2006－OS－102（2）
2006／5／12

ケーパビリティに基づくアクセス制御を持つスケジュール管理アプリケーションの構想

馬渕 充啓† 新城 靖‡ 大木 薫† 加藤 和彦‡

†筑波大学システム情報工学研究科 コンピュータサイエンス専攻
‡筑波大学システム情報工学研究科 コンピュータサイエンス専攻 / 科学技術振興機構（JST）

要 旨

ACL(Access Control List) に基づくアクセス制御には、2 つの問題がある。1 つは、一時的にアクセス権を与えたい時にシステム管理者の負担が大きくなるという問題である。2 つ目は、必ずユーザ認証をともなうので、異なるユーザ認証のドメインに属するユーザ間で協調作業を行うことが難しいという問題である。本論文は、スケジュール管理アプリケーションを題材として、アクセス制御の方式としてケーパビリティに基づく方式を導入することで、これらの問題点を解決する方法について述べている。その特徴は、権限が低いケーパビリティを実現するために、スタッカブル・オブジェクトを用いている点にある。スケジュール管理アプリケーションは、XML Web サービスのサーバとクライアントにより実装される。

Towards a Schedule Management Application with Capability-based Access Control

Mitsuhiro Mabuchi† Yasushi Shinjo‡ Oki Kaoru† Kazuhiko Kato‡

†Department of Computer Science, University of Tsukuba
‡Department of Computer Science, University of Tsukuba / Japan Sience and Tecnology Agency（JST）

Abstract

A access control based on Access Control List (ACL) has two problems. First, the effort of a system administrator becomes large when we allow a user to temporary access an object. Second, since this access control requires user authentication, it is hard for users in different user authentication domains to work together. To solve these problems, this paper introduces capabilities-based access control to a schedule management application as an example. Stackable objects are used to realize lesser capabilities. The schedule management application consists of servers and clients of XML Web Services.

1 はじめに

現在、アクセス制御方式としては、ACL(Access Control List) に基づくものが主流である。ACL とは、アクセスの主体（ユーザ、プロセス等）があるオブジェクト（ファイル等）に対してどのような操作を行うことができるか（またはできないか）を列挙したリストのことである。ACL に基づくアクセス制御では、ACL をオブジェクト側に設置する。ある主体がオブジェクトにアクセスする時、まず、主体の認証が行われ、そして、その主体が ACL を満たすかが検査される。

既存のアプリケーションに用いられている ACL に基づくアクセス制御には、一時的にアクセス権を与えたい時にシステム管理者の負担が大きくなるという問題がある。また、必ずユーザ認証をともなうので、異なるユーザ認証のドメインで連携した作業を行うことが難しいという問題点もある。

アクセス制御には、ACL に基づくものの他に、ケーパビリティに基づくものがある。これは、オブジェクトへの参照と、オブジェクトに対するアクセス権をアクセスの主体に持たせるものである。ケーパビリティに基づくアクセス制御では、アクセス権の委譲にシステム管理者の手を煩わせる必要がないという利点がある [10]。また、ユーザ認証なしにアクセス制御が行えるので、異なるユーザ認証のドメインで連携した作業を行うことが容易である。このように、ケーパビリティに基づくアクセス制御には、様々な利点があるが、インターネットのような開かれた環境においてはその実現方式はまだ確立されていない。

本研究では、スケジュール管理アプリケーションを題材として、インターネットのような開かれた環境において、ケーパビリティに基づくアクセス制御の実現方式を明らかにしていく [6]。スケジュール管理アプリケーションを選んだ理由は、第１に、センシティブな情報を扱うために、アクセス制御が重要であるからである。第２の理由は、インターネットのような開かれたな環境において、個々のユーザが自律性を保ちつつ、かつ、ユーザ同士が連携してプロジェクトを進めていくためには、スケジュール管理が重要であるからである。また、従来の研究では、ファイルの読み書きやメッセージ・ポートの送信などの単純な操作を扱っていた。これに対して、スケジュール管理では、より複雑な操作を必要としている。そのような複雑な操作を伴うアクセス制御においては、操作の限定されたケーパビリティ（弱いケーパビリティ）の実現方式は明らかにはなっていない。スケジュール管理における弱いケーパビリティといい、特定のスケジュールに対してイベントの存在だけを示し内容を隠したり、特定の範囲に限って書き込み権限を与えることが考えられる。本研究では、スタッカブル・オブジェクトを利用することで弱いケーパビリティを実現する [8]。

