Lesson 2 – Introduction to SOAP

Service Oriented Architectures

Module 1 - Basic technologies

Unit 1 – Introduction

Ernesto Damiani

Università di Milano

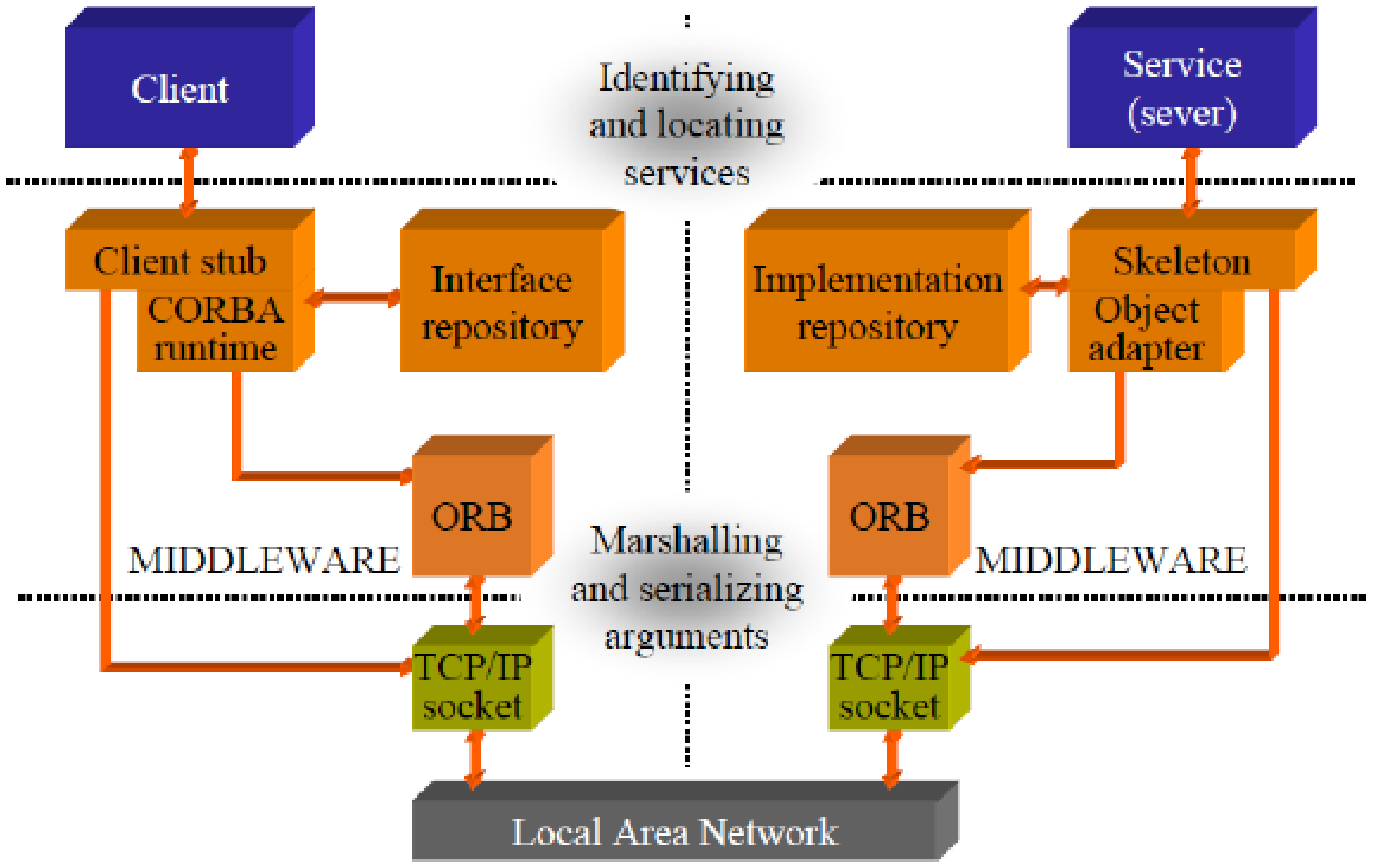
Basic problems to solve (1)

- How to make the service invocation part of the language in a more or less transparent manner
- How to exchange data between machines that might use different representations for different data types
  - This involves two aspects: data type formats (e.g., byte orders in different architectures) and data structures (need to be flattened and then reconstructed)

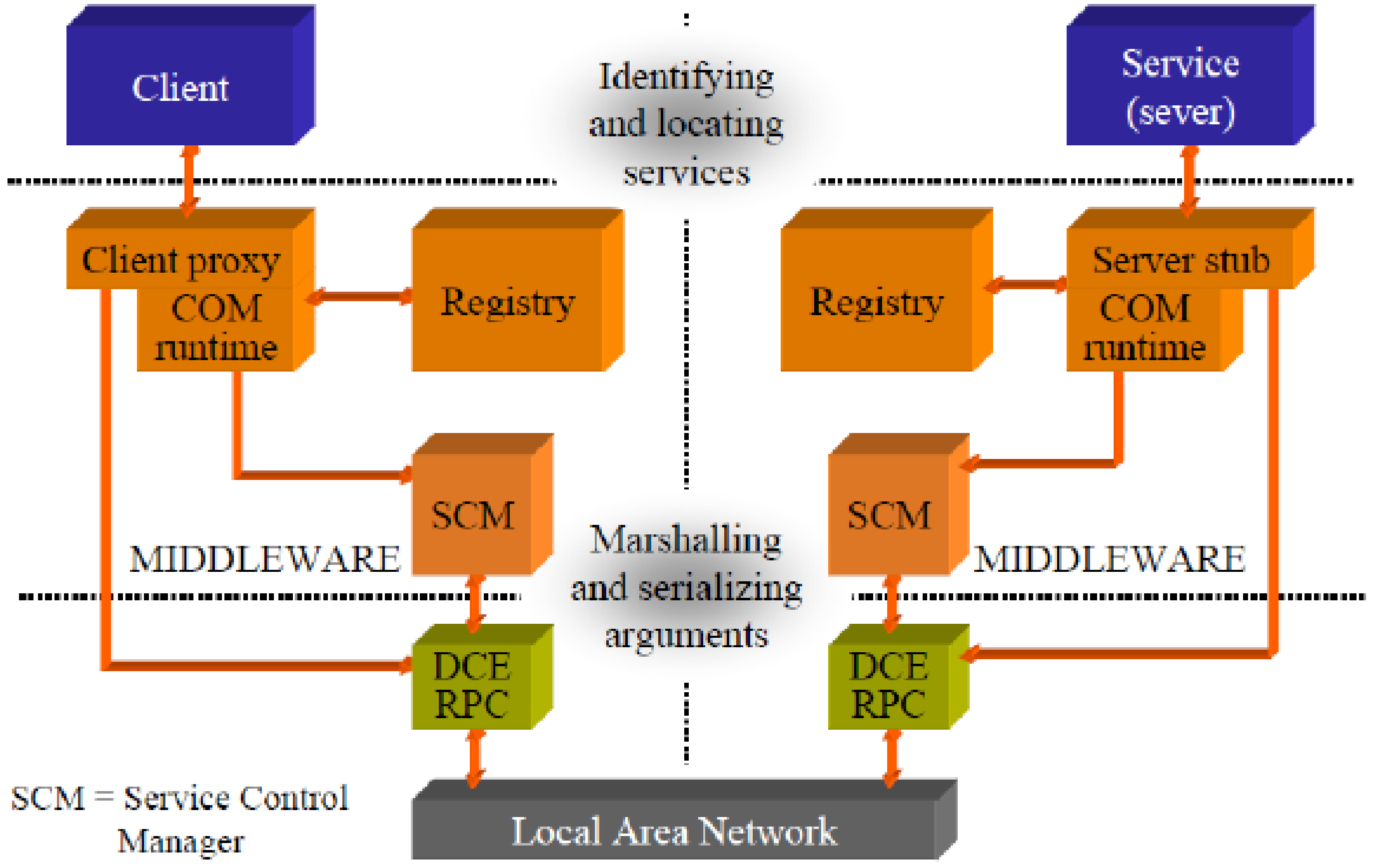
Basic problems to solve (2)

