CUC 2002 Talk

Title: Process monitoring and control using Internet and cellular telephony

Alternative title: A system for remote process control using Internet and mobile telephony

Authors: dr. Mario Stipčević, Ruđer Bošković Institute, Zagreb; Krešimir Jakovčić, Ruđer Bošković Institute, Zagreb

Submission date: 01. September 2002.

Abstract

We present a system designed for general control and monitoring of processes over the Internet and optionally the cellular telephony. It was specifically used to control and monitor a scientific experiment and was also used to monitor temperature and humidity of a computer cluster installation. The system consists of both hardware and software parts. It also enables to preset any number of user-configurable alarms that would be signaled to user(s) through the Internet, e-mail or cellular telephones when a specific condition(s) is met.

Introduction

Most systems for controlling scientific experiments (processes), from simple ones such as running a simple didactic demonstration to most complicated ones, such as data taking and slow-control of a large high-energy particle physics experiment at CERN, are operated by programs that run on local computers which are physically linked to the controlling hardware. The hardware may be anything from specialized non-standard units to industry-standard modular CAMAC or NIM crate building blocks. In most cases the software is a proprietary software matching the respective hardware and a little can be done to extract the data or insert a parallel operating routines without an effort of writing ones own software basically from the scratch. The tendency for instrumentation in certain branches of scientific research instrumentation (notably anything that starts with "bio-" or "medi-") is that an instrument even does not need a separate PC or workstation but is all-integrated in one rugged casing with a sufficient displaying and printing capabilities to operate its specialized purpose in an ignorant-proof way. Unfortunately, such instrumentation is often completely LAN and Internet unaware and consequently it may be very hard if not impossible to access the data from it or to control it from a distant computer or a steering program that for example would be used to master a complex measurement/control process.

Expensive industrial solutions (often crate based) possess enough flexibility to be operated from a computer network, but even so, most applications supplied with them do not offer this possibility directly, that is user needs to do a substantial amount of programming by himself in order to achieve this goal.

The problem

Our problem (in the year 2000.) was to perform a scientific experiment which was observing rare events. To obtain enough statistics, the experiment must be ran for some ~100 days, preferably in continuo. To complete the measurement in a reasonable time, we preferred that the experiment be run without interruption, 24 hours a day 7 days per week. This was a problem because only two men were designated to perform the experiment and the two were too few to organize shifts.

Experimental apparatus (FIG 1.) includes: a high-vacuum chamber equipped with a water cooler and an electric heater, two vacuum pumps, laser, cooled photon counter, two vacuum gauges covering the full range of pressures from 1 bar down to 10^{-12} bar, and two multi-layer refractors separated with a gas-conductive light trap. Some 20 variables had to be monitored (i.e. sampled and stored) periodically (with period of ~60 sec) as they were all important for deriving the scientific result. On top of that, several parameters of the experiment needed to be controlled: pump heating and cooling, vacuum chamber heating and cooling, PM cooling, laser power, switching pumps on and off, resetting the photon counter etc.

