Distributed Systems

Distributed Object-Based Systems

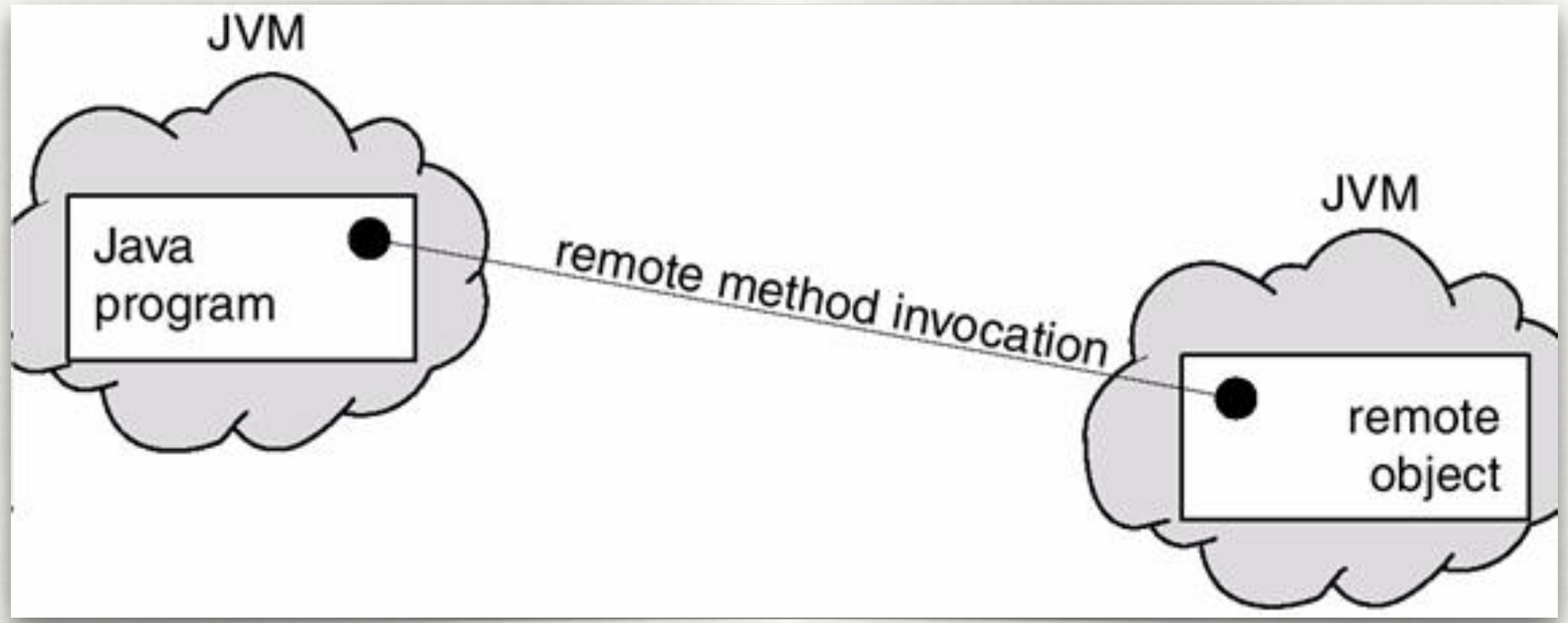
Björn Franke
University of Edinburgh, 2016

OVERVIEW

- Basic Concepts
- Middleware Technologies
  - Apache Ignite (see coursework lecture)
  - CORBA
  - DCOM
  - .NET Remote Procedure Calls/Windows Communication Foundation
  - Google gRPC

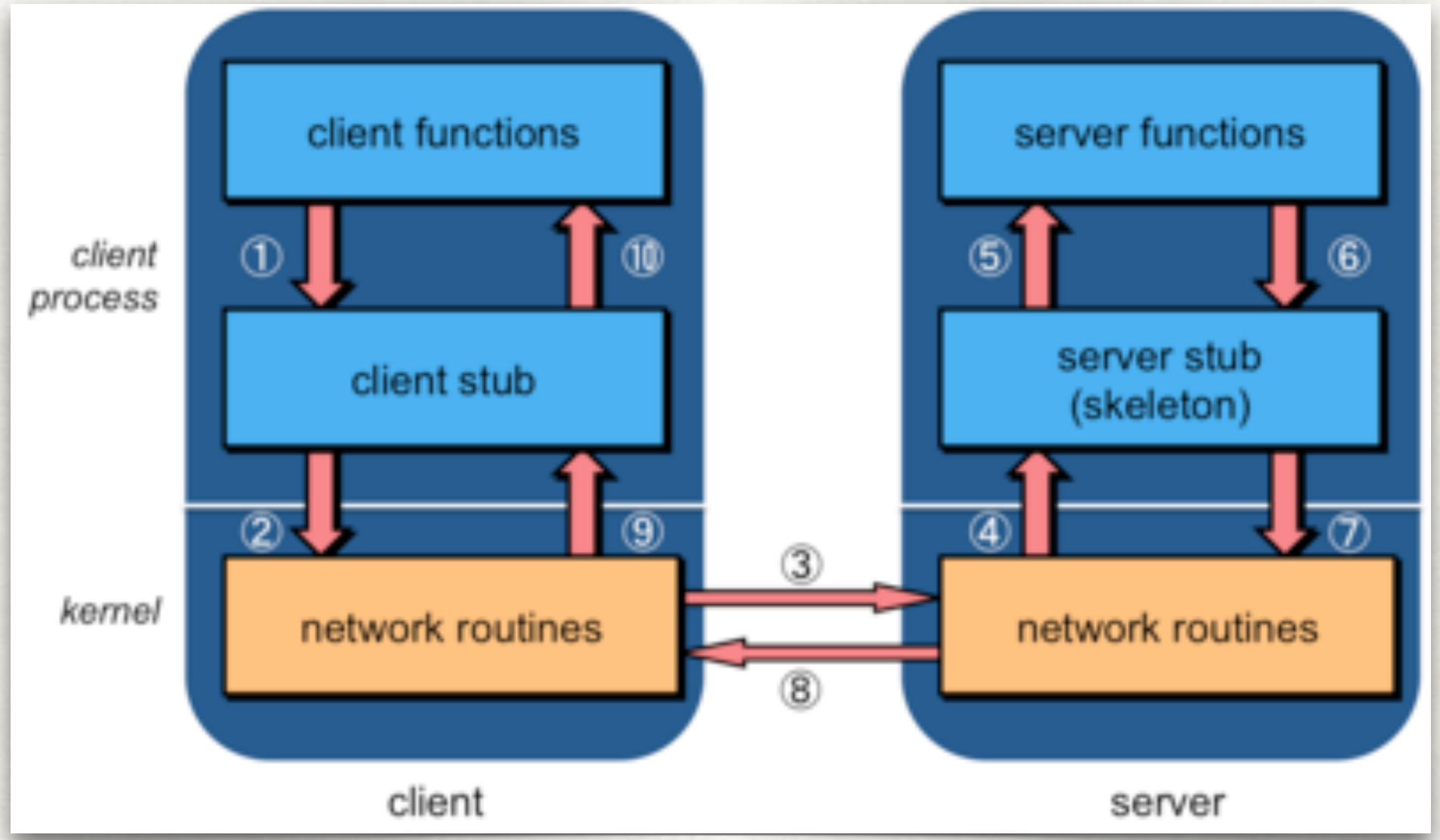
BASIC CONCEPTS

REMOTE PROCEDURE CALLS (RPC)



