CMPU4021
Distributed Systems

Web Services

Introduction

- Web services have been created to solve a problem:
  - “How do you get applications to talk to each other automatically via the web?”

Web Services

- Set of protocols by which services can be published, discovered, and used in a technology neutral form
  - Language & architecture independent
- Provide a basis where a client program in one organisation may interact with a server in another organisation without human supervision.
- Based on the ability to use an HTTP request to cause the execution of a program.
  - An result is produced by called program and then returned.

Web Services

- General principles
  - Payloads are text (XML or JSON)
    - Technology-neutral
  - HTTP used for transport
    - Use existing infrastructure: web servers, firewalls, load-balancers
- Web server
  - Provides a basic HTPP service
- Web service
  - Provides a service based on the operation defined in its interface.
- Applications will typically invoke multiple remote services
  - Service Oriented Architecture (SOA)
  - SOA = Programming model

Service Oriented Architecture (SOA)

- SOA = Programming model
- App is integration of network-accessible services (components)
- Each service has a well-defined interface
- Components are
  - Unassociated
    - Neither service depends on the other: all are mutually independent
  - Loosely coupled
    - Neither service needs to know about the internal structure of the others

Benefits of SOA

- Autonomous modules
  - Each module does one thing well
  - Supports reuse of modules across applications

- Loose coupling
  - Requires minimal knowledge – don't need to know implementation
  - Migration:
    - Services can be located and relocated on any servers
  - Scalability:
    - new services can be added/removed on demand… and on different servers – or load balanced
  - Updates: Individual services can be replaced without interruption

General Principles of Web Services

- Platform neutral
  - Messages don’t rely on the underlying language, OS, or hardware
  - Standardized protocols & data formats
  - Payloads are text (XML or JSON)
- Message-oriented
  - Communicate by exchanging messages
- HTTP often used for transport
  - Use existing infrastructure: web servers, authentication, encryption, firewalls, load-balancers

Web Services

- Provide a standard way to package any business logic (i.e. a service)
  - and make it accessible to any other service (e.g. a business partner's systems)
  - via the web.

- A *web service*
  - a piece of business logic, located somewhere on the Internet, this is accessible through standard Internet protocols (e.g. HTTP)

- Can access and encapsulate other services to perform its function

Web services infrastructure and components

- Web services and applications may be built on top other web services.
- Some particular web services provide general functionality required for the operation of a large number of other web services:
  - Directory services,
  - Security
- A web service generally provides a *service description*, which includes an interface definition and other information, such as the server's URL

Properties of Web Services

- Combination of web services
  - The provision of a service interface allows its operations to be combined with those of other services to provide new functionality
- Communication patterns
  - In general, web services use either
    - Synchronous request-reply pattern of communication with their clients, or
    - They communicate by means of asynchronous messages
    - An event-style pattern can also be used:
      - e.g., clients of a directory service may register for events of interest – and they will be notified whenever certain events occur. For example, an event could be the arrival or departure of service.

Properties of Web Services

- No particular programming model
  - Designed to support distributed computing in the Internet, in which many different programming languages are used.
  - They are independent of any particular programming paradigm.
- The main differences from the distributed object model are:
  - Remote objects can not be instantiated – a web service consists of a single remote object and therefore:
    - Garbage collection is irrelevant;
    - Remote object references are irrelevant.

Properties of Web Services

- Service references
  - Each web service Has a URI, which clients use to refer to it. The URL is the most frequently used form of URI.

- Activation of services
  - A web service will be accessed via the computer whose domain name is included in its current URL. That computer may run the web service itself or it may run it on another server computer.
    - The URL is a persistent reference – meaning that it will continue to refer to the service for as long as that server exists.

Properties of Web Services

- Transparency
  - Middleware platform protects the programmer from the details of data representation and marshalling. None of these things is provided as part of an infrastructure or middleware platform for web services.
  - At the simplest level, clients and servers may read and write their messages directly in SOAP, using XML.
  - The details of SOAP and XML are generally hidden by a local API in a programming language such as Java, Perl, Python or C++.

