

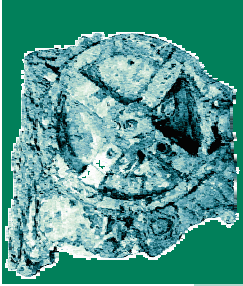
Performance Evaluation of an IPv6-capable H323 Application

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

Research Academic Computer Technology Institute,
Greece

University of Patras, Computer Engineering and
Informatics Department, Greece

<http://ru6.cti.gr>



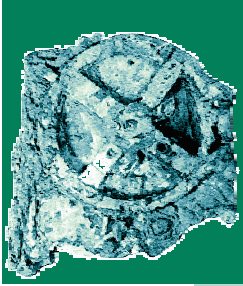
The Transition to IPv6



- 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

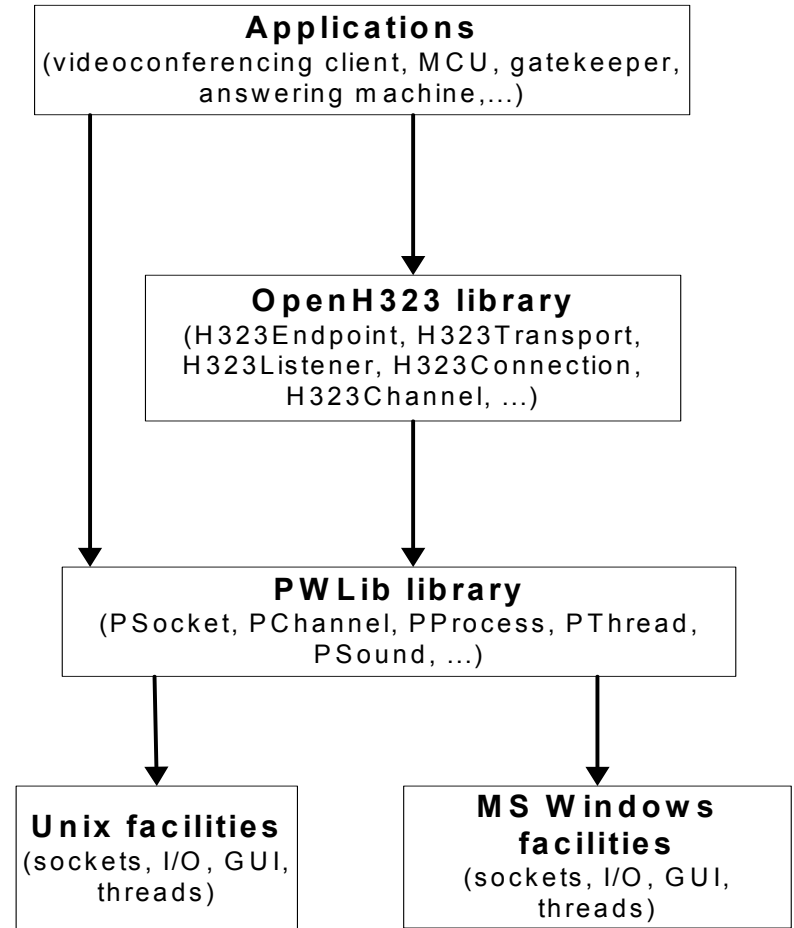
Interoperability between IPv4 and IPv6 versions running on dual-stack hosts

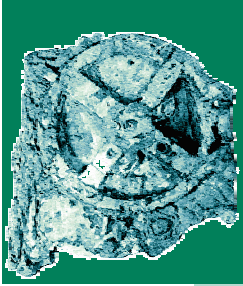


The OpenH323 project



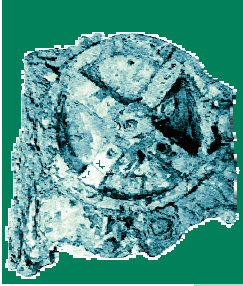
- H.323: A standard approved by ITU that defines how audiovisual conferencing data is transmitted across networks
- OpenH323: open-source H.323 implementation
- PWLib: open-source library that encapsulates I/O, GUI, multi-threading and networking functionalities, and implements basic “container” classes. 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

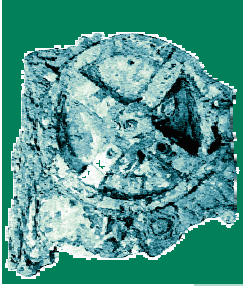
1. Study and understand the source code, highlighting the points where a change in the program's logic is probably necessary
2. Parse the source code with an automatic tool like Microsoft's Checkv4.exe
3. Modify the source code lines reported by the automatic tool, which are probably going to be rather 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