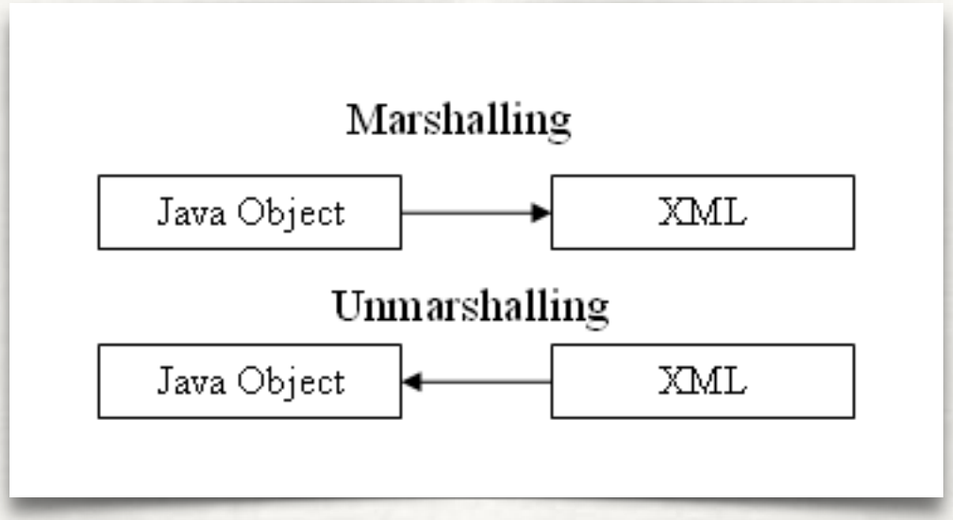
BASIC CONCEPTS

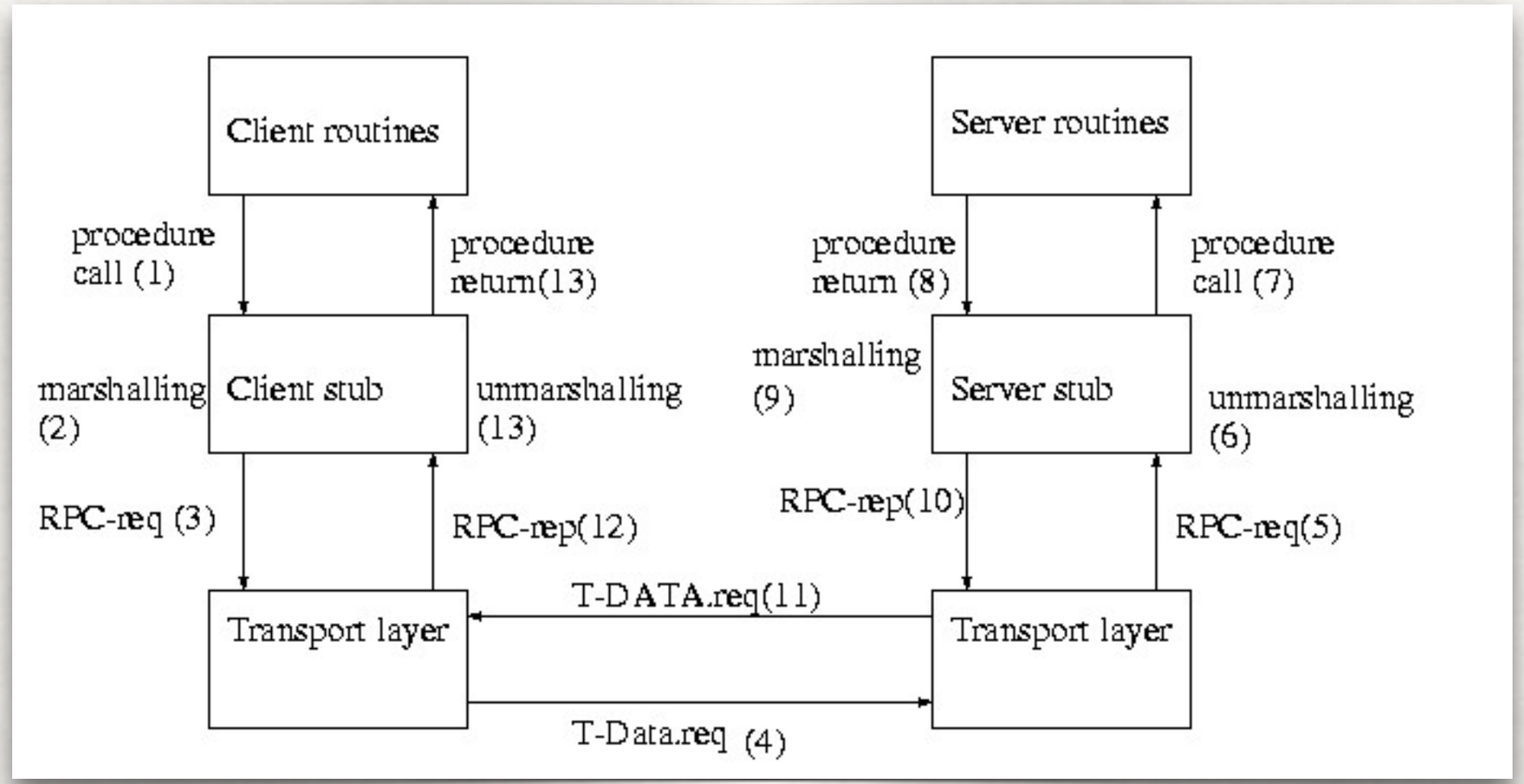
REMOTE PROCEDURE CALLS (RPC)