Web Services vs. Distributed Objects

Web Services	Distributed Objects

• Document Oriented
  • Exchange documents
	• Object Oriented
	  • Instantiate remote objects
	  • Request operations on a remote object
	  • Receive result
	  • …
	  • Eventually release the object

• Document design is the key
  • Interfaces are just a way to pass documents
	• Interface design is the key
	  • Data structures just package data

• Stateless computing
  • State is contained within the documents that are exchanged (e.g., customer ID)
	• Stateful computing
	  • Remote object maintains state

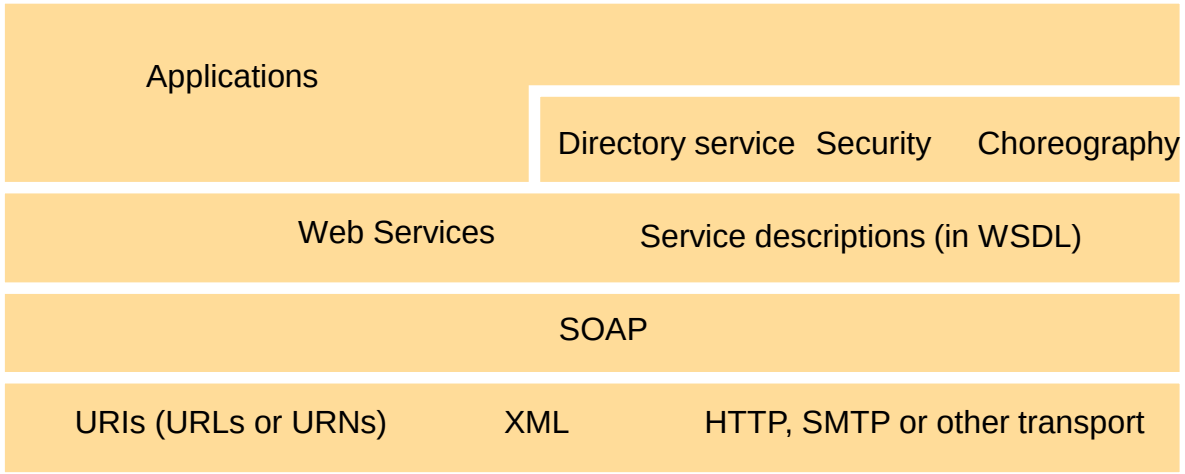
Web Services types

- Main types of web services
  - SOAP web services
    - SOAP is a protocol
  - RESTful web services
    - REST an architectural style not a protocol.

SOAP WEB SERVICES AND WSDL

Infrastructure and components

- Typical communication architecture in which web services operate:
  - A web service is identified by URI and can be accessed by clients using messages formatted in XML.
  - Simple Object Access Protocol (SOAP) is used to encapsulate these messages and transmit them over HTTP or another protocol, e.g. TCP or SMTP.
  - A web service deploys service descriptions to specify the interface and other aspects of the service for the benefit of potential clients



URI, URL and URN

- The Uniform Resource Identifier (URI)
  - a general resource identifier, whose value may be either a URL or a URN.
    - Uniform Resource Names (URNs) are location independent
      - they rely on a lookup service to map them onto the URLs of resources.

