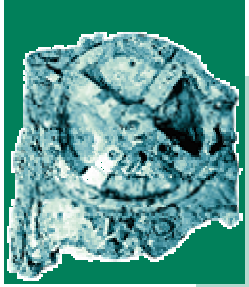
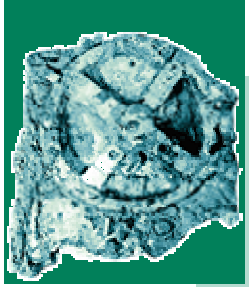


Issues for the performance monitoring of an open source H.323 implementation ported to IPv6-enabled networks with QoS characteristics

Authors: Ch. Bouras, A. Gkamas, A. Karaliotas, D.Primpas, K. Stamos

Research Academic Computer Technology Institute, Greece
University of Patras, Computer Engineering and Informatics Department, Greece

http://ru6.cti.gr



The Transition to IPv6

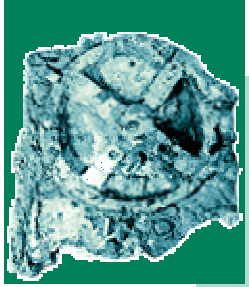
- The new version of IP, IPv6, constitutes an effort to overcome the inborn limitations of IPv4, in order for the new protocol be able to respond to the new needs as they shape today in the Internet

- The transition phase is an obstacle and the main reason for the slow adoption of IPv6

- The vast majority of network applications in existence today presume the use of the IPv4 protocol, so a transition to IPv6 will have to be accompanied by the development of new applications and/or the modification of the existing ones, so that they can be used in IPv6 environments

	IPv4 server	**IPv6 server**
IPv4 client	Communicate using IPv4	Communicate using IPv4, server sees IPv4-mapped IPv6 address
IPv6 client	Can communicate if the IPv6 client uses an IPv4-mapped IPv6 address	Communicate using IPv6

Interoperability between IPv4 and IPv6 versions running on dual-stack hosts



The OpenH323 project

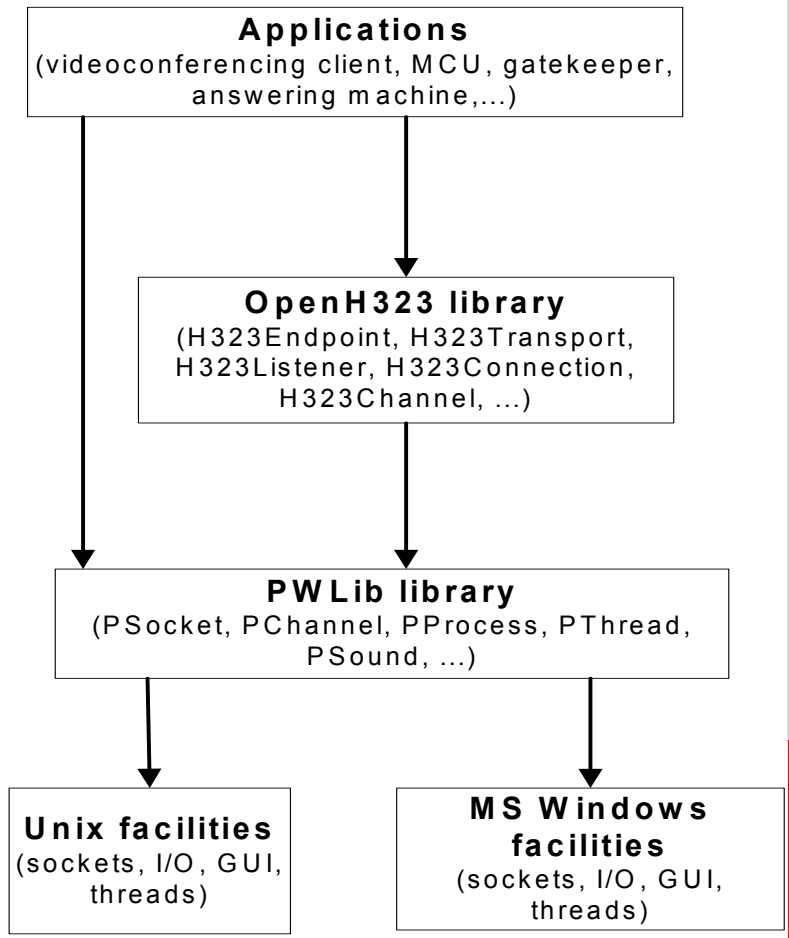
- H.323: A standard approved by ITU that defines how audiovisual conferencing data is transmitted across networks