BASIC CONCEPTS

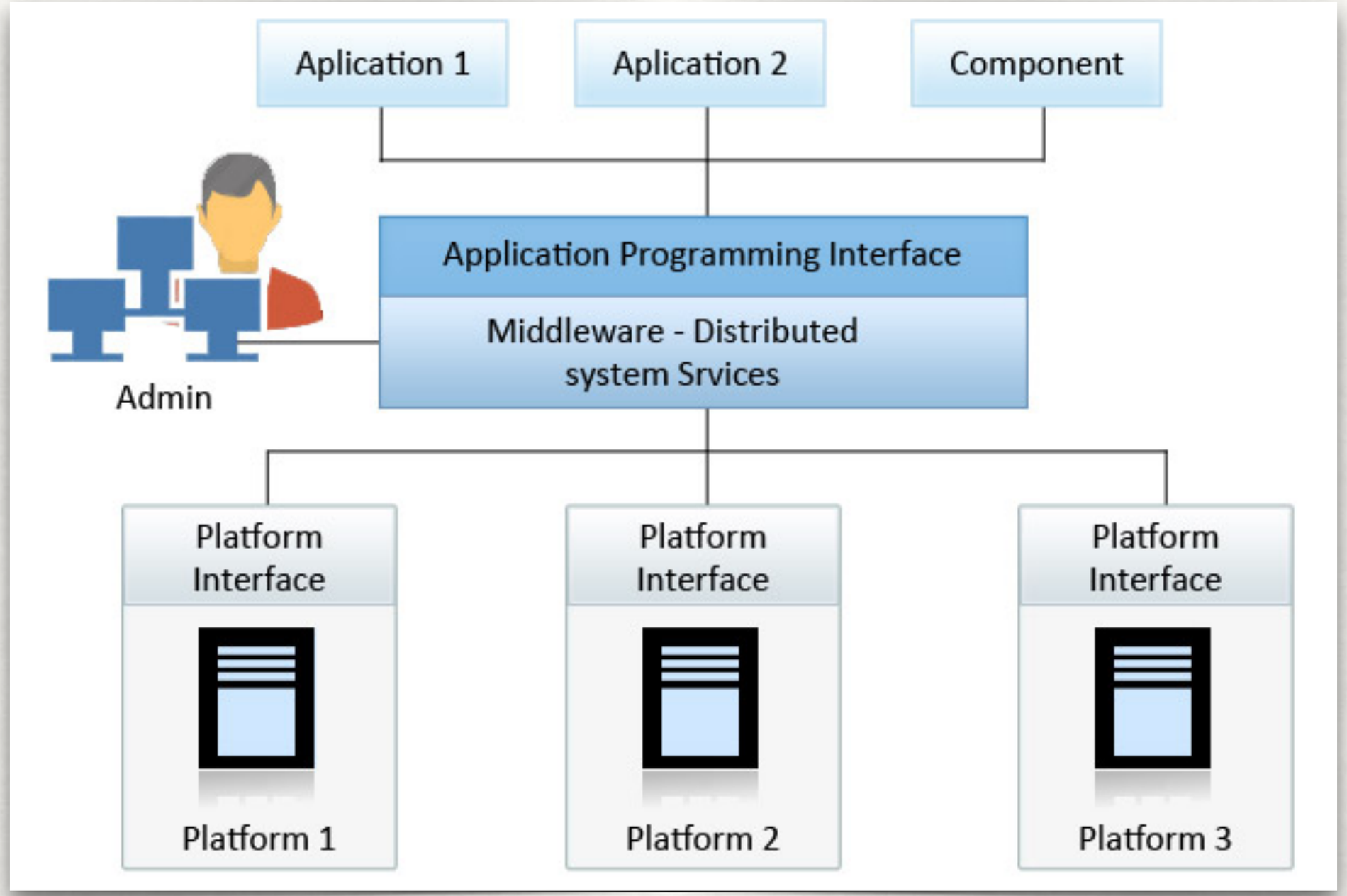
MARSHALLING/UNMARSHALLING





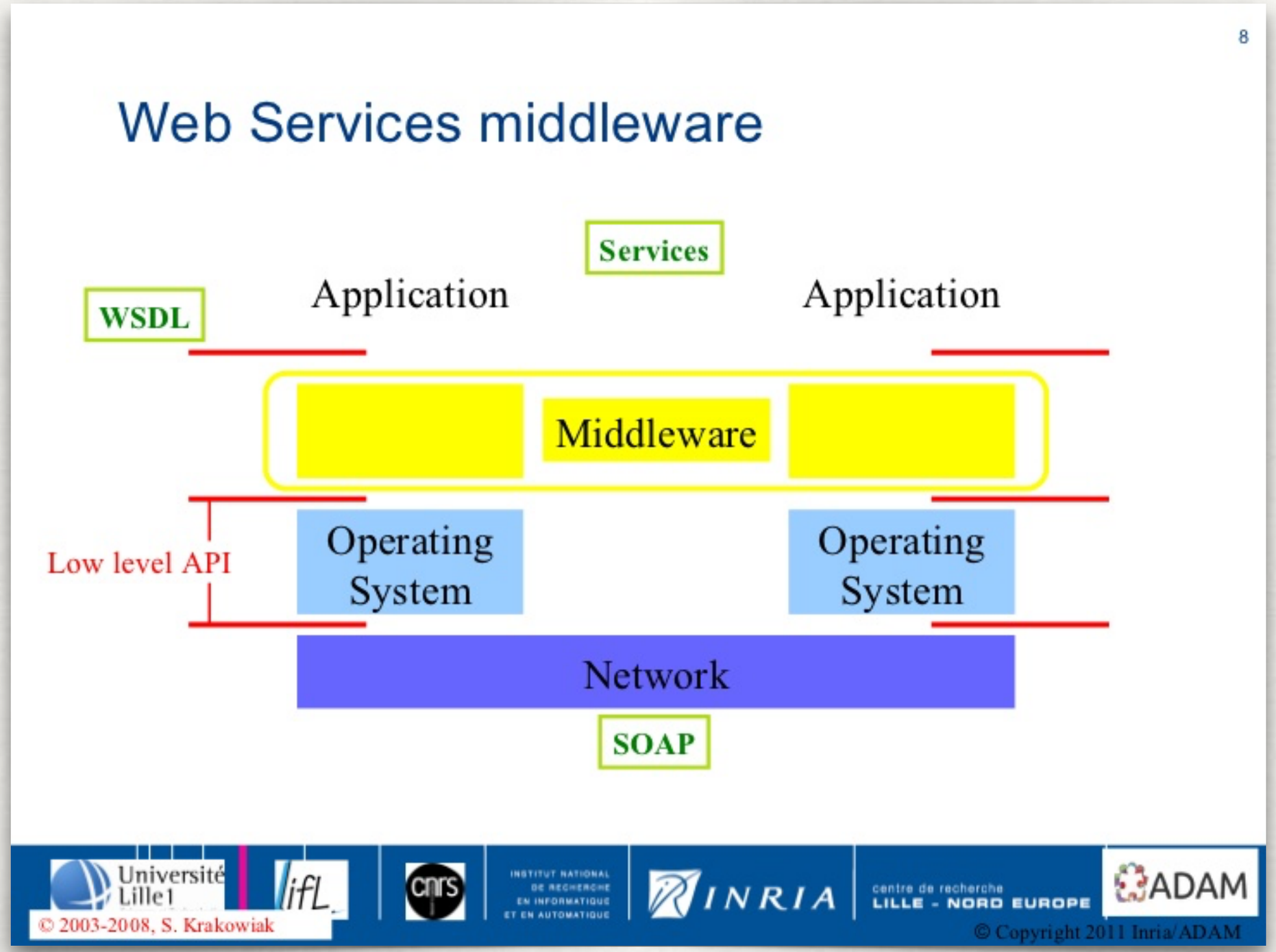
MIDDLEWARE TECHNOLOGIES

HIGH-LEVEL VIEW



MIDDLEWARE TECHNOLOGIES

SLIGHTLY MORE DETAIL



- High-level abstractions & API
- Heterogeneity Hidden
- Transparent Distribution
- General purpose services e.g. directory/naming

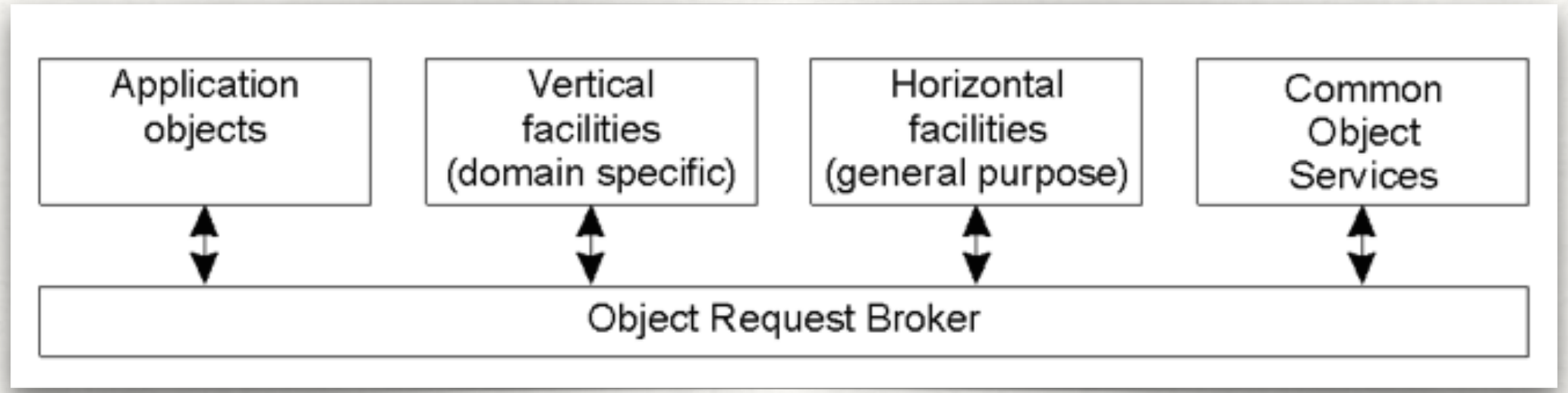
CORBA

COMMON OBJECT REQUEST BROKER ARCHITECTURE

- **Industry-defined standard** for distributed objects
- Enables collaboration between systems on different operating systems, programming languages, and computing hardware
- **Interface definition language (IDL)** to specify object interfaces. CORBA then specifies a mapping from IDL to a specific implementation language.
- Central: **Object request brokers (ORBs)**
  - Application initialises ORB, and accesses an internal **Object Adapter**, which maintains things like reference counting, object (and reference) instantiation policies, and object lifetime policies.
  - **Object Adapter** is used to register instances of the generated code classes (result of compiling the user IDL code, which translates interface definitions into an OS- and language-specific class base for use by the user application).

CORBA

GLOBAL ARCHITECTURE



- The **Object Request Broker (ORB)** forms the core of any CORBA distributed system.
- **Horizontal facilities** consist of general-purpose high-level services that are independent of application domains.
  - User interface
  - Information management
  - System management
  - Task management
- **Vertical facilities** consist of high-level services that are targeted to a specific application domain such as electronic commerce, banking, manufacturing.

CORBA

OBJECT MODEL

- CORBA has a traditional remote object model in which an object residing at an object server is remote accessible through proxies
- All CORBA specifications are given by means of interface descriptions, expressed in an **Interface Definition Language (IDL)**.
  - An interface is a collection of methods, and objects specify which interfaces they implement.
  - It provides a precise syntax for expressing methods and their parameters.
- (In DCOM, interfaces can be specified at a lower level in the form of tables, called **binary interfaces**.)

CORBA

OBJECT MODEL

- **Object Request Broker (ORB):** CORBA's object broker that connects clients, objects, and services
- **Proxy/Skeleton:** Precompiled code that takes care of (un)marshalling invocations and results
- **Dynamic Invocation/Skeleton Interface (DII/DSI):** To allow clients to construct invocation requests at runtime instead of calling methods at a proxy, and having the server side reconstruct those request into regular method invocations
- **Object adapter:** Server side code that handles incoming invocation requests.

