでは、仕組みの上からはクライアント、サーバといった主従関係がなく、クライアントとサーバの機能を自由に構成することができるため、とくにN:Nの通信が頻繁に行われるようなアプリケーションや、ユーザごとのカスタマイズが必要なシステム、ユーザ数がダイナミックに変化するようなシステムにおいてはきわめて有効なシステムである。

これに加えて、エージェントがモバイルコードで記述される場合には、エージェントをネットワークの中でダイナミックに移動させていくことができるため、すいているリソースを探して処理

を委託したり、サーバ機能を端末に一時的にオフロードしたりといったことが可能になり、ネットワークのリソースを有効に利用することができる。インターネットの環境では負荷が集中するサーバに比べクライアントはリソースが遊んでいる時間が多いにもかかわらず、サーバのボトルネックでアクセス性が犠牲になっている場合が多い。ネットワーク上のリソース全体にわたって負荷分散が可能なモバイルエージェント技術はこれからのインターネットにおける重要な要素技術の1つになるものと考えられる。

2.3 エージェントの問題点

このようにエージェントは、将来のインターネット技術において有望な技術であるが、現時点では実用的にまだいくつか問題点を残している。以下主要な3つの課題について述べる。

(1) セキュリティ

最も大きな課題は、セキュリティである。モバイルコードが自分のホストに飛んできて仕事をやって戻っていくという仕組みは、コンピュータウイルスやワームが活躍するための格好の環境である。一般的には、認証によりエージェントの素性をチェックするとともに、エージェントの実行に制約を課するのが普通である。特定メモリや外部ファイルへのアクセス禁止、CPUパワーの制限、エージェントの寿命などを規定する方法がある。また、コードを移動させるときに盗聴されることを防止するためには、エージェント自身を暗号化して送る仕組みを組み込むことも必要になる。

(2) リソースの所在管理

エージェント導入の目的は、個々のリソースのアドレスを隠蔽することにより操作性をあげることである。たとえばあるサービス機能を指定することで、それが実行できる最も近いリソースをアクセスする仕組みを実現する機能などである。また、エージェント間の通信をサポートするためには、目的とするエージェントがどこにあるのかを検索するための上位レベルでのディレクトリ管理が必要である。したがってせっかくエージェントで処理を分散したにもかかわらず、この検索処理に負荷が集中してボトルネックにならないような考慮が重要である。

(3) 性能

マルチエージェントシステムは、その構成上、

集中型のシステムに比べて通信処理のオーバーヘッドが大きくなりがちである。また、モバイルコードに対応させるためには、実行時に解釈しながら処理するインタプリタ型言語にする必要があり、バイナリ言語まで変換するコンパイラ型言語に比べて処理速度が遅くなることは避けられない。したがってこのマイナス要素を上回るダイナミックな負荷分散効果を達成しないとメリットが出てこないわけである。そのためにはネットワーク全体で負荷を自由に移動できるアーキテクチャを早急に確立する必要がある。また、場合によってはバイナリまで落とすなどの上手な組合せも必要になってくるであろう。

3. エージェント記述言語

エージェントを記述するためのプログラミング言語は、いくつか世の中に存在している。以下では、各言語の特徴を紹介しながら、エージェント指向プログラミングに必要とされる概念を考察する。また、Sun Microsystems の Java¹⁾は、もともとエージェントを記述するために開発された言語ではないが、ここでは Java を含めて考察する。

3.1 April

複数のエージェントが協調動作を行う場合の最も一般的な、あるいは古典的な仕組みはメッセージパッシング・モデルである。この場合、メッセージはエージェント間に生成された通信チャネルをデータグラムとして送受信される。通常エージェントは受信したメッセージの意味を解釈し、特定の処理を実行する。したがって、エージェントをプログラミング言語で定義する場合には、受信メッセージを特定の処理に受け渡す処理によりエージェントの振舞いを規定する。

April²⁾はメッセージパッシングをサポートした代表的なエージェント記述言語であり、このメッセージ処理の部分が非常に柔軟に定義できるのが特徴である。たとえば、April におけるメッセージ受信部分の定義は以下ようになる。

```
repeat {
  (find, string?keyword)->{/* find の処理 */}
  lprint -> {/* print の処理 */}
  ltrue -> {/* その他の処理 */}
} until quit;
```

