演習室の利用状況を求めるシステムの処理分担による効率化の研究

97H718 藤原 裕司
huziwara@h-ps011.ise.osaka-sandai.ac.jp

2001年 2月 19日

目 次

1 はじめに **1**

2 前卒研生の研究内容について **2**
2.1 前提・方法 2
2.2 仕組み 2

3 作成したプログラムとネットワークツール MRTG との連係について **3**
3.1 前卒研生のシステムの問題点と改善策 3
3.2 作成したプログラムとネットワークツールとを連係する利点 4
3.3 ネットワークツール MRTG を使う理由 5
3.3.1 MRTG とは 5
3.4 MRTG と連係する外部プログラムについて 6
3.4.1 作成したシステムについて 6
3.4.2 MRTG から呼び出される外部プログラムの仕組み 6
3.4.3 外部プログラムのフローチャート 7
3.5 全体・各クライアントマシンの稼働状況 9
3.6 全体の詳細な稼働状況 10
3.7 各クライアントマシンの詳細な稼働状況 10

4 まとめ **11**
4.1 まとめ 11
4.2 実現できなかったウィンドウズマシンのログイン状況に関する考察 11

5 謝辞 **12**

1　はじめに

今日のコンピュータの普及、発達には目覚しいものがある。そして、企業、大学といった教育機関にも実習や研究等のために演習室があり、その管理方法が問われている。演習室では、データの共有、プリンタの共有等を目的に LAN を利用している場合が多い。こういった演習室で、数十台以上のクライアントマシンを管理するのは困難である。そこで、管理者は、数十台以上のクライアントマシンを管理する上で、効率良く管理できることを望んでいる。本論文では、LAN で繋がっている数十台以上のクライアントマシンを管理する上で、管理者、場合によっては利用者に対して、演習室の各クライアントマシンの稼働状況等の提供をすることを目的とする。

2　前卒研生の研究内容について

2.1　前提・方法

管理者が、演習室の個々のクライアントマシンを管理運用することはかなりの手間がかかる。これをすこしでもやわらげる。その方法として、長期間使われていないクライアントマシンを探すこと (長期間使われていないということは、故障等の不具合のある可能性があるからである) によって故障等の不具合のあるクライアントマシンを探すシステムを前卒研生が作成した [1]。

方法としては、LAN 上にあるクライアントマシンが「ネットワーク的に繋がっているか」の情報を調べる。これにより、クライアントマシンが動作しているかを調べる。そして、その情報を集計することによって求めたクライアントマシンの利用状況によりある程度、状況が知ることができる。

2.2　仕組み

設定ファイルを読み込む。その設定ファイルに対象となるクライアントマシンの IP の範囲や使用コマンドのパス等、ログ等を排出する場所をあらかじめ記述しておく。そして、それを元にブロードキャストアドレスへの ping を行う。さらに、arp コマンドを使い、arp テーブルへ問い合わせを行う。他にも DNS に逆引きを行なうことによって、クライアントマシン名を調べる。それらの情報、すなわち、起動しているマシンとしていないマシン等の情報を元に集計する。そして、クライアントマシンのリストを作成し、指定したディレクトリにログとして出力する。そして、そのログを元にして利用状況を記述した HTML 形式のファイルを作成する。動作は、crontab により、指定した間隔ごとに起動する。それで、作成した HTML 形式のデータを Web サーバーに置いておくことで、ブラウザ上より閲覧できる。よって、管理者は、利用状況を閲覧できる。

前卒研生の仕組みでは、図 1に示すように、クライアントマシンの状態から、ログの整理・解析までのすべての処理を自作のプログラムで行なっている。

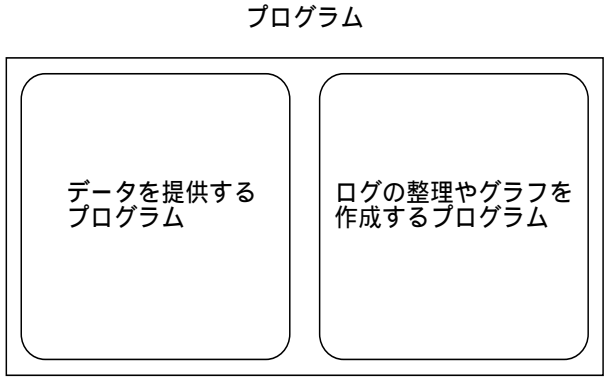


図 1: 前卒研生のプログラムの全体

3　作成したプログラムとネットワークツール MRTG との連係について

前卒研生が作成したシステムでは、新たに提供情報を追加した場合や提供したい情報を変更・追加した場合にシステムを大幅に変更しなくてはならない。そこで、処理分担により効率化したシステムを提案・作成する。

3.1　前卒研生のシステムの問題点と改善策

前卒研生が作成したシステムは、クライアントマシンの情報の取得、データの保存、整理・解析、グラフ等の提供情報への変換等をすべて自作プログラムで処理している。

このシステムの問題として、

- 今後、提供したい情報を増やしたい場合 (クライアントマシンに使用者 (ログイン者) がいるか等)
- 演習室の状況を分かりやすくしたい場合 (グラフ、統計情報等)