- How to find the service one actually wants among a potentially large collection of services and servers. The client does not necessarily need to know where the server resides or even which server provides the service
- How to deal with errors in the service invocation in a more or less elegant manner:
  - server is down or busy
  - communication is down
  - duplicated requests

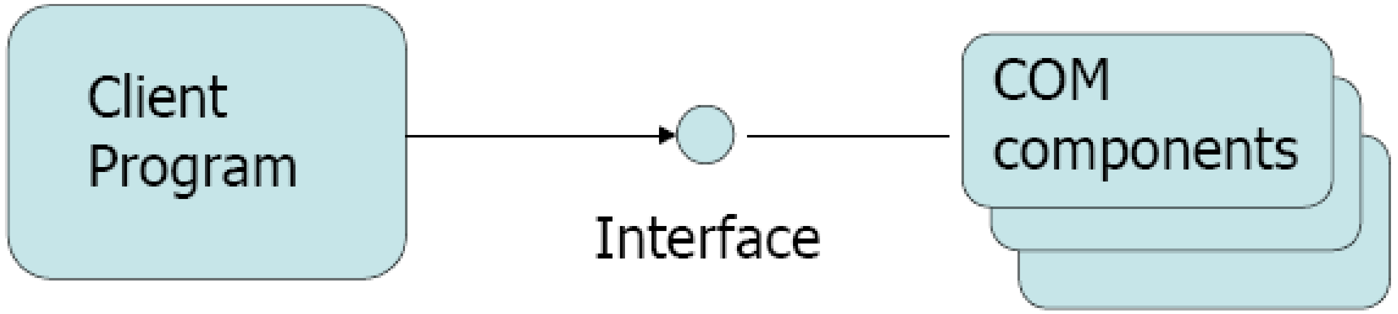
CORBA invocations



DCOM invocations



COM model



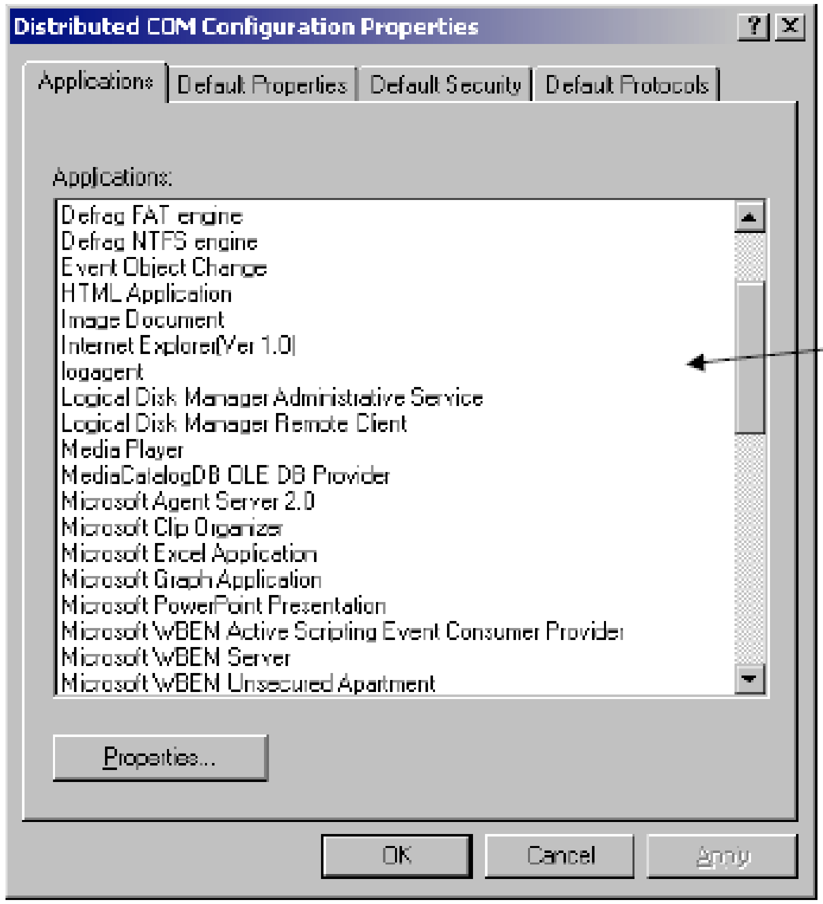
DCOM runtime

- Installed by default
  - Windows XP, 2k, (98, Me)
- Not installed by default
  - Windows NT
- But installed with other apps (ex. IE)

DCOMCNFG.exe

- DCOM Configuration Tool
- View installed DCOM-enable applications list

List of DCOM-enabled apps



Applications:

Defrag FAT engine
Defrag NTFS engine
Event Object Change
HTML Application
Image Document
Internet Explorer(Ver 1.0)
logagent
Logical Disk Manager Administrative Service
Logical Disk Manager Remote Client
Media Player
MediaCatalogDB OLE DB Provider
Microsoft Agent Server 2.0
Microsoft Clip Organizer
Microsoft Excel Application
Microsoft Graph Application
Microsoft PowerPoint Presentation
Microsoft WBEM Active Scripting Event Consumer Provider
Microsoft WBEM Server
Microsoft WBEM Unsecured Apartment

DCOM-enabled apps

Windows built-in DCOM apps

- Internet Explorer
- Windows Media Player
- Windows Scripting Host
- Sound recorder
- WordPad
  - and more…

