Chapter 16

by

David G. Messerschmitt

Layering a computing infrastructure
### Spanning layer

- **Application**
- **Distributed object management**

<table>
<thead>
<tr>
<th>Windows NT</th>
<th>Mac OS</th>
<th>UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>TCP</td>
<td>TCP</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP</td>
<td>UDP</td>
</tr>
</tbody>
</table>

### Internet protocol

- **Network 1**
- **Network 2**
- **Spanning layers**

### The new infrastructure: middleware

- **Layer of software between OS and application**
  - Hides heterogeneity
  - Provides generic common services
  - Increases level of abstraction
- **By its nature, not single platform, nor bundled with equipment**
  - Microsoft attempts to be an exception
The new infrastructure

• Middleware is
  – where new capabilities are added by layering
  – where much of the experimentation and innovation happens in the infrastructure
  – where the successful approaches have a chance to become a spanning layer and/or integrated into a distributed OS

• Boundary to a distributed OS is fuzzy

Middleware objectives

• Hide heterogeneity
• Location independence
• Common functionality needed by many applications
• Software portability and mobile code
• Help integrate legacy facilities
• Aid application interoperability
• Aid scalability
Some middleware categories

- **Transaction processing**
  - Simplify the coordination of complementary resource managers

- **Message-oriented middleware**
  - Support message and queuing capabilities where resource managers are not available simultaneously (like workflow)

- **Distributed object management**
  - Support applications that are distributed across heterogeneous platforms and organizations

- **Mobile code**
  - Allow application code to be moved and executed on heterogeneous platforms
  - Without prior software installation
Infrastructure software today

• With networks, new emphasis on:
  – Portability: applications run across multiple platforms (avoid lock-in)
  – Interoperability: pieces of application must work together (benefit from network effects)

What are some examples of each?

Transaction processing

by

David G. Messerschmitt
The transaction

Durable starting state → Collection of resource actions → Durable, consistent, ending state

Abort → Successful completion

Rollback

Transaction architecture

Application logic

Join

Transaction manager

Prepare, commit, abort

Resource managers
Commit or abort

Transaction manager

One or more “no’s”

Phase 1

Transaction manager

preparation

prepare() yes or no

Phase 2

Transaction manager

all “yes”

abort()

Commit or abort

Atomic series of resource actions

Client_1 Client_2 Server

Group of RMI’s

Starting state

Inconsistent state

Final state
Locking to prevent conflicts

Client_1  Client_2  Server

Starting state

Lock

Group of RMI’s

Refused

Unlock

Final state

Abort

Client_1  Client_2  Server

Starting state

Group of RMI’s

Something goes awry

Inconsistent state

Final state
Mobile code and Java

by

David G. Messerschmitt

Reminder: two key requirements

• With networks, new emphasis on:
  – Portability: applications run across multiple platforms (avoid lock-in)
  – Interoperability: pieces of application must work together (benefit from network effects)
Dynamic portability: mobile code

1. Send code (as a message) to a host

2. Execute the program represented by that code

Mobile code:
Code representing a software program that can be moved to heterogeneous platforms and executed there

Portability can aid interoperability

Mobile code originating from a common source can enhance interoperability
Java

- **Portability**
  - “Write once, run anywhere”

- **Programming productivity**
  - Garbage collection (no memory leaks)
  - Multi-threaded

- **Scalability**
  - Move execution cycles

- **Interoperability**
  - Software components come from common repository

---

**Mobile code:** Code representing a software program

**Mobile agent:** Code and data representing an object or component

1. Send as a message to a host
2. Execute the program

---
Some mobile code advantages

- Executing program closer to user can enhance interactivity.
- Mobile code originating from a common source can enhance interoperability and bypass network effects.
- Shifting location of computation can enhance scalability.

Idea of mobile agents

- Agent executes in each host, modifying its state.
- Agent launched.
- Agent returns.
SUN/Java strategy

- License Java freely, even to rival Microsoft
  - Why?
- License terms give Sun a modicum of control over the “standard”
  - Why?
- How does Sun expect to make money?
Distributed object management

- Emphasis is on interoperability
  - Allows objects on one host to invoke methods of objects on another host
  - Platform, language independent

- CORBA vs DCOM

- Portability is not the emphasis

Interoperability

Client object can invoke methods of server object, even if they are running on different platforms, written in different languages

What else is needed?
Interoperability

- Common structure of data
- Common interpretation of data
- Agreement on protocols

Before and after

Potential spanning layer
What is the acronym?

• Common Object Request Broker Architecture

or

• Concerned Off-Road Bicyclist Association?
### Protocol Layer

- **Location-independent application**
- **Object Management Group CORBA standard**
- **Internet Inter-ORB Protocol (IIOP)**
- **User datagram protocol (UDP)**
- **Transmission control protocol (TCP)**
- **Internet protocol (IP)**
- **Subnetworks**

### Portability not promised

- **Location-dependent application**
- **Interoperability**
- **CORBA standard does not insure ORB-to-ORB portability**
OMG process

- Identify need
- Request for proposals
- Process to
  - choose best
  - or ask proposal advocates to work together

Which is most effective?

- Industry de facto standard effort (CORBA)
- Single vendor integrated solution (DCOM)?
**CORBA vs DCOM**

<table>
<thead>
<tr>
<th>CORBA</th>
<th>DCOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Integrate best ideas</td>
<td>• Fast, no consensus required</td>
</tr>
<tr>
<td>• Multi-vendor support</td>
<td>• No vendor interoperability issues</td>
</tr>
<tr>
<td>• Cross-platform and language</td>
<td></td>
</tr>
</tbody>
</table>

Two methods for application interaction

- CORBA and DCOM
- Exchange documents (XML)

What are their relative merits?
## CORBA vs XML

<table>
<thead>
<tr>
<th>CORBA</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Natural OOP extension</td>
<td>• Flexible data-sharing</td>
</tr>
<tr>
<td>• No document interpretation</td>
<td>• Good for document-like objects</td>
</tr>
<tr>
<td>• Good for back-and-forth protocols</td>
<td>• Natural cross-platform capability</td>
</tr>
<tr>
<td><strong>Both have need for standardization of data or document interpretation</strong></td>
<td>• No protocol standardization</td>
</tr>
</tbody>
</table>

### What CORBA provides

- **Java**
  - Language bindings
    - Transportable objects
- **“Inter-galactic” software bus**
  - Cross-platform and language
  - Interoperability (but not portability)
- **High levels of abstraction**
  - Remote method invocation on objects
What CORBA provides (con’t)

- Run-time flexibility
  - Everything self-describing
  - Interface discovery
  - Dynamic data structures and binding
- Useful services
  - Naming
  - Security
  - Many others

Importance of CORBA

- Inter-enterprise computing
  - Platform and language independence
  - Electronic commerce, network management, etc
- Reduction of network effects
  - Another spanning layer
  - Significance of platform reduced
Are Java and CORBA competitive or complementary?

- Both offer interoperability across different platforms
- Java offers portability and transportability
- CORBA offers heterogeneous language bindings
- CORBA offers many services, metadata, etc.
- Bottom line: they are complementary!
  - (but some Java proponents may not agree)

Who favors what?

Would these vendors be in favor of:

- Interoperability?
  - Microsoft
  - Intel
  - SUN
  - Novell
  - Iona

- Portability?

Understanding Networked Applications A First Course