

JRPC: Connecting Java Applications with Legacy ONC RPC Server

Martin Gergeleit

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Verteilte Systeme

gergeleit@ivs.cs.uni-magdeburg.de



Agenda

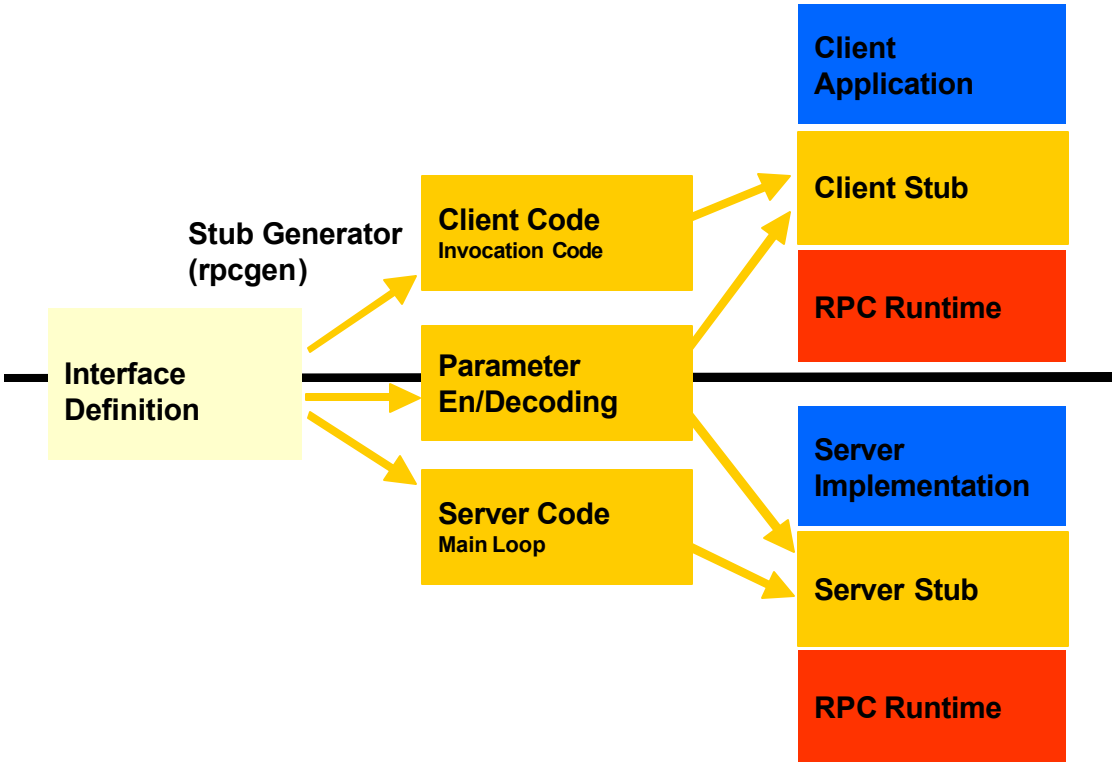
- Motivation
- RPC - Overview
- Design for Java
- Applet Security Issues
- Performance
- Summary



Why RPC for Java?

- **Typical Java application:**
  Internet-based front-end for existing services
  - requires middleware support
- **Existing middleware solutions**
  - Java RMI - connects Java objects with other Java objects
  - Java IDL - connects to CORBA compliant services
  - JDBC - connects to data-bases
- **ONC RPC (RFC 1831)**
  - More than ten years old (formerly called Sun RPC)
  - Standard on all Unix platforms
  - Many existing applications (e.g NFS, YP, NIS)
  - but, no Java binding so far ...

Building Client/Server Applications with RPC





Design for Java

- **XDR to Java type mapping**
  - Java has no
    - no enumeration type
    - no type name aliasing (typedef)
    - reference for simple types
- **Stub generation**
  - Java has
    - multithreading
  - Java has no
    - function pointers
- **Java-specific problems**
  - Applet security



XDR to Java Type Mapping

XDR Type	Java Type
(unsigned) long	long
variable length array	Java array
optional data (pointer *x)	reference to class x
struct x	class x
union x	class x (no overlay)
enum x	class x with int *value* and constants
typedef x y	class x with y *value*

