物联网中间件

第三章 远程过程调用

参考材料

- Rutgers University： **Remote Procedure Calls**
  - https://www.cs.rutgers.edu/~pxk/417/notes/03-rpc.html

套接字通信的局限性

- 套接字提供的接口只有 write/read 的功能，只能传输字符流或者数据流，因此使用套接字不能直接调用其他电脑上的过程（函数）。
- 网络编程使用函数结构更加方便
- 问题：本机如何使用套接字调用服务器上的过程function A(int x)?

套接字通信的局限性

- 套接字提供的接口只有 write/read 的功能，只能传输字符流或者数据流，因此使用套接字不能直接调用其他电脑上的过程（函数）。

- 问题：本机如何使用套接字调用服务器上的过程A(int x)?
  - 将过程名以及参数打包（marshals），使用套接字传递
  - 服务器通过套接字接受，再解包（unmarshals）分析，完成过程调用后将结果打包，通过套接字返回
  - 本机通过套接字接受，再将结果解包

套接字通信的局限性

- 直接调用系统函数（套接字）的缺陷：
  - 不易修改，不易测试
  - 只有过程A(int x)是业务相关的，打包、解包、传输等代码都是与业务无关的，业务的变动，以及不同的业务都需要重写这部分代码。
  - 整个系统耦合度很高，维护代码时修改一个地方会牵连到很多地方，如果修改时没有理清这些耦合关系，那么带来的后果可能会是灾难性的，特别是对于业务变化较多以及多人协作开发维护的项目，修改一个地方会引起本来已经运行稳定的模块错误，严重时会导致恶性循环。

RPCs

- 1984： Birrell and Nelson 提出了远程过程调用（Remote Procedure Calls)的机制

- 机器A上的过程可以调用机器B上的过程，调用时机器A上的过程阻塞，机器B计算完成将返回值传递给A后，机器A上的过程才继续运行

本地过程调用（local procedure calls）

- 不同的编译器、处理器架构有着不同的机制
- 一般情况下:
- 编译器负责处理参数转换，判定是否执行调用等等工作

- 处理器提供的机制：处理器提供调用（call）指令，将顺序执行的下一条指令的地址PUSH到堆栈中，同时将控制转换到调用地址。调用过程结束后，执行返回（return）指令，将堆栈顶部的地址POP出来，同时将控制转换到POP出来的地址处。

RPCs

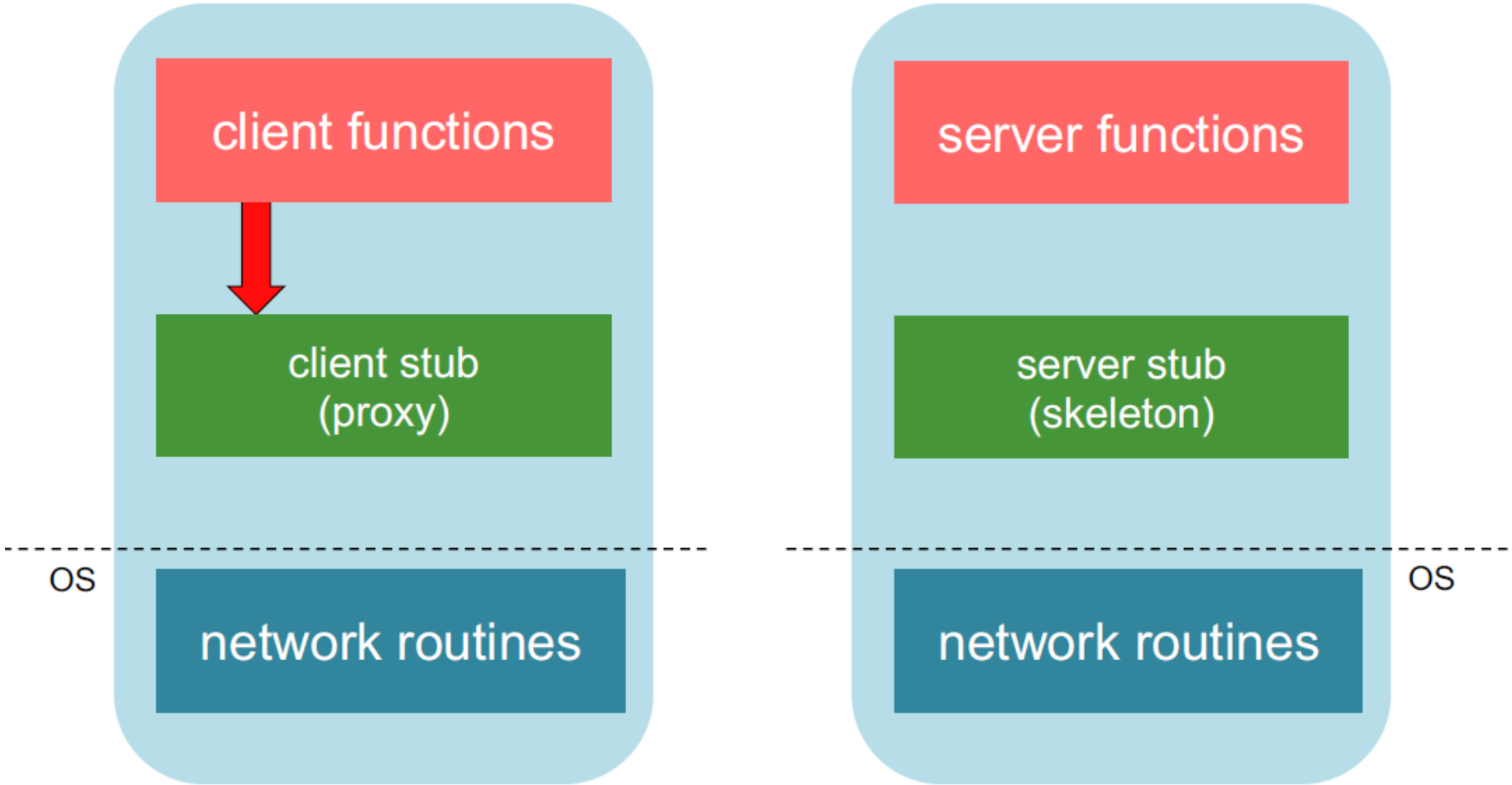
- 当我们需要调用远程电脑上的过程时，处理器和编译器所提供的本地过程调用机制都无法使用