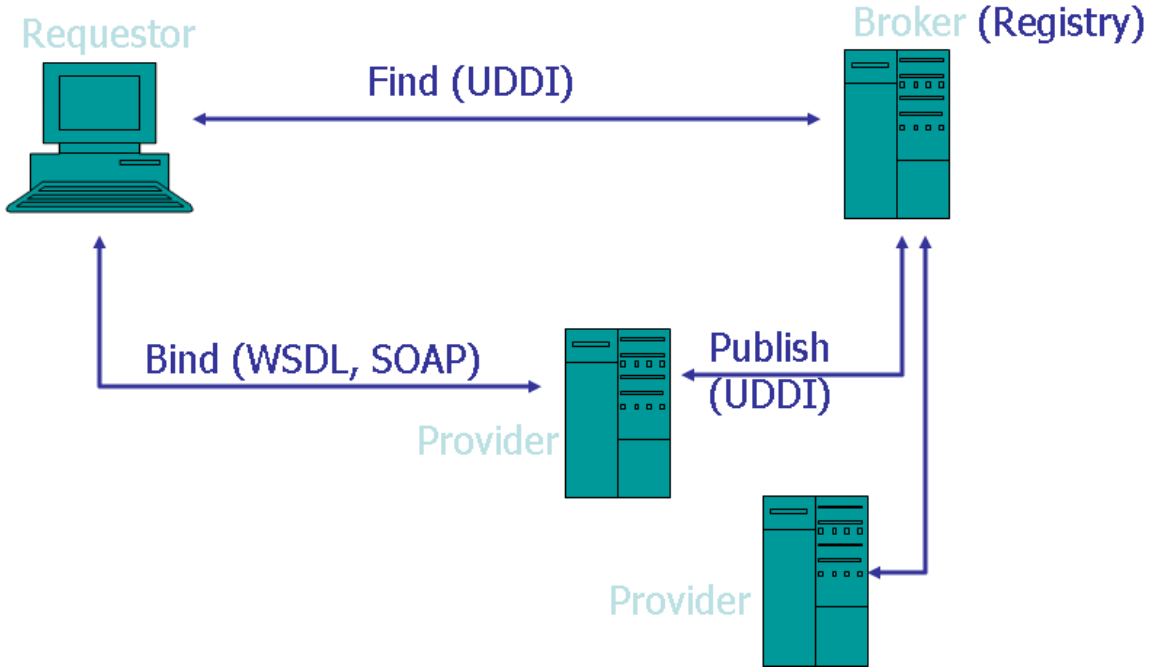
SOAP

- Designed to enable both client-server and asynchronous interaction over the Internet.
- It defines a scheme for using XML to represent
  - the contents of request and reply messages
  - a scheme for the communication of documents.
- Objects marshalled and unmarshalled to SOAP-format XML
- SOAP is a messaging format
  - No garbage collection or object references
  - Does not define transport

Web Services Architecture

Architecture needs to cater for:

- Providing a list of available services
  - E.g.what credit checking web services are available for a mortgage service to use?)
- Providing a description or specification for the service
  - (e.g.… what parameters does the selected credit check service expect.. client name/PPS number?, what does it return? …approval rating?)
- Sending messages between the client (I.e. *requestor* of the service) and service…
  - (… message to credit rating service from client application, containing client name etc.?)



Web Services Architecture

- A web service is identified by URI and can be accessed by clients using messages formatted in XML.
- SOAP is used to encapsulate these messages and transmit them over HTTP or another protocol, e.g. TCP or SMTP.
- A web service deploys service descriptions to specify the interface and other aspects of the service for the benefit of potential clients