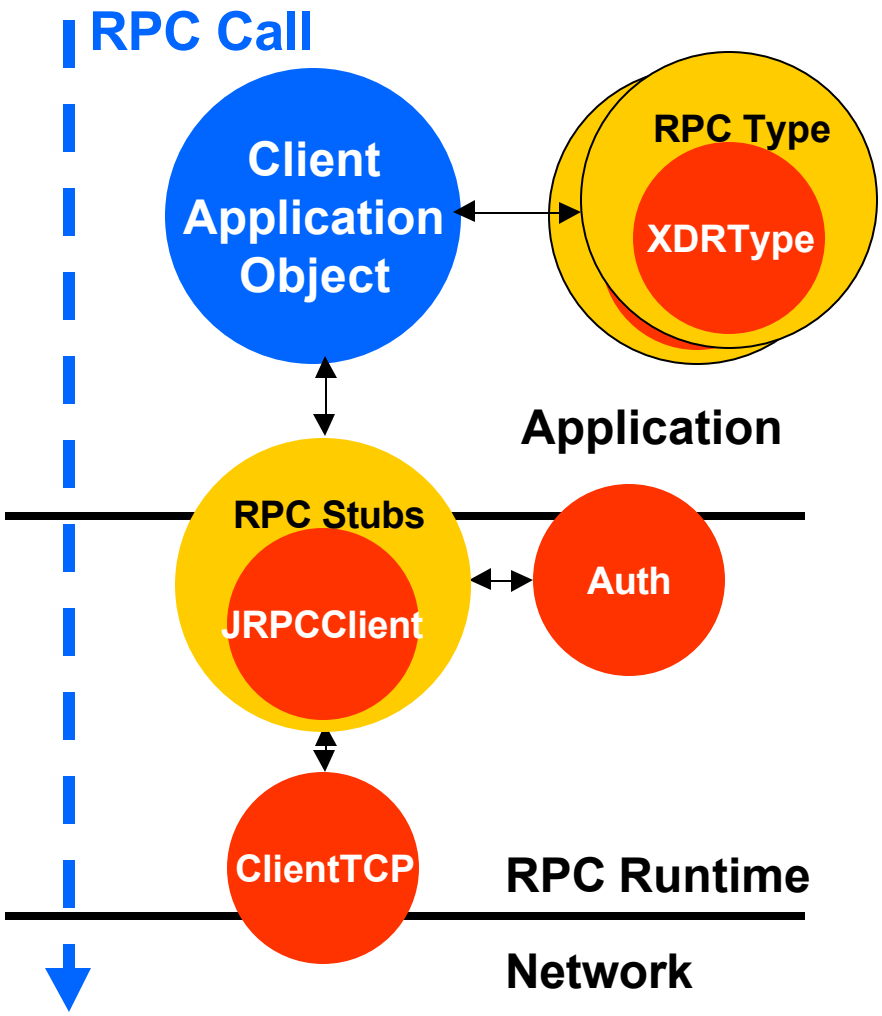


JRPC Class Hierarchy

- class java.lang.Object
  - class JRPC.ClientGeneric (implements java.io.Serializable)
    - class JRPC.ClientHTTP
    - class JRPC.ClientTCP
    - class JRPC.ClientUDP
  - class JRPC.JRPCClient (implements java.io.Serializable)
  - class JRPC.JRPCServer (implements java.lang.Runnable, java.io.Serializable)
    - class JRPC.Portmapper
    - class JRPC.RPCBind
  - class JRPC.Pmap (implements JRPC.XDRType)
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class JRPC.RPCError
        - class JRPC.RPCDecodeError
        - class JRPC.RPCServerError
        - class JRPC.RPCTimeoutError
  - class JRPC.XDRStream
    - class JRPC.XDRInputStream
    - class JRPC.XDROutputStream
  - interface JRPC.XDRType (extends java.io.Serializable)
  - class JRPC.XDRboolean (implements JRPC.XDRType)
  - class JRPC.XDRfloat (implements JRPC.XDRType)
  - class JRPC.XDRint (implements JRPC.XDRType)
  - class JRPC.XDRstring (implements JRPC.XDRType)