本論文は、以下のように構成されている。2 章では、本スケジュール管理アプリケーションの目標と ACL に基づくアクセス制御の問題点について述べる。3 章では、アプリケーションの概要について述べている。4 章では、XML Web サービスとしての実現ついて述べている。5 章では、関連研究について述べると共に、関連研究と本研究との違いについても述べる。そして、6 章でまとめと今後の課題について述べる。

2 本スケジュール管理アプリケーションの目標と ACL に基づくアクセス制御の問題点

スケジュール管理機能を持つアプリケーションには大きく分けて、2 種類ある。

1. 個人情報管理 (Personal Information Management, PIM) アプリケーション：主に個人の情報を管理する。携帯端末と情報を共有する機能や外部のスケジュールを取り入れたり、スケジュールの一部を外部に取り出す機能を持つ。
2. グループウェアが持つスケジュール管理機能：登録されたメンバのスケジュールを一括して管理する機能を持つ。

本アプリケーションでは、後者と同じく、複数のユーザのグループを対象としたスケジュール管理を行えるようにする。このグループは、次のような特性を持つ。

- ある目的をもったユーザが緩いグループを構成し、それらの単位でスケジュールを共有する。スケジュールを共有しているユーザがグループを構成する。
- あるユーザは、複数のグループのメンバになる

ことができる。

- グループは、オーバーラップしていることもある。
- グループのメンバをあらかじめ規定できないことがある。

このようなグループを対象として、本研究では次のような特性を持つスケジュール管理アプリケーションを実現する。

- 一般的なグループウェアとは異なり、オープンな環境でスケジュールの共有ができる。一般的なグループウェアでは、特定のユーザ認証のドメインの内部（レルム）に閉じた範囲でのみスケジュールの共有ができる。
- あるスケジュールに対して、当初想定していないアクセス権を設定することができる。
- 普段はアクセスできないユーザに対して、一時的なアクセスを可能にする。

このようなスケジュール管理アプリケーションを実現しようとした場合、既存の ACL に基づくアクセス制御を用いようとすると、次のような問題点がある。

(1) ユーザ管理の手間が大きくなる。アクセス制御を実現するためには、主体となるユーザを必ず事前に明確にする必要がある。すなわち、アクセス制御を行うためには、まずシステム管理者によるユーザ管理が必要となる。プロジェクトの進行に従って、プロジェクトのメンバが増えたり減ったりした場合、システム管理者の手間が大きくなる。場合によっては、一時的にあるユーザにアクセスを許可したいことがある。このような場合にも、そのユーザを一時的に登録し、ACL を設定し、アクセスさせたい期間が過ぎたらユーザの削除と ACL の変更を行う必要がある。このような作業は、間違いが許されず、システム管理者にとって重たい仕事になっている。

(2) 運用上の問題として、ユーザ登録が許されない場合、過大な権限を与えてしまう。たとえば、ある組織のグループウェアには正式な構成員だけしか登録ができず、その私設秘書は登録できないことがある。その場合、構成員は秘書に自分のユーザ名とパスワードを教えてしまうという運用がしばしばなされる。こうなると秘書は、構成員の全権限を得ることになる。例えば、秘書にはスケジュールの読み書きだけを許したいとしても、別の操作、たとえば電子メールの読み書きまで許してしまうことになる。

(3) ユーザ認証のドメインを超えて連携作業を行うことが難しい。例えば、組織Aと組織Bが、それぞれ別のユーザ認証の仕組みを持っていたとする。組織Aと組織Bが連携し、互いの資源の一部を相手側にアクセスを許したいとする。この場合、次のような問題点がある。

