


## Script generated by TTT


Title: Distributed\_Applications (29.04.2014)

Date: Tue Apr 29 14:30:43 CEST 2014

Duration: 90:18 min

Pages: 21



Bidirectional communication 

---

Usage of the request-answer scheme for message exchange.

**Sockets**

**Call semantics**

Communication between sender and receiver is influenced by the following situations

- loss of request messages.
- loss of answer messages.
- sender crashes and is restarted.
- receiver crashes and is restarted.

[Different types of call semantics](#)

Generated by Targeteam



### Different types of call semantics



Any communication between a sender and a receiver is subject to communication failures. Therefore, we distinguish between different call semantics.

[at-least-once semantics](#)

[exactly-once semantics](#)

#### last semantics

Under a last semantics, the requested service operation is processed once or several times, however, only the last processing produces a result and, potentially, some side-effects.

#### at-most-once semantics

Under an at-most-once semantics, the requested service operation is processed once or not at all.

Example for providing at-most-once semantics

After timeout at the sending site the request is not retransmitted.

The request is transmitted in the context of a transaction.

Generated by Targeteam



### Different types of call semantics



Any communication between a sender and a receiver is subject to communication failures. Therefore, we distinguish between different call semantics.

[at-least-once semantics](#)

[exactly-once semantics](#)

#### last semantics

Under a last semantics, the requested service operation is processed once or several times, however, only the last processing produces a result and, potentially, some side-effects.

#### at-most-once semantics

Under an at-most-once semantics, the requested service operation is processed once or not at all.

Example for providing at-most-once semantics

After timeout at the sending site the request is not retransmitted.

The request is transmitted in the context of a transaction.

Generated by Targeteam



Usage of the request-answer scheme for message exchange.

Sockets

**Call semantics**

Communication between sender and receiver is influenced by the following situations

- loss of request messages.
- loss of answer messages.
- sender crashes and is restarted.
- receiver crashes and is restarted.

Different types of call semantics

Generated by Targeteam

Information Sharing

Message exchange

Naming entities

Bidirectional communication

Producer-consumer interaction

Client-server model

Peer-to-peer model

Group model

Publish-Subscribe model

**Taxonomy of communication**

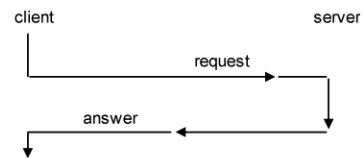
Message serialization

Levels of Abstraction

Generated by Targeteam



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.



Generated by Targeteam

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

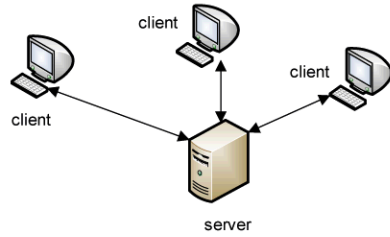
SOA contains 3 roles: service requestor, service provider and service registry.

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

Generated by Targeteam



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

*Generated by Targeteam*



Communication between Web Browser and Web Server is based on the HTTP protocol

stateless protocol.

based on TCP sockets using typically port 80.

session information is handled by the application layer (cookies).

HTTP protocol supports

the methods get, put, post, ..

return values / status code, such as

404: not found

401: unauthorized

400: bad request

*Generated by Targeteam*



Information Sharing

Message exchange

Naming entities

Bidirectional communication

Producer-consumer interaction

Client-server model

Peer-to-peer model

Group model

Publish-Subscribe model

**Taxonomy of communication**

Message serialization

Levels of Abstraction

*Generated by Targeteam*



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

*Generated by Targeteam*

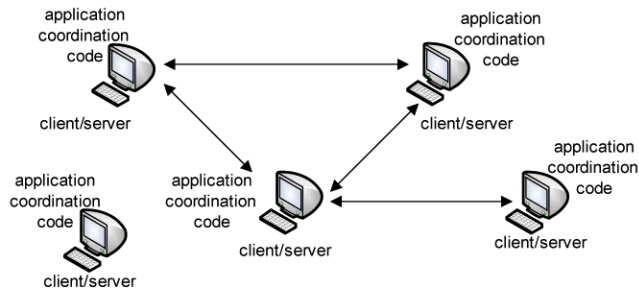


All processes play a similar role

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](#)

*Generated by Targeteam*



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.

