ID2208 Programming Web Services

http://people.kth.se/~misha/ID2208/index.html

Mihhail Matskin:

misha@kth.se

Spring 2016

Course info

http://ict.kth.se/courses/ID2208

Coordinator

Mihhail Matskin
misha@kth.se
tel. 08-790 41 28

Lectures & Tutorials

see Schedule

Assistants

Edward Tjörnhammar and Shatha Jaradat

Written examination (4.5 p.)

March 23 at 9-13

Registration at least 21 days before exam period

Homework and project assignments (3 p.)

Homework (preliminary schedule)

Start Date	Due Date	Description
2016-01-27	2016-02-05	Homework 1
2016-02-05	2016-02-12	Homework 2
2016-02-12	2016-02-19	Homework 3

It is assumed that the Homework are done by groups of 2 students - there will be no additional bonus for doing them alone

Project

Aims of the project

- To apply knowledge obtained during the course to design and implementation of web services

Your task

There will be suggested topic. You can also make your own project proposal for the system - the proposal must be approved by the course coordinator (the last date for approval of your own project proposal is **February 18**).

Bonus

1. Delivering all Homeworks 1,2,3 in due time gives 5 bonus points to written exam (this assumes that all Homeworks are approved)
2. Delivering project work before March 2 gives 5 bonus points to written exam (this assumes that the project work is approved)

Bonuses are only valid on the first exam

Course literature

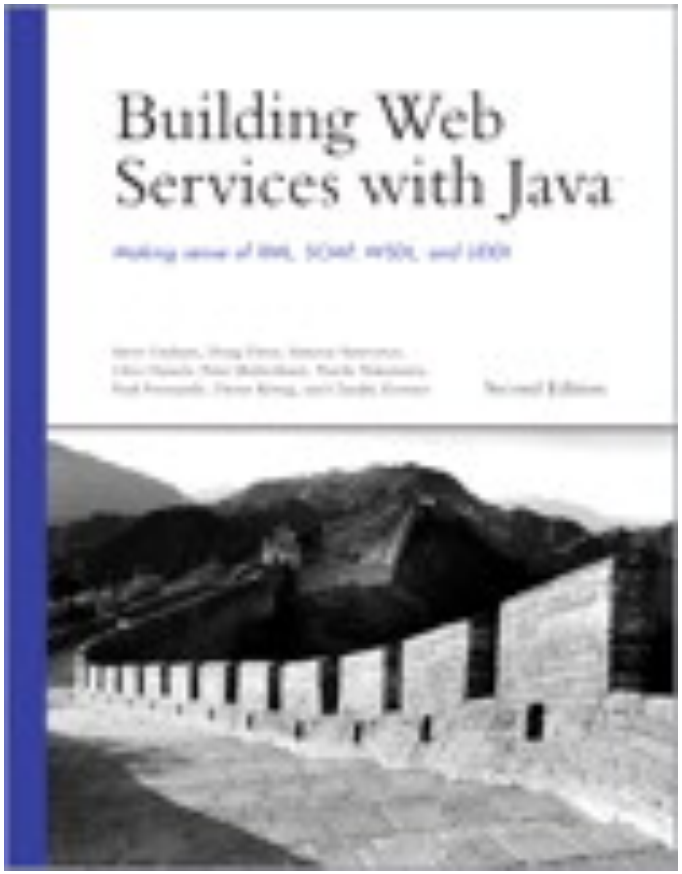
- **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition**

  By Steve Graham, Doug Davis, Simeon Simeonov, Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle, Dieter Koenig, Claudia Zentner. Published by Sams. Series: Developer's Library

Available at
http://proquest.safaribooksonline.com/0672326418

(you have to be logged with KTH account)

- lecture notes
- selected papers (an additional list of literature may be provided in the course)



Very Tentative Lecture Plan

	Date	Lecture
1	20.01.2016	Introduction and Overview, SOA
2	22.01.2016	XML Basics
3	27.01.2016	SOAP, RESTful Web services
4	27.01.2016	Service description - WSDL, WS-Policy
5	03.02.2016	Service discovery - UDDI
6	03.02.2016	Security - WS-Security
7	08.02.2016	Coordination and transaction - WS-Coordination
8	08.02.2016	Web Services Composition (BPEL4WS)
9	15.02.2016	Stateful Web services - WS-Resources
10	15.02.2016	Semantic Web Services

What will you learn from this course?

1. What is Web Services and Service Oriented Architecture
2. What are main Web Services Standards
3. What are basic mechanisms in Web Services
4. Could Web Services and Service Oriented Architecture ” save the world”?

Introduction. Content

- Introduction
  - Historical remarks
  - Software services
  - Middleware
  - Web technologies
- Web services: a new computing paradigm?
  - What is Web Services
  - Service Oriented Architecture
  - Web services architecture stack

This lecture reference

- Text book **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition**

Chapter 1

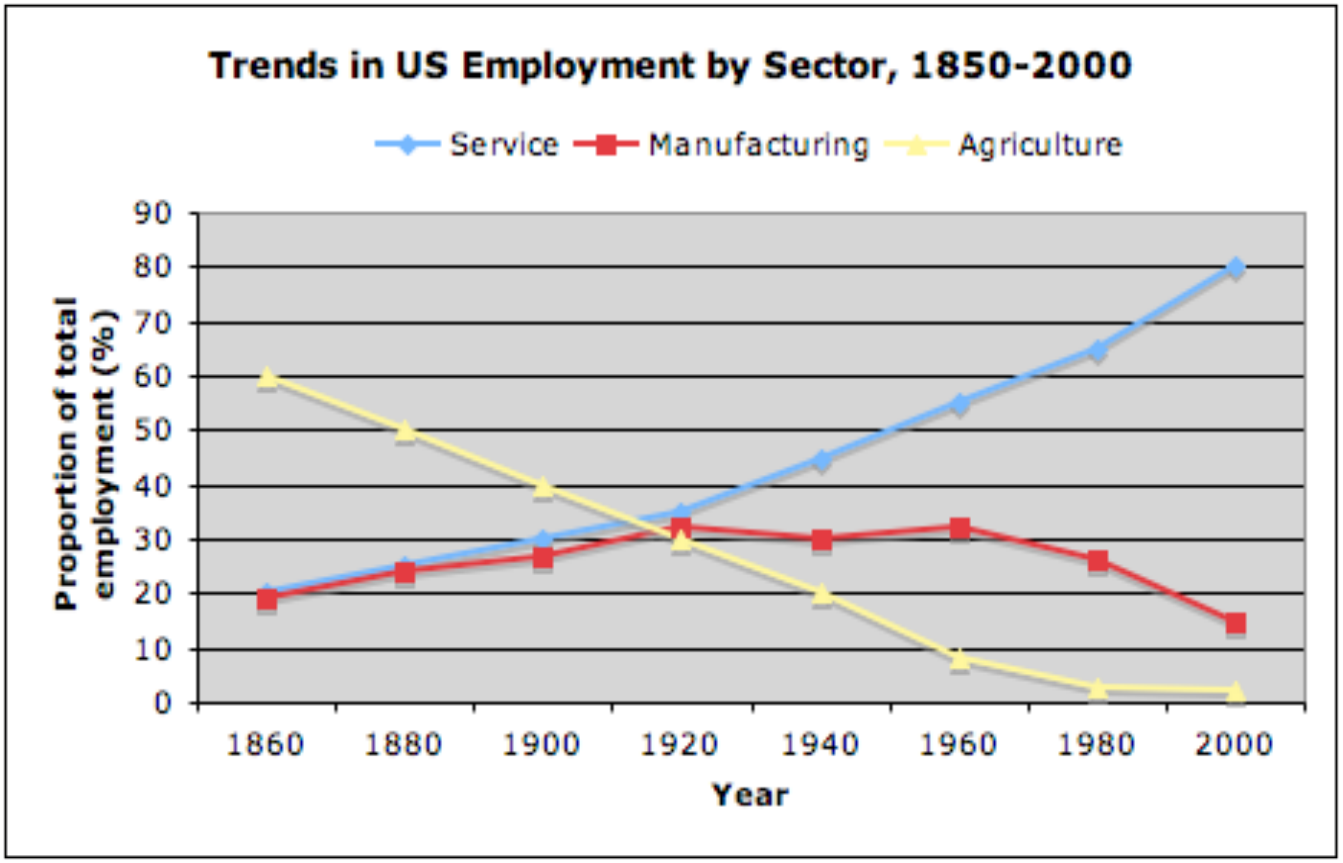
Other literature used (recommended)

- M. P. Papazoglou. Web Services: Principles and Technology Pearson Prentice Hall, 2012
- H. M. Deitel at al. Web Services. A Technical Introduction. Pearson Education. 2003
- E. Cerami. Web Services Essentials. O'Reilly and Associates.2002.
- XML and Web Services, Sams
- G. Glass. Web Services. Building Blocks for Distributed Systems. Prentice Hall. 2002
- G. Alonso. Web Services. Concepts, Architectures and Applications. Springer, 2004

What are services?

- Services are autonomous, platform independent, business functions
- In economics and marketing, a service is the non-material equivalent of a good.
- A service is a provider-to-client interaction that creates and captures value while sharing risks of the interactions
- Services are value that can be rented
- Services are the application of specialized competences (skills and knowledge) for the benefit of another entity or the entity itself

What are Services



(U.S. Department of Commerce, 1995, p. 417).

Percent Employment in Service Jobs

	1980	**1987**	**1993**	**1999**
USA	67.1	71	74.3	80.4
Canada	67.2	70.8	74.8	73.9
Japan	54.5	58.1	59.9	72.4
France	56.9	63.6	66.4	70.8
Italy	48.7	57.7	60.2	61.1
China	13.1	17.8	21.2	26.4

(United Nations, 1999).

Software as a service

- **Software as a service** (**SaaS**) is a software application delivery model where a software vendor develops a software application and hosts and operates (either independently or through a third-party) the application for use by its customers.
- Customers do not pay for owning the software itself but rather for using it.

From Wikipedia, the free encyclopedia

Service-Orientation

- A design paradigm comprised of specific set of design principles
- Emphasizes the creation of very specific design characteristics and de-emphasizes some other characteristics

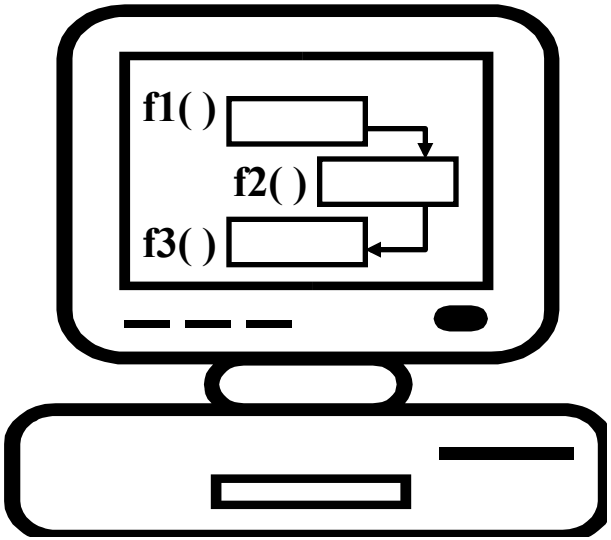
From Erl, 2008

Software Services

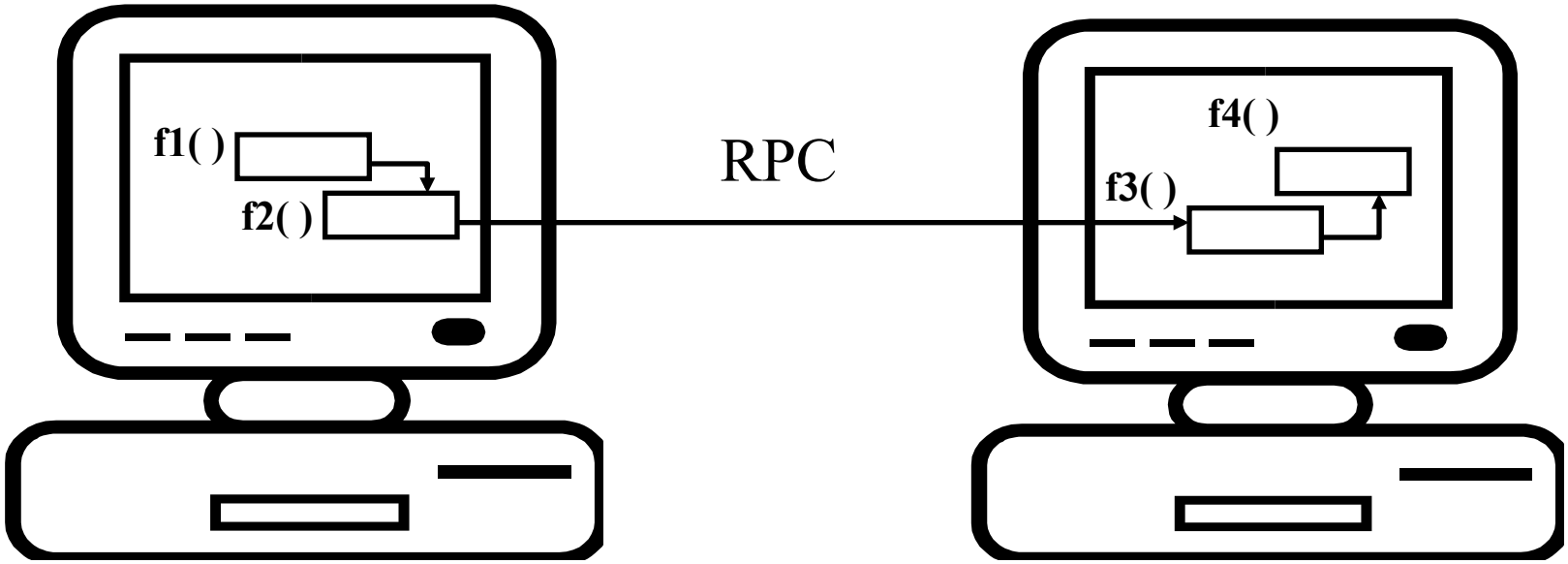
- **A software service is something that accepts digital request and returns digital response**
  - a C function, a Java object and SQL-stored procedure are all examples of software services
  - a computer application can be viewed as a set of software services

Software Services (a historical perspective)

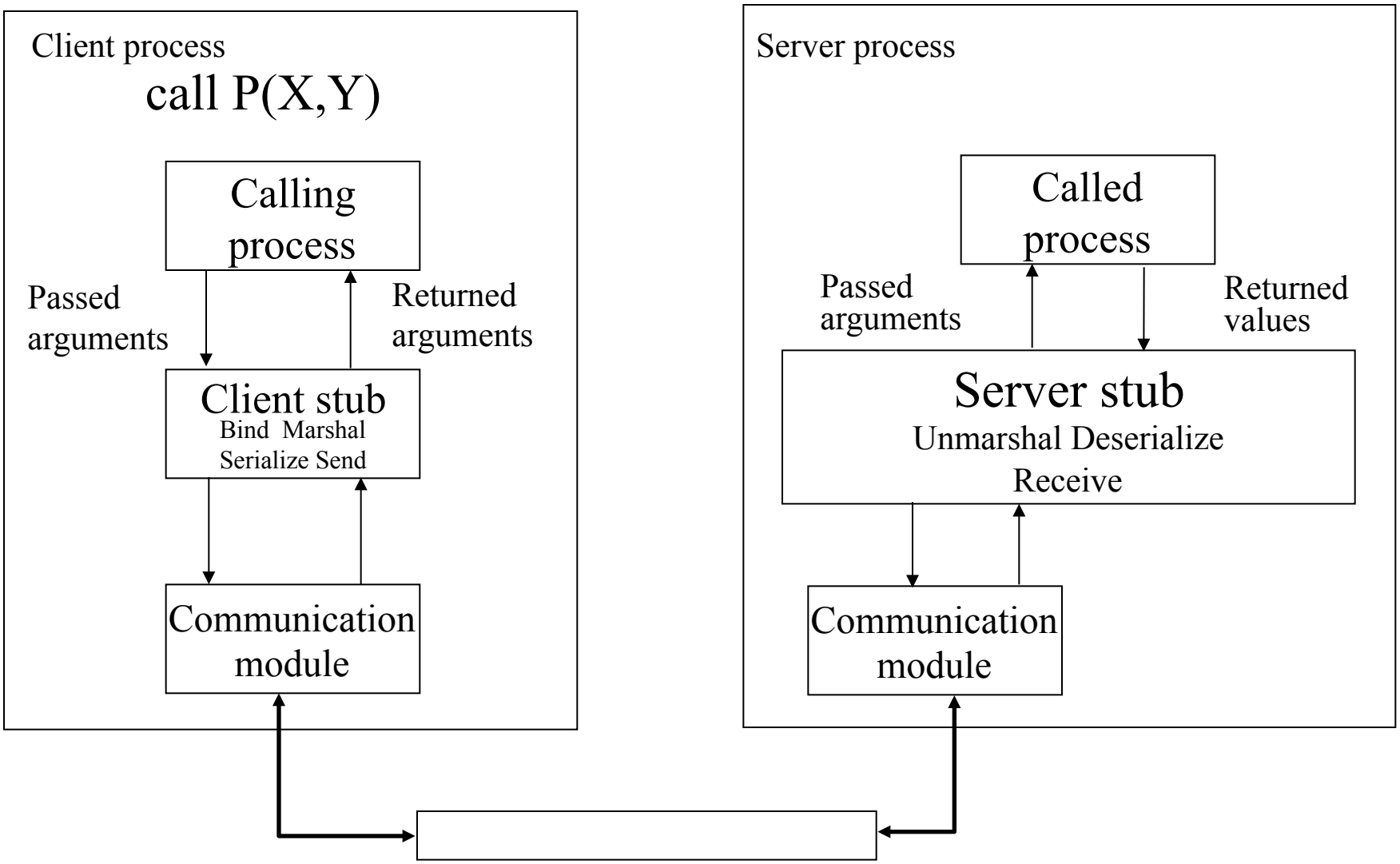
- Assembly language: subroutines
- Procedural languages: functions orchestrated by control structures



Network computing: Remote Procedure Call

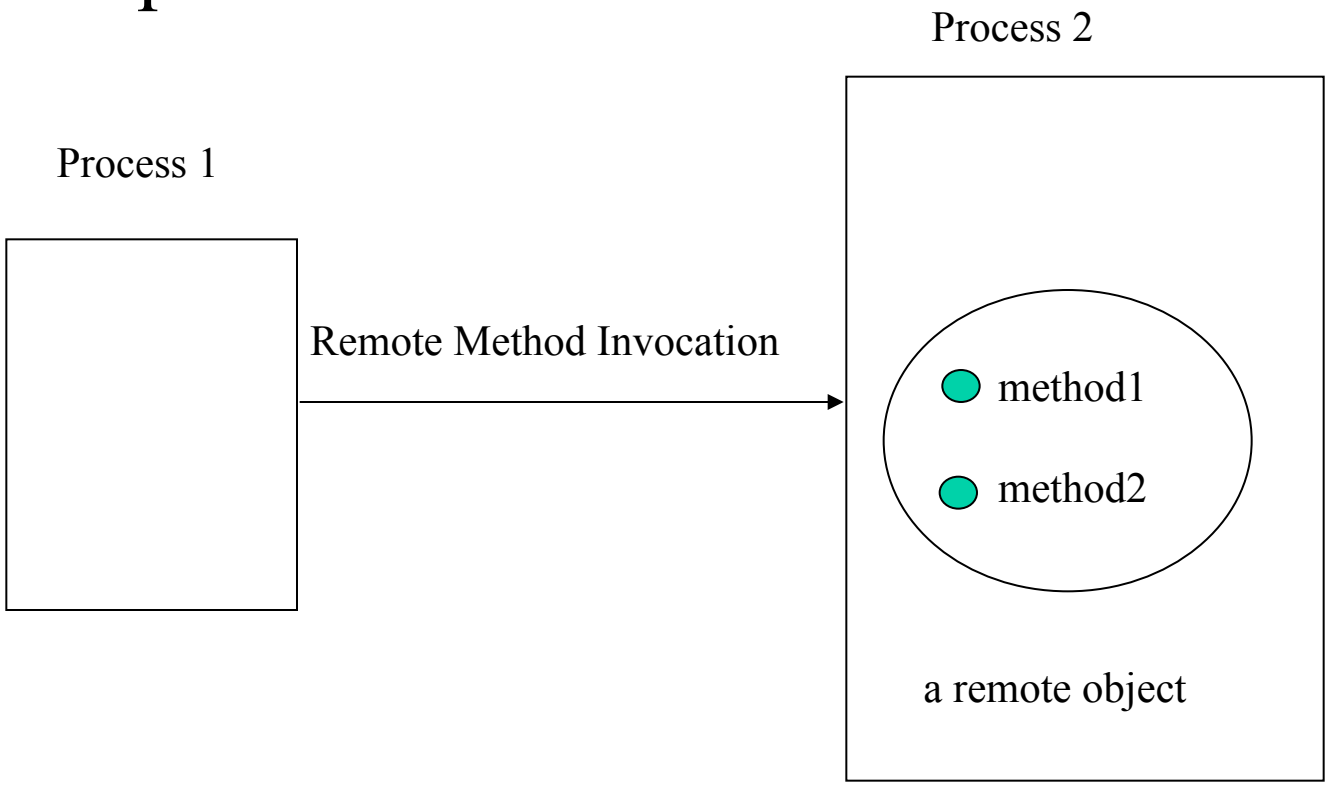


Remote Procedure Calls



Remote Method Invocation (RMI)