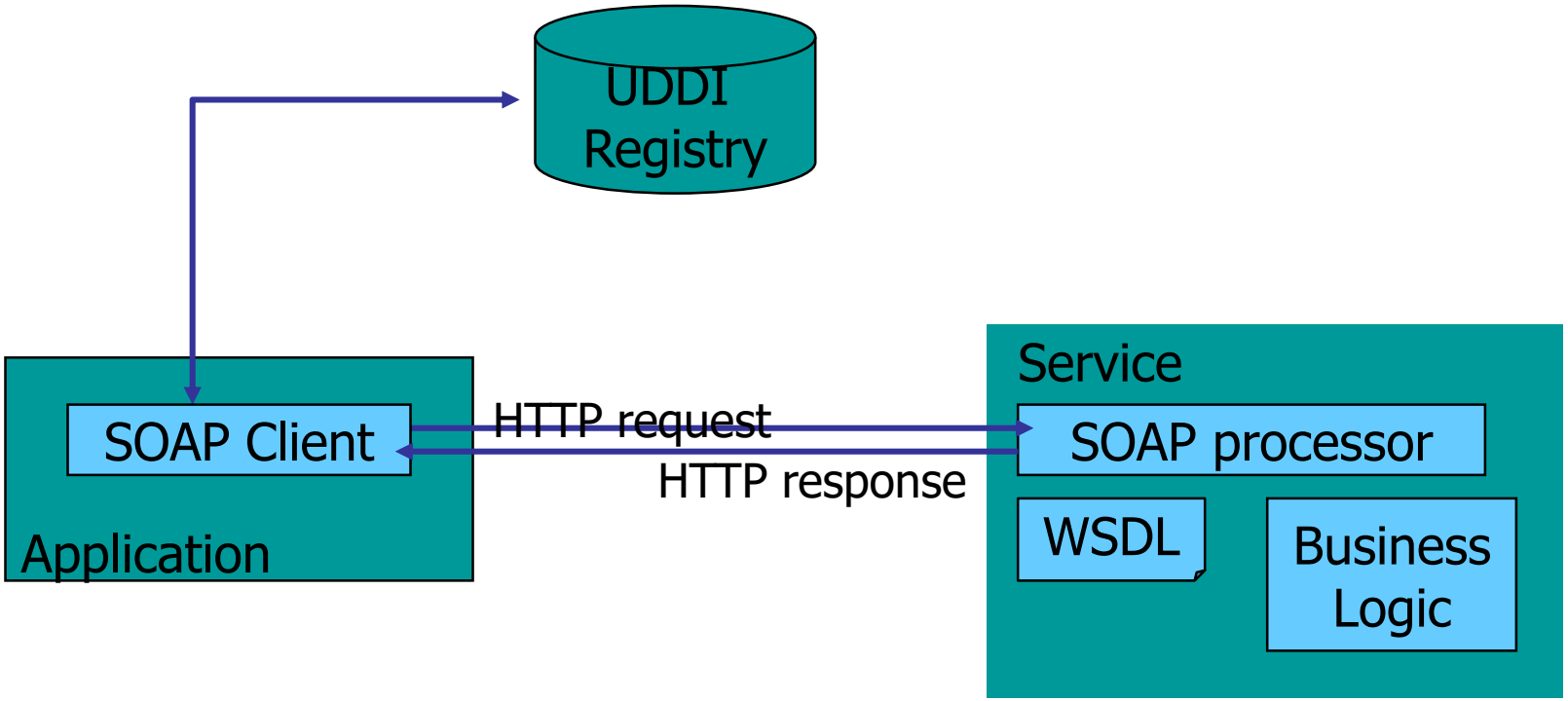
Technologies

Many implementations/frameworks for web services but common technologies:

- **SOAP** (Simple Object Access Protocol)
  - provides a way to communicate between applications running on different operating systems, with different technologies and programming languages
  - XML based protocol (on HTTP)
- **WSDL** (Web Service Description Language)
  - machine-readable descriptions of Web services interfaces.
  - XML based
- **UDDI (**Universal Description, Discovery and Integration)
  - UDDI provides an interface for publishing and updating information about web services.

Technologies in the Process

1. client queries UDDI registry for a service
   – by name, category, identifier or some other criteria stored by registry
2. client then obtains information about location of WSDL doc from UDDI registry
3. WSDL doc contains info about how to contact service and format of request msg
4. client creates SOAP msg in accordance with WSDL and sends request to host where service is
5. service responds with a SOAP msg to indicate results of service request



