研究論文

鳴門教育大学情報教育ジャーナル No.9 pp.37-42 2012

情報基盤センターにおける時系列データ活用

曽根直人*

情報基盤センターでは，ネットワークやコンピュータなどさまざまな機器の運用・管理を行っている。その中には学内の情報基盤を支える重要なシステムもあり，利用者からは24時間，365 日の安定したサービスの提供が望まれている。重要なサービスの安定稼働のため，一定間隔毎にサービスの稼働状況を確認するヘルスチェックシステムを運用しており，サービスが止まればシステム管理者へ自動的に連絡が届くようにしている。またシステムの状況を把握するために様々なデータを集め，時系列のデータとしてグラフ化し，システムの状況把握に努めている。本稿では，不具合の予防や原因追究につながる時系列データの活用について述べる。

[**キーワード**：時系列データ，グラフ，状況把握，リソース管理]

1. はじめに

情報基盤センターでは数多くのサーバやネットワーク機器を管理・運用している。それらは学内の ICT インフラとして利用されており，安定した稼働を求められている。基盤センターシステム分野では安定した運用を提供するためにリモートからのヘルスチェックプログラムによるサービス監視やサーバのローカル環境でのサービス監視プログラムを利用し，システムの不具合を早急に検知し対策を行うようにしている。

不具合を発生させないための予防的な対策として，ネットワークトラフィックやシステムの状況を継続的に把握することで，リソース不足による不具合を予防したり，システムの異常を早期発見したりすることができる。安定したサービスを提供するには時系列データを継続的に取得し状況把握・監視を行うことが必要不可欠である。そこで我々はOSSのリソースモニターツールMunin[1]を利用し，さまざまなシステムの状況把握に利用している。本稿では情報基盤センターでの Munin を利用した時系列データの活用について述べる。

2. Munin

2.1. Muninの概要

MuninはCPUやネットワークそのほか様々なデータをモニターし，RRD Tool[2]を利用してグラフ化するOSSソフトである。同様の機能を持つ MRTG[3]や cacti[4]よりも導入が簡単にでき，pluginによりグラフ化するデータの対象を追加する拡張性も備えている。これは他のソフトウェアの多くがsnmp経由で観測するデバイスの値を取得するのに対して，Muninは munin-node と呼ばれるエージェント経由で観測デバイスの値を取得するためである。

Munin 本体は Perl で記述されており多くの OS 上で動作する。但しデータを収集するための plugin の一部は機種に依存したバイナリ形式の実行ファイルとなるため動作しない場合がある。情報基盤センターでは，Debian/GNU Linux Version 5(squeeze) amd64 に Munin を導入し運用している。Debian であれば導入は

```
apt-get install munin munin-node
```

で完了する。

2.2. pluginによる拡張

Munin では plugin を利用することで観測対象を拡張することができる。情報基盤センターにおいてもこの拡張機能を利用し，標準では対応していない観測対象を追加している。

pluginはテキスト形式で以下の値を返すことで実現できる。

- 観測対象のラベル名および値を返す。
- グラフ作成時に適切なグラフを描画するための設定情報を config モードで返す。

テキスト形式での応答を行うだけであるため，シェルスクリプトやPerl, Ruby といったスクリプト言語がよく利用される。簡単な plugin の例として文献[5]に紹介されているLinuxマシン上のロードアベレージを観測するための plugin スクリプトを以下に示す。

* 鳴門教育大学 大学院 自然・生活系教育部

```
#!/bin/sh

case $1 in
  config)
      cat <<'EOM'
graph_title Load average
graph_vlabel load
load.label load
EOM
      exit 0;;
esac

echo -n "load.value "
cut -d' ' -f2  /proc/loadavg
```