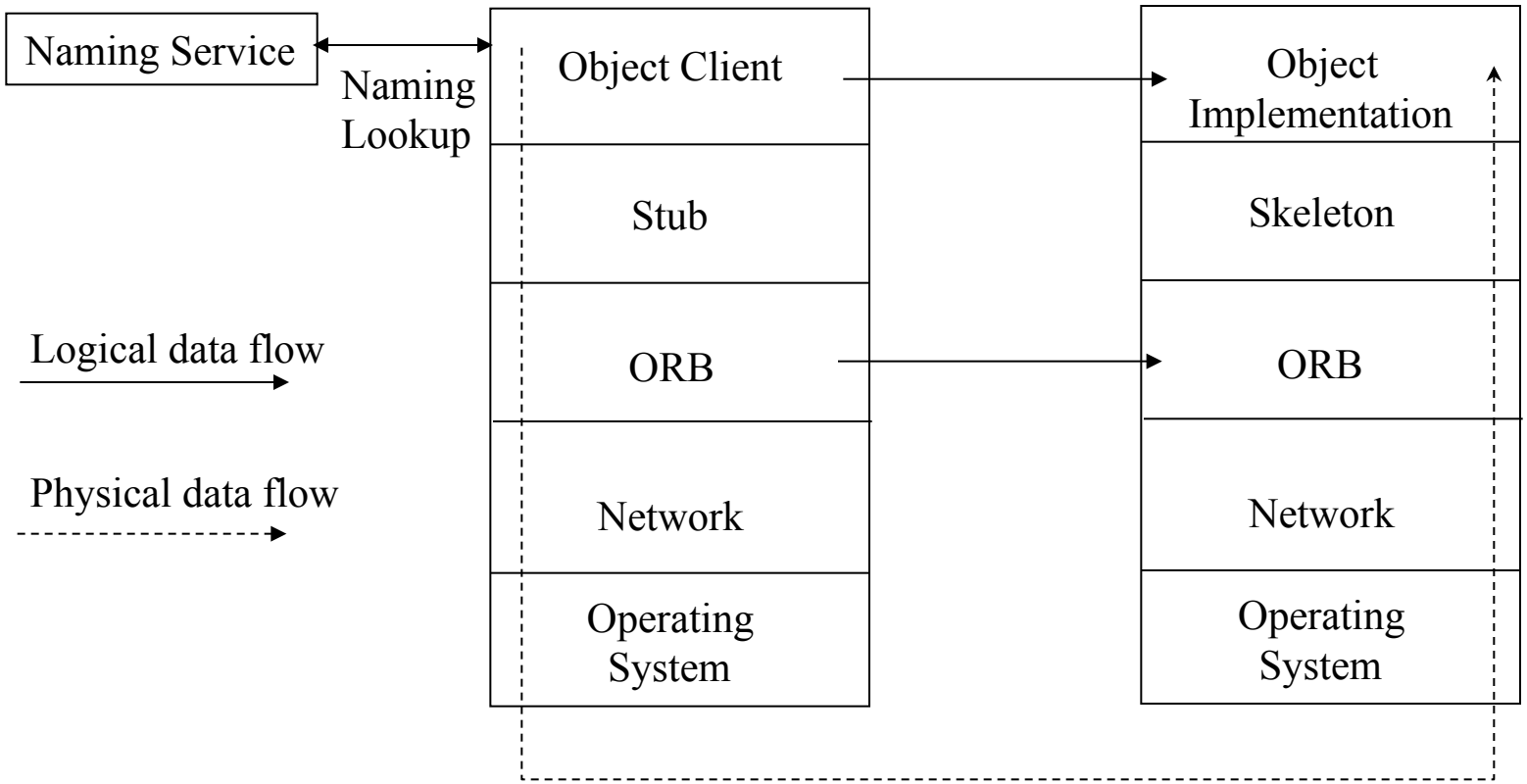
- OO equivalent of RPC



Object Broker Paradigm (ORB)

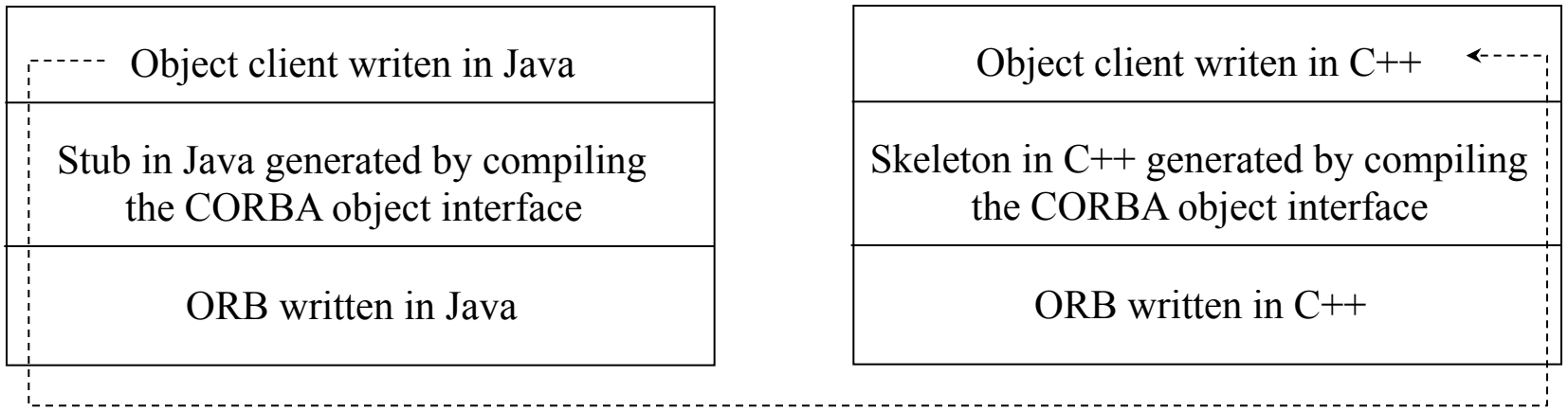
- A process issues requests to ORB which directs it to an appropriate object that provides the desired service
- Resembles RMI
- ORB functions as middleware allowing a requestor to potentially access multiple remote (or local) objects
- ORB may also function as a mediator for heterogeneous objects

Basic CORBA Architecture



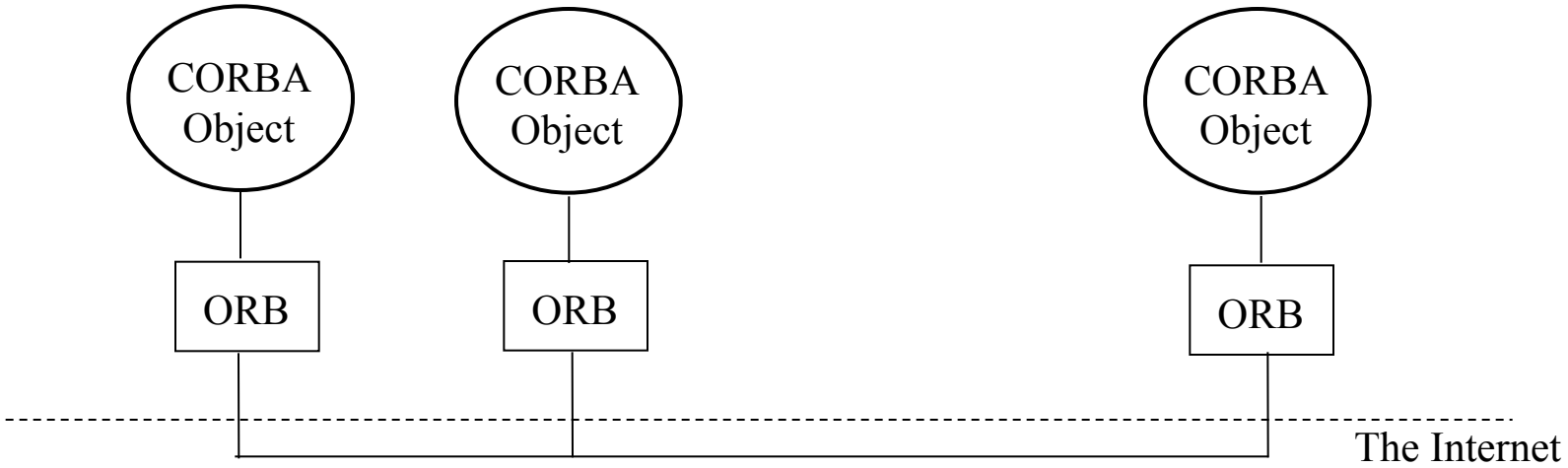
The CORBA Object Interface (language independence)

- The interface is defined using a universal language with a distinct syntax, known as the CORBA Interface Definition Language (IDL)
- For most of languages there is a standardized mapping from CORBA IDL to the programming language

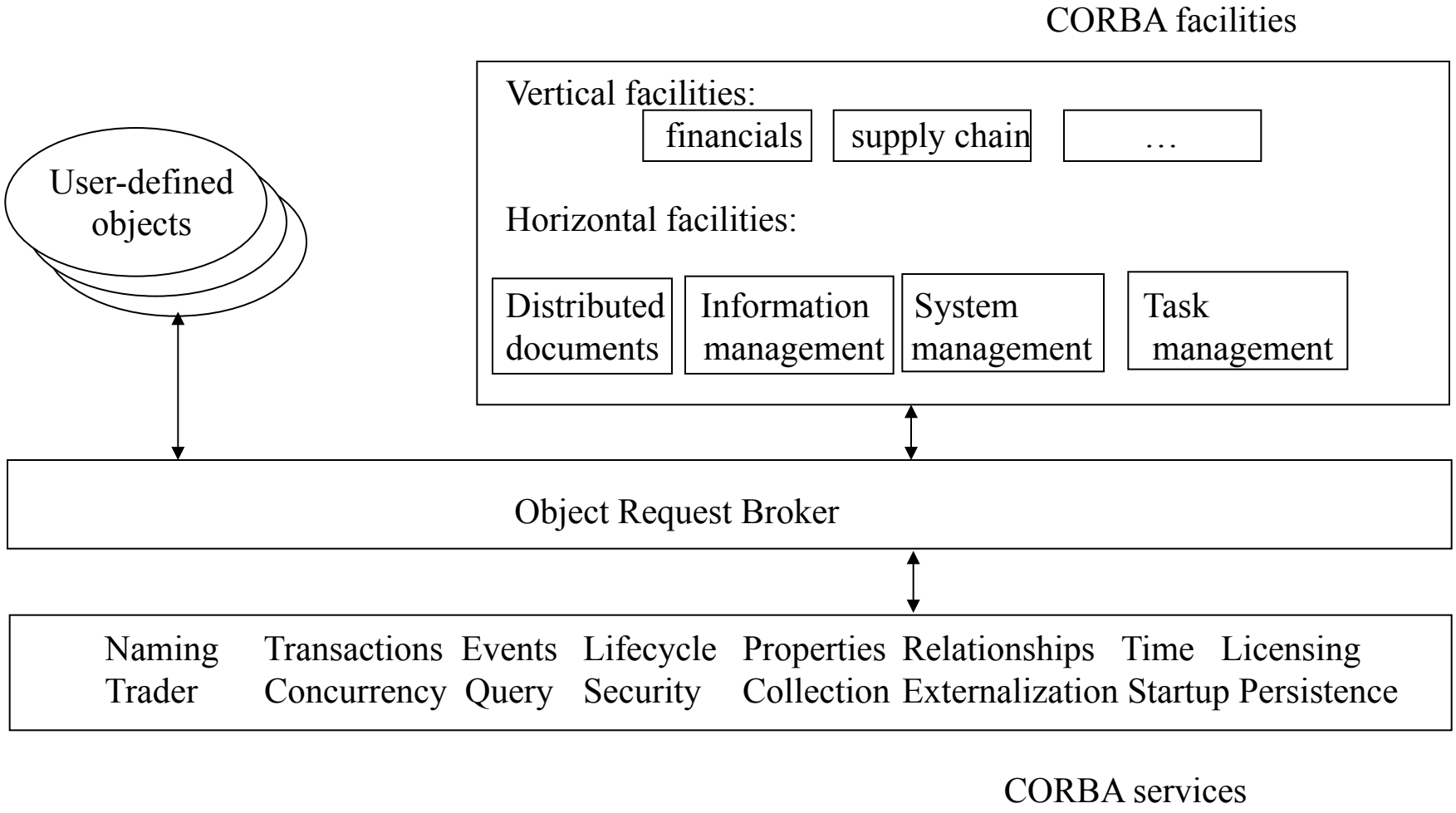


Inter-ORB Protocols

- General Inter-ORB Protocol (GIOP) provides a general framework for interoperable protocols to be built on top of specific transport layers
- A special case of the protocol is the Internet Inter-ORB protocol (IIOP) applied to TCP/IP
- Object bus: the Internet is seen as a bus that interconnects CORBA objects

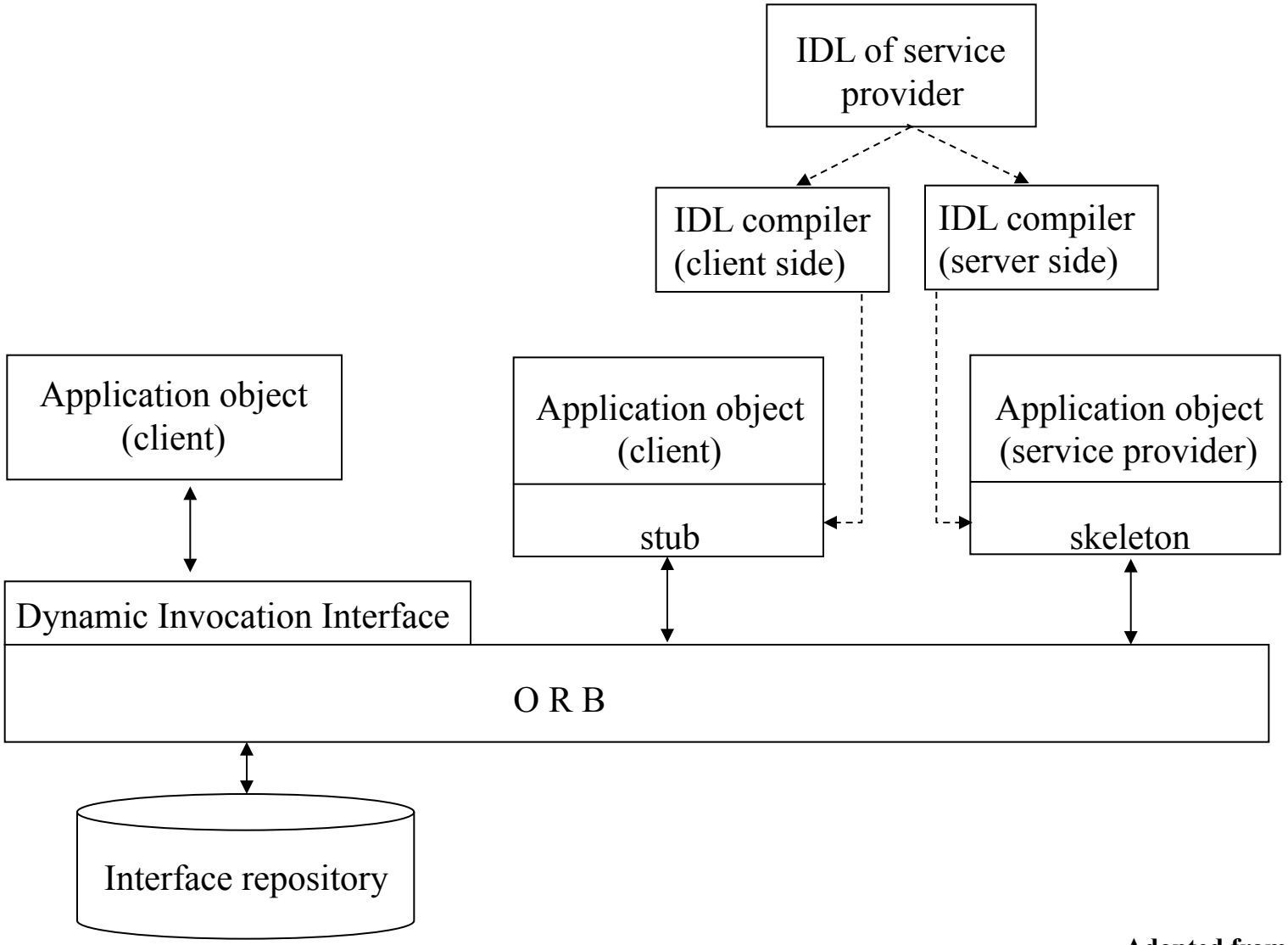


High level view of the CORBA architecture



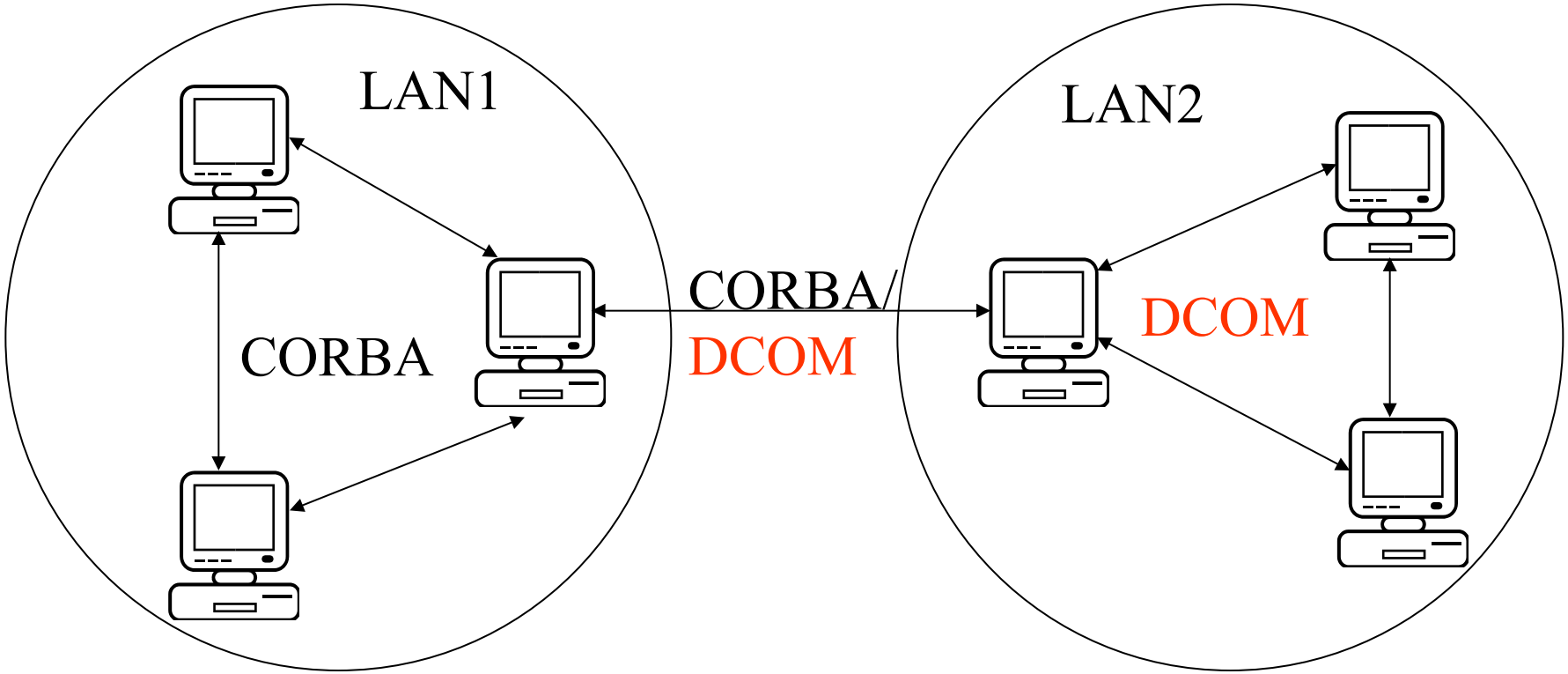
Adopted from Alomso et al.

Dynamic Service Selection and Invocation



Adopted from Alomso et al.

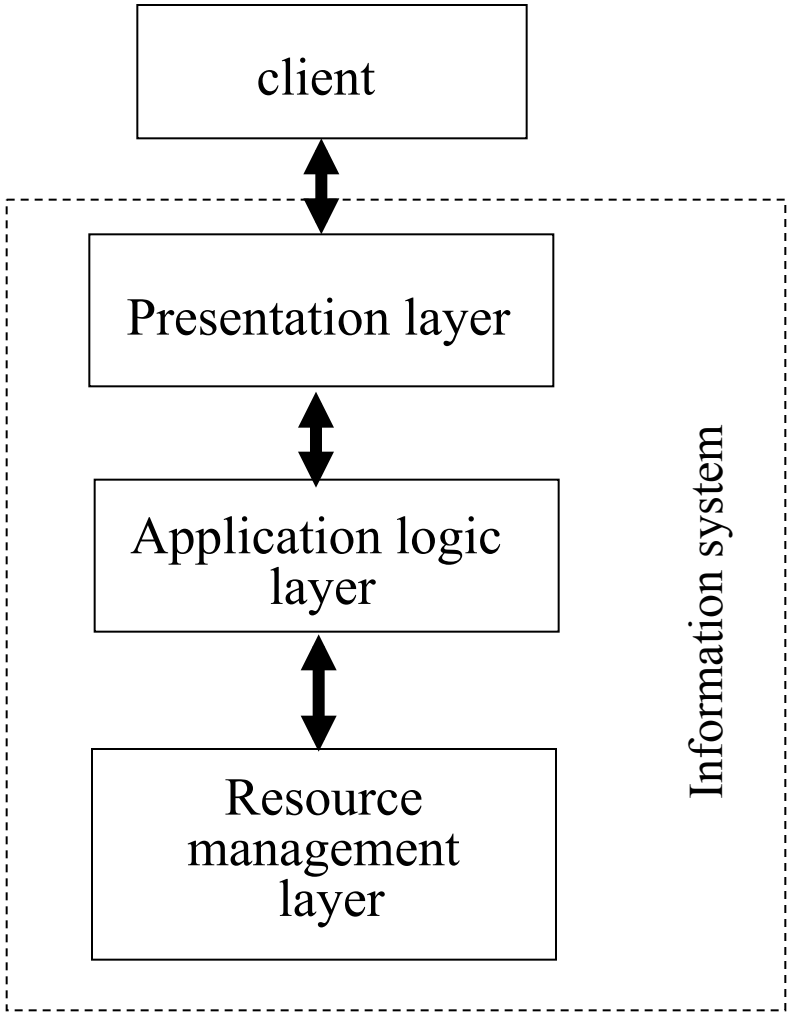
Object-oriented languages: CORBA, DCOM



Message-Based Interoperability

- Previous solutions are mainly based on synchronous method invocation
- Message-Based interoperability refers to communication by exchanging messages.
- Once client and service provider agree on a set of message types, they can communicate by exchanging messages
- It also provides a concept of message queue.