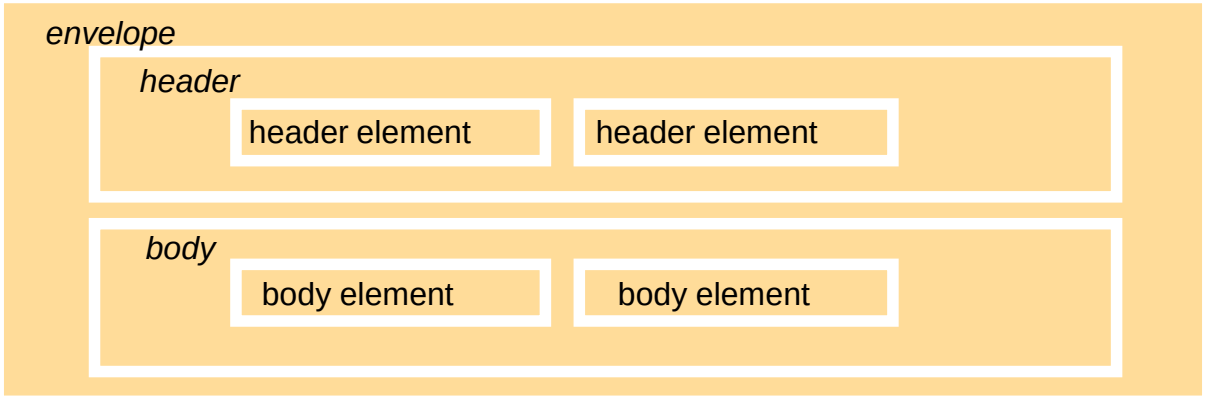
SOAP

- Simple Object Access Protocol
  - Extension of XML-RPC

- A SOAP message is an ordinary XML document containing the following elements:
  - An Envelope element that identifies the XML document as a SOAP message
  - A Header element that contains header information
  - A Body element that contains call and response information
  - A Fault element containing errors and status information

- Scheme for
  - Using XML to represent the contents of request and reply messages
  - Communication of documents

- Originally transported only over HTTP POST
  - Can also be transported over SMTP (e-mail), TCP, UDP



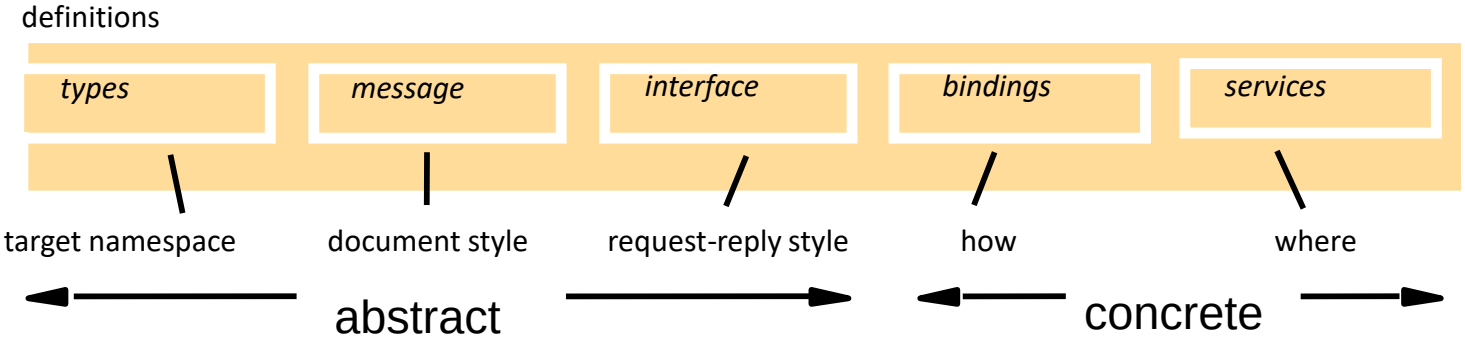
SOAP – the future?

