F DCOM

F.1 Overview

- Terminology
- COM Architecture
- Comparison to CORBA

F.2 References

EdEd98. G. Eddon, H. Eddon: *Inside Distributed COM*. Microsoft Programming Series, Microsoft Press, Redmond, Wash., 1998.

OHE96. R. Orfali, D. Harkey, J. Edwards: *The essential distributed objects survival guide*. John Wiley & Sons, 1996.

Micr96. Microsoft Corporation: *DCOM technical overview*. White paper. Redmond, Wash., 1996.

Micr98. Microsoft Corporation: *DCOM architecture*. White paper. Redmond, Wash., 1998.

.

Wan+97. P. Chung, Y. Huang, S. Yajnik, D.-R. Liang, J. Shih, C.-Y. Wang, Y.-M. Wang: "DCOM and CORBA Side by Side, Step By Step, and Layer by Layer." In *C++ Report*, Jan. 1998. `http://akpublic.research.att.com/~ymwang/papers/HTML/DCOMnCORBA/S.html`

Kirt97. M. Kirtland: "The COM+ Programming Model Makes it Easy to Write Components in Any Language." In *Microsoft Systems Journal*, Dec. 1997. `http://www.microsoft.com/msj/1297/complus2/complus2.htm`

F.3 Terminology

1 OLE – Object Linking and Embedding

- Microsoft's standard for collaboration of software components
  - ◆ E.g., spreadsheet table cells in a text document
  - ◆ E.g., graphics in a spreadsheet table cell
- Defines object/component interfaces and protocols for
  - ◆ Linkage and notification for embedded components
  - ◆ "Drag and drop" of graphical objects
  - ◆ Clipboard
  - ◆ Structured storage (Compound files)
  - ◆ Scripting
- Microsoft Foundation Classes (MFC)
  - ◆ GUI programming and handling

2 COM – Component Object Model

- OLE's components belong to different processes/programs
  - ◆ Communication substrate needed
- COM as an object request broker and service provider
  - ◆ OLE components are COM objects
  - ◆ Single-machine environment
- Intra–address-space communication
  - ◆ Forwarding requests to other COM objects
  - ◆ Integration into the MFC event model
- Inter–address-space communication
  - ◆ Stubs
  - ◆ Light-weight RPC (LRPC)

3 DCOM – Distributed COM

- Extends COM to a distributed environment
  - DCE/RPC with at-most-once/exactly-once semantics

4 ActiveX

- COM enabled for the Internet (whatever that means)
  - *Just a marketing buzzword!*

5 COM+

- Improved programming environment for COM
  - Maps COM+ objects to COM objects
  - Handles reference counting and other standard procedures

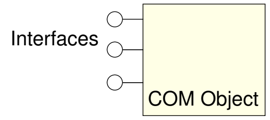
F.4 COM Architecture

1 IDL – Interface Definition Language

- ▲ Not the same as CORBA IDL!
- Language for describing object interfaces
  - Independent from the target programming language
  - No mapping to language constructs
  - Definition of a binary object invocation interface *(vtables)*
- MIDL compiler = stub generator
  - Client stubs (proxies)
  - Server stubs

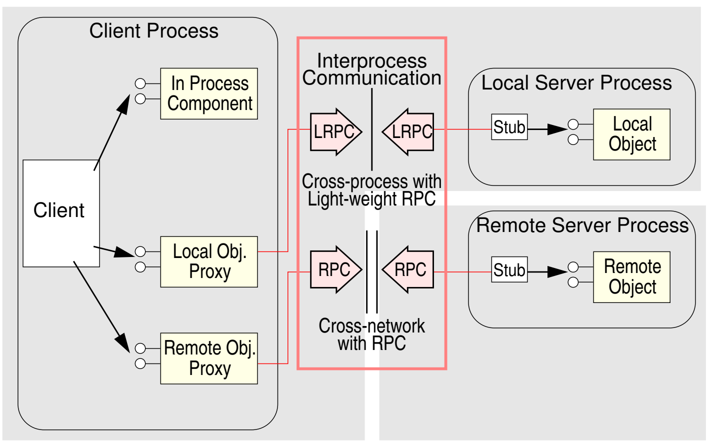
2 Object Model

- Objects can have multiple interfaces
  - Multiple versions of one interfaces
  - Different interfaces for different purposes
  - Means to investigate the other interfaces
- Single inheritance on interfaces
  - Every interface inherits from `IUnknown`, which implements methods for finding other interfaces
  - Multiple inheritance must be emulated by multiple interfaces



- Centralized object approach

2 Object Model (2)



3 Process of Creation and Binding

- Creation of a server object
  - Description of the object interfaces in IDL
  - Programming server class and class factory in a target language
  - Registration of the class factory in the registry
  - On client demand an object is created
  - A transient object reference is marshalled and handed out to the client
- Binding to the server object at the client site
  - Retrieve class ID of factory object from the registry
  - Invoke `CoCreateInstance()` method, which returns a reference to the object
  - Proxy (client stub) is automatically installed (code needs to be registered in the registry)
  - Method invocations using the proxy

3 Process of Creation and Binding (2)

- Proxies are COM objects
  - Class of the proxy object must be known at the client site (registered at the registry)
- *Custom Marshalling*
  - User may create his own proxy objects
    - Intelligent proxies
    - Non-RPC communication
  - Custom marshalling is similar to the fragmented object approach

4 Monikers

- COM does not know persistent object references
  - If a server object is deactivated the object reference will be invalid.
- Monikers
  - COM object
  - Knows a name for a "persistent" object
  - Can (re-)create the object and
  - feed it with its former state
- "Names"
  - URLs
  - Filenames
  - e.g., `c:\windows\test.xls!a1-d4` for spreadsheet cells in a particular file

F.5 Comparison to CORBA

- IDL and language mapping
  - **CORBA:** IDL is mapped to language constructs
    - Mapping is easier
  - **DCOM:** IDL defines binary data layout, language constructs are mapped to this layout
    - Heterogeneous binary component can be hosted in one address space
- Persistent object references
  - **CORBA:** POA and implementation repository
    - Arbitrary and user-defined implementations
  - **DCOM:** Monikers as mediators

F.5 Comparison to CORBA (2)

- Communication
  - **CORBA:** RPC-based invocation (at-most-once/exactly-once)
  - **DCOM:** RPC-based invocation (at-most-once/exactly-once) plus Custom Marshalling
    - Arbitrary communication mechanisms can be used
- Binding
  - **CORBA:** Interface-dependent stub must be known at client site
  - **DCOM:** Class ID and code of proxy must be registered at the registry
- Dynamic invocation
  - **CORBA:** DII, interface repository
  - **DCOM:** `IDispatch` interface, type library

F.5 Comparison to CORBA (3)

- Availability
  - **CORBA:** Virtually all platforms
  - **DCOM:** Windows 95/98/NT/2000, MacOS, recently Solaris
- Bodies
  - **CORBA:** OMG and its several hundred members
  - **DCOM:** Microsoft and some supporters
- ★ CORBA defines gateways to the DCOM world
  - "Fully" interoperable