Layers of an Information System

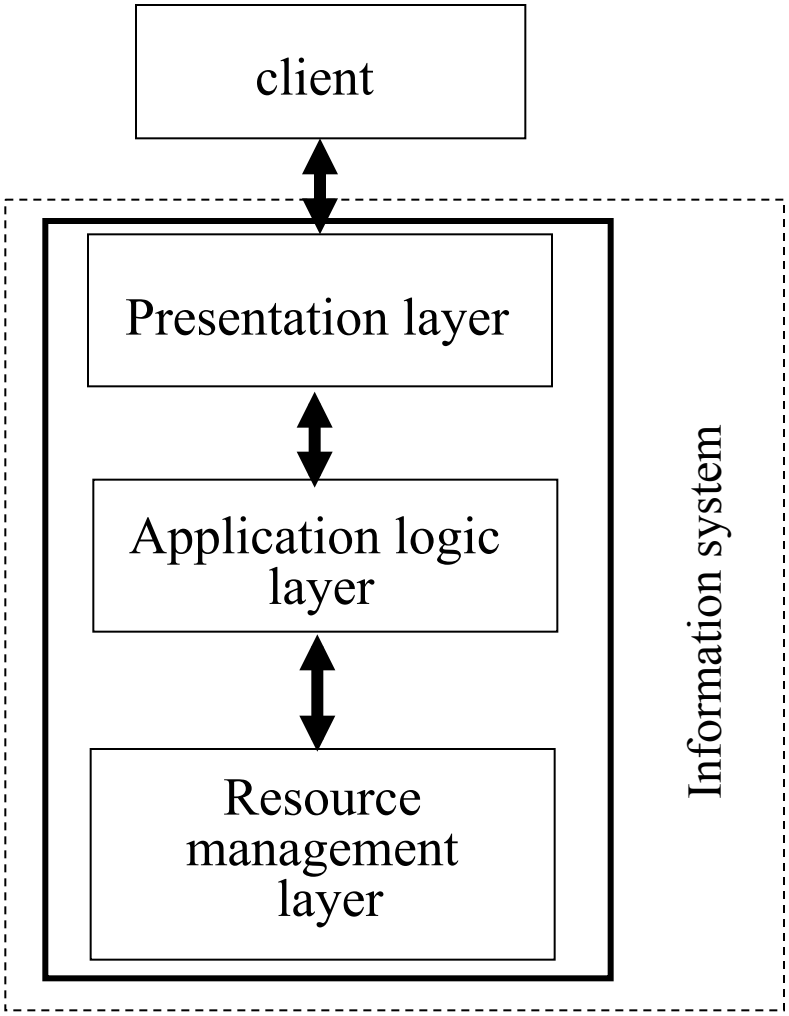


Adopted from Alomso et al.

Layers of an Information System

- Presentation layer – presenting information to external entities and allowing them interact with the system by submitting operations and getting responses
- Application logic layer – programs that implement the actual operations requested by the client through the presentation layer
- Resource management layer – deals with and implements the different data sources of an information system

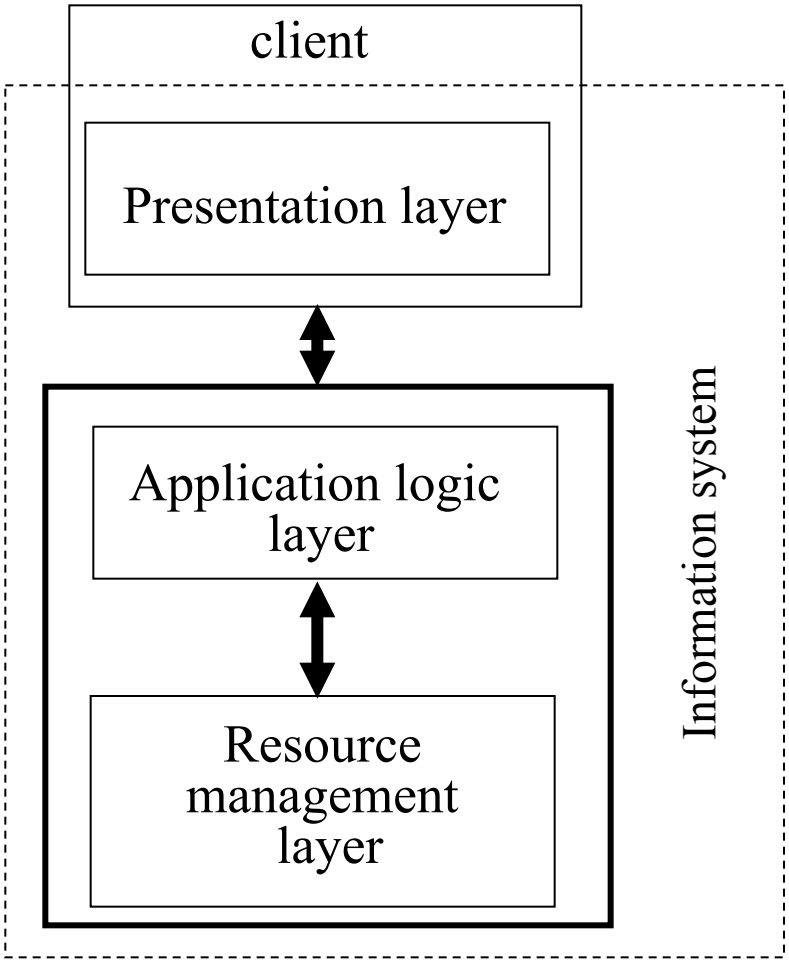
Architecture of an Information System (1-tier)



- Combines all layers in a single tier – monolithic architecture

Adopted from Alomso et al.

Architecture of an Information System (2-tier)



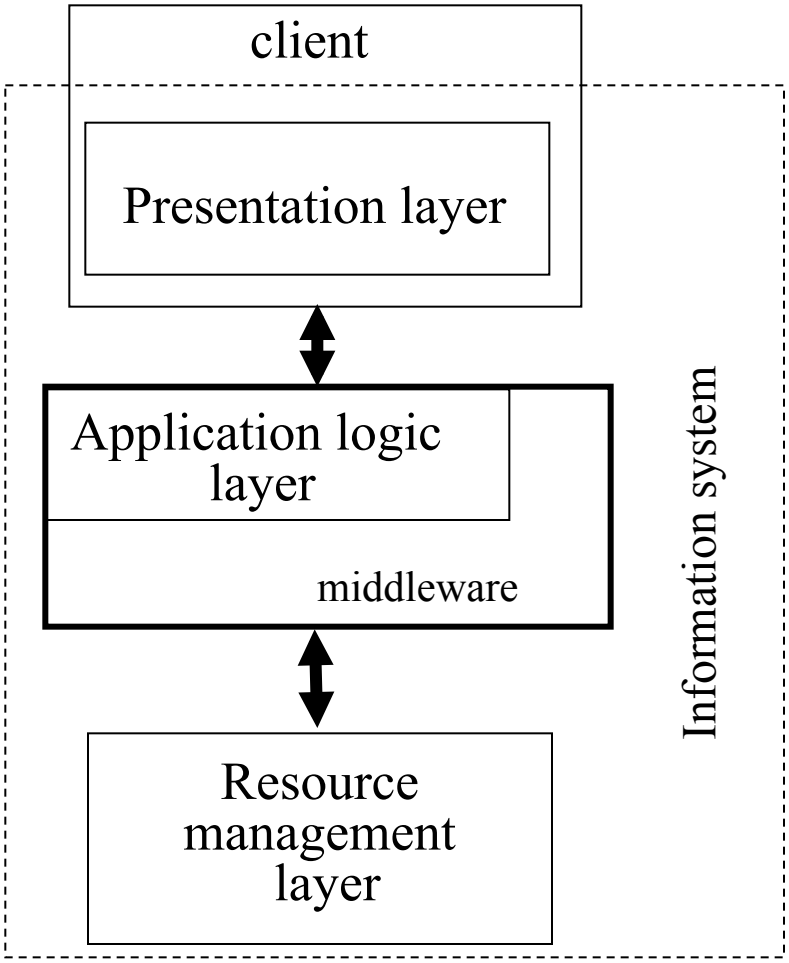
- Separates the presentation layer – the results is client/server system

Adopted from Alomso et al.

2-tier architecture

- Very popular
- Thin (minimal functionality) and fat (provides a wide range of functionality) clients
- Allows efficiency and portability in implementation
- Problems:
  - Single server can only support a limited number of clients
  - 2-tiers systems received a reputation of less scalable than expected
  - 2-tier systems run into troubles when clients want to connect to more than one server.

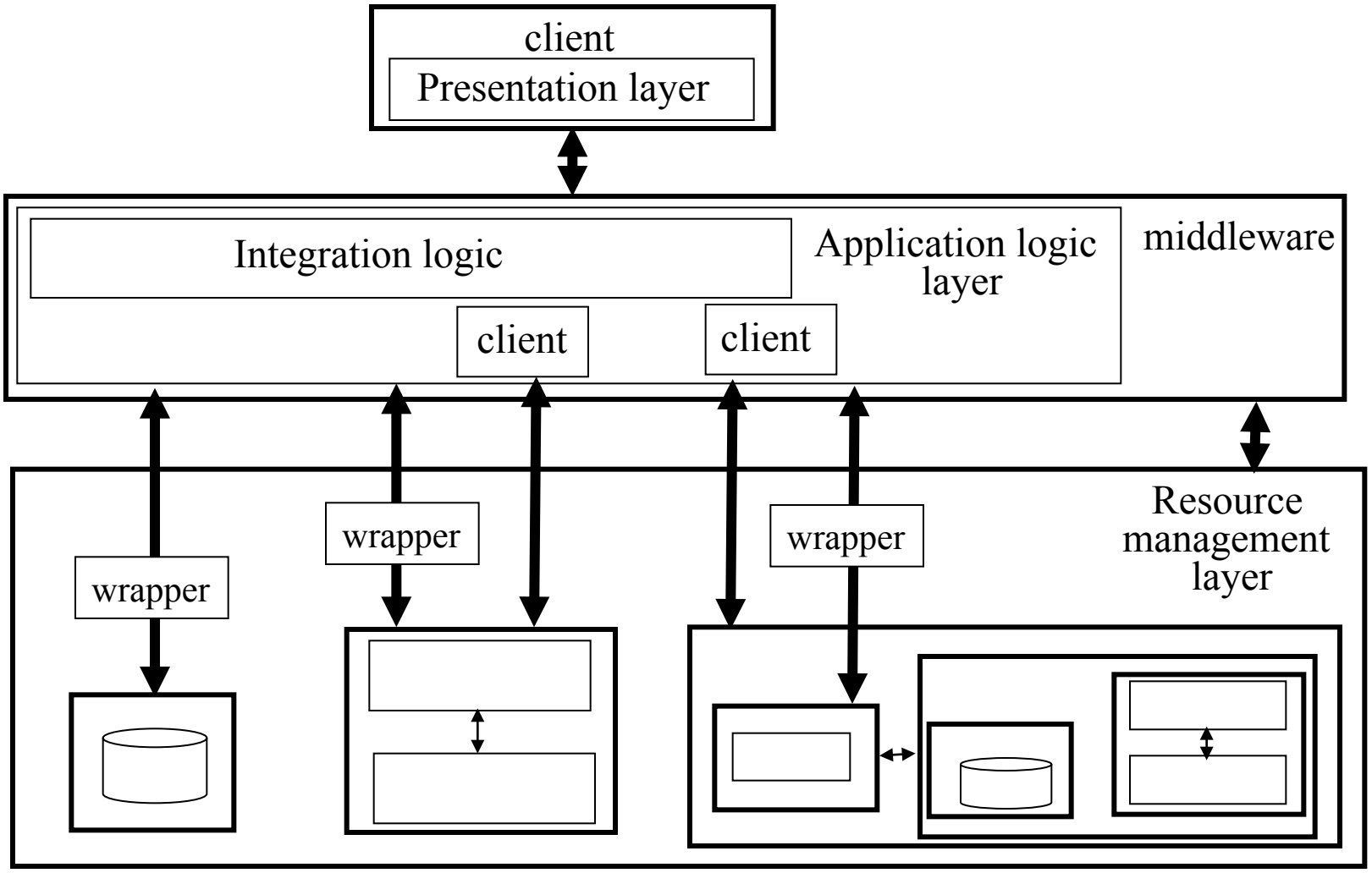
Architecture of an Information System (3-tier)



- Introduces a middleware layer

Adopted from Alomso et al.

Architecture of an Information System (3-tier)

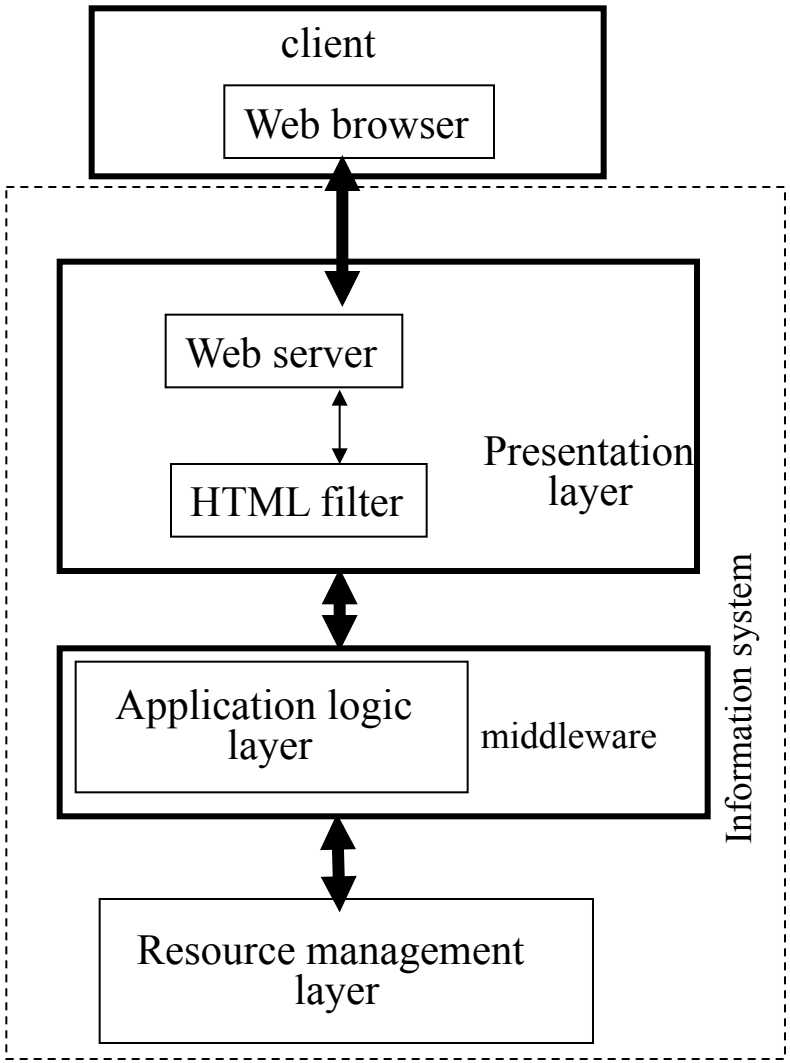


Adopted from Alomso et al.

Architecture of an Information System (3-tier)

- Main advantage – provision of an additional tier where integration logic can reside – better flexibility
- Provides better scalability and reliability
- Also may relax the server load
- However, may cause performance loss, especially when communicating with resource management layer

Architecture of an Information System (N-tier)



Adopted from Alomso et al.

Architecture of an Information System (N-tier)

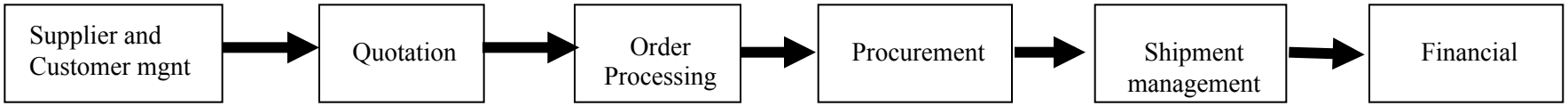
- Architecture of information systems encompasses many different tiers integrating systems that becomes building blocks for future integration
- Main disadvantages
  - too much middleware involved (often with redundant functionality)
  - Costs of developing, maintenance etc. increases almost exponentially with the number of tiers

Enterprise Application Integration (EAI)

- Each system may run on different OS
- Each system may support different interfaces and functionality
- Each system may use a different data format
- Each system may have different security requirements
- Each system may have different infrastructure, interaction models and protocols
- There are also non-technical challenges

Enterprise Application Integration (EAI)

- Behind any system integration effort there is a need to automate and streamline procedures
- The EAI problem appears when all different steps are to be combined into a coherent and seamless process

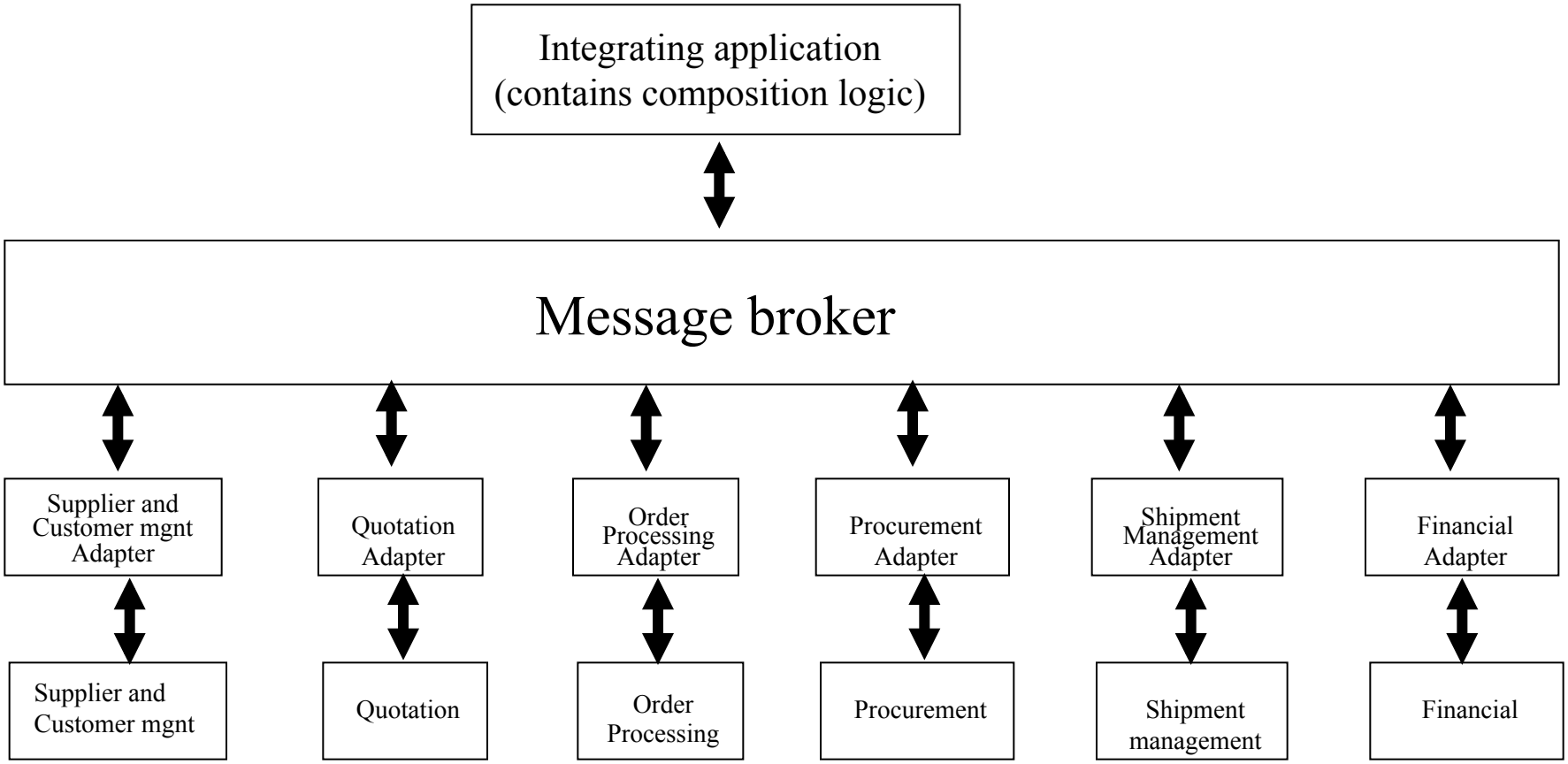


Supply chain

EAI middleware: Message Brokers

- Descendants of the platforms for message-based interoperability
- Derived from the new requirements posted by EAI
- Extend previous platforms with capability of attaching logic to the message and of processing messages directly at the middleware level
- Not only transporting messages but also in charge of routing, filtering and even processing the message
- Most message brokers provide adapters that mask heterogeneity

EAI middleware: Message Brokers



Adopted from Alomso et al.

