ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/TC–99/22
19 Ottobre 1999

IPACCOUNTING
Monitoring and accounting TCP/IP traffic through a Cisco router on a Unix system
Version 3.0

Claudio Strizzolo[1]

INFN–Sezione di Trieste, Via Valerio 2, I–34127 Trieste, Italy

Abstract

IPaccounting makes you able to monitor and account TCP/IP traffic on a Cisco router, which connects an "accounting domain" to the rest of the world. An "accounting domain" is a set of networks whose traffic to the "world" is monitored by IPaccounting.

Version 3.0 is the first "compiled" release of IPaccounting, running dramatically faster than previous ones. It includes also some improvements based on the feedback by users of the previous releases.

This document includes the installation, the administrator and the user's guides.

Published by ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

[1] E–mail: Claudio.Strizzolo@ts.infn.it

IPaccounting - Version 3.0

This software has been developed at Istituto Nazionale di Fisica Nucleare (I.N.F.N.) - Sezione di Trieste, by Claudio Strizzolo, on April, 1999

This software is distributed “as is”, and without any implicit nor explicit warranties. No responsibility is assumed for any bugs, features, wanted or unwanted side effects.

Trademarks included in this document are acknowledged as such.

Part I
Introduction

1 Product description

IPaccounting makes you able to monitor and account TCP/IP traffic on a Cisco router which connects an accounting domain to the rest of the world.

In this document, an *accounting domain* is defined as a set of networks whose traffic from/to the *world*, consisting of everything outside the accounting domain itself, is monitored by IPaccounting. An accounting domain usually covers everything "on this side" of the router: it might be a single network, or even a more complex entity, such as a whole Autonomous System.

The term *accounting subdomain* refers to a subset of this domain, usually covering one or more networks belonging to it.

Figure 1 shows the relationship between an accounting domain and its subdomains.

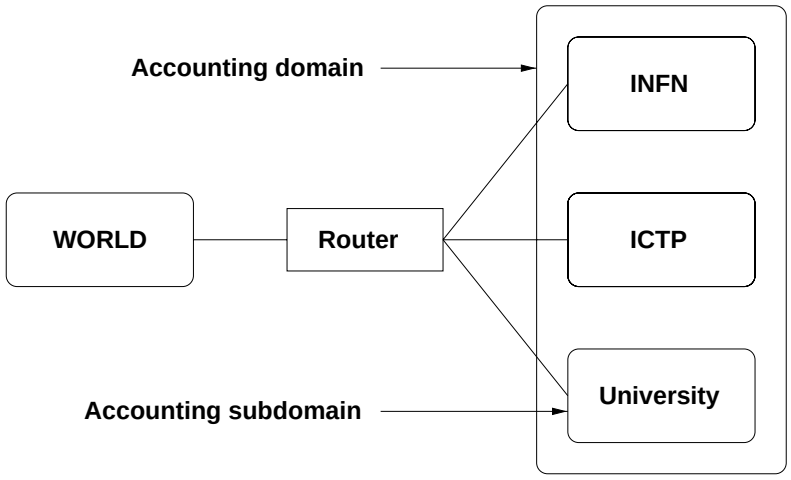

Figure 1: Accounting domain and subdomains.

IPaccounting builds daily lists of top communicators, and comparative tables and

graphics for daily or monthly traffic analysis, either about the whole accounting domain and its subdomains.

The results are presented in HTML format, and are therefore accessible through any Web browser. The graphics built by IPaccounting can be displayed in PostScript or GIF format.

IPaccounting can be used to monitor several routers at once, by producing separate or aggregate statistics. It is customizable, in order to satisfy a wide range of requirements.

2 Structure of this document

This document is composed of three parts.

The first part, that you are currently reading, includes general information on the package, and the main changes in IPaccounting 3.0, in comparison with the previous releases.

The second part is the guide for the IPaccounting Administrator, that is the person in charge for installing, configuring and managing the software described in this document on your site.

The third part is the User's Guide, describing access to the data and the statistics collected by IPaccounting through a Web browser.

3 What is new

Version 3.0 is the first "compiled" release of IPaccounting. The package has been completely rewritten using C language (versions since 2.0 have been all written in Perl). This required some work, in particular to organize data structures and optimize algorithms, but resulted in the following advantages:

1. Version 3.0 is dramatically faster than previous releases, both in the statistics computing phase, and in the tables and graphics output phase. On a Digital Unix Alpha system, used as the IPaccounting server at INFN Trieste, the daily construction of statistics resulted to be *almost ten times faster* than with version 2.1.

2. On some platforms, due to a known bug in some versions of Perl, IPaccounting 2.*x* often ended prematurely because of "Out of memory" problems, while building statistics for large collections of events. The new release is not prone to this problem.

3. Version 3.0 need less external libraries to be installed.

Version 3.0 uses the new package organization that was introduced with version 2.1, which allows an easier management of the package.

It also includes a simplified installation and configuration procedure, which is particularly useful in the case you are upgrading from version 2.1.

Last but not least, the documentation has been completely reviewed and updated.

4 Terminology

The concepts of *accounting domain* and *accounting subdomain* have been described in section 1. Before going on, some more terms, which have a special meaning for IPaccounting, will be defined in this section.

4.1 Ordinary and special subdomains

The accounting subdomains are classified as either *ordinary* or *special*.

The first ones should be complementary: they should not share subnets or nodes, in order to grant consistent computing and to achieve meaningful percentages and totals. Usually, this is the most frequently needed subdomains class. The union of all defined ordinary subdomains generally covers the whole accounting domain.

Special subdomains, on the other side, do not respond to any restriction: they can be freely built according to your needs, including nodes or subnets belonging to another subdomains, either ordinary or special.

For instance, you might like to build a special subdomain including all the httpd servers belonging to your accounting domain, no matter which subdomain they belong to. This helps you putting into evidence the traffic generated by the httpd service. Another special subdomain might include only a particular node which provides a special service, i.e. news server, e-mail gateway or documents repository.

The computations based on special subdomains should not be directly compared to the ones based on ordinary subdomains, due to the total absence of rules in definitions. Therefore, they are commonly used as a tool to monitor some special traffic streams, while ordinary subdomains are used for periodic accounting.

4.2 Communication partners

A *communication partner* is a set of nodes or networks belonging to the “world” (usually, but not necessarily, outside the considered accounting domain), about which you want to monitor the traffic to/from the accounting domain. For instance, you might like to monitor

the traffic generated by the accounting domain (or its subdomains) towards the networks belonging to your country, or to the whole world.

5 More information

The IPaccounting home page is reachable at the following address:

`http://www.ts.infn.it/computing/IPaccounting/`

Part II
Administrator's guide

This part of the document describes the tasks that must be performed by the administrator of IPaccounting on your site: installation or upgrade, configuration, ordinary maintainance.

6 Pre-installation requirements

6.1 System requirements

Version 3.0 of IPaccounting is not as portable as the previous releases. The latest were written in Perl, and were therefore able to run on almost any machine where a Perl interpreter was available. The current version, instead, needs to be compiled, and the compilation process may change dramatically on different platforms.

IPaccounting 3.0 has been tested in a Digital Unix 4.0d environment, and on a Linux RedHat v.5.2 machine.

You should be able to port it to other Unix platforms, but probably you will have to change some platform-dependent items inside the Makefile in order to make it work. However, if you succeed installing IPaccounting on a different system, please notify the author: your experience might be very useful to someone else!

6.2 Software requirements

In order to install IPaccounting, you need some tools to be available on your system:

1. gcc (Gnu C) compiler.
2. As the user interface to access the statistics is entirely HTML-based, the node hosting IPaccounting must be a httpd server. The package has been tested in combination with Apache httpd server version 1.3, but other servers should work well too.
3. Tim Pearson's PGPLOT [1] graphic library is required. The package was tested with PGPLOT version 5.2.0. The PGPERL library that was required by version 2.*x* is not needed any more.

7 Software by other people

The IPaccounting kit includes some software developed by other people, that will be set up automatically during IPaccounting installation:

- Tim Behrendsen's "aa" (Associative Array) library [2].
- Eugene Eric Kim's CGIHTML library [3].

8 Getting the kits

IPaccounting 3.0: The kit is available at the following URL: `ftp://ftp.ts.infn.it/pub/unix/IPaccounting/IPaccounting.3.0.tar.gz`.

pgplot: A pointer to the kit can be found at the following URL: `http://astro.caltech.edu/~tjp/pgplot/`.

9 Installing

This chapter describes a new installation of IPaccounting 3.0. Upgrading from an older version is described in chapters 11 and 12.

9.1 Installing PGPLOT

IPaccounting needs the PGPLOT graphics library in order to build output graphics. If PGPLOT is not available on the system where you are installing IPaccounting, you must install it first.

The PGPLOT kit includes the documentation you need to perform the installation. The following notes do not replace that documentation, they just give some more hints that might be useful to you.

- The following PGPLOT drivers are *mandatory*: /GIF, /VGIF, /PS, /VPS, /CPS, /VCPS. You must enable all of them in the `drivers.list` file of PGPLOT.
- On some systems, some platform-specific special tasks must be performed, in order to make PGPLOT run. They are described in some subdirectories of the PGPLOT kit.
- Please notice that you *must* define the environment variable PGPLOT_DIR, as described in the installation document, and also install the *C binding cpg*, as they are required by IPaccounting.
- After the installation of PGPLOT, please get sure it has been installed correctly, by running the examples provided with the kit.

For any more information about PGPLOT, please refer to the documentation which is accessible in the PGPLOT home page.

9.2 Configuring the router

The Cisco router must be enabled to execute remote commands through a remote shell (*rsh*) call.

All the configuration commands must be executed from the `configure` mode of the router.

You must create an entry in the authentication database of the Cisco router, for a remote user that will be enabled to execute remote *rsh* commands on the router itself:

```
ip rcmd remote-host  local-user-name host-IP-address remote-user-name
enable
```

For instance:

```
ip rcmd remote-host root 140.105.6.158 root enable
```

This enables user *root* on host 140.105.6.158 to execute *rsh* commands on the router.
Then, enable support for *rsh* commands, with the following command:

```
ip rcmd rsh-enable
```

and enable the following

```
ip accounting output-packets
```

on all the interfaces you mean to monitor.