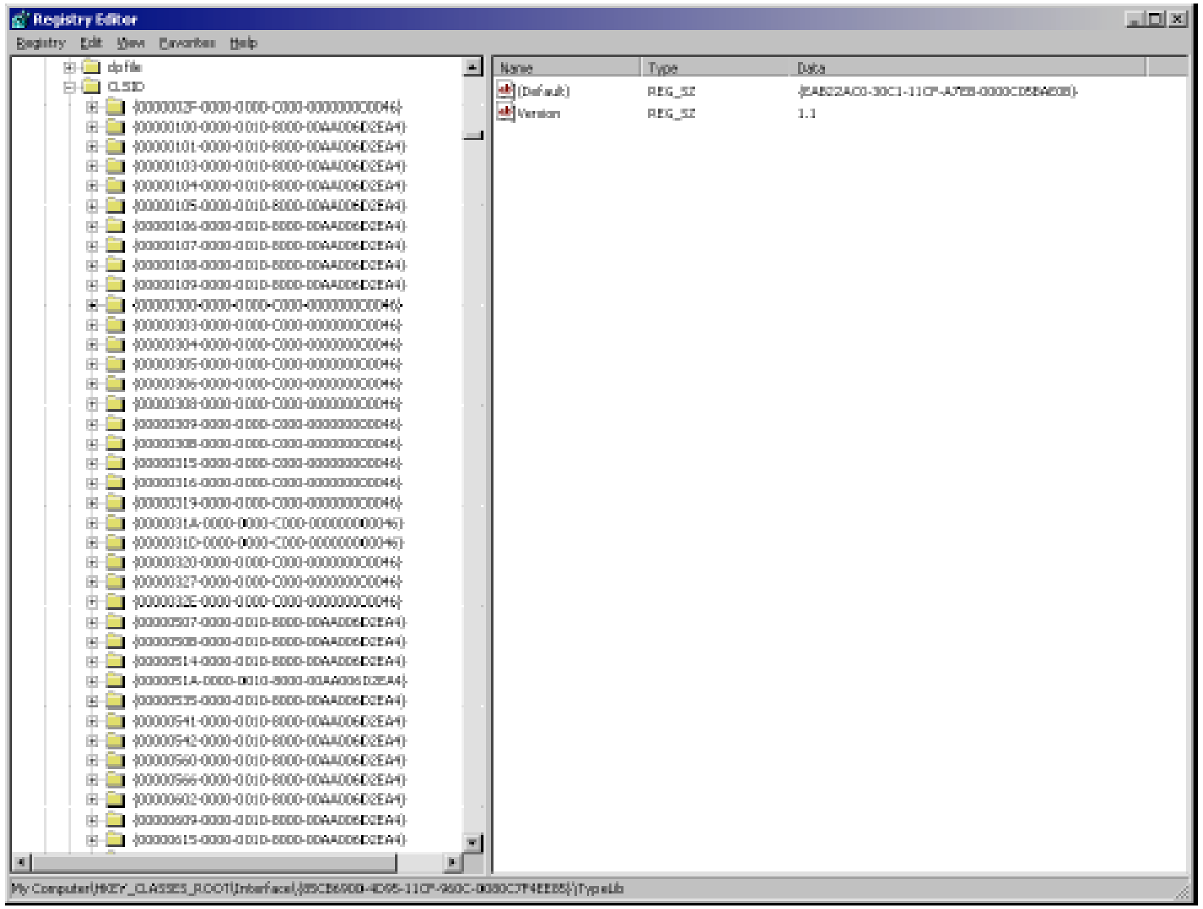
Other applications

- Word
- Excel
- Outlook
- PowerPoint
  - and more …

COM components on Windows

- Windows has many COM components
- They are registered under “\HKEY_CLASSES_ROOT\CLSID” in the registry

COM components in Registry

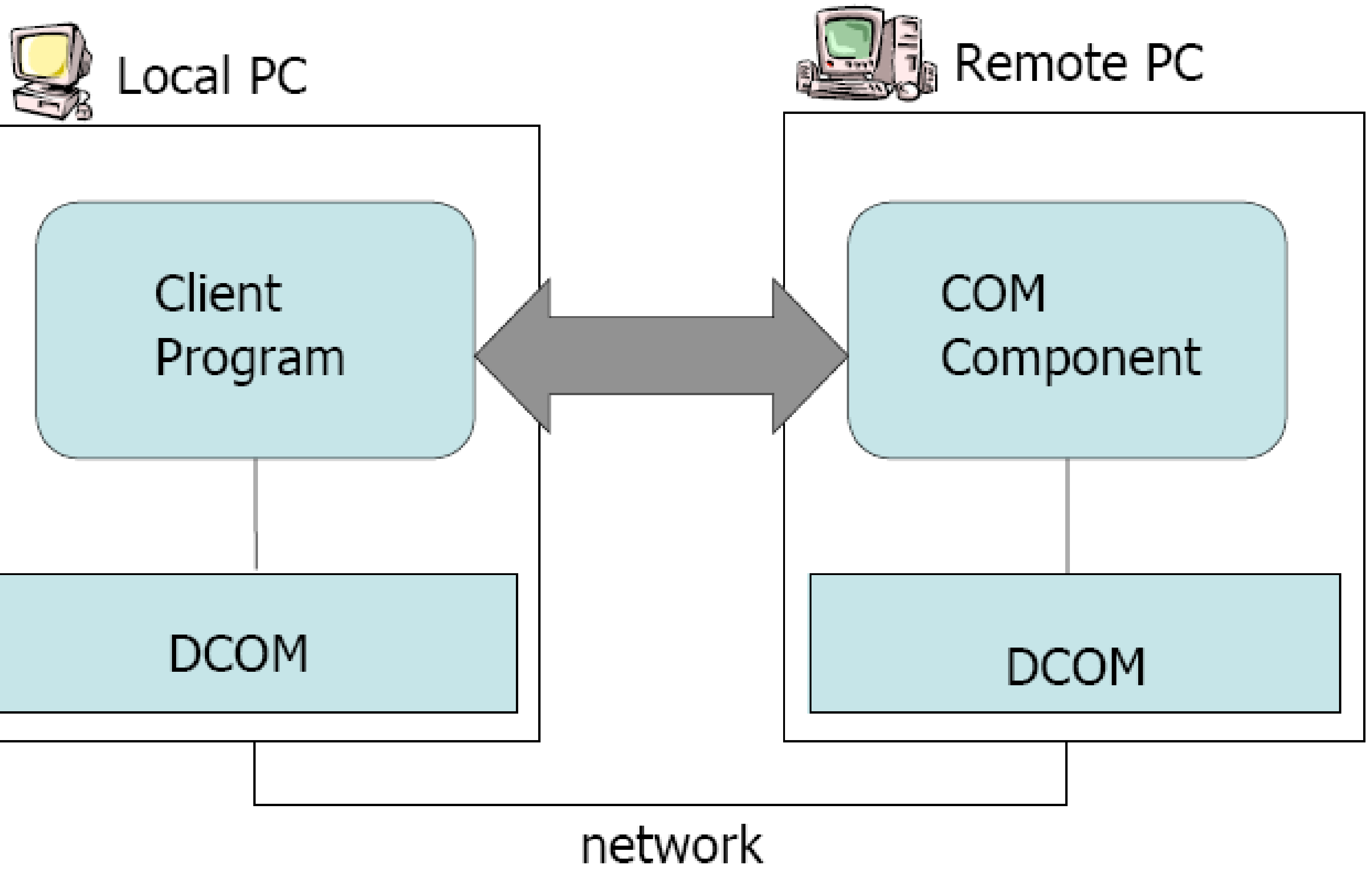


```
dpfile
CLSID
{0000002F-0000-0000-C000-000000000046}
{00000100-0000-0010-8000-00AA006D2EA4}
{00000101-0000-0010-8000-00AA006D2EA4}
{00000103-0000-0010-8000-00AA006D2EA4}
{00000104-0000-0010-8000-00AA006D2EA4}
{00000105-0000-0010-8000-00AA006D2EA4}
{00000106-0000-0010-8000-00AA006D2EA4}
{00000107-0000-0010-8000-00AA006D2EA4}
{00000108-0000-0010-8000-00AA006D2EA4}
{00000109-0000-0010-8000-00AA006D2EA4}
{00000300-0000-0000-C000-000000000046}
{00000303-0000-0000-C000-000000000046}
{00000304-0000-0000-C000-000000000046}
{00000305-0000-0000-C000-000000000046}
{00000306-0000-0000-C000-000000000046}
{00000308-0000-0000-C000-000000000046}
{00000309-0000-0000-C000-000000000046}
{0000030B-0000-0000-C000-000000000046}
{00000315-0000-0000-C000-000000000046}
{00000316-0000-0000-C000-000000000046}
{00000319-0000-0000-C000-000000000046}
{0000031A-0000-0000-C000-000000000046}
{0000031D-0000-0000-C000-000000000046}
{00000320-0000-0000-C000-000000000046}
{00000327-0000-0000-C000-000000000046}
{0000032E-0000-0000-C000-000000000046}
{00000507-0000-0010-8000-00AA006D2EA4}
{0000050B-0000-0010-8000-00AA006D2EA4}
{00000514-0000-0010-8000-00AA006D2EA4}
{0000051A-0000-0010-8000-00AA006D2EA4}
{00000535-0000-0010-8000-00AA006D2EA4}
{00000541-0000-0010-8000-00AA006D2EA4}
{00000542-0000-0010-8000-00AA006D2EA4}
{00000560-0000-0010-8000-00AA006D2EA4}
{00000566-0000-0010-8000-00AA006D2EA4}
{00000602-0000-0010-8000-00AA006D2EA4}
{00000609-0000-0010-8000-00AA006D2EA4}
{00000615-0000-0010-8000-00AA006D2EA4}
```

