proxy による授業資料のアクセス行動の把握と他サイトへのアクセス制限

2007MI150 武藤 悠矢　　2007MI254 梅村 香輔
指導教員　後藤 邦夫

1　はじめに

近年, 情報化・ネットワーク化に伴いパーソナルコンピューター (以後 PC とする) の普及は著しいものとなっている. 参考文献 [1] によると, 授業に情報通信技術 (ICT) を活用している私立大学教員数は, 19 年度から 22 年度の 3 年間で全体の 2 割が増加した. また, ICT の主な活用として, 教室内にネットワーク環境を設置し, web 上にアップした自作講義資料ページがある. しかし, 教室内にネットワーク環境があるので, 学生が別の web サイトを閲覧したり, 授業内の小テストの回答をインターネットで探すなどの行為により, 授業を聞かないという問題点が増加している.

そこで本研究では, 学生に対して授業への参加強制, 授業中の行動把握という視点から上記の問題解決のための授業支援 proxy(以後 *TASP:Teaching Assistance Special Proxy* とする) を試作する. まず, 私立大学教員が現在の授業内での教員側が感じている問題 [1] を明らかにし, それらを考慮して 必要なシステムを提案, 実現する.

本研究の成果として, Apache2 のアクセスログを確認することで学生の貸与 PC が特定できるため, 学生本人が本当に参加しているか判断でき, 授業に関係ない web サイトをアクセス制限することで授業中の行動を制限することができるので, 授業の効率が良くなる.

なお, 武藤は主にシステム考察, システムの動作手順, 図の作成を, 梅村は主にシステムのプログラム, 文書作成を担当した.

2　システムの概要

本節では, システムの概要について説明する.

2.1　前提

1. 大学側のルータの設定は, 管理権限がないので変更できない.
2. 各学生が proxy 設定するのは手間がかかる.
3. 講義受講者人数は 200 人以下とする [2].
4. 大学内は DHCP サーバにより, IP アドレスを適当に学生の PC に割り振っているので, DNS 逆引きによって学生の貸与 PC が特定できる.

ネットワーク構成を次の図 1 に示す.

学生が web サイトにアクセスするとき, 教員 PC の web サーバにアクセスして *TASP* を介して web サイトにアクセスするようになっている.

2.2　教員が学生に感じている問題

現在教員側が感じている問題点を以下の 表 1 に示す.

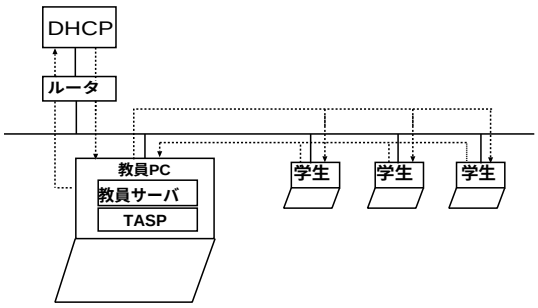

図 1　ネットワーク構成図

表 1 より, 「授業に出席しない」,「学習意欲が低い」という点に着目する. 本研究ではこれらを軸にシステムを提案する. 「授業に出席しない」については, 出席をとる時間だけ教室にいる, または友人に代返を頼むといったことも考慮する. また「学習意欲が低い」とは, 学生が情報通信機器の授業とは関係のないことをしており, 授業に集中してないことが主な原因である.

表 1　授業で直面している問題点 [1](学生に関する問題)

問題点	私立大学	私立短期大学
	20,543 名	1,359 名
授業に出席しない	8.1%	5.3%
教員の言葉を理解できない	8.7%	12.3%
基礎学力が低い	42.5%	52.1%
学習意欲が低い	36.8%	33.8%
自発的に質問・発言しない	40.7%	35.9%
その他	8.9%	8.2%

2.3　必要なシステムの機能

今回開発するシステム一覧

- web サイトの閲覧制限
  - 許可リストを作成して授業に関係ない web サイトを閲覧制限することで, 学生が授業中に他サイトを閲覧していて授業に関係ないことをするのを防ぎ, 学習意欲を高める.
- アクセス履歴の記録
  - アクセス履歴をとることにより, 学生が授業に本当に参加しているか確認できる. また, 教員がプログラムファイルをダウンロードする指示をしたときに指定したページを GET したか履歴に残し, 学生が指示した通りにダウンロードしたか判別し, 学生がどれほど授

業に集中しているか判断できる.

