Algorithms and Programming IV
Introducing the Concepts of Distributed Programming

Summer Term 2021 | 07.06.2021
Barry Linnert
Our Topics

1 Defining Distributed Systems

2 Possible Application Domains

3 Our Distributed System Model

4 An Architectural Model of Distributed Systems
   - Communicating entities
   - Communication paradigm
   - Roles and responsibilities
   - Placement
Characteristics of a Centralized System Model

- One component with non-autonomous parts
- Component shared by users all the time
- Homogenous architecture – all execution units are from the same kind
- All resources accessible
- A running application is based on a single program

- Single point of control
- Single point of failure
Exchange Information between CERN-Laboratories

- In 1989, Tim Berners-Lee proposed a project to address this problem. The underlying idea was based on the hypertext to create a world-wide scientific network for information sharing.

- Tim Berners Lee developed the following concepts: Hyperlinks, URL, HTTP, HTML, the Webbrowser WorldWideWeb and the Webserver NeXTSTEP.

- Christmas 1990: the first website info.cern.ch.
Distributed System Model

Distributed System

- Multiple autonomous components
- Components are not shared by all users
- Resources may not be accessible
- Software runs in concurrent processes on different processors
- Multiple points of control
- Multiple points of failure
What is a Distributed System?

„A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.“

Leslie Lamport

Email message sent to a DEC SRC bulletin board at 12:23:29 PDT on 28 May 87
Let’s have a bit more detail…

“A distributed system consists of a *collection of autonomous computer* linked by a *computer network* and equipped with distributed system software. Distributed system *software enables* computers *to coordinate their activities* and *to share the resources* of the system – hardware, software, and data [...] so that users perceive the system as a single, integrated computing facility.”

(Coulouris et al., 1994)
Distributed System Model
## Possible Application Domains

<table>
<thead>
<tr>
<th>Domain</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance and commerce</td>
<td>eCommerce Application, e.g. Amazon and eBay, PayPal, and online banking and trading</td>
</tr>
<tr>
<td>Information Systems &amp; Social Media</td>
<td>Web information and search engines, ebooks, Wikipedia; social networking: Facebook, Twitter.</td>
</tr>
<tr>
<td>Entertainment</td>
<td>Online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr</td>
</tr>
<tr>
<td>Healthcare</td>
<td>Health informatics, online patient records, monitoring patients</td>
</tr>
<tr>
<td>Education</td>
<td>E-learning, virtual learning environments; distance learning</td>
</tr>
<tr>
<td>Transport and logistics</td>
<td>GPS in route finding systems, map services: Google Maps, Google Earth</td>
</tr>
<tr>
<td>Science</td>
<td>Grid as an enabling technology for collaboration between scientists</td>
</tr>
<tr>
<td>Environmental management</td>
<td>Sensor technology to monitor earthquakes, floods or tsunamis</td>
</tr>
</tbody>
</table>
Defining Computer Supported Cooperative Work (CSCW)

• CSCW describes the cooperation in teams with the help of groupware to fulfill a shared task. Groupware especially supports the communication, the coordination, the making of group decisions, and the joint processing of information objects.

• Groupware is computer-software and related computer networks that enable collections of people to cooperate distributed.

## A two-dimensional Collaboration Framework

<table>
<thead>
<tr>
<th></th>
<th>Real time</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Communication</strong></td>
<td>• Telephone</td>
<td>• Email</td>
</tr>
<tr>
<td></td>
<td>• Video conferencing</td>
<td>• Voice mail</td>
</tr>
<tr>
<td></td>
<td>• Instant messaging</td>
<td>• Blogs</td>
</tr>
<tr>
<td></td>
<td>• Texting</td>
<td>• Social networking sites</td>
</tr>
<tr>
<td><strong>Information sharing</strong></td>
<td>• Whiteboards</td>
<td>• Document repositories</td>
</tr>
<tr>
<td></td>
<td>• Application sharing</td>
<td>• Wikis</td>
</tr>
<tr>
<td></td>
<td>• Meeting facilitation</td>
<td>• Web sites</td>
</tr>
<tr>
<td></td>
<td>• Virtual worlds</td>
<td>• Team workspaces</td>
</tr>
<tr>
<td><strong>Coordination</strong></td>
<td>• Floor control</td>
<td>• Workflow management</td>
</tr>
<tr>
<td></td>
<td>• Session management</td>
<td>• Project management</td>
</tr>
<tr>
<td></td>
<td>• Location tracking</td>
<td>• Calendar scheduling</td>
</tr>
</tbody>
</table>

Using People Aktivities to feed Systems

Social Computing

• has to do with digital systems that support online social interaction.
• is concerned with **how** digital systems support social interaction.

\[ PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)} \]
AN ARCHITECTURAL MODEL OF DISTRIBUTED SYSTEMS
Architectural Model

An architectural model of a distributed system simplifies and abstracts the functions of the individual components of a distributed system.