Enterprise Application Integration (drawbacks)

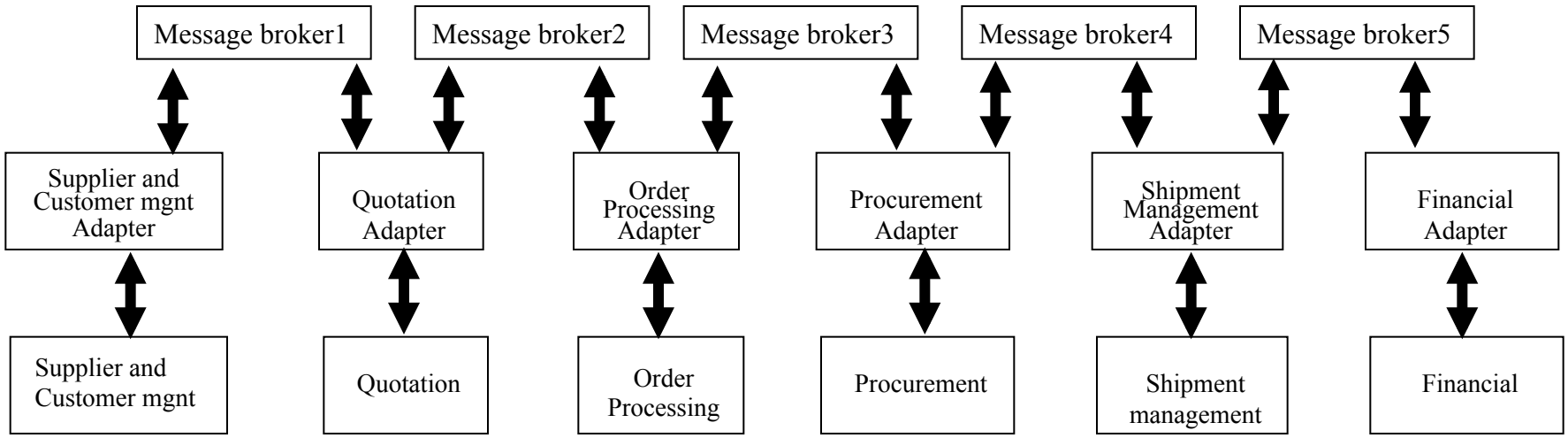
- Software licenses are extremely expensive and EAI message bus, along with its development and management tools, may cost a lot
- Each adapter may cost a lot
- Training costs a lot
- Development of integrated application is quite complex and expensive (including configuration of adapters, developing new adapters for systems not supporting EAI platform)
- This often discourage small and medium enterprises from adopting EAI platforms, or even, from performing integration

B2B integration

- Cross-organizational interactions
- The basic idea for conventional middleware is to reside between integrated applications and to mediate integration. While the application are distributed the middleware is centralized.
- While applying the same approach to B2B integration is possible it is very rarely happens in practice due to lack of trust, autonomy and confidentiality reasons

B2B integration

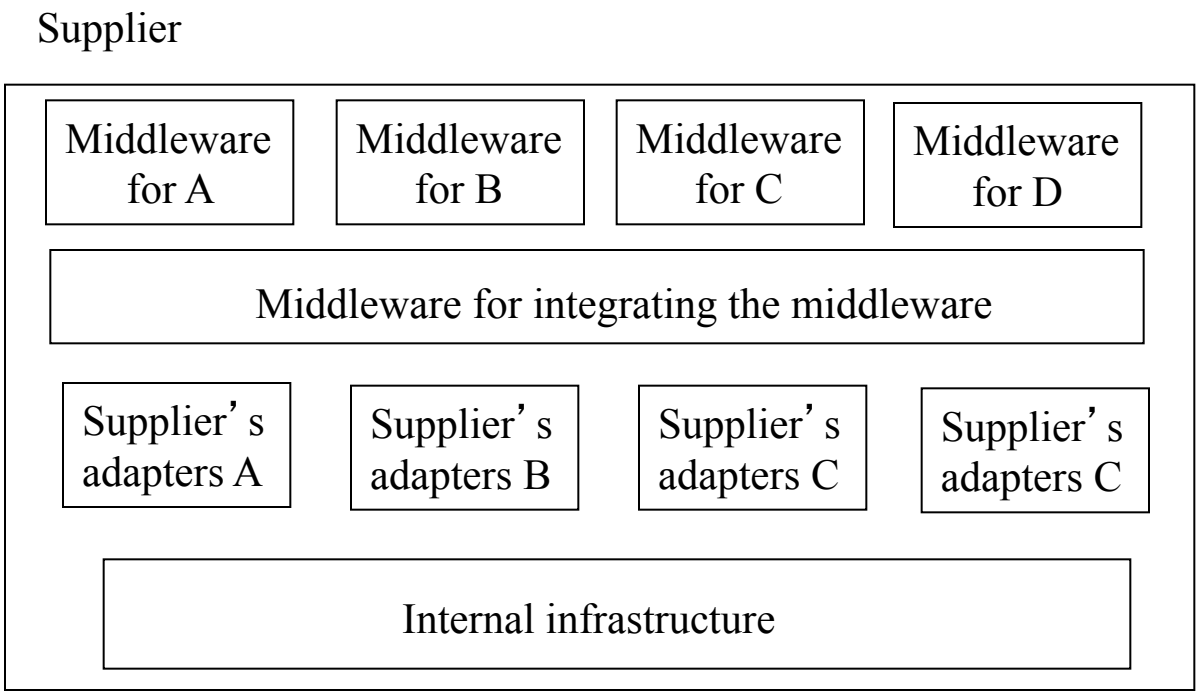
- Alternative solution is a point-to point integration



Adopted from Alomso et al.

B2B integration

- A point-to-point integration



Adopted from Alomso et al.

Web technologies

- The Web emerged as a technology for sharing information on the Internet
- It quickly became the medium for connecting remote clients with applications across the Internet
- More recently it becomes a medium for integrating application across the Web

Web Technologies

- Information exchange over the Internet
  - ARPANET,1969
  - **TCP** (Transmission Control Protocol) – handles conversion between messages and streams of packets
  - **IP** (Internet Protocol) – handles addressing of packets across networks
  - **TCP/IP** – defining technology of the Internet, enables packets to be sent across multiple networks using multiple standards

Web Technologies

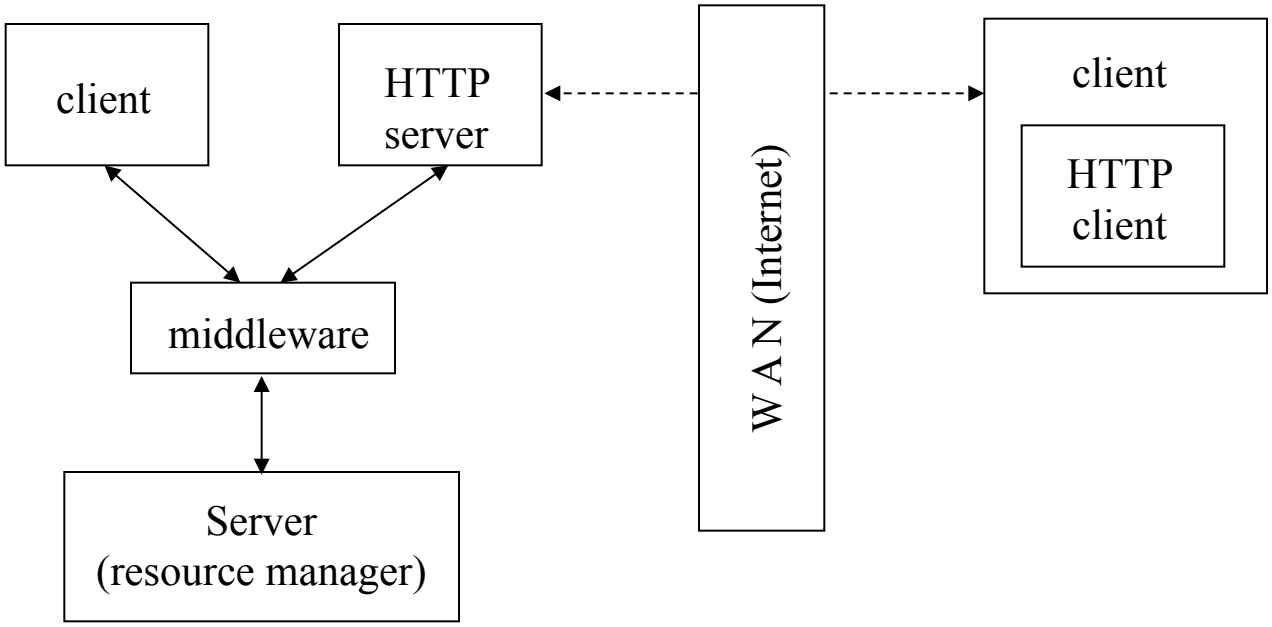
- Before the Web
  - **telnet** protocol and **SMTP** (Simple Mail Transfer Protocol) – they specify different ways of directly connecting accounts on different systems, regardless of OS and platforms
  - **FTP** (File Transfer Protocol) – supports file transfer between Internet sites (allows a system publish a set of files by hosting FTP server)

Web Technologies

- HTTP (HyperText Transfer Protocol) – a generic stateless protocol that governs a file transfer across a network (also supports access to FTP and SMTP)
  - Information exchanged in form of documents identified by URI (Uniform Resource Identifier)
  - Documents can be static or dynamic (generated at access time)
  - HTTP is designed to support hypertext (in particular, HTML)

Web Technologies (remote clients)

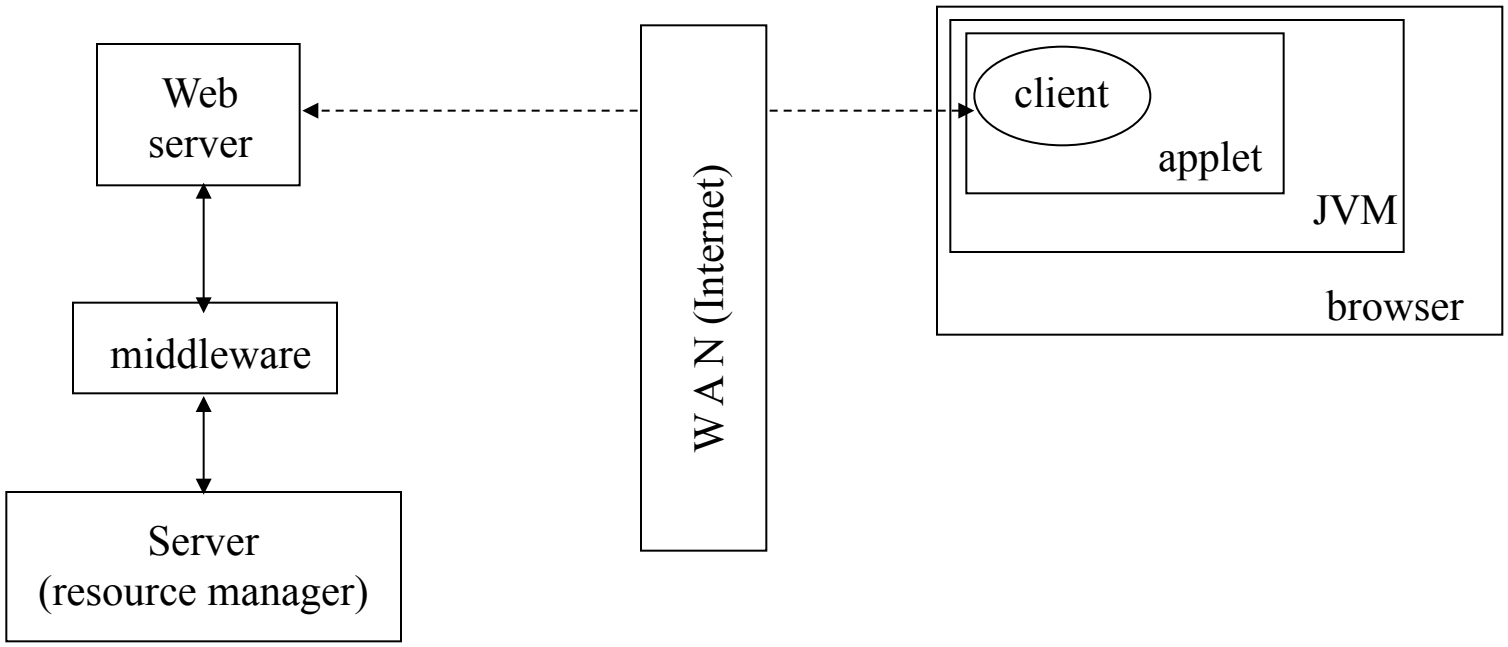
- Letting the remote computer use a Web browser as a client
- Since the Web browsers are standard tools, no application specific clients need to be installed
- Using widely available tools for building browser-based channels for company's applications
- Wrapping local information systems to support access channels



Adopted from Alomso et al.

Web Technologies (Applets)

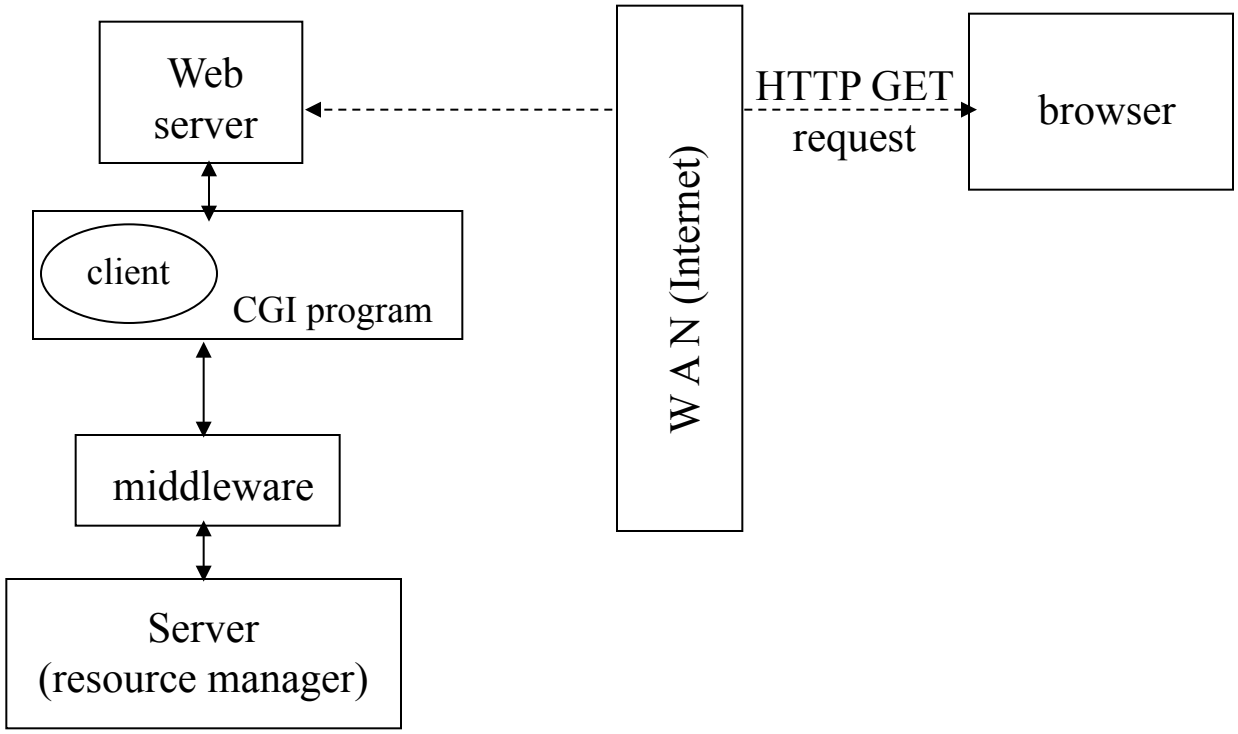
- Web browsers were initially intended to display static documents only
- Applets are Java programs that can be embedded in an HTML documents
- When the document is downloaded the program is executed by Java Virtual Machine (JVM)
- The way to turn the browser into a client is to send the client code as an applet



Adopted from Alomso et al.

Web Technologies (Common Gateway Interface - CGI)

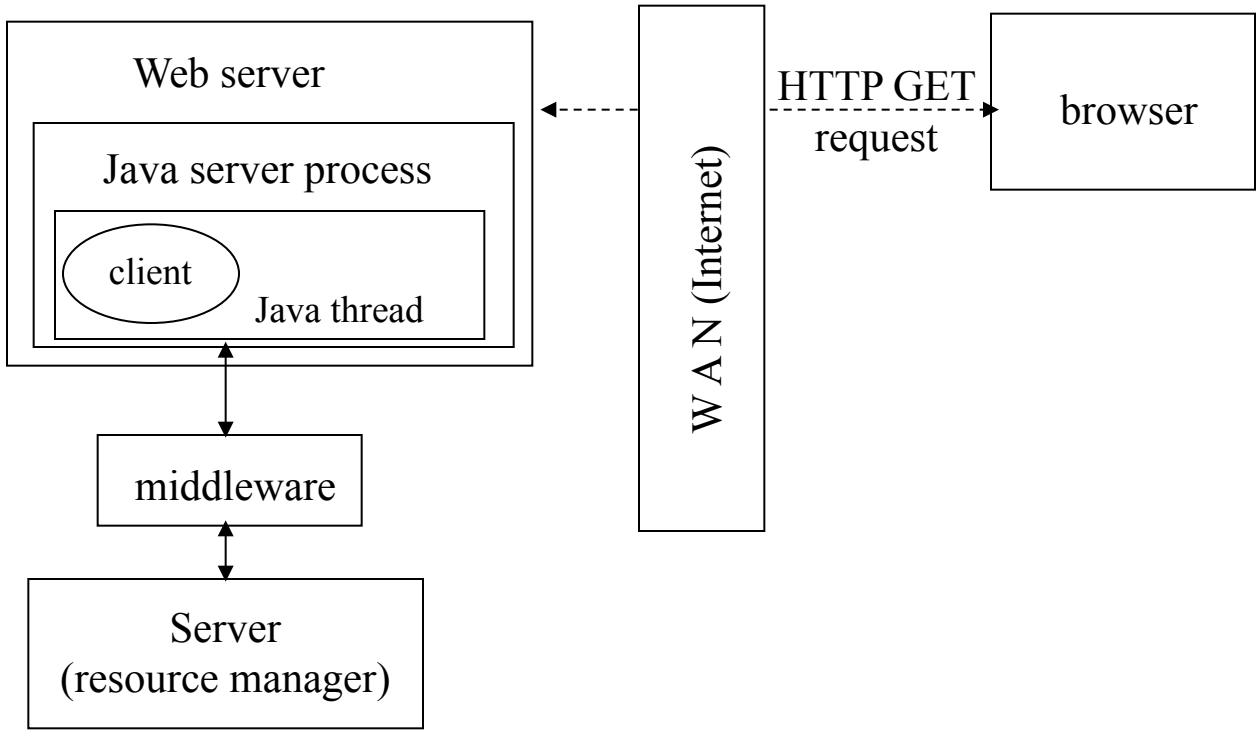
- Web server can respond to a request by invoking an application that automatically generates a document to be returned
- CGI is a standard mechanism that enables HTTP servers to interface with external applications
- CGI assigns programs to URLs, so when the URL is invoked the program is executed



Adopted from Alomso et al.

Web Technologies (Servlets)

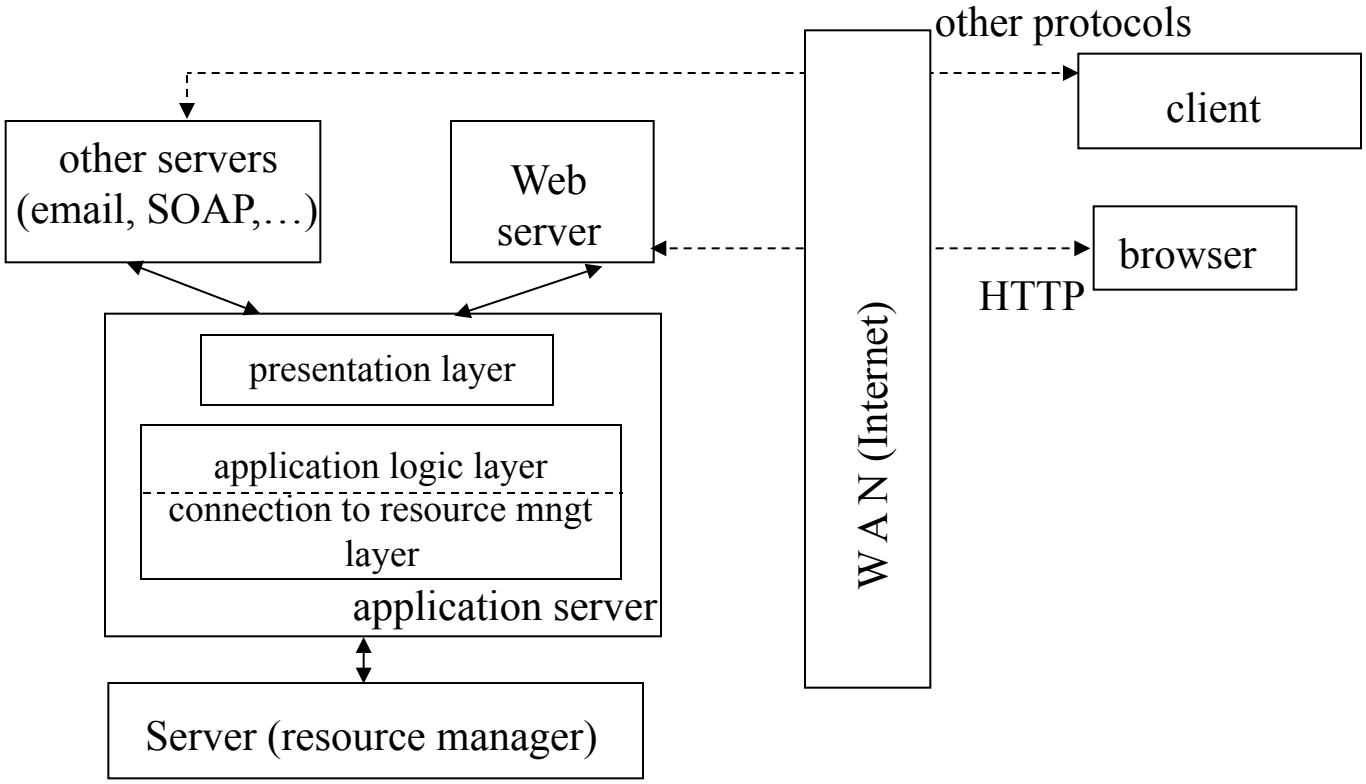
- Java servlets can be used instead of CGI programs – they differ only in implementation
- The execution of servlets is triggered by addressing URL (as for CGI). Servlets are invoked directly by embedding servlet-specific information within HTTP request
- Servlets run as threads of Java server rather than independent processes and they run as a part of Web server



Adopted from Alomso et al.