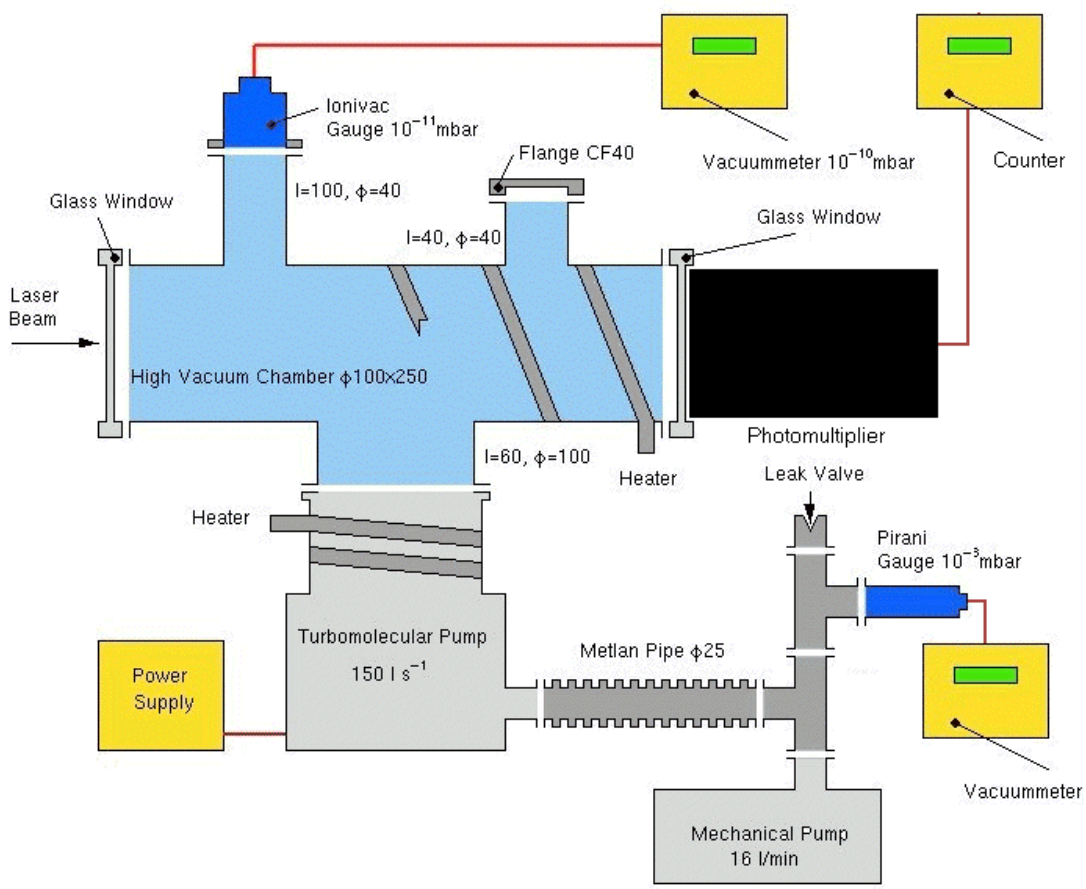

FIG 1. The scientific experiment at the Ruđer Bošković Institute

Fortunately, it was possible to conceive the experimental setup in such a way that only rare human interventions (such as mounting or dismounting parts of the detector,

cleaning of the high vacuum parts etc.) were needed. Actually, once the experimental apparatus was built and properly sealed, it was able to operate for couple of months without any need to make mechanical interventions.

Hardware used

Having only a relatively small budget at our disposal we could not afford to buy existing industrial blocks to build the experimental setup. We also felt that the existing building blocks were not ideally suited for our purpose. Therefore, we have decided to design and build our own hardware solutions. This hardware was supposed to be operable from Linux and well suited for control over the LAN.

In order to achieve that, we have built the following hardware:

- Master input-output unit (SIO-box) with 56 input bits and 32 output bits, operable from Linux through the parallel PC port
- Sensor box with 6 universal sensor channels (for temperature sensors etc.)
- 13 channel ADC, each channel independently settable in 4 sensitivity ranges
- 8 decade counter (CMOS I/O)
- 220 VAC Z-box with 2 controlled on/off switches, 2 adjustable power lines each adjustable in 15 hardware linear steps from 0 to the full power (extendable up to virtually infinite number of steps by a software).

SIO-box is an interface between the PC parallel port and all other electronics devices.

Sensor box was mainly used to detect temperatures.

ADC is an analog-to-digital converter needed to convert voltages, for example from the sensor box.

Counter was used to count photons.

Z-box, in its essence, is a type of bistable switch that upon a single failure of the 220 VAC power network disconnects from the power line the whole experiment, including itself. In this way, everything stays shut off even after the power resumes. This is important, because switching the experiment ON is a complicated procedure. Normally, after the power resumes, Z-box must be reset by hand. To automate this process, our Z-box has a dedicated line which enables it to be securely reset by the PC. Our Z-box also incorporates other computer controllable switches and stepped power regulators.

Solution to the problem using the Internet

Quite generally, to control an experiment (or any process) it is necessary to write a series of routines (so called primitives) for performing simple operations or communicate with individual hardware like: switching ON and OFF a pump, bringing a heater to a specified temperature, reading voltages from the ADC etc., and a steering program that makes all these to work as a whole. A steering program is an interface

that makes possible for the scientist (the user) to have control over the experiment (or processes). It normally takes advantage of graphics interface to display status, data etc. and to receive commands from the user through buttons (mouse) and keyboard input fields. Steering program isn't, however, meant for complex data analysis.