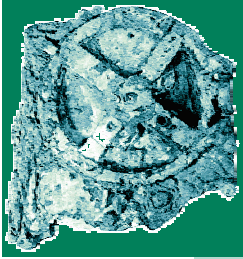
- For the initial phases of the porting an automatic tool that parses the source code and reports the source code lines that contain IPv4-dependent code can prove very useful
- The relevant changes are probably rather straightforward, and can proceed in a mechanistic way
- The tool we chose was Checkv4.exe 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++



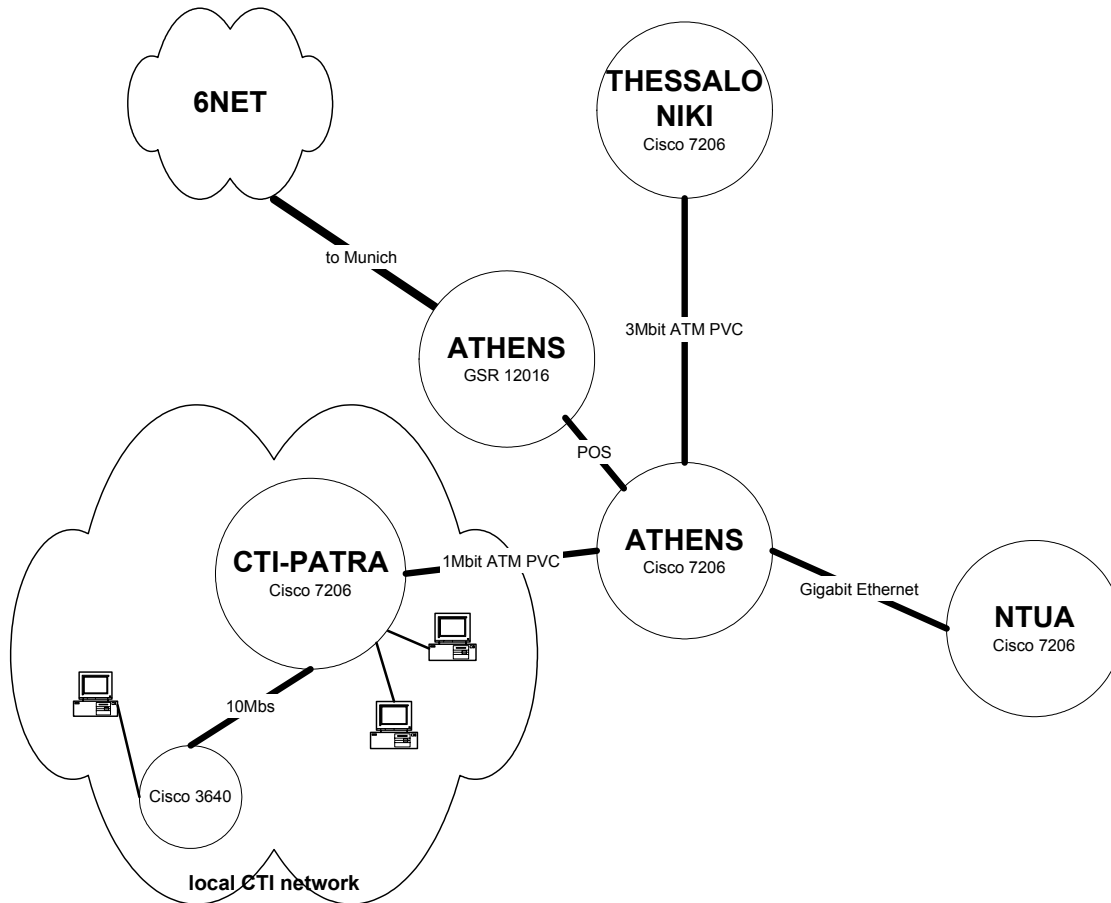
Evaluation of an application ported to IPv6



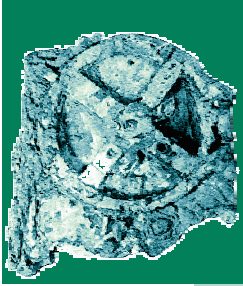
- (1) Ability to work with IPv6: The application will have to seamlessly work with IPv6.
- (2) IPv6 features involved: It is beneficiary if the application can make use of the new enhanced features of IPv6, e.g. flow id label.
- (3) Dependency on IPv4 aspects: Should not depend e.g. on IPv4 DNS, IPv4 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.