- Considered heavyweight
  - XML
  - verbose messaging structure
- Still used, but:
  - Dropped by Google APIs in 2006
  - Still used by many, including Microsoft APIs

Service Descriptions

- Interface definitions needed to allow clients to communicate with services
- Java Remote Interfaces ~= IDL ~= WSDL
- WSDL (Web Services Description Language)
  - Describe operations, a set of services
  - Name, operations, parameters, where to send requests
  - Provide URI, Identify Transport Protocol
  - Organizations exchange WSDL documents
- All the information required by the client
- Describes either
  - Types of messages it can receive
  - Types of operations it can perform

The main elements in a WSDL description



<types>
data type used by web service: defined via XML Schema syntax
</types>
<message>
describes data elements of operations: parameters
</message>
<portType>
describes service: operations and messages involved
</portType>
<binding>
defines message format & protocol details for each port
</binding>

Directory Service

- A way to obtain service descriptions.
- UDDI (Universal Description, Discovery and Integration) of Web Services
  - An industry effort to define a searchable registry of services
- UDDI provides both
  - a name service and
  - a directory service

REST

Representational State Transfer (REST)

- The key characteristic of most web services is that they can process XML formatted SOAP messages
  - An alternative is the REST approach
- REST is a web standards based architecture
  - Uses HTTP Protocol for data communication
  - Resource-oriented
    - every component is a resource
    - a resource is accessed by a common interface using HTTP standard methods
- Clients use URLs and the HTTP operations GET, PUT, DELETE and POST to manipulate resources
  - The emphasis is on the manipulation of *data resources* rather than on interfaces.

REST

- When a new resource is created, it has a new URL by which it can be accessed or updated.
  - Clients are supplied with the entire state of a resource instead of calling an operation to get some part of it.
- The Amazon web services may be accessed either by SOAP or by REST

REST

- REST Server
  - provides access to resources
- REST client
  - accesses and presents the resources
- REST resources
  - each resource is identified by URIs/ Global IDs
  - representations of a resource
    - Text, JSON and XML
    - JSON is now the most popular format

RESTful Web Services

- A web service is:
  - A collection of open protocols
  - Standards used for exchanging data between applications or systems
  - Interoperability between different languages (Java and Python) or platforms (Windows and Linux)

- Web services based on REST Architecture are known as RESTful Web Services
  - Use HTTP methods to implement the concept of REST architecture
  - URI (Uniform Resource Identifier) to define a RESTful service
  - Resources representation: JSON

REST

- Everything is a resource
- Any interaction of a RESTful API is an interaction with a resource.
- Resources are sources of information,
  - typically documents or services, or
  - Users (e.g. as a URL of their GitHub)

REST characteristics

- Resources are identified through a single naming scheme.
- All services offer the same interface via basic operations
- Messages sent to or from a service are fully self-described
- Component forgets everything about the caller
  - after executing an operation at a service

REST

- Describes how resources on web servers should be accessed via the HTTP protocol
- Resource identification through a uniform resource identifier (URI)

HTTP Methods in a REST based architecture

Basic *four* operations (CRUD: *Create, Read, Update, Delete*)

- PUT
  - Create, Used to create a new resource.
- GET
  - Read, Provides a read only access to a resource.
- POST
  - Update, Used to update an existing resource or create a new resource.
- DELETE
  - Delete, Used to remove a resource.

Fifth operation – determine options associated with a resource

- OPTIONS − Query, Used to get the supported operations on a resource.

REST conventions

- Easily understood consistent naming scheme
- Traversing the path from more generic to more specific, you are navigating the data

www.myappdomain.com

- retrieve a list of users: GET www. myappdomain.com/users
- retrieve user 23: GET www. myappdomain.com/users/23
- create a new user: POST www. myappdomain.com/users
- update user 23: PUT www. myappdomain.com/users/23
- remove user 23: DELETE www.mmyappdomain.com/users/23

Examples of REST services

- Various Amazon & Microsoft APIs
- Facebook Graph API
- Yahoo! Search APIs
- Flickr
- Twitter

SOAP vs RESTful Web Services

SOAP

- A protocol
  - comes with strict rules and advanced security features such as built-in ACID compliance and authorization.
- Permits XML data format only
- SOAP can't use REST
- Has higher complexity, and requires more bandwidth and resources, which can lead to slower page load times.

