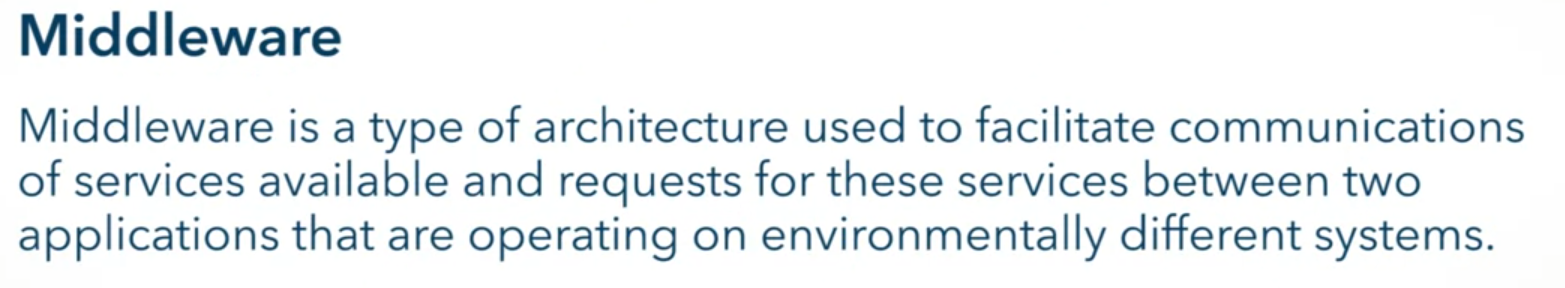
Clients

* Designed for a single user
* For exceptional usability

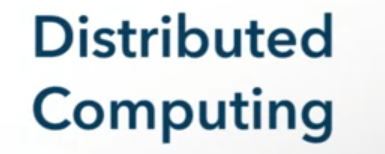
Servers

* Will have to provide computing power for multiple concurrent users

How they communicate?