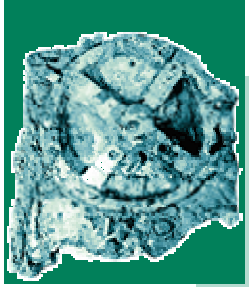
- OpenH323: open-source H.323 implementation

- PWLib: open-source library that encapsulates I/O, GUI, multi-threading and networking functionalities, and implements basic “container” classes. The goal is to support applications that can run both on Microsoft Windows and Unix systems

- OpenH323 and PWLib are comprised of 500,000 source code lines in over 400 classes (C++)

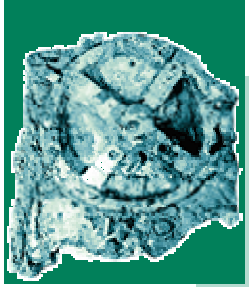
- The H.323 applications that have been developed on top of OpenH323 include: command line client, MCU, answering machine, gatekeeper, H.323 to PSTN and fax modem to T.38 gateways, and GnomeMeeting, a graphical H.323 client for Unix





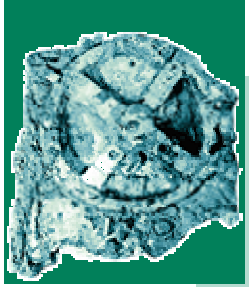
Methodology for porting procedure

1. Study and understand the source code, highlighting the points where a change in the program's logic is probably necessary
2. Parse the source code with an automatic tool like Microsoft's Checkv4.exe
3. Modify the source code lines reported by the automatic tool, which are probably going to be rather straightforward
4. Make any other modifications in more subtle places not reported by automatic tool
5. Test and debug the code, correcting any issues that arise
6. Verify completeness of porting effort
   - High-level testing: This testing strategy emphasizes on testing applications that use a wide range of functionality from the supporting libraries
   - Low-level testing: The opposite approach is to try and isolate specific classes and methods and try to test their behavior by using simple test applications
   - Comparative (back-to-back) testing: This strategy can be used when different versions of the same system are available. The two versions can be tested together and compared
   - We used a combination of the above techniques, with emphasis to comparative testing



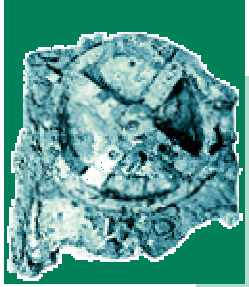
Automated Tools

- For the initial phases of the porting an automatic tool that parses the source code and reports the source code lines that contain IPv4-dependent code can prove very useful
- The relevant changes are probably rather straightforward, and can proceed in a mechanistic way
- The tool we chose was Checkv4.exe by Microsoft, which is offered as part of the experimental IPv6 stack for Windows 2000.
- Some of the tools of this kind available:
  - Microsoft's Checkv4 for MS Windows
  - Sun's Socket Scrubber for Solaris
  - Compaq's IPv6 Porting Assistant for Tru64 Unix
- Most of these tools are free and work with C/C++