このようにMuniでは簡単なスクリプトをpluginとして用意することで観測対象を追加できる。

2.3. pluginの自作

ソーラーパネル発電量

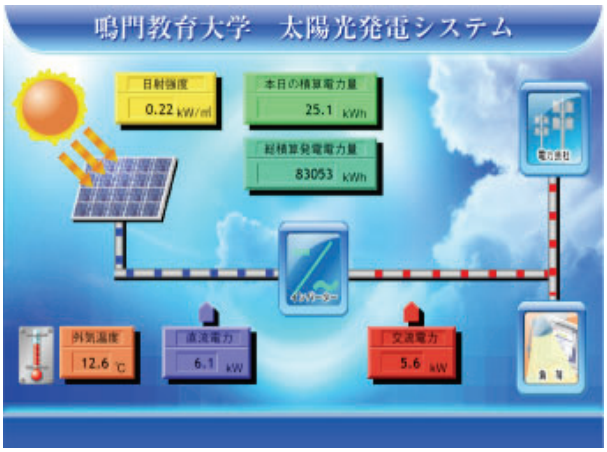

図1 太陽光発電システムWebページ

本学の本部棟屋上にはソーラーパネルが設置され太陽光発電を行っている。発電の状況は本部棟に設置されたパネルに表示されている他，Webページでも見ることができる。Webページは Flash で構成されており，日射強度，直流電力，交流電力，気温，本日の積算電力量，総積算発電電力量のパラメータを表示する。Webページのソースを確認するとそれぞれのパラメータはソース中の parm 要素，属性 name=FlashVars で指定される属性 value にパラメータ毎の値が格納されている。plugin スクリプトは Ruby で記述し，Webページスクレイピングライブラリ Mechanize を利用し HTML ソースから各パラメータの値を抜き出すこととした。

```
agent = WWW::Mechanize.new

agent.get('http://www.foo.bar/')

values = Hash.new()

agent.page.search('param').each{|x|
  if x.get_attribute("name") == "FlashVars" then
    x.get_attribute("value").split('&').each{|v|
      (key,value) = v.split('=')
      values[key] = value
    }
  end
}
if ( $0 =~ /_power$/) then
print "dc.value ", values['dc_power'], "¥n"
print "ac.value ", values['ac_power'], "¥n"
print "nisha.value ", values['nisha'], "¥n"
else
print "kion.value ", values['kion'], "¥n"
end
```

このpluginでは直流・交流電力および日射強度と気温のグラフを分けるためpluginの呼び出し名により応答する値を変えている。積算値はグラフで利用しないため無視している。pluginにより気温をグラフ化したものを図２，発電電力(交流,直流)および日射強度をグラフ化したものを図３に示す。

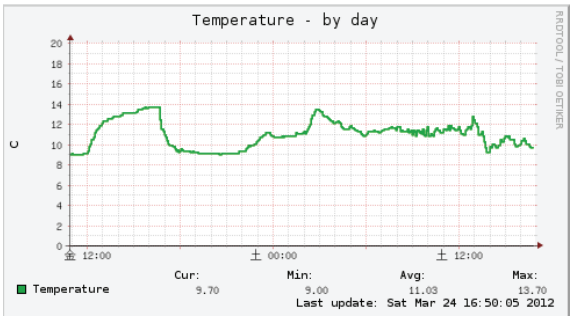

図２ 気温

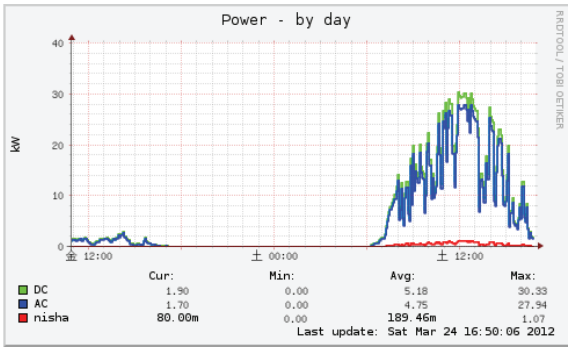

図３ 太陽光発電電力および日射強度

無線LAN利用状況

情報基盤センターで提供している無線LANはWeb認証システム FEREC によりユーザー認証を行っている。FEREC は NAPT モードで運用しているため無線 LAN セグメントに対する DHCP 割り当てアドレスの上限は機器の制限により 250 になっている。また無線 LAN の利用者は増加傾向にあるため，実際にどの程度の利用があるのか確認するために plugin を作成した。

