The European Traffic Observatory Measurement Infraestructure (ETOMIC)

E. Magaña, D. Morató, M. Izal, J. Aracil, F. Naranjo, F. Astiz and U. Alonso
Universidad Pública de Navarra, 31006 Pamplona, Spain
{eduardo.magana,daniel.morato,mikel.izal,javier.aracil,ulisses.alonso}@unavarra.es

I. Csabai, P. Haga, G. Simon, J. Steger and G. Vattay
Collegium Budapest, H-1014 Budapest, Szentharomsag u. 2.
{csabai,haga,simon,steger,vattay}@colbud.hu

Abstract— **The European Traffic Observatory is a European Union VI Framework Program sponsored effort, within the Integrated Project EVERGROW, that aims at providing an paneuropean traffic measurement infrastructure with high-precision, GPS-synchronized monitoring nodes. This paper describes the system and node architectures, together with the management system.**

Keywords: Traffic measurement plattforms, active and passive measurement experiments

I. Introduction and system architecture

Internet traffic is growing at an extraodinary pace and, as a consequence, traffic control and forecasting are becoming fundamental issues for network operators. The European Union VI Framework Program is sponsoring a major research effort that aims at building a solid understanding of the Internet evolution until 2025, from a complex systems perspective. The EVERGROW [1] Integrated Project provides a multifaceted approach to forecasting the Internet evolution. The different subprojects that make up Evergrow cover a wide range of topics, from distributed systems and message passing to measuring Internet traffic in real time. Precisely, one of the subprojects is specifically focused on realizing a paneuropean measurement infrastructure, consisting of 50 measurement nodes which will be deployed at selected European locations. Such measurement infrastructure is called *European Traffic Observatory Measurement InfrastruCture* (ETOMIC).

ETOMIC is targeted to provide the scientific comunity with a measurement plattform that is i) fully open and reconfigurable and ii) extremely accurate (nanoseconds) and GPS-synchronized. First, ETOMIC has been designed to allow researchers to perform *any kind* of measurement experiments. To do so, researchers are provided with an interface from which software upload to the measurement nodes is possible. A choice of measurement scripts are also provided, that cover the most popular measurement techniques (packet pairs, etc). On the other hand, the researcher may also provide his/her own code for the experiments, that will be automatically compiled by the ETOMIC management system. Then, node reservation can be performed through the web-based user interface. Finally, the ETOMIC management kernel takes care of the software upload and experiment execution, in a fully automated fashion. On the other hand, ETOMIC is a high-precision infrastructure, due to the fact that Endace DAG cards [1][2] are incorporated to the nodes. Such cards are specifically designed to transmit packet trains with strict timing, in the range of nanoseconds. Actually, the DAG card transmit code has been specifically modified according to the ETOMIC requirements. Furthermore, a GPS is also incorporated to the measurement nodes, so that the whole measurement infrastructure is synchronized to the same reference clock. For example, a possible application is one-way delay measurements between nodes.

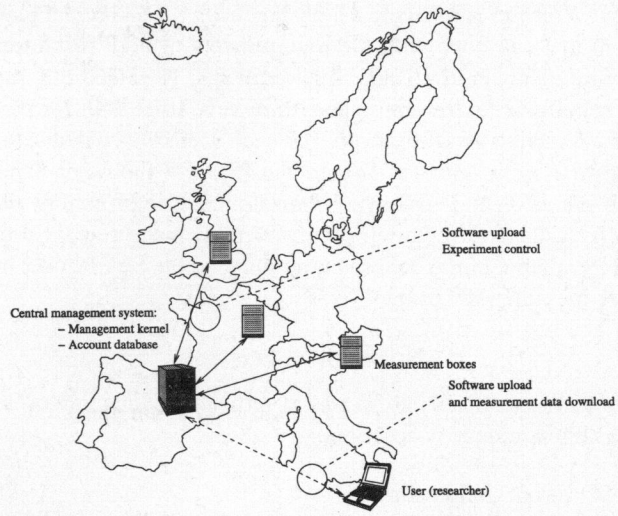

Fig. 1. ETOMIC

The system components are depicted in figure 1. A central management system is in charge of system control, comprising not only the scheduling and execution of measurements experiments, but also system monitoring (survivability) and configuration. The central management system is composed by a workstation and a traffic repository to which measurements can