- 编译器模仿本地过程调用的形式对RPCs提供支持
  - 编译器自动生成代码来发送信息调用远程函数
  - RPCs： language level construct（编程语言级别的构造），由编译器提供
- Socket：operating system level construct（操作系统级别的构造），由操作系统提供

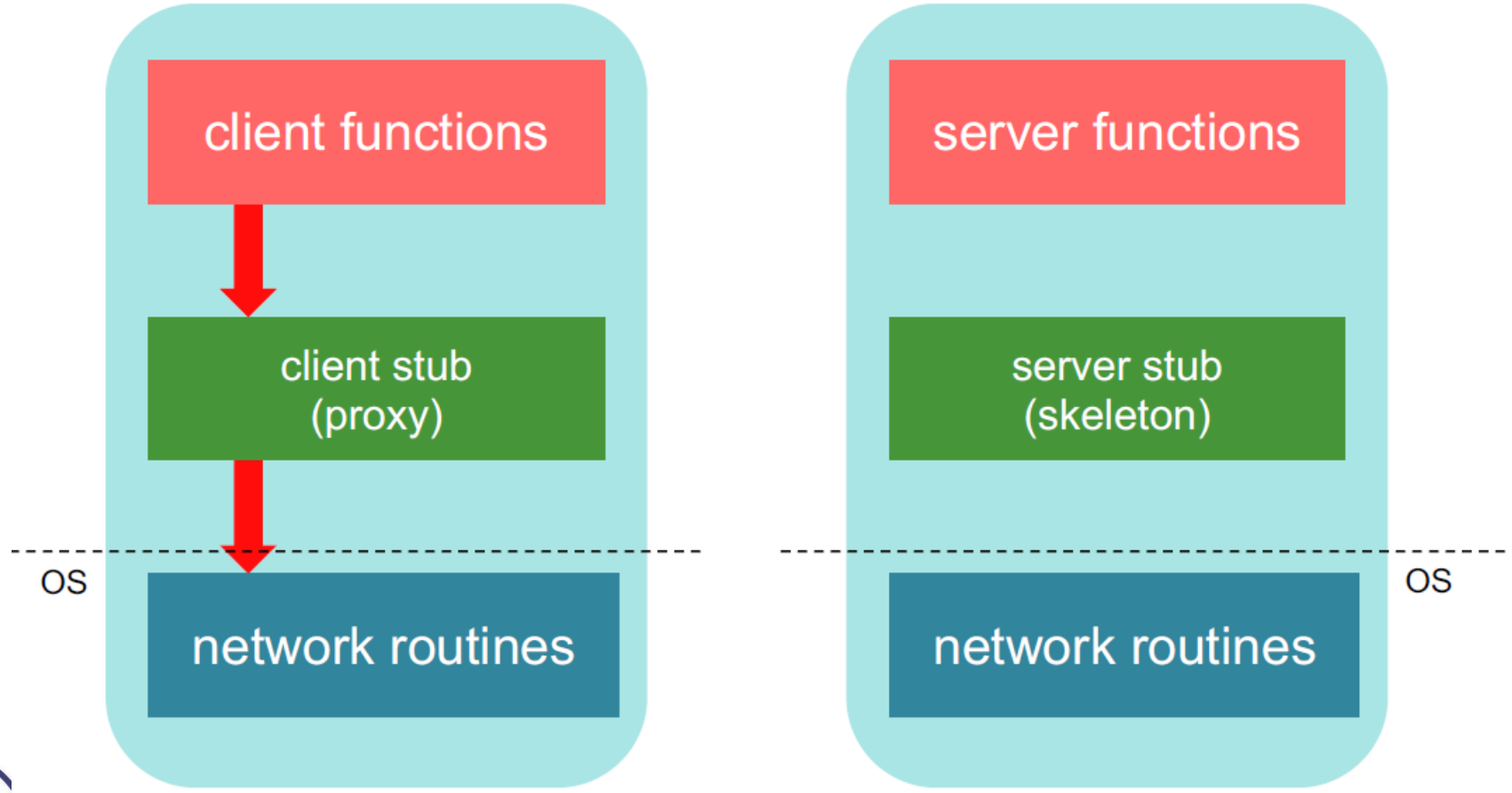
RPCs的实现

- 通过使用桩函数（stub functions)实现RPCs
- 客户端：桩函数（stub/proxy），客户端桩函数的接口和远程主机上的函数相同，可以视作远程函数，实际上桩函数包含通过网络发送和接受信息的代码。
- 服务器端：桩函数（ stub/skeleton）接受请求，调用本地函数

1.客户端调用桩函数


client functions
client stub
(proxy)
OS
network routines
server functions
server stub
(skeleton)
OS
network routines