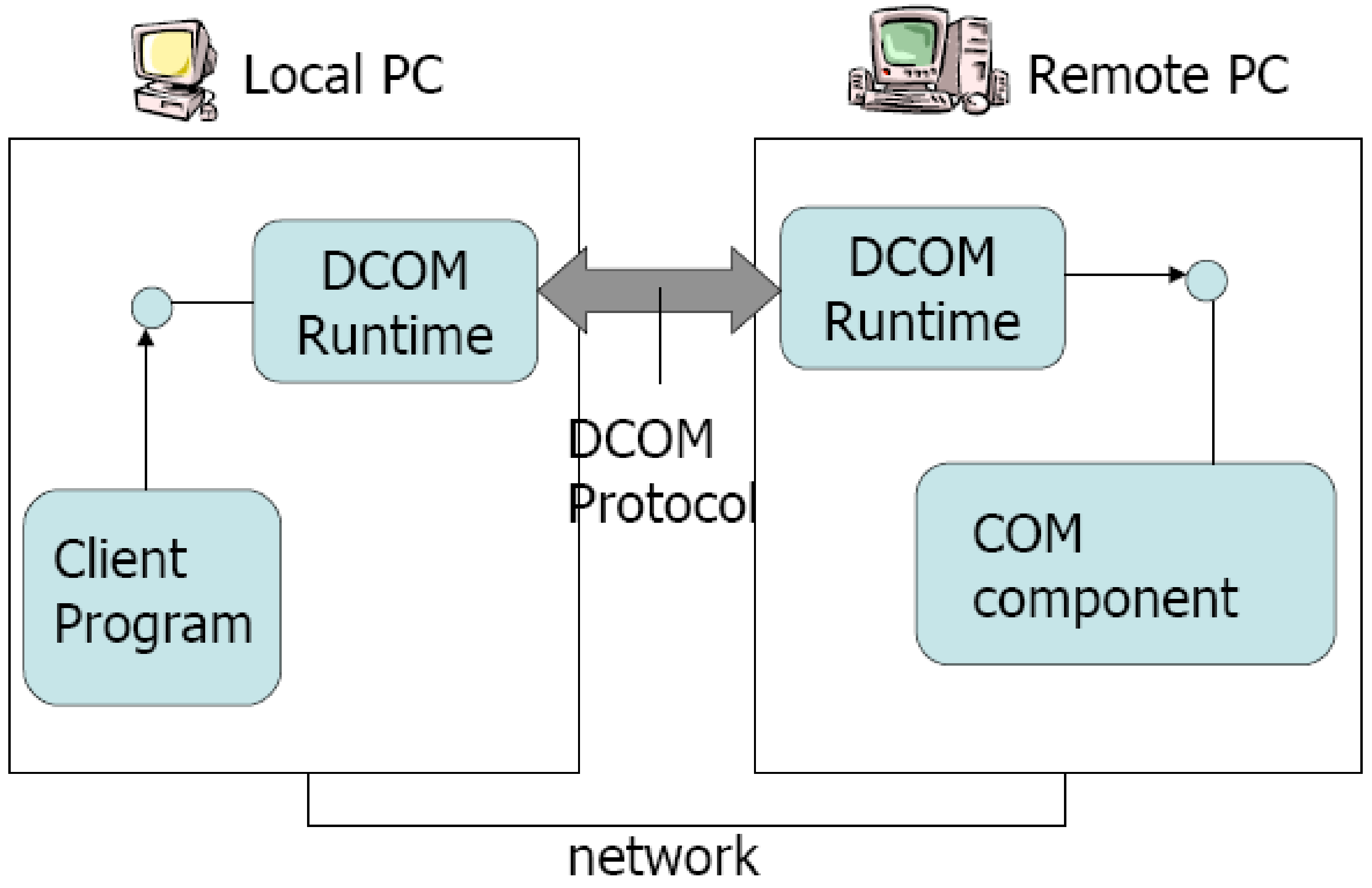
Name	Type	Data
(Default)	REG_SZ	{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}
Version	REG_SZ	1.1

```
My Computer\HKEY_CLASSES_ROOT\Interface\{85CB6900-4D95-11CF-960C-0080C7F4EE85}\TypeLib
```