You are suggested to increase the default maximum number of accounting entries that will be stored in the memory of the router, until they are collected by IPaccounting. The default value is 512 entries. This number must be set in order to avoid losing meaningful data, according to the volume of traffic passing through the router and to the memory available on the router. You are suggested to assign a very large number (for instance, 10000), if the memory installed on the router allows it:

```
ip accounting-threshold 10000
```

9.3 Installing IPaccounting 3.0

1. You must expand the IPaccounting tar file in a directory which is *not* the one you mean to use as the final destination of the package (i.e. `/tmp/IPaccounting.3.0`). The tar file should contain the following files:

   ```
   README.txt
   LICENSE.txt
   INSTALL.txt
   install
   kit.tar
   ```

Read the LICENSE.txt file before going on.

Do not untar the `kit.tar` file, as it will be handled entirely by the `install` script.

2. Create the final destination directory for IPaccounting (i.e. `/usr/local/IPaccounting`):

```
# mkdir /usr/local/IPaccounting
```

   All the IPaccounting stuff (both the software and the data files) will be written below that directory.

3. Copy the `install` script distributed with the kit into the IPaccounting directory:

```
# cp -p /tmp/IPaccounting.3.0/install /usr/local/IPaccounting/
```

4. Move to the IPaccounting directory:

```
# cd /usr/local/IPaccounting/
```

5. Run the `install` script, and pass to it the location where you initially unpacked the IPaccounting tar file:

```
# ./install /tmp/IPaccounting.3.0
```

   The script will install all the required files into their final destination directory, and build the Makefile and some configuration files.

   An example follows:

```
# ./install /tmp/IPaccounting.3.0

*****************************************************************
*                                                               *
*            IPaccounting 3.0 Installation Script               *
*                                                               *
*****************************************************************

This script will help you installing IPaccounting 3.0, or
upgrading an existing v.2.1 installation to v.3.0.

Current directory does not contain any older IPaccounting
installation. We'll assume you are performing a new installation.

Do you want to proceed (Y/N)? y

Extracting the kit /tmp/IPaccounting.3.0/kit.tar in the current
directory...
(...)
Done
```

```
Creating config library src/IPconfig.h...
Done

Creating general-purpose configuration file conf/general.config...
Done

Creating src/Makefile...
Done

The installation is now complete.
Please check the files created or modified by this script, and
modify them, if needed:

   src/IPconfig.h      (you should check this file before going on)
   src/Makefile        (you should check this file before going on)
   conf/general.config (you may check this file later)

Then get inside src subdir and execute:

   # make

Then:

   # make install
```

6. You must check carefully the files `src/IPconfig.h` and `src/Makefile`.

   `src/IPconfig.h` should define only the top directory of IPaccounting. If you need to move IPaccounting to a different location, in the future, you will have to modify this file according to your new top IPaccounting directory, and compile the whole stuff again as described below.

   `src/Makefile` will be used to actually build IPaccounting. It contains some platform-specific assignments, that you must carefully check and customize, in order to make them work on your operating system. In particular, have a look at the definition of PGPLOT_FLAGS variable.

   Later on, you will need to configure `conf/general.config` as described in chapter 10.

7. Get into `src` subdirectory and build the whole stuff:

   ```
   # make
   ```

   If everything goes right, move the binaries into their final destination (that is `bin` and `cgi-bin` subdirectories of top IPaccounting directory) by typing:

   ```
   # make install
   ```

9.4 Configuring

Configuring the IPaccounting environment is a somehow complex task, therefore it is described in chapter 10 on his own. Read it very carefully and create appropriate configuration files according to your needs.

9.5 Setting up crontabs

After the package has been both installed and configured, you are ready to run.

Most of the tasks required to collect and compute IPaccounting data are usually performed on a regular schedule. Therefore, you are suggested to set up appropriate crontabs entries to make them automatic.

The number of entries you must add to the crontabs file of the user running IPaccounting (i.e. `/var/spool/cron/crontabs/root`), depends on your IPaccounting configuration. In the most common case (one monitored router), however, you just need a small set of tasks to be performed:

- Collect the data from the router. This job must be done rather frequently, in accordance with the amount of traffic passing through the router, and the maximum number of entries that can be stored by the router itself. The following crontabs line forces data collection every ten minutes:

  ```
  0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
  -c configuration-set-label > /dev/null 2>&1
  ```

  The `-c` qualifier is mandatory, in order to specify the label of the configuration set (see chapter 10) that must be taken into consideration.

  For instance:

  ```
  0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
  -c foo-gw1 > /dev/null 2>&1
  ```

- Build statistics. This task is usually performed once a day, around midnight, to compute together all the data collected during the day.

  This task may be simple or very complex, according to your accounting policy. You are recommended to read very carefully chapter 14 to understand more about this task and to be able to set up an appropriate crontabs entry, in particular if you need to monitor several routers.

  For a simple installation, with only one monitored router, an entry such as the following might do the job:

  ```
  59 23 * * * /usr/local/IPaccounting/bin/make_all_accounting
  -c configuration-set-label -n 500
  > /usr/local/IPaccounting/make_all_accounting.log 2>&1
  ```

Read chapter 14 for more information about `make_all_accounting` tool.

- Clean the directory containing temporary files. Since version 2.1 of IPaccounting, output graphics generated by cgi-bin tools, have been written on temporary files stored into `images/OUTPUTS` subdirectory of IPaccounting. An unique name generator is used to avoid them to be overwritten on any new call of the same cgi-bin tools.

  From time to time, you should run a cleaning task that removes obsolete files from `images/OUTPUTS`.

  You might also turn the `images/OUTPUTS` directory into a link to a different place (i.e. `/tmp`), provided that the new destination directory is writable by the `www` user running your httpd server. Just remove the `images/OUTPUTS` directory and do the following:

  ```
  # ln -s /tmp IPaccounting-homedir/images/OUTPUTS
  ```

  In this way, you only need to clean the `/tmp` directory from time to time (this is done as a normal routine on several sites).

The tasks to be performed in order to monitor several routers at once, are described in section 16.

10 Configuring

10.1 Configuration sets

Since version 2.1, IPaccounting has been using a new structure for configuration files. The new organization makes administration easier, in particular for complex accounting schedules.

The configuration is based on the concept of *configuration sets*. With a first approximation you might associate each *configuration set* with a router to be monitored; some exceptions to this rule will be described later on.

Each *configuration set* is composed by:

1. A configuration file named `config`[1]. Usually, this file includes a number of parameters that are specific to the configuration set the file refers to.

2. A file containing the definition of the accounting domain the router refers to, named `accounting_domain.dat`.

3. A directory containing the definitions of the ordinary subdomains, named `ordinary_subdomains`.

[1] Since version 3.0. Versions 2.*x* used to call it `configuration.pl`.

4. A directory containing the definitions of the special subdomains, named `special_subdomains`.

5. A directory containing the definitions of the communication partners, named `communication_partners`.

6. A directory where the raw data collected by the router will be saved, named `raw_data`.

7. A directory where sorted lists of traffic built by IPaccounting will be saved, named `lists`[2]

8. A directory where short daily summaries built by IPaccounting will be saved, named `reports`[3]

In the simplest and fortunately most frequent case, you will need just one configuration set, covering the whole accounting domain you mean to monitor, by collecting data from a unique router.

An arbitrary label must be assigned to every configuration set you build. Any string which can be used as a valid directory name is allowed as the label. The reason is that every configuration set resides in a subdirectory of *IPaccounting-directory*`/conf`, that will be identified with the label above. For instance: `conf/foo-gw1`.

10.2 Creating a new configuration set

The easiest way to create a new empty configuration is by duplicating the `conf/.template` subtree supplied with the kit:

```
# cp -Rp conf/.template conf/foo-gw1
```

You will have to set up the `config` file included in the new configuration set you just created. The following sections will help you setting up things into your configuration set.

10.3 General-purpose configuration file `general.config`

Besides configuration-set specific configuration files named `conf/`*label*`/config`, you are allowed to create a general-purpose configuration file, namely `conf/general.config`, containing a number of parameters that are common to all the configuration sets you build. This file is useful, in particular, when you have more than one configuration set.

You may set any configuration parameters either in the general-purpose or in a configuration-set-specific configuration file. If a parameter is defined in both files, the configuration-set-specific setting is used.

[2] Since version 3.0. Versions 2.*x* used to call it `archive`.

[3] Since version 3.0. Versions 2.*x* used to call it `daily_reports`.

A template for the `conf/general.config` file is created at first by the `install` script. The template includes a number of parameters that are usually general-purpose. You *must* check the values assigned to the parameters and change them according to your configuration.

10.4 Setting up parameters in `general.config` and `config` files

This section describes all the parameters that may be defined inside the general-purpose configuration file (`conf/general.config`) and the configuration-set-specific ones (`conf/`*label*`/config`).

You may change or set a parameter whenever you need, because the configuration files are read by any IPaccounting module as it is started.

The syntax to define a parameter is:

parameter-name = *parameter-value*

The parameter value should be quoted, if it is a string.

Comments may be included in the configuration files, by preceding them with the "#" character.

Several parameters have a default value embedded in the software, that will be used in case you do not include a different setting into one of the configuration files.

The available parameters are:

PGPLOT_DIR You must assign to this parameter the path to the directory where PGPLOT package has been installed on your system. As a rule, it is the same path that should have been assigned to the environment value PGPLOT_DIR.

For instance:

```
PGPLOT_DIR = "/usr/local/pgplot"
```

This parameter is usually defined in the general-purpose file.

The default value is: "/usr/local/pgplot".

USE_NSLOOKUP This boolean parameter must be set to TRUE if you mean to use external calls to the *nslookup* command (see `NSLOOKUP_COMMAND` parameter below) to resolve the addresses of nodes included into the statistics. If set to FALSE, *gethostbyaddr* function will be used. The latest is often considerably slower than `nslookup` approach, but does not require an external operating-system-specific command to be executed.

```
USE_NSLOOKUP = FALSE
```

This parameter is usually defined in the general-purpose file.

The default value is: TRUE.

NSLOOKUP_COMMAND This parameter defines the syntax for the `nslookup` command, that will be used to resolve the addresses of nodes included into the statistics, through `bind`. The syntax is platform-dependent.

This parameter is taken into consideration only if the USE_NSLOOKUP parameter is set to TRUE.

You are suggested to limit the number of retries to reduce the seeking time, for instance:

```
NSLOOKUP_COMMAND = "/bin/nslookup -retry=1"
```

This parameter is usually defined in the general-purpose file.