このように April では、「パターン」->「処理」

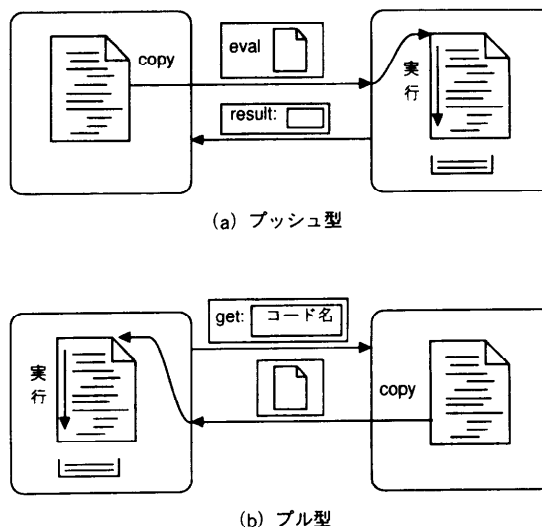


図-2 リモート・エバリュエーション

のような簡易な記述でよく、またメッセージのパターンが柔軟に定義できる。メッセージ送信の動作も以下のような簡易な記述で定義できる。

(find, "april") >> receiver;

ここで、(find, "april") の部分がメッセージであり、receiver はメッセージ送信先エージェントのリファレンスを示す。

April はメッセージパッシングに加えて、リモート・エバリュエーション (Remote Evaluation)³⁾の機能もサポートしてしている。リモート・エバリュエーションとは、あるエージェントがもっているプログラムの一部を別のエージェントに実行させる機能をいう。一般に、受信メッセージに応じて実行される処理の部分はエージェント内にあらかじめ用意しておく必要があるが、リモート・エバリュエーションの場合は受信側エージェントに実行すべきプログラムを用意しておく必要がないため、通常メッセージパッシングに比べてより柔軟な処理を定義できる(図-2 参照)。

リモート・エバリュエーションには別エージェントに対して非同期にプログラムを送りつけるプッシュ型(あるいはアップロード型)と、ほかのエージェントからプログラムを呼び込んで実行するプル型(あるいはダウンロード型)がある。April の場合には、メッセージのパターンにプログラムコードを含めることができるため、プッシュ型・プル型のいずれのリモート・エバリュエーションも容易に記述することができる。一方、Java に

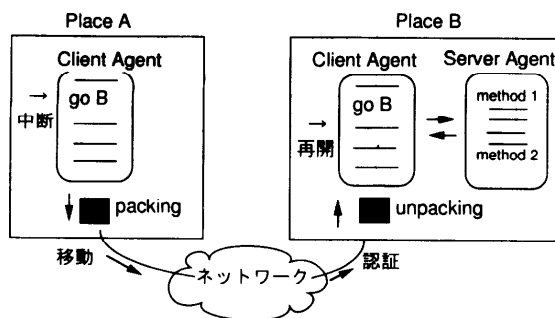


図-3 Telescript におけるエージェントの移動

における applet はブル型リモート・エバリュエーションの代表例ということができる。

3.2 Telescript

Telescript⁴⁾はネットワークで繋がったコンピュータ間をエージェントプログラムが自由に動き回ることができる機能を実現した言語である。これを実現するため、Telescript では移動前にエージェントの実行状態と実行コードを1つのデータグラムにパッキングし、移動先コンピュータ上で実行状態を復元したあとと停止した直後から実行コードを再開する機能が、インタプリタエンジン上に実装されている(図-3 参照)。April や Java によるリモート・エバリュエーションとの違いは、移動後のプログラムが必ずプログラムの先頭から実行される必要がないことである。したがって、リモート・エバリュエーションに比べてより自由な記述が可能となる。

Telescript では、Place と呼ばれる移動不可のプロセスがネットワーク内に分散しており、各々に一意に識別される識別名が割り当てられる。移動コードである Agent は必ずどれかの Place に属しており、Agent 移動の際は Place の識別名や Place の種類により移動先が決定される。たとえば、PlaceFoo という Place に移動する場合には、

```
self.go(Ticket(PlaceFoo, nil, nil, nil, nil, nil));
```