Distributed apps by using DCOM

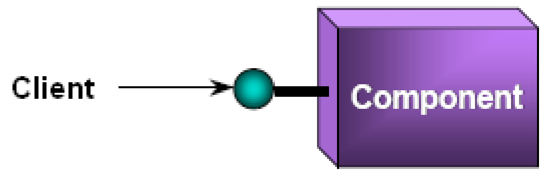


DCOM model

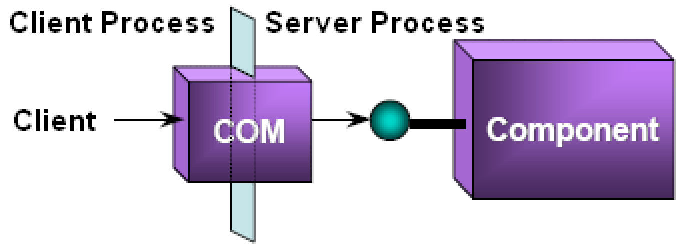

Local PC
Remote PC
DCOM Runtime
DCOM Runtime
DCOM Protocol
Client Program
COM component
network

The COM / DCOM scalability (1)

- In the same process
  - Fast, direct function calls

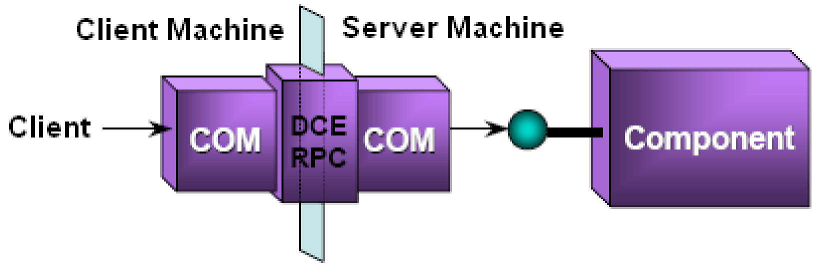


- On the same machine
  - Fast, secure IPC

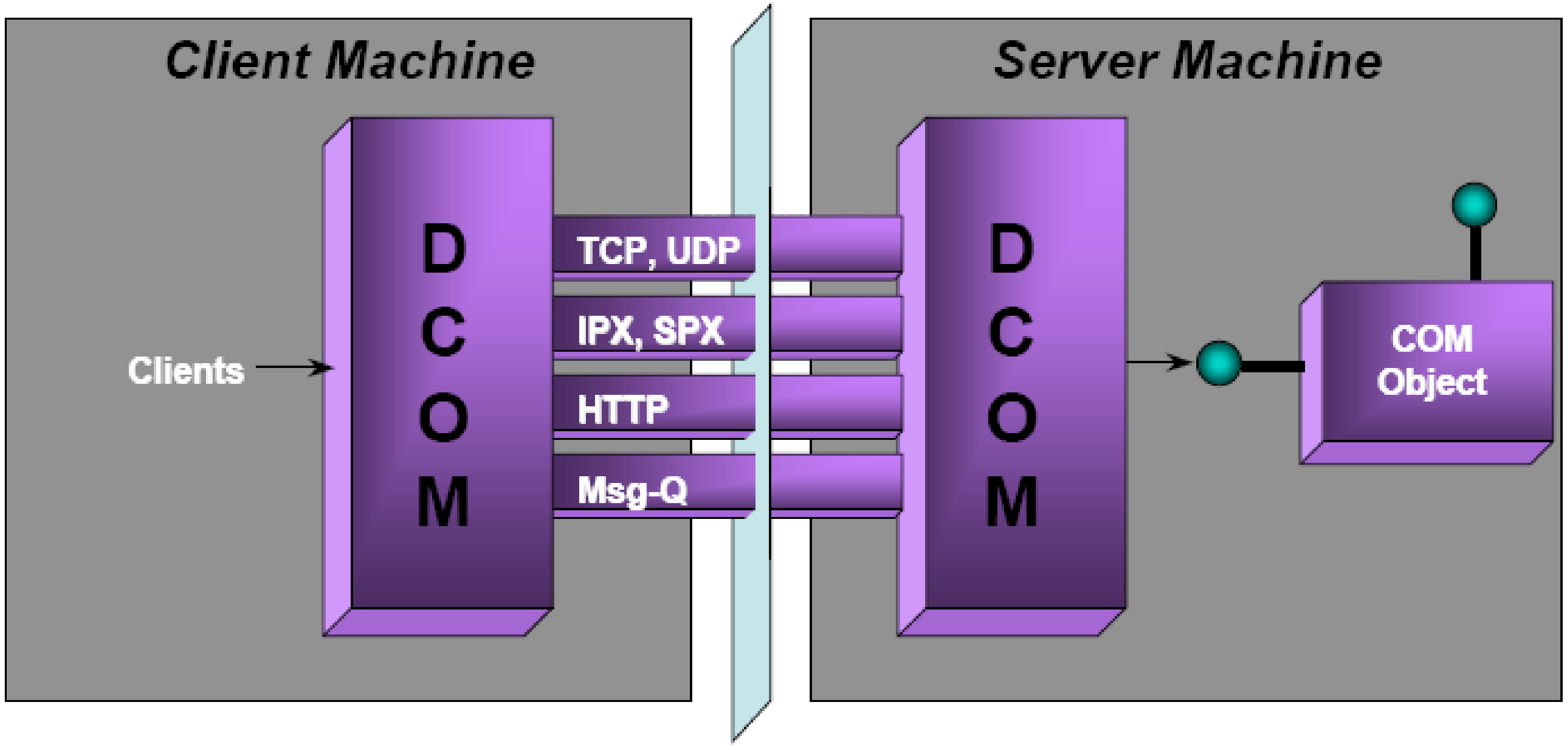


The COM / DCOM scalability (2)

