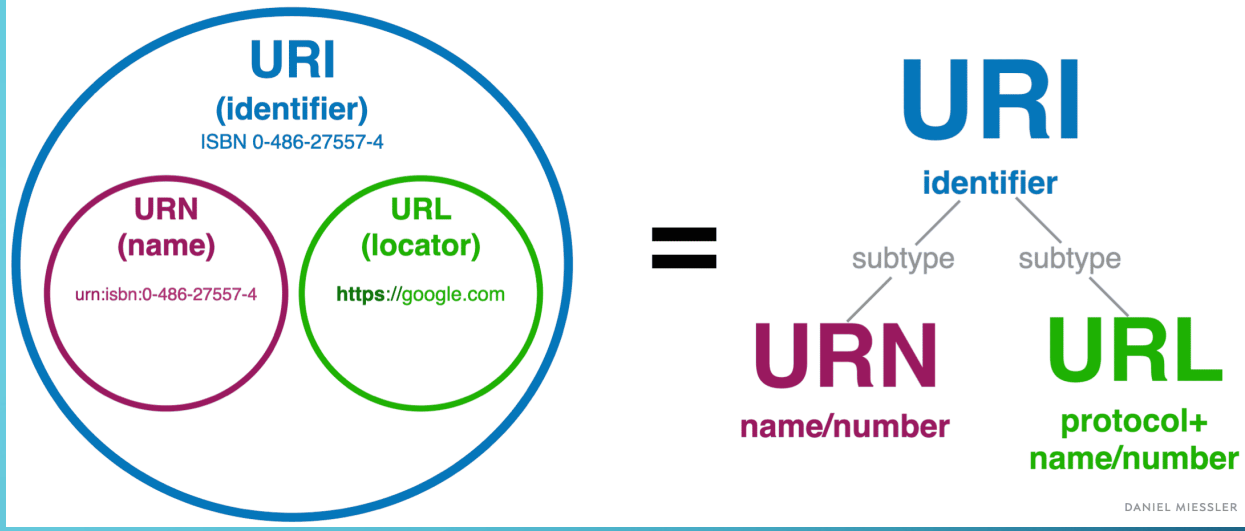
DISTRIBUTED AND CLOUD COMPUTING

LAB 6 WEB SERVICES

INTRODUCTION

- Web services are designed to support distributed computing on the Internet, in which many different programming languages and paradigms coexist. they are designed to be independent of any particular programming paradigm.
- Web services are accessed through Uniform Resource Identifiers (URIs) by clients using formatted messages.
- A web service interface generally consists of a collection of operations that can be used by a client over the Internet.