*Generated by Targeteam*



Gnutella was one of the first examples of a pure P2P system

system is not run by a single company

no nodes which act only as servers; Gnutella eliminates the directory server.

for sharing files the user must connect to the Gnutella network, a loose federation of computers running Gnutella

for connection the computer only has to know the address of one other Gnutella machine, e.g. machines published at well known web sites.

at first connection the computer receives hundreds of addresses of machines which may be used at subsequent occasions.

a Gnutella program tries to maintain 3 or 4 connections to other Gnutella machines at any one time.

find a file: send request with file name and current hop count to its neighbors.

neighbor has matching file: respond with the location of the file

increment hop count;

if hop count < maximum hop count, then propagate request to its neighbors.

*Generated by Targeteam*

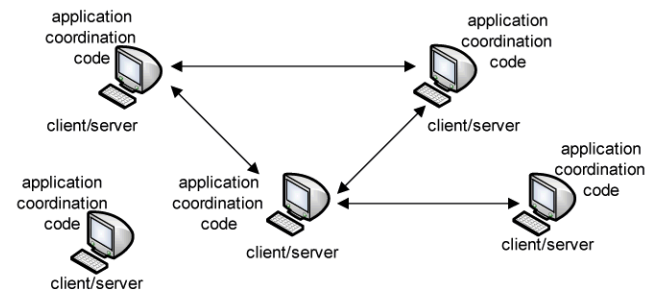


All processes play a similar role

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](#)

*Generated by Targeteam*



Propagate information in the same way as epidemic diseases spread.

approach explained informally

time  $t_0$  : suppose I know something new

time  $t_1$  : I pick a friend and tell him; now 2 people know.

time  $t_2$  : we each pick a friend and tell them; now 4 people know

time  $t_3$  : .....

information spreads at exponential rate.

due to re-infection information spreads at approx.  $1.8^k$  after k rounds.

combination of push and pull works best.

algorithm is quite robust and scalable

information travels on exponentially many paths.

difficult to slow down.

the load of the participating nodes is independent of the system size.

information spreads in  $\log(\text{system size})$  time.

network load is linear in system size.

Generated by Targeteam



Propagate information in the same way as epidemic diseases spread.

approach explained informally

time  $t_0$  : suppose I know something new

time  $t_1$  : I pick a friend and tell him; now 2 people know.

time  $t_2$  : we each pick a friend and tell them; now 4 people know

time  $t_3$  : .....

information spreads at exponential rate.

due to re-infection information spreads at approx.  $1.8^k$  after k rounds.

combination of push and pull works best.

algorithm is quite robust and scalable

information travels on exponentially many paths.

difficult to slow down.

the load of the participating nodes is independent of the system size.

information spreads in  $\log(\text{system size})$  time.

network load is linear in system size.

Generated by Targeteam



[Information Sharing](#)

[Message exchange](#)

[Naming entities](#)

[Bidirectional communication](#)

[Producer-consumer interaction](#)

[Client-server model](#)

[Peer-to-peer model](#)

[Group model](#)

[Publish-Subscribe model](#)

**Taxonomy of communication**

[Message serialization](#)

[Levels of Abstraction](#)

Generated by Targeteam



Publish-subscribe systems is most widely used of all indirect communication techniques.

publishers publish structured events to an event service

subscribers express interest in particular events through subscriptions

task of publish-subscribe system:

match subscriptions against published events and ensure correct delivery of event notification.

Characteristics:

support of heterogeneous environments.

notifications are sent asynchronously.

Generated by Targeteam



certain order for message delivery

messages to a group of recipients: messages arrive in different order, due to different transmission times.

### One sender

There are the following ordering schemes:

according to the message arrival on the recipient's side; different receivers can have different message arrival sequences.

according to message sequence number generated by the sender; this approach is sender-dominated.

receiver creates a serialization according its own criteria.

### [Several senders](#)



*Generated by Targeteam*

If several senders are involved, the following message ordering schemes may be applied:

1. no serialization.

2. loosely-synchronous.

There is a loosely synchronized global time which provides a consistent time ordering.

3. virtually-synchronous.

The message order is determined by [causal interdependencies](#) among the messages. For example, a message N has been sent after another message M has been received, i.e. N is potentially dependent on M.

4. totally ordered.

by token: before a sender can send a message, it must request the send token.

a selected component (the coordinator) determines the order of message delivery for all recipients.

*Generated by Targeteam*