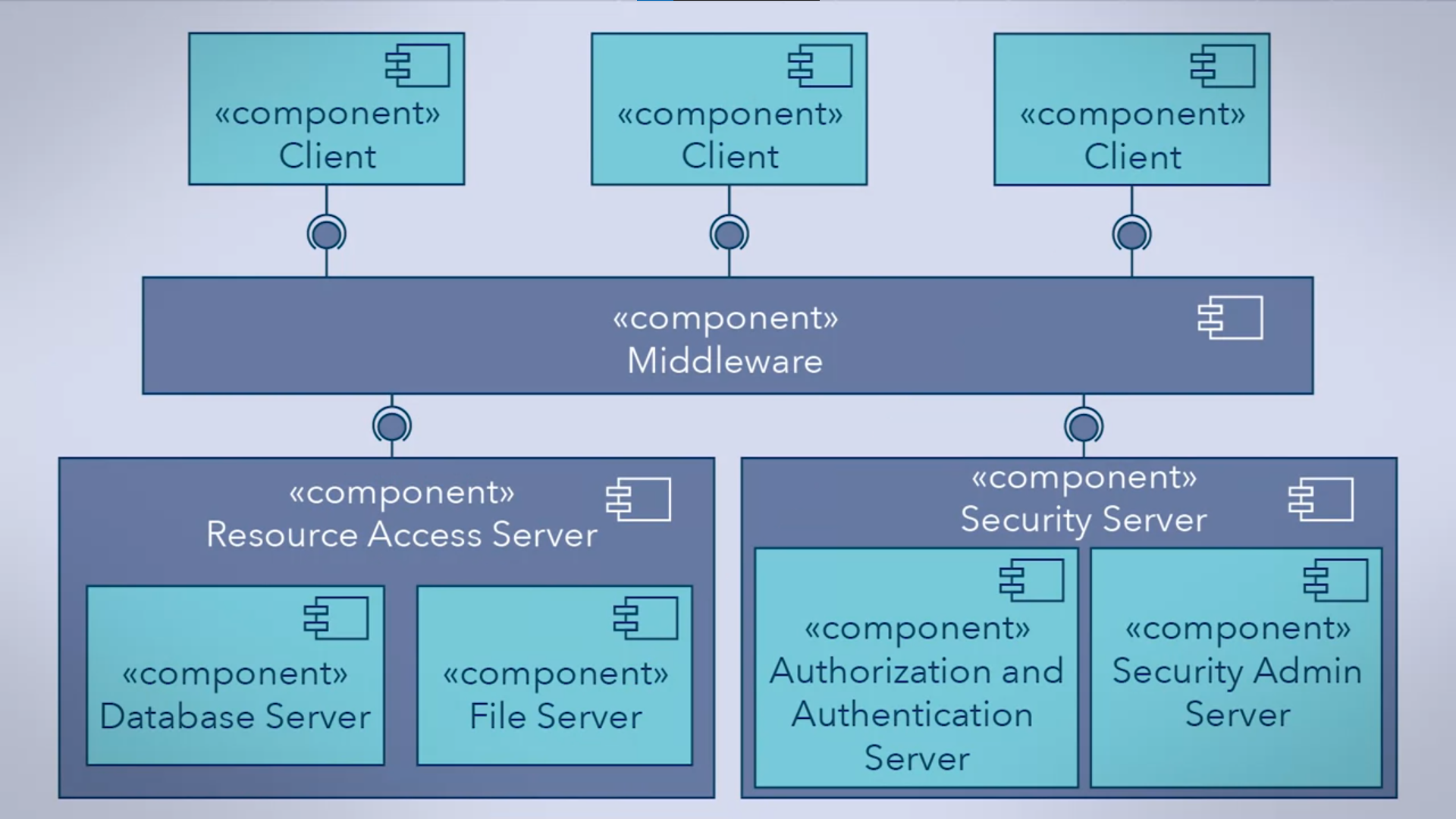


Network connectivity are now expected





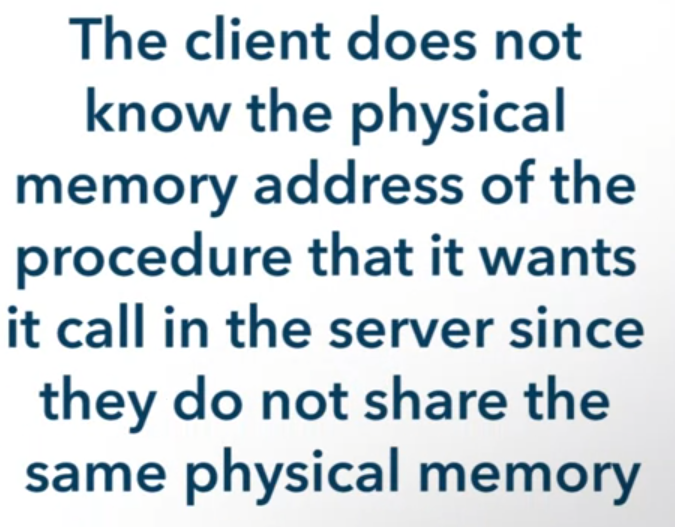




Remote Procedure Call (RPC)