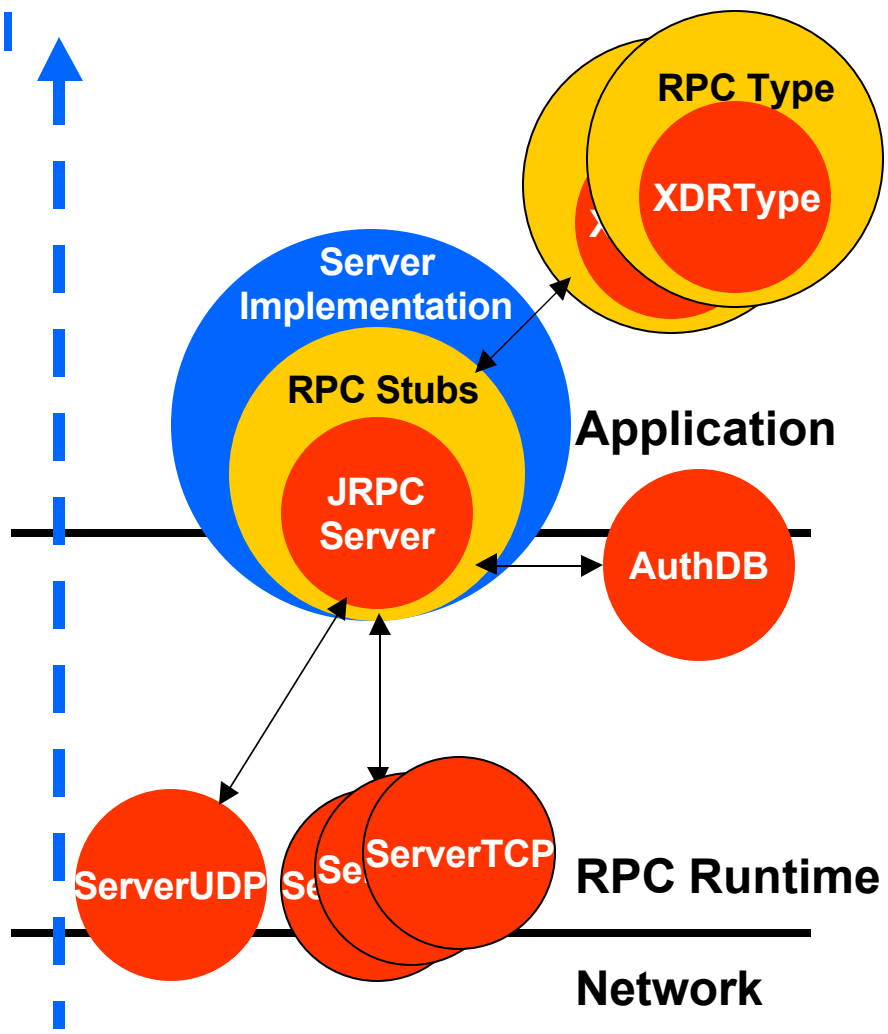
Client Stub Design

- **Parameter**
  - Base Class *XDRType*
  - *Jrpcgen* generates derived classes for RPC types
- **RPC Client Stub**
  - Base Class *JRPCClient*
  - *Jrpcgen* generates derived class for RPC stubs
  - Chooses protocol classes *ClientTCP*, *ClientUDP*, *ClientHTTP*, ...
- **Authentication**
  - Classes *Auth*, *AuthUnix*, or *AuthDES*
  - Chosen by the Application