といった場合に、システムを訂正し情報の処理方法等を考え直し作成する必要がある。

前卒研生が作成したシステムの問題を解決する。そのために、情報の提供方法等を変更する際に、変更が多い部分の処理を切り離し、その処理を分担することで解決することにした。前卒研生が作成したシステムの処理としては、以下である。

- 利用状況等の情報を取得する部分
- その情報を整理・解析する部分

この内、「利用状況等の情報を取得する部分」は、必然的に自作プログラムにより作成する必要がある。しかし、「その情報を整理・解析する部分」は、ログのデータをグラフ化等しているだけである。そこで、その部分の処理をネットワークツールで行なうことにより、効率化することにした。すなわち、図 2のように、「利用状況等の情報を取得する部分」「その情報を整理・解析する部分」とに処理を分担する。そして、「その情報を整理・解析する部分」をネットワークツールを使うことによって効率化したのである。

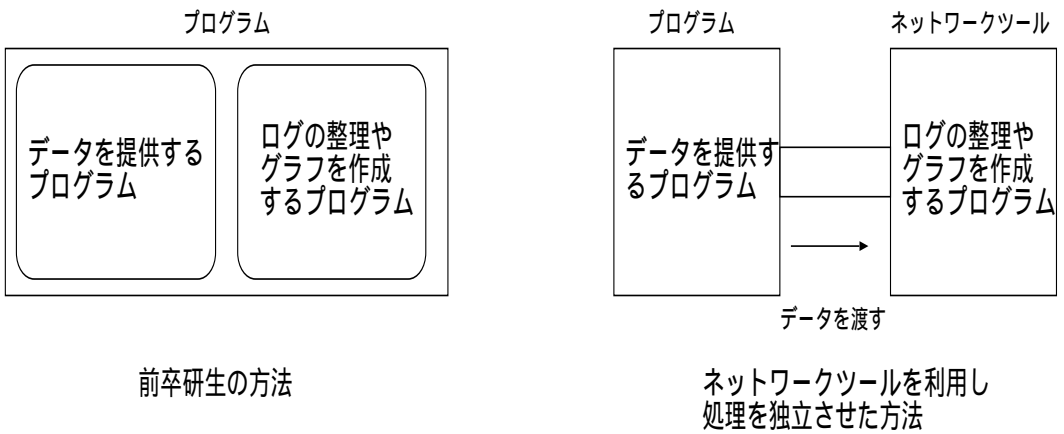


図 2: 以前の方法と発展させた方法

3.2　作成したプログラムとネットワークツールとを連係する利点

役割を分担することによって、プログラムは、データの取得と最新データをネットワークツールに渡すだけでよくなる。ネットワークツールは受け取ったデータをログとして整理・保存、解析しグラフ等を作成してくれる。よって、プログラムとしては、データの整理・解析等の処理を行なわなくてもよくなるのである。

他にも、プログラムとネットワークツールを連係する利点としては、ネットワークツールがデータを受け取る際にその出力形式が同じならば、容易に他のネットワークツールでログ処理や解析、提供できる情報を変更することも可能になるのである。すなわち、データを提供されたネットワークツールが、例えば、自動でログ等を整理・解析し、グラフや統計情報等を作成してくれるのである。しかも、データが同じ場合にネットワークツールを変えることによって閲覧できるデータの変更しやすい。すなわち、インタフェースがある程度、固定されているので変更しやすいのである。他にも、バージョンアップにも容易に対応できる。よって、図 3のような効果がえられる。

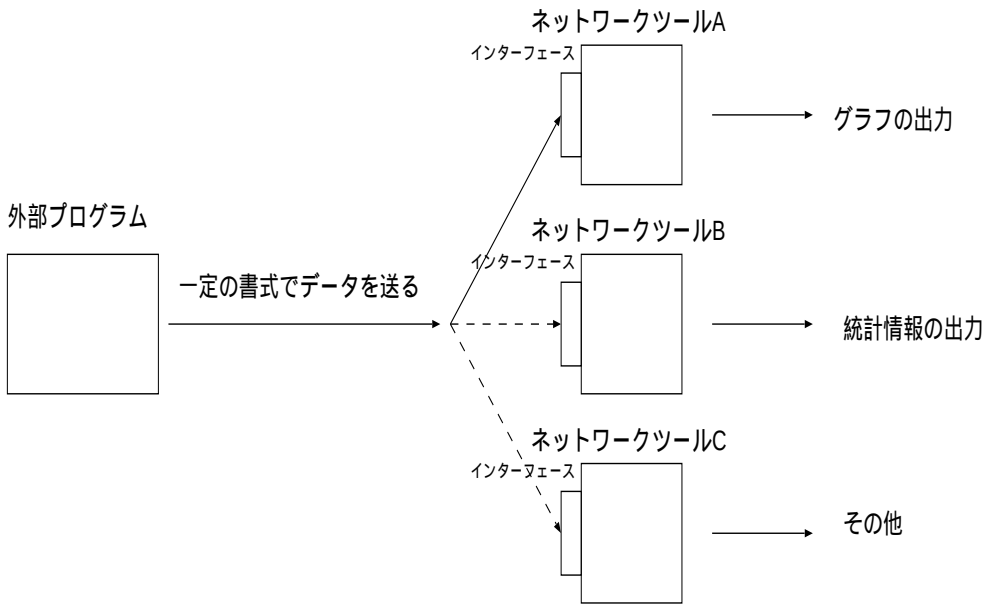


図 3: プログラムからネットワークツールへの流れ

よって、まとめると、

- 提供したい情報が増えた場合はネットワークツールとの連係を前提に作るのでデータ取得・出力のみでよくなる
- ログ等の整理・解析はネットワークツールが処理してくれる。そして、グラフ等も作成してくれる。よって、効率化できる