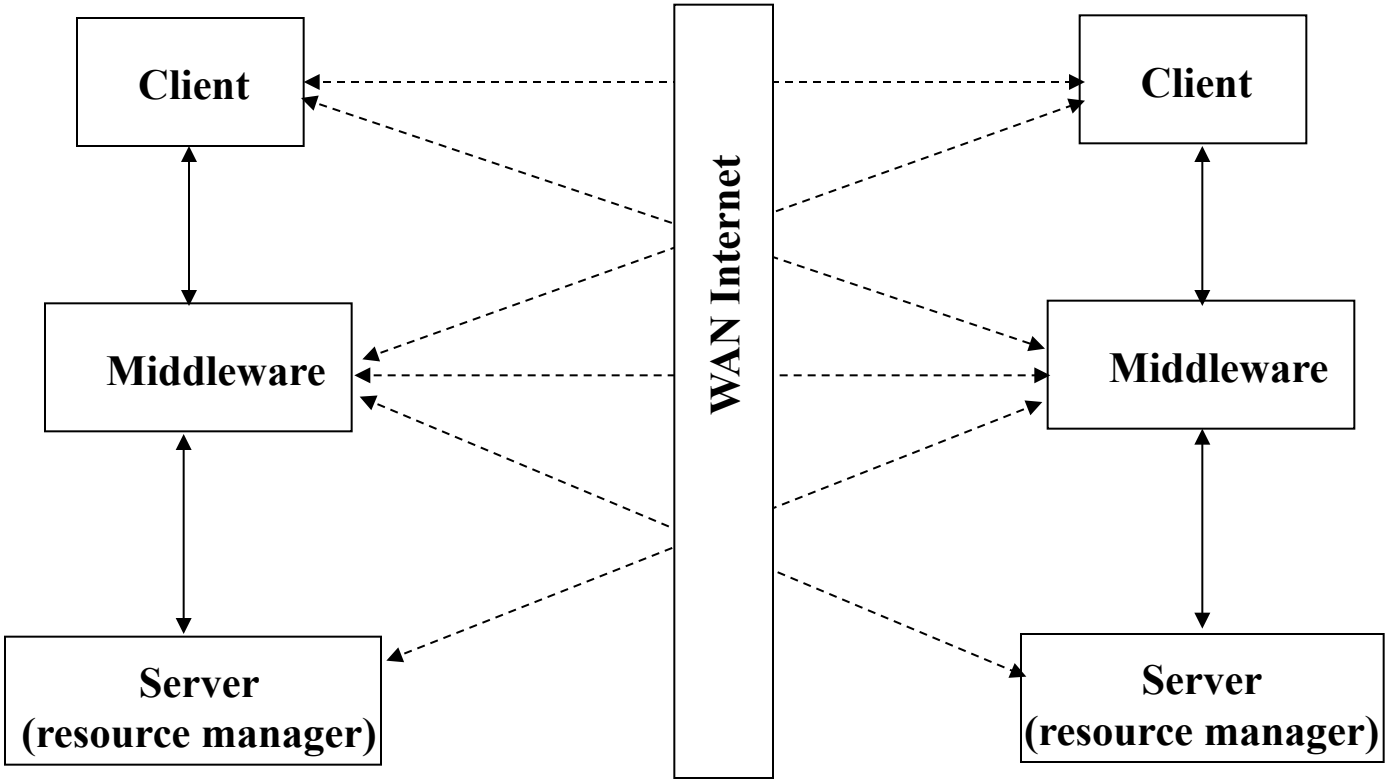
Application servers

- Application servers are equivalent to the middleware platforms but the main difference is that they incorporate the Web as a key access channel to the services implemented using the middleware



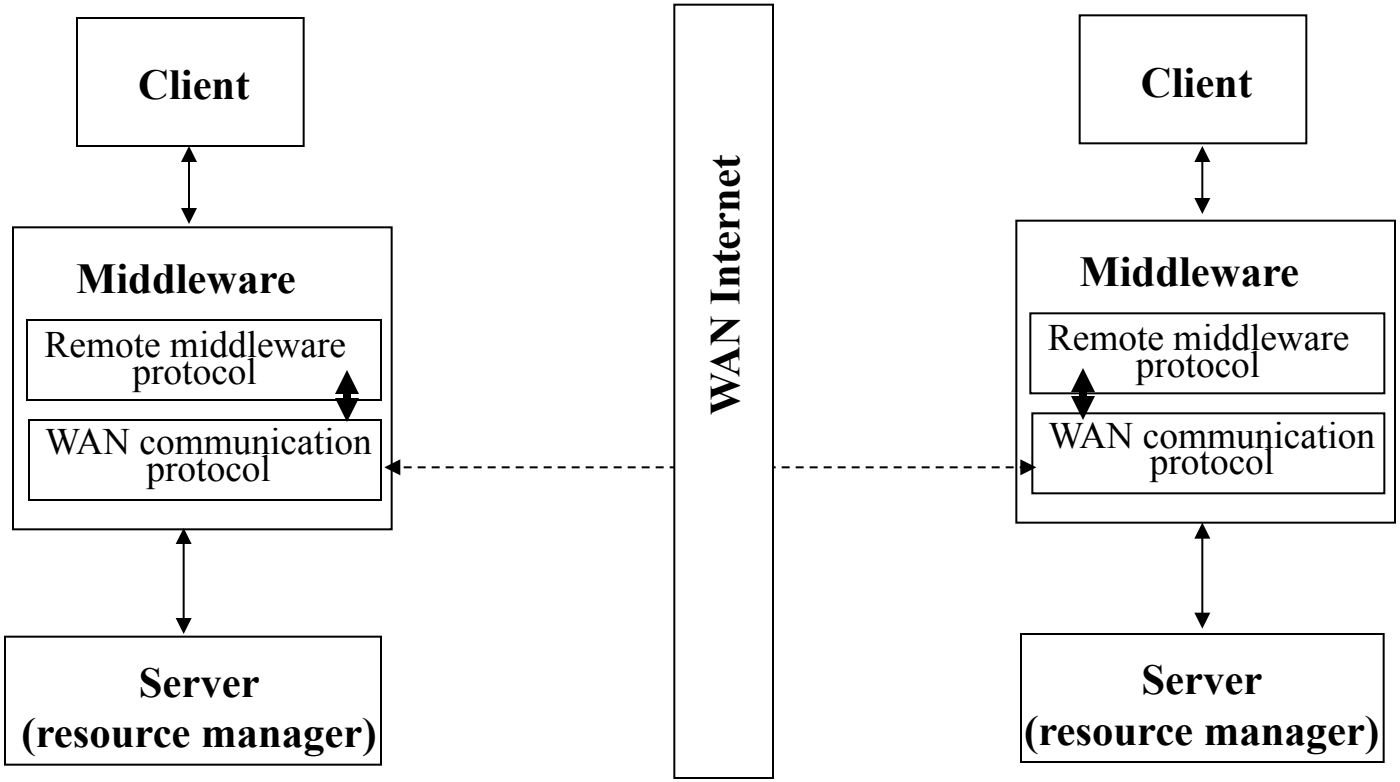
Adopted from Alomso et al.

Architecture for Wide Area Integration



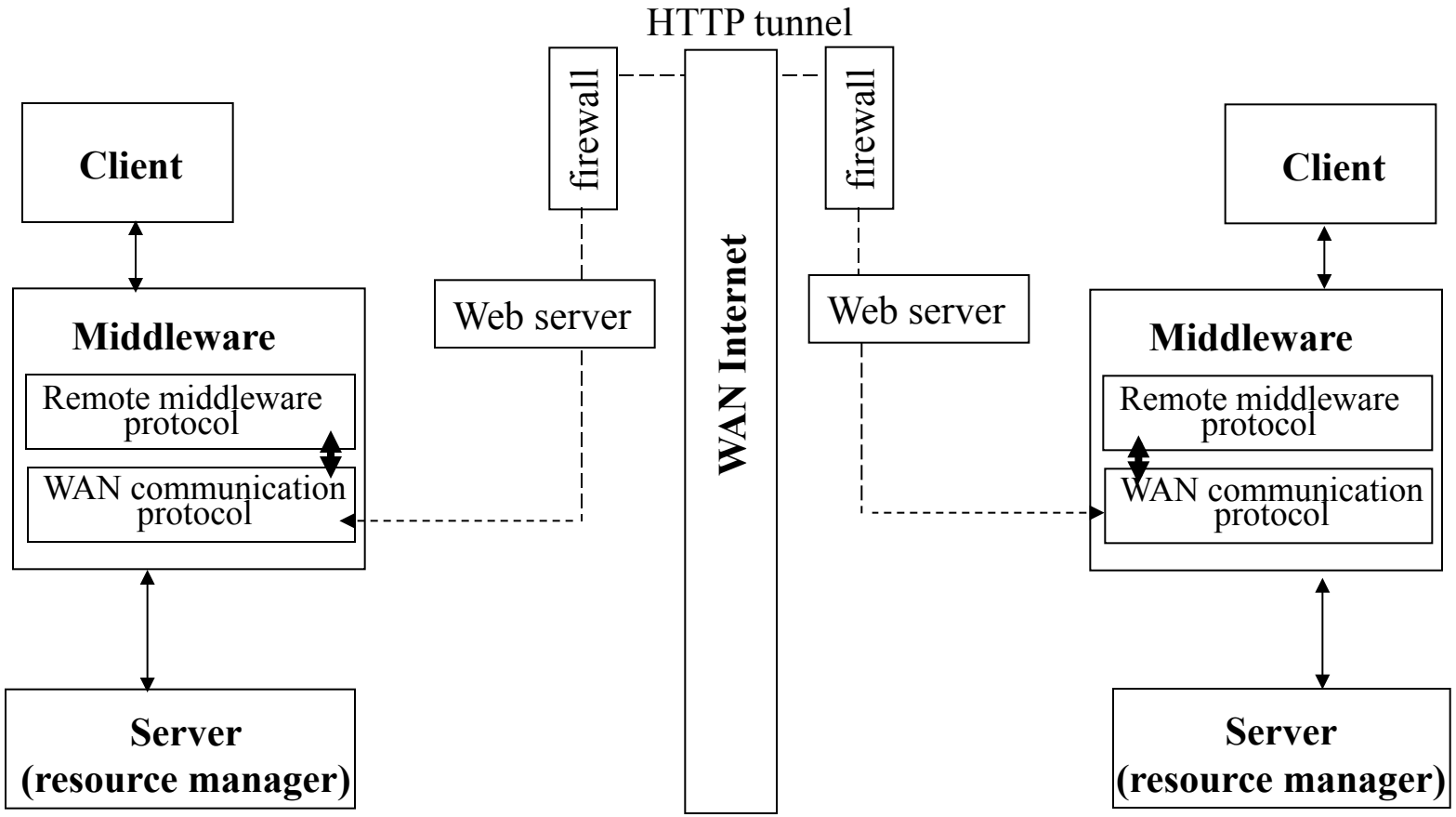
Adopted from Alomso et al.

Direct integration of middleware platforms



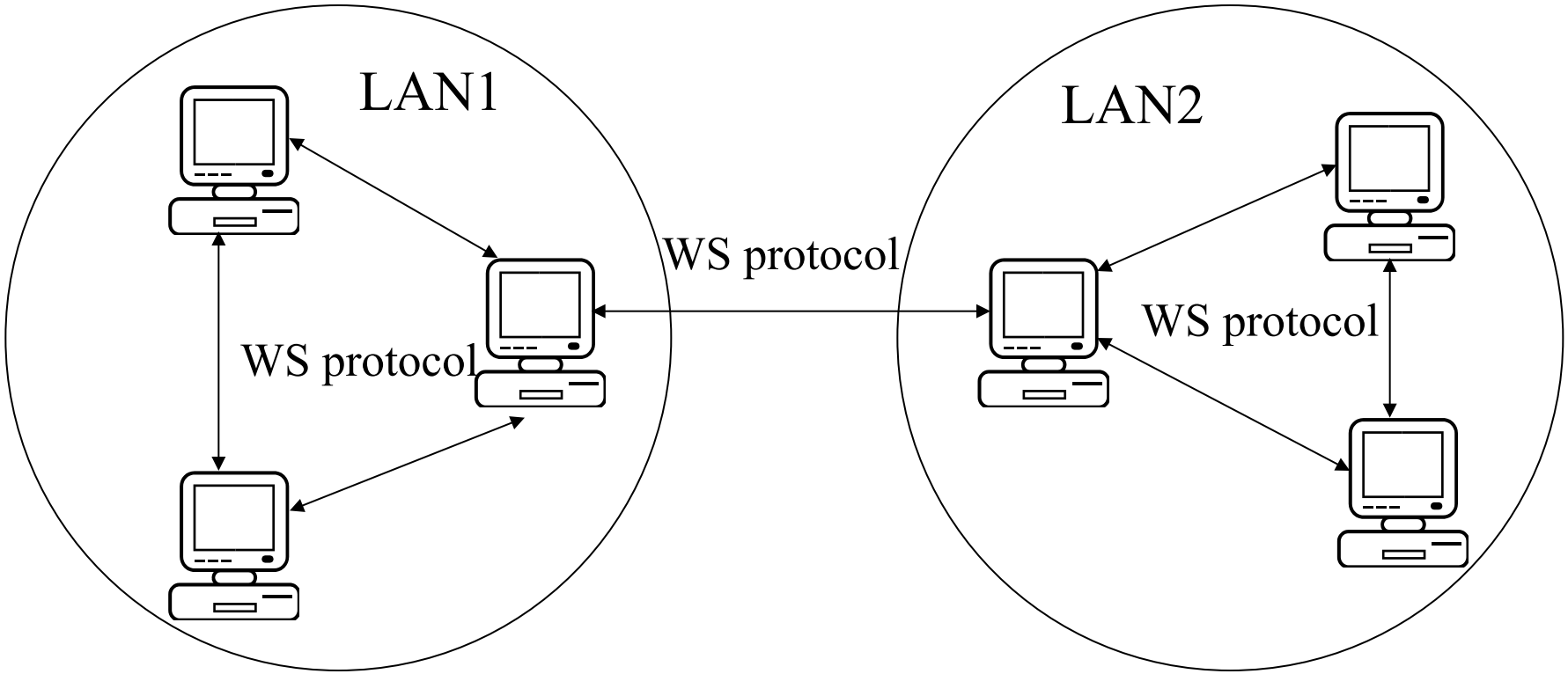
Adopted from Alomso et al.

B2B interaction using tunneling



Adopted from Alomso et al.

Web services



What are Web Services?

- “A Web service is often seen as an application accessible to other applications over the Web” (Fisher)
- “It is software designed to be used by other software via Internet protocols and formats.” (Forrester)
- “Web Services are loosely coupled software components delivered over the Internet via standards-based technologies like XML, and SOAP. “ (Gartner)

Web Services definitions

"Web Service are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business process.

Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service"

IBM web service tutorial

Web Services definitions

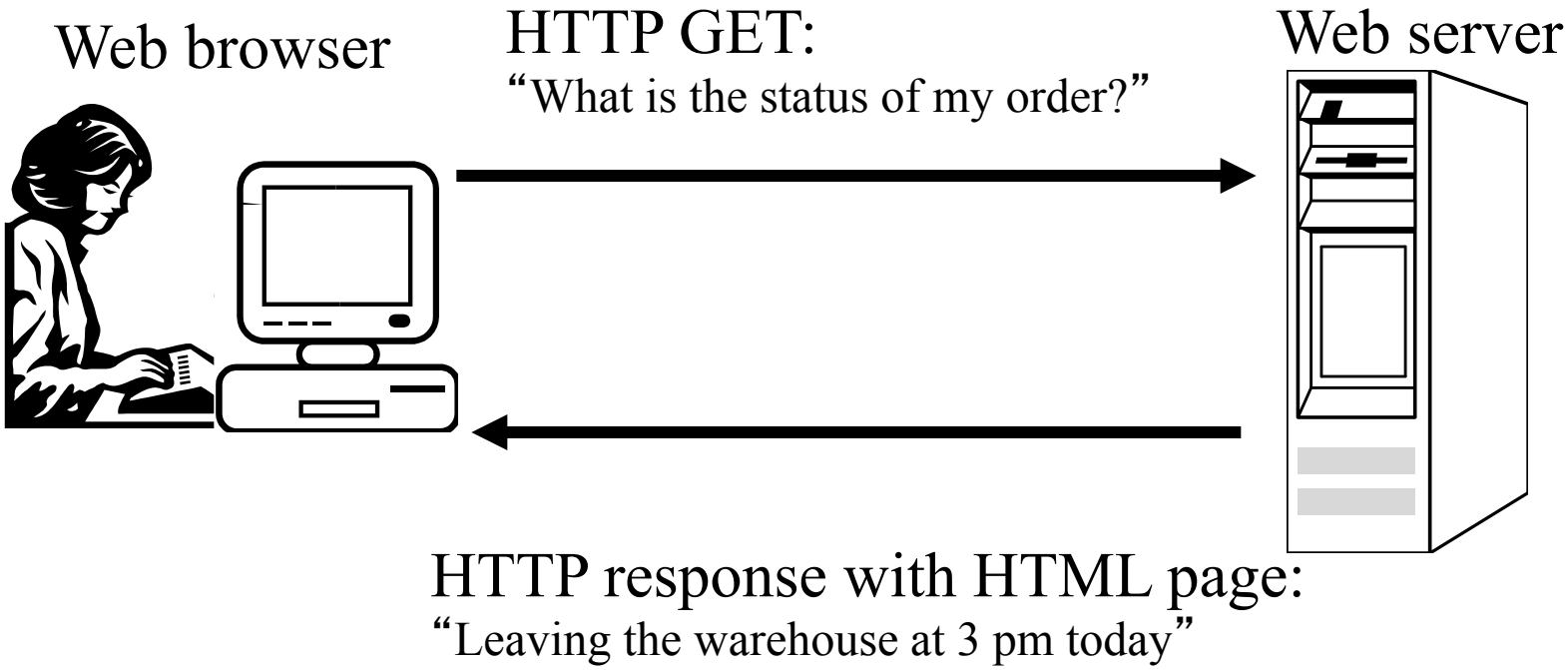
A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

W3C Working Group

Comparison with Component-Based Model

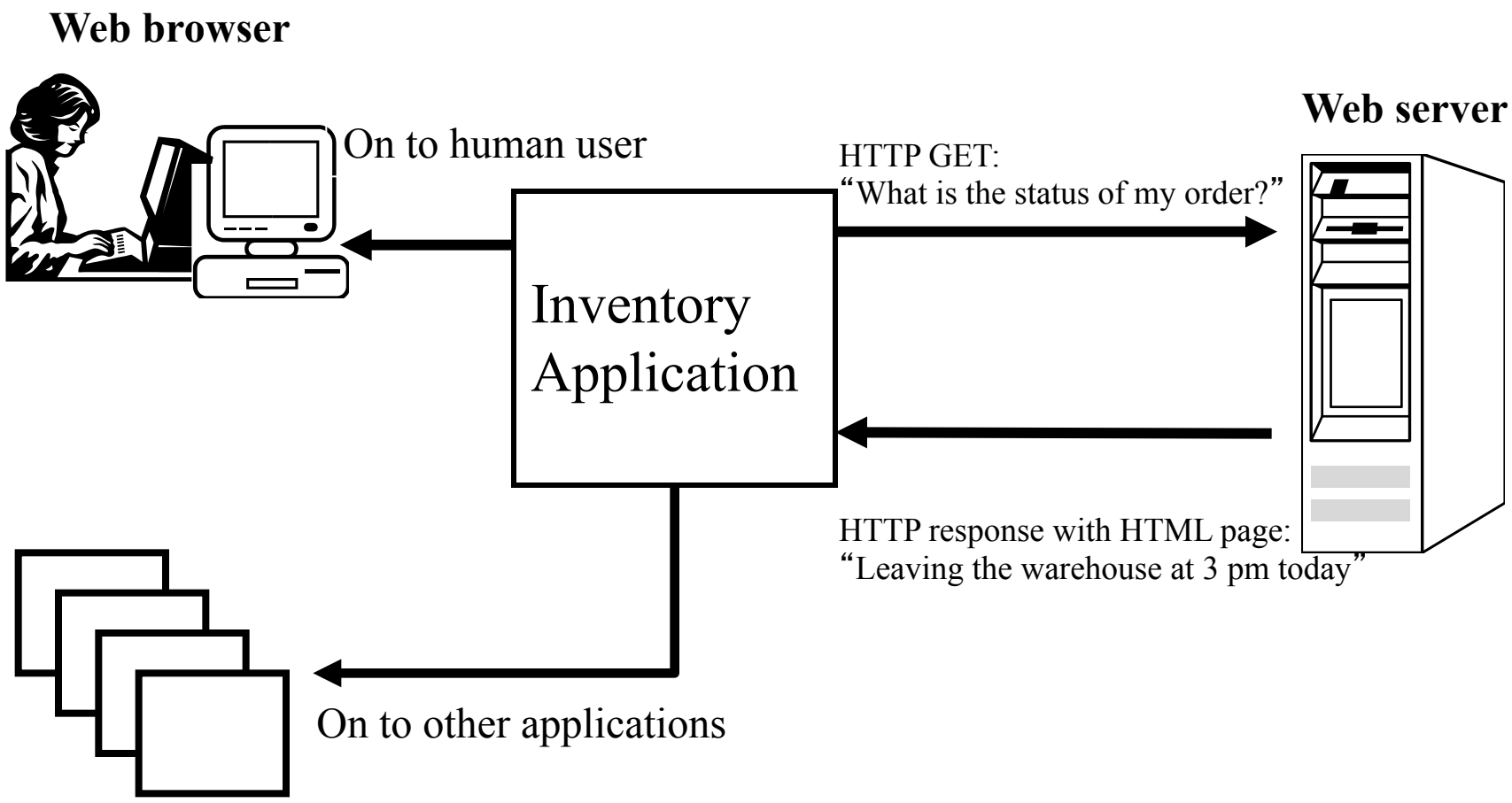
Component-Based Model	Web Services Model
Tightly coupled **software applications (high dependencies between systems)**	Loosely coupled **software applications (low dependencies between applications)**
Mainly designed for processes within the enterprise	**Mainly designed for** processes across enterprises
Uses different protocols and technologies **(e.g., Microsoft DCOM, CORBA)**	**Uses** common protocols and technologies **(e.g., XML, SOAP, WSDL, HTTP)**