Testbed used for experiments



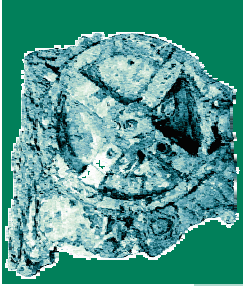
- Internal CTI network is part of pan-European 6NET network
- Communication from one endpoint to the other had to pass through 2 hops
- Bottleneck link: 10 Mbps



Experiments setup



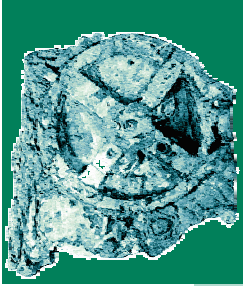
- Background traffic: Iperf tool, capable of producing TCP/UDP traffic in both IPv4 and IPv6
- For transmission/reception data:
 - RTP/RTCP feedback from the OpenH323 library
 - Ethereal and Sniff'Em network monitoring tools
- Transmission of video data: OpenH323 built-in H.261 codec with CIF resolution
- Audio transmission: G.711 (muLaw variation), PCM scheme at 64Kbps.
- Applications used
 - OpenPhone GUI client
 - OpenMCU implementation of a software MCU
 - OpenWAV application for transmitting pre-recorded audio files.



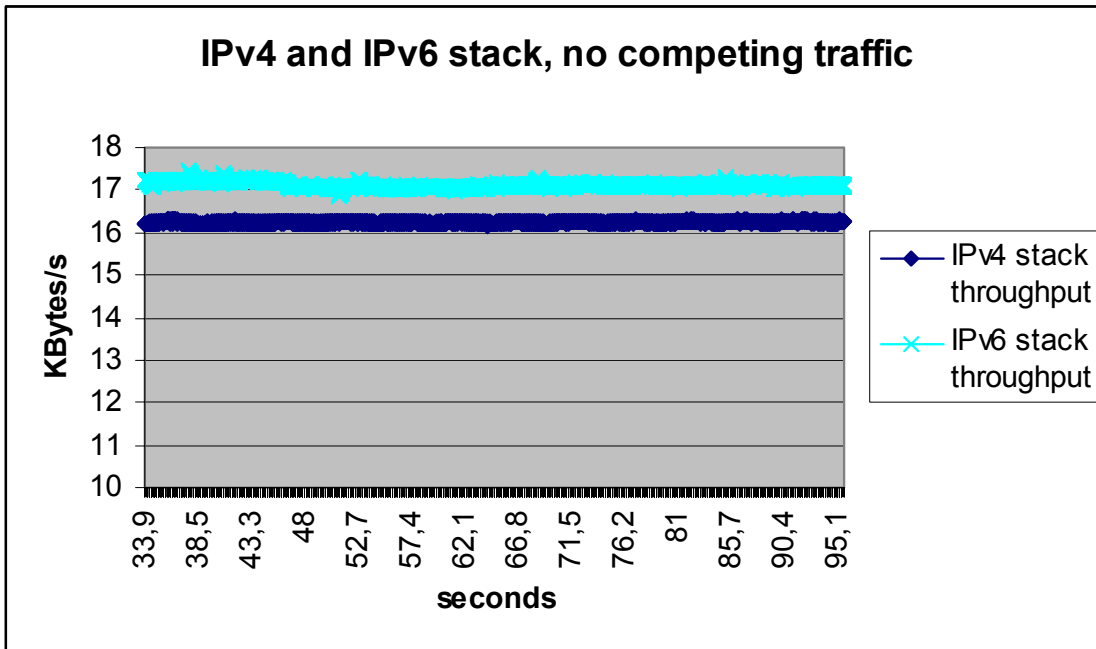
Purpose



- evaluate the IPv6 version of the OpenH323 library, and particularly in comparison with the IPv4 version
- observe how both versions behave when a link in the transmission path is congested because of the simultaneous transmission of other traffic
- test with the competing traffic being both UDP and TCP, because of the different behaviour characteristics of the two transport protocols, and the different impact that they have on the rest of the applications that use the same network links.

