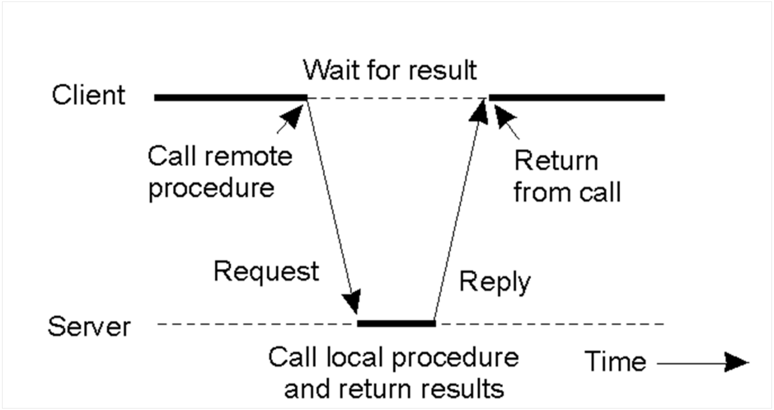
(g)RPC - Remote Procedure Call

February 13, 2019

Remote Procedure Call (RPC)



- a form of inter-process communication

RPC - Protocols

- XML-RPC
  - **xmlrpc2** (CRAN), **XMLRPC** (omegahat)
- JSON-RPC
  - https://www.jsonrpc.org/
- gRPC
  - open source remote procedure call (RPC) system initially developed at Google
- ...

XML-RPC Example - 1

- Used by the NEOS-Server (**ROI.plugin.neos**, **rneos**)

NEOS-Example

```
(body <- xmlrpc2::to_xmlrpc(method = "ping", params = list()))

## {xml_document}
## <methodCall>
## [1] <methodName>ping</methodName>
## [2] <params/>

handle <- curl::new_handle()
curl::handle_setopt(handle, port = 3333)
curl::handle_setopt(handle, customrequest = "POST")
curl::handle_setopt(handle, followlocation = TRUE)
curl::handle_setopt(handle, postfields = as.character(body))
curl::handle_setheaders(handle, `Content-Type` = "text/xml",
  `User-Agent` = "xmlrpc")
response <- curl::curl_fetch_memory("https://www.neos-server.org", handle)
xmlrpc2::from_xmlrpc(rawToChar(response$content))

## [1] "NeosServer is alive\n"
```

XML-RPC Example - 2

- Used by the NEOS-Server (**ROI.plugin.neos**, **rneos**)

NEOS-Example

```
nurl <- "https://www.neos-server.org"
xmlrpc2::xmlrpc(nurl, "ping")

## [1] "NeosServer is alive\n"

xmlrpc2::xmlrpc(nurl, "version")

## [1] "neos version 5 (Madison)"

tail(unlist(xmlrpc2::xmlrpc(nurl, "listAllSolvers")), 14)

##  [1] "kestrel:GAMS-AMPL:GAMS"   "nco:ANTIGONE:GAMS"
##  [3] "go:ANTIGONE:GAMS"         "mpec:Knitro:AMPL"
##  [5] "lp:FICO-Xpress:NL"        "milp:FICO-Xpress:NL"
##  [7] "socp:FICO-Xpress:NL"      "nco:Ipopt:NL"
##  [9] "lp:MOSEK:NL"              "milp:MOSEK:NL"
## [11] "sdp:mosek:SPARSE_SDPA"    "sdp:mosek:MATLAB_BINARY"
## [13] "nco:MOSEK:AMPL"           "nco:SNOPT:NL"
```

gRPC

- Apache license 2.0
- Based on HTTP/2
- Streaming support
- Designed for harsh environments (cancellation, timeout, load-balancing, ...)
- Payload agnostic (allows to use protocol buffers, JSON, XML, and Thrift)
- C library
- Official support for C++, Java, Python, Go Ruby, Node.js, ...
- gRPC - R (`https://github.com/nfultz/grpc`)

Protocol Buffers

- Protocol buffers are a flexible, efficient, automated mechanism for serializing structured data (like XML but smaller and faster).
- User defines the data structures (called messages) in a `.proto file`, and compiles it with protoc into source code (several languages are supported).
- **RProtoBuf**

Why gRPC

- fast
- can make use of binary data rather than just text
- type-safe
- supports streaming
- ...