The Web today (human-centric)



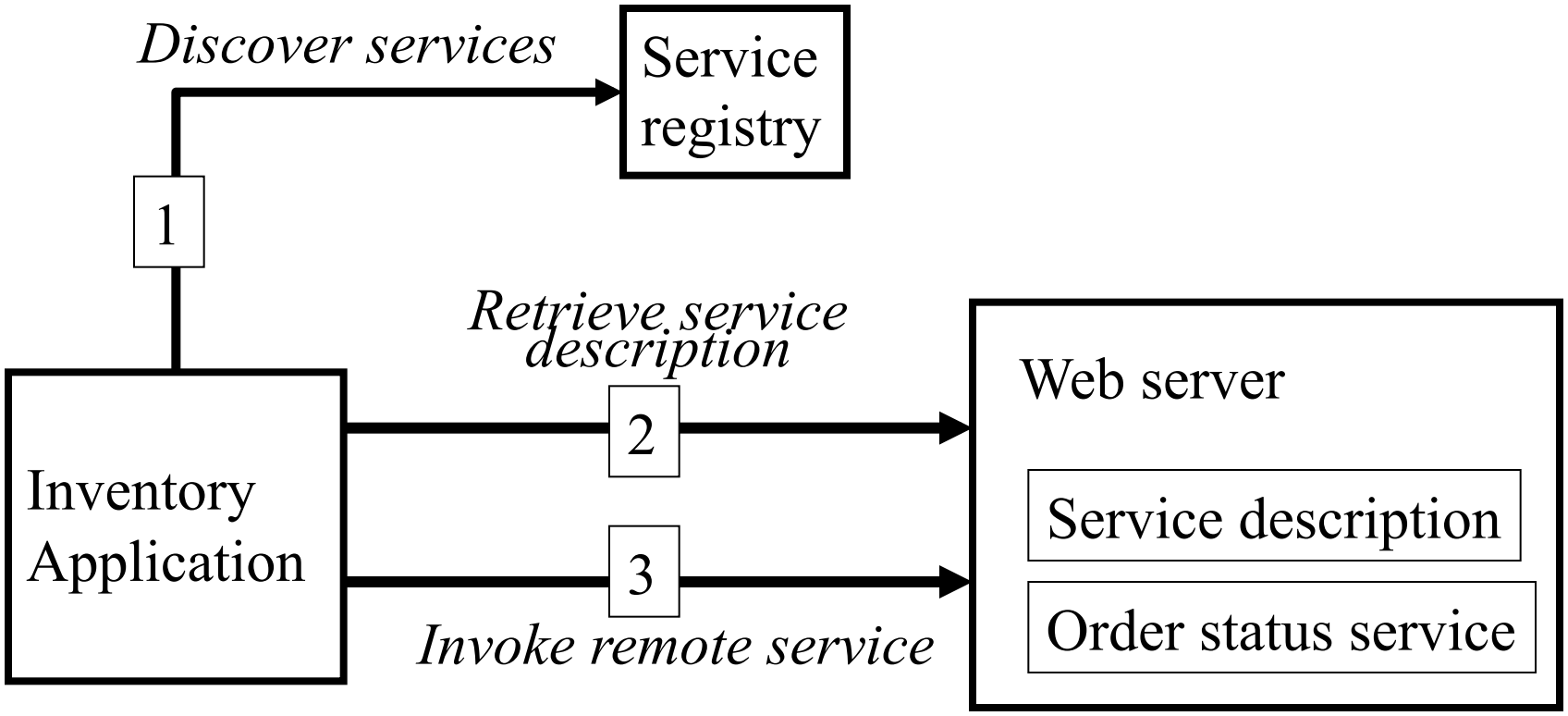
Adopted from Cerami

Web Services (application-centric web)



Adopted from Cerami

Web Services vision (automated web)



Adopted from Cerami

Web Services Advantages

- There is nothing special about Web Services
  - Many enabling technologies existed before
- Web Services may change the world!
  - Companies can expose and access Web Services using technology already in place
  - Web Services are more interoperable
  - Nearly all major software vendors have agreed to use the same core standards
  - All these factors indicate that Web Services have the capability to provide direct application-to-application communication
  - Web service have potential to change the entire process of designing, developing and deploying software

Web Services Advantages

- Operate using open, text-based standards (communication of components in different languages and platforms)
- Promote a modular approach to programming
- Comparatively easy and inexpensive implement?
- Significantly reduce the cost of enterprise application integration and B2B communications
- Can be implemented incrementally

Web Services and Application Service Providers (ASPs)

- ASPs provide customized (typically commonly used) business software applications over the Web
- Allow access necessary applications over the Internet
- Assume responsibility for maintaining the applications
- Web services, compare to ASPs, need not be developed and maintained by different bodies
- Web services can have different granularity

Service-oriented paradigm

- “everything is a service”
- Different services are autonomous and independent (loosely coupled)

Service Oriented Architecture

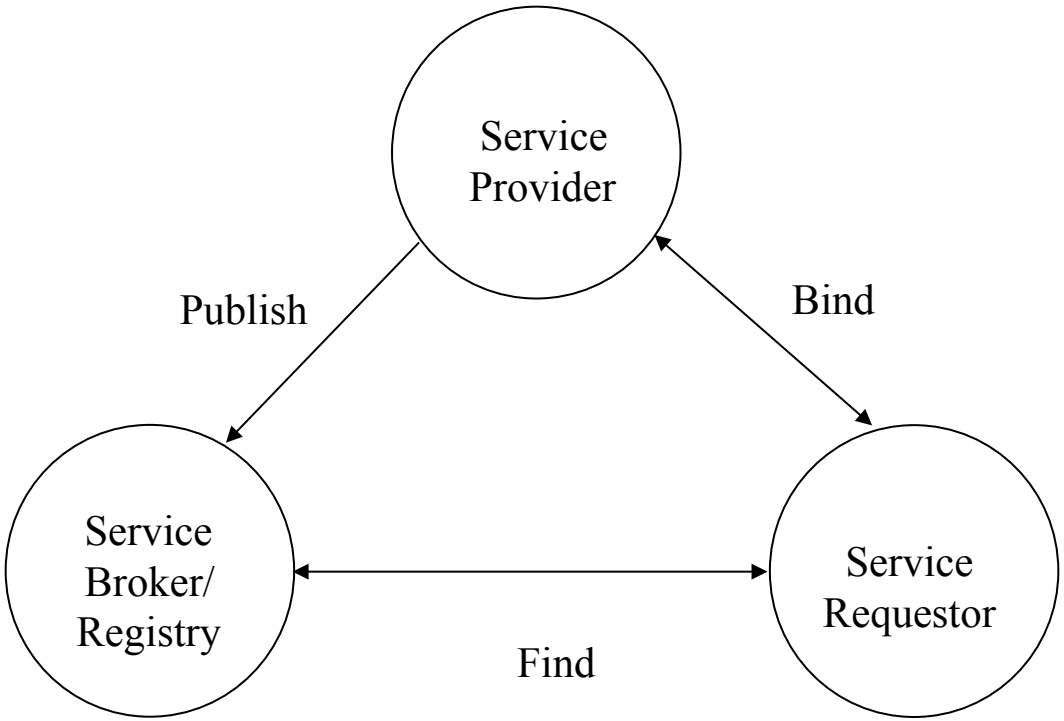
SOA is a form of distributed systems architecture that is typically characterized by the following properties:

- **Logical view**: The service is an abstracted, *logical* view of actual programs, databases, business processes, etc., defined in terms of what it *does*, typically carrying out a business-level operation
- **Message orientation**: The service is formally defined in terms of the messages exchanged between provider agents and requester agents
- **Description orientation**: A service is described by machine-processable meta data.
- **Granularity:** Services tend to use a small number of operations with relatively large and complex messages
- **Network orientation:** Services tend to be oriented toward use over a network
- **Platform neutral:** Messages are sent in a platform-neutral format

Service Oriented Architecture (SOA) (individual roles 1)

- Service provider
  - Implements service and makes it available on the Internet
- Service requestor
  - Utilizes existing web service by opening network connection and sending (XML) request
- Service registry
  - Place for publishing new services and finding existing ones

Service Oriented Architecture (individual roles 2)



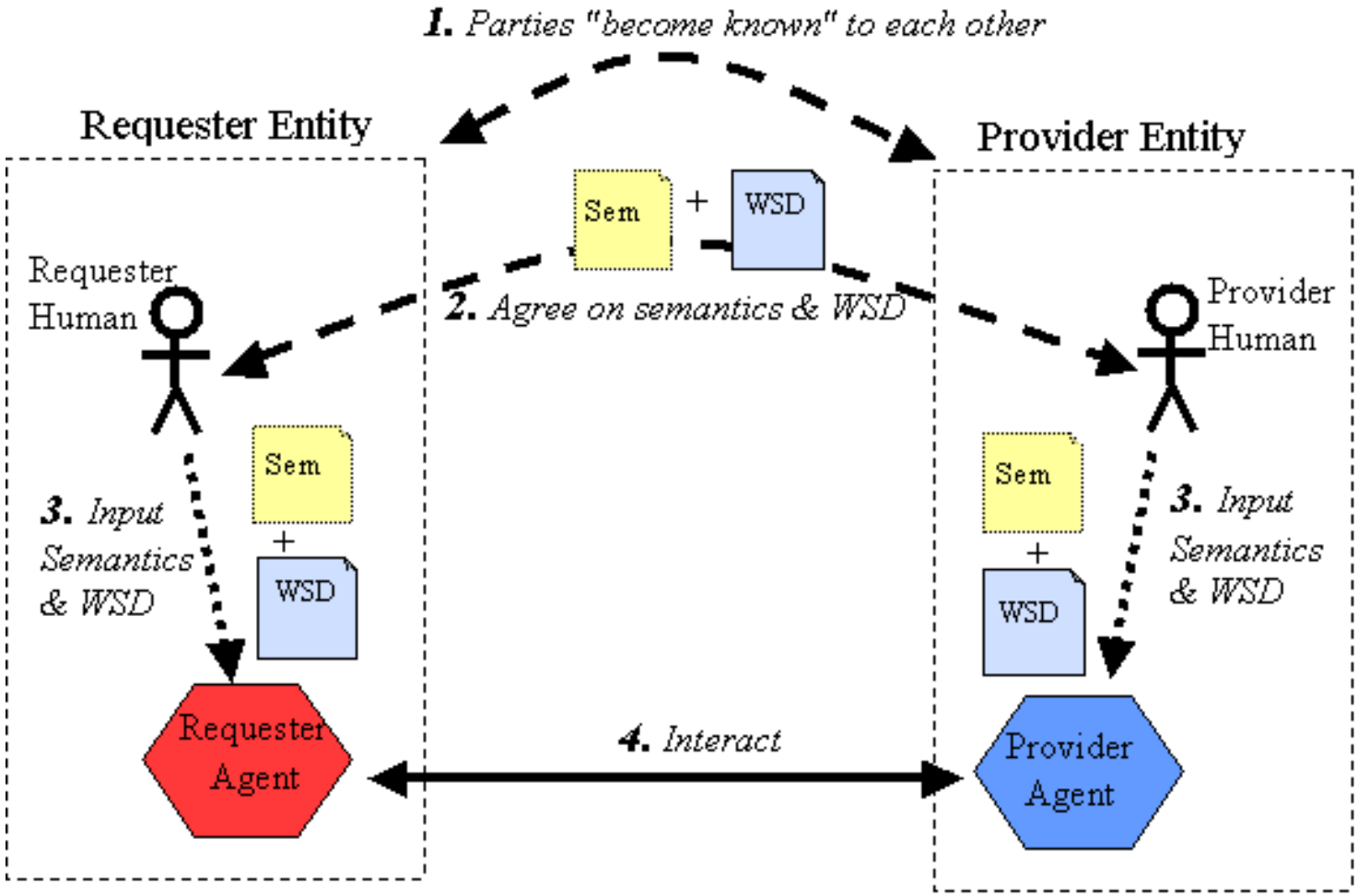
Why SOA is important

- With SOA approach the whole view to entire software portfolio is different.
- Existing applications can be easily converted to the services.
- Eventually monolithic, inflexible applications will be replaced by SOA –architecture applications
- Very important aspect – bridging IT concepts and business concepts together

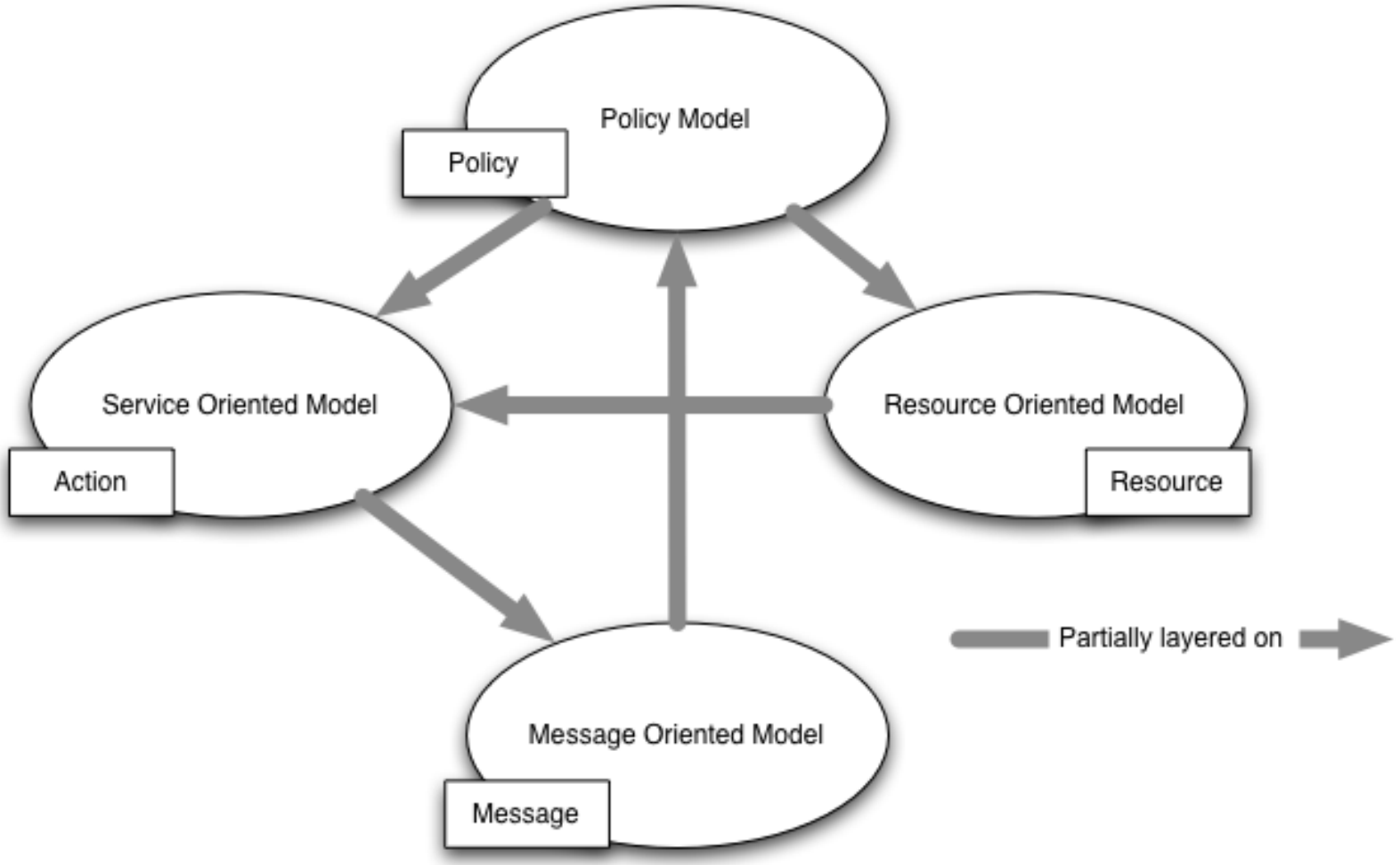
Web Service Architecture (W3C)

- The WSA describes both the minimal characteristics that are common to all Web services, and a number of characteristics that are needed by many, but not all, Web services.
- Web service is an abstract notion, while it is implemented by a particular agent
- The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided
- The purpose of Web service is providing some functionality on behalf of its owner
- Web Services Architecture is an interoperability architecture

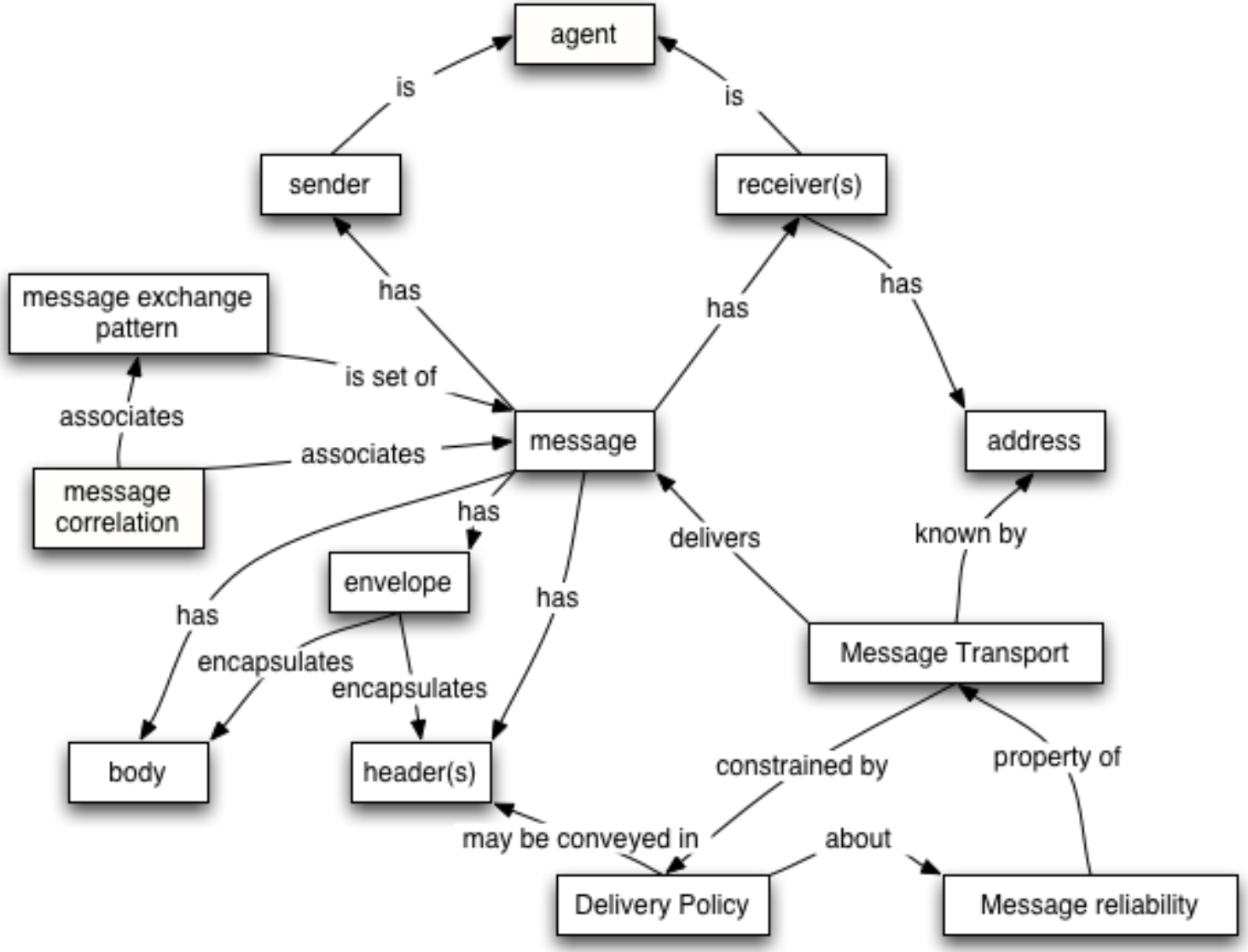
A protocol for services discovery and selection