のように記述する。ここで、Ticket オブジェクトがエージェントの移動先を決定する。

Telescript のエージェント間協調の仕組みは、メッセージパッシングではなくメソッドコールにより実現される。たとえば、PlaceFoo に存在する AgentBar に以下のようなメソッドが定義されているとする。

```
whatIs: op(word: String) String = {
```

```
return dictionary.lookup(word);
```

```
}
```

この AgentBar と協調したい場合は、go() で PlaceFoo に移動した後、AgentBar に対してメソッド whatIs を発行すればよい。

メッセージパッシングが基本の April とは、一見、異なったモデルのように思われるが、メソッド名とその引数をメッセージと捉えたと、両者の違いはほとんどないといえる。メッセージベースのエージェントでは、受信するメッセージは何らかの処理に受け渡されるが、この処理の部分がメソッドの実装部分に対応するわけである。

Telescript 向きのアプリケーションは、セールスマン・エージェントであろう。セールスマンはユーザにカタログをみせ、商談交渉をして、ユーザ間を渡り歩く。このようなアプリケーションではセンタとなるホストとのインタラクションが必要ないため、クライアントサーバモデルに比べて、センタへの負荷が小さくすむ。逆に、クライアントサーバモデルに比べてデータ転送のオーバーヘッドが大きいため、センタとのインタラクションを頻繁に行うアプリケーションには向かない。

3.3 Java

Java は C++ に似たマルチプラットフォームのインタプリタ言語で、従来、スタンドアロンのアプリケーションや WWW サービスを拡張する applet の開発のために用いられてきた。したがってエージェントのように複数のホストに分散したプロセスが連携するアプリケーションを記述するためには、直接ソケットをアクセスするとともに、エージェント間でメッセージをやりとりするコードを記述しなければならないため、April や Telescript に比べて余計な労力を必要とする。しかし、JavaRMI⁵⁾や HORB⁶⁾など Java を拡張するライブラリが発表され、いわゆる Remote Method Call(RMC)、あるいは Remote Method Invocation(RMI)と呼ばれる分散オブジェクト環境の通信機能を用いて、エージェントらしいプログラミングが可能になりつつある。たとえば RMI のサーバオブジェクトをエージェントとして捉えたと、

```
RemoteAgent ra = (RemoteAgent)
```

```
Naming.lookup("//host.domain/agentbar/");
```

```
String s = ra.whatIs("Agent");
```

のように Telescript に似たメソッドのやりとりを基本にするエージェントが定義できる。エージェントの名前は URL(Universal Resource Locator)をベースにしているため、Internet との親和性が高い。また、Telescript や April で実現されている双方向のモバイルコードの機能を、Java をベースにして実現しようとする試みも行われるようになっている⁷⁾。

以上、代表的な3つのエージェント記述言語について述べたが、現時点では標準となるべき言語が固まっているわけではなく、一長一短である。April は、もともとマルチエージェントシステムを指向した言語のため、エージェントの記述性やコミュニケーションのサポートに関しては一日の長がある。Telescript はリモートプログラミングの視点からは、自分自身の移動を実行途中で起動できるというほかにはない機能をもっており、セキュリティに関しても最も進んでいる。Java はもともとエージェントを指向した言語ではないが、近年の機能拡張で、先行している April や Telescript に近づきつつあり、エージェントの実行環境としても無視できない存在となっている。

4. エージェント技術とネットワーク応用

ここでは、エージェントの特性をネットワークサービスへ適用する試みを紹介する。最近の移動電話の普及には目を見張るものがあり、モデムを介して屋外でネットワークにアクセスして電子メールの読み書きをする光景もそれほど珍しくなくなってきた。今後情報端末に無線データ通信が一体化されると、オフィスの環境とそれほど変わらない条件でネットワークを利用できるようになるものと考えられる。しかし、携帯端末をもって常にその端末を使わなければならない、という制約を設けるのは、ユーザに別の意味での不便さを強いるものである。無線の通信品質は有線に比べて低