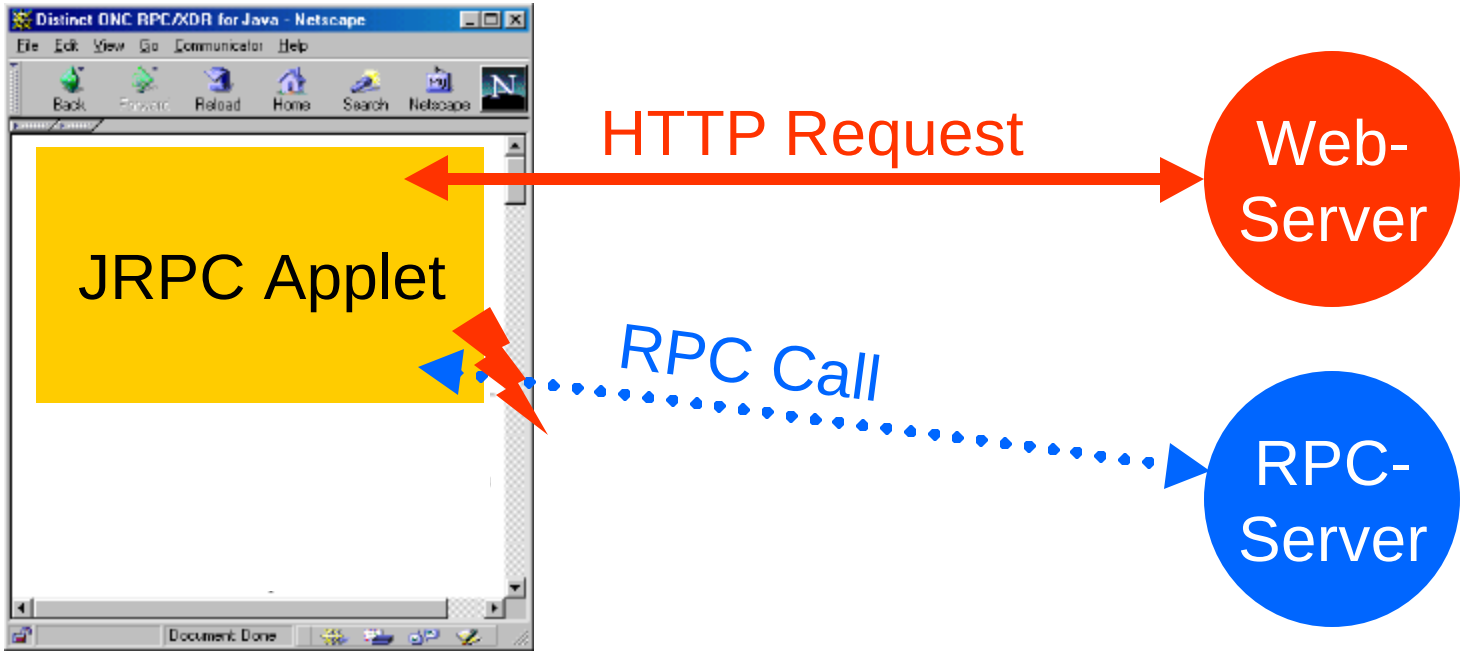
Server Stub Design

- **RPC Server Stubs**
  - Base Class *JRPCServer*
  - *Jrpcgen* generates derived abstract class for RPC stubs
  - Application derives class for server implementation
- **Protocol Classes**
  - Classses *ServerTCP*, *ServerUDP*
  - One object with one thread per active client
- **Authentication**
  - Class AuthDB implements callbacks for authentication checks