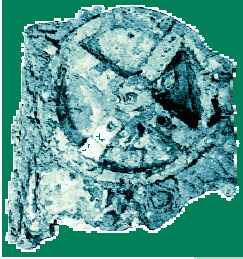


Experiment 1

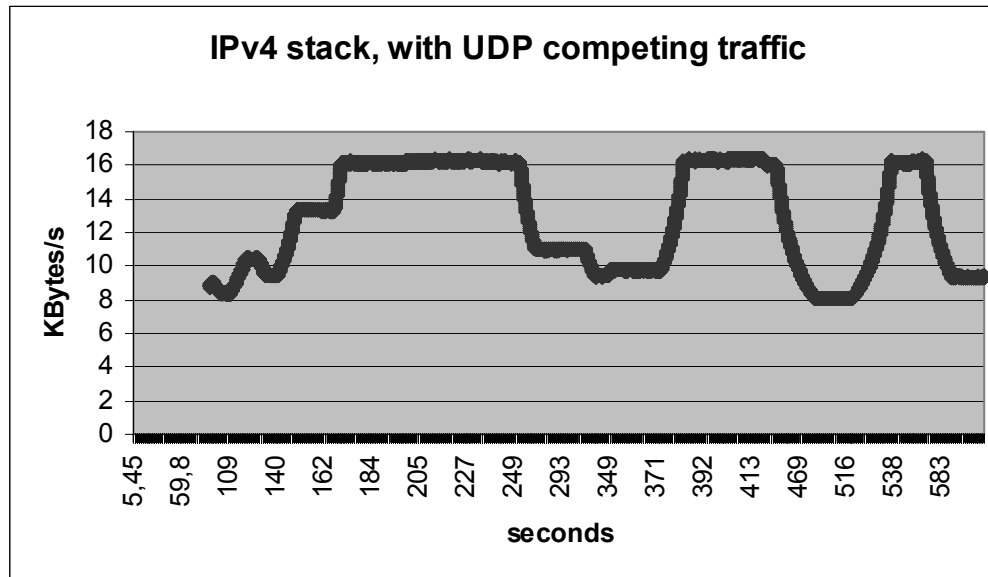


-7% higher transmission rate for IPv6

-The Data-Link layer was carrying 294-byte packets in the case of IPv4, and 314-byte packets in the case of IPv6. The standard IPv6 header is 20 bytes larger than the standard IPv4 header, which produces the 7% overhead.

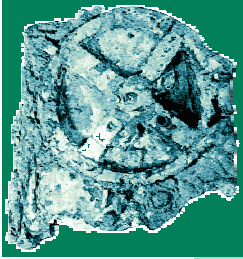


Experiment 2

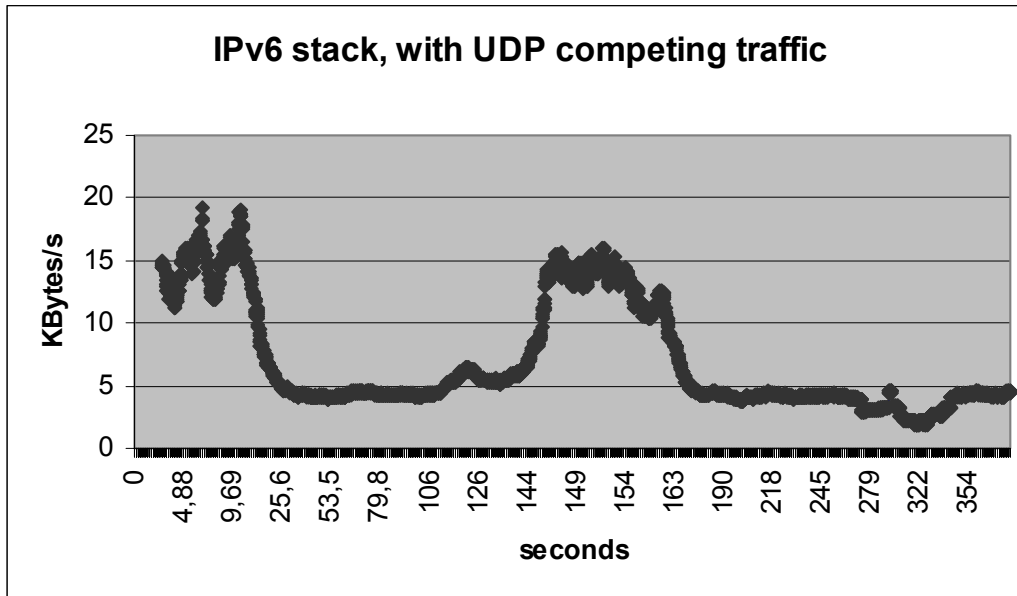


-the competing traffic reduced the transmission rate of the H.323 traffic, and also the quality of the video received at the other endpoint