FEREC 管理ページ

ログイン中のユーザ数	23
接続方式	NAPT

図4 FERECログインユーザ情報

図5 FEREC DHCP情報

FERECはWeb管理コンソールにアクセスすることでDHCP の割り当て済みアドレス（図 4），認証済みユーザー数（図 5）の確認ができる。plugin は Ruby で記述しており，Mechanize を利用して値を読み取る。また実際に無線 LAN セグメントを利用中の端末台数を調べるため，L3 スイッチに telnet 接続し無線 LAN セグメントに対応する VLAN の MAC アドレスをカウントしている。Cisco 製のスイッチであれば SNMP 経由で VLAN に含まれる MAC アドレス数が求められるようだが，本学で利用している HP 製のスイッチでは SNMP 経由では求められない。そのため telnet 接続し，show mac-address vlan コマンドを実行した結果から MAC アドレス数を求めている。plugin スクリプトの主要部分を以下に示す。

```
agent = WWW::Mechanize.new
agent.auth(account,password)

agent.get('https://ferec.foo.bar/top.gsp')

users = ¥
agent.page.search("table[@class='config']/td")[1].in
    ner_text
users.chomp!

agent.get('https://ferec.foo.bar/dhcp_list.gsp')
leased = ¥
(agent.page.search("table[@class='list']/tr").size-1
    )
ans = Array.new()

telnet = Net::Telnet.new("Host" => swip)
telnet.waitfor(/Username: /)
telnet.cmd("String" => account, "Match" => /Password:/)
telnet.cmd("String" => password, "Match" => /Prompt/)
telnet.cmd("String" => "show mac-address vlan xxx",
    "Match" => /Prompt/){|l|
  telnet.puts("¥n") if l =~ /-- MORE --/
  l.split(/¥n/).each{|x|
    ans.push(x) if x =~ /[0-9a-f]{6}¥-[0-9a-f]{6}/
  }
}
telnet.cmd("String" => "logout", "Match" =>/Do you want
    to log out/ )
telnet.cmd("String" => "y")

return ans.size
```

図6に得られたグラフを示す。このグラフよりリース数と MAC アドレス数の乖離が大きいことがわかる。つまり DHCP でアドレスを取得した後，リース終了以前に無線 LAN の利用を終えている端末が多く，割り当てを行っているものの利用されていないアドレスが多くあり，アドレスの割り当てが無駄になっている。これらの

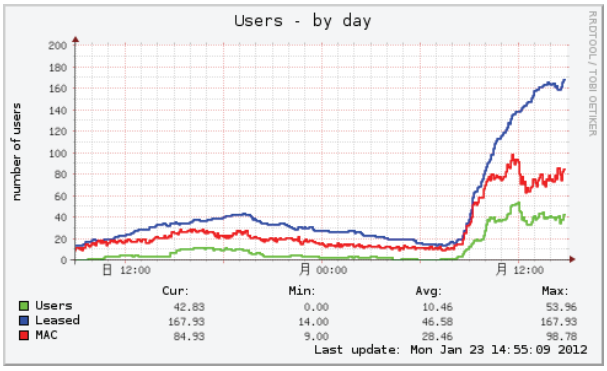

図 6 FEREC 利用状況

情報から，無線 LAN セグメントにおけるリース時間が長すぎると判断し，リース時間を 240 分から 40 分に短縮した。リース時間短縮後の利用状況を図 7 に示す。以前よりリース数と MAC アドレス数の乖離が減少しており，適切なアドレス配布が実現できている。