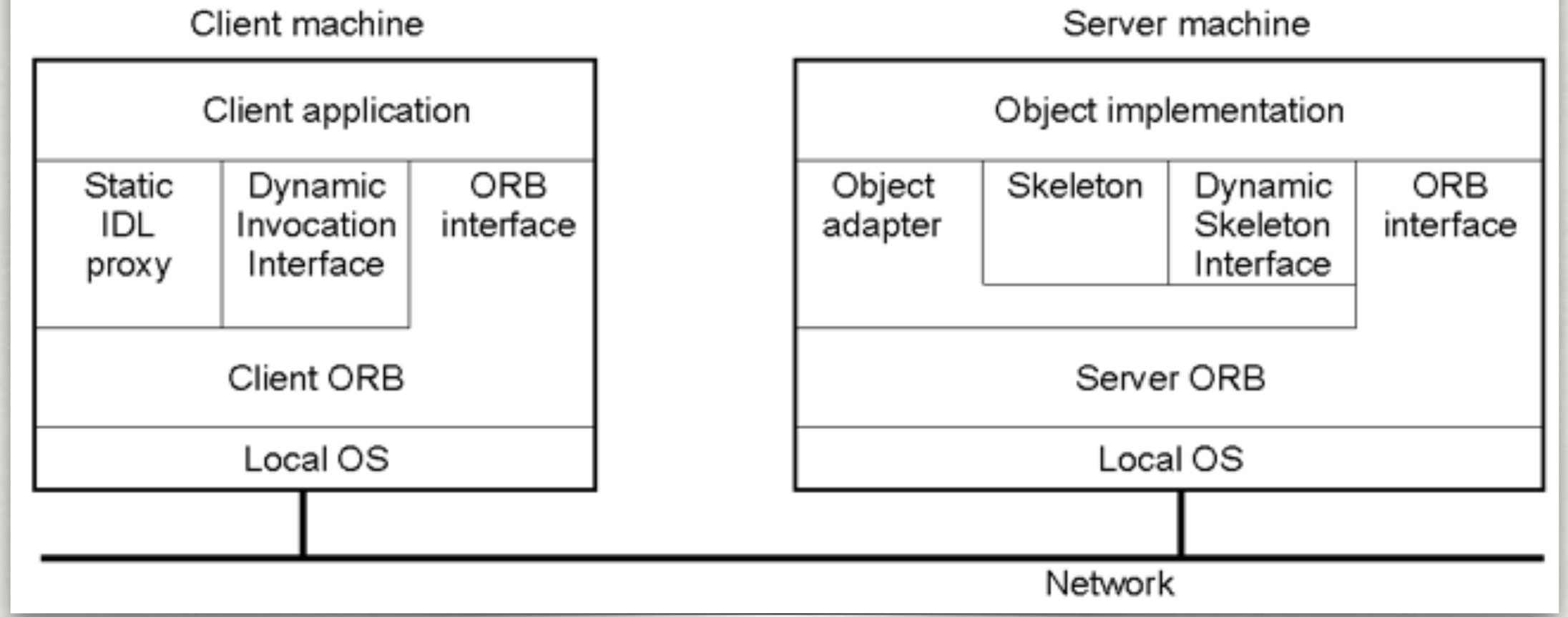
CORBA

OBJECT MODEL

- **Interface repository:**
  - Database containing interface definitions and which can be queried at runtime (similar in purpose to Java Reflection)
  - Whenever an interface definition is compiled, the IDL compiler assigns a repository identifier to that interface.
- **Implementation repository:**
  - Database containing the implementation (code, and possibly also state) of objects.
  - Given an object reference, an object adaptor could contact the implementation repository to find out exactly what needs to be done.

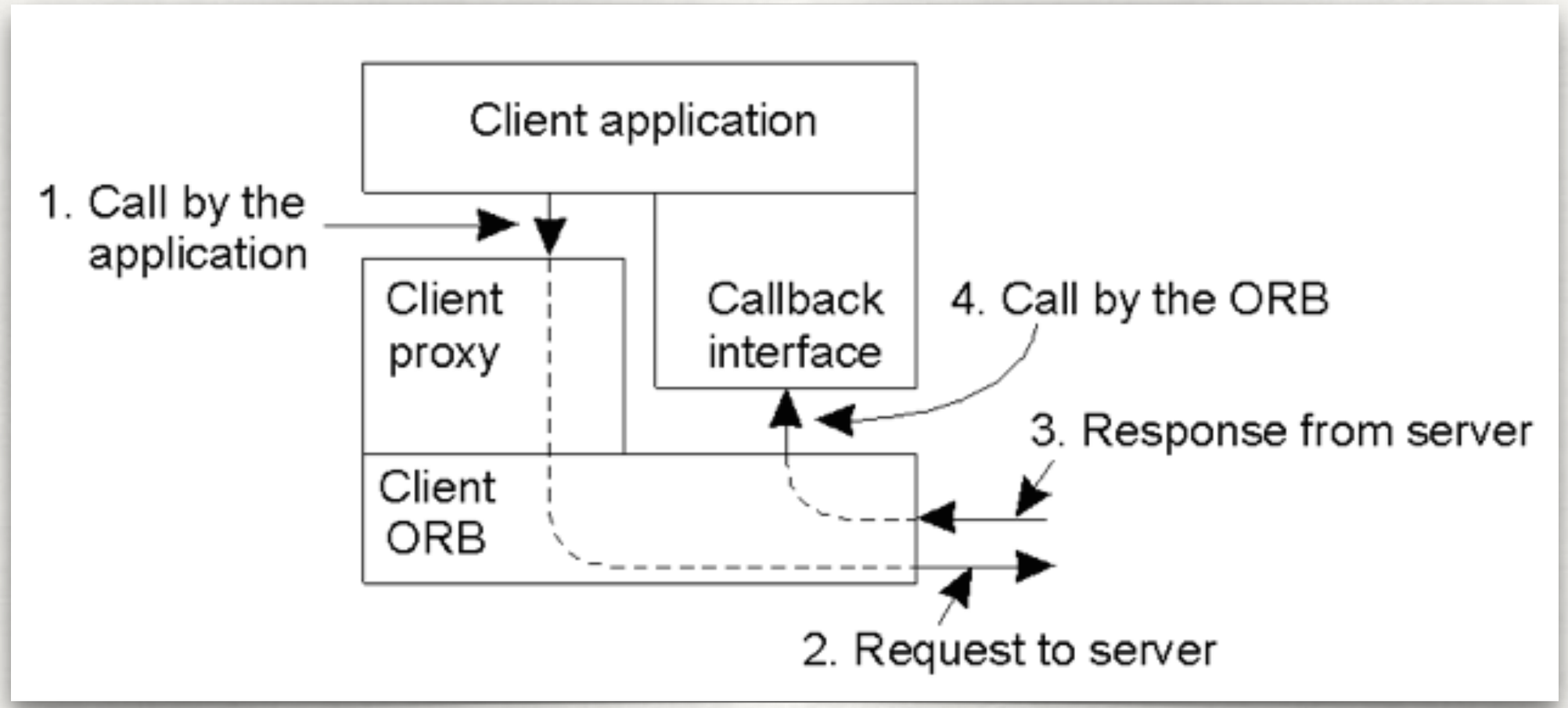
CORBA

GENERAL ORGANISATION



CORBA

MESSAGING



COM

COMPONENT OBJECT MODEL TECHNOLOGIES

- Microsoft COM (**Component Object Model**) technology in the Microsoft Windows-family of Operating Systems enables software components to communicate.

- COM is used by developers to create re-usable software components, link components together to build applications, and take advantage of Windows services.

- The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX® Controls.

- For new development, Microsoft recommends .NET as a preferred technology because of its powerful managed runtime environment and services.

HOW ARE COM AND .NET RELATED?

- COM and .NET are complimentary development technologies.
- COM and .NET applications and components can use functionality from each system.
- COM and .NET can achieve similar results. The .NET Framework provides developers with a significant number of benefits including a more robust, evidence-based security model, automatic memory management and native Web services support.

WHAT IS COM+?

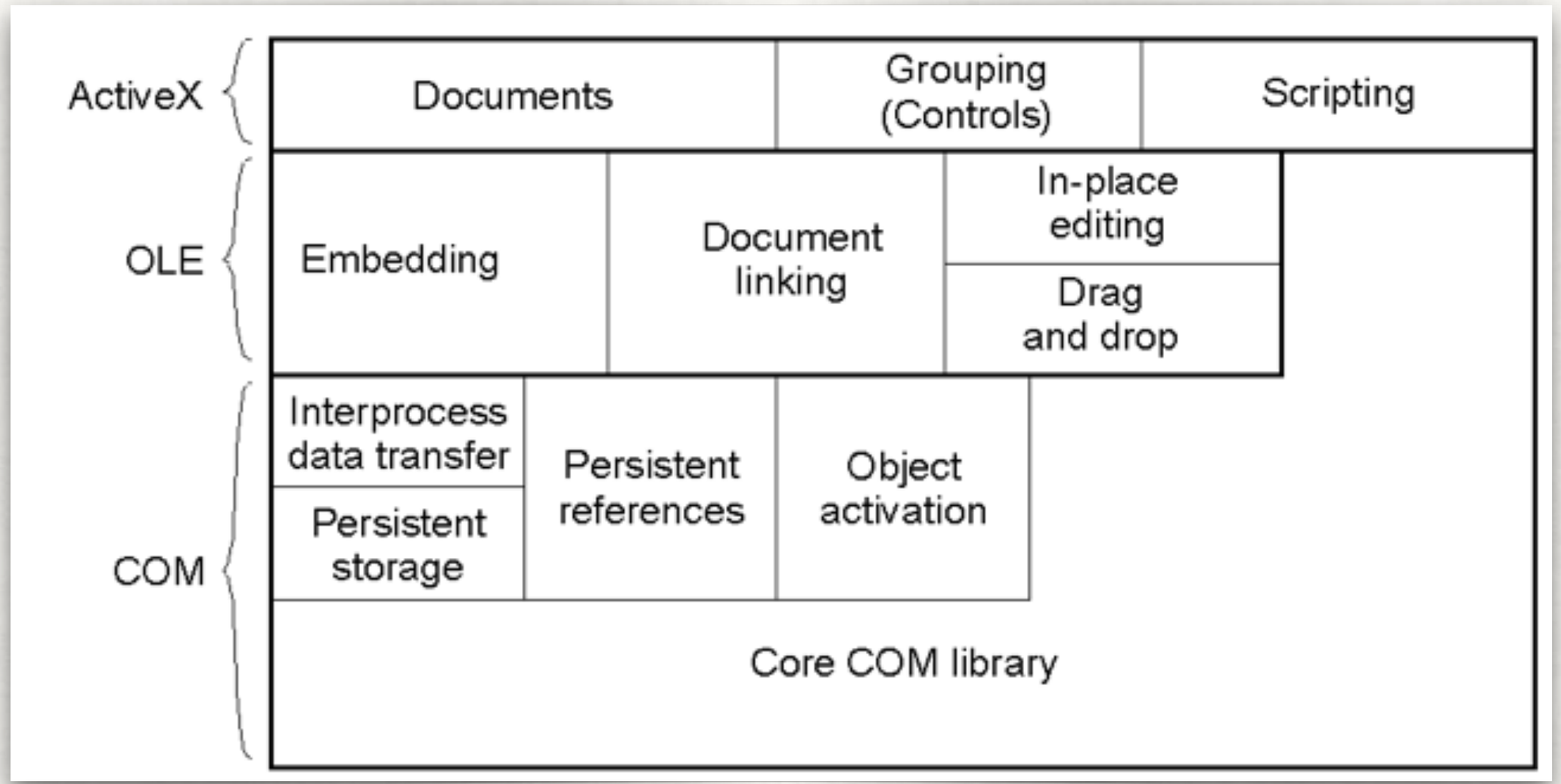
- COM+ is the name of the COM-based services and technologies first released in Windows 2000.
- COM+ brought together the technology of COM components and the application host of Microsoft Transaction Server (MTS).
- COM+ automatically handles difficult programming tasks such as resource pooling, disconnected applications, event publication and subscription and distributed transactions.
- COM+ infrastructure also provides services to .NET developers and applications through the System.EnterpriseServices namespace of the .NET Framework.

