2007－OS－105 (13)
2007／4／5

Webページに対するケーパビリティを用いたアクセス制御のプロキシによる実現

松井 慧悟† 佐藤 聡† 新城 靖†,‡ 板野 肯三† 馬渕 充啓†
加藤 和彦†,‡

†筑波大学院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 CREST

要 旨

Webページに対してアクセスできる権利を一時的に他の人に委譲したいという要求がある。現在、多くのWebページに対するアクセス制御はアクセスコントロールリスト(ACL)に基づいたものが使われている。ACLに基づくアクセス制御において、一時的な権限委譲を行おうとした場合、Webページの管理者への負担が重くなってしまう。本研究では、ケーパビリティという概念を用いることにより、この問題を解決する。ケーパビリティの性質として他人に渡すことが可能な点とあるケーパビリティから弱い権利を持ったケーパビリティ作成可能な点がある。このような性質を利用することでユーザは、自分の持っているWebページへのアクセス権を他のユーザに渡すことが可能になり、管理者の負担を軽減することができる。

Implementation of capability-based access control for Web pages with a proxy

Keigo Matsui† Akira Sato† Yasushi Shinjo†,‡ Kozo Itano†
Mitsuhiro Mabuchi† Kazuhiko Kato†,‡

†Department of Computer Science, University of Tsukuba
‡Japan Science and Technology Agency, CREST

Abstract

It is demanded to delegate access rights to other users for a temporary time. Currently, most web pages are protected by the access control mechanism based on Access Control Lists (ACLs). In this mechanism, web contents managers have to change configurations when users delegate their access rights to other users. We solve this problem by applying the concept of capabilities. In capability-based access control, users are allowed to pass their capabilities to other users without contents managers' efforts. Furthermore, users can make restricted capabilities from base capabilities. These characteristics make it possible for users to transfer their access rights to other users for a temporal time. Therefore, this reduces the contents managers' efforts.

1 序論

協調作業において、自分の権限を他人に委譲することにより作業の効率化を図ることができる。例えば、あるグループのリーダは、ある書類を閲覧する権限を持つが、他のメンバは、その権限を持たないとする。このとき、リーダがメンバに対して書類を閲覧する権限をメンバに委譲することにより、メンバが直接書類を閲覧できるため、効率化を図ることができる。近年、Web インタフェースを有するアプリケーションが増えたため、Web ページの閲覧に関しても権限を一時的に譲渡したいという要求が高まってきている。このとき、リーダが所有している Web ページを常時閲覧できる権限を全て委譲すると問題が発生するため、制限されたアクセス権を用意すると便利である。例えば、ある Web ページを 1 時間に限り閲覧させる権限や、10 回だけアクセスできる権限を用意すると便利である。

現在、多くの Web ページに対するアクセス制御は、アクセスコントロールリスト（Access Control List：ACL）に基づいたものが使われている。Web ページの管理者は Web ページに対するアクセス制御を行うために、このリストに対する変更、および、ユーザの管理を行う。ACL を用いたアクセス制御において、一時的な権限委譲を行う場合、閲覧を許可する新たなユーザの追加、および、不要となったユーザの削除、新たにユーザが追加されたことによる ACL への変更など Web ページの管理者が行う作業が多数あり、負担が重くなっている。

本研究では、ケーパビリティという概念を Web ページのアクセス制御に適応することにより、一時的な権限委譲を容易に行えるシステムの提案を目的とする。ケーパビリティとはオブジェクトへの操作を行う権利を含んだ参照のことである。ケーパビリティの性質として他人に渡すことが可能な点と、あるケーパビリティから弱い権利を持ったケーパビリティを持ったケーパビリティを作成可能な点があげられる。この性質を利用して、自分の持っているケーパビリティから使用回数や有効期限を制限した弱いケーパビリティを作成し、他人に渡す。これにより新たなユーザの追加や、ACL を書き換えを行わずに、他人に Web ページの閲覧を一時的に許可することが可能になり、管理者の負担を軽減することができる。