The default value is: “nslookup”.

CACHE_LIFETIME A cache file is used by IPaccounting to make addresses resolutions faster (you will read more about this in section 14.2.1). The entries into the cache file should “expire” after some time, in order to reduce the size of the cache file by removing obsolete entries, and in particular to force IPaccounting to refresh the data stored into the cache, by querying them again through *bind*.

This parameter defines the lifetime of cache entries (in days):

```
CACHE_LIFETIME = 10
```

This parameter is usually defined in the general-purpose file.

The default value is: 30.

ADMINISTRATOR This parameter defines the name and/or the e-mail address of the local IPaccounting administrator for your organization.

For instance:

```
ADMINISTRATOR = "Donald Duck (donald@acme.org)"
```

This parameter is usually defined in the general-purpose file.

IMAGES_URL This parameter defines the URL that will make the `images` subdirectory of the IPaccounting tree accessible through a Web browser. For instance:

```
IMAGES_URL = "http://www.acme.org/IPaccounting/"
```

The `images` subdirectory includes both some pictures that are embedded in the IPaccounting pages, and the `images/OUTPUTS` subdirectory, in which the temporary output graphics will be created.

You must create a link from the httpd server tree, pointing to the `images` directory, in order to make it reachable through the Web. For instance:

```
# ln -s /usr/local/IPaccounting/images
        /usr/local/etc/httpd/htdocs/IPaccounting
```

Paths depend on your implementation.

This parameter is usually defined in the general-purpose file.

CGI_URL This parameter defines the URL that will make the `cgi-bin` subdirectory of the IPaccounting tree accessible through a Web browser. For instance:

```
CGI_URL = "http://www.acme.org/cgi-bin/IPaccounting/"
```

You must create a link from the `cgi-bin` subtree of the httpd server tree, towards the `cgi-bin` subdirectory of IPaccounting, in order to make it reachable through the Web. For instance:

```
# ln -s /usr/local/IPaccounting/cgi-bin
        /usr/local/etc/httpd/cgi-bin/IPaccounting
```

Paths depend on your implementation.

This parameter is usually defined in the general-purpose file.

RSH_1, RSH_2 and RSH_3 IPaccounting handles the connection to the monitored routers by executing a sequence of three rsh calls. The three parameters RSH_1, RSH_2 and RSH_3 define the syntax for the three rsh calls, in the right sequence.

In most of the cases, you should not need to touch them.

An exception is just in the case you are not meaning to use the root account on the IPaccounting server, as the IPaccounting manager. In this particular case, just change any "root" appearance in the RSH_*x* definitions with your favourite user name[4].

The default value for RSH_1 is: "rsh %s -l root clear ip accounting".

The default value for RSH_2 is: "rsh -n %s show ip account checkpoint >> %s".

The default value for RSH_3 is: "rsh %s -l root clear ip account checkpoint".

The "%s" fields are filled by the IPaccounting tools, with the address of the router and the name of the raw data file.

These parameters are usually defined in the general-purpose file.

ROUTER_NAME An arbitrary descriptive string identifying the monitored router. This string will be used in the graphics and reports. For instance:

```
ROUTER_NAME = "Router trieste-gw1.ts.infn.it in Trieste"
```

[4]The user must be enabled for rsh execution on the router, too.

This parameter is usually configuration-set specific.

ROUTER_ADDRESS The TCP/IP address of the router to be monitored. You may assign the numeric address or the hostname, the latest both in extended or short form, if the address can be resolved through `bind`:

```
ROUTER_ADDRESS = "trieste-gw1.ts.infn.it"
```

This parameter is usually configuration-set specific.

DOMAIN_NAME An arbitrary descriptive string identifying the accounting domain the configuration set refers to:

```
DOMAIN_NAME = "Scientific network in Trieste (AS 5449)"
```

This parameter is usually configuration-set specific.

10.5 Defining the accounting domain

The `accounting_domain.dat` file must be filled with the list of networks/nodes belonging to your accounting domain, one entry per line.

You may exclude a node or a network by the entity, by preceding it with the "-" sign. You may also use the wildcard "$*$" to replace any sequence of characters.

Comments start with the "#" character. For instance:

```
#
# This is an example of an accounting domain
#
# The domain includes networks 140.105. and 147.122.,
# except subnets 140.105.6. and 147.122.7.
#
140.105.
147.122.
-140.105.6.
-147.122.7.
```

The `accounting_domain.dat` file may be modified at any later time.

10.6 Defining subdomains and communication partners

You can build a new subdomain or a communication partner by creating a definition file for each entity. The rules are the same described in section 10.5.

10.6.1 Ordinary subdomains

In order to create a new ordinary subdomain, just edit a new definition file inside `ordinary_subdomains` subdirectory. The name you assign to the file will be used as the subdomain name in the reports and in the graphics.

You must create at the least one ordinary subdomain. As a minimum requirement, you might just design one covering the whole accounting domain.

You are suggested to choose short names for accounting subdomains and communication partners (less then 12 characters, if possible), in order to avoid them to be truncated in the graphics.

The administrator should ensure that the ordinary subdomains are not overriding each other, in order to get consistent and meaningful percentages and totals. There is no automatic control on the consistency: it is all under the administrator's responsibility.

Proceed in the same way to create as many ordinary subdomains as needed.

10.6.2 Special subdomains

The same rules apply to special subdomains. Those files are not mandatory, however. In case you need some, just create a file for each of them into `special_subdomains` subdirectory.

10.6.3 Communication partners

The communication partners can be built in the same way, by creating appropriate definition files into `communication_partners` subdirectory.

By default, a communication partner labelled "$*$" ("anywhere") is implicitly available to IPaccounting. Its definition is embedded in the code. This particular communication partner includes every node in the world.

A communication partner, which is often useful, includes all the networks and nodes outside your accounting domain. For this purpose, you might create a file which looks similar to the following (and name it something like `conf/`*label*`/communication_partners/world_outside_my_domain`:

```
#
# All the world, but accounting domain AS5449
#
*
-140.105.
-147.122.
-193.43.109.
-193.205.241.
```

In such a file, you include any node in the world, but the ones belonging to your accounting domain.

All the definitions files can be modified/removed/added at any later time.

11 Upgrading from version 2.1

The IPaccounting 3.0 kit includes a script that will help you upgrading your existing v.2.1 installation, making both configuration sets and data files available to the new release.

The upgrade will be performed in the same directory where version 2.1 has been installed.

An "uninstall" script will make you able to return back to version 2.1, in case of problems with 3.0.

11.1 Stopping previous version

Before starting the upgrade process, you should stop data acquisition by version 2.1. Data collection is usually performed by means of crontabs entries. You should remove or comment the lines invoking IPaccounting tasks in the crontab file of the user which administrates IPaccounting (i.e. `/var/spool/cron/crontabs/root`). Then, restart cron in order to stop IPaccounting periodic processes.

11.2 PGPLOT installation

You should have installed PGPLOT on your system, when you set up version 2.1 of IPaccounting. Just in case you did not, or if you want to upgrade PGPLOT to the latest release, read section 9.1.

11.3 Configuring the router

Even this step should have been performed in the previous installation. If you are in doubt, read section 9.2.

11.4 Installing IPaccounting 3.0

1. You must expand the IPaccounting tar file in a directory, which is *not* the one where the previous release has been installed.

   The final destination of the upgrade will actually be the directory where version 2.1 has been installed, but you must perform the first expansion elsewhere.

   In the following paragraphs, it is supposed that your top IPaccounting directory, where version 2.1 was installed, is `/usr/local/IPaccounting`, and that you expanded the kit under `/tmp/IPaccounting.3.0`.

   The tar file should contain the following files:

   ```
   README.txt
   LICENSE.txt
   INSTALL.txt
   install
   kit.tar
   ```

Read the LICENSE.txt file before going on.

Do not untar the `kit.tar` file, as it will be handled by the `install` script.

2. Copy the `install` script into IPaccounting directory:

```
# cp -p ./install /usr/local/IPaccounting/
```

   All the IPaccounting 3.0 stuff will be written below that directory.

3. Move to IPaccounting directory:

```
# cd /usr/local/IPaccounting/
```

4. Run the `install` script, and pass to it the location where you unpacked the IPaccounting tar file:

```
# ./install /tmp/IPaccounting.3.0
```

   The script will save your IPaccounting 2.1 tree (scripts only, the data will remain untouched) into a file named `pre3.0.tar`. This will make you able to return back to version 2.1 in case of problems with 3.0, by using the `uninstall` script, which will be described later.

   The `install` script will then remove the old stuff and install version 3.0. Finally, it will update any existing configuration file named `conf/`*label*`/configuration.pl`, creating a `conf/`*label*`/config`, suitable for version 3.0.

   In the following example, an IPaccounting 2.1 structure including two configuration sets, named foo-gw1 and bar-gw2, is upgraded:

```
# ./install /tmp/IPaccounting.3.0

*****************************************************************
*                                                               *
*            IPaccounting 3.0 Installation Script               *
*                                                               *
*****************************************************************

This script will help you installing IPaccounting 3.0, or
upgrading an existing v.2.1 installation to v.3.0.

IPaccounting v.2.1 installation found in the current directory.
This script will automatically upgrade the old structure to v.3.0,
making existing configuration sets and data files available to the
new version.
The v.2.1 stuff will be saved into a .tar file, to make you able
to return back to the old version in case of problems with v.3.0
If you do not like it, please quit and move this script to a new
location, for a new installation; in this case, you will have to
```

```
update the configuration structure by hand.

Do you want to proceed (Y/N)? y

The old release (except "conf" tree) will be saved into
./pre3.0.tar. This will make you able to return back to the
previous version, in case you have problems with v.3.0.
The "conf" tree, however, won't be changed, in order to make it
available to the new version. Only configuration files will be
updated.
Do you want to proceed (Y/N)? y

Saving the old stuff into ./pre3.0.tar...
(...)
Done

Removing the old stuff...
(...)
Done

Extracting the kit /tmp/IPaccounting.3.0/kit.tar in the current
directory...
(...)
Done

Creating config library src/IPconfig.h...
Done

Creating general-purpose configuration file conf/general.config...
Done

Creating src/Makefile...
Done

Updating specific configuration files...
   Upgrading conf/foo-gw1/configuration.pl to conf/foo-gw1/config...
   Upgrading conf/bar-gw2/configuration.pl to conf/bar-gw2/config...
Done

The installation is now complete.
Please check the files created or modified by this script, and
modify them, if needed:

   src/IPconfig.h      (you should check this file before going on)
   src/Makefile        (you should check this file before going on)
   conf/general.config (you may check this file later)
   conf/foo-gw1/config (you may check this file later)
   conf/bar-gw2/config (you may check this file later)

Then get inside src subdir and execute:

   # make
```

```
Then:

   # make install
```

The upgrade will be performed in the same directory where version 2.1 was installed. Therefore, the data files included in the old structure will not be touched. Just the configuration files will be upgraded and some appropriate links will be created inside the configuration sets subdirectories, in order to make the data directories accessible through the naming conventions of version 3.0.