のようなことが可能となるのである。

3.3 ネットワークツール MRTG を使う理由

今回、ネットワークツールとして、MRTG を使用することにした。MRTG は本来、トラフィックや SNMP 変数を測定しグラフ化するソフトである。しかし、外部プログラムを使用することによって、グラフ化したいデータを一定の書式で出力することによってどんなデータでもグラフ化できる。ほかにも、MRTG は、普及率が高く、ほとんどの OS で動作し、外部プログラムは、データを出力できるものであれば、プログラム言語も選ばないといった理由により利用することにした。

3.3.1 MRTG とは

MRTG(Multi Router Traffic Grapher) とは、SNMP(Simple Network Management Protocol) を使ってルータやマシンの NIC 毎のネットワークトラフィックやあらゆる SNMP 変数を監視しグラフ (日毎、週毎、月毎、年毎) に表示してくれる監視ソフトである。

他にも特徴として、他の外部プログラムを使用して、それらから出力されるデータを収集し、グラフ化することもできる。

方法としては、外部プログラムは設定ファイルから呼び出され実行される。外部プログラムは実行した際にグラフ化したいデータ

- 1 行目：In の値 (整数、必須)
- 2 行目：Out の値 (整数、必須)
- 3 行目：システムが起動してから経過した時間 (uptime)
- 4 行目：値の説明

を以上のように渡す。MRTG はこの情報を元にグラフ化を行なう。

本研究ではこの機能を使って、作成したプログラムから出力されるデータをグラフ化している。

グラフは図 4のように出力される。グラフのデータは PNG 形式で保存される。

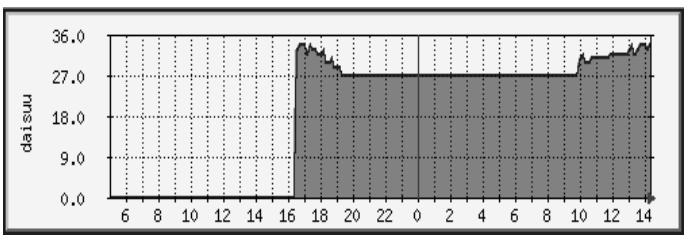


図 4: グラフのサンプル

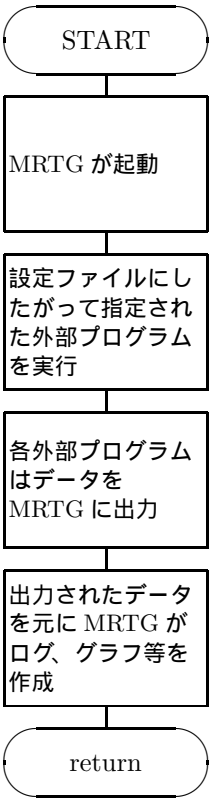
他に HTML 形式のデータも同時に出力される。なお、グラフ、HTML 形式のデータは設定ファイルにより表示方法を自由にカスタマイズできる。

3.4 MRTG と連係する外部プログラムについて

3.4.1 作成したシステムについて

前卒研生のマシンの稼働状況を求めるプログラムの問題を解決するために、今回既存のネットワークツール MRTG を使用している。そして、MRTG とプログラムが連係できるように、前卒研生の方法を利用してプログラムを作り直した。

原理としては、まず、プログラム (ファイル名：login_cfg.pl) より作成した MRTG の設定ファイル作成プログラムにより MRTG の設定ファイルを作成する。そして、若干の手直し (プログラムのパス等の変更) をして設定ファイルを完成させる。そして、MRTG を crontab 等で定期的に動作させる。他にも、このプログラムにより、各クライアントマシンの稼働状態を MRTG に渡すプログラムが作成される。