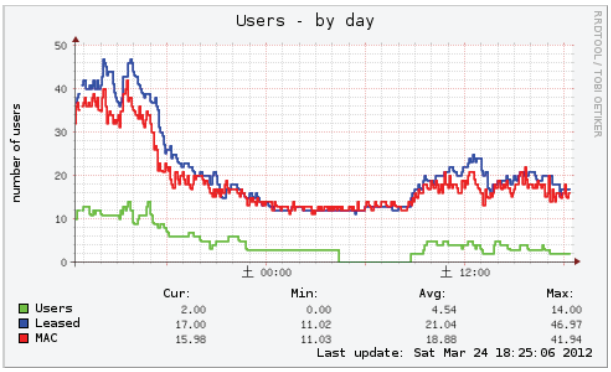

図7 FEREC 利用状況（リース時間短縮後）

学内ネットワークの利用情報

本学のネットワークはコアスイッチ HP 8212zl を中心とし，各建物のエッジスイッチと接続する形で構成されている．コアスイッチの情報は SNMP 経由で取得できる．SNMP ではさまざまな情報を得ることができるが，現在観測しているのはコアスイッチの CPU 負荷およびいくつかのサブネットにおける MAC アドレスの登録数である．MAC 数を観測しているのは端末室や教室，学生寮のサブネットであり，これらのネットワークの利用状況を把握することができる．

HP 社製のスイッチにおける MIB で CPU 負荷を得るには(OID:1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0)を利用する．プラグインでは SNMP 経由で上記の OID の値を取得する．MAC の数はスイッチの arp テーブルの情報を返す ipNetToMediaPhysAddress (OID:1.3.6.1.2.1.4.22.1.2) を利用し，スイッチの arp 情報から観測対象のネットワークに属する IP を調べカウントする。

```
Room = { 'subnet' => "description" }
SNMP::Manager.open(:Host => 'switch ip'){ |manager|
    manager.walk(["ipNetToMediaPhysAddress"]){ |row|
       row.each{ |vb|
              ip                                    =
    vb.name.index("1.3.6.1.2.1.4.22.1.2").to_s.split
    ('.')[1..4].j
')
```

```
            mac = vb.value.unpack('H12')
            next if ( mac == ["000000000000"] )
            Room.each{|net,name|
                        numofpc[name]                =
    numofpc[name] + 1 if ( ip =~ /^#{net}/ )
            }
        }
    }
}

Room.values.uniq.sort.each{|num|
      print num,".value ",numofpc[num],"¥n"
}
```

plugin により得られたグラフを図 8 に示す。図 8 のグラフから各端末室や学生宿舎ネットワークの利用状況を把握することができる。

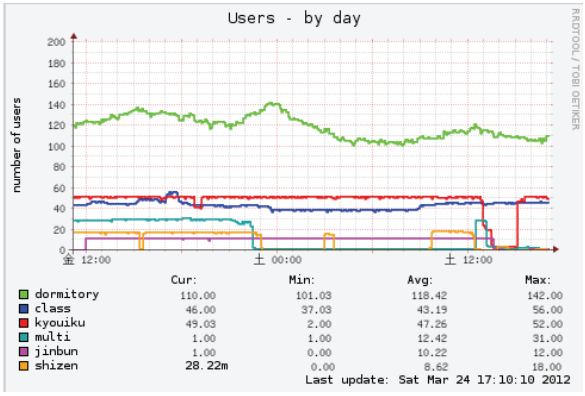

図 8 ネットワーク毎の MAC アドレス登録数

ブレードサーバの温度センサー情報

サーバ室は一般に強力な空調により温度を下げて ICT 機器を冷やしている。必要以上に冷やしすぎず適切な室温に設定できれば省電力化を図れるが, そのためにはサーバなどの ICT 機器の環境温度を監視する必要がある。サーバの多くはシステムを遠隔管理，監視するためのリモートコントロール機能をもっており，それを利用することでシステムの温度情報などを得ることができる。この情報を利用し，温度を監視することでサーバ室の温度最適化を試みた。現システムのエンジニアを介してサーバの推奨温度情報を確認したが，多くのサーバ室は 25 度の設定で運用しているという情報が得られた。基盤センターも 25 度に空調を設定したが，局所的に高温になる場所が存在しないか温度情報を監視することにした。

