Distributed Object Systems

Overview

- Goals
  - object-based approach
  - hide communication details

- Advantages
  - more space
  - more CPU
  - redundancy
  - locality

Problems

- Coherency
  - ensure that same object definition is used

- Interoperability
  - serialization
  - type consistency
  - type mapping

- Object life-time
  - distributed garbage collection
Distributed Object Systems
Architecture

Client

Application
Proxy
Object Broker
Call Context
Message

Server

Stub
Object Broker
Impl.
Impl. Skeleton

IDL-Compiler
IDL
IDL-Compiler
Remote Procedure Invocation

Overview

- Problem
  - send structured information from A to B
  - A and B may have different memory layouts
  - “endianness”
  - How is 0x1234 (2 bytes) represented in memory?

**Big-Endian:** MSB before LSB
- IBM, Motorola, Sparc

**Little-Endian:** LSB before MSB
- VAX, Intel
Definitions

- **Serialization**
  - conversion of an object’s instance into a byte stream

- **Deserialization**
  - conversion of a stream of bytes into an object’s instance

- **Marshaling**
  - gathering and conversion (may require serialization) to an appropriate format of all relevant data, e.g. in a remote method call; includes details like name representation.
Remote Procedure Invocation
Protocol Overview

- **Protocols**
  - RPC + XDR (Sun)
    - RFC 1014, June 1987
    - RFC 1057, June 1988
  - IIOP / CORBA (OMG)
    - V2.0, February 1997
    - V3.0, August 2002
  - SOAP / XML (W3C)
    - V1.1, May 2000
  - ...

- **XDR Type System**
  - [unsigned] Integer (32-bit)
  - [unsigned] Hyper-Integer (64-bit)
  - Enumeration (unsigned int)
  - Boolean (Enum)
  - Float / Double (IEEE 32/64-bit)
  - Opaque
  - String
  - Array (fix + variable size)
  - Structure
  - Union
  - Void

big-endian representation
Remote Procedure Invocation
RPC Protocol

- Remote Procedure Call
- Marshalling of procedure parameters
- Message Format
- Authentication
- Naming

Client

PROCEDURE P(a, b, c)
- pack parameters
- send message to server
- await response
- unpack response

Server

Server
- unpack parameters
- find procedure
- invoke
- pack response
- send response

P(a, b, c)
Distributed Object Systems
Details

- References vs. Values
  - client receives **reference** to remote object
  - data **values** are copied to client for efficiency reasons
  - decide whether an object is sent as reference or a value
    - serializable (Java, .NET), valuETYPE (CORBA)
    - MarshalByRefObject (.NET), java/RMI/Remote (Java), default (CORBA)

- object creation
  - server creates objects
  - client creates objects
  - server can return references

- object instances
  - one object for all requests
  - one object for each request
  - one object per proxy

- conversation state
  - stateless
  - stateful
Distributed Object Systems
Distr. Object Systems vs. Service Architecture

- Dist. Object System
  - object oriented model
  - object references
  - stateful / stateless
  - tight coupling

- Service Architecture
  - OO-model / RPC
  - service references
  - stateless
  - loose coupling

internal communication between application’s tiers

external communication between applications
Distributed Object Systems
Distr. Object Systems vs. Service Architecture

- components / objects (distributed object system)
- stateful and stateless conversation
- transactions

- services
  - remote procedure calls
  - stateless conversation (session?)
  - message

- tight coupling
- loose coupling

- Web Services
- CORBA
- Remoting

- homogeneous
- heterogeneous
Distributed Object Systems
Type Mapping

Type System 1
Possible Types
Mappable Types

Interoperability Type System
Possible Types
Interop Subset

Type System 2
Possible Types
Mappable Types
Distributed Object Systems
Type Mapping, Example

Java Type System
- double
- char
- union

CORBA Type System
- char
- enum
- double
- wchar
- union

CLS Type System
- enum
- double
- char
- union

Custom implementation

© P. Reali / M. Corti
Distributed Object Systems

Examples

- Standards
  - OMG CORBA
    - IIOP
  - Web Services
    - SOAP

- Frameworks
  - Java RMI (Sun)
  - DCOM (Microsoft)
  - .NET Remoting (Microsoft)
    - IIOP.NET
Distributed Object Systems