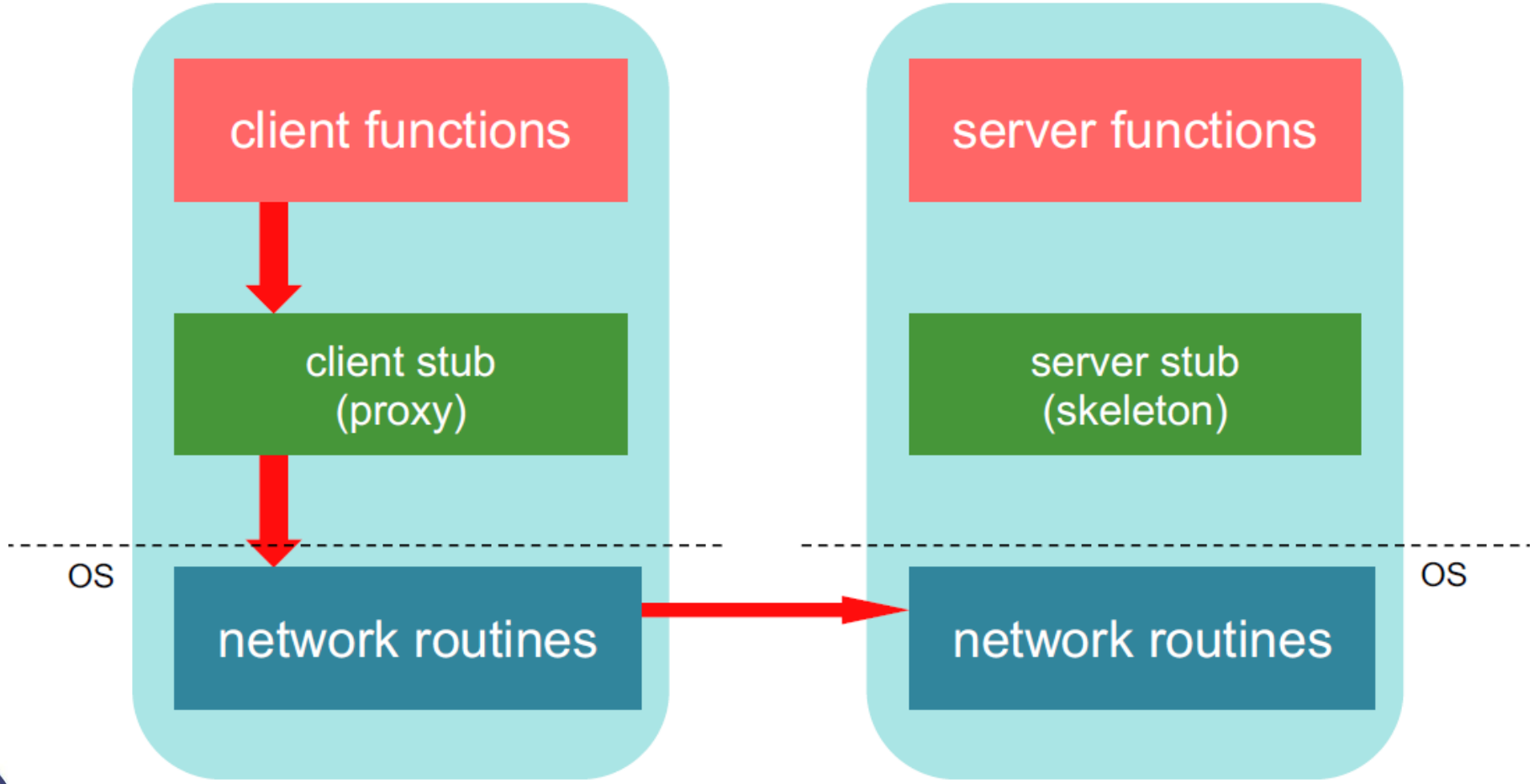
2.桩函数将参数打包成网络消息



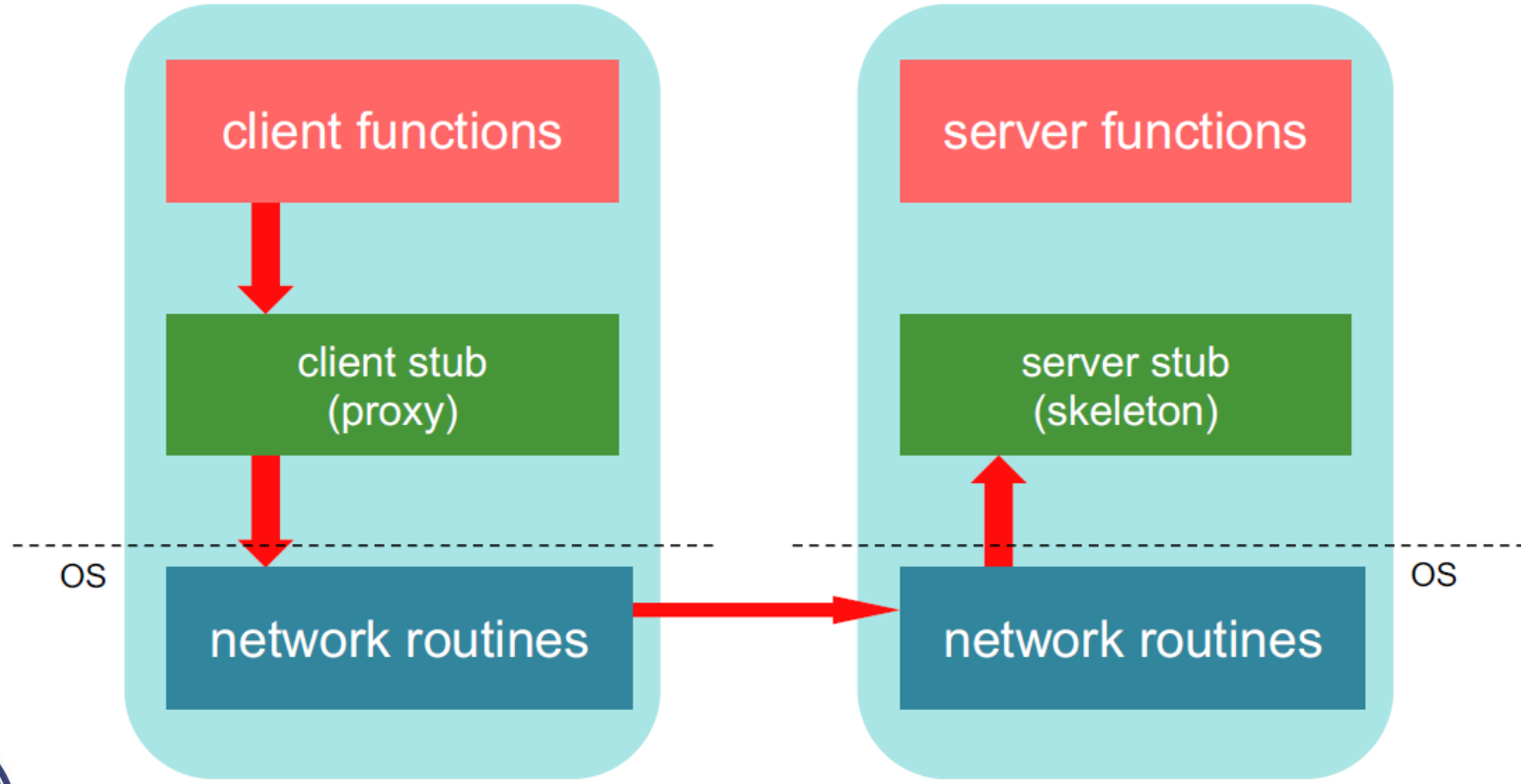
Marshaling & Serialization

- Marshaling（数据打包） 和 Serialization（序列化）基本上可以看做同义词
- 数据打包（Marshaling）： 将参数变为可以被其他进程重建（unmarshaling)的形式。数据打包采用序列化的方法。
- 序列化（Serialization）： 把一个对象转换成一串字符串，使得该对象可以通过网络传输

