Distributed Applications (18.06.2013)

Title: Distributed Applications (18.06.2013)
Date: Tue Jun 18 14:30:46 CEST 2013
Duration: 88:30 min
Pages: 32

Issues
The following section discusses several important basic issues of distributed applications.
- Data representation in heterogeneous environments.
- Discussion of an execution model for distributed applications.
- What is the appropriate error handling?
- What are the characteristics of distributed transactions?
- What are the basic aspects of group communication (e.g., algorithms used by ISIs)?
- How are messages propagated and delivered within a process group in order to maintain a consistent state?

External data representation

Time

Distributed execution model

Failure handling in distributed applications

Distributed transactions

Group communication

Distributed Consensus

Authentication service Kerberos

Authentication service Kerberos

Definition: Authentication means verifying the identities of the communicating partners to one another in a secure manner.

Kerberos has been developed at the MIT as part of the distributed framework Athena. Kerberos is part of a variety authentication components. The Kerberos authentication protocol is based on the protocol by Needham and Schröder.

Introduction

This course provides only a short introduction to Kerberos (for further information, consult the Kerberos Web Site)

Motivation

Security objects of Kerberos

Authentication process scenario

Kerberos assumes the following components

- Client C
- Server S
- Key distribution center KDC, and
- Ticket granting service TGS

Goal of Kerberos

A client C requests the service of the server S. KDC and TGS are supposed to guarantee the secrecy and authenticity requirements.

1. KDC manages the secret keys of the registered components.
2. Within a session TGS provides the client C with tickets for authentication with servers of the distributed system.
Graphical representation

Authentication process scenario

Graphical representation

Description of exchanged messages

Problems with Kerberos

Manipulation of local computer clocks to circumvent the validity time of tickets
i.e. synchronization of clocks in distributed systems must be authorized and authenticated.

Example: user login with Kerberos

Message 3: C to TGS

C → TGS with information \((C, T_C, K_{C,TGS})\)

\(\text{ticket}(C, TGS)_{k_{TGS}}\)

\(S\)

TGS determines a random session key \(K_{C,TGS}\), if

TGS ticket is still valid.

\(T_C\) is current, and

field \(C\) matches (of the first parameter and of the ticket).

Overview

Introduction

Architecture of distributed systems

Remote Invocation (RPC/RMI)

Basic mechanisms for distributed applications

Web Services

Design of distributed applications

Distributed file service

Distributed Shared Memory

Object-based Distributed Systems

Summary
Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the W3C community.

**Motivation - Example**

Web Services - Characteristics

Web Services Architecture

Simple Object Access Protocol (SOAP)

Web Services Description Language (WSDL)

Universal Description, Discovery, and Integration (UDDI)

REST

Web Service Composition

Adopting Web Services

Mashups

---

SOA vs. Component based Architecture

<table>
<thead>
<tr>
<th>component-based</th>
<th>SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>tight integration</td>
<td>loose horizontal integration</td>
</tr>
<tr>
<td>code-oriented development</td>
<td>process-oriented development</td>
</tr>
<tr>
<td>technical complexity of the IT infrastructure interoperable architecture for business and IT</td>
<td>build to last</td>
</tr>
<tr>
<td>build to last</td>
<td>build to change</td>
</tr>
</tbody>
</table>

---

SOA evolved from component-based architectures. SOA is a collection of services with a loose coupling and dynamic binding between services.

**Characteristics**

**Layered Approach**

Adopting Service Oriented Architecture (SOA)

SOA blueprints initiative: define the requirements for a reference example that highlights the best SOA practices. Web services are an approach of building a SOA based on Web technologies:

- encapsulation of application components in web services
The adoption within organizations depends on a variety of issues:

**Supporting Issues**
- interoperable networked applications
- easier exchange of distributed data
- easier access of enterprise wide data
- availability of external services
- cross-organizational computing
- reduced maintenance cost
- small effects on existing operational systems

**Restraining Issues**
- different formats and semantics of data sources
- security issues due to network access
- standards are evolving and some are not fixed
- lack of understanding

The **Enterprise Services Bus (ESB)** refers to both a software architecture and a class of software products used for the realization of SOA.

Messaging middleware that provides interoperability between enterprise applications via XML, Web Services interfaces and standardized rule-based routing of documents.

Mule is an Open Source ESB.

---

**Web Services - Characteristics**

A Web Service is a standardized way of integrating Web-based applications.