[1] http://www.evergrow.org

0-7803-8836-4/04/$20.00 ©2004 IEEE.

be downloaded for subsequent processing. The software running in the management system is called the *management kernel*. On the other hand, researchers interact with the management system through the *user interface and account database*. ETOMIC provides an account to any registered researcher, so that he/she can upload software and download measurement results. Finally, the interface also serves for the system manager to enter configuration data about the measurement nodes and researcher accounts.

The paper is structured as follows. The remaining of the section is devoted to the state of the art. Section II presents the node architecture, together with the network interface card and GPS details. Section III presents the management kernel and section IV is devoted to the user interface. Finally, section V presents the conclusions that can be drawn from this paper.

A. State of the art

ETOMIC is not the only measurement infrastructure that is currently available in the state of the art, which includes the Surveyor, Felix, IPMA and AMP [3]. On the other hand, NIMI (National Internet Measurement Infrastructure) [4] is a large-scale measurement infrastructure with diverse administered sites. ETOMIC differs from the previous measurement infrastructures since it combines in the same plattform i) high-precision and GPS-synchronization and ii) capabilities to run all kinds of measurement experiments. ETOMIC incorporates dedicated measurement hardware and dedicated machines that are administered by a single management site, in contrast to NIMI. The aim is to create a high-precision measurement plattform that can cope with the requirements of high resolution measurements in high-speed environments. Note that the timing requirements for Gbps speeds are very strict both for passive and active measurements. For such scenarios, a dedicated hardware approach is often adopted [5]. On the other hand, high-precision and GPS-synchronization is also offered by the RIPE NCC infrastructure [6]. However, the measurement software available in the boxes is not uploaded by the end user on a per-measurement basis.

II. Node Architecture

In this section, the hardware and software architecture of the measurement node is described.

A. Hardware architecture

The measurement nodes are based on standard PC hardware. The motherboard is an Intel S875WP1-E with Intel Pentium 4 2.6 GHz processor, 1GB RAM, 200GB hard disk, two ethernet interfaces (one Gbit and another 10/100Mbit). The PC also includes watchdog functionalities. The operating system is based on a Debian GNU/Linux 3.0 (Woody) with enhanced kernel capabilities for low level network access without root privileges.

For the network monitoring interface an Endace DAG 3.6 GE is used, which is specifically designed for active and passive monitoring. Such cards do not use interrupts to signal packet arrivals to the kernel, and, thus, packets can be captured at gigabit speeds. Shared memory is used as a means to relay packets to the analysis program running in user space, in such a way that interrupts are avoided. Furthermore, the GPS reference signal is used to timestamp the incoming packets with high resolution. On the other hand, instead of having the kernel timestamp the arriving packets, timestamping is performed and as soon as the packet arrives by the card itself, since the GPS is directly connected to the card. As a result, no kernel induced jitter is present in the packet timestamp. On the other hand, DAG cards provide advanced capabilities for transmission: a burst composed by several packets can be transmitted with precise interpacket timing (nanoseconds). To this end, the DAG card has an internal high resolution clock and uses the GPS reference too.

The GPS reference is based on a Garmin GPS 35 HVS. It is a low-cost water-resistant GPS receiver that is used to synchronize the measurement nodes. Specifically, the GPS provides a PPS signal (pulse per second) directly to the DAG card. The resulting accuracy is 100 nanoseconds in the packet timestamps and interframe generation intervals.

An RS232-RS422 converter has been designed in order to bridge the mismatch between GPS receiver and PC ports. Furthermore, this converter transforms the RS232 signal from the GPS to a RS422 signal that allows larger distances between the GPS and the PC. This is a very important issue because the GPS receiver has to be placed outdoors with direct sky visibility, mainly on the roof. The converter, DAG card and GPS unit are displayed in figure 2.

