# Integrating IPv6 in Java

Joaquín Salvachúa

Juan Quemada

Oscar Saavedra

DIT - UPM

# IPv6 and Applications

- There is a gap between applications and network facilities.
- Most applications works the same under IPv4 and IPv6.

- Java is a good language for writing network applications.
- GOAL: porting of actual java applications and design of new applications with IPv6.

# Applications

- Porting:
  - Same functionality than IPv4 application.
- Reengineering:
  - Redesign applications or subset.
  - Advanced API.
  - Still not.
  - Identify reengineering OOP patterns.
- Advanced new applications:
  - New facilities or services enabled by new features.
  - Can afford building from scratch.

# Java and Networking

- Language designed with networking support from start.
- Not an external library but part of the core language.
- Good OO design and use widely use of software patterns.

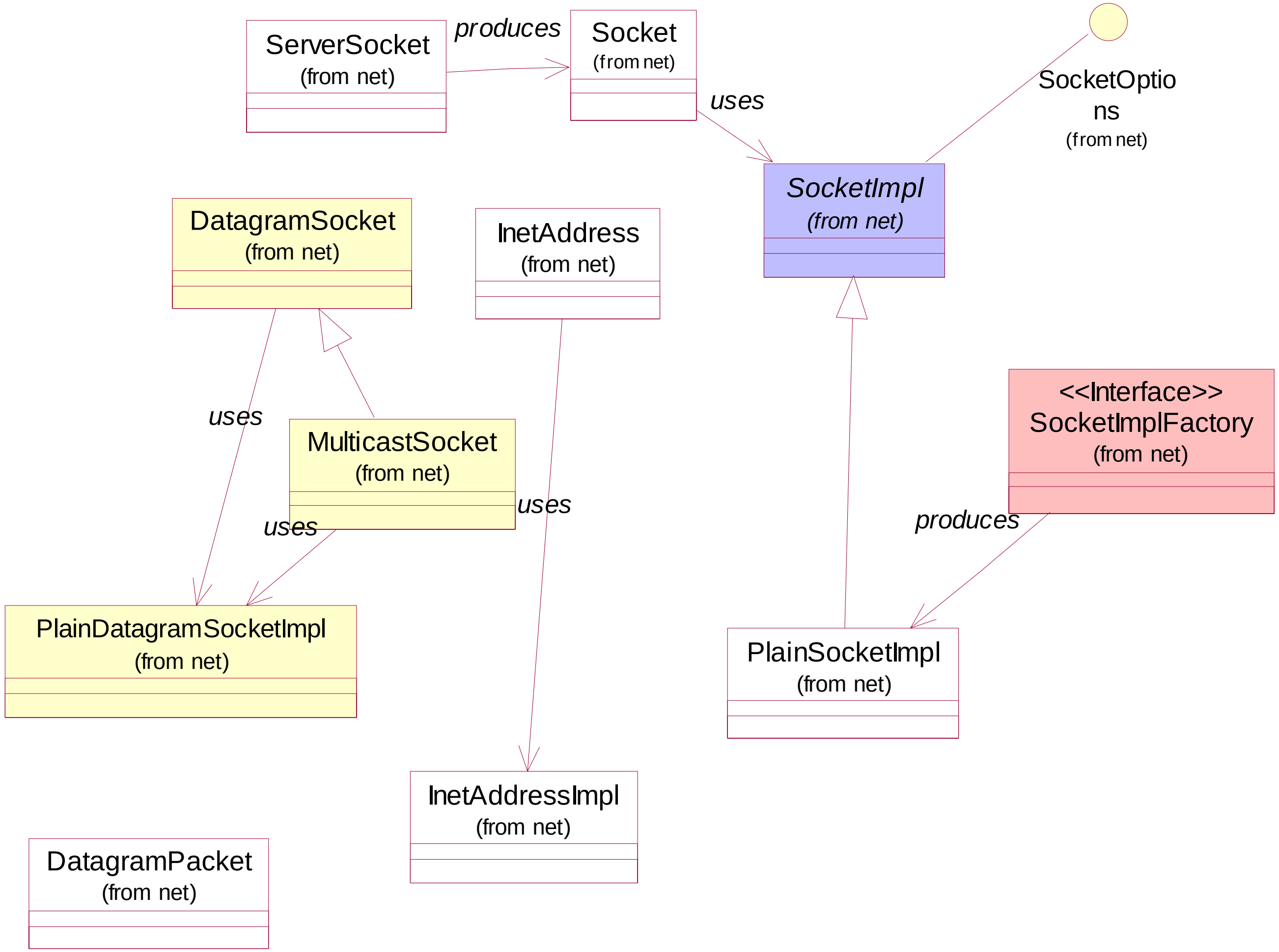
# Application Porting

- Most network applications uses only few facilities:
  - 90 % of application use TCP sockets.
  - Just connect and don´t care about the networking.
- This can be changed in java very easily.
- No changes in application only on runtime support.
- Only IP no new protocols right now.