対象＼主体	ユーザ A	ユーザ B	ユーザ C
ファイル A	読み出し 書き込み	読み出し	

図 1: アクセス制御リスト

2 既存のアクセス制御方式

2.1 アクセス制御リスト

アクセス制御リスト (Access Control List:ACL) を用いるアクセス制御方式は、現在、Web ページやオペレーティングシステムにおいて広く使われているアクセス制御の方法である。図 1 にアクセス制御リストの例を示す。アクセス制御リストは対象ごとにあり、その対象に対してどの主体がどの操作を行えるかを記述したものである。

ある主体がある対象に操作を行おうとした場合、その対象のアクセス制御リストの中にその主体が、行おうとしている操作が記述されているか調べる。その操作が記述されていれば、アクセスが許可され操作が行われる。ある主体のある対象へのアクセス権を変更したい場合には、その対象のアクセス制御リストを変更する。

2.2 ケーパビリティを用いたアクセス制御

ケーパビリティとは対象への操作を行う権利を含んだ参照のことである。図 2 にケーパビリティの例を示す。ある主体がある対象に対するケーパビリティを所持している場合、その主体はその対象に対してケーパビリティに含まれている操作を行う権利を有することになる。よって、ある主体がある対象に対して操作を行おうとした場合、その主体がその操作を行う権利を含んだケーパビリティを所持していれば、その対象に対するアクセスが許可され操作が行われる。ある主体のある対象へのアクセス権を変更したい場合には、その対象に対するケーパビリティを新たに配布し、古いケーパビリティを無効化する。

ケーパビリティの特徴は、他人に渡すことが可能な点とあるケーパビリティから弱い権利を持ったケーパビリティを作成可能な点にある。ケーパビリティは参照であるため、他人に渡すことが可能である。その参照を渡された時点でその対象に対するア

主体＼対象	ファイル A	ファイル B	ファイル C
ユーザ A	読み出し 書き込み		読み出し

図 2: ケーパビリティリスト

クセス権を有することになる。

2.3 ケーパビリティを用いているシステム

Amoeba は 1980 年代から 1990 年代にかけて開発された分散型オペレーティングシステムである [6]。Amoeba は、オブジェクトベースのシステムである。Amoeba では、ケーパビリティは各オブジェクトを特定し、保護するために利用されている。Amoeba でのケーパビリティは 128 ビットのバイナリである。

我々は、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を開発している [9]。このアプリケーションでは、インターネットのような開いた環境において、スケジュールの公開、作成、および、変更を自由に行い、スケジュール調整を容易に行うことを目的としている。このアプリケーションでは、カレンダーをオブジェクトとして扱い、ケーパビリティはカレンダーオブジェクトを取得するための WSDL(Web Service Description Language) として表している。WSDL に乱数を含め、予測不可能にすることで、ケーパビリティとして利用している。

3 Web ページに対するケーパビリティを用いたアクセス制御

3.1 要求要件

本研究では、既存の ACL を用いたアクセス制御が行われている Web ページに対して、その仕組みを ACL に変更することなく、ケーパビリティを用いたアクセス制御を導入し、自分の所有する権利を一時的に委譲することを目的とする。この目的を達成するために実現すべき要求用件を以下に示す。

- ケーパビリティは容易に他の人に渡すことができる。

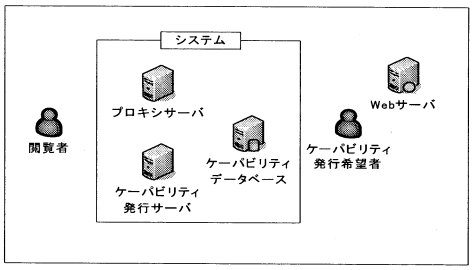

図 3: システム全体図

- 一時的に権利を委譲するケーパビリティに対して、使用回数や有効期限に制限を設ける。
- ケーパビリティの発行者が自分が発行してもらったケーパビリティを無効化することができる。