- 相手組織のユーザ認証の仕組みが信用できないと使えない。
- ACL に、相手組織のユーザを登録していく方法をとると、ユーザの数が多いと手間がかかる。
- あるユーザが組織Aと組織Bの両方に登録されていたとする。どの組織のユーザ認証を受けるかによって、権限が違ってしまう。

(4) システム設計時には想定していなかったようなアクセス制御の方法を導入することが難しい。例えば、ユーザAが持つレンダーに対して、「2006 年 5 月 20 日の 10:00 - 18:00 の範囲に限り書き込みできる権限」を、別のユーザBに与えたいとする。アプリケーションの設計時に、このような機能をあらかじめ想定して設計してあれば問題はないが、想定していなかった場合には後で付加することは難しい。アクセス制御について、スクリプト言語等により拡張を許す方法も考えられるが、この場合、ポリシーの記述が複雑になってしまうという問題が生じる。

3 スケジュール管理アプリケーションの概要

2 章では、本スケジュール管理アプリケーションの目的と、ACL に基づくアクセス制御の問題点について述べた。この章では、それらの問題点を解決するようなスケジュール管理アプリケーションの機能について述べる。その特徴としては、ケーパビリティに基づくアクセス制御を備えている点にある。

3.1 機能

本アプリケーションでは、共通の目的を持つユーザ同士が連携してスケジュール管理を行うために以下のような機能を提供する。

1. 特定の権限だけを他の利用者に渡せるようにする。例えば、あるユーザが、他のユーザに対してスケジュールを閲覧のみ可能にし、閲覧させ

ること以外の権限を与えないようにできる。
2. スケジュールの一部の項目だけ閲覧できるようにする。例えば、予定が入っていることは閲覧できるが、予定の内容までは閲覧できないようにする。
3. あるユーザが、他のユーザに対してスケジュールの指定された時間帯にだけ書き込めるようにする。

その他の機能として、複数のカレンダーをマージして 1 つのカレンダーとして扱う機能を提供する。マージ機能により、複数のカレンダーを持つユーザが自分のスケジュールを把握しやすくなる。

3.2 ケーパビリティに基づくアクセス制御の導入による ACL の問題の解決

3.1 節で述べた機能を、ケーパビリティに基づくアクセス制御を導入することで提供する。たとえば、任意の操作が可能なケーパビリティを元にして、閲覧だけが許されたケーパビリティを生成して利用可能にする。このように、操作が制限されたケーパビリティを弱いケーパビリティと呼ぶことにする。

ケーパビリティに基づくアクセス制御においては、ユーザ認証が不要になるので、2 章で述べた問題 (1)、および、問題 (3) が解決される。また、弱いケーパビリティを利用可能にすることにより、問題 (2) が解決される。

弱いケーパビリティの実現、および、問題 (4) を解決するために、本研究ではスタッカブル・オブジェクトを用いる [8]。スタッカブル・オブジェクトとは、下位層に他の類似のオブジェクトを持ち、自分自身が受け取ったメッセージを、下位層のオブジェクトに中継しながら処理を行うようなオブジェクトである。

スタッカブル・オブジェクトの例を図 1 に示す。Object B1 が下位のオブジェクトで、その上にスタックされている Object A1 がスタッカブル・オブジェクトである。スタッカブル・オブジェクトへのアクセスは、スタッカブル・オブジェクトの持つメソッドを呼ぶため、それ以上のことはできない。図 1 では、Object A1 は read メソッドしか持っていないため、write や delete は実行することはできない。このようにして、弱いケーパビリティを実現することができる。

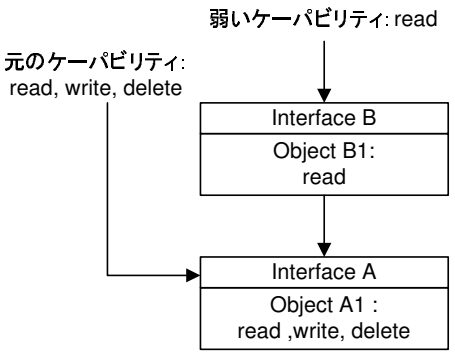

図 1: スタッカブル・オブジェクトによる弱いケーパビリティの実現

スタッカブル・オブジェクトは、元々のシステムに対して新たな機能を追加するためにも利用できる。あらかじめ想定できないような機能を提供するために、新たなタイプのスタッカブル・オブジェクトを導入可能にする。これにより、問題 (4) を解決することができる。