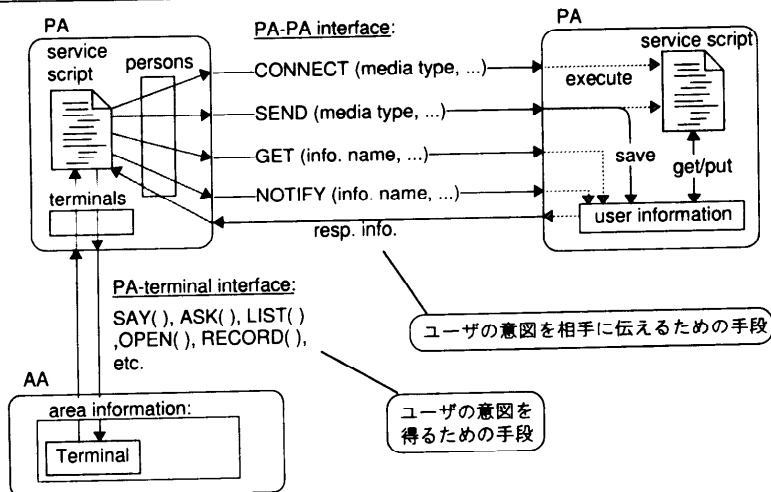


図-4 パーソナルエージェントのコミュニケーションインタフェース

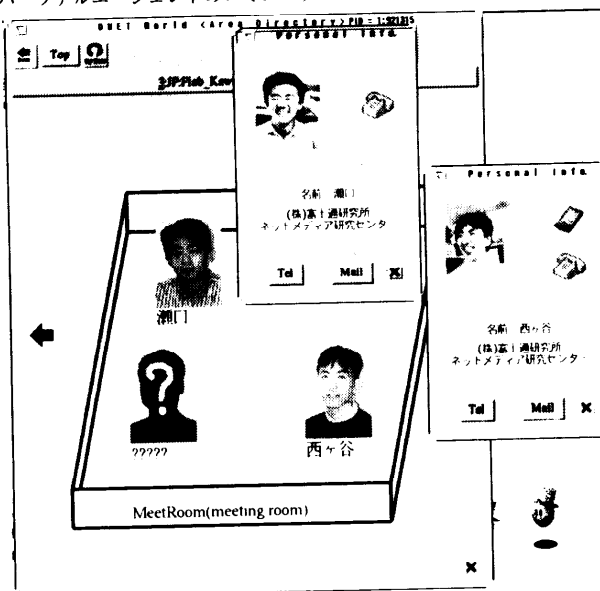


図-5 DUET サービスの表示画面

く、また携帯端末はヒューマンインタフェース的には使いづらい。したがってユーザは、場所に応じて端末を使い分けるのが一般的になっていくだろう。こういった状況でコミュニケーションするときに問題になるのが、現在相手の人がどういう環境なのかを悩まなくてはならず、それに応じたアドレスや番号を指定しなければならないことである。理想としては、相手の名前を指定すればそのとき最も適したネットワークと端末を自動的に選択してほしい。このように、異種ネットワークにまたがって最適なサービスをシームレスに提供するために、ユーザの視点でこれらを隠蔽するエージェント技術が重要になる⁸⁾。

ここでは我々のグループで以前より進めているパーソナルエージェントシステム DUET^{9),10)}のアーキテクチャを紹介する。コミュニケーションの本来の目的は、使う端末やネットワークには直接関係せず、「〇〇君にこの伝言を伝えてくれ」とか「至急××さんと連絡とって」といった抽象的な指示をするのが理想である。有能な秘書はこのようなあいまいな指示を的確にこなして最適な対応をするわけである。

DUET では、ユーザ 1 人 1 人にパーソナルエージェント (PA) を割り当て、PA がそれぞれのユーザを専門に管理し、ほかの PA と協調しながらコミュニケーションを図る。図-4 に PA の構成とコミュニケーションインタフェースを示す。ユーザは、相手が今どういう環境にいるのか、どの番号でつながればいいのか、といったことに煩わされることなく、自分の PA に要求すれば最適なコミュニケーションが提供される。

図-5 に DUET の画面表示例を示す。これは、相手の現在の状態を教えてくれているところであり、ユーザは相手の名前を指定するだけで、電話番号や IP アドレスなどを指定することなく相手にとって最も都合のよいコミュニケーション手段を選んで接続することができる。