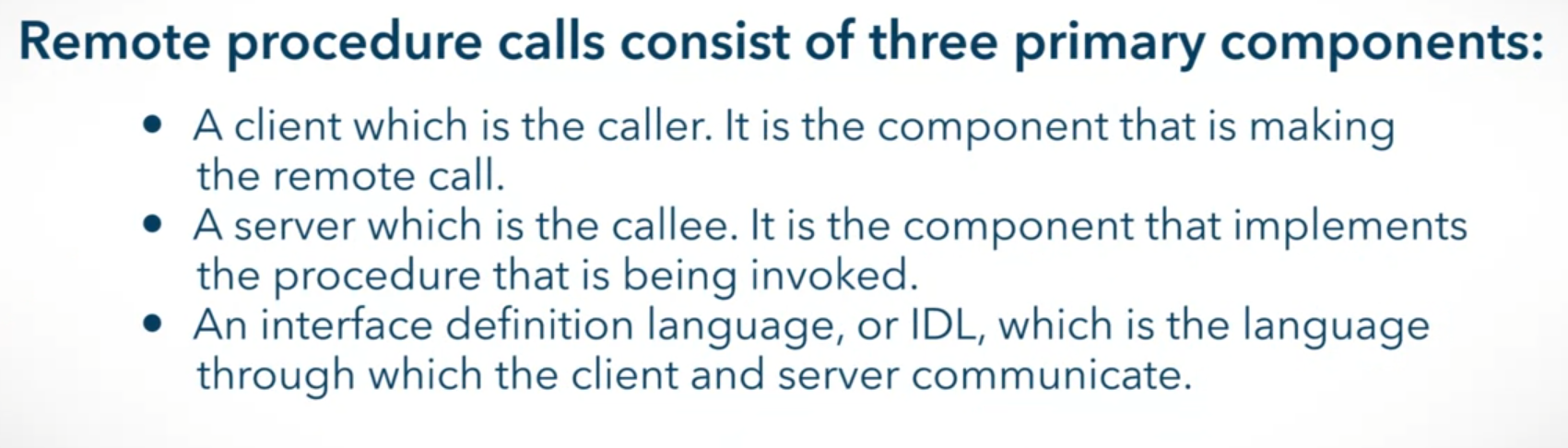
* Middleware component for network systems
* Facilitate calls between a client and server

Client and Server are either on:

* Completely separate machines
  + 
* Different virtual instances on the same machine
  + 

History of RPC

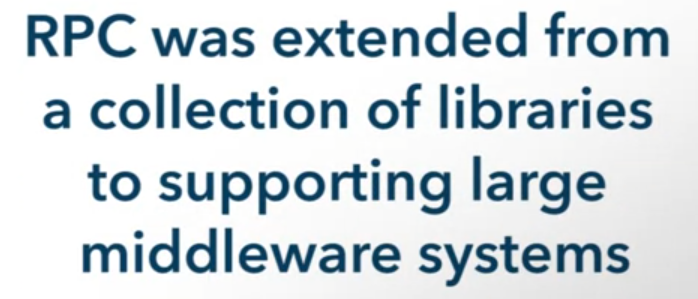
* Developed by Birell and Nielson
* 1980s