INTRODUCTION

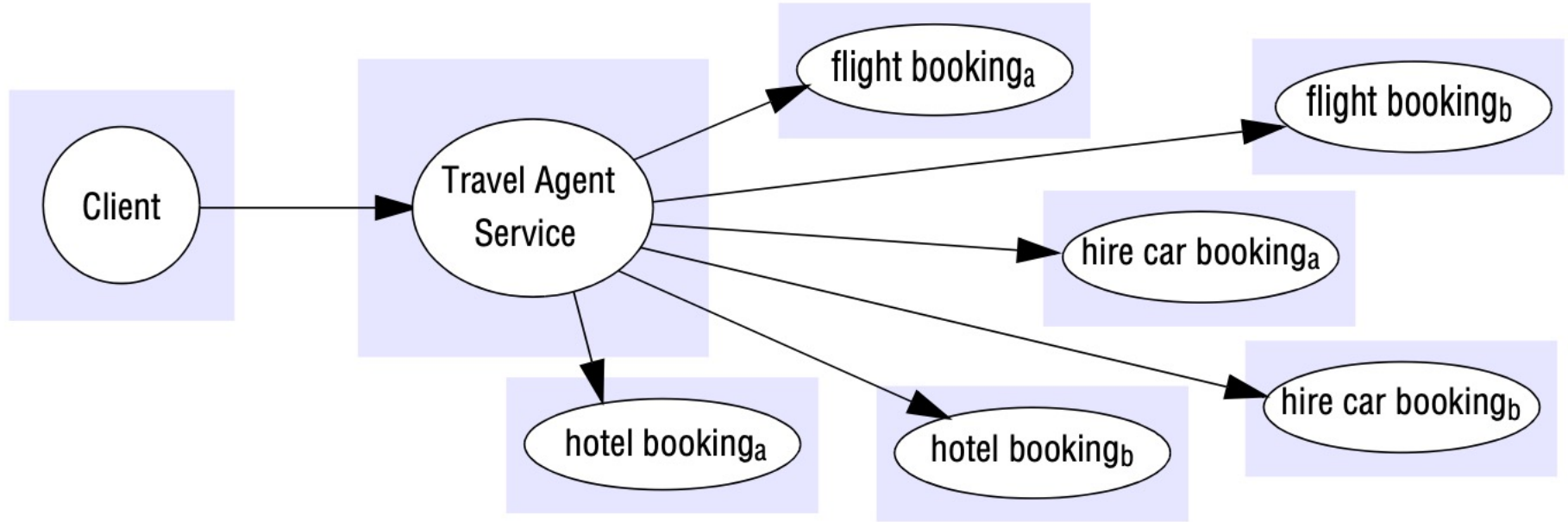


- Web services use service descriptions
  - more general than interface definitions
  - specifying:
    - Interface definition
    - Endpoint of the service (URIs)
    - Protocols used (e.g. SOAP and HTTP) for message-based communication

CHARACTERISTICS OF WEB SERVICE

- Combination of web services

Figure 9.2 The 'travel agent service' combines other web services



CHARACTERISTICS OF WEB SERVICE

- Loose coupling: minimising the dependencies between services in order to have a flexible underlying architecture (reducing the risk that a change in one service will have a knock-on effect on other services).
  - Programming with interfaces, separating the interface from its implementation
  - A trend towards simple, generic interfaces in distributed systems – the REST approach in web services
    - data becomes more important than operation
  - Can use a variety of communication paradigms, including request-reply communication, asynchronous messaging or indeed indirect communication paradigms

CHARACTERISTICS OF WEB SERVICE

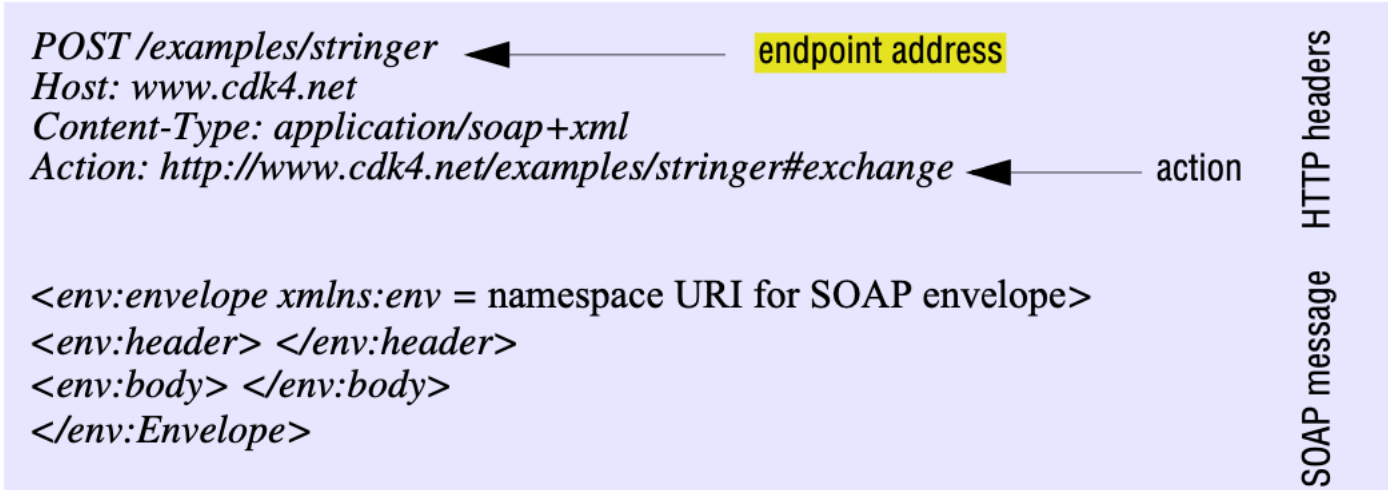
- Representation of messages
  - Textual representations, vs. binary
    - Both SOAP and the data it carries are represented in XML
- Service references
  - Each web service has a URI. Clients can use that URI to refer to the service. URL is the most frequently used form of URI.