-reduction not constant: associated with the fact that the H.323 traffic was relatively small compared to the artificially generated UDP traffic → minor variations at the generated traffic (processor or network stack limitations) reflected more heavily at the H.323 traffic



Experiment 3

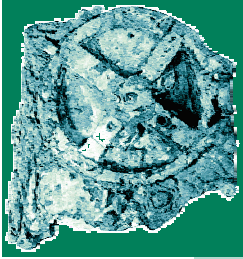


-results even more obvious → slightly bigger bandwidth consumption of IPv6

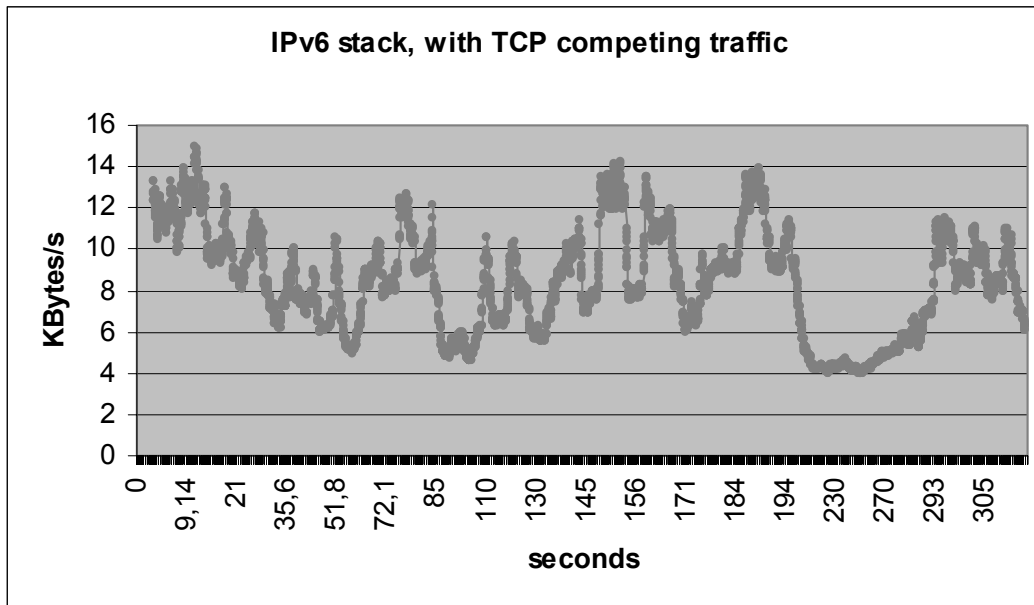
-reduced transmission rate and video quality

-100% more losses

-the Windows 2000 IPv6 stack that was used for transmission is experimental, and is therefore probably not as optimized as the Windows 2000 IPv4 stack



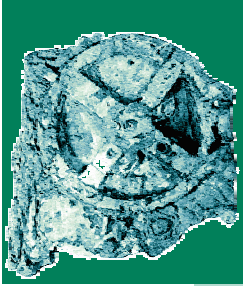
Experiment 4



-5 to 14 Kbps range, with variations both in the transmitting rate and the reception quality of the video image

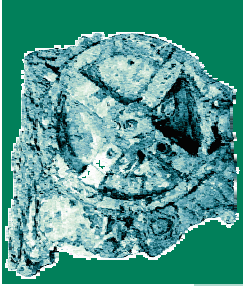
-TCP protocol slowly tries to regain bandwidth that it has lost due to congestion through an AIMD (Additive Increase, Multiplicative Decrease) algorithm

-RTCP reported a quite high 5,5% packet loss rate



Conclusions – Future Work

- Clearly demonstrated the need for QoS mechanisms that will be able to compensate for the loss of quality in a congested link (especially with UDP competing traffic)
- IPv4 and IPv6 versions behave roughly the same, although the slightly larger overhead of IPv6 due to the larger standard header makes the IPv6 version a bit more sensitive to congestion
- At the next stages, we plan to repeat and expand the above described trials outside the CTI internal network, with other parties in the Greek IPv6 network and with parties outside Greece in order to investigate the operation of OpenH323 platform for WAN communication with many more hops
- Comparison of those results with the ones presented.
- Investigate the behaviour and the outcome of the trials using QoS DiffServ mechanisms like the gold service, that benefit the OpenH323 traffic compared to the rest of the background traffic



Thank you



Email Info:

Christos Bouras (bouras@cti.gr)

Apostolos Gkamas (gkamas@cti.gr)

Dimitris Primpas (primpas@cti.gr)

Kostas Stamos (stamos@cti.gr)