ケーパビリティには、あるユーザに与えてしまったケーパビリティを取り戻すことができない問題がある。たとえば、図 2 は、User A, B, C が 1 つのプロジェクトを実行していて 1 つのスケジュールを共有していることを示している。User B が、プロジェクトを離れた時、一度 User B に与えてしまったケーパビリティを取り戻すことができない。

その問題に対しては、ケーパビリティに有効期限を付ける方法とケーパビリティを無効化する方法で対応する。有効期限とケーパビリティの無効化の実現方法については、4.5 節で述べる。

4 XML Web サービスとしての実現

本スケジュール管理アプリケーションを、XML Web サービスのサーバ、および、クライアントとして実現する。XML Web サービスを利用した理由は、第 1 に、オンラインであれば、世界中どこにいてもアプリケーションが使用できることである。第 2 に、異なるプラットフォームでも動作可能であることである。

Web サービスのサーバとクライアントは、Java と Glue を用いて開発している [4]。Glue とは、Java

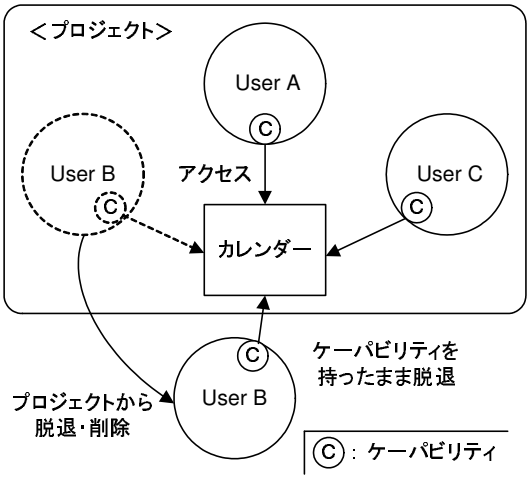

図 2: ケーパビリティの無効化の問題

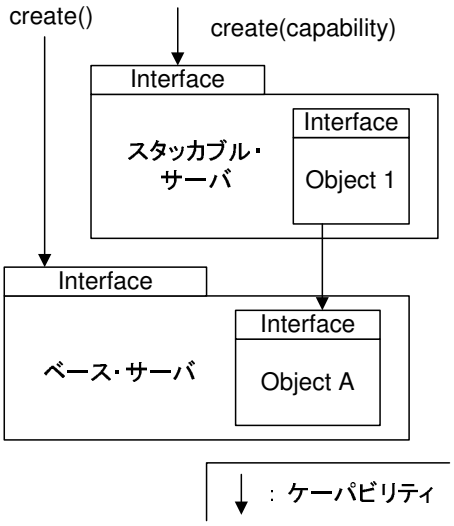

図 3: ベース・サーバとスタッカブル・サーバ

オブジェクトによる Web サービスの実装を可能にするフレームワークである。

4.1 ケーパビリティの表現

ケーパビリティには、WSDL（Web Service Description Language）の文書を用いる。WSDL の文書は、Web サービスにおけるインタフェース記述を含む XML 形式の文書である。WSDL による記述は、サーバやオブジェクトの位置情報（URL）を含むため、それだけでオブジェクトの手続きを呼び出し、操作することが可能である。ケーパビリティには、パスワード等のユーザ識別機能がないため盗聴や偽造等から保護しなければいけない。本システムでは、URL に乱数を使用し予測されないようにすることで、ケーパビリティの保護を行っている [9]。