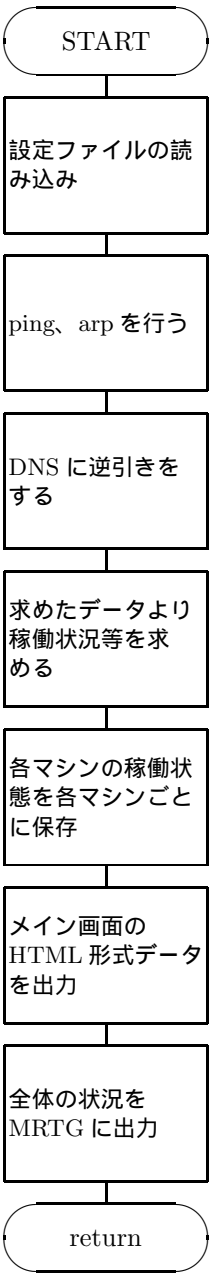


システムの全体の流れ

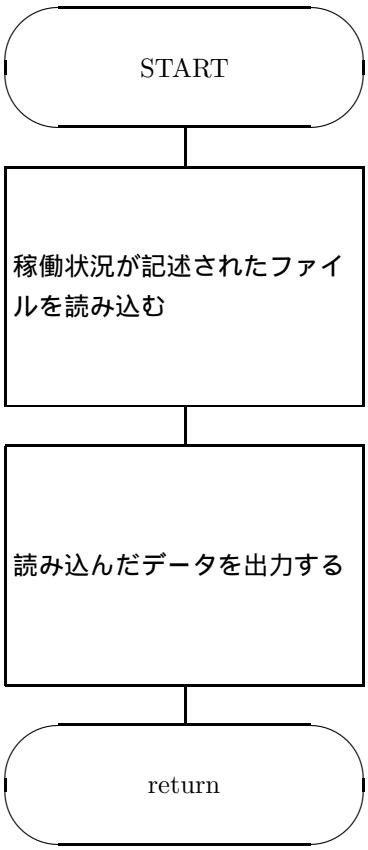
3.4.2 MRTG から呼び出される外部プログラムの仕組み

稼働状態判定プログラムは、各クライアントマシンがネットワーク的に繋がっているかどうか (稼働しているかどうか) を調べる。そして、各クライアントマシンに対して、繋がっているどうかを記述したファイルを作成する。そして、最後に HTML 形式のファイルを作成して MRTG に全体の状況を出力して終了する。処理は MRTG に戻り、全体の状況が記述されたグラフ等が作成される。そして、残りの処理、すなわち上記で作成した各クライアントマシンの状況を MRTG に出力するプログラムが呼び出され残りのグラフ等が作成される。

3.4.3 外部プログラムのフローチャート



稼働状況判定プログラムの流れ



各クライアントマシンの状況を MRTG に渡すプログラムの流れ

3.5　全体・各クライアントマシンの稼働状況

図 5は、稼働状況判定プログラムにより作成している。グラフ等のデータは MRTG により作成されたものを使っている。グラフにリンクが張ってあるので、図 6、図 7ような年毎、月毎、週毎、日毎のグラフを閲覧することができる。