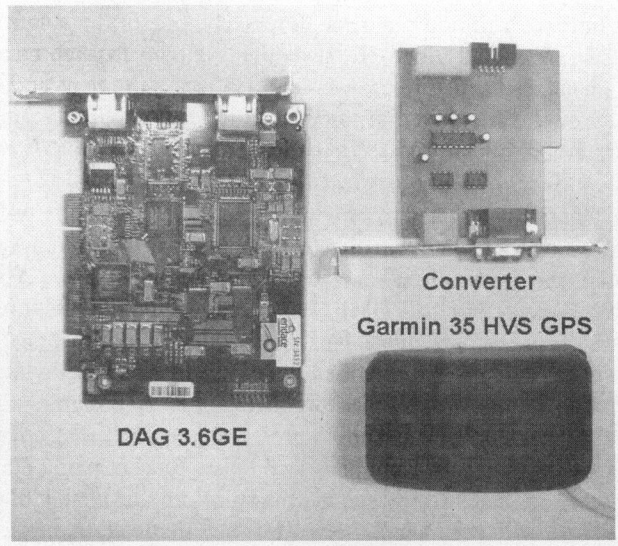

Fig. 2. DAG 3.6GE, GPS and converter

B. Measurement node software and API

The measurement node software is implemented over GNU Linux operating system. Measurement nodes are always listening for commands from the management kernel. On the other hand, the nodes are stateless and their functionality is reduced to a minimum in order to improve the robustness of the whole plattform in case of a single node failure. Accordingly, the node software only provides transfer resume functionality, and some other basic management tasks.

An API (Application Program Interface) has been developed for the DAG card, providing a high level interface that mimics the BSD socket interface programming style. In the future, we plan to extend the API with advanced capabilities such as traffic filtering inside the DAG card or even on-line calculation of basic traffic statistics.

III. Management kernel

The management kernel constitutes the core of the central management system. It is in charge of scheduling the experiments and keeping the corresponding results in the traffic repository. The software is implemented for GNU Linux operating system.

The researcher is expected to use the (web-based) interface in order to book several nodes during a certain time interval and in order to upload the measurement client and server. Such bookings are stored in the central database, that will be described in the next section. Note that the database contains all the relevant information and software about the measurement experiment. Thus, the management kernel is continuously checking the database for new measurement requests. Then, SSL (Secure Sockets Layer, RFC 2246) is used by the kernel to securely command the measurement nodes. Such nodes are totally stateless for failsafe reasons, and communications are always started by the server.

Once a new experiment has been defined and the deadline for execution approaches, the management kernel performs the following tasks:

- Software upload and measurement node configuration
- Experiment execution
- Results download

Such tasks are internal to the system and transparent to the researchers, that will only use the web interface to upload software to the central management system and retrieve measurements.

The *software upload and measurement node configuration* task is performed before the experiment starting time. In this task, the measurement programs are uploaded to the measurement nodes. Accompanying data files may also be uploaded that contain traces or measurement parameters. This allows to upload many different parameters for one single experiment. The data transfer has been enhanced with transfer resume functionalities in order to be able to manage very large data files. Once the files are uploaded, the measurement nodes are programmed with the starting and ending time for each of the executables that are going to be run for the experiment. This time scheduling is performed using the *atd* service in Linux. Several subtasks are performed for this task, such as quota reservation for the experiment, file upload and time programming for each of the measurement nodes.

During the *experiment execution* task, the management kernel is on standby until the end of the experiment. It only has to ensure that no other experiment is using the same measurement nodes in the same time interval. The aim is to completely isolate the measurement nodes so that the high-precision measurement hardware can be exclusively devoted to a single experiment.