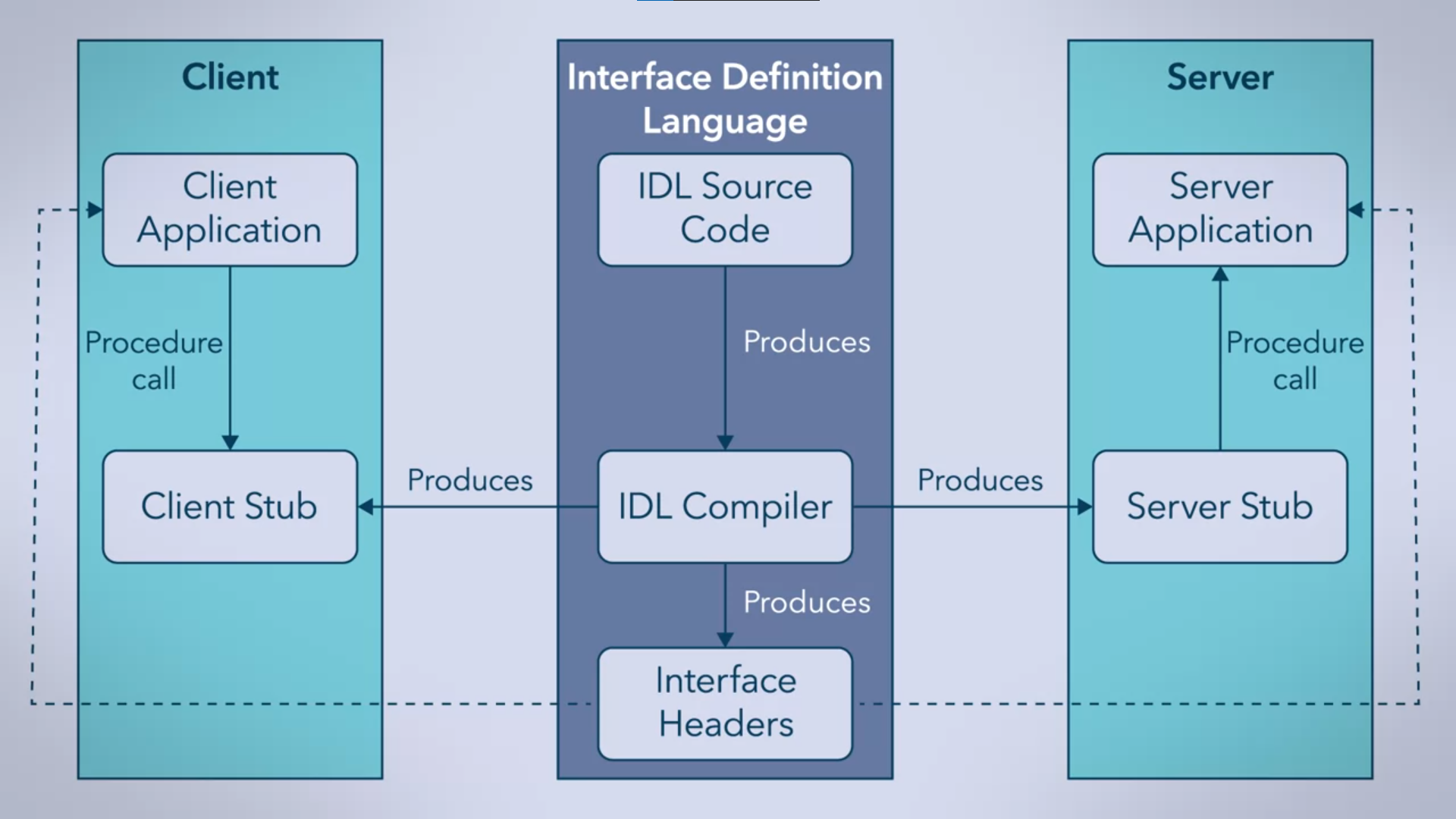


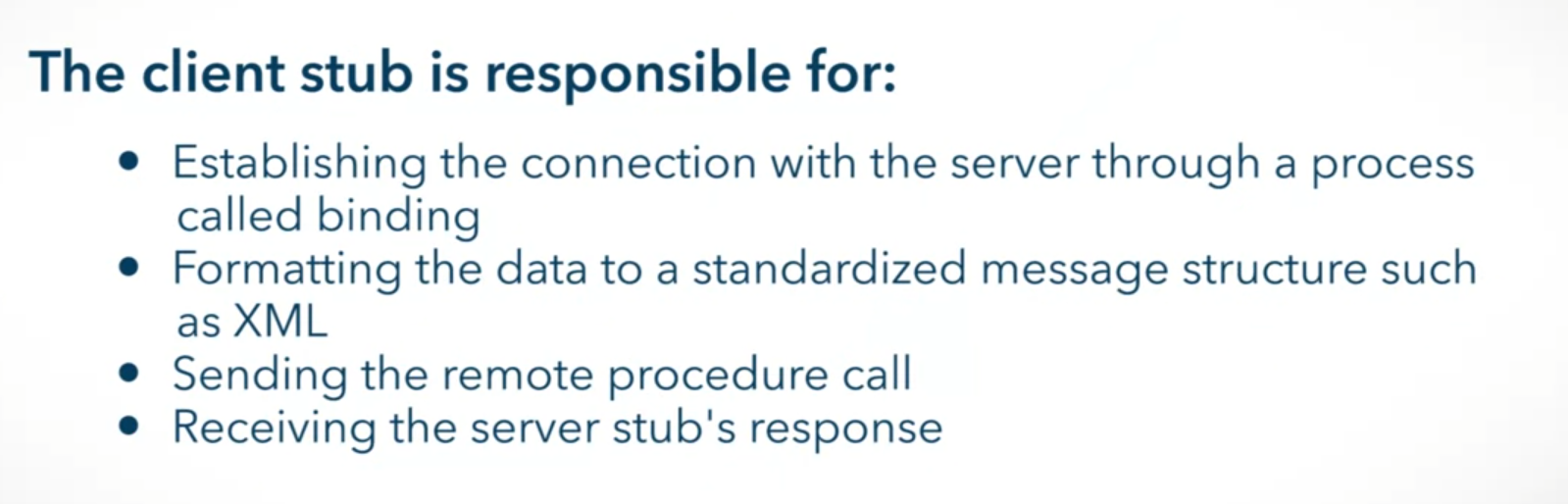
In today’s system

* Stored procedures
* XML

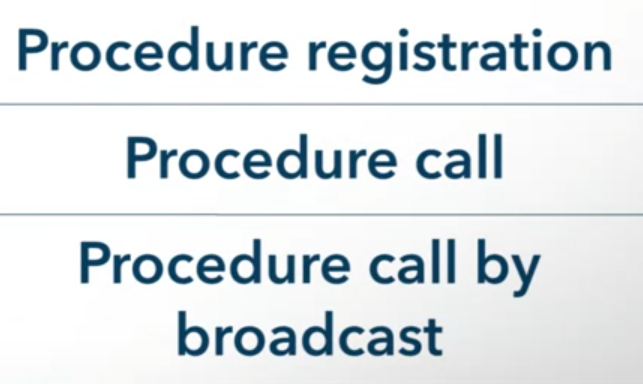


How does an RPC work?



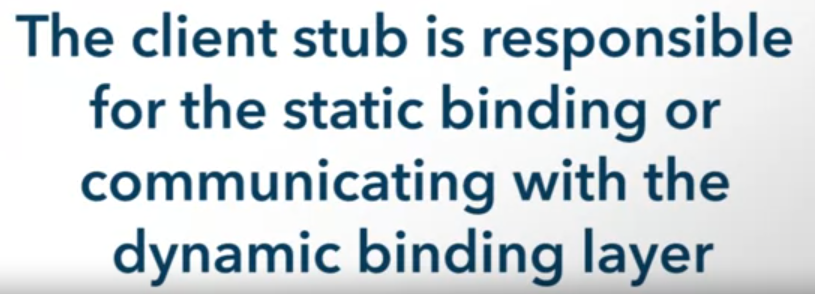


Samples of Interface Headers



How RPC are performed?

1. Client component invokes the procedure
2. Client stub marshalls the parameters
3. The message is sent to the server
   1. Static Binding – hard-coded hostname…
   2. Dynamic Binding
      * Adds a name and directory server tier
        + Keeping track of which servers are bound
        + Balancing loads
        + Change information of servers if servers are change



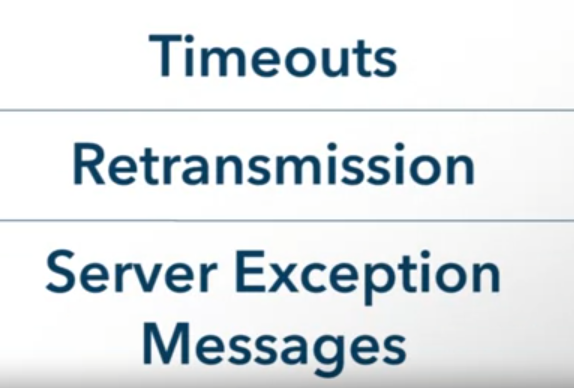
1. The server stub receives and unmarshalls the message
2. The server component executes the procedure and return the result to the server stub (marshall again to desired format)
3. The results are returned back to the client stub
4. The client component receives and processes (unmarshalls) the results

RPC

* Originally synchronous (blocking)

Issues in being synchronous

* What if the server never returns a response?
* How long do we wait for the server to respond?
* Do we re-transmit a remote procedure call?



Modern RPCs

* Asynchronous (non-blocking)