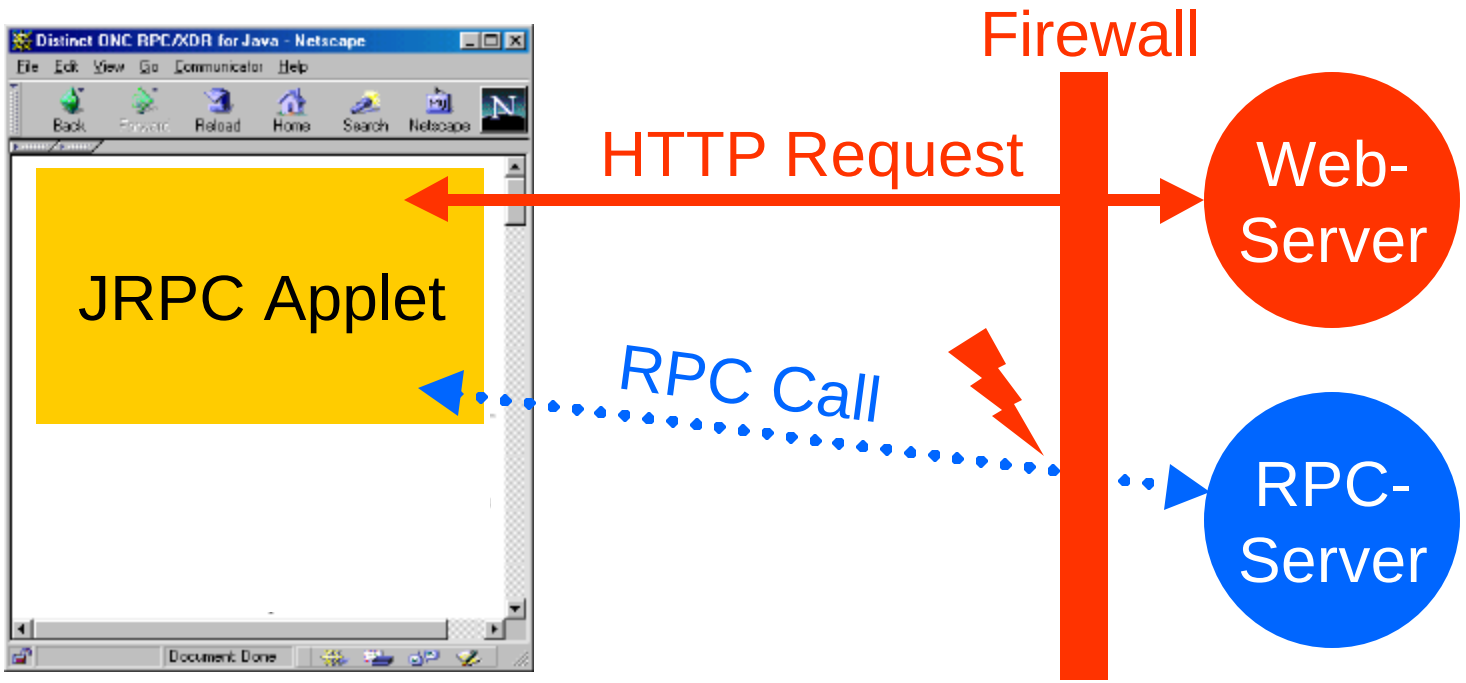
Problem: Applet Security

- Web- and Application-server typically on separate machines
- Untrusted code cannot access other server



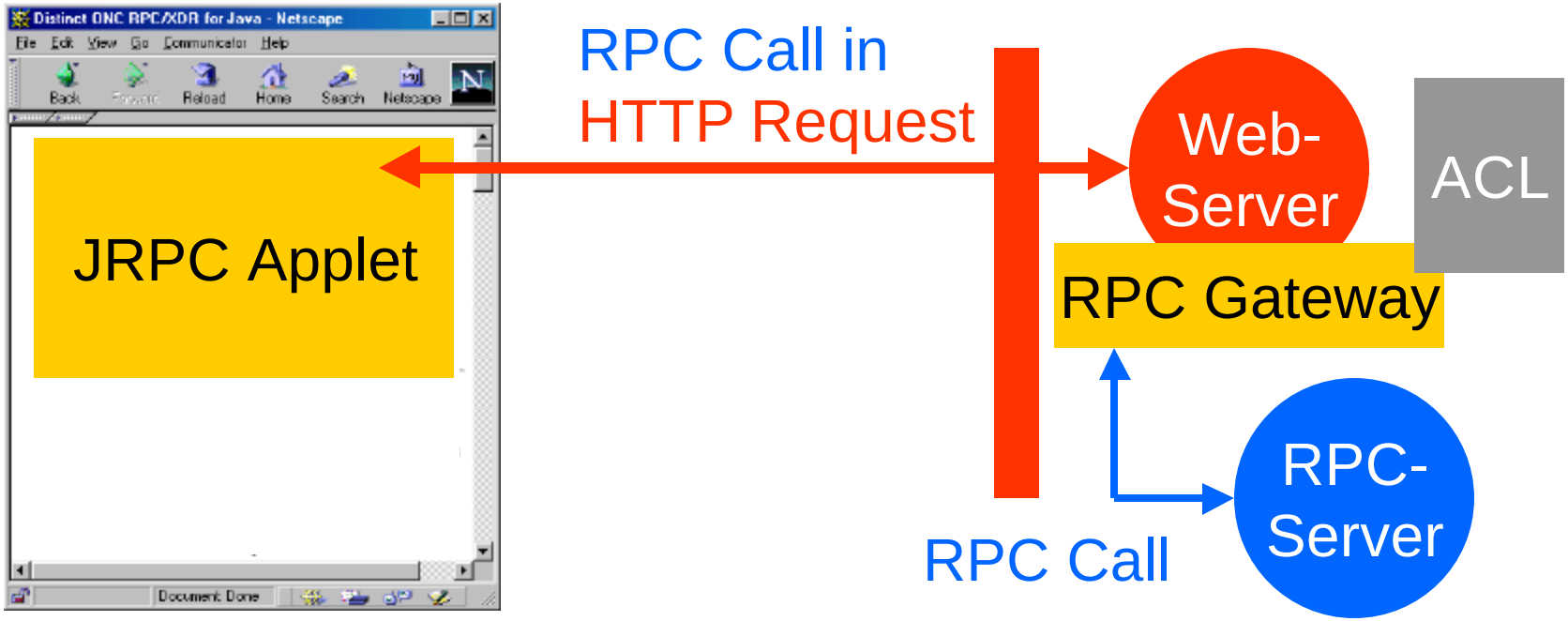
Problem: Firewalls

- ONC RPC was not designed for the Internet
- RPC uses arbitrary port-numbers
- Firewalls typically block the RPC Protocol