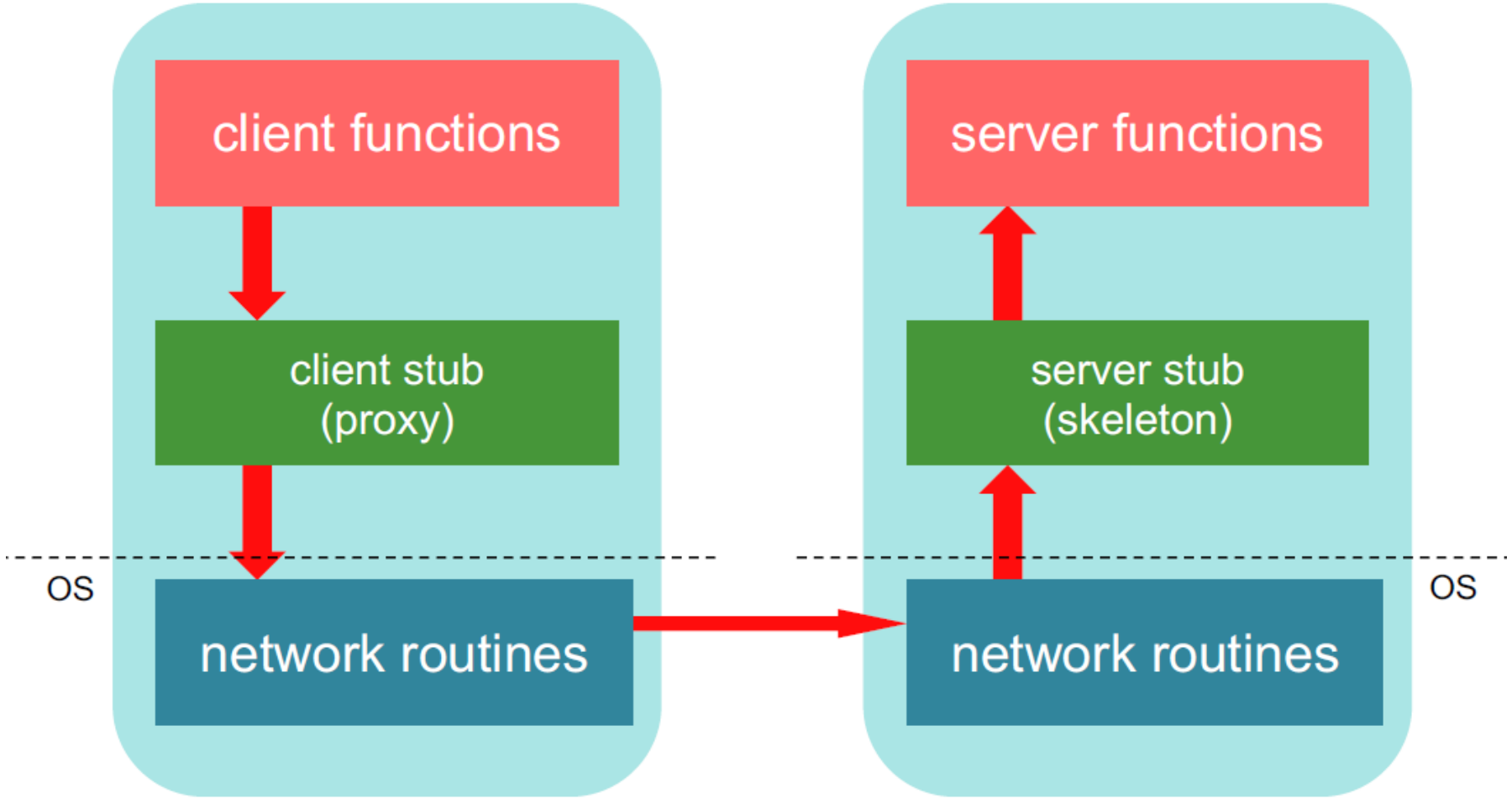
3.把网络消息发送到服务器