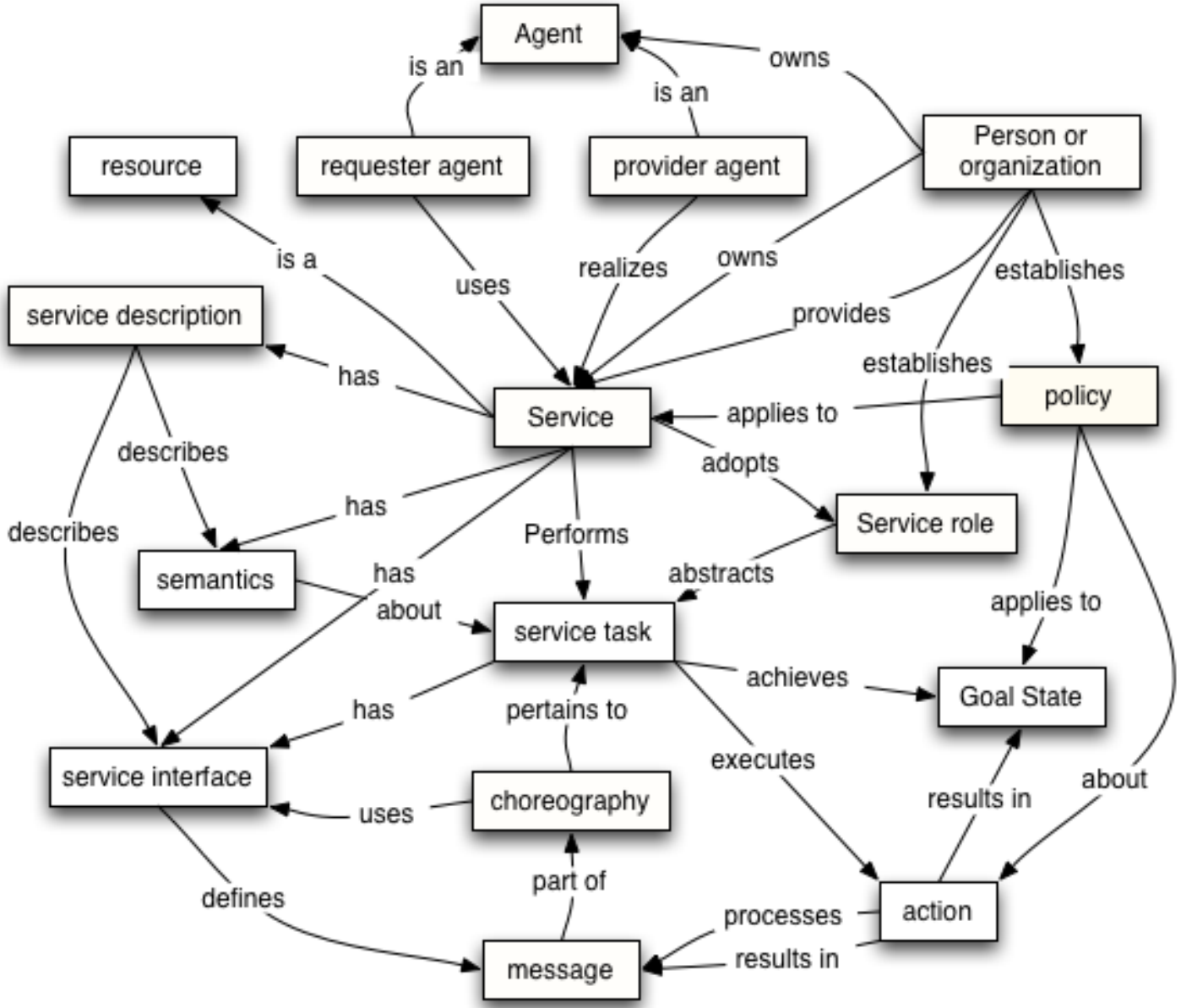
General Service Architecture (The Architectural Models)



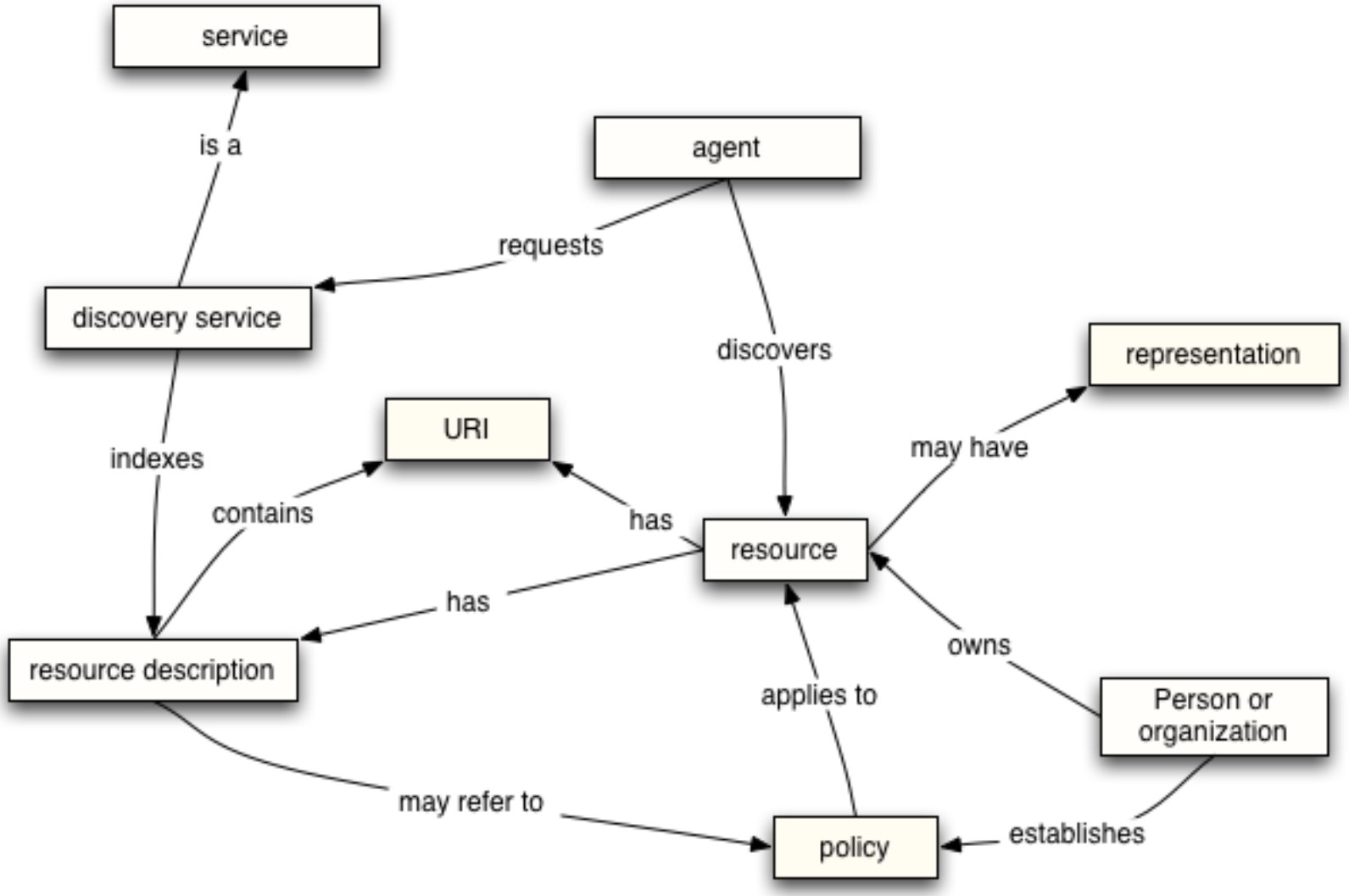
Message Oriented Model



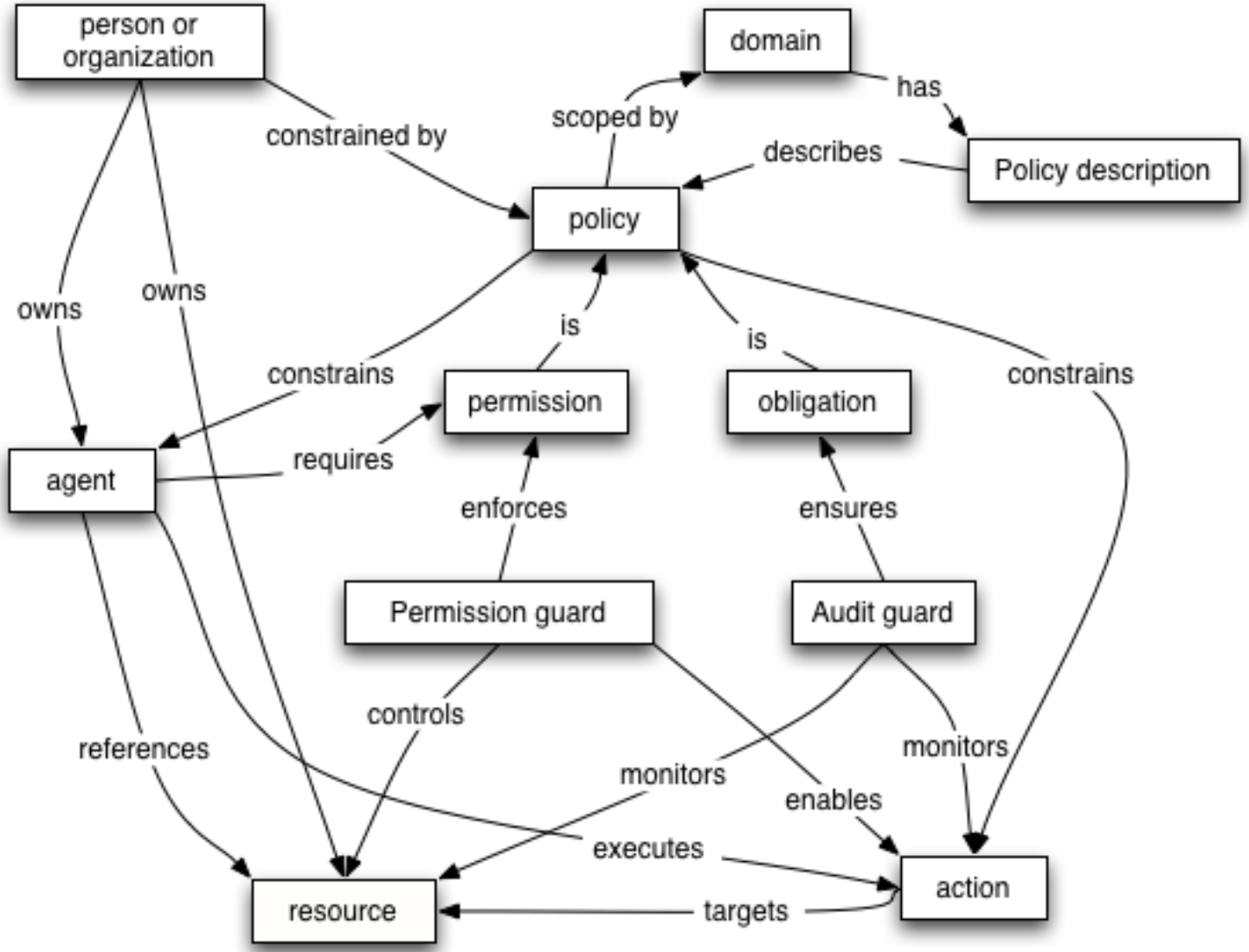
The Service Oriented Model



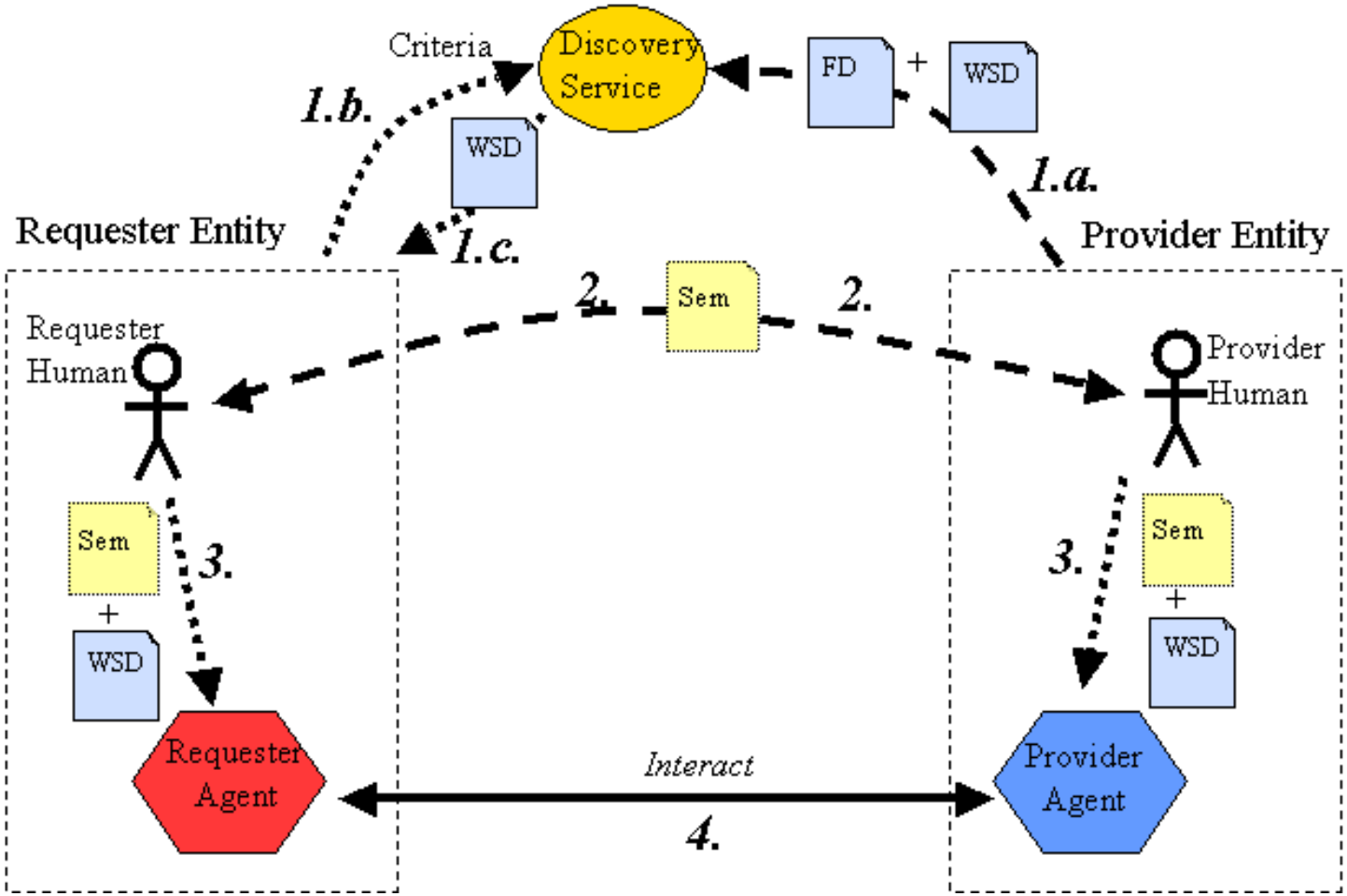
The Resource Oriented Model



The Policy Model



Discovery service

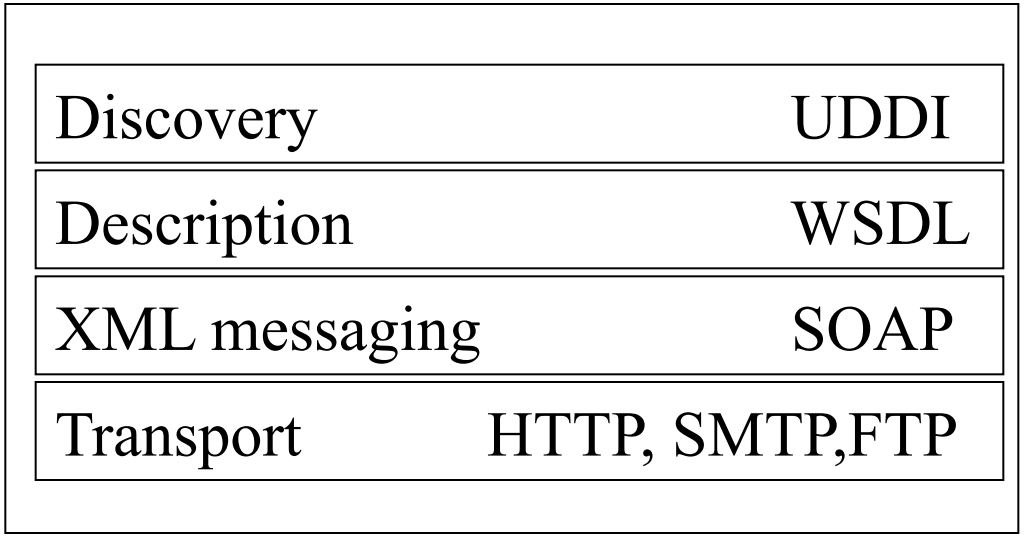


Discovery:

- The Registry Approach
- The Index Approach
- Peer-to-Peer (P2P) Discovery

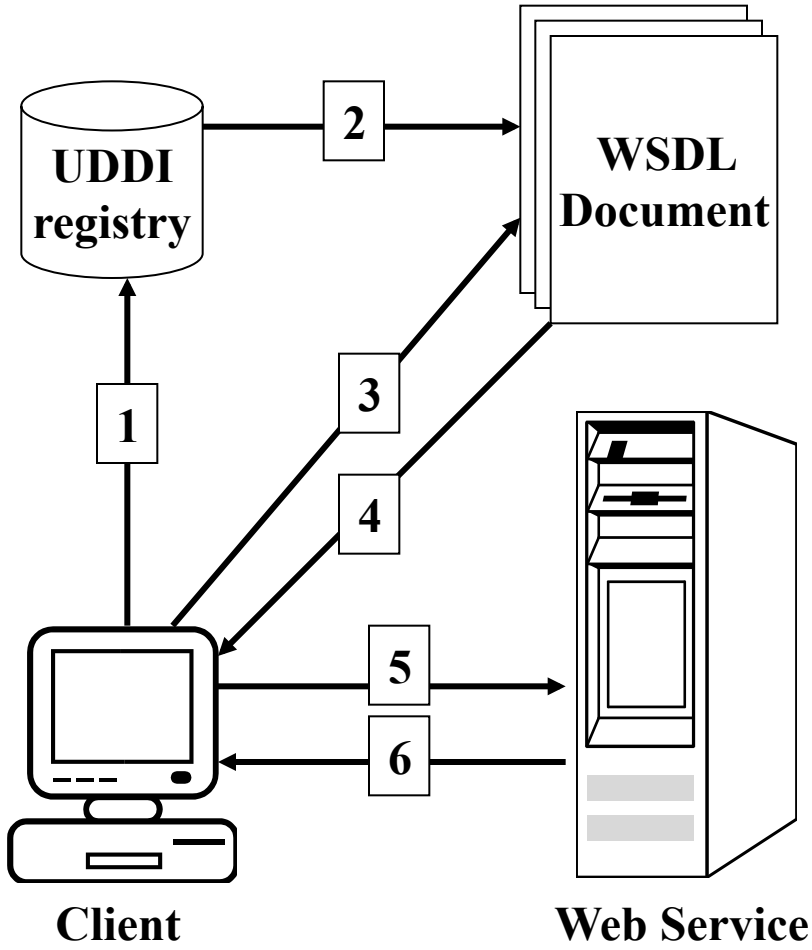
Web Services Architecture (protocol stack)

- Service transport
  - Transporting message between applications
- XML messaging
  - Encoding messages in a common XML format
- Service description
  - Describing the public interface to a specific service
- Service discovery
  - Centralizing services into a common registry



SOAP, UDDI and WSDL in Web Service interaction

1. Client queries registry to locate services
2. Registry refers client to WSDL document
3. Client accesses WSDL document
4. WSDL provides data to interact with Web Service
5. Client sends SOAP-message request
6. Web Service returns SOAP-message response