3.2 プロキシを用いた実現

節 3.1 で述べた要求要件を実現する方法としては、次の 3 つの方法が考えられる。

1. ケーパビリティを扱えるように HTTP を拡張する。また、拡張した HTTP を解釈するように Web ブラウザ、および、クライアントを実装する。
2. Web サーバ自体を変更し、現在の HTTP をそのまま用いてケーパビリティの概念を実現する。
3. プロキシを用いてケーパビリティの概念を実現する。現在の HTTP、Web サーバをそのまま用いる。

3 つ目のプロキシを用いる方法は、既存の技術に変更を加えずに実現可能であり、複数の Web サーバで使うことが可能であるため、応用範囲が広い。したがって、本研究では、プロキシを用いた方法でケーパビリティを用いたアクセス制御について設計する。

本研究で実装するシステムの全体図を図 3 に示す。本研究では、次の 2 種類のユーザを扱う。

ケーパビリティの発行希望者 アクセス権を持つ Web ページの URL をもとにして、初期ケーパビリティの作成をシステムに要求する。

閲覧者 ケーパビリティをうけとり、Web ページを閲覧する。

本研究で提案するシステムは以下に示す3つのサブシステムから構成する。

プロ獅子サーバ ユーザとWebサーバの間でユーザのアクセス制御を行う。

ケーパビリティデータベース ケーパビリティを管理する。

ケーパビリティ発行サーバ ケーパビリティを発行する。

アクセス制御の対象となるオリジナルのWebページは、Basic認証等でプロキシサーバのみがアクセス可能になるように設定する。

3.3 ケーパビリティの形式

本研究ではケーパビリティは、Webページへの閲覧の権利を含んだ参照とし、URLで表現する。通常、Webページの閲覧にはURLが用いられており、URL自身を知っていることがそのURLで表される情報資源閲覧の権利を有していると考えられる。したがって、ケーパビリティの表現としてURLを用いることは自然であるといえる。ただし、ケーパビリティの場合それが予測可能なものであってはならない。予測された場合、本来アクセス権を持たないユーザが不正にアクセス権を得たことになる。

本研究では、RFC 1808[7]を参考にし、ケーパビリティによりアクセス制御を行いたいWebページのURLを以下のように捉える。

```
httpurl =
  baseurl [ relurl ]
baseurl =
  "http://" hostport [ "/" *[ hsegment
   "/"] ]
relurl =
  [ hsegmet *["/" hsegment] "?" search ]
hsegment =
  *[ uchar | ";" | ":" | "@" | "&" | "=" ]
search =
  *[ uchar | ";" | ":" | "@" | "&" | "=" ]
```

このとき、baseurl および relurl の部分の双方にある hsegment の繰り返しの回数はそれぞれケーパビリティの発行者が決定できるものとする。baseurl が同じとなる httpurl については、閲覧に関して同一の権限を与えることが多い。例えば、Apache の Web サーバにおける.htaccess ファイルを用いたACLに基づくアクセス制御では、このファイルが置かれているディレクトリとサブディレクトリすべてに対して同一のアクセス制御を行う設計となっている。したがって、本研究では上記の baseurl を対象としてケーパビリティを定めるようにする。

本研究で用いるケーパビリティを表すURLを以下に示す。

```
capability =
  "http://" anotherhostport accesspoint
  "/" random "/" relurl
anotherhostport =
  hostport
accesspoint =
  *[ "/" hsegment ]
```

ここで random は、乱数により生成された予測不可能な文字列、anotherhostport はサーバの情報、accesspoint はそのサーバの中でプロキシサーバを指定する文字列とする。