Having made our own hardware, we had no choice but to write our own primitives and the steering program. Writing the primitives was a rather straightforward task. But writing the steering program was an important milestone. Namely, rather than writing a "normal" program and using some standard graphical library (such as TclTk or Xlib) we have decided to make the steering program as a set of CGI executables with HTML forms for the graphics part.

When working from the local computer (the one that is actually connected to the hardware) it doesn't matter whether the steering program is a stand-alone program or set of CGI routines with HTML interface. BUT the important difference is, of course, in the fact that the program in CGI technology can be run all the same from any distant computer connected to the local computer network, as if it would run on the local computer. This also implies that the local computer must run a WEB server.

Since the local network of our Institute is connected to the Internet it is possible to access the experiment from the Internet as well.

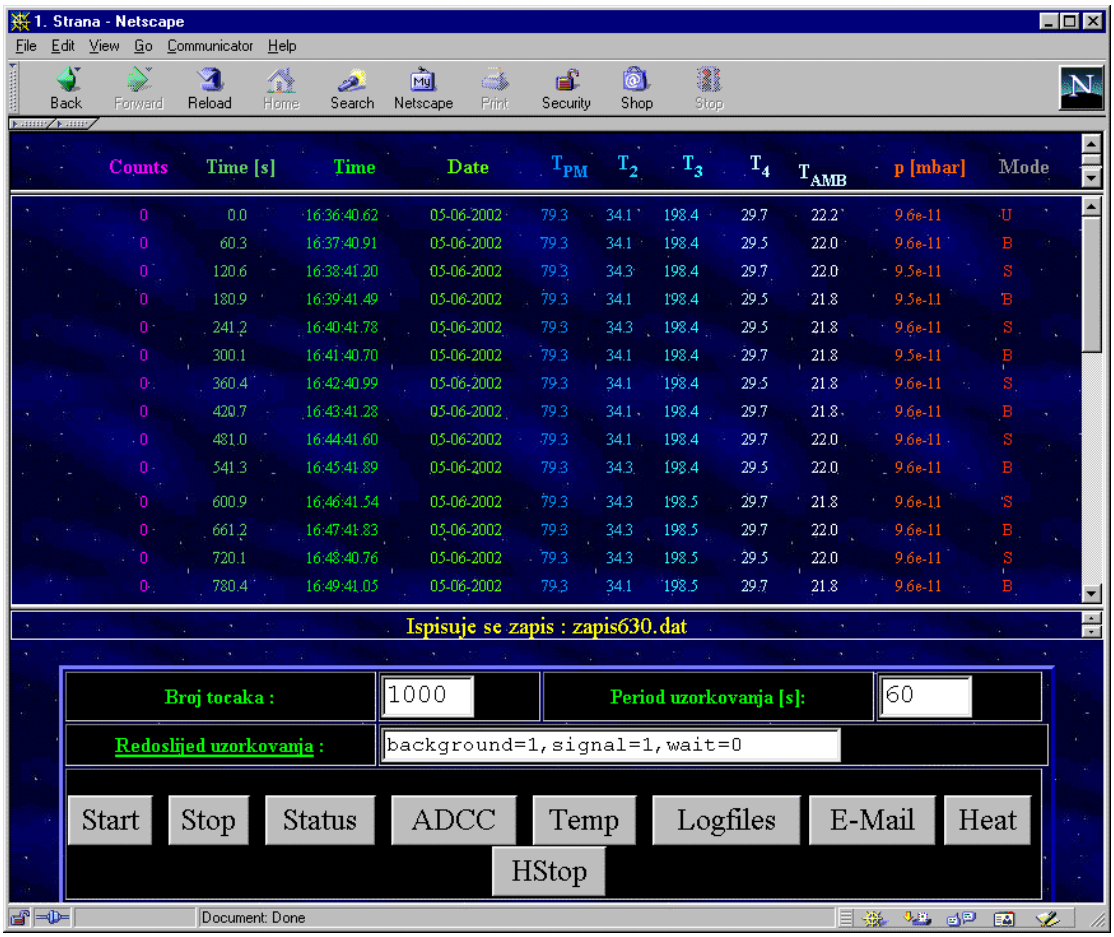

FIG 2. The WEB form + CGI based steering program

Additionally, a very nice feature of our approach is that programming in the CGI technology is even easier than working with any standard graphics tool available for the X-window system !

The appearance of the main page of the steering program is shown on the FIG. 2.