SOAP

- Simple Object Access Protocol
- It defines a scheme for using XML to represent the contents of request and reply messages as well as a scheme for the communication of documents
  - how XML can be used to represent the contents of individual messages;
  - how a pair of single messages can be combined to produce a request-reply pattern;
  - the rules as to how the recipients of messages should process the XML;
  - how HTTP and SMTP should be used to communicate SOAP messages.

SOAP

Use of HTTP POST Request in SOAP client-server communication



SOAP WEB SERVICES AND WSDL

- Web Services Description Language
  - Analogous to an IDL
- A WSDL document describes a set of services
  - Name, operations, parameters, where to send requests
  - The goal is that organizations will exchange WSDL documents
    - If you get a WSDL document, you can feed it to a program to generate software that is able to send and receive SOAP messages

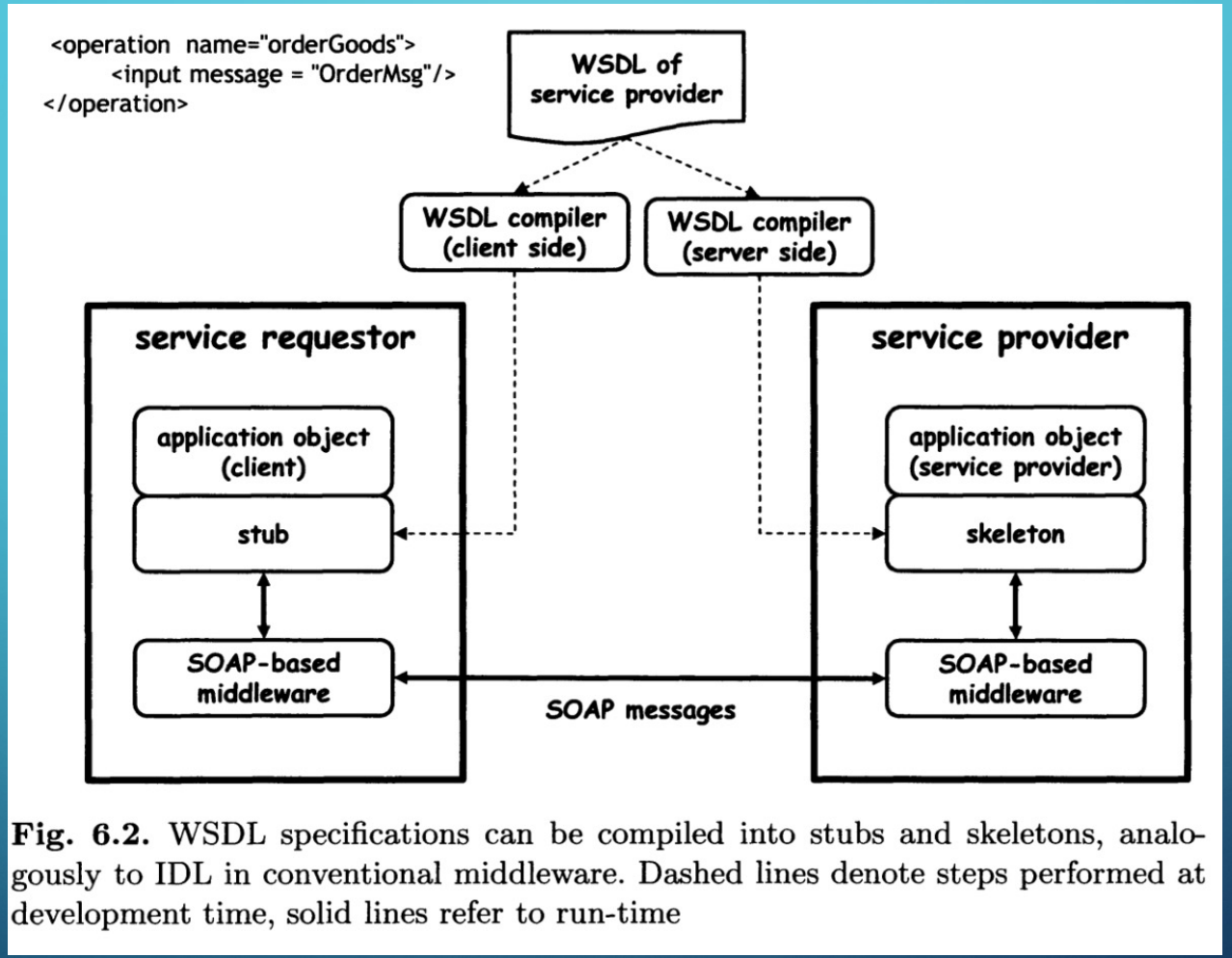


Fig. 6.2. WSDL specifications can be compiled into stubs and skeletons, analogously to IDL in conventional middleware. Dashed lines denote steps performed at development time, solid lines refer to run-time

https://link.springer.com/book/10.1007/978-3-662-10876-5

SOAP WEB SERVICES AND WSDL

An example of a WSDL document template

```
<definitions>
    <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>
</definitions>
```

THE FUTURE OF SOAP?

- Still used but…
  - Language support not always great
  - Hard to understand & hard to use in many cases
  - Allegedly complex because "*we want our tools to read it, not people*" – *unnamed Microsoft employee*
  - Heavyweight: XML + verbose messaging structure
- Dropped by Google APIs in 2006