5. As is suggested by the `install` script, you must carefully check the files `src/IPconfig.h` and `src/Makefile`.

   `src/IPconfig.h` should define only the top directory of IPaccounting. If you need to move IPaccounting to a different location, in the future, you will have to modify this file accordingly to your new top IPaccounting directory, and compile the whole stuff again as described below.

   `src/Makefile` will be used to actually build IPaccounting. It contains some platform-specific assignments, that you must carefully check and customize, in order to make them work on your operating system. In particular, have a look at the definition of PGPLOT_FLAGS variable.

   Later on, you will have to configure `conf/general.config` and all the configuration files that were upgraded by the `install` script, as described in chapter 10.

6. Get into `src` subdirectory and build the whole stuff:

   ```
   # make
   ```

   If everything goes right, move the binaries to their final destination (that is `bin` and `cgi-bin` subdirectories of top IPaccounting directory):

   ```
   # make install
   ```

11.5 Setting up crontabs

Section 9.5 describes how crontabs tasks must be set up. Notice that the syntax is quite different from previous releases.

11.6 Uninstalling IPaccounting 3.0

IPaccounting 3.0 kit includes a script named `uninstall`, which makes you able to return back to version 2.1, in case of problems with the new release. This script may be used only if you upgraded from version 2.1, and did not touch the `pre3.0.tar` file that was created by the `install` script. `uninstall` will remove v.3.0 stuff and restore v.2.1 files.

An example follows:

```
# ./uninstall

*****************************************************************
*                                                               *
*            IPaccounting 3.0 Uninstallation Script             *
*                                                               *
*****************************************************************

This script will help you uninstalling IPaccounting 3.0, and
returning back to your v.2.1 environment. It is of no use at all
if you did not upgrade a 2.1 environment to 3.0.

All v.3.0 stuff will be removed, and the available 2.1 files
will be restored from ./pre3.0.tar

Do you want to proceed to uninstall v.3.0 (Y/N)? y
Removing v.3.0 stuff...
(...)
Done

Restoring v.2.1 stuff from ./pre3.0.tar...
(...)
Done

You might want to remove "config" files inside configuration sets
under "conf" subdirectory, as they were created during the upgrade
to v.3.0.

You might also want to remove the ./pre3.0.tar file.

IPaccounting v.3.0 uninstalled. V.2.1 was recovered.

Please keep the author (IPaccounting@ts.infn.it) informed about
the reason why you decided to return back to v.2.1: any bugs,
unwanted or missing features, ...
```

12 Upgrading from a version older than 2.1

Upgrading from an IPaccounting version older than 2.1 is not easy to do. The data structure was completely changed with version 2.1, and older versions are not compatible with the new data structure.

You have the following choices:

1. Upgrade to version 2.1, and then upgrade to 3.0.
2. Install version 3.0 as a brand new installation. You will have to port your old data structures and configurations to the new release. This is usually easy if you have a simple accounting organization, but might be complex otherwise. If you choose

this way, read very carefully the chapters concerning the installation and the configuration of IPaccounting 3.0.

13 Collecting data from the router

IPaccounting periodically collects traffic information from the router. This is usually done through a crontab entry, as described in section 9.5.

You should create as many crontabs entries for this task, as the number of routers you need to monitor. For instance:

```
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-c foo-gw1 > /dev/null 2>&1
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-c bar-gw2 > /dev/null 2>&1
```

The `collect_data` tool just collects data from the router, and saves them according to the rules established for the configuration set, whose name is passed to the tool itself (option `-c`).

To ensure that data collection works correctly, just look at the size of the file in which the raw data are stored (`IPaccounting.dat` into each configuration set) and check if it grows after each `collect_data` call.

14 Computing statistics

Once a day, some daily accounting tasks must be performed, in order to compute statistics on the raw data collected by the router(s) during the day. This is usually done around midnight, in order to take into consideration all the data collected during the day at once.

Daily accounting is completed in two steps:

1. Three lists of traffic are built: by source, by target and by source-target couples. This is done by a tool named `make_lists`.

2. A short daily report, that will be used to build graphics and tables, is created. This is done by a tool named `make_reports`.

The second step may be performed only if the first one completed successfully, as it needs the lists generated during the first step.

Usually, the two phases are performed automatically, by means of appropriate crontabs entries.

A tool named `make_all_accounting` is available, in order to invoke `make_lists` and `make_reports` in the right sequence. This makes you able to do everything with a single shot, but can be used only on simple accounting organizations, i.e. a single router to be monitored, or several routers to be monitored indipendently. Fortunately, this covers most of the cases.

More complex accounting managements require different tasks, that are described in chapter 16.

All the tools described in this chapter are stored in the `bin` subdirectory of the IPaccounting tree.

14.1 Doing everything with a single shot: make_all_accounting

The `make_all_accounting` tool may be used in order to complete the whole daily accounting process, in the sites where only one router is monitored, or several routers are monitored as independent entities. In the latter case, the tool must be invoked separately for each configuration set which refers to a router to be monitored.

If your accounting organization is more complex, read section 16.

In section 9.5 you should have read some information about setting up an appropriate crontabs entry for `make_all_accounting`.

The complete syntax of `make_all_accounting` is the following:

```
# make_all_accounting   -c configuration-set-label
                        [-d date]
                        [-n number-of-entries]
                        [-a]
                        [-v]
```

The following options are available:

`-c` parameter is mandatory. You must always supply the label which identifies the configuration set to be used.

`-d` option indicates the date in which the data to be processed were collected. This is needed to identify the archive data file name to be used, which contains the date embedded in its name (`IPaccounting`.*date*). The *date* field is in the format *yyyymmdd*. If omitted, current date is used.

`-n` option can be used to make processing faster. While creating the lists of traffic, IPaccounting tries to resolve the IP addresses of the nodes through *bind* or *gethostbyaddr*, in order to make the output lists more readable, by displaying the hostnames instead of the numeric addresses. This task might require a very long time. To make things faster, IPaccounting uses two tricks: caching (see section 14.2.1), and the `-n` option. The latter sets the number of entries (at the top of the list, associated with the nodes that communicated the most) for which the resolution will be attempted. If you restrict this number, the computing time will decrease meaningfully. The final lists will include the complete information for the most significant nodes, while the translations for nodes which are (usually) less interesting, will not be included in the list.

If this option is omitted, the resolution will be attempted for all the addresses found in the lists.

`-a` option skips archiving the file containing the data collected from the router. Usually, the first task performed by the `make_all_accounting` script concerns moving the file containing the data taken from the router (`IPaccounting.dat`) to `raw_data/IPaccounting.`*date*. This is done both to save the data into their final location, and to clean the data file from the information collected daily.

If option `-a` is set, this step is not performed. This might be useful to process again a raw data file which has been saved already, for instance if the accounting procedure failed for some reasons, after the file had been renamed.

This option is seldom used. In particular it is almost always omitted from crontabs invokations of `make_all_accounting`.

`-v` option enables verbose mode, which will cause more messages to be displayed on your terminal. It is usually enabled for debugging purposes only.

`make_all_accounting` does actually nothing special: it is just an interface which invokes the two tools which really do the job: `make_lists` and `make_reports`. Therefore, even if `make_all_accounting` is all you need to invoke to compute your accounting, you should read the following sections to understand what really happens when you run `make_all_accounting`.

14.2 Building sorted lists of traffic: `make_lists`

`make_lists` is usually invoked by `make_all_accounting`, in order to build sorted lists of traffic by source, by target and by source-target couples. You might need running this script by hand (that is, not through `make_all_accounting`) only for a complex accounting management.

`make_lists` performs the following tasks:

1. It moves the `IPaccounting.dat` file, containing the data collected from the router, to `raw_data/IPaccounting.`*date*.

2. It builds three lists of traffic: by source (`acc_source.`*date*), by target (`acc_target.`*date*) and by source-target couple (`acc_source_target.`*date*). The lists are saved in the `lists` subdirectory of the configuration set at issue.

You will find more information on the traffic lists created by `make_lists` in chapter 17.1.

The complete syntax for `make_lists` is the following:

```
# make_lists      -c configuration-set-label
                 [-d date]
                 [-n number-of-entries]
                 [-a]
                 [-m]
                 [-v]
```

The meaning of -c, -d, -a, -n and -v options is the same described for the homonymous options of make_all_accounting in section 14.1.

Option -m does the opposite of -a: no computing is done, just the archiving of the data file will be performed. This option is useful only for very particular managements of several router, that will be described in section 16.

14.2.1 Caching for addresses resolution

The lists of traffic built by make_lists include the resolution of the IP addresses of the nodes which did some traffic through the monitored router. The addresses are resolved through *bind* or *gethostbyaddr*. In order to make resolution faster, a caching system is handled by make_lists, which stores the list of resolved addresses, with their translation. The file which contains the cache is named .cache and is located in the main IPaccounting directory. The content of this file looks like follows:

```
128.141.201.71 na47sun05.cern.ch 19990316
140.105.6.100 unixts.ts.infn.it 19990401
147.122.11.23 neumann.sissa.it 19990318
147.162.75.2 ? 19990404
```

The three fields composing each record store the numeric address and its resolution, and the date when the resolution was written into the cache file.

Every time IPaccounting needs to resolve an address, it does the following, in sequence:

1. It looks for the resolution in the cache file.

2. If it is not available, it tries to resolve the address by using the nslookup command or through *gethostbyaddr*, according to USE_NSLOOKUP configuration parameter. If the address is not resolved, the script assumes that the hostname is not defined and translates it as "?" (see the example above). In any case, the address and its translation are written into the cache file.