Lastly, we have connected a web-camera so that we could actually see the experiment in real time. In our case it was just for fun because all relevant data could be accessed over the Web pages but it was nice to be able to actually see for example the counter display and check that it really matches the number reported on the steering Web page.

Solution to the problem using cellular telephony and the Internet

The possibility to run the experiment, and monitor it over the Internet was really thrilling. However, a painful revelation came when the summer came and we decided to take our regular vacations. We realized that even if one has a notebook PC (that we did not have at the time) as far as the Internet is concerned, Croatia is like a desert. As soon as you are away from your home or institute/faculty or possibility to connect to a fixed telephone line, you have no place to connect: post offices do not offer Internet or phone connection, internet cafe-shops are rare as oasis. Actually, there are 2-3 in Zagreb and probably none in the rest of the state. Internet kiosks or stands are very rare and in most cases out of service. Buying a notebook PC just for the purpose was out of the question, even without mentioning that the price of 1kByte over such connection was 100 times more expensive and 6 times slower that a connection over the fixed telephone line. However, we had cellular (GSM) phone(s) at our disposal.

So, what could we do to continue running our important experiment during the summer ? We could not afford to stop it for a whole month. We could not afford not to take the vacations either. Therefore we came up with an original solution: let's do everything over a cellular GSM phone !

Having very little time for realization of this project (only two weeks), we have decided to rely as much as possible on the programs made already for the Internet. To this end we have invented the "SMS Robot".

SMS Robot is a virtual being (a program) that has an account on the local computer. It also has an e-mail account and is able to receive, read and analyze its mail. What is important is that SMS Robot does not understand the contents of the mail that concerns the scientific experiment, but only its format. This is the reason why it was so easy to make the Robot it in such a short notice. This is also the beauty of it because it is something universal and can be used in any other application other than steering our experiment.

Robot parses the contents of incoming mails and sends the result to the steering program. For this to work we needed to convert the steering program into a daemon.

By our definition a Robot must have the following capabilities:

1. Recognize the message format and parses it. The parsed result is then conveyed to another daemon program (e.g. a steering program) that is not a part of the Robot. Robot does not need to understand these commands;
2. There must be a set of commands to configure the Robot itself. Robot must understand this commands;
3. Robot must understand who has sent a message and be able to reply to the user as necessary;
4. It must be possible to preset a set of "alarms" that the Robot can receive from the daemon and would send to specified users (two-way Robot).

How does the SMS Robot operate ? First, it is important that the cellular phone in question may be able to send normal e-mails. Many GSM operators offer that possibility (eg. CRONET). The Robot simply reads instantly any incoming mail messages and parses them according to the following format:

command [parameter list] [authentication_response]

where parameter list is a space-separated list of tokens of the form:

parameter=value

and authentication_response is used in the enhanced security mode (see below).

Here is a non-comprehensive list of commands that we used to operate our scientific experiment:

- **start [parameter_list]**

Example:

start np=2000,dt=60

would start the data acquisition (DAQ) program, that would sample all relevant parameters 2001 times at regular intervals of 60 seconds.

- **stop**

Stop the DAQ.

- **report**

Request for a short report concerning present activity of the experiment. Example of the answer by the Robot:

```
Report 15:08 02-May (s=10,np=20000,dt=30) file=494,
pt=5888, mode=SH,Ta=28.2,Tpm=28.5,TH0=61.2,TH1=30.3
p=2.47E-07mb, since 14:04 30-Apr
```

- **status**

Request for a longer status report concerning present activity of the experiment. This report is generally more than one SMS long.

- **file [number]**

Request for a summary of the measurements (and preliminary results) contained in tfe file identified by the "number".

- **sreq** ***command***

A command to pass low level commands directly to the steering daemon. Whenever there is a new low-level command added to the steering program, it can be addressed this way without any changes to the Robot. Example:

sreq heater0 82 hold0 300

means: bring heater number 0 to 82 degrees Celsius and hold at that temperature for the next 300 minutes and then shut off the heater. Any numbner of commands may follow the "sreq" keyword.

- **email set|add|erase|status e-adresa [h1:m1 [h2:m2 ...]...]**

Using this command each user can separately preset its own table of hours when he/she wants te receive reports about the experiment. First parameter determines whether: times are to be set, added to the existing table, erase all times or to get a list of actual times.

