Title: Distributed_Applications (07.05.2012)
Date: Mon May 07 09:16:45 CEST 2012
Duration: 43:46 min
Pages: 13

**Paradigms for distributed applications**

- Information Sharing
- Message exchange
- Naming entities
- Bidirectional communication
- Producer-consumer interaction
- Client-server model
- Peer-to-peer model
- Group model
- Taxonomy of communication
  - Message serialization
  - Levels of Abstraction

**Producer-consumer interaction**

In this interaction type (also called fire & forget interaction) after an invocation of the consumer, the producer resumes its execution immediately (and is not suspended).

Special case: Pipe mechanism (similar to Unix pipes); after information has been provided to the consumer, the producer terminates the execution.
The **client-server model** implements a sort of handshaking principle, i.e., a client invokes a server operation, suspends operation (in most of the implementations), and resumes work once the server has fulfilled the requested service.

![Diagram of client-server model]

**Service-oriented architecture (SOA)**: abstract architectural approach

- loose coupling and dynamic binding between services
- based on principles of modularized software and interface/component-based design

**Collection of services**

- services communicate with each other, e.g., data passing or remote invocation
- each service must manage its own data

SOA contains 3 roles: service requestor, service provider, and service registry.

Web services represent an implementation of SOA concept (currently the most important one)

**A central component (the server) provides a service to requesting clients.**

[Diagram showing a central server providing services to clients]

**Request-Answer Interaction**

**SOA**

**Examples for servers**

In a distributed environment, a server manages access to shared resources (e.g., a file server).

**Problems:**

- server crash → resource is no longer available in the network.
- server becomes a bottleneck for accessing the resource.

Internet Explorer, Netscape/Opera browser are examples for clients and Apache Web-Server is an example for a server.

**Web Server - HTTP**
Paradigms for distributed applications

Information Sharing
Message exchange
Naming entities
Bidirectional communication
Producer-consumer interaction
Client-server model
Peer-to-peer model
Group model
Taxonomy of communication
  Message serialization
Levels of Abstraction

Peer-to-peer model

All processes play a similar role
interacting cooperatively as peers to perform a distributed computation.
there is no distinction between clients and servers.
clients talk directly to one-another.

Client-Server
Servers are centrally maintained and administered
Client has fewer resources than a server

Peer-to-Peer (P2P)
A peer's resources are similar to the resources of the other participants,
peers communicate directly with other peers and share resources.

Issues of P2P
Peer discovery and group management
Data location and placement
Reliable and efficient file exchange
Security/privacy/anonymity/trust

Napster was one of the first P2P applications.
Napster server
publish list of music
Search music list
Home PC
Load music from source
Home PC

Technical issues
Many clients aren't accessible
many client systems come and go
Firewalls limit incoming connections to clients
round trip times to some regions are very slow
most clients had slow upload links
Clients might withdraw a file unexpectedly.

Legal issues
When service was launched, Napster designers hoped they had a way around the legal limits of sharing music:
clients advertise stuff
if some of that stuff happens to be music. That is the responsibility of the person who does it,
the directory server "helps clients to advertise stuff" but it does not endorse the sharing of protected
intellectual property.

Napster is making money by integrating Ads.

In the court case the judges saw it differently; Napster’s clear purpose is to facilitate theft of intellectual
property.

All processes play a similar role
interacting cooperatively as peers to perform a distributed computation.
there is no distinction between clients and servers.

clients talk directly to one-another.

---

**Client-Server vs. Peer-to-Peer**

**Napster**

**Gnutella**

**Other System Examples**