CORBA

- Common Object Request Broker Architecture

Client Application

Client Stub

CORBA Runtime

Interface Repository

ORB

Remote Architecture

GIOP/IIOP

TCP/IP Socket

Object

Object Skeleton

Object Adaptor CORBA Runtime

Server

„Object-Bus“
Distributed Object Systems

CORBA

- CORBA is a standard from OMG
  - Object Management Group
  - Common Object Request Broker Architecture

- CORBA is useful for...
  - building distributed object systems
  - heterogeneous environments
  - tight integration

- CORBA defines...
  - an object-oriented type system
  - an interface definition language (IDL)
  - an object request broker (ORB)
  - an inter-orb protocol (IIOP) to serialize data and marshall method invocations
  - language mappings from Java, C++, Ada, COBOL, Smalltalk, Lisp, Phyton
  - ... and many additional standards and interfaces for distributed security, transactions, ...
Distributed Object Systems
CORBA

- Basic Types
  - integers
    - 16-, 32-, 64bit integers (signed and unsigned)
  - IEEE floating point
    - 32-, 64-bit and extended-precision numbers
  - fixed point
  - char, string
    - 8bit and wide
  - boolean
  - opaque (8bit), any
  - enumerations

- Compound Types
  - struct
  - union
  - sequence (variable-length array)
  - array (fixed-length)
  - interface
    - concrete (pass-by-reference)
    - abstract (pure definition)
  - value type
    - pass-by-value
    - abstract (no state)

- Operations
  - in / out / inout parameters
  - raises

- Attributes
Distributed Object Systems
CORBA / General Inter-ORB Protocol (GIOP)

- CDR (Common Data Representation)
  - Variable byte ordering
  - Aligned primitive types
  - All CORBA Types supported

- IIOP (Internet IOP)
  - GIOP over TCP/IP
  - Defines Interoperable Object Reference (IOR)
    - host
    - post
    - key

- Message Format
  - Defined in IDL
  - Messages
    - Request, Reply
    - CancelRequest, CancelReply
    - LocateRequest, LocateReply
    - CloseConnection
    - MessageError
    - Fragment
  - Byte ordering flag
  - Connection Management
    - request multiplexing
    - asymmetrical / bidirectional connections
Distributed Object Systems
CORBA / GIOP Message in IDL

module GIOP {
struct Version {
octet major;
octet minor;
}
enum MsgType_1_0 {
    Request, Reply,
    CancelRequest, CancelReply,
    LocateRequest, LocateReply,
    CloseConnection, Error
}
struct MessageHeader {
    char Magic[4];
    Version GIOP_Version;
    boolean byte_order;
octet message_size;
    unsigned long message_type;
}
} // module end GIOP
Distributed Object Systems
CORBA Services

- CORBA Services
  - System-level services defined in IDL
  - Provide functionality required by most applications

- Naming Service
  - Allows local or remote objects to be located by name
  - Given a name, returns an object reference
  - Hierarchical directory-like naming tree
  - Allows getting initial reference of object

- Event Service
  - Allows objects to dynamically register interest in an event
  - Object will be notified when event occurs
  - Push and pull models

- ... and more
  - Trader, LifeCycle, Persistence, Transaction, Security
Distributed Object Systems

WebServices

- Service-oriented architecture
- Rely on existing protocols
  - SOAP
    - messaging protocol
  - WSDL
    - service description protocol
  - UDDI
    - service location protocol

Web Services

- SOAP
- HTTP
- TCP/IP
Distributed Object Systems

SOAP

- Simple Object Access Protocol
- communication protocol
- XML-based
- describes object values
- XML Schemas as interface description language
  - basic types
    - string, boolean, decimal, float, double, duration, datetime, time, date, hexBinary, base64Binary, URI, QName, NOTATION
  - structured types
    - list, union

- SOAP Message
  - SOAP Envelope
  - SOAP Header
  - SOAP Body

- Method Call
  - packed as structure
  - messages are self-contained
  - no external object references
Distributed Object Systems