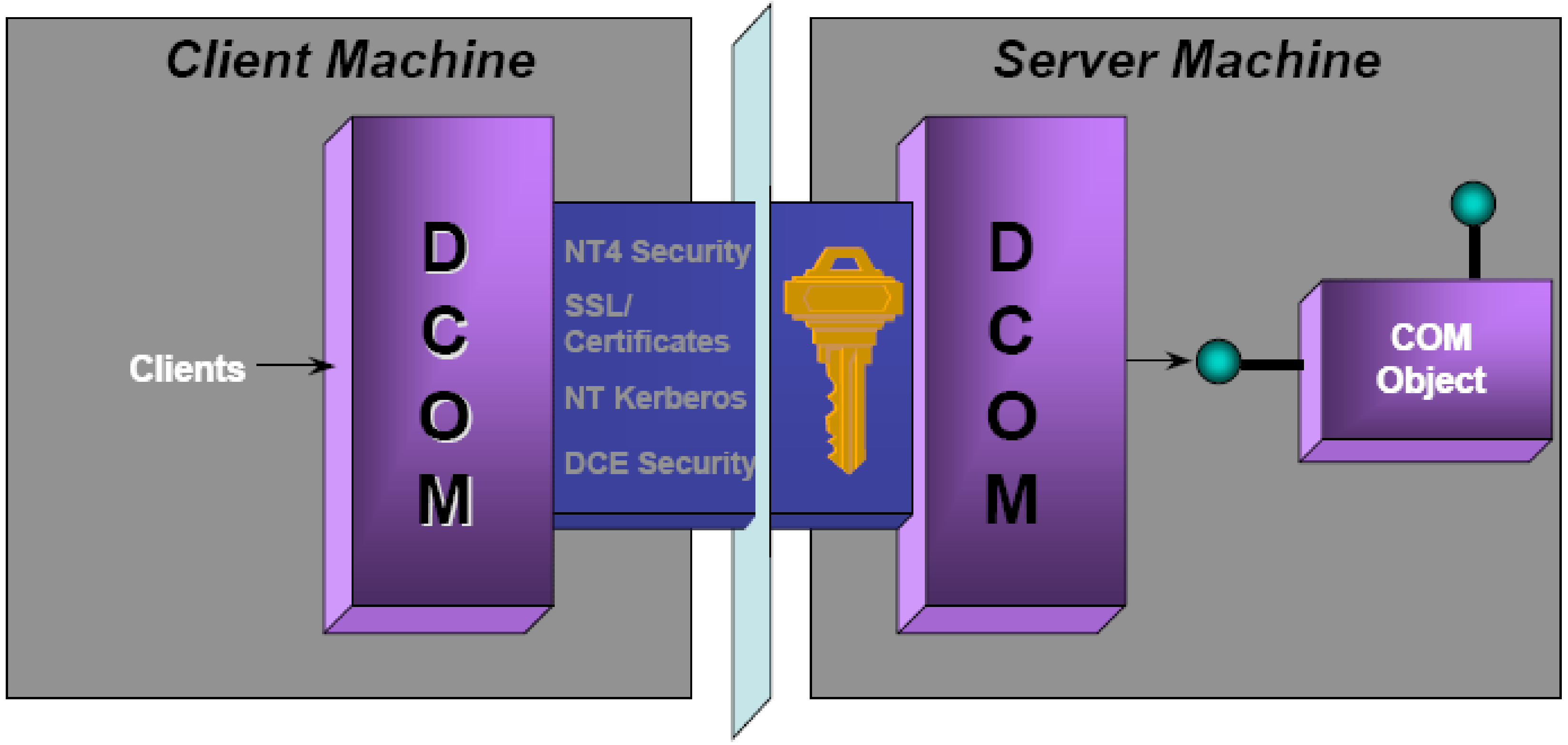
- Across machines
  - Secure, reliable and flexible DCE-RPC based DCOM protocol



DCOM transports

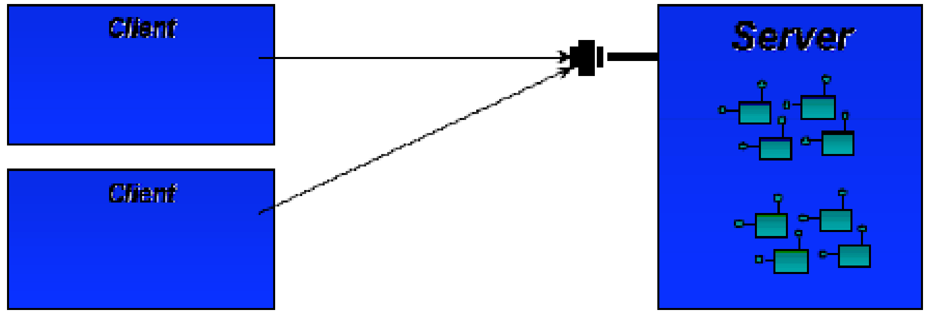


DCOM security



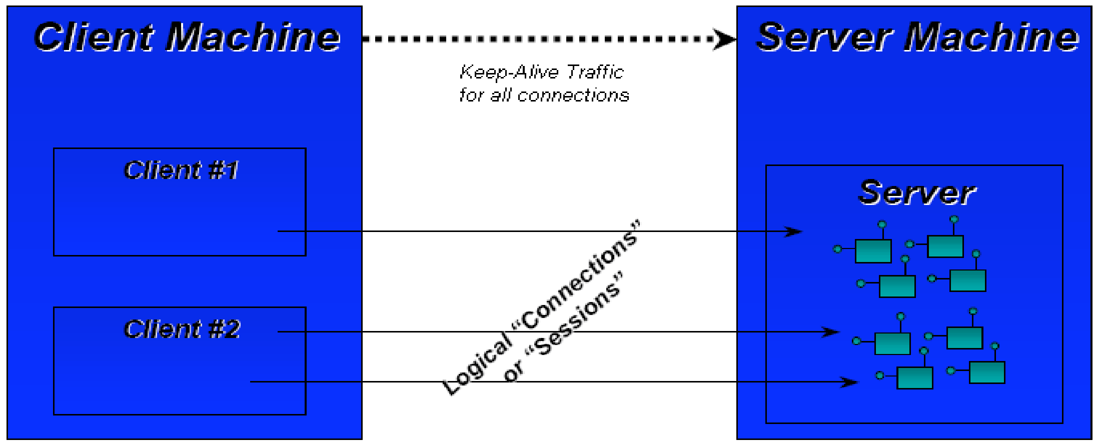
DCOM architecture (1)

- Multiplexing - Single Port per-protocol, per server process, regardless of # of objects
- Scalable - Connection-Less Protocols like UDP Preferred
- Established Connection-Oriented (TCP) Sessions Reused by same client



DCOM architecture (2)

- Low bandwidth
  - Header is 28 bytes over DCE-RPC
  - Keep-Alive Messages bundled for all connections between machines


Client Machine
Server Machine
Keep-Alive Traffic for all connections
Client #1
Client #2
Server
Logical "Connections" or "Sessions"

What’s right with COM?

- Focus is on binary object standard and scalable/fine-grained component re-use
- Concreteness and depth of definition, for example security, lifetime management, activation, installation & deployment
- Architected extensibility

What’s wrong with CORBA/IIOP?

- Focus is on cross-node or network reuse/integration
  - in practice useful for vertical solutions, not horizontal reuse/integration
- Incomplete specification
  - marshaling format of certain types of data-structures
  - implications of lack of services (e.g. Naming, Events, Lifetime management)
- No architected extensibility