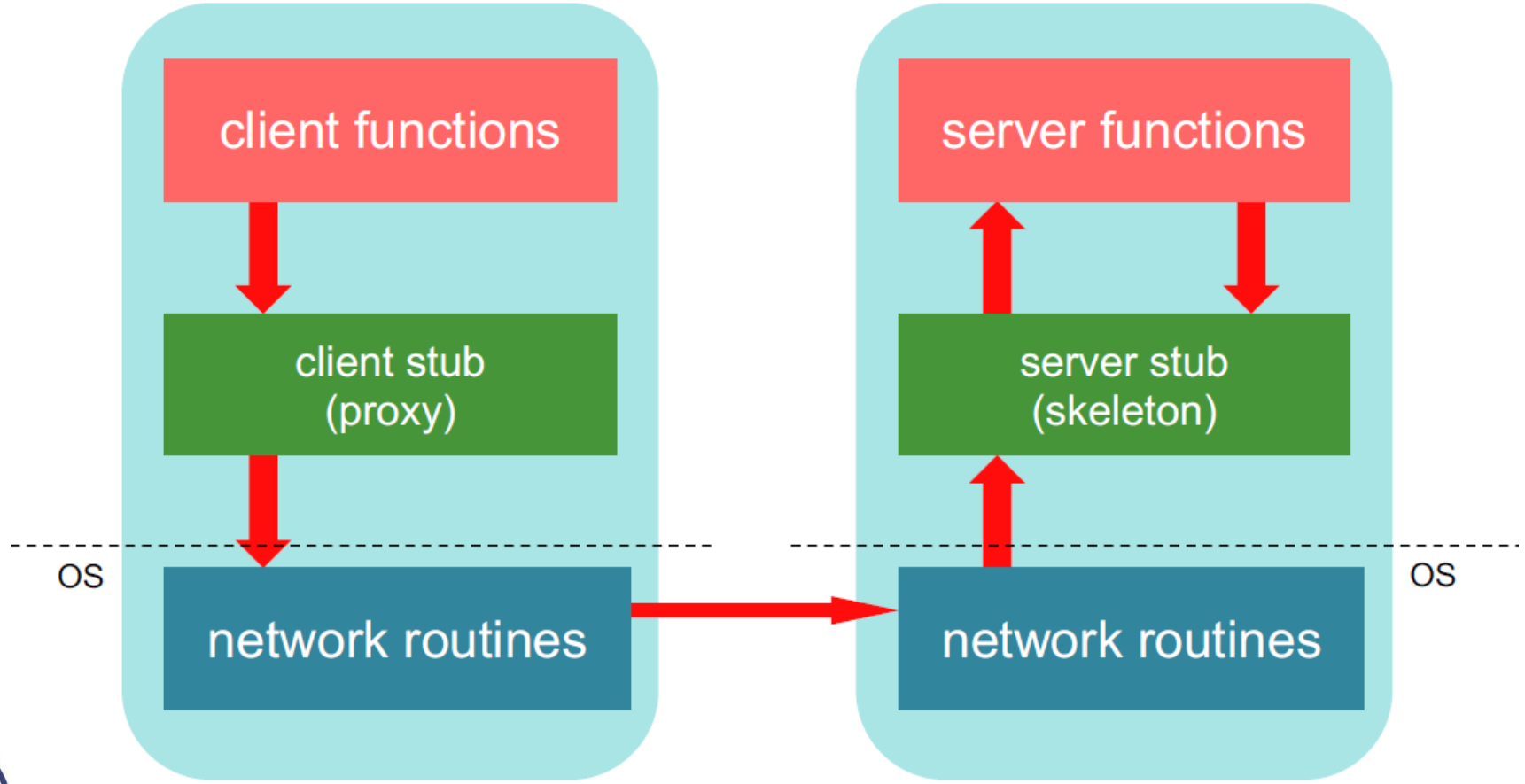
4.服务器桩函数接收消息