In this way, the cache file is progressively filled with addresses resolutions, which may be obtained in the next processes without using *bind* or *gethostbyaddr*, improving efficiency.

The date field appended to each record, is used to force the expiration of the entries after some time (defined by the CACHE_LIFETIME parameter in the configuration files). This is done to grant the refresh of addresses which might otherwise become obsolete.

An example of the output of make_lists, even when invoked by make_all_accounting, follows:

```
make_lists starts on Thu Apr 15 15:01:31 1999

*** Creating traffic lists for 15-Apr-1999 ***
Reading data...
Sorting list by source...
```

```
Writing list by source...
Sorting list by target...
Writing list by target...
Sorting list by source-target...
Writing list by source-target...
The output lists are now complete.
Translating IP addresses (output lists won't be modified
until successful completion) ...
     Loading cache...
     Working on list by source-target...
     Working on list by source...
     Working on list by target...
     Updating cache...
make_lists ends on Thu Apr 15 15:01:47 1999
```

14.3 Building daily reports: make_reports

`make_reports` is usually invoked by `make_all_accounting` in order to build a short daily report from the `archive/acc_source_target.`*date* lists created by `make_lists`. The report contains a summary of the daily traffic between each accounting subdomain and each communication partner. The information stored in the report will be used to create tables and graphics accessible through the World Wide Web.

You might need running this script by hand (that is, not through `make_all_accounting`) only for a complex accounting management.

The daily reports are stored in the `reports` subdirectory of the configuration sets. They are named `reports/daily.`*date*. You should store only the files at issue in that subdirectory.

You will read more about the format of daily reports in chapter 17.2.

The complete syntax of `make_reports` is the following:

```
# make_reports   -c configuration-set-label
                [-d date]
                [-v]
```

The meaning and usage of the three options is the same described for the homonymous options of `make_all_accounting` in section 14.1.

An example of the output of `make_reports` (even when invoked by `make_all_accounting`) follows (verbose mode disabled):

```
make_reports starts on Thu Apr 15 15:20:50 1999

*** Creating daily reports for 15-Apr-1999 ***
Loading communication partners data from /usr/local/IPaccoun
ting/conf/foo-gw1/communication_partners/*...
Loading accounting domain data from /usr/local/IPaccounting/
conf/foo-gw1/accounting_domain.dat...
Loading ordinary subdomains data from /usr/local/IPaccountin
g/conf/foo-gw1/ordinary_subdomains/*...
```

```
Loading special subdomains data from /usr/local/IPaccounting
/conf/foo-gw1/special_subdomains/*...
Building reports...

(...)

Writing output...

make_reports ends on Thu Apr 15 15:23:19 1999
```

While "Building reports..." the script might display a lot of messages concerning "strange" entries found inside the list of traffic. They might be of two kinds:

1. Both addresses of source and target nodes, found in a list entry, do not belong to the accounting domain. In this way you should be able to identify any eventual "foreign" traffic[5] passing through your router.

2. One address belonging to the accounting domain do not belong to any ordinary subdomains. This might be a desired setting, but is signaled anyway as a "strange" entry, as you might just have forgotten to include the address in one ordinary subdomain. In fact, ordinary subdomains usually cover the whole accounting domain.

15 Making the data accessible through the Web

IPaccounting output lists and reports are accessibile through a Web browser. You should have created appropriate links to the `images` and `cgi-bin` subdirectories of the IPaccounting tree when you created the configuration files (see chapter 10).

IPaccounting HTML pages are all generated "on the fly" by a set of CGIs. The CGI in charge for generating the IPaccounting main page, giving access to a set of IPaccounting data, is `cgi-bin/main`.

You might want to include an URL pointing to that CGI into one of your HTML pages. You just have to pass to it the label of the configuration set you mean to consider. In this way you may also set up different URLs, pointing to `cgi-bin/main` with different configuration sets labels, making you able to access the statistics collected by different routers. For instance:

```
<a href=http://www.ts.infn.it/cgi-bin/IPaccounting/main?foo-gw1>
TCP/IP usage statistics home page for router foo-gw1</a>
<a href=http://www.ts.infn.it/cgi-bin/IPaccounting/main?bar-gw2>
TCP/IP usage statistics home page for router bar-gw2</a>
```

You *must* indicate explicitly the label of the configuration set which you are referring to, by appending the "?" character followed by the label itself.

The path to the `main` CGI is the same that you should have defined through the CGI_URL configuration parameter.

[5] That is, coming from an external node and going to another external node

For more information about the Web pages generated by IPaccounting's CGIs, read the third part of this document.

15.1 Protecting data on the Web

If you need to protect the statistics built by IPaccounting, with the usual methods available on your httpd server (by password or by network, for instance), just disable the access to the `cgi-bin` subdirectory of IPaccounting. The way to do this depends on your httpd server. On some servers, for instance, you must create a `.htaccess` file containing the protection rules, into the directory you mean to protect.

For more information, read the documentation of your httpd server.

16 Monitoring more than one router

IPaccounting makes you able to manage the traffic information collected by several routers at the same time. You may want to monitor the router separately, or aggregate the data coming from different routers.

If you choose to administer the routers separately, the data should be collected on a per-router basis, and all the following processing tasks should be done for each router on its own. In this way, you may easily evaluate the traffic that passes through each router.

If you prefer to handle several routers as a unique entity, the data should be collected from the routers and possibly stored separately, to make you able to do some more processing on the single files, if needed. Then the data should be aggregated to form a unique file including all of them. The file will then be processed as if it referred to a single router.

In both cases, a configuration set for each processing sequence must be created. The following sections will help you setting up a complex accounting structure.

16.1 Monitoring routers separately

If you need to manage several routers separately, you must create a specific configuration set for each router.

In this way, all the data, starting from raw data collected by the routers up to the daily reports, will be stored in the appropriate subdirectories of the configuration set you supply to the accounting scripts.

To make things automatic, you should add appropriate crontabs entries for any router you need to monitor. For instance, supposing you built two configuration sets named *foo-gw1* and *bar-gw2*, you will need something like the following in the crontabs file:

```
#
# Get data from foo-gw1 every 10 minutes
#
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-c foo-gw1 > /dev/null 2>&1
```

```
#
# Get data from bar-gw2 every 10 minutes
#
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-c bar-gw2 > /dev/null 2>&1
#
# Every night, make accounting for foo-gw1
#
59 23 * * * /usr/local/IPaccounting/bin/make_all_accounting
-c foo-gw1 -n 500 > /usr/local/IPaccounting/acct_foo-gw1.log 2>&1
#
# Every night, make accounting for bar-gw2
#
59 23 * * * /usr/local/IPaccounting/bin/make_all_accounting
-c bar-gw2 -n 500 > /usr/local/IPaccounting/acct_bar-gw2.log 2>&1
```

16.2 Monitoring several routers as a unique entity

In some cases, you might like to manage several routers, bordering a unique accounting domain, as a single entity, to make global computing on a per-domain basis instead of a per-router basis. In this case, you have to aggregate the data collected by the routers, sooner or later in the processing sequence.

A complete processing sequence consists of the following steps:

1. Saving data coming from the router in the raw data archive.
2. Building lists of traffic from the raw data.
3. Building daily reports from the lists of traffic.

You are free to choose the point of the chain where the data must be aggregated: after that point, the data will be handled as if they referred to an unique router.

For instance, you might choose to aggregate the raw data at the beginning, creating a unique raw data file containing the data coming from all the routers. Or you might like to keep the raw data files separate, for alternative analysis, and to produce just a unique traffic list for your domain.

According to the chosen algorithm, a script (or a set of scripts) should be built, in order to perform the needed tasks in the right sequence.

Due to the variety of possible cases, the construction of such a script is not easy to examplify. However, what follows is an example of such a configuration, currently in use at I.N.F.N., Sezione di Trieste: it allows monitoring two routers in aggregation. The raw data coming from the routers are collected and archived separately. Then raw data are concatenated, and a unique set of traffic lists and daily reports is produced, in order to include all the information concerning the accounting domain bordered by the two routers.

Three configuration sets have been defined: one for each router (namely: *trieste-gw1* and *trieste-gw2*), handling the first phase of the processing, before the aggregation, and one, named *global*, which handles operations after the aggregation.

Some items are common to the configuration sets. For instance, the definition for the accounting domain and subdomains are the same for the three configuration sets. On a Unix machine, this can be handled easily, by replacing some of the files and directories included in the configuration sets with appropriate links.

For instance, the `conf/trieste-gw1` directory might be structured like this:

```
-rw-r--r--  (...) 175242 (...)  IPaccounting.dat
lrwxrwxrwx  (...)     31 (...)  accounting_domain.dat ->
../global/accounting_domain.dat
lrwxrwxrwx  (...)     32 (...)  communication_partners ->
../global/communication_partners
-rw-r--r--  (...)    292 (...)  config
lrwxrwxrwx  (...)     15 (...)  lists -> ../global/lists
lrwxrwxrwx  (...)     29 (...)  ordinary_subdomains ->
../global/ordinary_subdomains
drwxr-xr-x  (...)   1024 (...)  raw_data
lrwxrwxrwx  (...)     17 (...)  reports -> ../global/reports
lrwxrwxrwx  (...)     28 (...)  special_subdomains ->
../global/special_subdomains
```

The `conf/trieste-gw2` directory is similar:

```
-rw-r--r--  (...)  93847 (...)  IPaccounting.dat
lrwxrwxrwx  (...)     31 (...)  accounting_domain.dat ->
../global/accounting_domain.dat
lrwxrwxrwx  (...)     32 (...)  communication_partners ->
../global/communication_partners
-rw-r--r--  (...)    280 (...)  config
lrwxrwxrwx  (...)     15 (...)  lists -> ../global/lists
lrwxrwxrwx  (...)     29 (...)  ordinary_subdomains ->
../global/ordinary_subdomains
drwxr-xr-x  (...)   1024 (...)  raw_data
lrwxrwxrwx  (...)     17 (...)  reports -> ../global/reports
lrwxrwxrwx  (...)     28 (...)  special_subdomains ->
../global/special_subdomains
```

The `conf/global` directory is like follows:

```
-rw-r--r--  (...)    187 (...)  accounting_domain.dat
drwxr-xr-x  (...)   8192 (...)  communication_partners
-rw-r--r--  (...)    308 (...)  config
drwxr-xr-x  (...)   8192 (...)  lists
drwxr-xr-x  (...)   8192 (...)  ordinary_subdomains
drwxr-xr-x  (...)   8192 (...)  raw_data
drwxr-xr-x  (...)   8192 (...)  reports
drwxr-xr-x  (...)   8192 (...)  special_subdomains
```

Some configuration parameters can be given the same value for all the configuration sets, and can be therefore defined inside the general configuration file `conf/general.-config`. Others are specific to the different configuration sets. Some parameters are

unused by some configuration sets: for instance, the ROUTER_ADDRESS is not used by the global configuration set, as it is used only to collect data from the routers. The same applies for the IMAGES_URL and CGI_URL parameters, which are used only to access data through the Web, by the global configuration set.