Evaluation of an application ported to IPv6

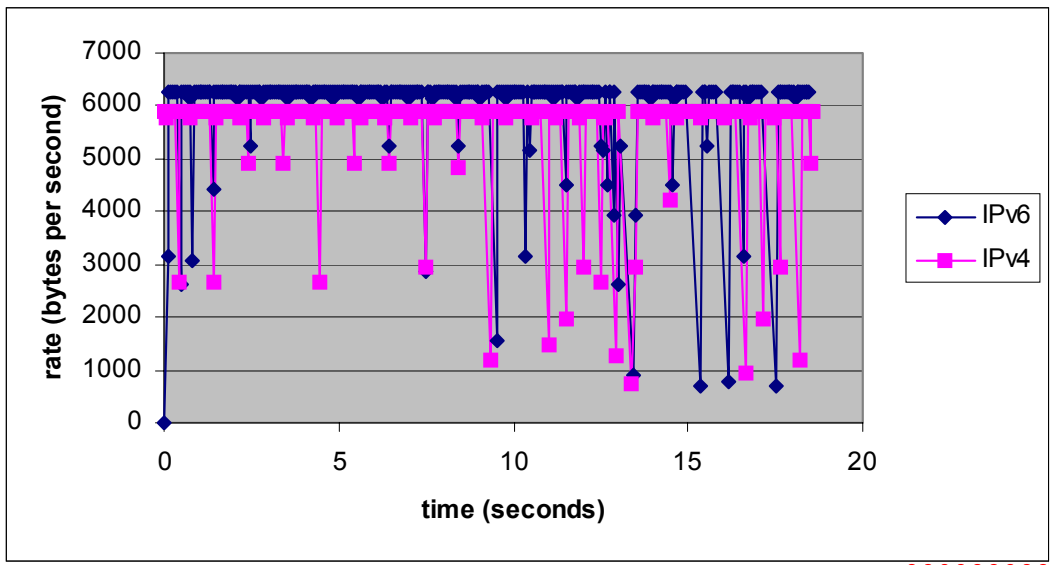
(1) Ability to work with IPv6: The application will have to seamlessly work with IPv6.
(2) IPv6 features involved: It is beneficiary if the application can make use of the new enhanced features of IPv6, e.g. flow id label.
(3) Dependency on IPv4 aspects: Should not depend e.g. on IPv4 DNS, IPv4 LDAP.
(4) IPv4-IPv6 simultaneous support: It is preferable for system administrators and developers to have a single version to maintain.
(5) RFC compatibility: e.g. with RFC 2732 for literal addresses in URLs.
(6) Dual-stack safe: The dual stack mechanism is going to be widely used for the foreseeable future, so an IPv6-enabled application has to be able to operate in a system with dual stack.
(7) Multiple DNS: The DNS mechanism plays an important role for the communication between IPv4 and IPv6 hosts. In most cases it still has to be able to differentiate and handle each returned address by the DNS resolver properly.
(8) Multicast, anycast: Apart from the unicast method of communication, IPv6 also makes use of multicast and introduces anycast. The multicast mechanism is especially useful for real-time applications.

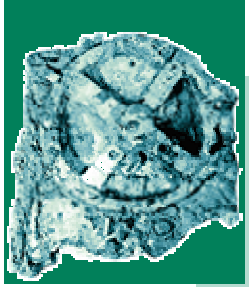


Initial experimentation

- We performed some initial experimentation with a ported OpenH323 application, and compared the bandwidth rate required for an IPv4 and an IPv6 call. The application was a simple H.323 VoIP application using the G.711 A-Law codec for voice transmission. The tests were performed on a 10 Mbps Ethernet LAN, and there was no competition.

- IPv6 maintains a data rate at around 50 Kbps, while IPv4 maintains a data rate at around 47 Kbps, almost 7% lower.





Planned Experiments

- We plan to deploy our detailed experiments in a real environment in 2 stages. First, we are going to use the local experimental network testbed of CTI, and at a second stage we are going to use the infrastructure of 6NET, the largest European project on IPv6.

- Testing scenarios using best-effort: In these scenarios no QoS mechanism will be used. Apart from the OpenH323 traffic, we will use artificially generated traffic using a traffic generator.

- Testing scenarios using QoS on gold service: This time the OpenH323 traffic will be treated using the gold service. The basic idea of gold service is that all the in profile packets must arrive to the destination, with the minimum possible delay and jitter. The out of profile packets may arrive or not, depending on the network's status. This service can guarantee delay, jitter and packet loss.