情報基盤センターで多く利用している HP 社製のサーバでは ilo2 と呼ばれるリモートコントロール機能が用意されている。通常は Onboard Administrator と呼ばれる Web インタフェースを介してこれらの情報を閲覧するが, munin の plugin として利用するには JavaScript が多用されており，スクレイピングで必要な情報を取得することは困難である。そこで ilo2 に対する他のアクセス手段を調べたところ，XML を利用した情報の取得が可能でありサンプルとして Perl を用いた例が示されていることがわかった[6]。XML にて query を送ると対応する情報が XML で得られる。温度などの情報を得るには

```
<?xml version="1.0"?>
<RIBCL VERSION="2.21">
  <LOGIN USER_LOGIN="#{user}" PASSWORD="#{pw}">
    <SERVER_INFO MODE="read">
       <GET_EMBEDDED_HEALTH />
    </SERVER_INFO>
  </LOGIN>
</RIBCL>
```

を送る。応答される XML は以下のようになる。

```
<GET_EMBEDDED_HEALTH_DATA>
<FANS>
  ...
</FANS>
<TEMPERATURE>
  <TEMP>
    <LABEL VALUE = "Temp 1"/>
    <LOCATION VALUE = "Ambient"/>
    <STATUS VALUE = "Ok"/>
    <CURRENTREADING VALUE = "24" UNIT="Celsius"/>
    <CAUTION VALUE = "42" UNIT="Celsius"/>
    <CRITICAL VALUE = "46" UNIT="Celsius"/>
  </TEMP>
 ...
</TEMPERATURE>
 ...
</GET_EMBEDDED_HEALTH_DATA>
```

plugin では XML を解析し munin で監視すべき値を取り出す。plugin は Ruby で記述し XML の解析には rexml/document を利用した。

```
doc = REXML::Document.new answer
doc.elements.each("*/TEMPERATURE/TEMP/") {|x|
 label = ""
 temp = ""
 x.each_element("LABEL"){|e|
  label = e.attributes.values.to_s.sub!(/¥s/,"_")
 }
 x.each_element("CURRENTREADING"){|e|      temp      =
    e.attributes["VALUE"].to_s}
 printf("%s.value %s¥n", label, temp)
}
```

plugin により得られたグラフを図 9 に示す。このグラフより，空調温度を 25 度に設定しても主要機器は十分に冷却されておりまだ動作環境的には余裕があることが分かる。

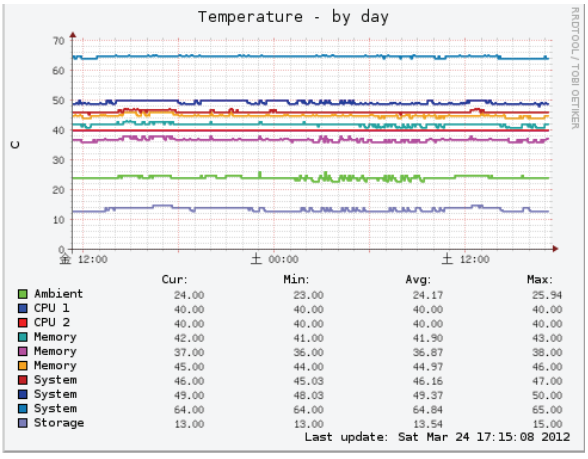

図 9 ブレードサーバの温度センサー情報

ターミナルサーバ情報

図書館に設置している SunRay 端末の接続先であるターミナルサーバ(TS:Windows Server 2008)の利用者数を監視するため，munin-node をインストールし TS の利用者数を調べる plugin を追加した。TS の利用者はコマンドプロンプトにて”query session”を実行し，状態が

Activeなセッション数をカウントすることで確かめられる（図10)。カウントを行う処理をvbsスクリプトとして作成したがそのままでは権限の問題で実行できなかった。そこでバッチファイルからvbsスクリプトを起動することとし，munin.iniのExternalPluginにバッチファイルを追加した。

