Distributed Systems

Seminar 3 – Remote Procedure Call

Dr. Thomas Springer
What is the definition of an RPC according to Nelson? Distinguish, on the basis of this definition, RPC from the following:

- A local procedure call,
- Communication via email,
- Communication in a shared memory multiprocessor system.
Definition (by Nelson)

- Synchronous transfer of control thread
- Level of programming language
- Separate address spaces
- Coupling via relatively narrow channels
- Data exchange: call parameters and results

Synchronous transfer control thread and data by means of a procedure call with parameters between programs in separated address spaces via a narrow channel

Narrow channel: low bandwidth and/or high delay (relative to local communication)
<table>
<thead>
<tr>
<th><strong>RPC defined by Nelson</strong></th>
<th><strong>Local procedure call</strong></th>
<th><strong>Email</strong></th>
<th><strong>Communication in a shared memory multiprocessor system</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous transfer of control thread</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Level of programming language</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Separate address spaces</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Coupling via relatively narrow channels</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Data exchange: call parameters and results</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
The RPC is an essential communication mechanism in distributed systems.

- Discuss the basic steps of an RPC!
- What is the meaning of the terms “marshalling” and “unmarshalling”?
- Explain the connection between RPC interface descriptions and stub components!
Remote Procedure Call

Client-computer

- Local call
- Local result

Client-Stub
- Call encoding

Runtime-system
- Send
- Wait
- Receive

Network
- Call
- Packet
- Result
- Packet
- Import

Server-computer

Runtime-system
- Receive

Server-Stub
- Decoding

Server
- Call
- Execute
- Result

Export
The RPC is an essential communication mechanism in distributed systems.

What is the meaning of the terms “marshalling” and “unmarshalling”?

Marshalling is:
- Transforming of a typed and possibly structured parameter set
- from internal/sender-specific representation (byte order, structure representation)
- into a serialized format
- which allows to transfer the data via a network as message

Unmarshalling is:
- Reconstruction of typed and possibly structured data
- From data serialized in messages
- Into a receiver-specific data representation
- To allow local processing of data at receiver

Why necessary?
- Serialization of structured data (local pointers not valid on server machine)
- Heterogeneity of internal data representations (8 Bit/16 Bit for int, Big/Little Endian, ...)

Dr. Thomas Springer
- Interfaces used by compiler to generate client and server stubs
- Stubs encapsulate functionality to:
  - transmit local call to remote computer
  - receive call on remote computer and pass on
  - convert data
    - before transmission on client (marshalling)
    - on arrival at remote computer (unmarshalling)
Binding is an important prerequisite for establishing an RPC-based communication.

- What is the purpose of binding?
- Sketch the binding process for the methods direct addressing, broadcast-request and directory service.
- Discuss the advantages and disadvantages of the various methods of binding.
- Binding is the process of coupling a client and a server by resolving the address of the server at client side.
- Maybe further exchange/negotiation of communication parameters.
- Address is e.g. IP address + Port number or Or Service endpoint (URI).
- **Direct addressing, e.g. IP address configured directly**
  - As variable in code, in a configuration file, etc.
- **Broadcast request, e.g. using a logic name (sub-net)**
  - Discover servers in a local network or flush the internet
- **Directory service**
  - Server registers with logic name and address (and attributes) at directory service
  - Client requests server with logic name (and attributes)
  - Client gets address of server from directory service
  - Client can call procedures using address of server
Solution E3.3c

Direct Addressing
+ Fast binding process without network communication and service
+ High scalability for search
  - Static configuration – No rebinding at runtime
    - Binding information is invalid if address of server changes
    - Changes of binding information might require recompilation

Broadcast
+ Flexible binding and rebinding based on logic name possible
+ No service for binding required, direct comm. with potential services
+ Works well in intranet/LAN
  - Broadcast might not be supported in network
  - High overhead in large network infrastructures – flooding of network with search requests
    - high network load

Naming/Directory Service
+ Flexible binding and rebinding based on logic name possible
+ Highly scalable with distributed binding service
  - Binding process relatively complex - Network communication and binding service required
RPC systems have to handle different types of errors of the remote communication. The RPC error semantics defines what classes of errors can be handled.

- Which error classes are defined, which errors types can be handled and what mechanisms have to be used for handling?
- What error class should be used for the following functionality of the online-shop:
  - User access of product catalog via Browser and HTTP
  - The submission of a product order
  - Money transfer for order payment
  - User request to order state
  - Removing a product from warehouse and adding it to a dispatch list
Solution E3.4a

- **Message delay or loss (request/response)**
  - **Request gets lost**
    - Server is not aware about request
    - no response, Client is blocked (forever or until a timeout)
  - **Response gets lost**
    - Server performs request
    - sends response
    - Client is blocked (forever or until a timeout)

- **Client crash**
  - after request
    - Server performs request
    - Client do not get response

- **Server crash**
  - after request
    - Server is not aware about request
    - no response, Client is blocked (forever or until a timeout)
- **Error semantics (Spector):**
  - **Maybe**
    - single execution without notification in the case of errors
    - only for “non-important” operations
  - **At least once**
    - at least once execution (if no machine crashes happen)
    - only for idempotent operations
    - Repeated request in case of request/response lost after timeout
  - **At most once**
    - duplicate recognition (sequence number) and removing;
      - masks comm. failures
    - one execution if no machine crashes happen
  - **Exactly once**
    - exactly once execution
    - masks machine crashes, too
      - transaction concepts with warm restart and recovery
User access of product catalog via Browser and HTTP
  • **Maybe, At-least-once**
  • Read access, user wants result but reading content could be performed several times without change of system data

The submission of a product order
  • **At-most-once**
  • Order should not be performed twice

Money transfer for order payment
  • **Exactly-once**
  • Transactional to ensure consistent accounts

User request to order state
  • **Maybe/at-least-once**
  • State can be easily requested again but user may expect answer

Removing a product from warehouse and adding it to a dispatch list
  • **Exactly-once**
  • Transactional to ensure consistent state of warehouse and dispatch list