DCOM

DISTRIBUTED COM

- Microsoft's solution to establishing **inter-process communication**, possibly **across machine boundaries**.
- DCOM uses the **RPC mechanism** to transparently send and receive information between COM components (i.e., clients and servers) on the same network.
- Supports a primitive notion of distributed objects
- Evolved from early Windows versions to NT-based systems (including Windows 2000/XP)
- Comparable to CORBA's object request broker

OVERVIEW OF DCOM

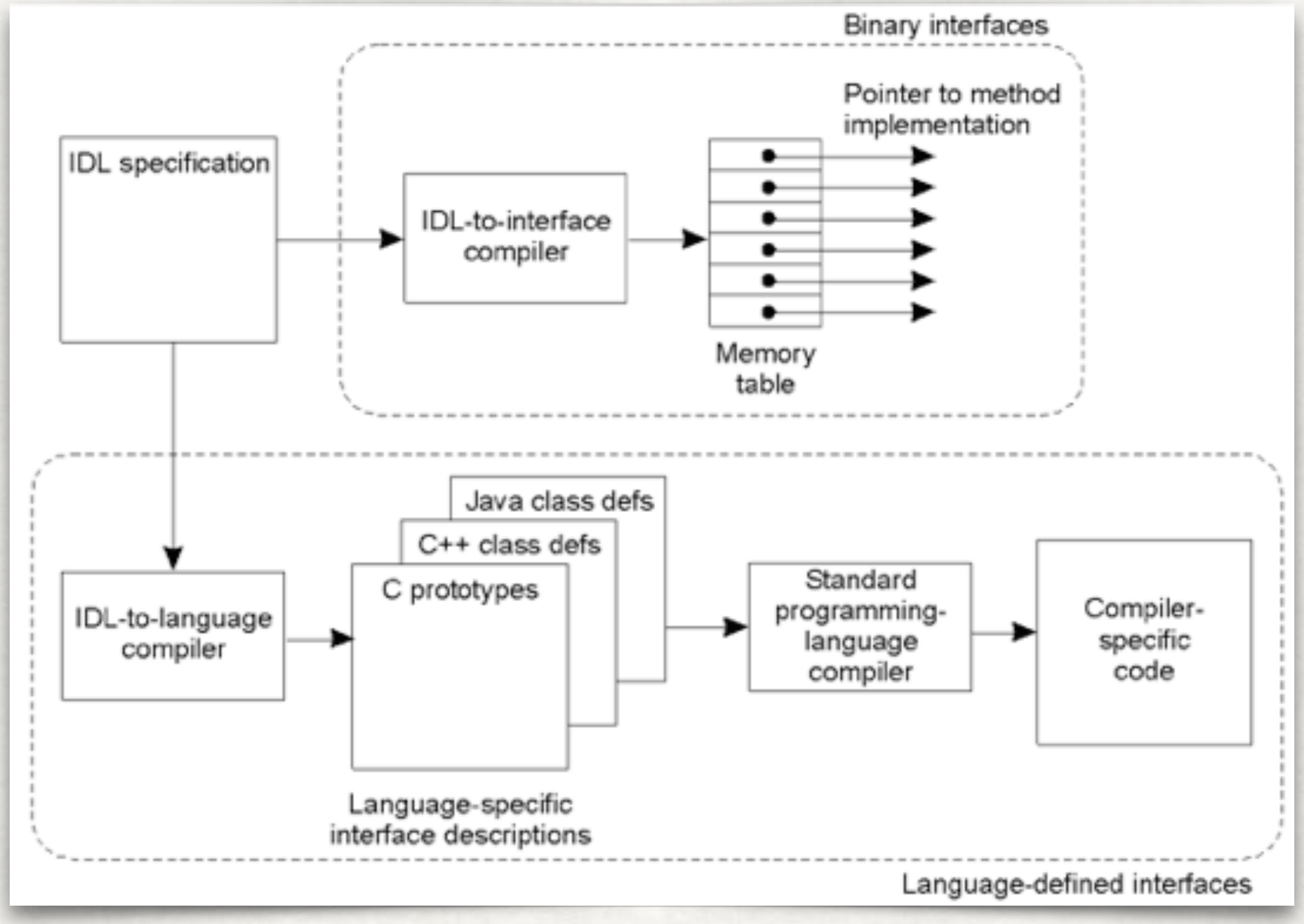


DCOM OBJECT MODEL

- An interface is a collection of **semantically related operations**
- Each interface is **typed**, and therefore has a **globally unique interface identifier**
- A client always requests an implementation of an interface:
  - Locate a class that implements the interface
  - Instantiate that class, i.e., create an object
  - Throw the object away when the client is done
- Note: COM+ is effectively COM plus services that were previously available in an ad-hoc fashion

