3U-6 バッテリ残量を気にするモバイルエージェントの退避行動

金子 平祐 *, 深澤 良彰 *, 粂野 文洋 † 本位田 真一 ‡
* 早稲田大学 理工学部, † 株式会社三菱総合研究所, ‡ 国立情報学研究所

1 はじめに

ノートPCやPDAなどの携帯端末の普及、高性能化、ネットワーク対応に加え、これらをターゲットとしたJava技術等が追い風となり、様々なアプリケーションサービスが、いつでも、どこにいても受けられるようになる。すなわち、高速なネットワークを介してダウンロードしてきたアプリケーションソフトウェアを、自分の所持している携帯端末上で(OSの違いを意識することなく)自由に実行できるようになるのである。

ところで、バッテリにより駆動される小型携帯端末にはバッテリ切れという特有の不安要素が存在する。この問題に対して、省電力デバイスの採用や、命令タイミングのスケジューリングなどにより、バッテリの持ち時間を長くして対処しようという研究は従来より積極的に行なわれてきた。しかし、いくらバッテリの稼働時間が長くなろうが、アプリケーション作業中にバッテリ切れが起こるという可能性を完全に取り除くことは不可能である。そこで、本研究では、**モバイルエージェントの自律性**と**モビリティ**を活かし、バッテリ切れに対してセーフティな対応をとれるような、頑健な小型携帯端末向けシステムを提案する。

2 ソリューション

本研究では、端末がバッテリ切れに陥りそうになると、モバイルエージェントとして作成されたアプリケーションが、自律的に安全な別ホストへ退避できるようにし、エージェント自身、およびその作業情報の消失を防ぐようにする(**自律的サバイバル**)。さらに、退避完了後は作業中断時までの情報の回復を図り、避難先において処理の再開などの適切な行動をとれるようにする。

このように、本研究ではバッテリ不足時におけるエージェントの自律的退避行動を、

- シェルターホストへのエージェントの**待避**
- 待避後の**リカバリー**

の2つの視点から捉え、セーフティなシステムを実現する。

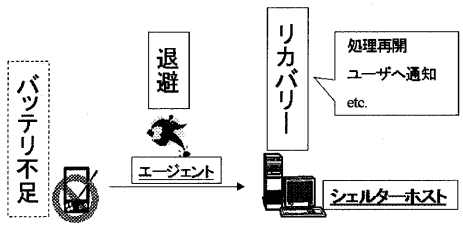

図 1: 自律的退避行動

2.1 退避

2.1.1 シェルターホストへの避難

携帯端末上のモバイルエージェントはバッテリの不足を感知すると、安全なホストへの退避を行ない、それまでの計算結果と共に、生き残りを図ろうとする。この待避先ホストを**シェルターホスト**と定義する。ただし、ネットワークの状況などにより、避難に失敗するケースも十分ありうる。移動失敗時を考慮して、シェルターホストへの退避の前に、エージェントをモバイル端末上にファイルの形で保存し、エージェントが永遠に失われる、という最悪の状況を避け、より頑健なシステムを得るようにする。

シェルターホストへ退避したエージェントは、そのホスト上で作業を続行させたり、退避時専用のルーチンを開始させたりすることができるが、ファイルに保存した場合は、最低限それまでの作業内容が保存されるだけであり、リアルタイムシステムや、マルチエージェント環境下等ではシステム全体に致命的な影響を与える可能性も十分ありうる。したがって、可能な限りシェルターホストへの避難を試みるべきである。

2.1.2 圧縮待避

一般に、モバイル端末が利用できるネットワークの転送速度は、固定ネットワークのそれに比べて低速になってしまう。したがって、避難エージェントのサイズが大きいときには、避難の最中に電源遮断、ネットワーク遮断などのアクシデントが起こってしまう可能性もある。このネットワーク転送中のアクシデントに対して、避難前にエージェントを圧縮することにより、その可能性をできる限り少なくする。本研究では、移動方式に圧縮転

Shelter behavior of Mobile agent that is worried about amount of the battery remainder

Heisuke Kaneko*, Yoshiaki Fukazawa*, Fumihiro Kumeno†, Shinichi Honiden‡
*Department of Computer & Information Science, School of Science & Engineering, Waseda University
†Mitsubishi Research Institute, Inc.
‡National Institute of Informatics