Security issues

Up till now we did not comment security of the system(s) for remote process control using Internet and mobile telephony. Since the system was conceptually new and unpublished at the time, we did not have any fears that it may be used by somebody else. Nevertheless, we have developed a full-fledged security system.

For the Internet part, the WEB pages are secured by password and are available only through the https protocol, so that the password may not be observed by sniffers.

For the e-mail/SMS part two layers of security were installed.

The trivial one is that SMS Robot could have been configured to answer only to specific incoming e-mail addresses. As is well known, e-mail addresses could be relatively easy faked and therefore this is not a great security shield, but it offers some level of protection.

The strong one is using a challenge-response type of authentication and is based on a specially designed cryptographic protocol. A challenge-response type of authentication is used in case of unsecured connection, such as in this case. Normally, the server and the client(s) agree previously on a secret password(s), and the authentication is made in the following way. The server sends a one-time, publicly

known hash function upon what the client responds by hashing the password. The server then does the same and compares hashed values: if they match then the client is authenticated. In this scheme it is important to use every time a new hash function. In our case there was no shared secret password but rather the hashing function was a secret. The server would simply send a random number and the client would send a hashed number to prove that he knows the hash function. Since both challenge and response look like random numbers and there is no obvious correlation between them, an eavesdropper cannot authenticate himself even if the communication line is unsecured. It is, however, important not to repeat any of the challenges. The authentication scheme is shown in FIG 3. This second layer was not mandatory. It was possible to switch OFF the authentication, but to do that one needed to authenticate himself. On the other hand, anybody could turn the authentication ON.

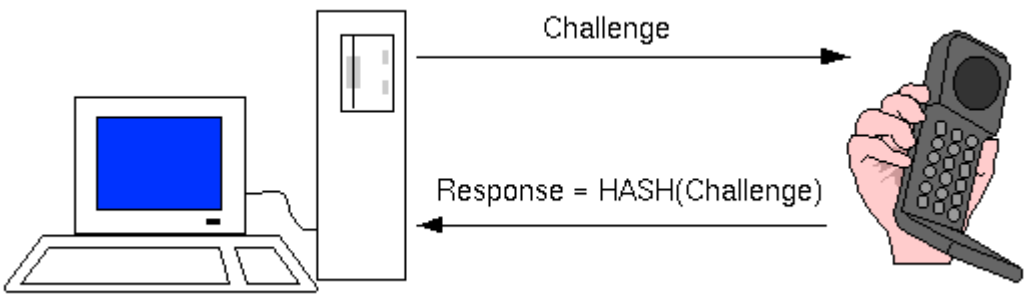

FIG 3. Authentication scheme for the SMS Robot

The formula used was simple enough so that all operations could be performed "in the head" even on the sea shore at temperatures of 40 degrees in the shade ! This was an important issue because it was absolutely necessary to be able to operate the experiment during the summer vacations even if it would be from the beach.

Other uses of the SMS Robots

Currently, there is another copy of the Robot running at the Computing Center of the Ruđer Bošković Institute. We have installed the same robot and a part of our original electronics to monitor temperature, pressure and humidity in the super-computer room. The measuring system may be monitored over a secured WEB page and GSM cellular phones. Computer operators are warned by SMS messages if any level of alert is reached. In this way it is not necessary to actively monitor the system. All the hardware, Internet and SMS services are run on a single Pentium 200 MMX computer, running Linux.

On top of that, operators have made possible to stop the jobs, shut off parts or the whole computer cluster from the Internet. Since a new air-conditioning system has been introduced it had several severe failures and the SMS Robot system actually played crucial role in saving computers from overheating at several occasions.

As a third example, an experimental Robot is used to access the directory of the Ruđer Bošković Institute.

Summary

A scientific experiment has been built at the Ruđer Bošković Institute. A specific electronics hardware was built to make possible the data acquisition, monitoring and control of the experiment. A set of small routines (primitives) that communicate with the hardware was written. To perform the experiment in a completely automated way and make possible to control the experiment from the Internet, a steering program was written using the CGI technology and a WEB user interface. An SMS Robot was built around the steering program in order to make possible to control the experiment over the GSM cellular (mobile) phones.

Both the steering program and the SMS Robot are very efficient programs written in C and compiled with gcc compiler. They both ran simultaneously on a simple 486 DX/2 computer running Linux.

List of figures