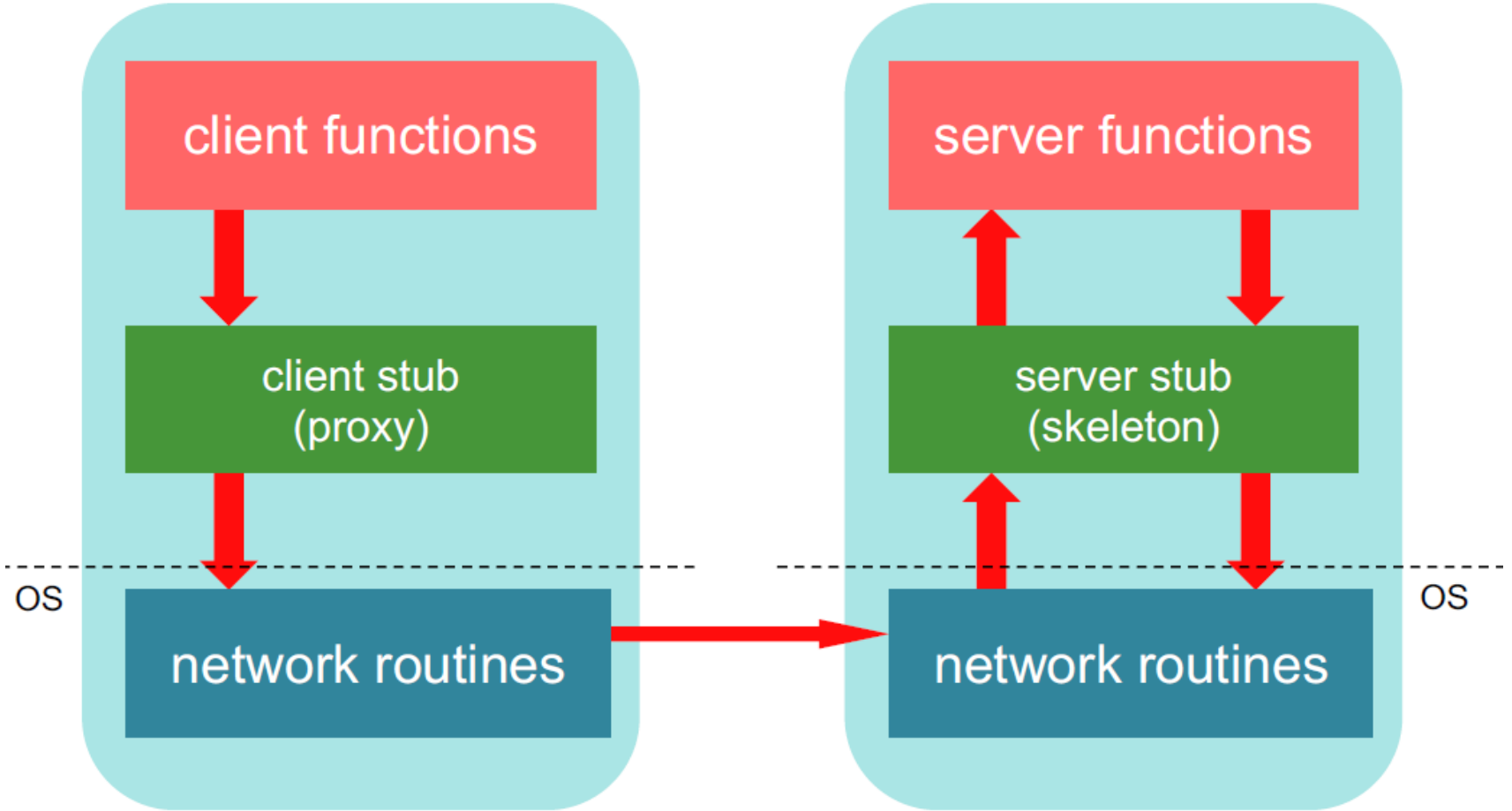
5.解包参数，调用服务器上的本地函数