It deals with the

• organization of components across the network of computers, and
• their interrelationship, i.e., how these components communicate with each other.
An Architectural Model of Distributed Systems

<table>
<thead>
<tr>
<th>Architectural elements</th>
<th>Communicating entities</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
</table>

An Architectural Model of Distributed Systems

### Architectural elements

<table>
<thead>
<tr>
<th>Communicating entities</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
</table>

What are the **entities** that are communicating in the distributed system?
System-oriented Perspective

In distributed systems the entities that communicate are typically processes.

Exceptions:

• In primitive environments such as sensor networks, there are no operating systems that provide any abstractions, therefore nodes communicate directly.
• In most environments processes are supplemented by threads, so threads are the endpoints of communications.
Problem-oriented perspective

Objects
- Computation consists of a number of interacting objects representing units of decomposition for a problem domain
- Objects are accessed via interfaces

Components
- Resemble objects in that they offer problem-oriented abstractions, also accessed via interfaces
- Specify not only their interfaces but also the assumptions they make in terms of other components/interfaces that must be present for a component to fulfil its function

Web services
- A software application which is identified via Uniform Resource Identifier (URI)
- Supports direct interactions with other software agents
An Architectural Model of Distributed Systems

<table>
<thead>
<tr>
<th>Architectural elements</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicating entities</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objects</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Components</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Architectural Model of Distributed Systems

<table>
<thead>
<tr>
<th>Architectural elements</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicating entities</td>
<td></td>
<td>What (potentially changing) roles and responsibilities do they have in the overall architecture?</td>
<td></td>
</tr>
</tbody>
</table>
Architectural Styles

- client-server
- peer-to-peer
Client-Server

Key:
Process: [Circle] 
Computer: [Rectangle]
Fundamental Issue with Client-Server

• Client-server offers a direct, relatively simple approach to the sharing of data and other resources, but it scales poorly.

• The centralization of service provision and management implied by placing a service at a single address does not scale well beyond the capacity of the computer that hosts the service and the bandwidth of its connections.

• Even though, there a several variations of the client-server architecture to respond to this problem but none of them really solve it.
Peer-to-Peer

- Is composed of a large number of peer processes running on separate computers
- All processes have client and server roles: servant
- Patterns of communication between them depends entirely on application requirements
- Need to place and retrieve individual computers is more complex than in client-server architecture
Example: Corona Tracing App

Source: https://www.suedkurier.de/baden-wuerttemberg/warum-sind-die-schweizer-so-schnell-corona-tracing-app-der-eidgenossen-soll-bald-verfuegbar-sein;art417930,10518170
Example: Corona Tracing App (Central)

Source: https://secpriv.wien/blog/de/tracing-app/overview/
Example: Corona Tracing App (Decentral)

Source: https://secpriv.wien/blog/de/tracing-app/overview/
### An Architectural Model of Distributed Systems

<table>
<thead>
<tr>
<th>Architectural elements</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicating entities</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How are they mapped on the physical distributed infrastructure (what is their **placement**)?
Services provided by multiple servers

Option 1
• Servers partition a set of objects in which the service is based and distribute them between themselves
• Example
  – In the Web in which each webserver manages its own set of resources
  – User can employ a browser to access a resource at any one of the servers

Option 2
• Server maintain replicated copies of them on several hosts
• Example:
  – Network Information Service (NIS) used by computers on a LAN
Proxy server and caches

A cache is a store of recently used data objects that is closer to the objects themselves. Caches might be co-located with each client or may be located in a proxy server that can be shared by several clients.

How does it work?

• If a new object is received at a computer, it is added to the cache store, replacing some existing objects if necessary.
• If an object is requested by a client process, the caching service checks the cache for an up-to-date copy.
• If copy is not available this copy is fetched.
Mobile code/agents

A typical well-known and widely-used example for mobile code are applets.
• Example: Javascript-widgets

A mobile agent is a running program (both code and data) that travels from one computer to another in a network carrying out a task on someone’s behalf, e.g. collecting information.
• Example: web crawler
An Architectural Model of Distributed Systems

<table>
<thead>
<tr>
<th>Architectural elements</th>
<th>Communication paradigm</th>
<th>Roles and responsibilities</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicating entities</td>
<td>Communication paradigm</td>
<td>Roles and responsibilities</td>
<td>Placement</td>
</tr>
</tbody>
</table>

How do they **communicate**, or, more specifically, what communication paradigm is used?
Types of Communication Paradigms

- Interprocess communication
- Remote invocation
- Indirect communication
Summary

Architectural elements

- Communicating entities
  - Processes
  - Objects
  - Components
  - Web Services

- Communication paradigm
  - Inter-process communication
    - UDP sockets
    - TCP sockets
    - Multicast
  - Indirect communication
  - Remote invocation

- Roles and responsibilities
  - Architectural styles
    - Client-server
    - Peer-to-peer
  - Mobile code

- Placement
  - Multiple server
  - Proxy/Cache
  - Mobile code
References


Algorithms and Programming IV
Communication Paradigms in Distributed Systems

Summer Term 2021 | 11.06.2021
Claudia Müller-Birn, Barry Linnert