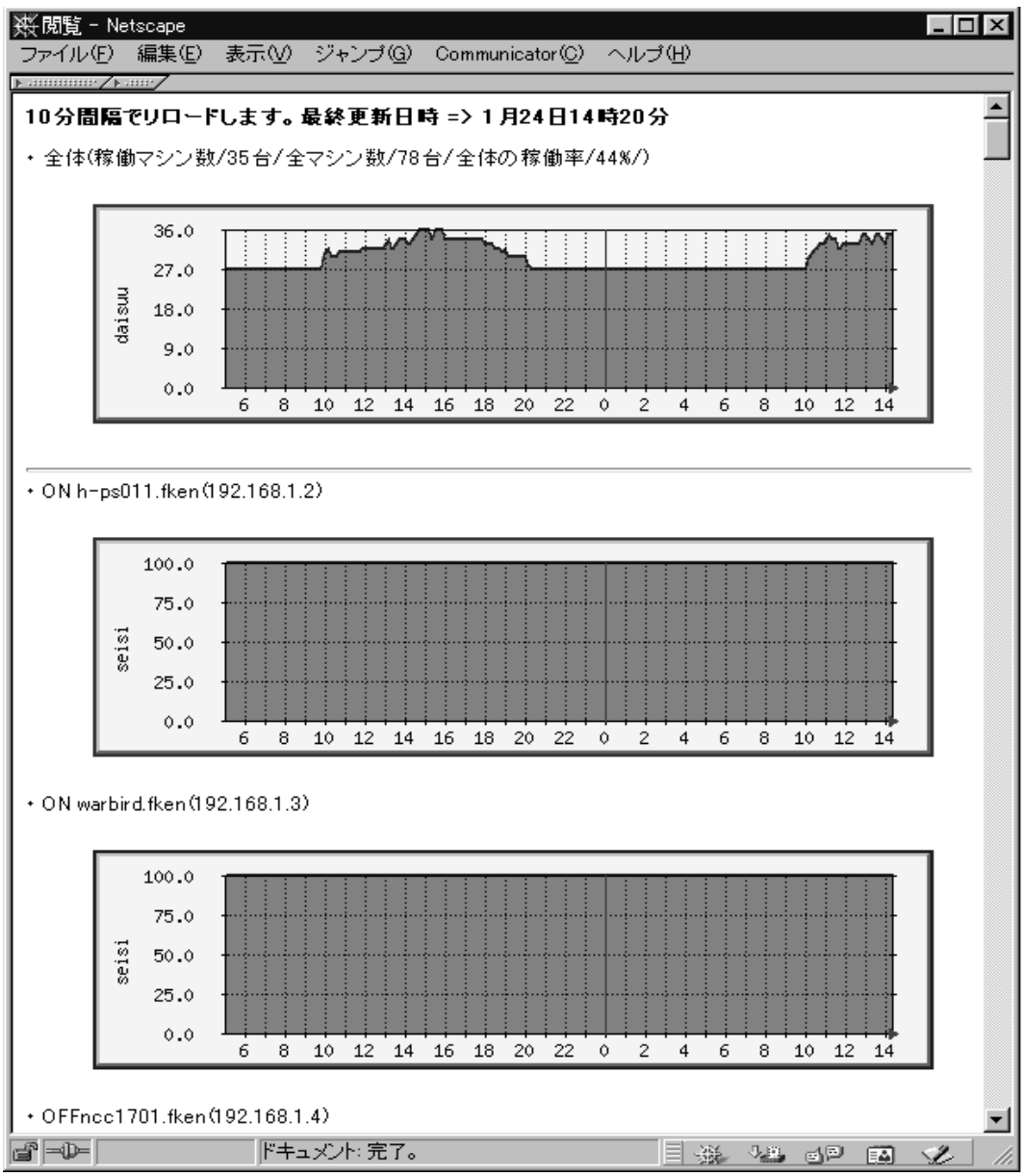


図 5: 全体・各クライアントマシンの稼働状況

3.6　全体の詳細な稼働状況

図 6は、全体の稼働状況を年毎、月毎、週毎、日毎で表示している。グラフは現在、起動しているクライアントマシンの台数の推移を表す。

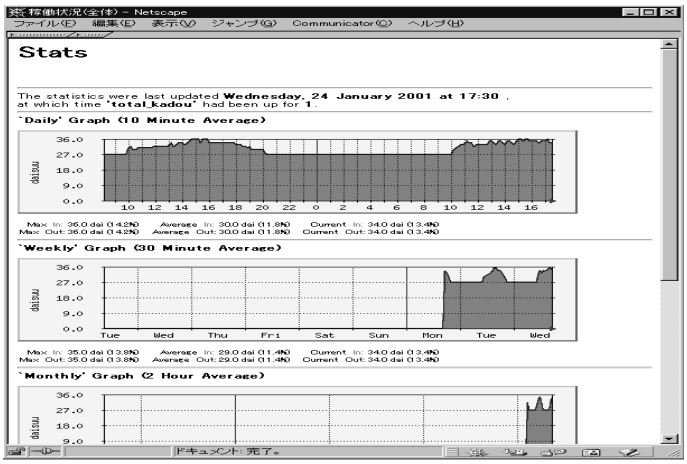


図 6: 全体の稼働状況をブラウザで表示した図

3.7　各クライアントマシンの詳細な稼働状況

図 7は、クライアントマシンの稼働状況を年毎、月毎、週毎、日毎で表示している。グラフの値が 0 の場合は起動していない状態で、それ以外の部分が稼働している状態を表す。

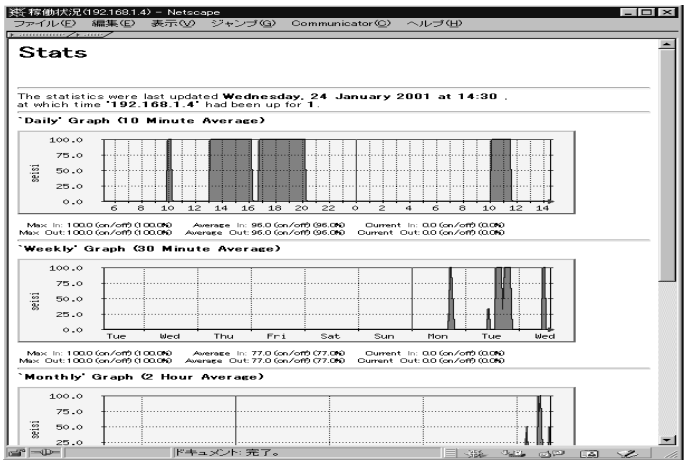


図 7: クライアントマシンの稼働状況を表示した図

4　まとめ

4.1　まとめ

プログラムとネットワークツール MRTG を連係させることによって、演習室等の稼働全マシン数、各マシンについて、現在と過去 (日毎、週毎、月毎、年毎) の状況を Web 上のグラフから容易に閲覧できるようになった。

4.2　実現できなかったウィンドウズマシンのログイン状況に関する考察

新たな情報を提供するためにログイン者を調べる。これを調べることができれば、演習室のマシンの使用状況が分かる。これにより、管理者は演習室のマシンの使用状況が把握でき、利用者は演習室の比較的空いている時間を狙って利用できる。

LAN で構築されたネットワーク内のクライアントマシンには、UNIX 系 OS、ウィンドウズ系 OS、他の OS が混じっている場合がある。UNIX 系 OS のクライアントマシンでは、ログイン状況は rwho コマンドを使用する、もしくは、syslogd 等からのログを解析することにより調べることができる。しかし、ウィンドウズ系 OS では、そういったものが存在しない。そこで、ウィンドウズ系 OS のクライアントマシンのログイン者を調べる方法を考えた。その方法としては、ファイルサーバーのログにウィンドウズ系 OS のクライアントマシンのログイン者等の情報があるかを調べた。しかし、調べた結果、そういったものは存在しないようだった。そして、LAN 内のクライアントマシン、各種サーバーにさまざまな OS が存在する場合を考慮に入れた場合、ログイン者の情報がのっているログが存在した場合でも、そのログがどの OS のサーバーに存在したかやログの出力方式によって、ログを解析するプログラムが毎回変わってしまう。そういった問題が発生するためにログからログイン状況を調べることができなかった。これについては、今後の課題である。