送を採用しているAgentSpace[1]を基盤エージェントシステムとして利用することにより、**圧縮待避**を実現する。

2.1.3 パニック現象と危機管理センター

携帯端末上のアプリケーションはその性質上、すぐに起動できるよう、完全に終了せずに、アイドリング状態を保っているケースの方が多い。したがって、1つの携帯端末上には多数のアプリケーションエージェントが稼働していると想定できる。このような場合、圧縮退避を適用していても、バッテリ不足を感知したエージェント群が一斉に避難を開始することから、ネットワークへの負荷が高くなってしまう(**パニック現象**)。これを防ぐために、携帯端末上の全体的な避難計画を随時発行する**危機管理センター**の概念を導入する。

各エージェントは自分の情報(サイズ、リソース情報など)を危機管理センターに登録してから、各々のタスクを行なう。バッテリが危険水準に達すると、危機管理センターは現在の危険レベルと全体の状況を確認しながら、各エージェントにシェルターホストの指定や退避行動を指示する。また、エージェントのサイズが大きすぎるときは、画像ファイルなどのリソースを破棄させる特殊指示も行なえるようにする。

2.2 リカバリー

避難を終えたエージェントには、作業中断時までの情報を復旧させ、作業再開や作業結果のユーザへの通知などの適切な行動をとらせる必要がある。アプリケーションドメインによって最適な行動は変化してしまうので、避難後の処理は各エージェント作成者が記述するべきである。したがって、避難後の行動に関するコールバックメソッドをシステム側で予め用意するべきである。

3 実装

ノートPC(OS:Windows98)で稼働しているアプリケーションエージェントが、バッテリ切れに対して自律的な退避行動がとれるようなエージェントシステムを構築している。基盤となるエージェントシステムとしてはAgentSpaceを採用し、本研究の目的に合わせて拡張していくことにした。全体的な構成は下図のようになる。

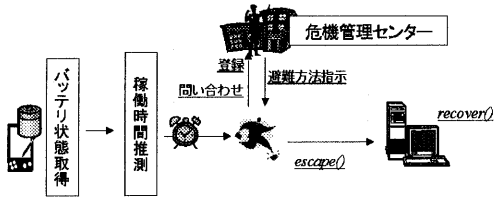

図2: システム構成

バッテリ状態取得　AgentSpaceは、JVM上で稼働するモバイルエージェントシステムである。しかし、Java APIにはバッテリ状態を参照できるようなものは用意されていない。そこで本研究では、Win32APIの電源状態取得関数 GetSystemPowerStatusをJNI(Java Native Interface)を用いて呼び出し、バッテリ状態へのアクセスを可能にする。

稼働時間推測ルーチン　バッテリの減り方が速ければ、残量が十分あったとしても危険度は大きいと言える。したがって、避難タイミングを決定する指標としては、バッテリの残量よりも残り時間を重視すべきである。本研究では、以下のように残り時間を算出する。

$$\text{残り時間} = 1\%\text{平均消費時間}(sec/\%) \times \text{バッテリ残量}(\%)$$

退避行動開始　バッテリ状態取得、および稼働時間推測ルーチンを用いて、残り稼働時間が設定時間に達すると、警告を発するタイマーを作成する。各エージェントはこのタイマーを所持しながら作業をこなし、警告を受け取ると、危機管理センターにシェルターホストの位置や退避順序などを問い合わせ、それぞれの退避行動を開始する。

コールバックメソッド　AgentSpaceの移動用コールバックメソッドdispatch()は、その結果の成否を返さない。そこで、移動の成否を返すような移動用コールバックメソッドescape()を作成し、避難失敗時の行動を可能にする。また、リカバリー実現のため、シェルターホストへの避難が無事に終了した後のエージェントの行動を記述するため、コールバックメソッドrecover()を用意した。

4 本システムの利用法

危機管理センターの機能を付加したエージェントシステムと、バッテリ不足時に退避行動を行なうクラスEscapeAgentが提供される。アプリケーション作成者はクラスEscapeAgentを継承させ、コールバックメソッドに各アプリケーションの目的に沿ったエージェントの動作を記述すれば、自律的避難機能を組み込んだアプリケーションエージェントを作成できる。

5 おわりに

今回の研究では、バッテリ不足に対するエージェントの行動に焦点を当てているが、今後はこれを拡張してネットワーク遮断に対してもセーフティな対応ができるようにしていく。また、これらの手法をフィードバックし、モバイル端末向けエージェントシステムの構築を目指していきたい。

謝辞　本研究をすすめるにあたり、AgentSpaceに関してご教示いただいた、お茶の水女子大学の佐藤一郎先生に深く感謝いたします。

参考文献

[1] 佐藤一郎 “A Mobile AgentSpace システムアーキテクチャ”, http://www.is.ocha.ac.jp/~ichiro, 1998