Once the experiment is finished, the management kernel will *download* the resulting data files from each measurement node. The experiment files will be deleted at that time. Note that during the download phase the measurement node cannot be assigned to another measurement experiment, since the measurement and the download would interfere with each other. An estimation of the download time for an experiment is performed by the management kernel and stored in the database as part of the reserved time interval for such experiment. However, since download time is highly dependent on the network connectivity, transfer resume capabilities are necessary. Thus, the download phase may be interrupted by the management kernel and resumed in the future, depending on the number of pending measurement requests.

Besides the experiment management tasks described above, maintenance tasks are also performed by the management kernel. Such tasks have low priority and they take place when no other task is scheduled for a given measurement node. An example of a maintenance task is to configure a newly incorporated measurement node to the network. To do so, the public key and basic software configuration is uploaded to the node. Finally, other maintenance tasks are related to keepalive functions such as connectivity checking and bandwidth estimation.

In order for the management kernel to do the scheduling of experiments and maintenance tasks a calendar will be used, as shown in figure 3. There is a calendar per measurement node that enforces resource isolation by not allowing concurrent experiments to run over the same node. The gaps between experiments are filled with maintenance tasks. Figure 3 shows an example of scheduling for experiments (2,3) and (4,5), that are running on non-overlapping sets of measurement nodes.

IV. Database and user interface

ETOMIC provides an interface for researchers and administrators which fullfils the different requirements they may have. The former require capabilities to define new experiments, to reserve the measurement nodes and to download results from the management system. The latter require functionalities to manage users, measurement nodes, software and experiments.

It turns out that the database and management kernel (section III) are strongly interdependent. Note that the database

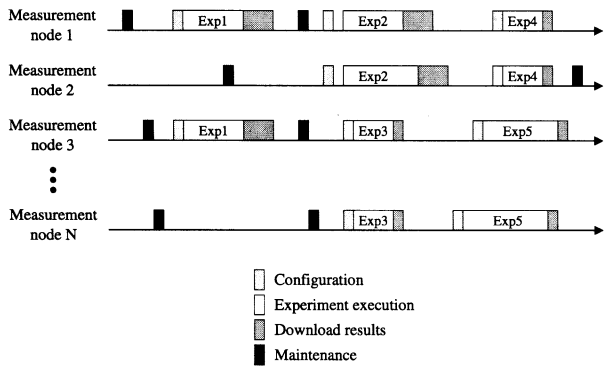

Fig. 3. Management kernel internal data structure

provides all the necessary software and data to keep track of measurement experiments. For example, the measurement nodes data (IP address for instance) is stored in the database. In addition, the database is used to store the experiment results, the experiment status and the measurement node status.

The strong usability requirements imposed by the researchers and administrators represented a significant challenge for the interface design. Solutions based on providing console access (*ssh* or *telnet*) to the nodes during the experiment were not deemed adequate, in order not to burden the researcher with launching the software at each node and with scheduling the tasks during the experiment. Instead, users are provided with a graphical interface for setting up the experiment beforehand. Then, the management kernel is in charge of experiment execution.

The interface is based on the ubiquitous Web service. Using HTML4, Javascript, PHP4 and Apache Web server (using a Secure Sockets Layer) we were able to provide interface capabilities which are close to those offered by end-user applications. The aim is to facilitate the definition of the experiments as much as possible. This targeted simplicity and ease of use posed more stringent challenges on the researchers interface. On the other hand, the ETOMIC administrators interface was easier to define.

Since the ETOMIC interface is organized around the different system users roles we adopt this approach in the following subsections. An snapshot of the interface is show in 4.

A. The researcher

ETOMIC will mostly be used by researchers willing to define measurement experiments. This kind of user is offered the following functionalities through the interface:

- **Adding a new program**: In order to run simple experiments, several scripts for sending and receiving IP packets with the DAG card high-precision timestamping are provided. However, not only *off-the-shelf* programs are offered but also an API for creating new programs for whatever experiment the researcher may have in mind. Hence, the interface offers capabilities to upload new programs, as source code. The programs are automatically built and made available as soon as the compilation process is over. Shell scripts can also be uploaded in such a way the researcher can easily create *batch-works* based on already existing programs.
- **Uploading data files**: User programs may require configuration parameters or input data files. Such files can be uploaded at any time.
- **Creating an** *experiment-Bundle*: The definition of a new experiment comprises several measurement nodes and several programs running on each node, with different starting and ending times. This is called an "Experiment Bundle". First, the monitoring nodes that make up the experiment must be selected. Then, the programs that run on each node are chosen. A choice of programs is offered, from the scripts mentioned before to the traditional unix network tools (ping, etc), besides the source code written by the user. As a last step, for each of the programs that make up the experiment bundle an starting and ending time can be provided *relative to the starting time of the experiment*. Thus, full flexibility is provided for defining asynchronous execution of several programs within the same experiment. It is noteworthy that the programs that make up the bundle can be reused for other experiment bundles, since they are permanently stored in the database.
- **Booking ETOMIC time**: Once the bundle definition is performed, the researcher should define the starting time for the experiment. To do so, the interface shows the availability intervals per node, together with a best-fit suggestion. However, it is up to the user to select any available time interval, depending on the specific experiment requirements (day time versus night time, for example).
- **Obtaining results**: The management kernel is in charge of downloading the data files created or modified during the experiment. Once the download is over, the researcher will have access to such files through the interface. It is expected that he/she will be willing to download them to his local computer, in order to process the results offline. Hence, functionalities to download a variable number of files, in compressed format, are offered.

B. The administrators

Different administrator types are envisioned for the ETOMIC system, which require different functionalities. While ETOMIC is centrally managed, there are local administrators per measurement node or set of nodes. That way, the management of a large number of measurement nodes is simplified, since node configuration can be performed locally (e.g. changing the IP address).

The measurement node local administrator will be the most common. The available parameters for such local administrator include the available disk space, the number of network interfaces, etc. On the other hand, the node administrator can also request downtime for maintenance.

Two ETOMIC system-wide administrators are defined. The ETOMIC account administrator has privileges to open new

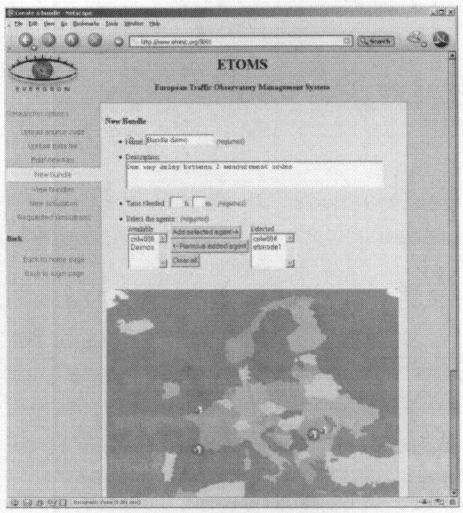

Fig. 4. User interface

accounts for researchers and to configure account parameters (maximum number of experiments, maximum quota in the database or in the measurement nodes, etc.). Finally, the ETOMIC superuser has privileges to access and modify low-level system information and to abort any experiment programmed or in progress. All these features are provided through a user-friendly web interface.

V. Conclusions

In this paper, the European Union VI Framework program sponsored European Traffic Observatory has been described. The traffic observatory will provide a high-precision fully opened measurement plattform for the benefit of traffic measurement researchers. Both the measurement nodes high-precision hardware and the user-friendly management system represent attractive capabilities that will surely spur the widespread use of the traffic observatory.

References

[1] Endace Measurement Systems, "http://www.endace.com."
[2] DAG Project at the Waikato University, "http://dag.cs.waikato.ac.nz/."
[3] Active Measurement Project, "http://amp.nlanr.net/."
[4] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale internet measurement," *IEEE Communications Magazine*, vol. 36, no. 8, pp. 48–54, 1998.
[5] J. Aracil, D. Morató, and M. Izal, "Analysis of Internet Services in IP over ATM networks," *IEEE Communications Magazine*, vol. 37, no. 12, pp. 92–97, December 1999.
[6] RIPE Network Coordination Center, "http://amp.ripe.net/."