2.4 システムの仕様

本節では, システムの仕様についてユースケース図 (図 2, 3) とシーケンス図 (図 4) を用いて説明する. また, ユースケース図は教員側からの視点と学生側からの視点に分ける.

- 教員側のユースケース図 (図 2)
  1. 教員が web アクセスするとき, proxy に HTTP 要求する.
  2-3. proxy が web サイトに HTTP 要求, 応答する.
  4. リンク先の html 文書内にあるリンクタグで囲まれた URL を抜き出す.
  5. 抜き出した URL を許可リストに追加する.
  6. proxy が教員に HTTP 応答を返す.

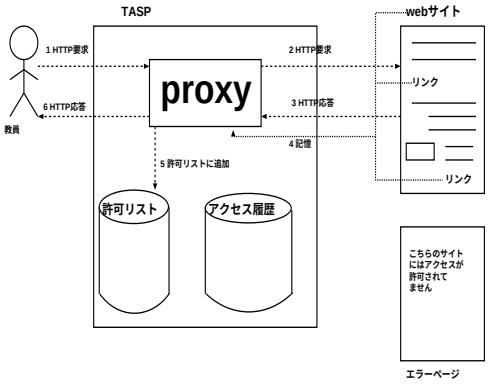

図 2　教員側のユースケース図

- 学生側のユースケース図 (図 3)
  1. web サイトにアクセスしようとする.
  2. proxy はその web サイトが許可リストに登録されているか判定する.
  3a. 登録あり:その web サイトに HTTP 要求をし結果を表示する. そして項目 4 へいく.
  3b. 登録なし:エラーメッセージを表示する.
  4. 学生のアクセス履歴を記録する.
- シーケンス図
  以下にシステムのシーケンス図 (図 4) を示す.
  - 教員側のシステムの流れ
    1. 教員の PC が proxy に HTTP 要求する.
    2-3. proxy が Web サーバに HTTP 要求, 応答する.
    4. proxy が許可リストを更新する.
    5. proxy が教員の PC に HTTP 応答する.
  - 学生側のシステムの流れ
    6. 学生の PC が proxy に HTTP 要求する.
    7-8. proxy は許可リストの要求, 応答する.
    9-10. proxy が Web サーバに HTTP 要求, 応答する.

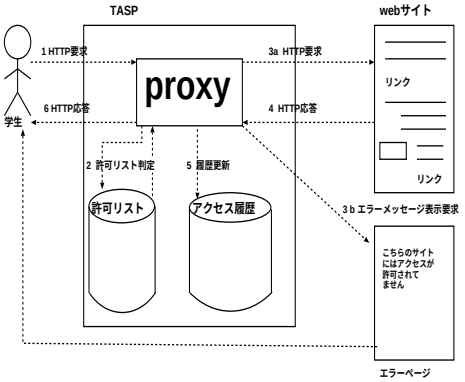

図 3　学生側のユースケース図

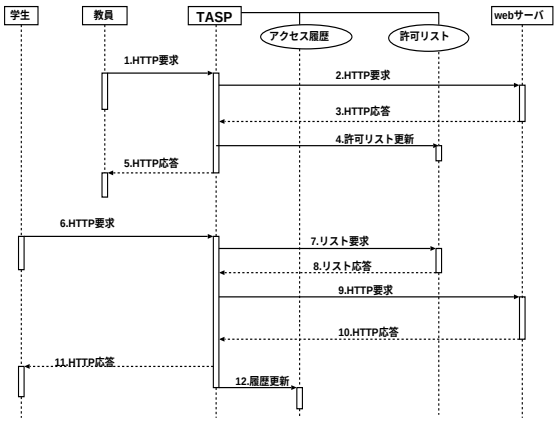

図 4　システム全体のシーケンス図

  11. proxy が学生に HTTP 応答する.
  12. proxy が学生のアクセス履歴を更新する.

3 システムの実現

本節では, 本研究でのシステムの実現を説明する.

3.1 実現方法

1. サーバとルータの間に *TASP* を入れる.
2. 既存の squid や apache といったソフトウェアの改造し *TASP* を作る.
3. Common Gateway Interface(CGI) プログラムと普通の web サーバを組み合わせ *TASP* を作る.