以下に具体例を示す。発行希望者が指定した baseurl が次の様であるとする。

```
http://dom/dir/
```

これに対して、提案システムによるケーパビリティは次のようになる

```
http://proxy/random/
```

このとき、オリジナルのコンテンツのURLと、ケーパビリティを用いてアクセスするときのURLの対応は以下の様になる。

- オリジナルのURL

```
http://dom/dir/index.html
http://dom/dir/dir2/index.html
```

- ケーパビリティを表すURL

```
http://proxy/random/index.html
http://proxy/random/dir2/index.html
```

本システムでは、ケーパビリティを発行する際に使用回数、有効期限などの制限を加えることを可能にする。また本システムでは、URLに対してURLとして表現されるケーパビリティを発行する点も可能にする。これにより、発行されたケーパビリティに対して制限を加えた弱いケーパビリティを発行することが容易になる。

発行したケーパビリティを渡すことは、その権限を委譲することである。委譲後にその権限を制限するには、あらかじめ制限を加えた弱いケーパビリ

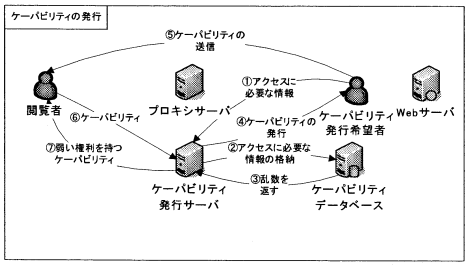

図 4: ケーパビリティの発行

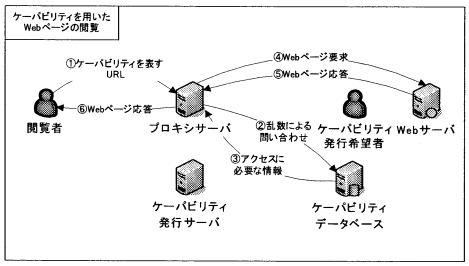

図 5: ケーパビリティを用いた Web ページの中継

ティを生成して渡すか、生成したケーパビリティの制限の変更を行える仕組みを用意する必要がある。無効化または制限の変更を行える仕組みについては今後の課題として取り組んで行く。

3.4 発行サーバ

提案システムによるケーパビリティの発行方法を図4に示す。ケーパビリティの発行希望者は、自分が権限を有するオリジナルの Web ページの baseurl、Basic 認証のユーザ名、および、パスワード等、オリジナルの Web ページの閲覧に必要な情報や使用回数、有効期限等の制限に関する情報を送る。発行サーバは、データベースに受け取った情報を格納する命令を発行し、その情報を問い合わせるためのキーとなる乱数を受け取る。その乱数を基にしてケーパビリティを表す URL を返す。ケーパビリティの発行希望者は、その URL をメールなどで配布して閲覧者に Web ページの閲覧を許可する。

3.5 プロキシサーバ

プロキシサーバを Web ページの閲覧方法を図5に示す。ケーパビリティを用いて Web ページを閲覧する閲覧者は、自分が持つケーパビリティを表す URL に基づいてプロキシサーバにアクセスする。プロキシサーバは HTTP により送られた URL 情報から乱数部 (`random`) と相対 URL 部 (`relurl`) を抽出する。次に、乱数部をキーとしてデータベースにアクセスし、オリジナルの Web ページの `baseurl`、Web ページのアクセス制御に必要な情報、ケーパビリティの使用回数、有効期限等の情報を取得する。使用回数や有効期限等が条件を満たしている場合には、データベースから取得した Web ページの `baseurl` と相対 URL 部よりオリジナルの Web ページの URL を復元し、アクセス制御に必要な情報や閲覧者から送られた HTTP 要求のヘッダを用いて、その URL に HTTP 要求を送信し、応答として得られたステータスコード、ヘッダ、メッセージボディをそのまま閲覧者に転送する。ここで、乱数がデータベースに格納されていない場合は、HTTP 応答のステータスコードとして 404("Not Found") を返す。また、使用回数や有効期限等が条件を満たしていない場合は、HTTP 応答のステータスコードとして、410("Gone") を返す。これらは RFC2068[8] を参考にして定めた。