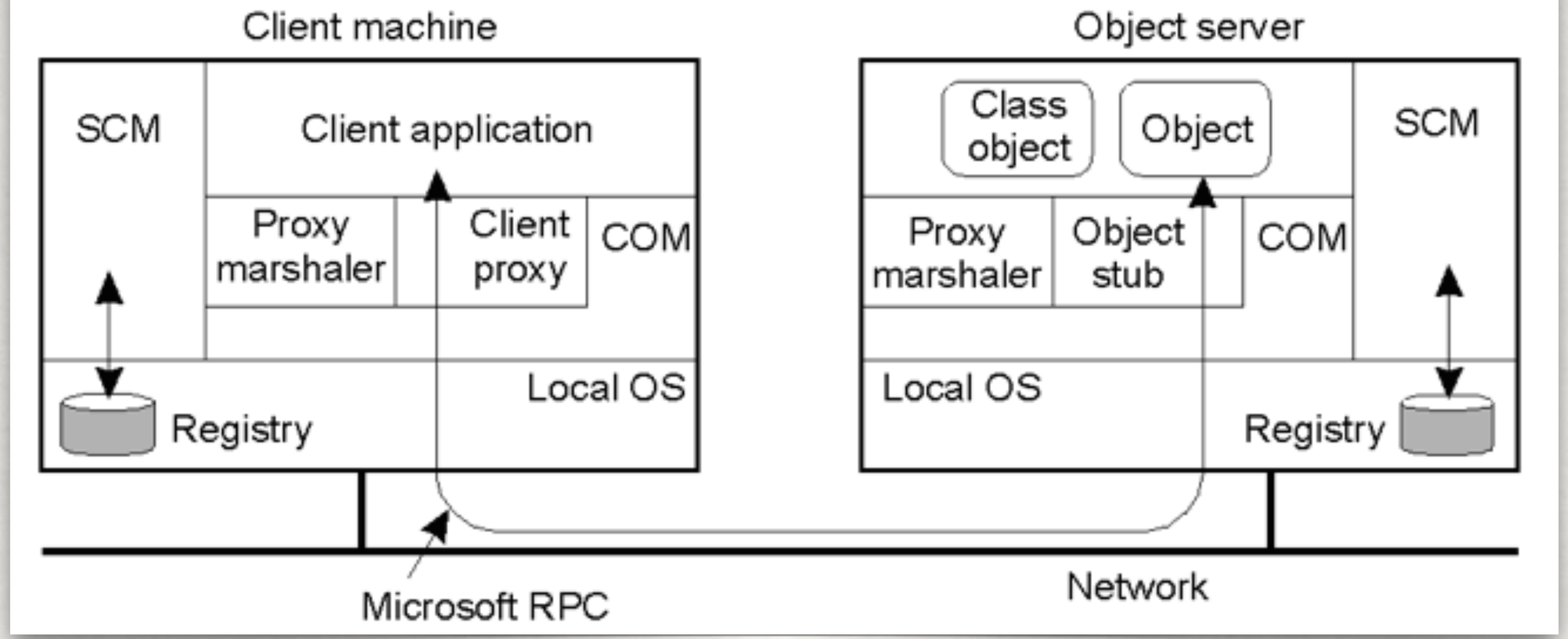
DCOM

OBJECT MODEL



DCOM

OVERALL ARCHITECTURE



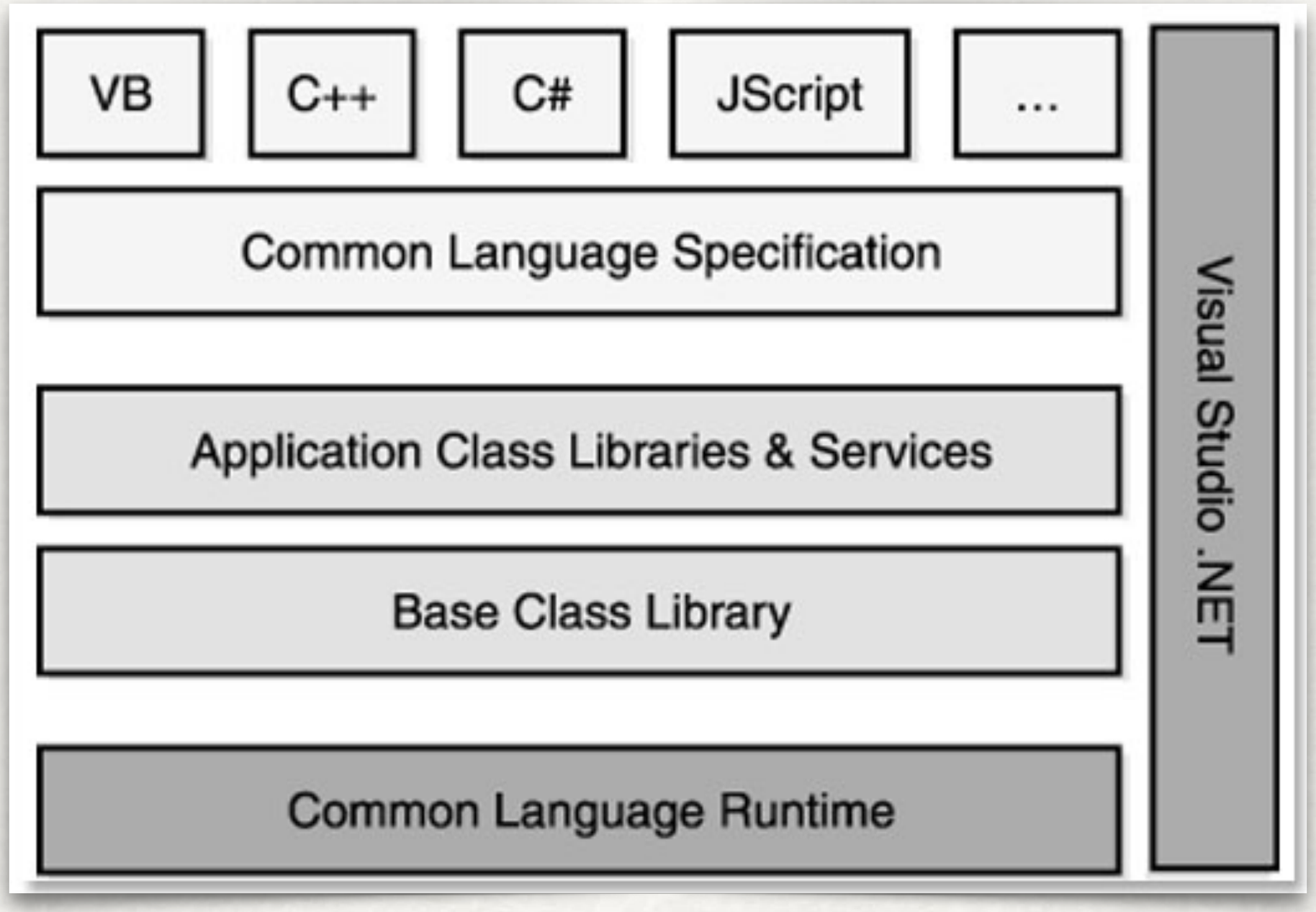
.NET

INFRASTRUCTURE

- .NET Framework: programming environment
- Web Services: .NET provides a standard syntax for input and output language that is defined for sites providing 'web services'.
- .NET Servers: Servers that work with .NET such as SQL Server
- .NET implementation
  - Windows: Microsoft .NET, Linux: Mono