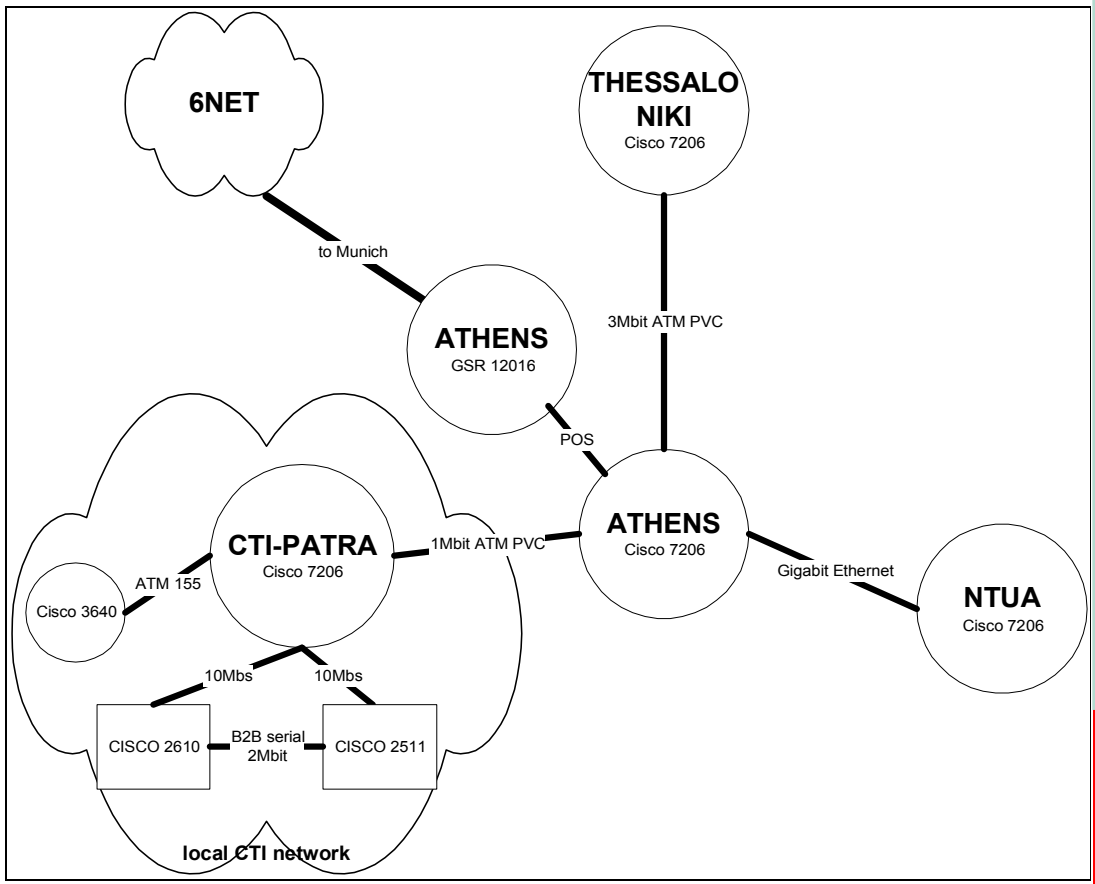
- Testing scenarios using QoS with more than one class of service: The QoS tests will then be continued, adding one more class of service. In particular, the additional class(es) will have worst treatment than the gold service but better than best effort. The testing scenarios will be the same as above, but an additional traffic flow will be produced for that service.

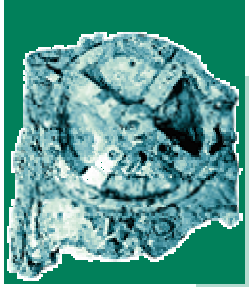
Planned Experiments (continued)

- For each scenario, we are going to compare the performance of the IPv6 OpenH323 application vs. the IPv4 application, in order to investigate the way the new Internet Protocol affects the network.

- In order to free ourselves from issues specific to one IPv6 or IPv4 implementation, we are going to use at least 2 different stacks for each scenario, Windows 2000 and Linux.

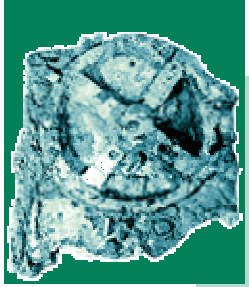
- The parameters we are going to measure for each experiment are packet loss, delay, jitter, throughput and bandwidth sharing with competing flows.





Conclusions – Future Work

- The number of required IP addresses for the near future (in order to address all the hosts and embedded systems) is expected to rise to the order of billions. IPv6 can provide this huge number of IP addresses, in addition to providing benefits like auto-configuration capabilities and QoS support. The larger IPv6 header introduces nevertheless some additional overhead, which is more significant for low-rate applications.
- Our future work includes the extension of the mentioned experiments to greater actual IPv6 networks, and in particular using the experimental IPv6 network of the 6NET project.
- Moreover, we intend to investigate possible benefits in the area of Quality of Service (QoS) by using the Traffic Class and Flow Label fields in the IPv6 header, and the benefits in the area of security by using the Authentication Header and the IP Encapsulating Security Payload (ESP). In the area of configuring the QoS mechanisms, we intend to further experiment and investigate the proper configuration parameters in order to optimize the performance of various traffic classes.



Email info

Christos Bouras (bouras@cti.gr)

Apostolos Gkamas (gkamas@cti.gr)

Anastasios Karaliotas (karaliot@cti.gr)

Dimitris Primpas (primpas@cti.gr)

Kostas Stamos (stamos@cti.gr)