4.2 本アプリケーションのオブジェクト

本アプリケーションは、カレンダー・オブジェクトから構成さる。カレンダー・オブジェクトは、イベントのリストを持つ。各イベントはイベント ID、開始時間、終了時間、イベント名を持つ。カレンダー・オブジェクトは、以下の 4 つのメソッドを持つ。

read　時間帯を指定すると、その時間帯に含まれるイベントを返す。

write　時間帯とイベント名をカレンダーに書き込む。

delete　イベント ID を指定すると、その ID を持つイベントを削除する。

kill　指定したカレンダーを削除する。

4.3 サーバ

本アプリケーションのサーバは、カレンダー・オブジェクトを管理するオブジェクトであり、オブジェクトを生成するためのメソッドして create を持つ。サーバには、ベース・サーバとスタッカブル・サーバが存在する (図 3)。

ベース・サーバ　ベース・サーバの create メソッドは、長さ 0 の引数を取る。create メソッドが呼ばれると、一般的なカレンダー・オブジェクト作成し、それに対するケーパビリティを返す。

スタッカブル・サーバ　スタッカブル・サーバの create メソッドは、引数として下位オブジェクトへのケーパビリティを必要とする。結果として、スタッカブルなカレンダー・オブジェクトを作成し、それに対するケーパビリティを返す。

4.4 スタッカブル・オブジェクトによる弱いケーパビリティの実現

現在、次の 4 種類のスタッカブル・オブジェクトとそのサーバを実現した。

- メソッドの数を減らしたスタッカブル・オブジェクト。例えば、read メソッドだけを持ち、write, delete, kill メソッドを受け付けない。
- 開始時間、終了時間のみ閲覧可能で、イベント ID とイベント名を隠して返す。
- create 時に引数として、書き込み可能な時間帯を指定することができる。指定された時間帯だけイベントを書き込むことができる。
- create 時に引数として、複数のケーパビリティを取る。そのなかで、read メソッドを含むケーパビリティを使用して、複数のカレンダーをマージして閲覧することができる。

4.5 スタッカブル・オブジェクトによるケーパビリティの有効期限設定と無効化の実現

無期限と有効期限付のケーパビリティには、以下の 3 種類が考えられる。

1. 有効期限のないケーパビリティ。
2. 有効期限付で延長可能なケーパビリティ。
3. 有効期限付で延長不可能なケーパビリティ。

有効期限付のケーパビリティを受け取ったユーザは、その期限が切れるまで使用可能になる。延長可能な場合は、延長可能な期限の間だけ延長を繰り返すことができる。延長可能な期限を付加することで、永続的に延長することを防ぐ。延長不可能な場合は、期限が切れると使用することはできなくなる。これにより、ケーパビリティの再利用の可能性を最小限にしている。

上記 2 のケーパビリティの実装方法について説明する。ケーパビリティに有効期限を付加するには、図 4 のように、タイマーを使用し一定時間経過後に消去されるオブジェクトをスタックすることで実現する。延長する場合は、そのカレンダー・オブジェクトのもう一つのインタフェースにある renew() メソッドを呼び出しタイマーをリセットする。

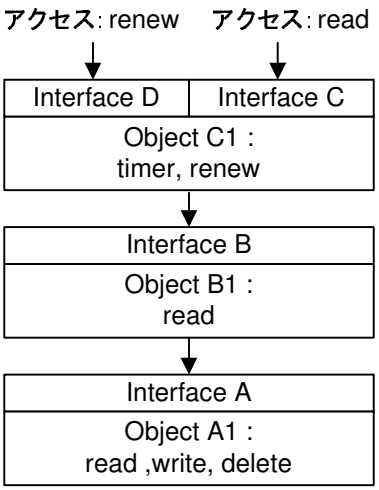

図 4: ケーパビリティの延長

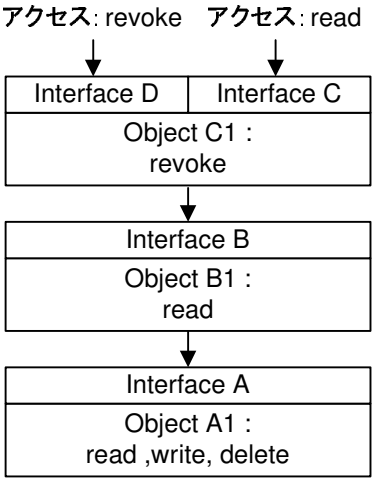

図 5: ケーパビリティの無効化

次に上記 3 のケーパビリティについては、2 のオブジェクトで renew メソッドを持たないオブジェクトをスタックすることで実現する。

ケーパビリティを無効化可能にするには、図 5 のように、revoke メソッドという特殊なメソッドを持ったオブジェクトをスタックすることで実現する。無効化する場合は、そのカレンダー・オブジェクトのもう一つのインタフェースにある revoke メソッドを呼び出しオブジェクトを消去する。