方法 1 は, 前述の前提での制約により実現できない. 方法 2 は, アクセス方法が通常の proxy と異なり, それらを改造するとなると非常に困難である. 方法 3 はルータと学生 PC との間に擬似的な proxy システムを介入することができ, 前提の条件も満たすことができる. また CGI プログラムは手軽に作成が可能である.

上記の理由により, 本研究では方法 3 を採用する. また, システムの開発時のプログラム言語は, 本システムでは文字列操作を多用するので, 文字列操作では他の言

語より優れている perl を用いて開発する.

3.2 プログラム構成

以下の図 5 にプログラム構成図を示す. まず *TASP* を 3 つのシステムに分け, さらにそれらに必要なモジュールに細かく分解する.

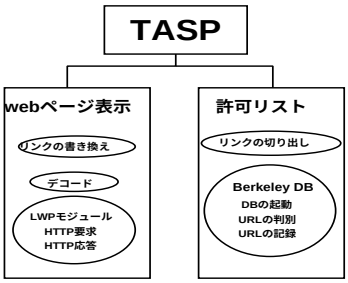

図 5　プログラム構成図

3.3 web サーバアクセス

web サーバに HTTP 要求し, それをブラウザ側に表示する proxy 本来の動作の実現について説明する.

- 使用したモジュール
  本システムの web サーバアクセスの動作を実現するにあたり, LWP モジュールを使用する. これらのモジュールは Perl での World Wide Web アクセスのためのモジュールである. このモジュールを用い web サーバへ HTTP 要求, 応答する. また, web サーバアクセス部分のプログラムに使用したクラスを以下に示す.
  - LWP::UserAgent リクエストを受け取るオブジェクト
  - HTTP::Request HTTP リクエストを送信する
  - HTTP::Response HTTP 応答を受け取る
- URL のデコード
  メソッド GET で URL を CGI へ渡すときに URL 内に含まれている記号がエンコードされてしまう. よってデコードする.
- リンク先へのアクセス
  表示されたページ内にリンクがあった場合, それらのリンクにアクセスしようとすると, *TASP* を経由できなくなる. それを防ぐために HTML 文書内にあるリンクを書き換える. 以下に書き換えの例を以下の表 2, 3 に示す.

表 2　元の HTML 文書

<A HREF=”http://URL1”>
<a HREF=”http://URL2”>
<A HREF=”URL3”>

表 3　書き換え後の HTML 文書

<A HREF=”…/proxy.cgi?url=URL1”>
<A HREF=”…/proxy.cgi?url=URL2”>
<A HREF=”…/proxy.cgi?url=http://URL3”>

表 2 の HTML 文書のように <A HREF が大文字と小文字で混ざっているものでも…/proxy.cgi?url=を追加して書き換え可能である. また URL5 のように省略されているものの前にも同じように追加されて書き換え可能である.

3.4 許可リスト

- 使用するモジュール
  処理速度やプログラム量を考えた際データベースを用いると効率が良い. そこで許可リストのシステムは berkelyDB を用いる.
  使用したクラス
  - BerkeleyDB::HASH：任意のキー/値の組をデータファイルに格納できる.
- 許可リストの内部的動作
  システムの流れを図 6 のフローチャートで示す.
  1. GET メソッドで送られてきた URL パラメータを受けとる (C++ CGI モジュール使用).
  2. GET 要求の URL がアクセス許可リストに入っていなければ, HTTP エラー応答を Web ブラウザに返す. 許可リストに入っているならば, (3) へ進む.
  3. その URL のサーバに対して, HTTP でリクエストを送って, 返事をもらう.
  4. 返事の HTML テキスト部分を書き換える.
  5. Web ブラウザに (3) の結果を送る.

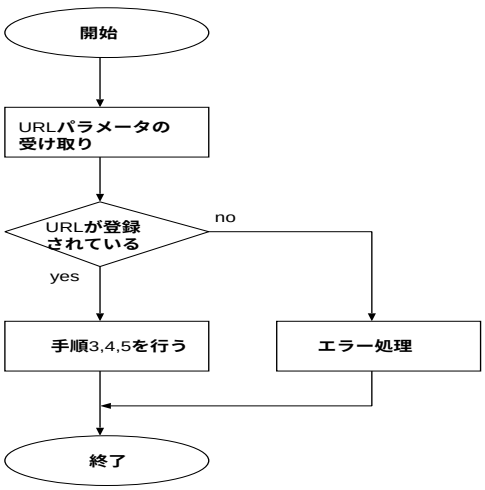

図 6　許可リストのフローチャート

- 許可リストの更新

教員が新しいページにアクセスした場合に許可リストをリアルタイムで更新する．これらは重複で更新してしまわないよう, データベースを用い管理する．

- 許可済 URL 判定
  学生が HTTP 要求するときにその URL が許可リストにあるか判定する．講義内ということで多くの学生からのアクセスが予想される．よってデータベースの検索機能を用いて, サーバに負荷がかからないよう処理速度をあげる．