- Still used in many places, including Microsoft APIs
- But we wanted something lighter and easier -- REST

https://people.cs.rutgers.edu/~pxk/417/notes/pdf/02d-web-services-slides.pdf

https://www.ibm.com/hk-en/cloud/learn/rest-apis
https://restfulapi.net
https://www.ruanyifeng.com/blog/2011/09/restful.html

RESTful API

- REST: **RE**presentational **S**tate **T**ransfer
- ***Resources*** as the abstraction of information, located by URI (Uniform Resource Identifier), specified by nouns
  - E.g. image is a type of resource, which can be represented as jpg, png, or jpg with additional metadata
  - E.g. http://api.example.com/device-management/managed-devices
  - E.g. http://api.example.com/device-management/managed-devices/{device-id}
- ***Resource representation***: the state of a resource at any particular instant, or timestamp
- ***State Transfer*** of the resources representation through resource methods
  - resource methods: PUT, GET, POST, DELETE (CRUD: Create, Read, Update, Delete)
- REST != HTTP
  - RESTful APIs communicate via HTTP

RESTful API

- Blog example
  - Get a user's blogroll – a list of blogs subscribed by a user

    HTTP GET http://myblogs.org/listsubs?user=paul
  - To get info about a specific blog (id = 12345):

    HTTP GET http://myblogs.org/bloginfo?id=12345

https://people.cs.rutgers.edu/~pxk/417/notes/pdf/02d-web-services-slides.pdf

PRACTICE

- JAX-WS and JAX-RS
  - Tutorial: https://javaee.github.io/tutorial/partwebsvcs.html
  - Code available at: https://github.com/javaee/tutorial-examples
- Environment setup:
  - Install Java EE 8 SDK:
    - https://javaee.github.io/tutorial/usingexamples001.html#GEXAJ
- Or, Jakarta RESTful Web Services (JAX-RS)
  - https://jakarta.ee/specifications/restful-ws/