**Informal Definition**

Integration
- allows integration of application functionality within organizations
- between business partners across organizational boundaries

**Features of Web Services**

**Potential of Web Services**

**Web Services - Distributed Objects**
A Web Service is a standardized way of integrating Web-based applications.

**Informal Definition**

Integration
- allows integration of application functionality
- within organizations
- between business partners across organizational boundaries

**Features of Web Services**

**Potential of Web Services**

**Web Services - Distributed Objects**

---

**Features of Web Services**

specific features of Web Services
- programmable: WS are accessed via a programmable interface
- self descriptive: meta data describe the WS.
- encapsulation: self contained application component.
- loosely coupled: communication via message passing using platform-independent and language-neutral protocols.
- location transparent: access to WS from different locations via network communication.
- protocol transparent: WS is based on Internet protocol suite; operation may support several protocols, e.g. HTTP, SMTP.
- composition: several WS may be combined into a new WS.

Web Services are software components which enable loosely coupled, component-oriented, cross-technology application implementations.

Web Services are document-centric:
- communication is by sending documents from the server and back,
- most properties are associated with the document itself, and not the service.

---

**Web Services - Distributed Objects**

Web services and distributed objects
- have some sort of description language
- what to call: operations, signatures, return types, exceptions.
- how to make an invocation.
- compilers generate client stub and server skeleton
- both have well-defined network interactions
- both have a similar mechanism for registering and discovering available components.

**Differences**

Web services are usually designed for stateless computing.
- Distributed objects enable stateful computing.

Web services are a technology supporting the integration on the Web.
- Distributed objects are mainly for intranet.
**Definition:** A Web service (WS) is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols.

A Web Service is a standardized way of integrating Web-based applications using XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone.

- **XML:** tag the data
- **SOAP:** transfer the data
- **WSDL:** describe the available services
- **UDDI:** list the available services.

**simplified view:** a Web service is a remote procedure call over the Internet using XML messages.

**Web Services Interoperability Stack**

**Basic Architecture**

**Roles**

**Operations of the Web Service Architecture**

**Basic Standard Technologies**

**Message Exchange Patterns**

---

**Compositional**

- BPEL4WS, WS-Notification

**Quality of Experience**

- WS-Security, WS-Transactions, ...

**Description**

- WSDL, UDDI, WS-Policy, ...

**Messaging**

- XML, SOAP, WS-Addressing

**Transport**

- HTTP, SMTP, ...

---

**Basic Architecture**

defines an interaction between software components as an exchange of messages between service requesters and service providers.

**Functions of the architecture**

- exchanging messages,
- describing Web services,
- publishing and discovering Web service descriptions.

The service: a Web service is an interface; implementation of it is the service.

The service description: details of the interface and the implementation of the service.

---

**Roles**

The basic Web service architecture models the interactions between three roles:

- **Service Provider**
  processes a Web service request.

- **Service Discovery Agency**
  agency through which a Web service description is published and made discoverable.

- **Service Requestor**
  requests the execution of a Web service.
Publish: a service needs to publish its description such that a requestor can subsequently find it.

Find: the requestor queries a registry for the required service and retrieves a service description.

Interact: a service needs to be invoked and the results are returned.

Message Exchange Patterns

define the sequence of one or more messages exchanged between service requestor and service provider.

Examples are: one-way, request/response, broadcast.

The Web service architecture may support different interaction scenarios.

- Peer-to-Peer
- Direct Interaction
- Intermediary

Web Service Messages

WSIL uses XML to define messages.

<element name="CustomerInfoRequest">
  <element name="account" type="string"/>
</element>

<element name="CustomerInfoResponse">
  <element name="name" type="string"/>
  <element name="phone" type="string"/>
  <element name="street" type="string"/>
  <element name="city" type="string"/>
</element>
define the sequence of one or more messages exchanged between service requestor and service provider.

Examples are: one-way, request/response, broadcast.

The Web service architecture may support different interaction scenarios.

- **Peer-to-Peer**
- **Direct Interaction**
- **Intermediary**

In the peer-to-peer scenario, each Web service instance serves in both the service requestor and service provider roles.

The role service requestor and discovery agency are fulfilled by the client.

Intermediaries may perform additional functions (besides the operations defined by the message exchange patterns) with a message such as routing, security, management.
Web Services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the W3C community.

Motivation - Example
Service Oriented Architecture - SOA
Web Services - Characteristics
Web Services Architecture
Simple Object Access Protocol (SOAP)
Web Services Description Language (WSDL)
Universal Description, Discovery, and Integration (UDDI)
REST
Web Service Composition
Adopting Web Services
Mashups