Solution: HTTP Tunneling

- JRPC encapsulates ONC RPC in HTTP
- Server extension (RPC Gateway) executes RPC
- Security: detailed access control list and log



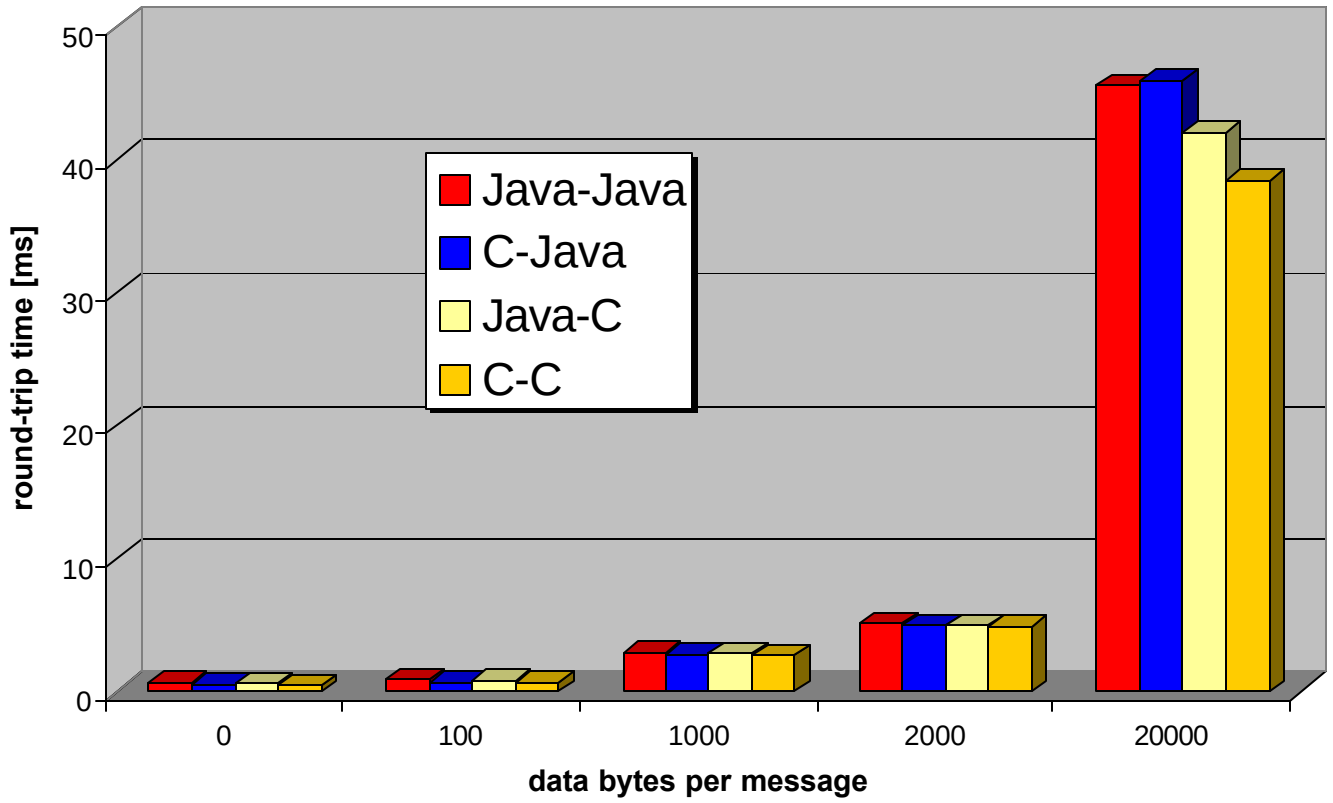


Components of JRPC

- **JRPC package**
  - Implements ONC RPC runtime
  - Supports TCP, UDP or HTTP transport
  - Pure Java, no native code
- **Jrpcgen compiler**
  - Translates XDR into Java
  - Generates client and server stubs
  - Implemented in Java
- **RPCGW server extension**
  - Used for HTTP-tunneling
  - CGI (C or Perl) or Servlet

Performance

- Comparison with ONC RPC implementation in C
- Java-Client/C-Server only 10-20% slower





JRPC Summary

- The first toolkit that fully bridges the gap between Java and ONC RPC
- Supports HTTP-Tunneling for use in Applets and other Internet-Applications
- Performance encourages migration from C to Java
- Transferred into a commercial version (certified “100% Pure Java”)
- Full-featured demo version included in the proceedings