5　謝辞

本研究を進めていく上で、ご指導いただいた大垣 斉講師、藤井 信夫教授、中村 孝講師に深く感謝の意を表します。

参考文献

[1] 1999 年度卒業生の中沼 崇の卒業論文
http://www-fken.ise.osaka-sandai.ac.jp/fken99g/

[2] UNIX コマンド例文事例 csh/tcsh 編
officeMASA
2000 年
ISBN 4-7741-1015-9

[3] Perl 入門第 2 版
エリー・クイグリー
2000 年
ISBN 4-89471-309-8

[4] 図解標準 最新 UNIX ハンドブック
伊藤 和人
2000 年
ISBN 4-87966-948-2

付録

クライアントマシンの稼働状態の取得等を行なうプログラム

ファイル名：login12_11.pl

```
#! /usr/bin/perl

# 設定ファイル場所等の設定
$s_file="/home/huziwara/proto/settei.conf";


# 設定ファイルの読み込み
open(SETUP, $s_file) || die "setupfile not found";
flock($s_file,2);
@SETUP=<SETUP>;
close(SETUP);
flock($s_file,8);
foreach(@SETUP){
$_=~s/\r\n|\r|\n//;
($HIKISU1,$HIKISU2)=split(/&/,$_);
$SET{$HIKISU1}=$HIKISU2;
#print $HIKISU2;
# print $SET{NETMASK};
}
# 時間の取得
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdat)=localtime(time);
$mon++;
$kadouritu_zitu=0;

# BROWDCAST ADD
($IP_M,$NET_M)=split(/-/,$SET{NETMASK});
#print "\n $NET_M";
$BC=2**(32-$NET_M)-2;
if($IP_M=~/(\d+)\.(\d+)\.(\d+)\.(\d+)/){
$NETADD="$1.$2.$3.";
$BCE=$4;
}
$BROWDCAST=$BC+1;

# ping コマンドの実行
open(PING, "ping -c2 $NETADD$BROWDCAST |");
@PING=<PING>;
close(PING);
shift(@PING);
```

```
shift(@PING);

# arp コマンドの実行
open(ARP, "$SET{ARP} -a |");
@ARP=<ARP>;
close(ARP);

$PING=@PING;
$ARP=@ARP;
$ARP=$ARP-1;
for($i=0; $i<=$ARP; $i++){
    # <incomplete>がある場合
    if($ARP[$i]=~/\((\d+)\.(\d+)\.(\d+)\.(\d+)\)\sat\s(\<\incomplete\>)/){
        #$ARP[$i]=~/\((\d+)\.(\d+)\.(\d+)\.(\d+)\)/;
        #if($NETADD eq "$1\.$2\.$3\."){
        #   $ARP{"$1\.$2\.$3\.$4"}=0;
        #   print "$1,$2,$3,$4***$ARP{\"192.168.1.$4\"}\n";
        #}
        #else{
        #print "$1,$2,$3,$4**\n";
        #}
    }
    else{
        $ARP[$i]=~/\((\d+)\.(\d+)\.(\d+)\.(\d+)\)/;
        if($NETADD eq "$1\.$2\.$3\."){ # && $5 ne "\<incomplete\>"){
          $ARP{"$1\.$2\.$3\.$4"}=$4;
          # print "$1,$2,$3,$4**\n";
        }

        #else{ # 外部
        #  print "$1,$2,$3,$4*\n";
        #}

    }
}

for($i=0; $i<=$PING; $i++){
    $PING[$i]=~/\s(\d+)\.(\d+)\.(\d+)\.(\d+)\:/;
    if($NETADD eq "$1\.$2\.$3\."){
$ARP{"$1\.$2\.$3\.$4"}=$4;
    }
}
```

```
# 実行マシンのデータ取得
$ARP{$SET{SERVER_IP}}=gethostbyaddr(pack('C4' ,split(/\./, $SET{SERVER_IP})), 2);


# ログの出力
    $count_on=0;
    $count_off=0;
$r_flag=0; # 履歴フラグ(OFF)
# LOOP START
for($i=$BCE+1; $i<=$BCE+$BC; $i++){
      $L_NETADD="$NETADD$i";
      $C_NAME=gethostbyaddr(pack('C4' ,split(/\./, $L_NETADD)), 2);
      if($C_NAME eq ""){
         # 名無し処理
         $C_NAME="NO-NAME";
      }


      # ---------------------------------------------------------------------
      # 今回、通ったかを調べて通っている場合に指定の処理(稼働率)
      if($ARP{$L_NETADD} ne ""){
          $seisi="ON ";
      }
      else{
      # NOT
          $seisi="OFF";
      }
