Birrell84

Important points:

- *Language agnosticism:* The RPC framework decouples user/server stubs, which could be easily generated on a per-language basis provided a foreign function interface, from the RPC runtime environment. In other words, one runtime fits all!
- *Communication protocol:* communication between the user- and server-facing RPC runtimes is facilitated through a custom protocol to minimize the volume of packet traffic (acks, data, etc.) vs. e.g. TCP (perhaps a manifestation of the end-to-end argument)
- *Multiprogramming runtime*: the framework is further optimized by keeping a process pool to handle communication between instances of the runtime

Flaw: I wish they had elaborated more on end-to-end encryption of call data as opposed to the minutiae of optimization.

Conclusion: the success of this framework from a design point of view is that it manages to make the act of a remote procedure call as seamless to user- and server-facing code as much as possible via a modular design of the runtime environment as well as optimizing communication to make calls as low-cost as possible. Many modern systems fail to achieve such a low-cost abstraction.