REST

- An architectural style
  - not a protocol
- Can use different messaging formats, such as HTML, CSV, JSON, XML
- REST can use SOAP web services because it is a concept
- Can use any protocol like HTTP, SOAP.
- Consumes fewer resources than SOAP because its messages are typically smaller.- better performance

SOAP versus REST

SOAP

- API calls cannot be cached
- Only XML
- Requires more bandwidth and computing power
- Better security, built-in extensibility, standardized
- Usage:
  - Banks or payment gateways, PayPal

REST

- API calls can be cached
- Plain text, HTML, XML, JSON, CSV…
- Requires fewer computing resources
- Faster, scalability
- Usage:
  - Facebook, Google, Twitter

COORDINATION OF WEB SERVICES

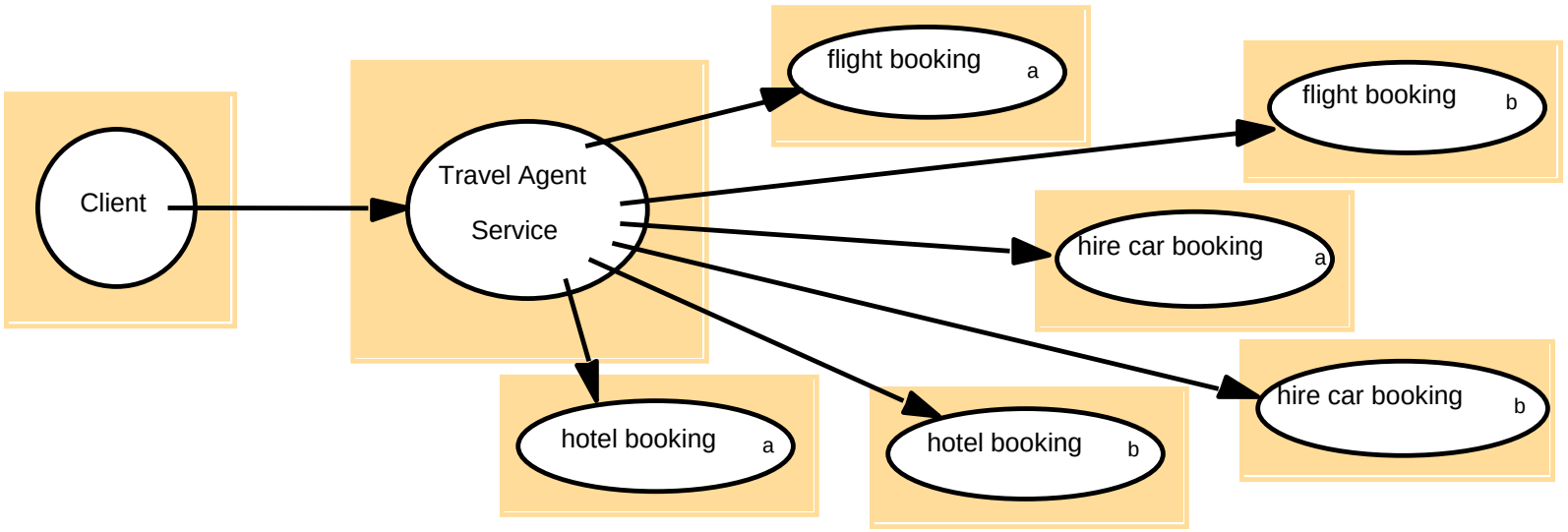
Coordination of web services

- The SOAP infrastructure supports single request-response interactions between clients and web services.
- Many applications involve several requests that need to be done in a particular order. The need for web services as clients to be provided with a description of a particular protocol to follow when interacting with other web services.
- For composite web services, a transactions management protocol such as 2PC is required
  - WS-Coordination
- Simpler approach is Web Service choreography
  - Global description of a set of interactions
  - Defines coordination
  - Enhances interaction