Adopted from Cerami

Service provider perspective

Step1: **Create core functionality**

Step2: **Create SOAP service wrapper**

Step3: **Create WSDL service description**

Step4: **Deploy service**

Step5: **Register new service via UDDI**

Adopted from Cerami

Service requester perspective

Step1: **Find service via UDDI**

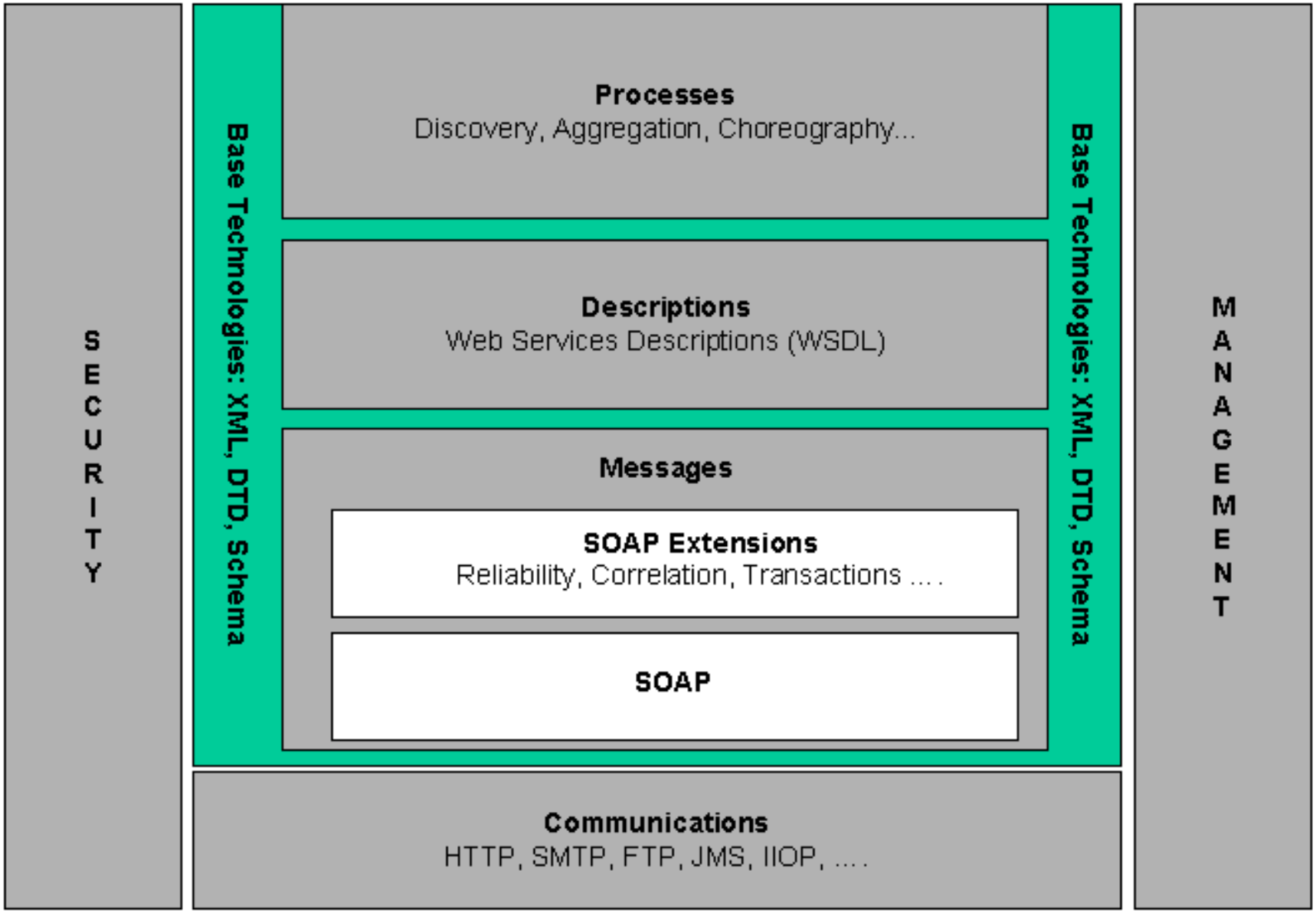
Step2: **Retrieve service description file: WSDL**

Step3: **Create SOAP client**

Step4: **Invoke remote service**

Adopted from Cerami

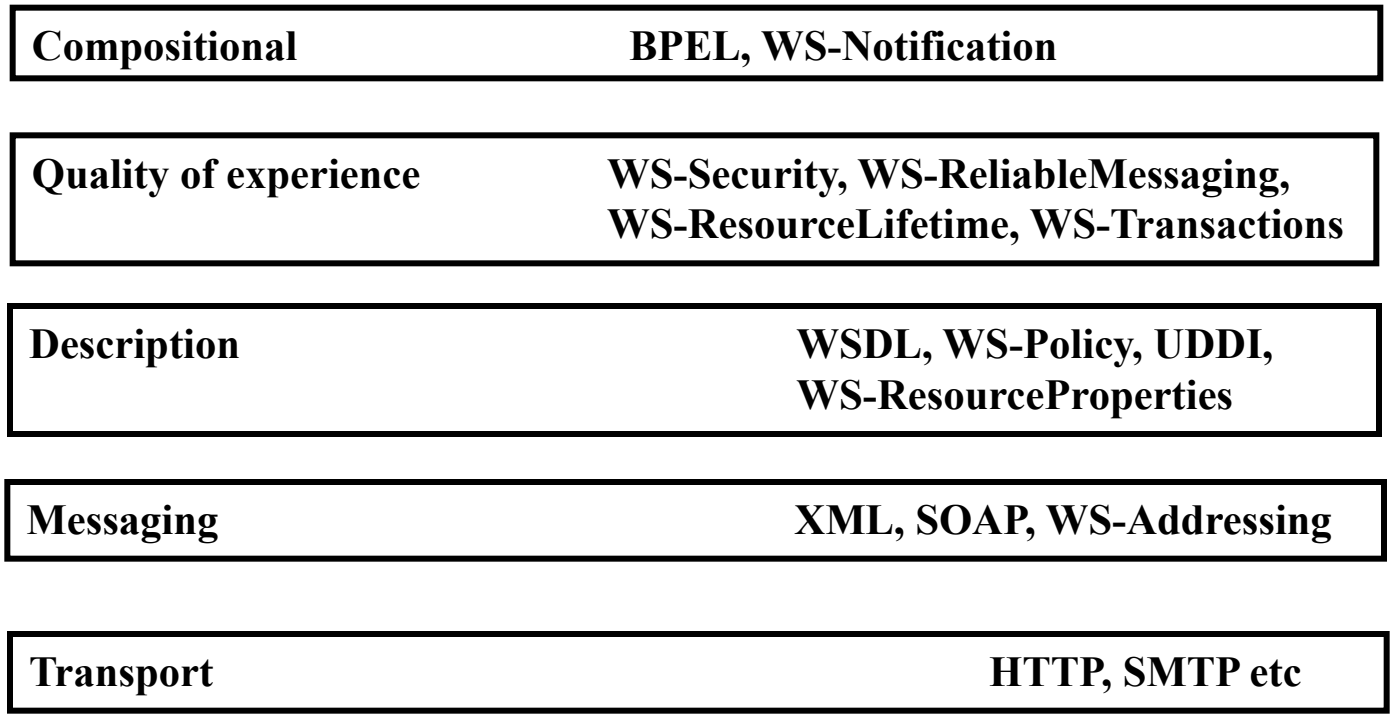
Web Service Architecture Stack



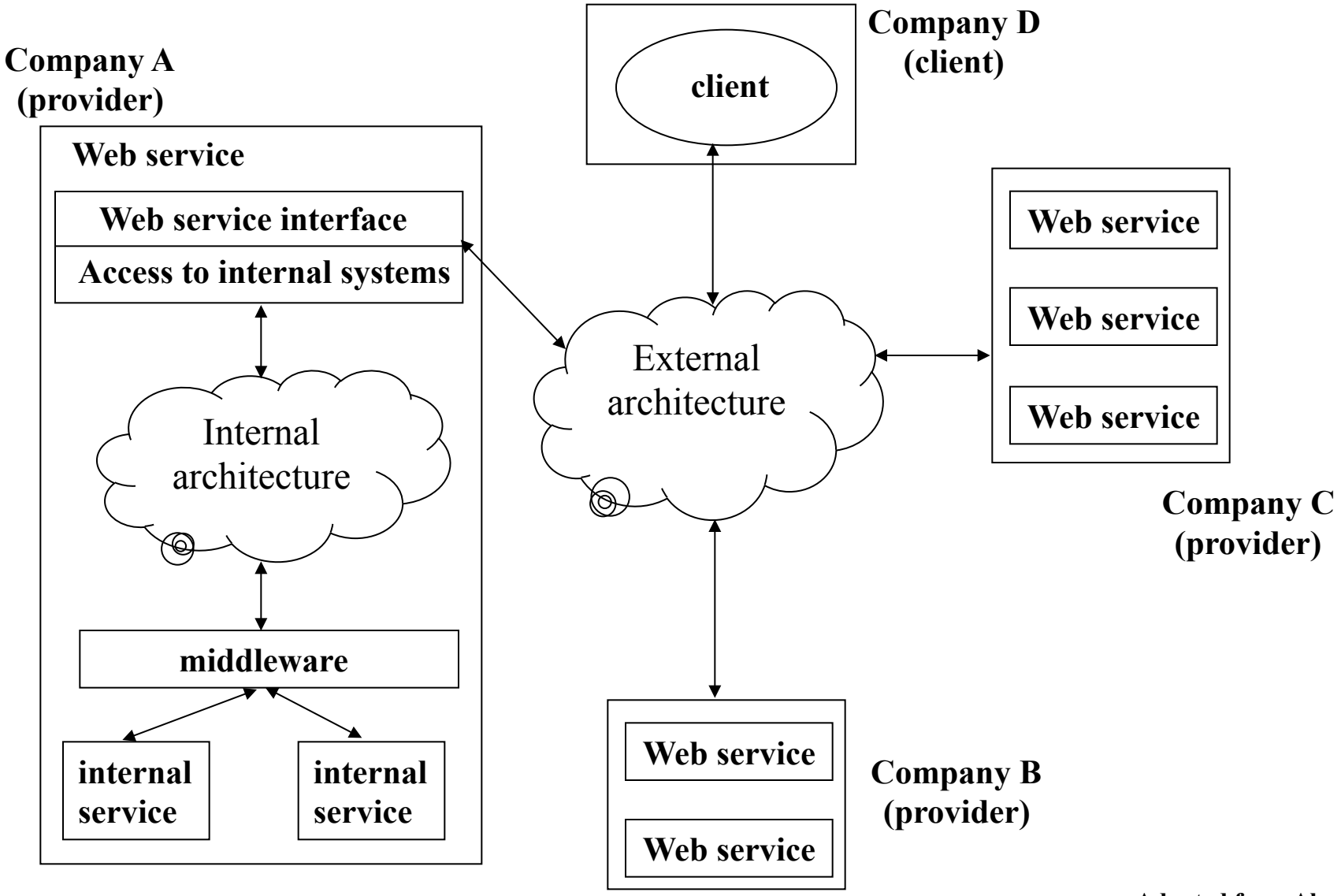
Web services interoperability stack

Adopted from Sobah Abbas Petersen

Web Service interoperability stack

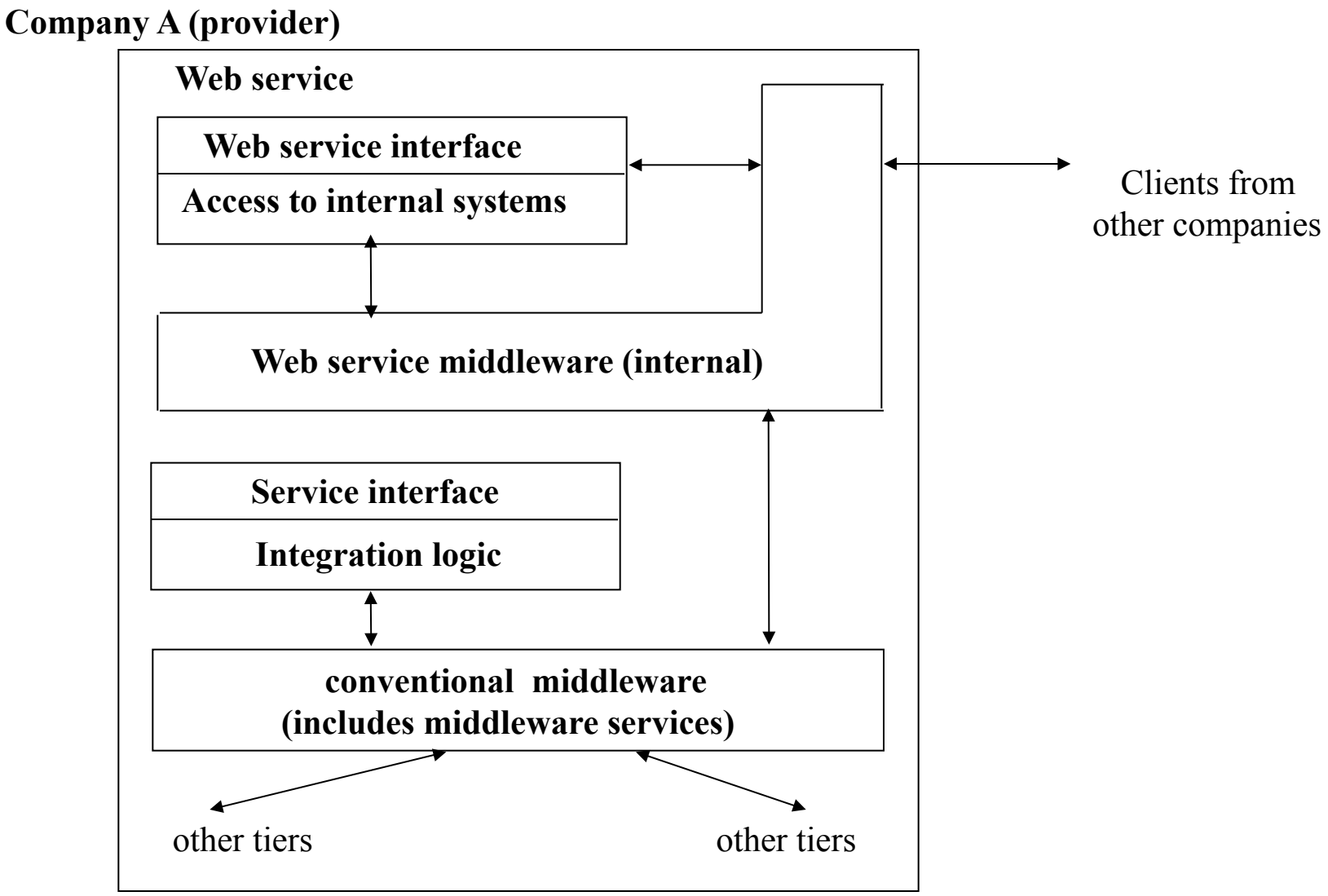


Two Facets of Web Services Architectures (implementation)



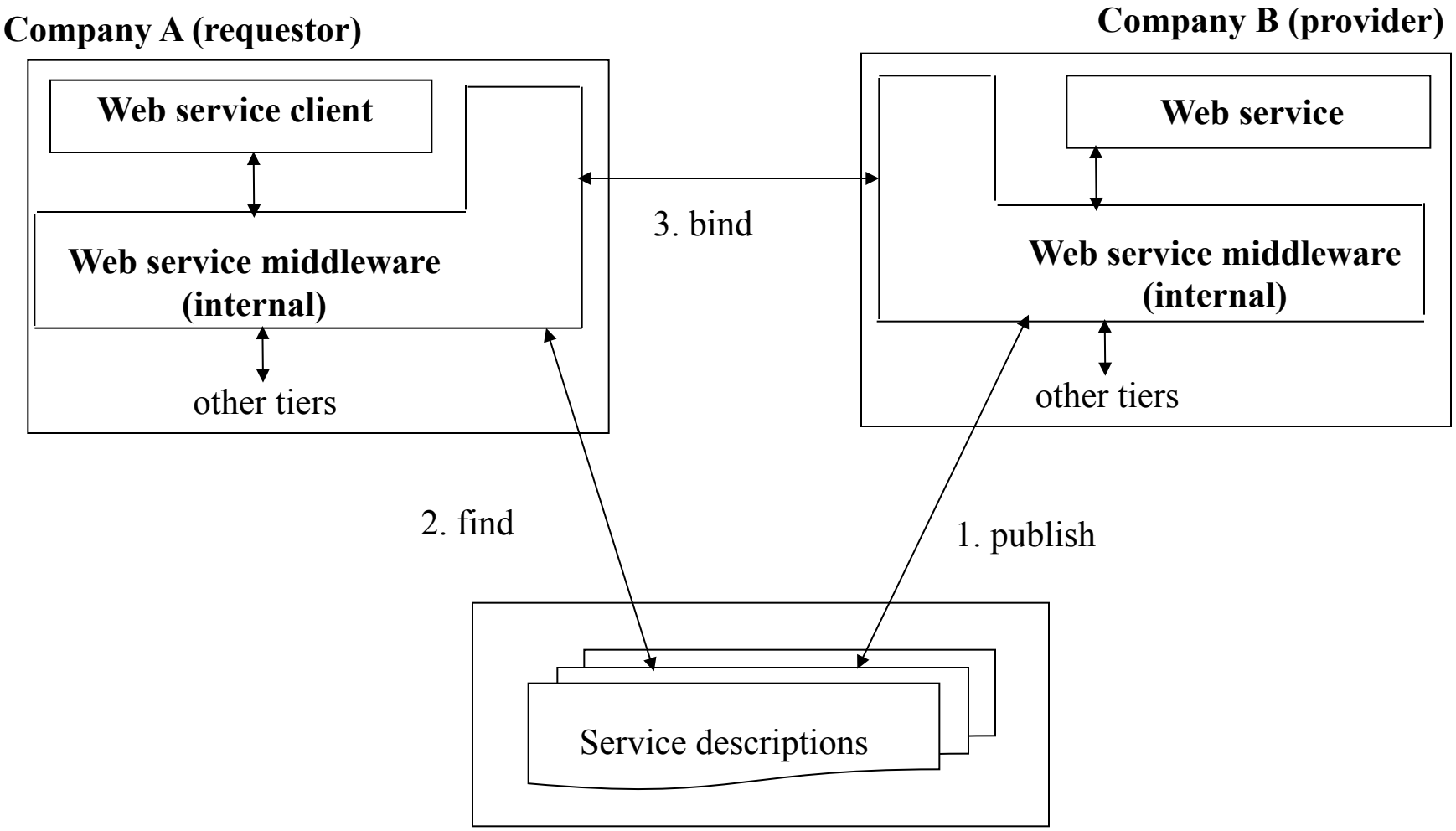
Adopted from Alomso et al.

Web Services Architectures (internal architecture)



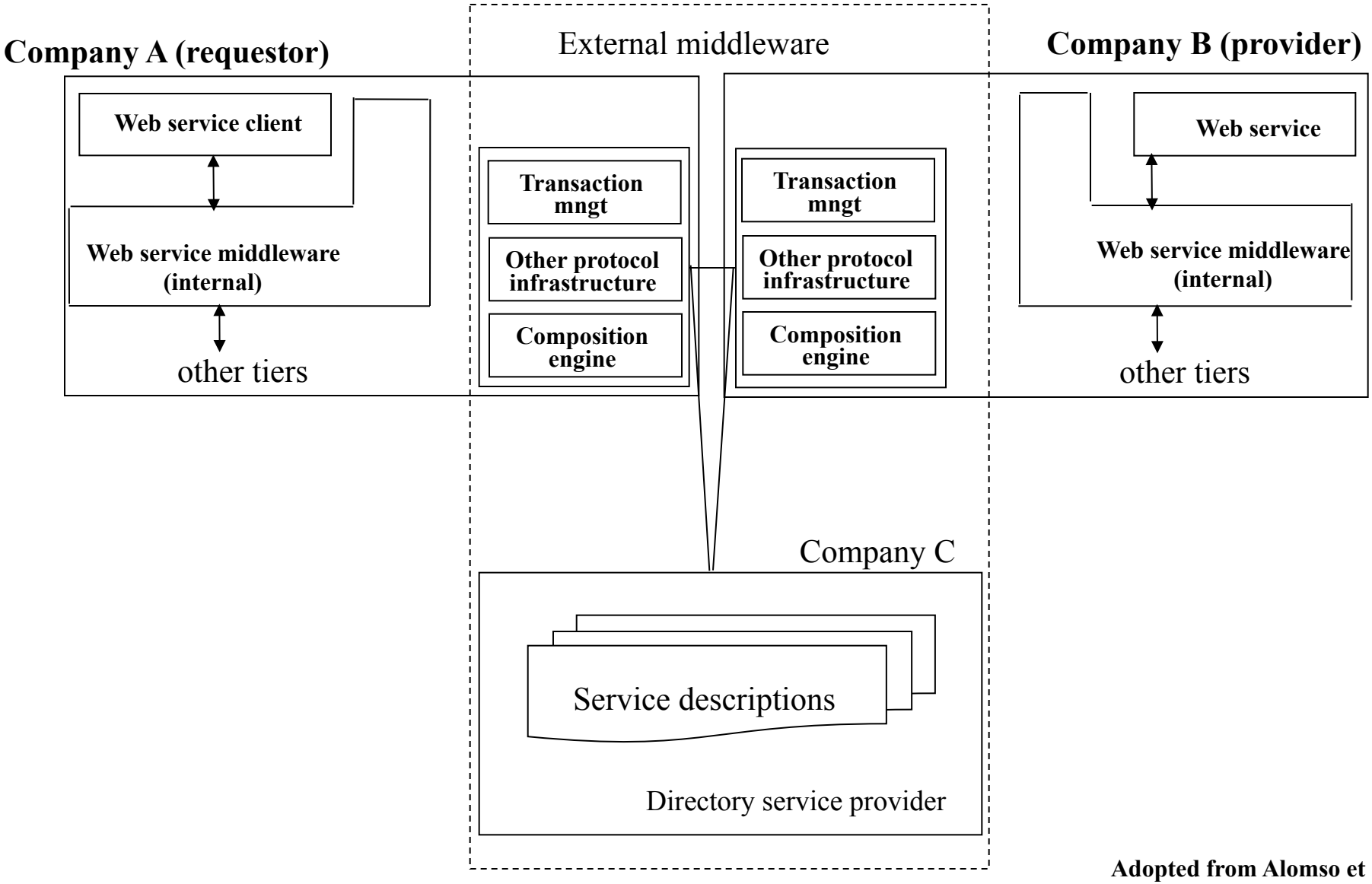
Adopted from Alomso et al.

Web Services Architectures (external architecture)



Adopted from Alomso et al.

External architecture for Web services augmented with P2P execution capability



Adopted from Alomso et al.

Next lecture

- XML basics

Text-book **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition**

Chapter 2