3.6 データベース

データベースは、ケーパビリティの発行時にユニークな乱数文字列を生成する。そして、その文字列が主キーとなる様に発行サーバから送られた各種情報とともに格納する。このとき生成される乱数はユニークであり、したがってこの乱数を用いて生成されるケーパビリティもユニークとなる。また閲覧時には、ケーパビリティから抽出された乱数部を主キーとして各種情報を取り出す。

4 実装

提案システムの有効性を確かめるために、プロトタイプシステムの実装を行った。データベースの実装には HSQLDB[3] を用い、以下の情報をデータベースに格納している。

- baseurl
- Basic 認証のユーザ名、および、パスワード

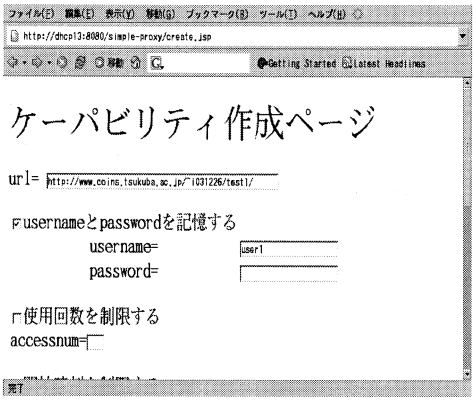

図 6: 情報入力ページ

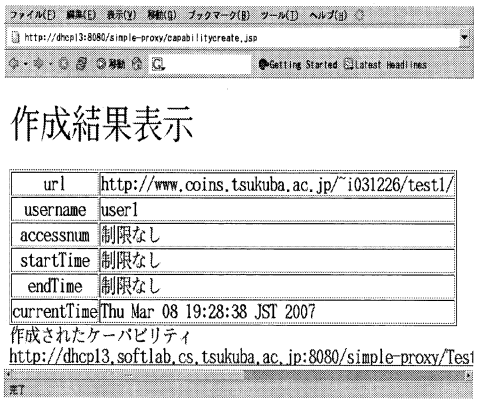

図 7: 作成結果表示ページ

- 使用回数
- 有効期限の開始時刻、および、終了時刻

Java でのデータベースへの情報の格納と問い合わせのために、Java の O/R マッピングのフレームワークである Hibernate[2] を用いた。

プロキシサーバの実装には、Servlet を用いた。サーブレットコンテナとして Tomcat[1] を用いた。Web ページの中継などの主な機能を実装した ProxyServlet クラスは、ケーパビリティを表す URL を受け取り “/” を区切り文字として accesspoint 部、乱数部、および、乱数により復元する baseurl に対する相対 URL 部 (relurl) に分割する。データベースから各種情報を取得し、Web ページへアクセスするか判断する、Web ページの取得に関して、Apache Jakarta Project の Commons HttpClient [4] を使用した。

発行サーバは、JSP を用いて実装した。ケーパビリティ作成に必要な情報入力のための JSP(図 6) と、ケーパビリティ作成、および、作成結果の表示を行う JSP(図 7) から構成した。

5 実験

実装した機能の動作検証、中継による Web ページの取得時間の計測した。実験環境として以下のものを使用した。

- 閲覧者用端末
  - OS:Debian GNU/Linux 3.1
  - CPU:Intel Pentium4 3.00GHz
  - メモリ:512MB
- プロキシサーバ、発行サーバとして用いるマシン。
  - OS:Debian GNU/Linux 3.1
  - CPU:Intel Pentium4 3.00GHz
  - メモリ:512MB
  - Servlet コンテナ:Tomcat-4.1.31

プロキシサーバと発行サーバは同一マシン上で動作させた。

5.1 動作検証

ケーパビリティの生成、生成したケーパビリティを用いた Web ページの閲覧、弱いケーパビリティの生成、弱いケーパビリティの譲渡と使用、使用回数制限、有効期限制限の動作検証を行った。