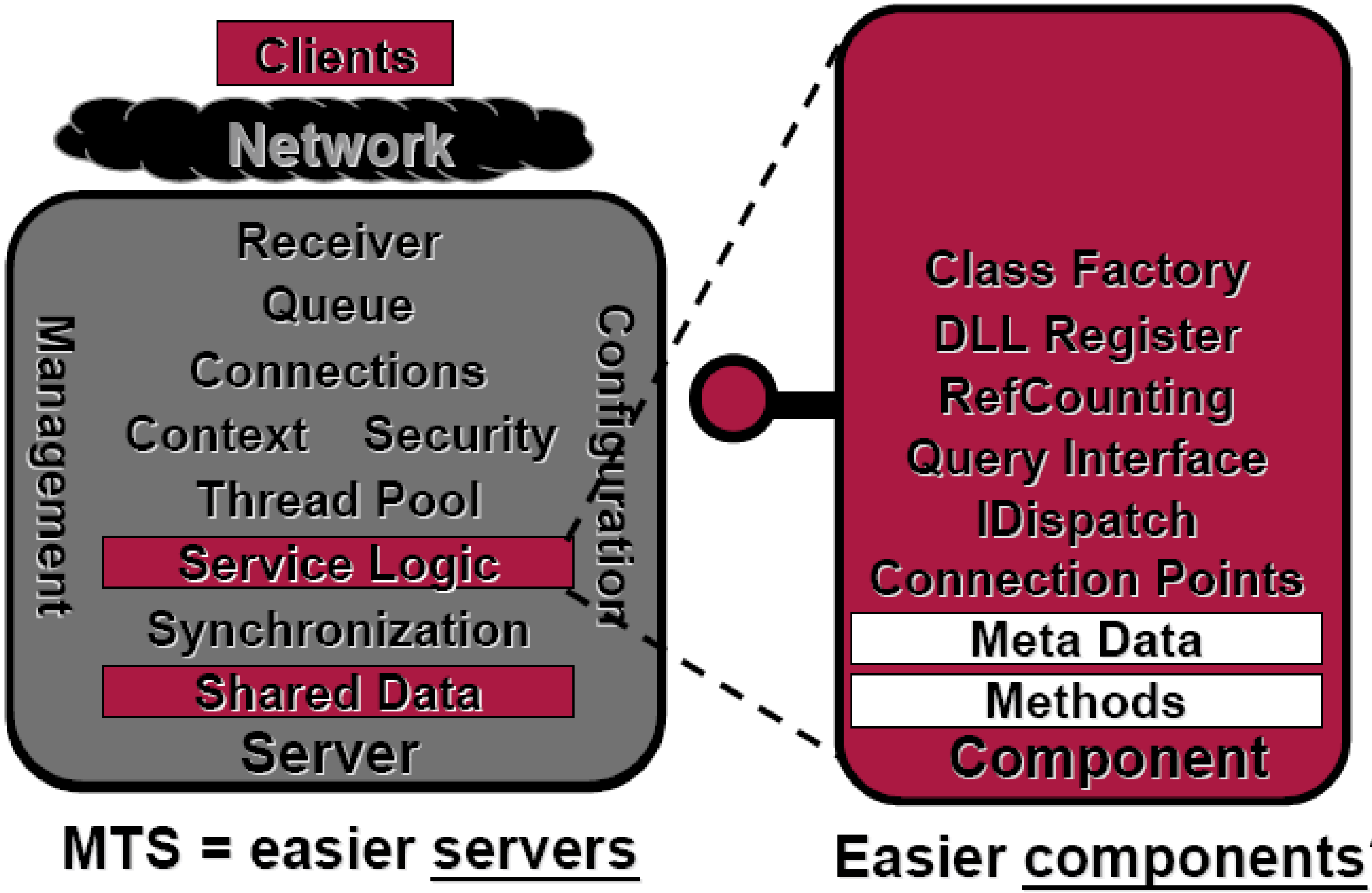
Application Management

- Distribution of Code + Data + Configuration Information
- Security and Security Delegation
  - Security “roles” and re-use of components
- Performance Monitoring
- Runtime Environment

Ease-of-Use

- What’s the next programming model layer to vastly improve ease-of-use?
  - Transactions?
  - Auto-caches & state management?
  - Auto-distribution & -execution?

Ease-of-Use: first steps



MTS = easier <u>servers</u>

Easier <u>components</u>?

COM / DCOM Reading list

•[Box1 97]
D. Box, Q&A ActiveX/COM, *Microsoft Systems Journal*, March 1997, pp. 93-105.

•[Box2 97]
D. Box, Q&A ActiveX/COM, *Microsoft Systems Journal*, July 1997, pp. 93-108.

•[Brockschmidt 93]
K. Brockschmidt, Inside OLE 2, Redmond, Washington: Microsoft Press, 1993.

•[Brown 96]
N. Brown, C. Kindel, Distributed Component Object Model Protocol -- DCOM/1.0
http://ds1.internic.net/internet-drafts/draft-brown-dcom-v1-spec-01.txt

•[Chappell 96]
D. Chappell, Understanding ActiveX and OLE, Redmond, Washington: Microsoft Press, 1996.

•[COM 95] The Component Object Model Specification,
http://www.microsoft.com/oledev/olecom/title.htm

•[DCE 95]
AES/Distributed Computing - Remote Procedure Call, Revision B, Open Software Foundation,

•http://www.osf.org/mall/dce/free_dce.htm

•[Rogerson 96]
D. Rogerson, Inside COM, Redmond, Washington: Microsoft Press, 1996.

•[Wang 97]
Y. M. Wang, COM/DCOM Resources,
http://www.research.att.com/~ymwang/resources/resources.htm

Problems with previous solutions

- RPC, CORBA, DCOM, even Java, use different mechanisms and protocols for communicating. All of them map to TCP or UDP one way or another, but use different syntax for marshalling, serializing and packaging messages
  - The problem is that these mechanisms are a legacy from the time when communications were mostly within LANs and within homogeneous systems
  - Building a B2B environment combining the systems of different companies becomes difficult because the protocols available in RPC, CORBA, or DCOM are too low level and not compatible among each other (gateways are needed, etc.)

The SOAP solution

- To address this problem, XML was used to define SOAP
  - SOAP is conceptually quite simple: RPC using HTTP
  - (at the client) turn an RPC call into an XML document
  - (at the server) turn the XML document into a procedure call
  - (at the server) turn the procedure's response into an XML document
  - (at the client) turn the XML document into the response to the RPC
  - use XML to serialize the arguments following the SOAP specification