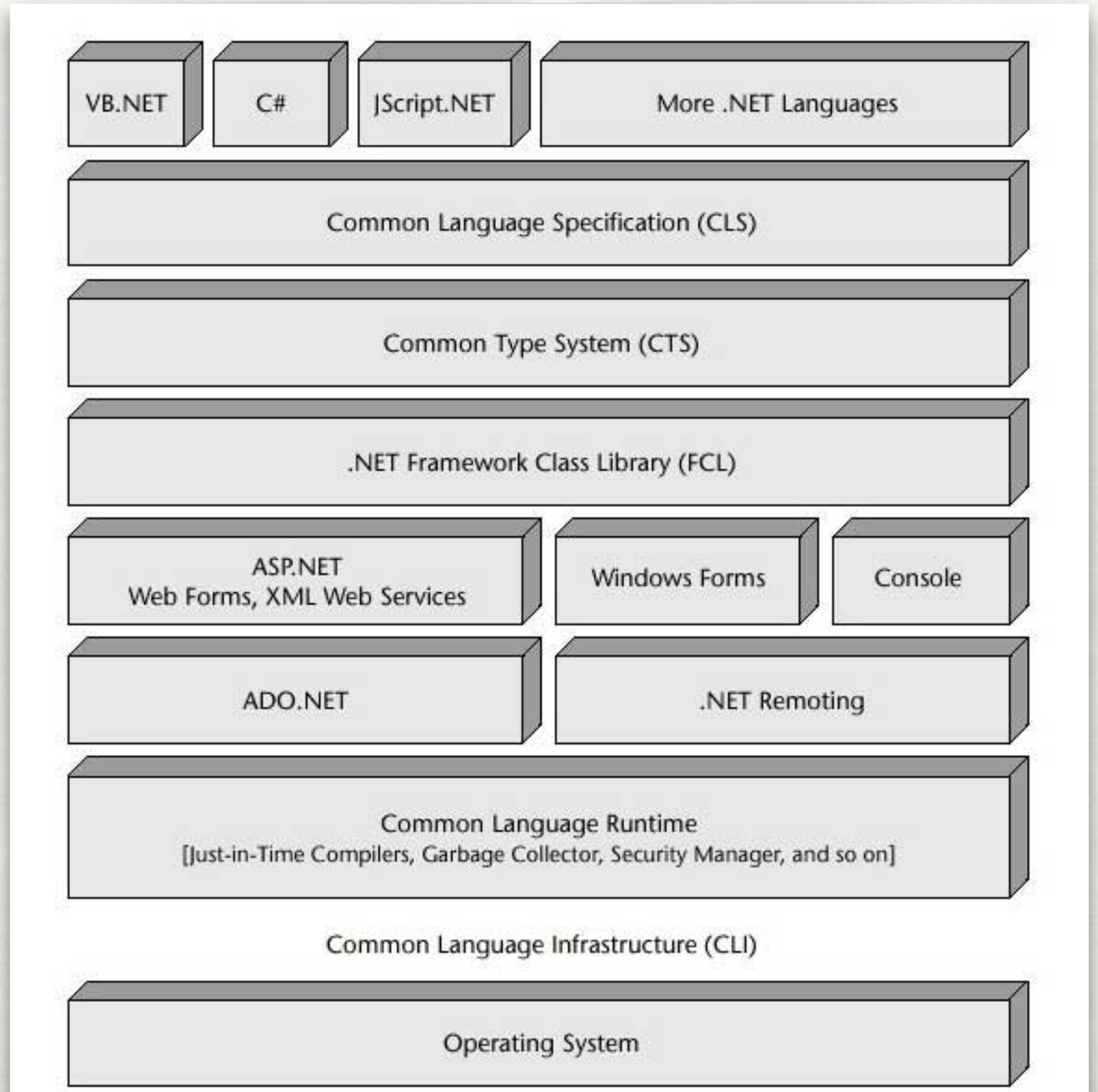
.NET

FRAMEWORK



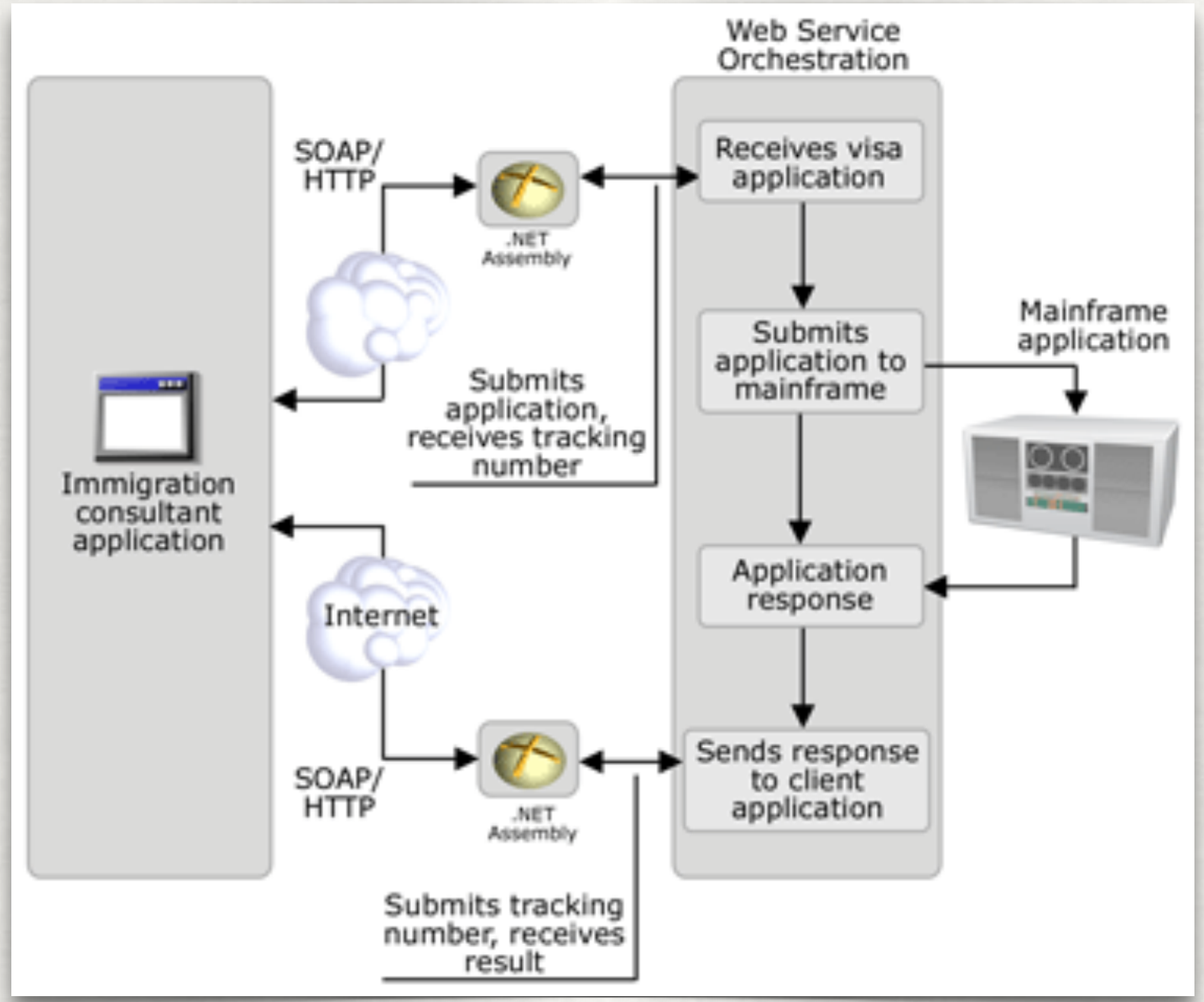
.NET

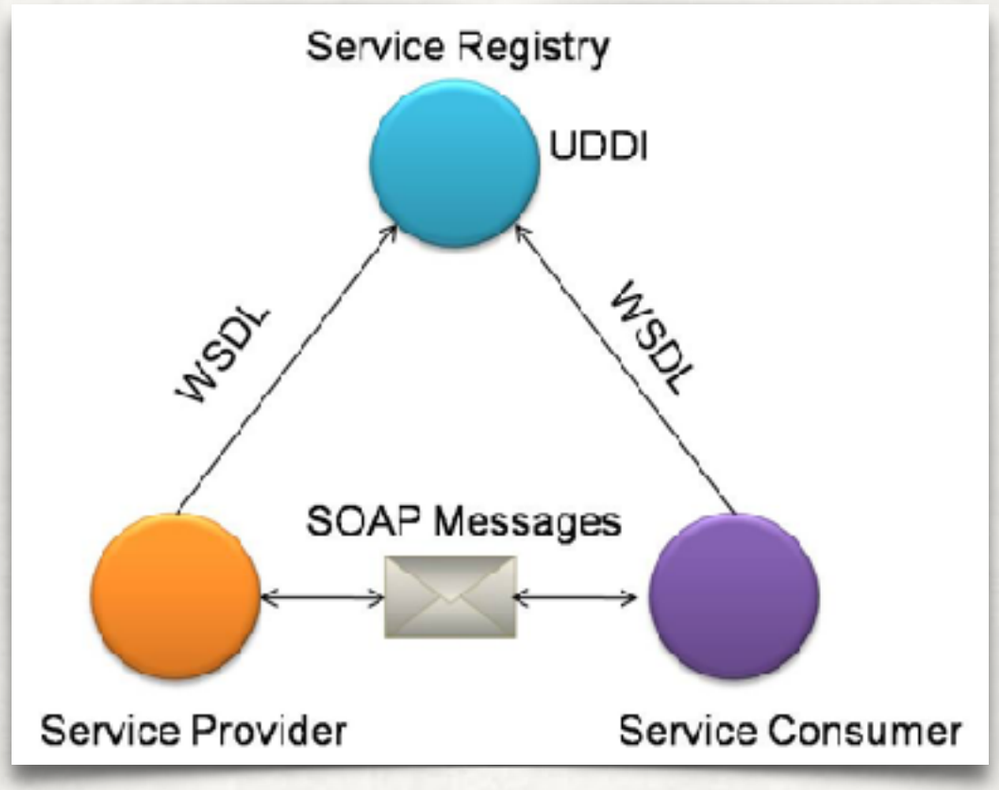
FRAMEWORK - MORE DETAIL



.NET

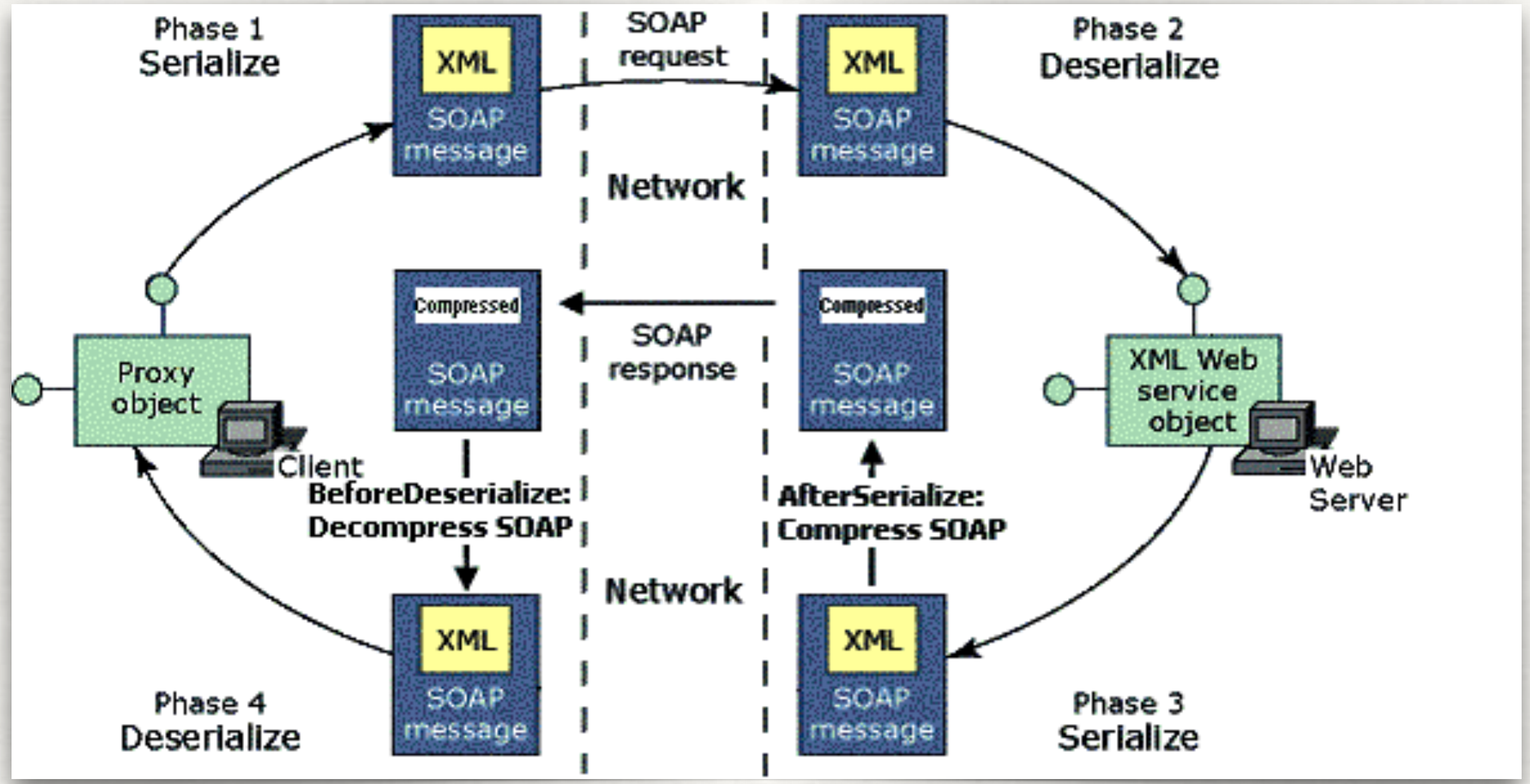
XML WEB SERVICES





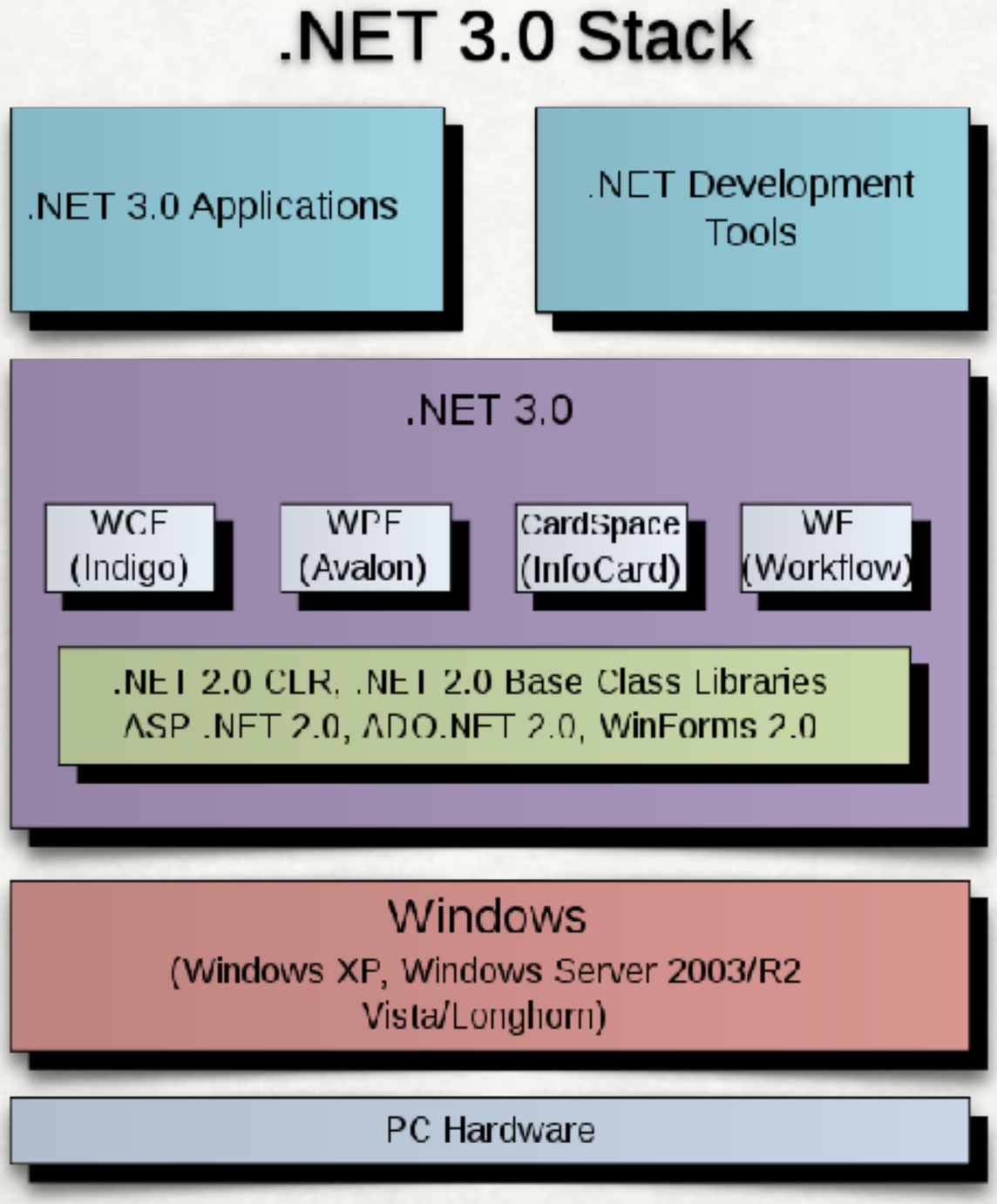
.NET

XML WEB SERVICES - SOAP



.NET

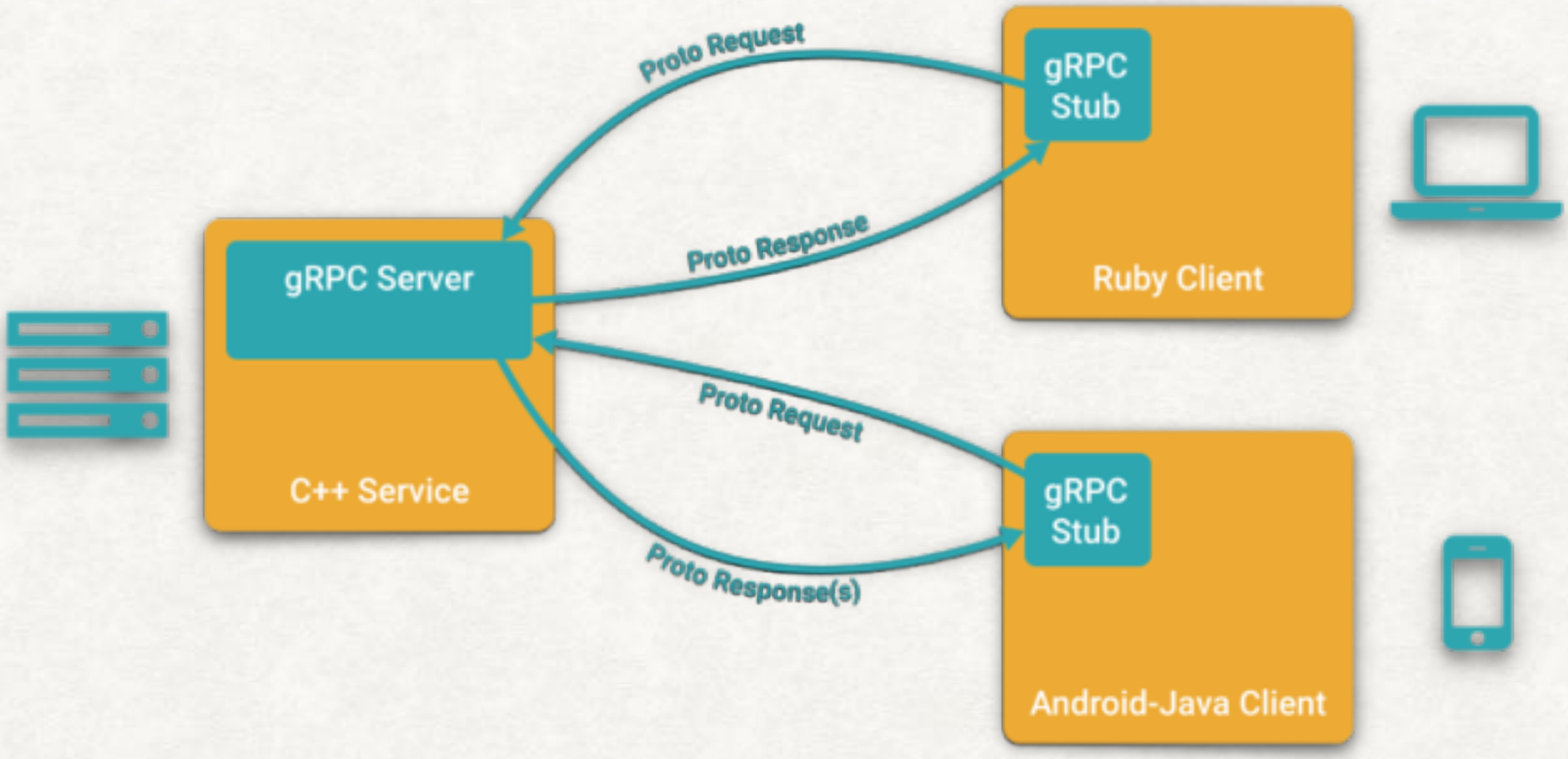
WINDOWS COMMUNICATION FOUNDATION (WCF)



- Implement/deploy a service-oriented architecture (SOA), where services have remote consumers
- Clients can consume multiple services; services can be consumed by multiple clients.
- Services are loosely coupled to each other.
- Services typically have a WSDL interface (Web Services Description Language) that any WCF client can use to consume the service
- WCF implements e.g. WS-Addressing, WS-ReliableMessaging and WS-Security, RSS Syndication Services, WS-Discovery, routing and support for REST services.

GOOGLE GRPC

BETA ANNOUNCED IN OCTOBER 2015



GOOGLE GRPC

BETA ANNOUNCED IN OCTOBER 2015

- Client application can directly call methods on a server application on a different machine as if it was a local object
- Based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types
- Server implements this interface and runs a gRPC server to handle client calls
- Client has a stub that provides exactly the same methods as the server
- Numerous languages supported (C++, Java, Go, Python, Ruby, C#, Objective-C and PHP)
- gRPC can use a variety of protocols for passing data across the wire, but the default is based on its own mechanism for serialising data, called protocol buffers, of which the latest version is called proto3