      $R_NETADD="$L_NETADD.dat";

      $k_seisi[$i] = $seisi;
      $k_cname[$i] = $C_NAME;
      $k_netadd[$i]= $L_NETADD;

      # 全体のデータ取得(稼働率)
      unless($k_seisi[$i] eq "OFF" && $k_cname[$i] eq "NO-NAME"){
  if($k_seisi[$i] eq "ON "){ # 1
      $count_on++;
  } # 1
          else{ # 2
              $count_off++;
  } # 2
          # 状態フラグの保存
      open(HANTEI, ">$SET{PIR}/$R_NETADD");
```

```
flock($SET{PIR}/$R_NETADD, 2);
                print HANTEI "$k_seisi[$i]";
        close(HANTEI);
                flock($SET{PIR}/$R_NETADD, 8);

          # 各マシンの状態フラグ
          # ファイルが無かった場合（初回起動のみ）
          unless(-e "$SET{PIR}/$R_NETADD"){ # 3
            open(RIREKI, ">$SET{PIR}/$R_NETADD") || die "Log3 Write Error";
              flock($SET{PIR}/$R_NETADD, 2);
              print RIREKI "$seisi";
              close(RIREKI);
              flock($SET{PIR}/$R_NETADD, 8);
          }  # 3

      } # unless END
}
# LOOP END

$count_total=$count_on+$count_off;
$total_kadouritu=int($count_on/$count_total*100);

# mtrg print
$in=int($count_on);
$out=int($count_on);
$dami=int(0);

# ------------------------------------------------------------------------------
#HTML データへの書き出し（main）
open(HTM, ">$SET{HTML}") || die "HTML Write Error\n";
flock($SET{HTML}, 2);
seek($SET{HTML}, 0, 0);
print HTM "<HTML LANG="ja">\n";
print HTM "<HEAD>\n";
print HTM "<TITLE>閲覧</TITLE>\n";
print HTM "<meta http-equiv=\"Content-Type\" content=\"text/html\" charset=\"euc-jp\">\n";
print HTM "<meta http-equiv=\"Refresh\" content=\"600\">\n";
print HTM "</HEAD>\n";
#print HTM "<BODY background-color: #CCCCFF>\n";
print HTM "<h4>10 分間隔でリロードします。最終更新日時 => $mon 月$mday 日$hour 時$min 分</h4>\n";

print HTM "<li>全体（稼働マシン数\/$count_on 台\/全マシン数\/$count_total 台\/全体の稼働率
\/$total_kadouritu\%\/)</li>";
```

```
print HTM "<ul><a href=\"mrtg/total_p.html\"><IMG VSPACE=10 WIDTH=500 HEIGHT=135 ALIGN=TOP SRC=\"mrtg/tot


print HTM "<HR>\n";

for($i=$BCE+1; $i<=$BCE+$BC; $i++){
    # OFF & NO-NAME であるとき以外
    unless($k_seisi[$i] eq "OFF" && $k_cname[$i] eq "NO-NAME"){
if($k_seisi[$i] eq "ON "){
        # ON
           print HTM "<li>$k_seisi[$i]$k_cname[$i]($k_netadd[$i])</li>\n";
           print HTM "<ul><a href=\"mrtg/$k_netadd[$i].html\"><IMG VSPACE=10 WIDTH=500 HEIGHT=135 ALIGN=T
        }
else{
        # OFF
           print HTM "<li>$k_seisi[$i]$k_cname[$i]($k_netadd[$i])</li>\n";
           print HTM "<ul><a href=\"mrtg/$k_netadd[$i].html\"><IMG VSPACE=10 WIDTH=500 HEIGHT=135 ALIGN=T
        }
    }
    # $hito = $memo = ""; # syokika

}
print HTM "</BODY>\n";
print HTM "</HTML>\n";
close(HTM);
flock($SET{HTML}, 8);

# ------------------------------------------------------------------------

$dami="total_kadou";

print "$in\n$out\n1\n$dami\n";

exit;
```