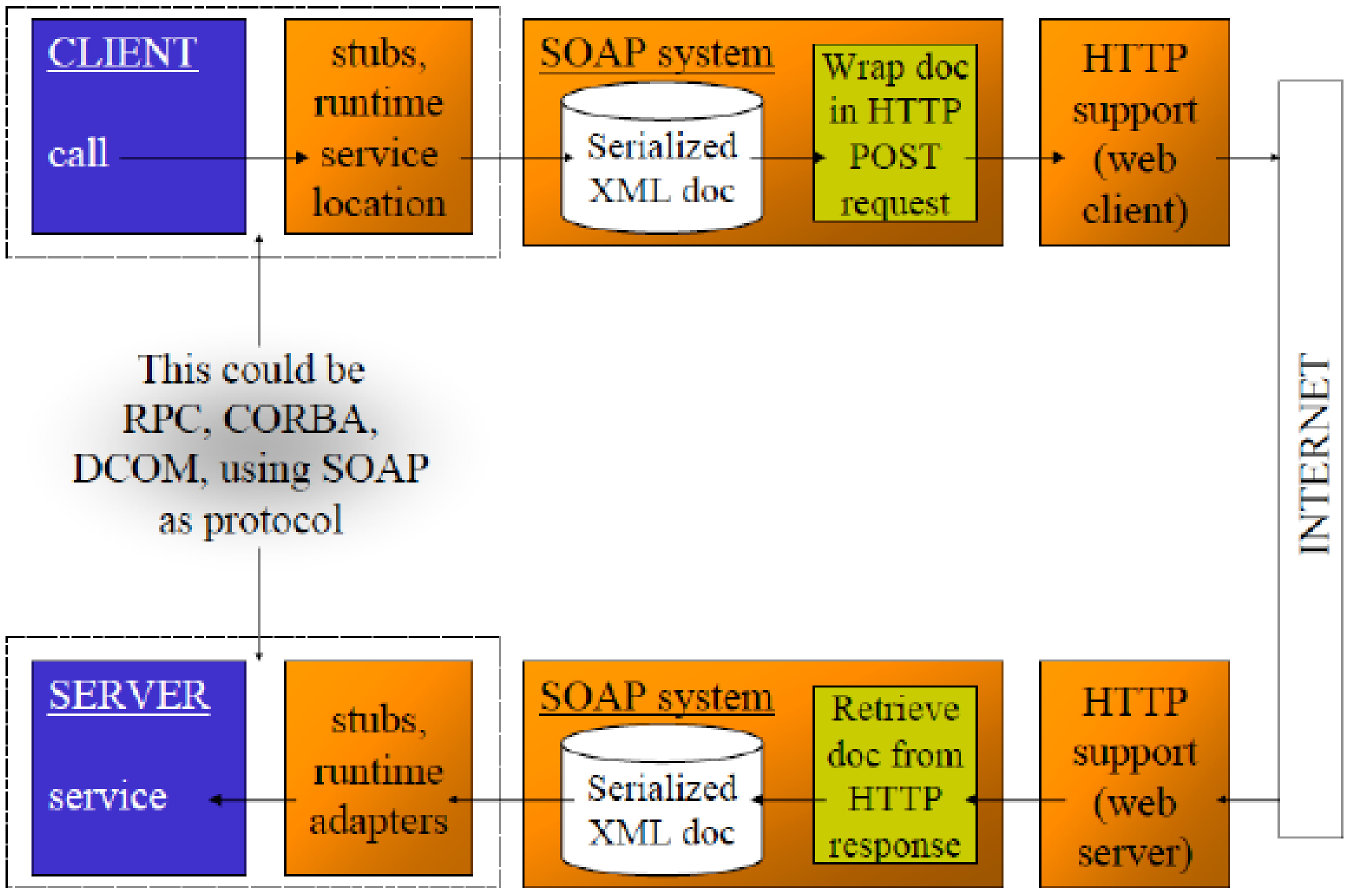
SOAP background (1)

- SOAP was originally conceived as the minimal possible infrastructure necessary to perform RPC through the Internet: use of XML as intermediate representation between systems
  - very simple message structure
  - mapping to HTTP for tunneling through firewalls and using the Web infrastructure

SOAP background (2)

- The idea was to avoid the problems associated with CORBA’s IIOP/GIOP (which fulfilled a similar role but using a non-standard intermediate representation and had to be tunneled through HTTP anyway)
  - The goal was to have an extension that could be easily plugged on top of existing middleware platforms to allow them to interact through the Internet rather than through a LAN as in the original case. Hence the emphasis on RPC from the very beginning (essentially all forms of middleware use RPC at one level or another)
- Eventually SOAP started to be presented as a generic vehicle for computer driven message exchanges through the Internet and then it was opened to support interactions other than RPC and protocols other then HTTP

SOAP invocation


CLIENT
call
stubs, runtime service location
SOAP system
Serialized XML doc
Wrap doc in HTTP POST request
HTTP support (web client)
INTERNET
This could be RPC, CORBA, DCOM, using SOAP as protocol
SERVER
service
stubs, runtime adapters
SOAP system
Serialized XML doc
Retrieve doc from HTTP response
HTTP support (web server)

SOAP history (1)

- The W3C started working on SOAP in 1999.

Originally: Simple Object Access Protocol

- SOAP covers the following main areas:
  - Message construct: a message format for one-way communication describing how a message can be packed into an XML document
  - Processing model: rules for processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Which parts of the messages should be read by whom and how to react in case the content is not understood
  - Extensibility model: how the basic message construct can be extended with application specific constructs

SOAP history (2)

- Protocol binding framework: allows SOAP messages to be transported using different protocols (HTTP, SMTP, …)
  - a concrete binding for HTTP
  - conventions on how to turn an RPC call into a SOAP message and
  - back as well as how to implement the RPC style of interaction

SOAP facts (1)

- SOAP is “a lightweight protocol intended for exchanging structured information […]”, “a stateless, one-way message exchange paradigm”
  - defines the general format of a message and how to process it
  - RPC is implemented on top of the core specification following conventions of the “SOAP RPC representation”

SOAP facts (2)

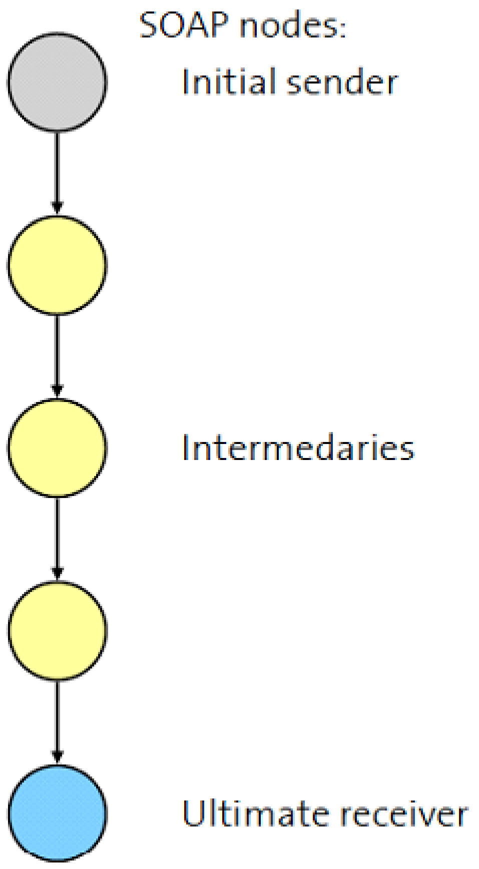
- SOAP ≠ RPC: since Version 1.1, SOAP abstracts from the RPC programming model
- SOAP ≠ HTTP: since Version 1.1, SOAP abstracts from the protocol used to transport the messages
  - HTTP is one of many possible transports

SOAP message path (1)

- A SOAP message can pass through multiple hops on the way from the initial sender to the ultimate receiver
- The entities involved in transporting the message are called SOAP nodes
- SOAP intermediaries forward the message and may manipulate it

SOAP message path (2)

- Every SOAP node assumes a certain role which influences the message processing at the node


SOAP nodes:
Initial sender
Intermedaries
Ultimate receiver