An example of the configuration files follows:

conf/general.config:

```
PGPLOT_DIR = "/usr/local/pgplot"
NSLOOKUP_COMMAND = "nslookup -retry=2"
USE_NSLOOKUP = TRUE
CACHE_LIFETIME = 10
ADMINISTRATOR = "root@www.ts.infn.it"
IMAGES_URL = "http://www.ts.infn.it/images/IPaccounting/"
CGI_URL = "http://www.ts.infn.it/cgi-bin/IPaccounting/"
RSH_1 = "rsh %s -l root clear ip accounting"
RSH_2 = "rsh -n %s show ip account checkpoint >> %s"
RSH_3 = "rsh %s -l root clear ip account checkpoint"
DOMAIN_NAME = "Scientific networks in Trieste"
```

conf/trieste-gw1/config:

```
ROUTER_NAME="trieste-gw1.ts.infn.it"
ROUTER_ADDRESS="trieste-gw1.ts.infn.it"
```

conf/trieste-gw2/config:

```
ROUTER_NAME="trieste-gw2.ts.infn.it"
ROUTER_ADDRESS="trieste-gw2.ts.infn.it"
```

conf/global/config:

```
ROUTER_NAME="trieste-gw1.ts.infn.it and trieste-gw2.sist.trieste.it"
```

While the data collection is performed in an ordinary way, the daily accounting is performed by running a script named make_ts_accounting, instead of the make_all-_accounting tool. The script looks like this:

```
#!/bin/sh
#
# make_ts_accounting
#
# By C.Strizzolo - INFN Trieste (Feb 1999)
#
# This script is invoked by crontab at INFN Trieste, to build the complete
# IP accounting for the two routers used in Trieste (namely: trieste-gw1
# and trieste-gw2). The data processing tree is as follows:
#
#    (data coming from trieste-gw1)       (data coming from trieste-gw2)
#                  |                                    |
```

```
#                |                                    |
#                V                                    V
#  conf/trieste-gw1/IPaccounting.dat    conf/trieste-gw2/IPaccounting.dat
#                |                                    |
#                |                                    |
#                V                                    V
#            archive to                           archive to
#     conf/trieste-gw1/raw_data            conf/trieste-gw2/raw_data
#                |                                    |
#                |                                    |
#                --------------------------------------
#                                    |
#                                    |
#                                    V
#                  aggregate data into a temporary file
#                                    |
#                                    |
#                                    V
#                         build unique statistic
#
# get date

if test $# != 0
then
  current_date="$1"
else
  current_date=‘date +"%Y%m%d"‘
fi

cd /usr/local/IPaccounting

# archive IPaccounting data coming from the routers

bin/make_lists -d$current_date -m -ctrieste-gw1
bin/make_lists -d$current_date -m -ctrieste-gw2

# aggregate the two archived files into a temporary file

echo "aggregating..."

cat conf/trieste-gw1/raw_data/IPaccounting.$current_date
    conf/trieste-gw2/raw_data/IPaccounting.$current_date
    > conf/global/raw_data/IPaccounting.$current_date

# make all accounting on the temporary file,
# as if it referred to a single router

bin/make_lists -d $current_date -n 500 -a -c global
status=$?
if [ "$status" != 0 ]
then
  exit
```

```
fi
bin/make_reports -d $current_date -c global
status=$?
if [ "$status" != 0 ]
then
  exit
fi

# remove temporary file

rm conf/global/raw_data/IPaccounting.$current_date
gzip conf/trieste-gw1/raw_data/IPaccounting.$current_date
gzip conf/trieste-gw2/raw_data/IPaccounting.$current_date
```

The crontabs entries look like this:

```
#
# Get data from trieste-gw1
#
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-ctrieste-gw1 > /dev/null 2>&1
#
# Get data from trieste-gw2
#
0,10,20,30,40,50 * * * * /usr/local/IPaccounting/bin/collect_data
-ctrieste-gw2 > /dev/null 2>&1
#
# Perform nightly accounting
#
58 23 * * * /usr/local/IPaccounting/make_ts_accounting >
/usr/local/IPaccounting/make_ts_accounting.log 2>&1
```

In order to make data accessible through the Web, the *global* configuration set label is passed to the `main` CGI, as it stores the final statistics:

```
<a href=http://www.ts.infn.it/cgi-bin/IPaccounting/main?global>
TCP/IP usage statistics</a>
```

17 Understanding output files

Usually, IPaccounting output reports are accessed through the Web. You will read more about that in the third part of this document. However, the IPaccounting administrator might need to check and understand the output files created by IPaccounting without passing through a browser. This approach is described in the following section.

17.1 Lists of traffic

Three lists of traffic are built every day, starting from the raw data collected by the router.

The lists are stored in the `lists` subdirectory of each configuration set. They contain aggregate data which are grouped with different criteria, as follows:

1. The `acc_source.`*date* file lists all the nodes which sent packets through the router, either belonging to the accounting domain or not. It is sorted by number of sent bytes and is usually called *the list by source*.

   For each node, the list includes its address, its resolution and the number of transmitted bytes:

   ```
   193.205.245.5 king.nis.garr.it 5162398
   140.105.16.91 sphe1.ictp.trieste.it 5131938
   140.105.6.99 cosine-gw.infn.it 4885588
   193.232.223.10 ? 4794760
   140.105.13.15 sci.area.trieste.it 4783823
   ```

2. The `acc_target.`*date* file lists all the nodes which received packets through the router, either belonging to the accounting domain or not. It is sorted by number of received bytes and is usually called *the list by target*.

   The format of this list is the same of the *list by source*.

3. The `acc_source_target.`*date* file lists all the source-target couples which communicated through the router. It is sorted by number of transferred bytes and is usually called *the list by source-target*.

   For each couple, the source and target nodes information and the number of transferred bytes are recorded:

   ```
   147.162.100.180 giotto.unipd.it 140.105.6.73 ? 235097
   129.187.181.23 ? 140.105.6.99 cosine-gw.infn.it 234645
   131.154.1.4 server1.infn.it 140.105.6.153 axts03.ts.infn.it 234421
   ```

The symbol "?" that you may find in the lists, in the place of a hostname, means that the address resolution through *bind* (or *gethostbyaddr*) was not successful.

The symbol "-" indicates the entries for which the resolution was not attempted at all, due to a limitation on the number of top entries to resolve, established by the administrator (see the *-n* option available to the accounting tools).

17.2 Daily reports

Daily reports contain daily summaries of the traffic. For each accounting subdomain, either ordinary or special, the total traffic to/from each registered communication partner is summarized. The information stored in the reports will be used to build tables and graphics accessible through the World Wide Web.

The reports are archived in the `reports` subdirectory of each configuration sets.

The reports are lists of records, with the following format:

subdomain communication-partner sent-bytes received-bytes

Each record indicates the total number of bytes sent and received by the accounting subdomain at issue, respectively to and from the communication partner. For instance:

```
INFN * 1446525134 299621328
INFN Italy 99609556 136211570
INFN outside_Italy 1346915578 163409758
INFN world_outside_AS5449 1382869390 236502771
Universita * 165891268 563306
Universita Italy 165891268 563306
Universita outside_Italy 0 0
Universita world_outside_AS5449 161853951 0
```

In the example above, three communication partners have been defined: Italy, outside_Italy and world_outside_AS5449. The fourth one, named "$*$", is actually the "anywhere" partner, which is defined by default to include any possible addresses. The report shows the traffic produced by subdomains "INFN" and "Universita" to and from each communication partner.

18 Removing obsolete files

The raw data files, contained in the `raw_data` subdirectories of the configuration sets, are used only to build the traffic lists. If the latter have been created correctly, the raw data may be removed or compressed, if needed.

The traffic lists and the daily reports are accessible through the Web. You should not remove or compress them, in order to provide access to as many data as possibile.

The temporary graphics files, created by the CGIs into the `images/OUTPUTS` subdirectory when a graphical output is required through the Web, should be cleaned periodically, as described in section 9.5.

Part III
User's guide

19 User interface characteristics

IPaccounting makes you able to access data and final reports through a Web browser, supporting HTML 3.0 or later. A color monitor is recommended, to take advantage of the chromatic features which provide better readability of graphic outputs. You must install a PostScript and/or a GIF files viewer, in case you want graphics to be displayed in one of those formats, and your browser is not able to manage them by itself.

IPaccounting pages are dynamic: they are built on the fly when they are invoked.

Every tree of pages created by IPaccounting, which usually refers to an accounting domain, has its root in the *main page*.

20 IPaccounting main page

The IPaccounting *main page* makes you able to access the data and statistics concerning an accounting domain.

One or more IPaccounting *main pages* may be available on an IPaccounting server, according to the number of monitored accounting domains.

See figure 2 for an example of an IPaccounting main page.

An IPaccounting main page is invoked by calling the right CGI tool, and passing to it the label of the configuration set you are interested in. The URL looks something like:

```
http://www.ts.infn.it/cgi-bin/IPaccounting/main?foo-gw1
```

However, your administrator probably set up such a pointer somewhere in a Web page; you just have to click on the pointer.

IPaccounting main menu makes you able to perform the following tasks:

- Display domains definitions.

  You are allowed to view the lists of subnets and nodes belonging to each defined domain/subdomain, such as they have been set up by the IPaccounting administrator. The following options are available:

  - Display the accounting domain definition.
  - Display the ordinary subdomains definitions.
  - Display the special subdomains definitions.
  - Display the communication partners definitions.