SOAP Message

- SOAP Message
  - SOAP Envelope
    - SOAP Header
    - SOAP Body

Example

float Multiply(float a, float b);
Distributed Object Systems

SOAP Example (Request)

POST /quickstart/aspplus/samples/services/MathService/CS/MathService.asmx
HTTP/1.1
Host: samples.gotdotnet.com
Content-Type: text/xml; charset=utf-8
Content-Length: length

SOAPAction: "http://tempuri.org/Multiply"

  <soap:Body>
    <Multiply xmlns="http://tempuri.org/"> <a>float</a> <b>float</b> </Multiply>
  </soap:Body>
</soap:Envelope>
Distributed Object Systems
SOAP Example (Answer)

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MultiplyResponse xmlns="http://tempuri.org/">
      <MultiplyResult>float</MultiplyResult>
    </MultiplyResponse>
  </soap:Body>
</soap:Envelope>
<?xml version="1.0" encoding="utf-8"?>
<definitions ....>
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="Multiply">
        <s:complexType><s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
          <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
        </s:sequence></s:complexType>
      </s:element>
    </s:schema>
  </types>
  <message name="MultiplySoapIn">
    <part name="parameters" element="s0:Multiply" />
  </message>
</definitions>
<binding name="MathServiceSoap" type="s0:MathServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="Multiply">
        <soap:operation soapAction="http://tempuri.org/Multiply" style="document"/>
        <input><soap:body use="literal" /></input>
        <output><soap:body use="literal" /></output>
    </operation>
</binding>

<service name="MathService">
    <port name="MathServiceSoap" binding="s0:MathServiceSoap">
        <soap:address location="http://samples.gotdotnet.com/quickstart/aspplus/samples/services/MathService/CS/MathService.asmx"/>
    </port>
</service>
</definitions>
Distributed Object Systems
WebServices

**Comments**
- XML (easily readable)
- system independent
- standard
- stateless (encouraged design pattern)
- bloated
- big messages (but easily compressed)
- requires expensive parsing

**Constraints**
- Services
  - no object references
  - server-activated servant
- Goes over HTTP
  - requires web server
Distributed Object Systems

WebService Future

- Use SOAP-Header to store additional information about message or context
- Many standards to come...
  - WS-Security
  - WS-Policy
  - WS-SecurityPolicy
  - WS-Trust
  - WS-SecureConversation
  - WS-Addressing
Distributed Object Systems
Java RMI

Java Remote Method Invocation
Distributed Object Systems
Java RMI Details

- Framework
  - supports various implementations
    - e.g. RMI/IIOP
      - mapping limited to the Java type system, workarounds needed
    - uses reflection to inspect objects
Distributed Object-Systems
Low-Level Details: Java RMI/IIOP

- Common Type-System
  - restricted CORBA
- Marshalling
  - name mapping
  - remote objects
    - only references
- Interface Description Language (IDL)
  - java to IDL mapping
- Message representation
- Underlying protocol
  - IIOP (CORBA)
Distributed Object Systems
Microsoft DCOM

Distributed Common Object Model

Client Application
Object Proxy
COM Runtime

Remote Architecture
SCMs and Registration
SCM
Registry

Client

Object
Object Stub
COM Runtime

Server

Network
OXID Resolver
RPC Channel
Ping Server

© P. Reali / M. Corti
Distributed Object Systems
Microsoft .NET Remoting

new Instance()
or
Activator.GetObject(...)

Client → Transparent Proxy → ObjRef → Channel

IChannelInfo IEnvoyInfo IRemotingTypeInfo
string ChannelInfo; EnvoyInfo; TypeInfo; URI;

Application Domain Boundary

Network → Channel → Instance
Distributed Object Systems
Microsoft .NET Remoting

Instance s = new Instance();
s.DoSomething();
Distributed Object Systems
Microsoft .NET Remoting

Activation
- client
  - one instance per activation
- server / Singleton
  - one instance of object
- server / SingleCall
  - one instance per call