Travel agent scenario

1. The client asks the travel agent service for information about a set of services; for example, flights, car hire and hotel bookings.
2. The travel agent service collects prices and availability information and sends it to the client, which chooses one of the following on behalf of the user:
   (a) refine the query, possibly involving more providers to get more information, then repeat step 2;
   (b) make reservations;
   (c) quit.
3. The client requests a reservation and the travel agent service checks availability.
4. Either all are available;
   or for services that are not available;
   either alternatives are offered to the client who goes back to step 3;
   or the client goes back to step 1.
5. Take deposit.
6. Give the client a reservation number as a confirmation.
7. During the period until the final payment, the client may modify or cancel reservations

Fig: The ‘travel agent service’ combines other web services



Applications of web services

- The major areas where web services have been employed extensively:
  - Service-oriented architecture
  - The Grid, and
  - Cloud computing

The Grid

- Middleware that is designed to enable the sharing of resources such as files, computers, software, data and sensors
  - on a very large scale

- An example of a Grid application
  - The World-Wide Telescope application developed at Microsoft Research.
    - This project is concerned with deploying the data resources shared by the astronomy community.

Cloud computing

- A set of Internet-based application, storage and computing services sufficient to support most users' needs
  - enabling them to largely or totally dispense with local data storage and application software.

- Promotes a view of everything as a service
  - from physical or virtual infrastructure,
  - through to software, often paid for on a per-usage basis rather than purchased.

- Examples:
  - Google App Engine
  - *Amazon Web Services* (AWS) [aws.amazon.com]
    - A set of cloud services implemented on the extensive physical infrastructure owned by Amazon.com.

Grid and Cloud Computing

- The development of the Grid preceded the emergence of cloud computing and was a significant factor in its emergence.
- They share the same goal of providing resources (services) out there in the greater Internet.
- Grid
  - tends to focus on high-end data-heavy or computationally expensive applications
- Cloud computation
  - more general, offering a range of services for individual computer users through to high-end users.
- The business model associated with cloud computing is also a distinguishing characteristic
  - The Grid is an early example of cloud computing, but cloud computing has developed significantly since then.

Amazon Web Services (AWS)

- A set of cloud services implemented on the extensive physical infrastructure owned by Amazon.com.

- Made available using web service standards described earlier.

AWS

- Amazon's AWS is based entirely on web service standards coupled with the REST philosophy of service construction.
- The approach enables interoperability across the Internet.
- Amazon also adopts the REST approach

Comparison of web services with distributed object model

- A web service has a service interface which can provide operations for accessing and updating the data resource it manages.

- At a superficial level, the interaction between client and server is similar to RMI, where a client uses a remote object reference to invoke an operation in a remote object.
  - For a web service, the client uses a URI to invoke an operation in the resource named by that URI.

Comparison of web services with distributed object model

- Remote object references are not very similar to URIs
  - The URI of a web service can be compared with the remote object reference of a single object.
  - However, in the distributed object model, objects can create remote objects dynamically and return remote reference to them.
  - The recipient of these remote references can use them to invoke operations in the object to which they refer.

Comparison of web services with distributed object model

- Servants
  - In the distributed object model, the server program is generally modelled as a collection of servants (potentially) remote objects.
  - In contrast, web services do not support servants. Therefore, web services applications cannot create servants as and when they are needed to handle different resources.
    - To enforce this situation, the implementation of web service interfaces must not have either constructors or main methods.

Web Service Examples

- An application that requires the presence of a web service is one that implements ‘sniping’ in eBay auctions
  - placing a bid during the last few seconds before an auction closes.
  - although humans can perform the same actions by direct interaction with the web page, they cannot do it as quickly.
- An inventory control and purchasing application that might order supplies of various commodities as they are needed from Amazon.com and automatically keep track of the changing status of each order.

References

- Chapter 9 - Coulouris, Dollimore, Kindberg and Blair (2012), Distributed Systems, Concepts and Designs (Edition 5), Addison Wesley

- RPC & Web Services: Case Studies, Paul Krzyzanowski, Rutgers University

- https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

- https://www.ibm.com/docs/en/was/8.5.5?topic=technologies-web-services