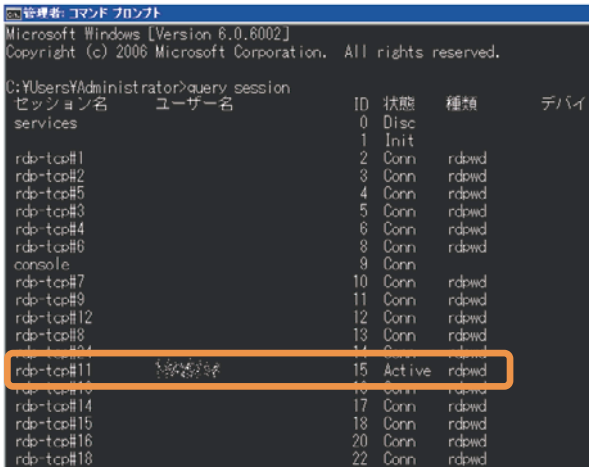

図10 queryコマンド

plugin により得られたグラフを図 11 に示す。横軸は1か月を示しているがこの月には最大14人の同時利用が行われたことが分かる。

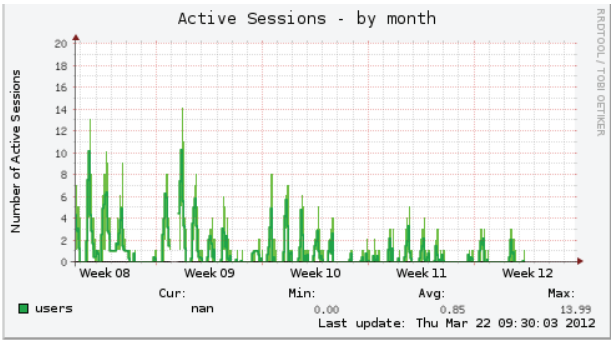

図11 ターミナルサーバ利用状況

温度測定

高温になりやすい設置場所や端末室の室温を観測するためにセンサーを設置し，温度変化をグラフ化している。

学生寮 LAN を構成する機材の一部は西日が扉を直射する分電盤に設置しており夏場の環境温度をモニターする必要がある。センターから離れた場所に設置することもあり，既製品ネットワーク対応の温度ロガー”おんどとり TR-7W(ティアンドデイ社製)”を導入し観測を行っている。TR-7W は Web サーバを内蔵しており，ブラウザでアクセスすれば温度，湿度の現在値モニターが表示される。plugin は応答される html ファイルを Ruby スクリプトを使って処理し，正規表現を使って温度センサーの情報を取り出している。以下に plugin の主要部分を示す。

```
response = http.get('/B/crrntdata/cdata.txt')
temp1 = $1 if ( response.body =~ ¥
    /Temperature1=(¥d+¥.¥d)/ )
temp2 = $1 if ( response.body =~ ¥
    /Temperature2=(¥d+¥.¥d)/ )
```

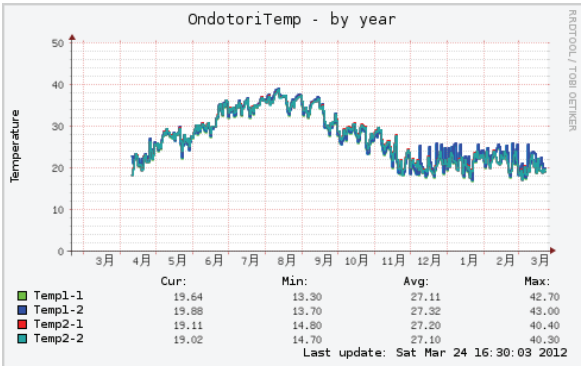

図12 おんどとり観測データ

plugin により得られたグラフを図 12 に示す。このグラフは横軸が 1 年になっている。これから夏場には設置場所の温度が 40 度を超えており，機器には過酷な環境であることが分かる。