# Basic Socket Facilities

- Specified on RFC 2133 and RFC 2553
- Basic API used by most of the applications.
- Porting:
  - Substitution of basic socket factories.
  - Extensible by use of OO design patterns.

# Socket Classes Diagram

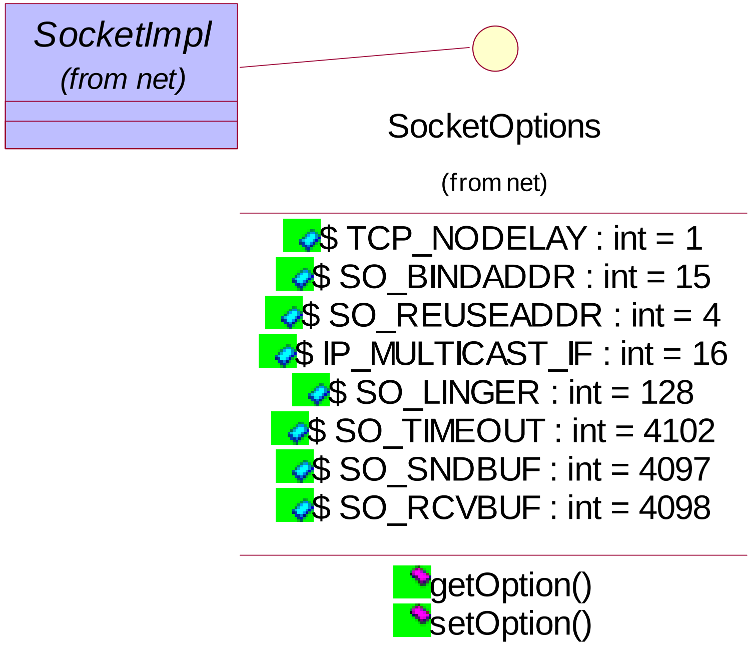


# Advanced Socket Facilities

- Specified on RFC 2292.
- Used by a low percentage of applications.

- Needs for a new “ioctl” like controls.
  - Projection from old commands.
    - No code needs to be change.
  - Support for new ones.
    - A new line of code when the socket is created.
    - If identify for all the sockets in an application only a configuration file.

# Modification Using SocketOptions



*SocketImpl*

*(from net)*

SocketOptions

(from net)

$ TCP_NODELAY : int = 1
$ SO_BINDADDR : int = 15
$ SO_REUSEADDR : int = 4
$ IP_MULTICAST_IF : int = 16
$ SO_LINGER : int = 128
$ SO_TIMEOUT : int = 4102
$ SO_SNDBUF : int = 4097
$ SO_RCVBUF : int = 4098

getOption()
setOption()

# Advanced New Facilities

- Security:
  - Added new flag for setting a secure connection.
  - Problem: feedback to application to really know that the connection is secure.
  - Comparations with SSL based solutions.
- Anycast:
  - Now load balancing: replicated servers.
  - Peer 2 peer solution without central server.

  - Both combined on a file sharing application.
  - Evolution of napster-like applications.

# QoS

- Up to now most done on the network side.
- Experimentation with on server QoS.
  - RSVPd in java implementation.
    - Also feedback for porting.
  - Diffserv on server.
- Now two different low level API based on the “C” ones (no OOP yet).

- Try to generalize to an abstract one.

# QoS (II)

- QoS transparent integration with a common API.
- Also working on a C++ version of QoS API.
- Thinking about with QoS definition language.

- Experiments with end to end congestion management (draft-ietf-ecm-cm-03.Txt).
- Experiments with JMF integration.

# Work in Progress

- Open Source approach.
- JANO project: Roman god of doorways
  - One face looking to the past (IPv4), and other to the future (IPv6).
  - http://sourceforge.net/projects/jano
  - http://www.dit.upm.es/~jsr/jano

- Collaboration with gipsy project
- Collaboration with SABA and LONG projects