MRTG 用の設定ファイル等を作成するプログラム

ファイル名：login_cfg.pl

内容：MRTG で使用する設定ファイルの基板作成と各クライアントマシン用の MRTG にデータを渡すプログラムを作成するプログラム

```
#! /usr/bin/perl

# 設定ファイル場所等の設定
$s_file="/home/huziwara/proto/settei.conf";



# 設定ファイルの読み込み
open(SETUP, $s_file) || die "setupfile not found";
flock($s_file,2);
@SETUP=<SETUP>;
close(SETUP);
flock($s_file,8);
foreach(@SETUP){
$_=~s/\r\n|\r|\n//;
($HIKISU1,$HIKISU2)=split(/&/,$_);
$SET{$HIKISU1}=$HIKISU2;
#print $HIKISU2;
# print $SET{NETMASK};
}

# 時間の取得
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdat)=localtime(time);
$mon++;
$kadouritu_zitu=0;

# BROWDCAST ADD
($IP_M,$NET_M)=split(/-/,$SET{NETMASK});
#print "\n $NET_M";
$BC=2**(32-$NET_M)-2;
if($IP_M=~/(\d+)\.(\d+)\.(\d+)\.(\d+)/){
$NETADD="$1.$2.$3.";
$BCE=$4;
}
$BROWDCAST=$BC+1;

# ping コマンドの実行
open(PING, "ping -c2 $NETADD$BROWDCAST |");
@PING=<PING>;
close(PING);
```

```
shift(@PING);
shift(@PING);

# arp コマンドの実行
open(ARP, "$SET{ARP} -a |");
@ARP=<ARP>;
close(ARP);

$PING=@PING;
$ARP=@ARP;
$ARP=$ARP-1;

for($i=0; $i<=$ARP; $i++){
    # <incomplete>がある場合
    if($ARP[$i]=~/\((\d+)\.(\d+)\.(\d+)\.(\d+)\)\sat\s(\<\incomplete\>)/){
        # kara
    }
    else{
        $ARP[$i]=~/\((\d+)\.(\d+)\.(\d+)\.(\d+)\)/;
        if($NETADD eq "$1\.$2\.$3\."){ # && $5 ne "\<incomplete\>"){
          $ARP{"$1\.$2\.$3\.$4"}=$4;
        }
    }
}

for($i=0; $i<=$PING; $i++){
    $PING[$i]=~/\s(\d+)\.(\d+)\.(\d+)\.(\d+)\:/;
    if($NETADD eq "$1\.$2\.$3\."){
$ARP{"$1\.$2\.$3\.$4"}=$4;
    }
}

# 実行マシンのデータ取得
$ARP{$SET{SERVER_IP}}=gethostbyaddr(pack('C4' ,split(/\./, $SET{SERVER_IP})), 2);
# ------------------------------------------------------------------------------
# LOOP START
for($i=$BCE+1; $i<=$BCE+$BC; $i++){
      $L_NETADD="$NETADD$i";
      $C_NAME=gethostbyaddr(pack('C4' ,split(/\./, $L_NETADD)), 2);
      if($C_NAME eq ""){
         # 名無し処理
         $C_NAME="NO-NAME";
      }
```

```
        # 今回、通ったかを調べて通っている場合に指定の処理（稼働率）
        if($ARP{$L_NETADD} ne ""){
        # ON
            $seisi="ON ";
        }
        else{
        # NOT
            $seisi="OFF";
        }

        $k_seisi[$i] = $seisi;
        $k_cname[$i] = $C_NAME;
        $k_netadd[$i]= $L_NETADD;

        # 各クライアントマシンに対して*.pl の作成
        unless($k_seisi[$i] eq "OFF" && $k_cname[$i] eq "NO-NAME"){
        $PL_NETADD="$L_NETADD.pl";
        unless(-e "$SET{PL}/$PL_NETADD"){
           open(PLL, ">$SET{PL}/$PL_NETADD") || die "Log3 Write Error";
                flock($SET{PL}/$PL_NETADD, 2);
                print PLL "\#\! \/usr\/bin\/perl\n";
                print PLL "\n";
                print PLL "open(ZYO, \"$SET{PIR}\/$R_NETADD\")\;\n";
                print PLL "flock\(\"$SET{PIR}\/$R_NETADD\", 2\)\;\n";
                print PLL "\@rireki=\<ZYO\>\;\n";
                print PLL "close\<ZYO\>\;\n";
                print PLL "flock\(\"$SET{PIR}\/$R_NETADD\", 8\)\;\n";
                print PLL "\$saisyu\=pop\(\@rireki\)\;\n";
                print PLL "if\(\$saisyu eq \"ON \"\)\{\n";
                print PLL "\$saisyu\=100\;\n";
                print PLL "\}\n";
                print PLL "else\{\n";
                print PLL "\$saisyu\=0\;\n";
                print PLL "\}\n";
                print PLL "print \"\$saisyu\\n\$saisyu\\n1\\n$L_NETADD\\n\";\n";
                close(PLL);
                flock($SET{PLL}/$PL_NETADD, 8);

            # パーミッションを 755 で作成したい場合にのみ#をはずす。
            # chmod 0755, "$SET{PL}/$PL_NETADD";
        }
        }
}
```

```
# LOOP END

# ------------------------------------------------------------------------------
# 書き出し (main)
open(HTM, ">$SET{MRTG}") || die "HTML Write Error\n";
flock($SET{MRTG}, 2);
seek($SET{MRTG}, 0, 0);

# 内容
print HTM "Workdir\: $SET{MRTG_WORK}\n";
print HTM "Interval:10\n";
print HTM "Options[_]:gauge,absolute,growright\n";
print HTM "Language:euc.jp\n";
print HTM "YLegend[_]:daisuu\n";
print HTM "ShortLegend[_]:dai\n";
print HTM "\# -Total- ----------------------------------------------\n";
print HTM "Target\[total_p\]\:\`perl $SET{PASS}\/login12_11.pl\`\n";
print HTM "MaxBytes\[total_p\]\: 254\n";
print HTM "Title\[total_p\]\:稼働状況 (全体)\n";
print HTM "PageTop\[total_p\]\:\<H1\>Stats\<\/H1\>\n";

# LOOP START
for($i=$BCE+1; $i<=$BCE+$BC; $i++){
    # OFF & NO-NAME であるとき以外
    unless($k_seisi[$i] eq "OFF" && $k_cname[$i] eq "NO-NAME"){
          print HTM "Target\[$k_netadd[$i]\]\:\`perl $SET{PL}\/$k_netadd[$i].pl\`\n";
          print HTM "MaxBytes\[$k_netadd[$i]\]\: 1\n";
          print HTM "Title\[$k_netadd[$i]\]\:稼働状況 ($k_netadd[$i])\n";
          print HTM "PageTop\[$k_netadd[$i]\]\:\<H1\>Stats\<\/H1\>\n\n";
    }
}
# LOOP END
close(HTM);
flock($SET{MRTG}, 8);

# ------------------------------------------------------------------------------

exit;
```