There may exist multiple redundant services; server copies and client copies are grouped together into server and client groups.

**Modular redundancy**

**Primary-standby-approach**

At any specific time, there is only one replica acting as master (primary replica); RPC requests are always propagated to the primary replica; at checkpoints the current state is propagated to the secondary replicas. In case of an error the master is replaced by a backup replica.

**Distinction between hot and cold standby.**
There may exist multiple redundant services; server copies and client copies are grouped together into server and client groups.

**Modular redundancy**

**Primary-standby approach**

---

**Definition:** Birrell and Nelson (1982) define an RPC as a synchronous flow of control and data passing scheme achieved through procedure calls between processes running in separate address spaces where the needed communication is via small channels (with respect to bandwidth and duration time).

- **synchronous**: The calling process (client) is blocked until it receives the answer of the called procedure (server); the answer contains the results of the processed request.
- **procedure calls**: the format of an RPC call is defined by the signature of the called procedure.
- **different address spaces**: it is necessary to handle pointers during parameter passing different from local procedure calls.
- **small channel**: reduced bandwidth for communication between involved computers.

---

Neither the client nor the server assume that the procedure call is performed over a network.

**Control flow for RPC calls**

1. **bind to server**
2. **RPC-request**
3. **RPC-response**
4. **unpack reply**
5. **time**

---

**Differences between RPC and local procedure call**

- **Basic RPC characteristics**
- **RPC and OSI**
- **RPC vs message exchange**
For an RPC, the caller and the callee run in different processes. Both processes (caller and callee) have:
- no shared address space,
- no common runtime environment,
- different life spans of client and server.

Handle errors occurring during a RPC call, e.g. caused by machine crashes or communication failures.
RPC-based applications must take communication failures into consideration.

<table>
<thead>
<tr>
<th>Integration of the RPC into ISO/OSI protocol stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>layer 7</strong> application layer</td>
</tr>
<tr>
<td><strong>layer 6</strong> presentation layer</td>
</tr>
<tr>
<td><strong>layer 5</strong> session layer</td>
</tr>
<tr>
<td><strong>layer 4</strong> transport layer</td>
</tr>
</tbody>
</table>

- transport protocols: UDP (User Datagram Protocol) transports data packets without guarantees; TCP (Transmission Control Protocol) verifies correct delivery of data streams.
- message exchange: socket interface to the underlying communication protocols.

**RPC** hides communication details behind a procedure call and helps bridge heterogeneous platforms.