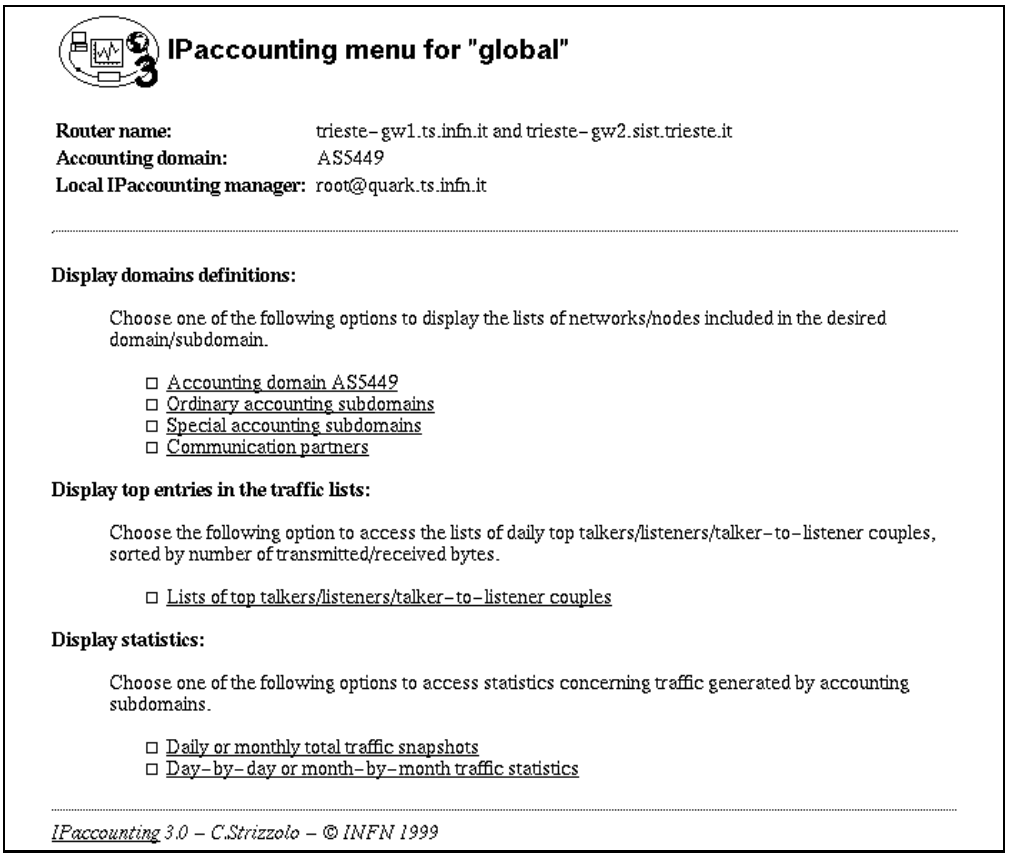

Figure 2: Example of an IPaccounting main page.

- Display top entries in the traffic lists.

  This option makes you able to display the daily top talkers/listeners/talker-to-listener couples, sorted by number of transferred bytes.

- Display traffic statistics, in a tabular or graphic format. Two different kinds of statistics are available:

  – Daily or monthly total traffic snapshots, displaying the total monitored traffic for a subdomain in a day/month.

  – Day-by-day or month-by-month traffic statistics.

21 Display domains definitions

The pages displaying the list of subnets and nodes belonging to an accounting domain, to the subdomains and to the communication partners, are structured in a very similar fashion.

In each list, items preceded by a "-" sign are excluded from the entity. As usual, the wildcard "$*$" replaces any sequence of characters. Comments start with the "#" symbol. For instance

```
#
# This is an example of a subdomain
#
# It includes networks 140.105. and 147.122.
# except subnet 140.105.6
140.105.
147.122.
-140.105.6.   # Let's exclude a network
```

22 Displaying traffic lists

Section 17.1 describes the lists of traffic and the data they contain. You are suggested to read it before going on.

Displaying those lists makes you able to answer questions such as: Which nodes sent/received the most on date *x*? Which communications have been the heaviest ones? Which nodes talked to node *x* on date *y*[6]?

You are allowed to access the lists by filling an HTML form with some information related to the data you want to display. The form looks like figure 3.

First, select the list you want to access (top source nodes, top target nodes or top source to target communications) and the date. Then, supply the number of entries to be displayed (100 by default); if you want to display all the entries, just let this field empty at all.

When your selection is complete, click on the "Do it" icon.

23 Display traffic statistics

23.1 Daily or monthly total traffic snapshots

IPaccounting makes you able to build a daily or monthly summary including one entry for each subdomain belonging to a subset you define. The report gives information about the total amount of traffic generated by each subdomain in a day/month.

In this case too, the access to the data is performed through a form. An example is in figure 4.

In order to fill the form, follow these steps:

1. Choose the computing time interval. This indicates if the summary must refer to a day or a month period.

2. Select the desired date. If at the previous step you choose the monthly option, only the month and the year are meaningful.

[6] Actually, to answer this, you should do some work on the lists, in order to extract the entries related to the node you are interested in.

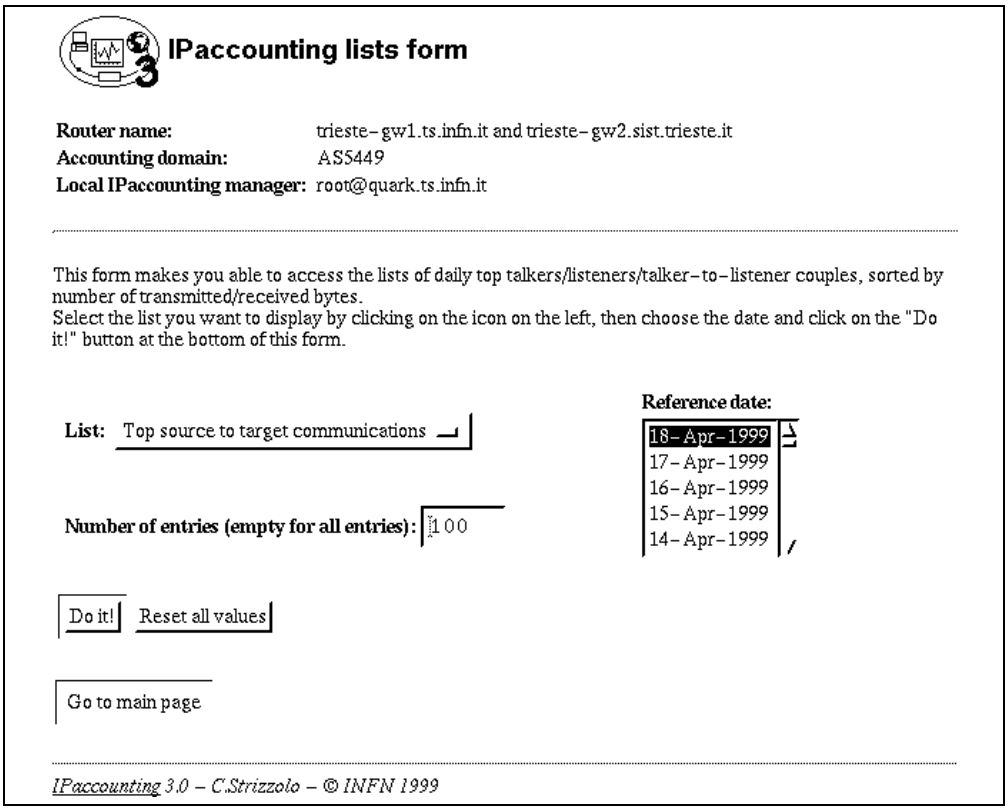

Figure 3: Traffic lists access form.

3. Select the unit of measure to be used in the summaries.
4. Select the output format.
5. Select the class of subdomains (ordinary or special) by clicking on the icon close to the two lists. The summary may include subdomains belonging to one of the two classes only.
6. In the list which corresponds to the subdomains class you have chosen, select the subdomains you are interested in (multiple choices allowed).
7. Select the communication partner you are interested in: the traffic between the selected partner and the above marked subdomains will be taken into consideration.
8. Select the traffic direction: *from* the subdomain, *to* the subdomain or both.
9. Click on “Do it” icon.

Two kinds of outputs are available: tabular or graphic.
An example of a tabular output of a summary snapshot is in figure 5.
A graphic output is in figure 6.

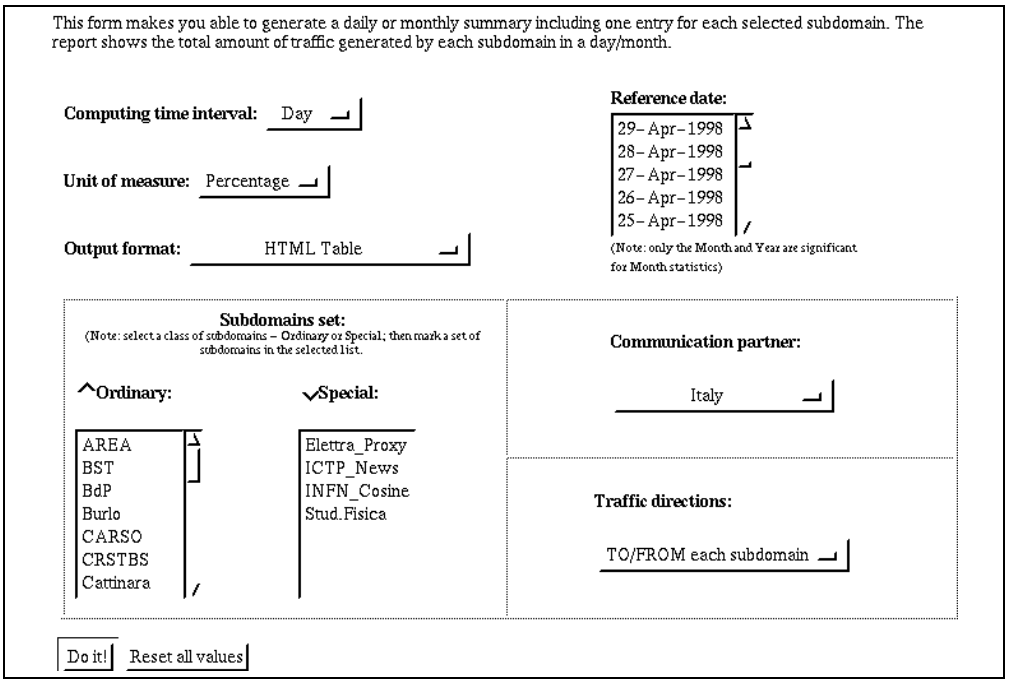

Figure 4: Traffic summary shapshot form.