Leases (Object Lifetimes)
- renew lease on call
- set maximal object lifetime

Serialization
- SOAP
  - Warning: non-standard types, only for .NET use
- binary
- user defined

Transport
- TCP
- HTTP
- user defined
Distributed Object Systems
Microsoft .NET Remoting (Object Marshalling)

- MarshalByRefObjects
  - remoted by reference
  - client receives an ObjRef object, which is a “pointer” to the original object

- [Serializable]
  - all fields of instance are cloned to the client
  - [NonSerialized] fields are ignored

- ISerializable
  - object has method to define own serialization
Distributed Object Systems
Microsoft .NET Remoting, Activation

- **Server-Side Activation**
  (Well-Known Objects)
  - Singleton Objects
    - only one instance is allocated to process all requests
  - SingleCall Objects
    - one instance per call is allocated

- **Client-Side Activation**
  - Client Activated Objects
    - the client allocates and controls the object on the server

“stateless”
“stateful”
Distributed Object Systems
Microsoft .NET Remoting, Limitations

- Server-Activated Objects
  - object configuration limited to the default constructor
- Client-Activated Objects
  - class must be instantiated, no access over interface
  - class hierarchy limitations
  - use Factory Pattern
    - to get interface reference
    - to allow parametrization of the constructor

- Furthermore...
  - interface information is lost when passing an object reference to another machine
  - no control over the channel
    - which channel is used
    - which peer is allowed to connect
Distributed Object Systems
Case Study: IIOP.NET

- Opensource project based on ETH-Diploma thesis
  - http://iiop-net.sourceforge.net/

- IIOP.NET (marketing)
  - „Provide seamless interoperability between .NET and CORBA-based peers (including J2EE)“

- IIOP.NET (technical)
  - .NET remoting channel implementing the CORBA IIOP protocol
  - Compiler to make .NET stubs from IDL definitions
  - IDL definition generator from .NET metadata
Distributed Object Systems
Case Study: IIOP.NET

- IIOP rather than SOAP
  - transparent reuse of existing servers
  - tight coupling
  - object-level granularity
  - efficiency

- Runtime: standard .NET remoting channel for IIOP
  - transport sink
  - formatter
  - type-mapper

- Build tools
  - IDL → CLS compiler
  - CLS → IDL generator
## Distributed Object Systems

### Case Study: IIOP.NET, Interoperability

<table>
<thead>
<tr>
<th>Application</th>
<th>This is what we want</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>Distributed Transaction Coordinator, Active Directory, ...</td>
</tr>
<tr>
<td>Conversation</td>
<td>Activation model (EJB, MBR), global naming, distributed garbage collection, conversational state,...</td>
</tr>
<tr>
<td>Contextual Data Interception Layer</td>
<td>SessionID, TransactionID, cultureID, logical threadID ...</td>
</tr>
<tr>
<td>Message Format</td>
<td>RPC, IIOP, HTTP, SOAP, proprietary binary format, messages, unknown data (exceptions), encryption</td>
</tr>
<tr>
<td>Data Model</td>
<td>Type system, mapping and conversion issues</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>TCP/UDP, Byte stream, point-to-point communication</td>
</tr>
</tbody>
</table>

© P. Reali / M. Corti
Distributed Object Systems
Case Study: IIOP.NET, Granularity

System

Service
Component
Object
Object
Component
Object
Object
Component
Object
Object

Granularity
Coupling, Interaction

Service
Component
Object
Object

Message-based Interface, Stateless
Strongly-typed Interface, Stateless or Stateful
Implementation Dependency, Stateful
Distributed Object Systems
Case Study: IIOP.NET

© P. Reali / M. Corti
Distributed Object Systems
Case Study: IIOP.NET, Performance

- Test Case:
  - WebSphere 5.0.1 as server
  - Clients
    - IBM SOAP-RPC Web Services
    - IBM Java RMI/IIOP
    - IIOP.NET

- Response time
  - receiving 100 beans from server
    - WS: 4.0 seconds
    - IIOP.NET: 0.5 seconds
  - when sending many more beans, WS are then 200% slower than IIOP.NET

Source: posted on IIOP.NET forum

© P. Reali / M. Corti