Abdullah
1802016

-Remote procedure calls – Question No. 0-

- **Call Semantics:**

  In RPC, the caller and callee processes resides on different systems, so their normal functioning may get disturbed due reasons mentioned below:

  1. Difference in data-representation on the client and server machines i.e. some systems uses big-endian and other uses little-endian (store the least significant byte at high memory address).
  2. RPC can fail, or b duplicated and executed more than once.
  3. In standard procedure calls, binding takes place during link, load and execution time to replace procedure name with memory address of that procedure. RPC requires similar method to bind client and server port but how client will know the port no. on the server. It doesn't have any information as both are different systems and don't share memory.

-It matters due to following-

- **Difference in Data-representation:**

  For solving such kind of problem, RPC system defines a machine-independent representation of data. Such representation is known as external data representation (XDR).

  - **XDR:** It is standard data serialization format, for uses such as computer network protocols. Portable between different operating systems and independent of transport layer.

  On client side, parameter marshalling involves converting the machine dependent data into XDR before they are sent to the server.

  On server side, the XDR data are unmarshalled and converted to the machine-dependent representation for the server.

- **Extreme circumstances (Duplication):**

  One solution is that messages are acted on exactly-once rather than at-most once but it is difficult to implement exactly-once functionality.

  - **At most once:** This is done by attaching a timestamp to each process. Server keeps history of timestamp of messages which are already processed or detect repeating processes. Repeating processes are ignored by which client can sent one or more messages but it is assured that they will execute once at server side.

- **Exactly once:** For this implementation, we need to remove the risk that the server will never receive the request but for this, server need to implement “At most once” functionality mentioned above but must also admit to the client that the RPC call was received and executed. The client must resend each RPC call periodically until it receives the admit for that call.

- **Binding:**

For this, there are two methods.

- **First**, the binding information may be predetermined, in the form of fixed port addresses. At compile time, an RPC call has a fixed port number associated with it. When a program is compiled, the server cannot change the port number of the requested service.

- **Second**, binding can be done dynamically by a rendezvous mechanism. Operating system provides a rendezvous (matchmaker) daemon on a fixed RPC port. A client then sends a message containing the name of the RPC to the rendezvous daemon requesting the port address of the RPC it needs to execute. The port number is returned, and the RPC calls can be sent to that port until the process terminates (or the server crashes). This method requires the extra overhead of the initial request but is more flexible than the first approach.

Abdullah
1802016