こうした個人専用のエージェントを用意することで以下のメリットが得られる。

- (1) ユーザごとにカスタマイズしたサービスの指定ができる。
- (2) 個人情報のエージェントにカプセル化することにより、アクセス権のチェックなどセキュリティ機能の強化が図れる。
- (3) モバイルコードで記述されるため、ユーザが移動するのに応じて PA も移動させ、アクセス性の向上と負荷の分散がはかれる。

PHS データ通信のサービスが来年より予定され、屋外から高速にインターネットをアクセスできる環境が整いつつある。PHS の基地局の位置を利用すれば、屋外にいてもユーザがどのあたりからアクセスしているかがシステムで把握することができるわけであり、現在 DUET で実現しているサービスは容易に屋外に拡張することができる。そうすれば、オフィスだけでなく、外回りに行っている場合にも連絡がとれるなど、利便性が高まる。また、こういったシステムは、ともしれば

ば人のプライバシーを侵害することになりがちであるが、エージェントを導入することにより、場所や時間、相手に応じて自分の情報を隠したりすることができる (図-5 参照)。こういった個人ごとのカスタマイズ機能を容易に実現できることがエージェント技術の最大の利点である。

また、モバイルコードはワイヤレス通信環境でも効果を発揮する。通信品質の変動が顕著な無線では、回線を張りっぱなしでメッセージをやりとりするより、コードを送りつけてローカルに処理をやらせるほうが効率がよいし、エンドエンドのプロトコル処理でスループットを低下させるより、無線区間はプロトコルを分離したほうがよい場合もある。今後移動通信とインターネットをからめたアプリケーションを開発するにあたって、エージェントは重要な技術になるであろう。

5. おわりに

エージェントについて、分散処理システムの切り口から整理してみた。エージェントの技術は、システムのセキュリティ、信頼性をどう確保するか、既存システムや異種エージェントとの整合性をどう解決するかなど課題は山積みである。オブジェクト指向技術が最近ようやく市民権を得たのと同様、エージェントが根づくまでにはまだ相当の期間が必要と思われる。その間の地道な研究の継続が重要である。

参 考 文 献

- 1) Gosling, J. and McGilton, H. : The Java Language Environment, A White Paper (1996).
- 2) McCabe, F. G. and Clark, K. L. : Programming in April - An Agent Process Interaction Language, Draft Book (1996).
- 3) Stamos, J.W. and Gifford, D. K. : Remote Evaluation, ACM Trans. on Programming Languages and Systems, Vol.12, No.4, pp.537-565 (1990).
- 4) White, J. E. : Mobile Agents (1995).
- 5) Java Remote Method Invocation Specification, Revision 0.9 Draft (1996).
- 6) Hirano, S. : The Magic Carpet for Network Computing: HORB Flyer's Guide (1996).
- 7) Lange, D. B. and Chang, D. T. : IBM Aglets Workbench, A White Paper, Draft (1996).
- 8) 飯田：パーソナル移動通信のためのソフトウェア, 信学誌 Vol.78, No.2, pp.150-155 (1995).
- 9) 西ヶ谷他：エージェント指向ネットワークアーキテクチャ DUET の提案, 信学論 B-1, Vol.J79-B-1,

No.5, pp.216-225(1996).

- 10) Iida, I., Nishigaya, T. and Murakami, K. : DUET : An Agent-based Personal Communications Network, IEEE Communications Magazine Nov., pp.44-49 (1995).

(平成 8 年 11 月 5 日受付)



西ヶ谷 岳

昭和 63 年静岡大学工学部電気工学科卒業。平成 2 年同大学院修士課程修了。同年(株)富士通研究所入社。以来、通信ネットワーク、パーソナル通信、エージェント技術などの研究開発に従事。平成 7 年電子情報通信学会学術奨励賞受賞。電子情報通信学会会員。



飯田 一朗

昭和 53 年東京大学工学部電子工学科卒業。昭和 58 年同大学院博士課程修了。工学博士。同年(株)富士通研究所入社。以来ローカルエリアネットワーク、ネットワークアーキテクチャ、エージェント技術などの研究開発に従事。現在同社ネットメディア研究センター主任研究員。昭和 58 年電子情報通信学会論文賞受賞。電子情報通信学会会員。