(1) オリジナルの Web ページの用意
オリジナルの Web ページを Web サーバ上に作成した。これ以降、この Web ページのことをオリジナルの Web ページとよぶ。この Web ページの内容を図 8 に示す。

(2) ケーパビリティの生成
ケーパビリティの作成フォームからケーパビリティの作成を行った。作成フォームに、オリジナルの Web ページの URL、オリジナルの Web ページのアクセス制御に必要なユーザ名とパスワードを入力し、使用回数、および、有効期限に制限を加えずに送信を行った。送信された情

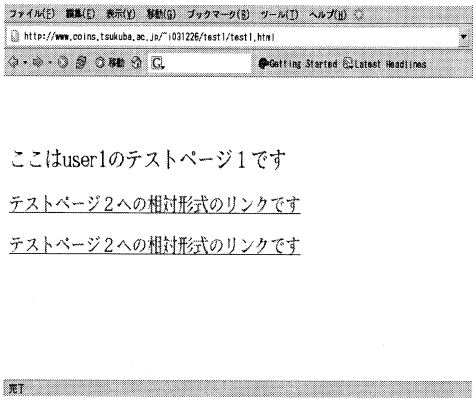

図 8: オリジナルの Web ページ

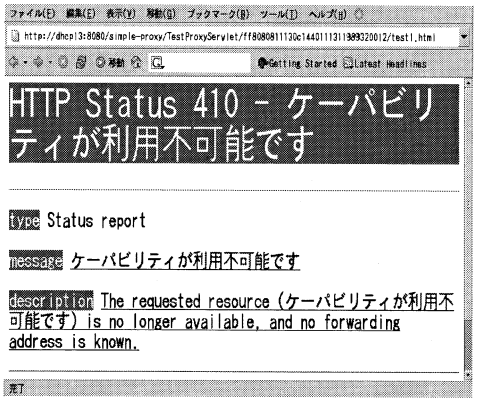

図 9: エラーを伝えるページ

報に基づいてケーパビリティが生成され、実行結果として、ケーパビリティを表す URL が生成された。

(3) ケーパビリティを用いた Web ページの閲覧
作成されたケーパビリティを用いたオリジナルの Web ページ中継の確認を行う。プロキシサーバに作成されたケーパビリティを送信する。プロキシサーバが送信されたケーパビリティを受け取りオリジナルの Web ページの中継が行われ、図 8 のオリジナルの Web ページと同じ内容が表示された。

(4) 弱いケーパビリティの作成
ケーパビリティの作成フォームから弱いケーパビリティの作成の確認を行う。作成フォームから先ほど作成したケーパビリティ、Basic 認証のユーザ名、および、パスワード、ケーパビリティの使用回数、ケーパビリティの有効期限の開始時刻、および、終了時刻を入力し送信する。送信した情報を使用して、ケーパビリティが生成された。

(5) 使用回数制限
ケーパビリティの使用回数制限の確認を行う。使用回数として 3 回の制限をつけた弱いケーパビリティを作成し、閲覧者に渡す。閲覧者側で、受け取ったケーパビリティを使用して Web ページの閲覧を行う。ケーパビリティをプロキシに 4 回目に送信した際に、図 9 の様にケーパビリティが使用できないことを確認した。これによりケーパビリティの使用回数制限が正しく動作することが確認できた。

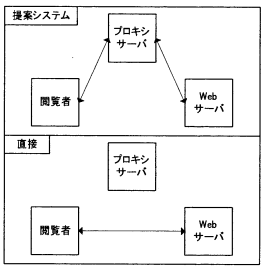

図 10: 経路図

(6) 有効期限制限
ケーパビリティの有効期限制限の確認を行う。有効期限の開始時刻を現在時刻より遅く設定したケーパビリティと、有効期限の終了時刻を現在時刻より早く設定したケーパビリティを作成した。作成したケーパビリティを、閲覧者に渡す。閲覧者側で、受け取ったケーパビリティを使用して Web ページの閲覧を行う。どちらのケーパビリティも、図 9 の様にケーパビリティが使用できないことを確認した。これによりケーパビリティの有効期限が正しく動作することが確認できた。