4.6 クライアント

現在、GUI を備えたクライアントの実装を行っている。ここでは、クライアントの持つ機能について述べる。

- 選択中のカレンダーを閲覧、追加、削除する機能：選択中のカレンダー・オブジェクトの read、write、delete メソッドを呼び出しカレンダーにイベントを閲覧、追加、削除する。
- 新しいカレンダーを作成する機能：ベース・サーバの create メソッドを呼び出すことで、カレンダーを作成しそのカレンダーへのケーパビリティを受け取る。
- 弱いケーパビリティを作成する機能：保持しているカレンダーを選択し、スタッカブル・サーバの create メソッドを呼び出し弱いケーパビリティを作成する。
- 保持しているケーパビリティを他のユーザに渡す機能：渡すケーパビリティと相手を決定し送信する。
- 他のユーザからケーパビリティを受け取る機能：送られてきたケーパビリティを受け取る。
- カレンダーを削除する機能：選択中のカレンダー・オブジェクトの kill メソッドを呼び出し、カレンダー・オブジェクトを削除する。
- カレンダーの一覧を表示する機能：ケーパビリティと弱いケーパビリティの関係を分かりやすく表示する。

4.7 ケーパビリティの受け渡し

ケーパビリティを送信する場合、相手が確実に本人であることを保証する必要がある。また、大量にケーパビリティを送信する場合の効率的な送信方法も必要である。現在のところ、人間が読み書きできるメッセージの送信機能を用いることを検討している [10]。具体的には、既存のインスタント・メッセンジャーや電子メールを使用し、暗号化機能やデジタル署名を用いることで安全性を高める。

5 関連研究

ケーパビリティに基づくアクセス制御は、初期の LAN (Local Area Network) を対象とした分散オペレーティング・システムの研究でしばしば使われた。たとえば、Mach システムでは、ケーパビリティを通信ポートへのアクセス制御を行うために利用している [1]。Mach システムでは、ケーパビリティは、カーネルにより保護されている。Amoeba システムでは、ケーパビリティは、一方向関数を使って改ざんできないようになっており、一般のプロセス間通信で受け渡しができるようになっている [7]。いずれのシステムも、ケーパビリティは、あるオペレーティング・システムの内部でのみ有効である。これに対して本研究では、インターネットのように開いた環境でケーパビリティに基づくアクセス制御を可能にする。

サイボウズ Office6 は、サイボウズ株式会社により開発されたグループウェアである [3]。このグループウェアは、利用者認証と ACL に基づくアクセス制御の仕組みを用いている。このグループウェアには、Palm Desktop や Outlook 等の個人情報管理アプリケーションとグループのスケジュールを同期させる機能がある。しかしながら、それぞれ独自のユーザ認証を持つ複数のグループウェアのインスタンスを連携させる機能はない。

Ariel AirOne Project A は、アリエル・ネットワーク株式会社により開発された P2P を用いたグループウェアである [2]。Ariel AirOne Project A では、ユーザ登録時に中央の認証局において公開鍵に基づく証明書を発行する。Ariel AirOne Project A では、ルームと呼ばれるスケジュール管理機能を持つグループを構成することができる。ルームには、共通鍵暗号に基づく鍵があり、それを他のメンバに暗号通信で送信することで、他のメンバへのルームへのアクセスを許す。ルームは、一般的な意味での複数の組織に属するメンバも登録できる。このことは、利用者認証の仕組みとして、中央の認証局を共有しているから可能になっているのであり、複数のレルムの間で連携が行えるのではない。また、ルームの鍵は、Mach システムのカーネルと同様に、特定のアプリケーションで保護しており、外部には取り出せないような仕組みにはなっている。しかしながら、その保護の仕組みが破られた場合には、ルームの管理者が意図していない人によるアクセスを許すことになる。これに対して本研究で用いているケーパビリティの仕組みは、他人にも権限に一部を積極的に渡せるようにしようとするものであり、考え方が大きく異なる。