3.5 学生のアクセス履歴

- 実現方法
  アクセス履歴を出力する方法として, 以下の 2 つの方法が考えられる．
  1. Apache2 の log を見る方法
  2. syslog を使用する方法

  本研究では, 方法 1 を採用する．方法 1 を採用する理由は, CGI-proxy プログラムに apache が含まれており, 学生が本研究のプロキシを経由する度にアクセスログを記録するからである．また,「何を記録するか」「どのファイルに記録するか」といったことを管理者が自由に定義できるからである．
  方法 2 の利点として, 現在の UNIX/Linux システムに標準的に備わっており, 順応性が高いことと, 複数のアクセスの同時書き込みできることである．しかし本研究で必要なアクセスログの情報は Apache の log に記録された情報で足りているため, 今回は簡単に確認できる方法 1 を採用する．

- 本研究に必要なアクセスログ一覧
  1. IP アドレス (DNS 引いてドメイン表示できる)
  2. 日時
  3. アクセス場所

  学内では DHCP サーバで IP アドレスが配られているため, ドメイン名が学生番号．教室.seto-private. となっている．なので Apache の記録でドメイン名をみれば学生番号がわかる．

4 システムの実験と評価

この節では, 実験とその評価の詳細を述べる．この実験により確認する点は, 学生が一斉にアクセスしたときにどの程度の人数まで耐えられるのかである．2.1 節の前提により, 200 人以上の負荷に耐えられなければいけない．まず, 本システムが正確に動作するか確認する．確認手順を次に示す．

4.1 動作確認手順

1. 学生側から *TASP* を経由して許可リストにない web サイトにアクセスする．
2. 教員側から手順 1 で学生がアクセスした web サイトにアクセスし, 許可リストの中身を確認する．
3. 再度学生が手順 1 の web サイトにアクセスする．

4.2 システムの実験方法

Apache2 では, インストールしたときに ApacheBench と呼ばれるベンチマークツールが付属されている．収録先ディレクトリは, /usr/sbin/ab である．ab コマンドを実行し, 本システムに負荷をかける．実行方法は次の通りである．

ab コマンド

```
ab -n [連続アクセス数] -c [同時アクセス数] http://[proxy.cgi]
```

4.3 サーバ負荷実験の結果

本システムでアクセスが集中する部分 URL に着目し, 同時アクセス数を 100 から 300 の間にして ab コマンドを実行した結果, 280 人までの負荷に耐えられることができた．

実験結果の一部

```
Concurrency Level:      280
Time taken for tests:   228.170 sec
Complete requests:      2000
Failed requests:        0
Write errors:           0
```

実行結果の内, Failed requests はリクエストに対して失敗した要求の数を示しているので Failed requests の値が 0 であることから無理な負荷はかかっていないことがわかる．この結果より, 本システムは 200 人の学生が一斉にアクセスして負荷をかけても, 耐えられることが確認できる．

5 おわりに

本研究では, 現在の私立大学教員が感じている問題点をあげてきた．今までには, 授業内にアニメーションや映像を取り入れたりと様々な工夫がされてきている．しかしそれに現代の学生達は慣れつつある．そこで現在学生の立場であることを利用し, 教員が求めるものと, 学生が求めるもの, 両方の立場からのシステムを開発した．最終的な目標として, 本研究の文頭であげられている問題点を改善するシステムになっていること．また, 現在の大学教員がぜひ使いたいと言って頂けるシステムを実現可能にすることである．

参考文献

[1] 公益社団法人私立大学情報教育協会：私立大学教員の授業改善白書 (May 2011).

[2] 中井俊樹：クラス規模は授業にどのような影響を与えるのか，名古屋高等教育研究. vol.6，pp. 5–19 (2006).