Date:	23-Mar-1999
Report type:	daily total snapshot
Unit of measure:	MBytes
Traffic:	to+from subdomain from+to world_outside_AS5449

Subdomain	to+from world_outside_AS5449
Dip.Fisica	210.1
Fis.Teorica	100.8
ICTP	3266.6
INFN	641.8
Universita	5961.1
TOTAL	**10180.3**

Figure 5: Example of a tabular summary snapshot.

23.2 Day-by-day or month-by-month traffic statistics

IPaccounting makes you able to display the traffic statistics, for one or more accounting subdomains, on a day-by-day or month-by-month basis.

This kind of reports allows comparing subdomains, or communication partners which communicated with different subdomains.

Access to the data is granted through a form, similar to the one used for the snapshot summaries. In this case, however, you are allowed to select multiple traffic directions and

communication partners at once.

Tables and graphics produced in this way are usually more complex than the one concerning snapshot summaries. For instance, a table might look like the one in figure 7.

Figure 8 is an example of a graphic output.

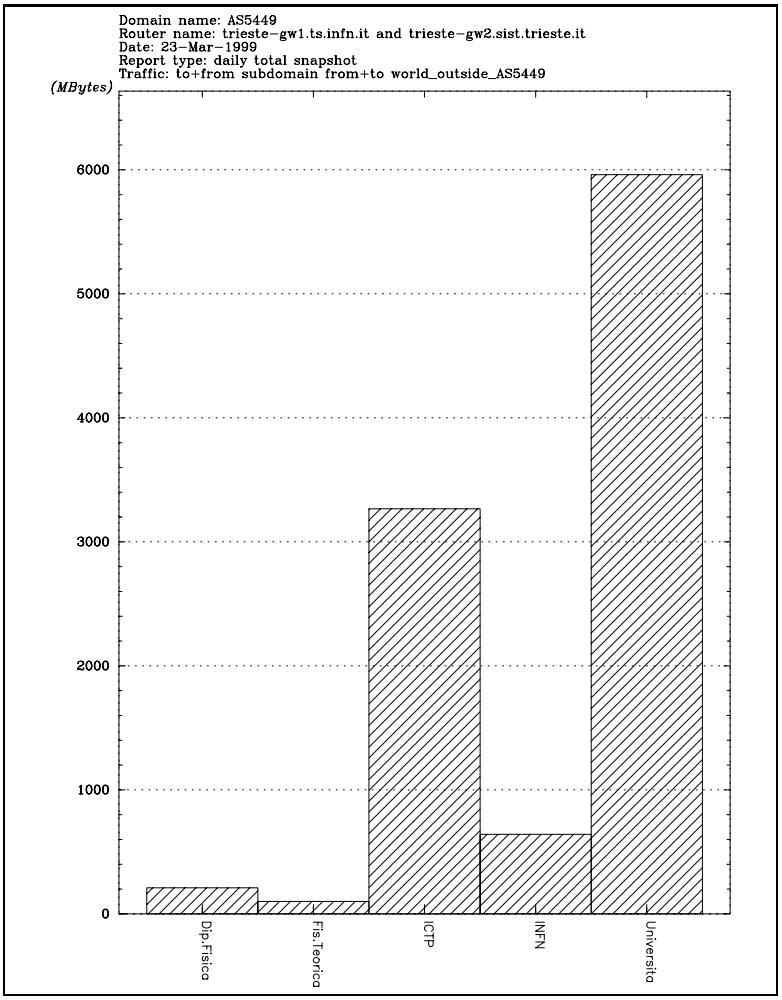

Figure 6: Example of a graphic summary snapshot.

Date:	1998
Report type:	month-by-month statistics
Unit of measure:	MBytes

Month	INFN to+from world_outside_AS5449	Universita to+from world_outside_AS5449	TOTAL
199801	31874.6	97036.1	**128910.7**
199802	33528.2	74357.0	**107885.1**
199803	32864.9	87585.8	**120450.7**
199804	20515.4	79832.6	**100347.9**
199805	33745.9	82017.3	**115763.2**
199806	16501.5	58269.3	**74770.8**
199807	12448.3	45221.0	**57669.3**
199808	18733.2	69551.6	**88284.8**
199809	28949.6	88868.2	**117817.8**
199810	20244.9	89796.2	**110041.2**
199811	19541.2	70979.6	**90520.8**
199812	34176.0	77892.2	**112068.2**
TOTAL	**303123.6**	**921406.9**	

Figure 7: Example of a tabular month-by-month statistic.

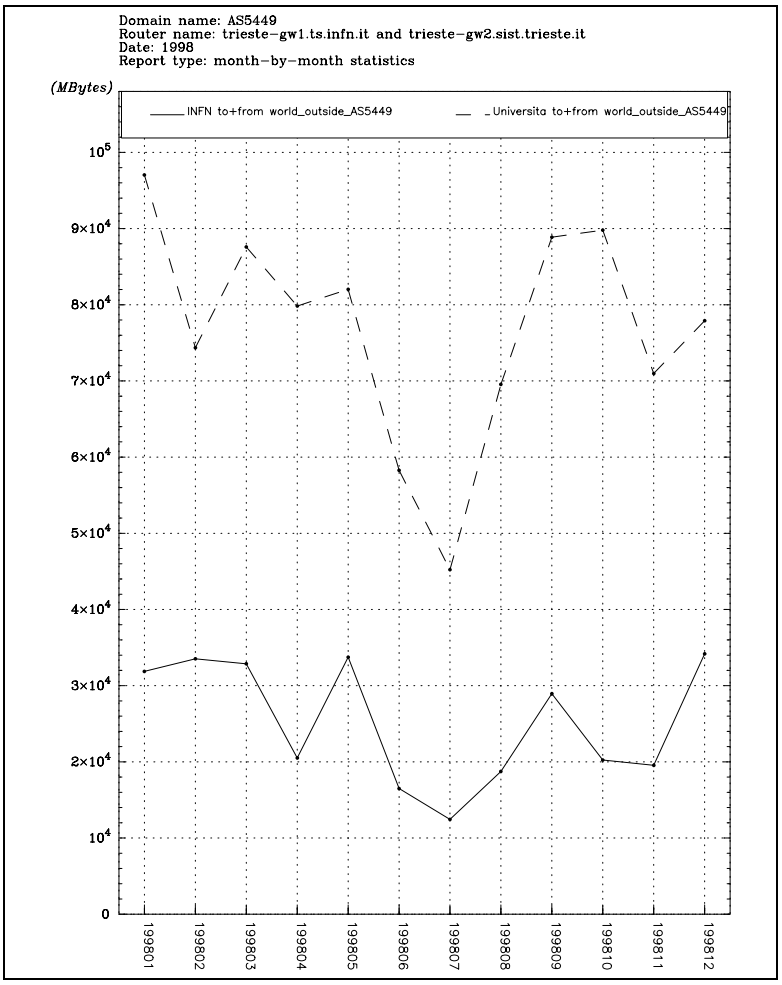

Figure 8: Example of a graphic month-by-month statistic.

Acknowledgements

Thanks to Roberto Gomezel for his precious help designing the new data structures, and to Giuseppe Della Ricca for his fundamental contribution porting the software to Linux.

Thanks also to all the people who provided feedback about the previous releases of the package.

PGPLOT package was developed by Tim Pearson, California Institute of Technology.

"aa" (Associative Array) library, included in the kit, was developed by Tim Behrendsen.

CGIHTML library, included in the kit, was developed by Eugene Eric Kim.

References

[1] Tim Pearson, PGPLOT documentation (http://astro.caltech.edu/ tjp/pgplot).

[2] Tim Behrendsen, Associative Array Libraries documentation (http://www.cerfnet.-com/ timb/alg_aa.htm).

[3] Eugene Eric Kim, CGIHTML documentation (http://www.eekim.com/software/-cgihtml/).

[4] Cisco Systems, Router products configuration guide.

Contents

I Introduction **3**

1 Product description **3**

2 Structure of this document **4**

3 What is new **4**

4 Terminology **5**
4.1 Ordinary and special subdomains 5
4.2 Communication partners 5

5 More information **6**

II Administrator's guide **7**

6 Pre-installation requirements **7**
6.1 System requirements 7
6.2 Software requirements 7

7 Software by other people **7**

8 Getting the kits **8**

9 Installing **8**
9.1 Installing PGPLOT 8
9.2 Configuring the router 9
9.3 Installing IPaccounting 3.0 9
9.4 Configuring 12
9.5 Setting up crontabs 12

10 Configuring **13**
10.1 Configuration sets 13
10.2 Creating a new configuration set 14
10.3 General-purpose configuration file `general.config` 14
10.4 Setting up parameters in `general.config` and `config` files 15
10.5 Defining the accounting domain 18
10.6 Defining subdomains and communication partners 18
10.6.1 Ordinary subdomains 19
10.6.2 Special subdomains 19
10.6.3 Communication partners 19

11 Upgrading from version 2.1 **20**
11.1 Stopping previous version 20
11.2 PGPLOT installation 20
11.3 Configuring the router 20
11.4 Installing IPaccounting 3.0 20
11.5 Setting up crontabs 23
11.6 Uninstalling IPaccounting 3.0 23

12 Upgrading from a version older than 2.1 **24**

13 Collecting data from the router **25**

14 Computing statistics **25**
14.1 Doing everything with a single shot: make_all_accounting 26
14.2 Building sorted lists of traffic: `make_lists` 27
14.2.1 Caching for addresses resolution 28
14.3 Building daily reports: make_reports 29

15 Making the data accessible through the Web **30**
15.1 Protecting data on the Web 31

16 Monitoring more than one router **31**
16.1 Monitoring routers separately 31
16.2 Monitoring several routers as a unique entity 32

17 Understanding output files **36**
17.1 Lists of traffic 36
17.2 Daily reports 37

18 Removing obsolete files **38**

III User's guide **39**

19 User interface characteristics **39**

20 IPaccounting main page **39**

21 Display domains definitions **40**

22 Displaying traffic lists **41**

23 Display traffic statistics **41**
23.1 Daily or monthly total traffic snapshots 41
23.2 Day-by-day or month-by-month traffic statistics 43