Kerberos は、シングルサインオンを実現するためのネットワーク認証システムである [11]。Kereros では、ユーザ認証が行われると、TGT (Ticket Granting Ticket) と呼ばれる値が得られる。一般のサービス、たとえば、遠隔ログインやメールボックスへのアクセスを受ける時には、TGT から生成したチケットを示すことで個別の重複したユーザ認証を避けることができる。このような Kerberos の仕組みは、ケーパビリティの仕組みに類似しているが、Kerberos の場合、ユーザ認証機能を省略するための機能しかない。ユーザ認証後のアクセスには、個々のサービスごとに ACL に基づくものが用いられる。

Liberty Alliance は、World Wide Web においてシングルサインオンを実現するための仕組みである [5]。Liberty Alliance では、複数の連携した Web サイト間でユーザ認証の情報を交換することができるが、アクセス権を交換するためのものではない。これに対して、本スケジュール管理アプリケーションでは、ケーパビリティというアクセス権を交換する。

6 おわりに

本論文では、ケーパビリティに基づくアクセス制御を持つスケジュール管理アプリケーションの構想について述べた。本アプリケーションの特徴は、アクセス制御に ACL ではなくケーパビリティを採用していることである。ケーパビリティを用いることで、ACL に基づくアクセス制御におけるシステム管理の問題や協調作業の問題を解決することができる。弱いケーパビリティは、スタッカブル・オブジェクトを用いて実装している。また、盗難や盗聴等にはケーパビリティに有効期限を付けることで対応する。本アプリケーションは、XML Web サービスのサーバ、および、クライアントとして実装する。現在までに、ベース・サーバ、および、いくつかのスタッカブル・サーバの実装を行った。

今後の課題は、GUI を持つクライアントを作成ことである。また、トランザクションの導入も考えている。

参考文献

[1] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian and M. Young : “Mach : A New Kernel Foundation for UNIX Development”, Proceeding of USENIX Summer Conference, pp. 93-112 (1986).

[2] Ariel AirOne Project A: Ariel Network(2006). http://www.ariel-network.com/

[3] サイボウズ Office6: Cybozu(2006). http://www.cybozu.co.jp/

[4] G.Glass: Web Services: Building Blocks for Distributed Systems (2001).

[5] Liberty Allience Project: Introduction to the Liberty Alliance Identity Architecture Revision 1.0 (March, 2003)

[6] 馬渕充啓, 新城 靖, 加藤和彦: ケーパビリティに基づくアクセス制御を持つスケジュール管理アプリケーション日本ソフトウェア科学会 ソフトウェアシステム研究会 SPA ポスター発表 (2006)

[7] S.J.Mullender, G. van Rossum, A.S.Tenenbaum, R. van Renesse, and H. van Staveren: “Amoeba: A Distributed Operating System for the 1990s”, IEEE Computer, Vol. 23, No.5, pp.44-53 (1990).

[8] Y.Shinjo, and Y.Kiyoki: “The Object-Stacking Model for Structuring Object-Based Systems”, IEEE International. Workshop on Object Oriented and Operating Systems, pp.328-340 (1992).

[9] 新城 靖, 阿部 聡, 板野 肯三: XML Web サービスのための大域的ファイル・サービスの提案, 情報処理学会システムソフトウェアとオペレーティングシステム研究会, 2004-OS-96, pp.15-22(2004).

[10] S.Suzuki, Y.Shinjo, T.Hirotsu, K.Kato and Kozo Itano: “Capability-based Egress Network Access Control for Transferring Access Rights”, The International Conference on Information Technology and Applications (ICITA-2005), pp.488-495 (2005).

[11] J. G. Steiner, B. C. Neuman, and J. I. Schiller: “Kerberos: An Authentication Service for Open Network Systems”, In Proceedings of the Winter 1988 Usenix Conference, pp.191-201(1988).