ネットワーク接続型温度計の試作

マルチメディア教育実習室は講義棟最上階という部屋の構造上，室温が上がりやすい。施設課に依頼し能力の高いエアコンを設置しているが夏場は暑すぎるという指摘が多々ある。夏場の環境改善のためにも実際に室温が何度になっているのかを測定することにした。前述の TR-7W を利用すれば測定は可能であるが，機材の導入コストや温度，湿度以外のデータを測定する場合も考え，NXP 社製マイコン mbed に温度センサーLM75 を接続したネットワーク対応の温度センサーを自作し，それを用いて観測を行った。mbed は 6 チャンネルのアナログ入力があり，1 チャンネルに温度センサーLM75 を接続し入力電圧を読み取って温度に変換した。また TCP/IP を扱うためのライブラリも用意されているため，簡単にネットワーク接続型の観測装置を自作することができる。

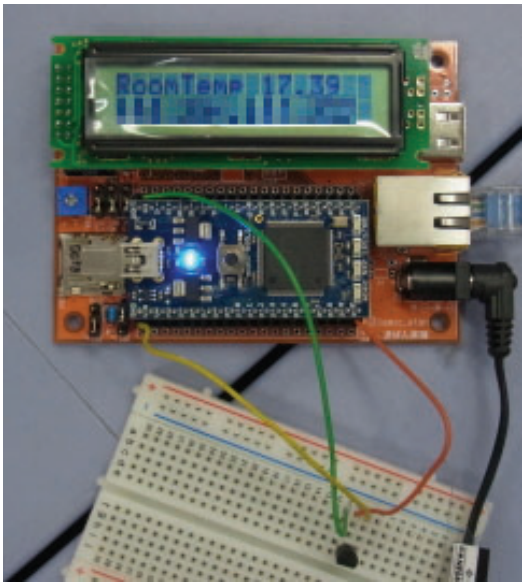

図13 mbedを利用したネットワーク対応温度計

mbed は観測を続けているとハングアップする場合があったため，ライブラリにあった watchdog 機能を追加することでハングアップ後に自動リセットを実行している。これにより安定してデータの観測ができるようになった。

mbed で動かすサーバプログラムは TCP12345 番ポートで接続を待ち，クライアントが接続すると温度情報を返す。plugin では mbed の TCP12345 番ポートに接続して温度を得る。plugin により描かれたグラフを図 14 に示す。

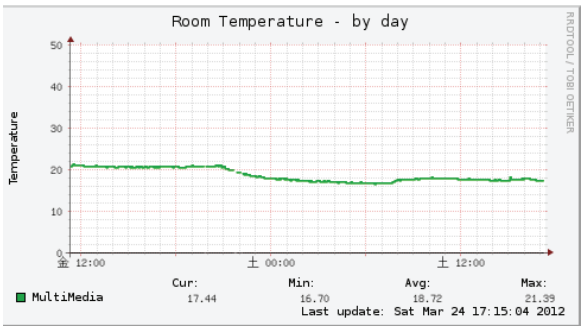

図14 mbed観測データ

3. まとめ

サーバやネットワーク機器，各種センサーから得られる情報を Munin により蓄積しグラフ化することで時系列に沿った変化を直観的に把握できるようになる。得られたグラフを見て状況を判断することで，無線 LAN セグメントの設定パラメータやサーバ室温度設定変更による省電力化などの改善を行うことができた。センターの持つ機材が提供する情報は様々なものがあり，まだ十分活用できてないものもある。今後は Munin の plugin を増やすなどより多くの情報を収集し，グラフ化して監視し，異常の発見や予防に活用したい。

参考文献

[1] Munin: http://munin-monitoring.org/
[2] RRDtool: http://oss.oetiker.ch/rrdtool/
[3] MRTG: http://oss.oetiker.ch/mrtg/
[4] Cacti: http://www.cacti.net/
[5] How to write Munin plugins: http://munin-monitoring.org/wiki/HowToWritePlugins
[6] HP Integrated Lights-Out Management Processor Scripting and Command Line Resource Guide: http://h10032.www1.hp.com/ctg/Manual/c02237707.pdf
[7] mbed を始めましょう！: http://mbed.org/users/nxpfan/notebook/lets_get_started_jp/