5.2 Web ページの転送時間の計測

プロキシサーバの中継による Web ページの転送時間の計測した。転送時間の計測には httperf[5] を使用した。時間の計測を行う対象としては、図 10 に示す 2 つの経路を用いた。提案システムは Web サー

バから直接オリジナルの Web ページを閲覧する場合である。直接はプロキシサーバを中継して Web ページを閲覧する場合である。

計測時間は各サイズ毎に 500 回計測した平均を用いた。実検の結果、提案システムでは、ファイルサイズが 32K バイトの場合、転送応答時間が 10.6 ミリ秒であった。この場合、直接 Web サーバをアクセスした時は、2.1 ミリ秒であった。このように、現在の実装では、プロキシを経由することで、約 8 ミリ秒の遅れが生じている。この時間は、インターネット上の通信遅延よりも小さい。このことから、プロキシを用いた実装による遅れは、主にネットワークの通信遅延により決定されるといえる。また転送時間は、提案システムでは、ファイルサイズが 32K バイトの場合、2.4 ミリ秒であり、応答時間ほど遅くない。その理由は、サーブレットによる実装のため、このサイズのファイルの場合、全体をため込んでから一度に送信しているためと思われる。

次に、単位時間当たり、受け付け可能な要求を測定した。結果から、提案システムでは、ファイルサイズが 32K バイトの場合、毎秒 200 程度の要求を処理することが分かった。通常の Web サーバでは、毎秒 1500 以上は処理できる。これと比較すると、現在の実装には性能上の問題が残っていると言える。今後は、別の実装方法を用いることを検討する。なお、プロキシを用いた方法では、DNS ラウンドロビンと組み合わせることで、複数の設置をすることで負荷 分散を図ることは容易である。したがって要求が集中した場合でも、対応できると考えられる。

6 結論

本研究では、Web ページに対するケーパビリティを用いたアクセス制御のプロキシによる実現を行った。ケーパビリティを用いたことにより、アクセス制御リストの変更やユーザ登録の変更を行うことなく、閲覧者自身が Web ページの閲覧の権利を一時的に委譲することを可能にした。既存の技術に変更を加えることなくケーパビリティを用いたアクセス制御のプロキシによる実現を行った。実装したプロトタイプについて、性能測定した。

今後の課題としては、発行したケーパビリティの無効化または制限の変更を行える仕組みを作ることである。また中継する Web ページに含まれる他の Web ページや画像への相対形式、および、 絶対形式のリンクの書き換えを行いたいと考えている。

参考文献

[1] Apache Tomcat 4 Documentation. http://tomcat.apache.org/tomcat-4.1-doc/index.html.

[2] Hibernate Reference Documentation. http://www.hibernate.org/hib_docs/v3/reference/en/html/.

[3] Hsqldb User Guide. http://hsqldb.sourceforge.net/doc/guide/index.html. 2005.

[4] Http Client User Guide. http://jakarta.apache.org/commons/httpclient/userguide.html.

[5] David Mosberger and Tai Jin. Httperf: a tool for measuring web server performance. *SIGMETRICS Perform. Eval. Rev.*, Vol. 26, No. 3, pp. 31–37, 1998.

[6] Sape J. Mullender, Guido van Rossum, Andrew S. Tanenbaum, Robbert van Renesse, and Hans van Staveren. Amoeba: A distributed operating system for the 1990s. *Computer*, Vol. 23, No. 5, pp. 44–53, 1990.

[7] RFC1808. Relative Uniform Resource Locators. 1995.

[8] RFC2068. Hypertext Transfer Protocol - HTTP/1.1. 1997.

[9] 馬渕充啓, 新城靖, 大木薫, 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の構想. 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, Vol. 2006-OS-102, pp. 9–16, 2006.