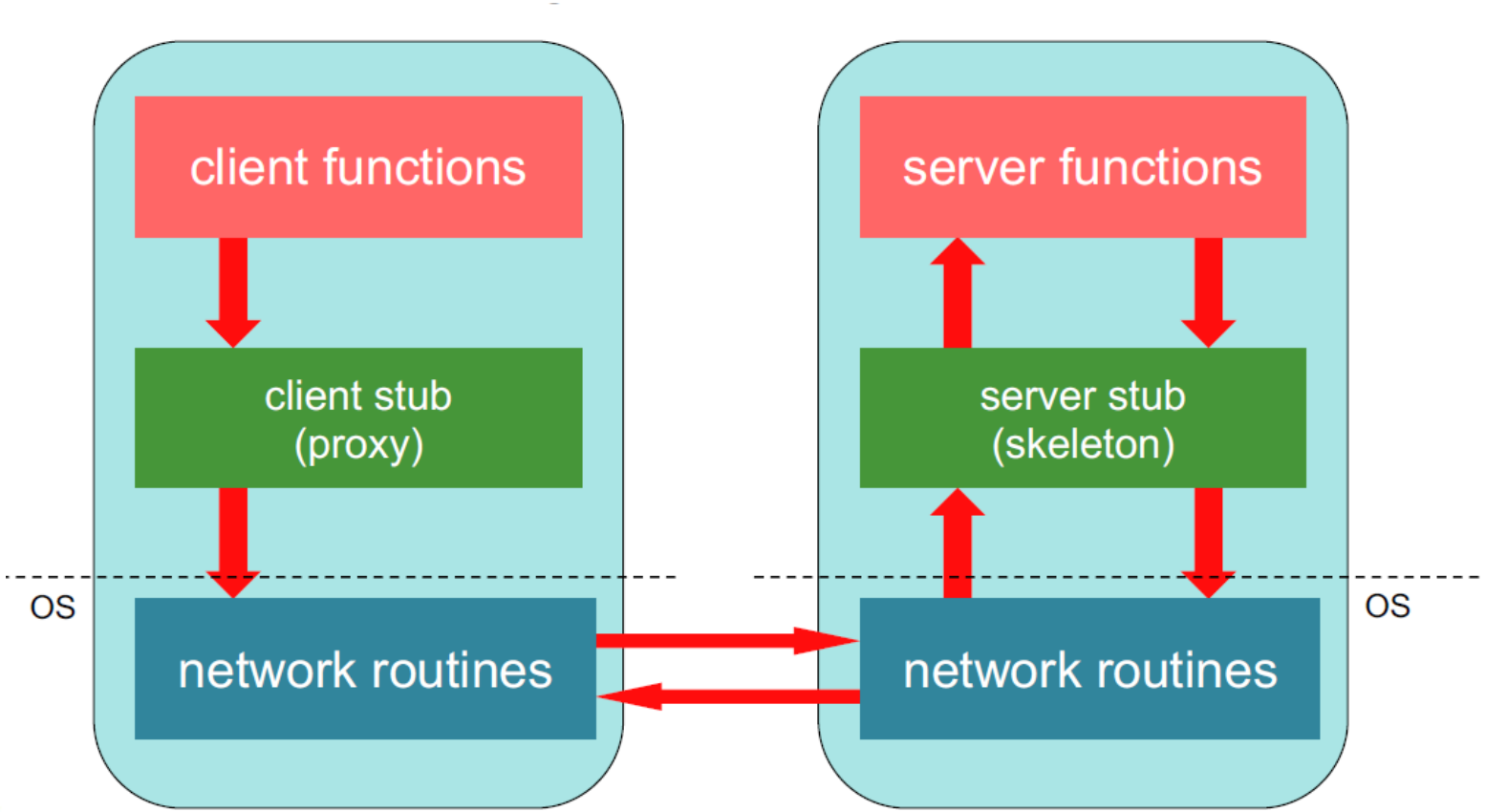
6.服务器上的本地函数返回计算结果



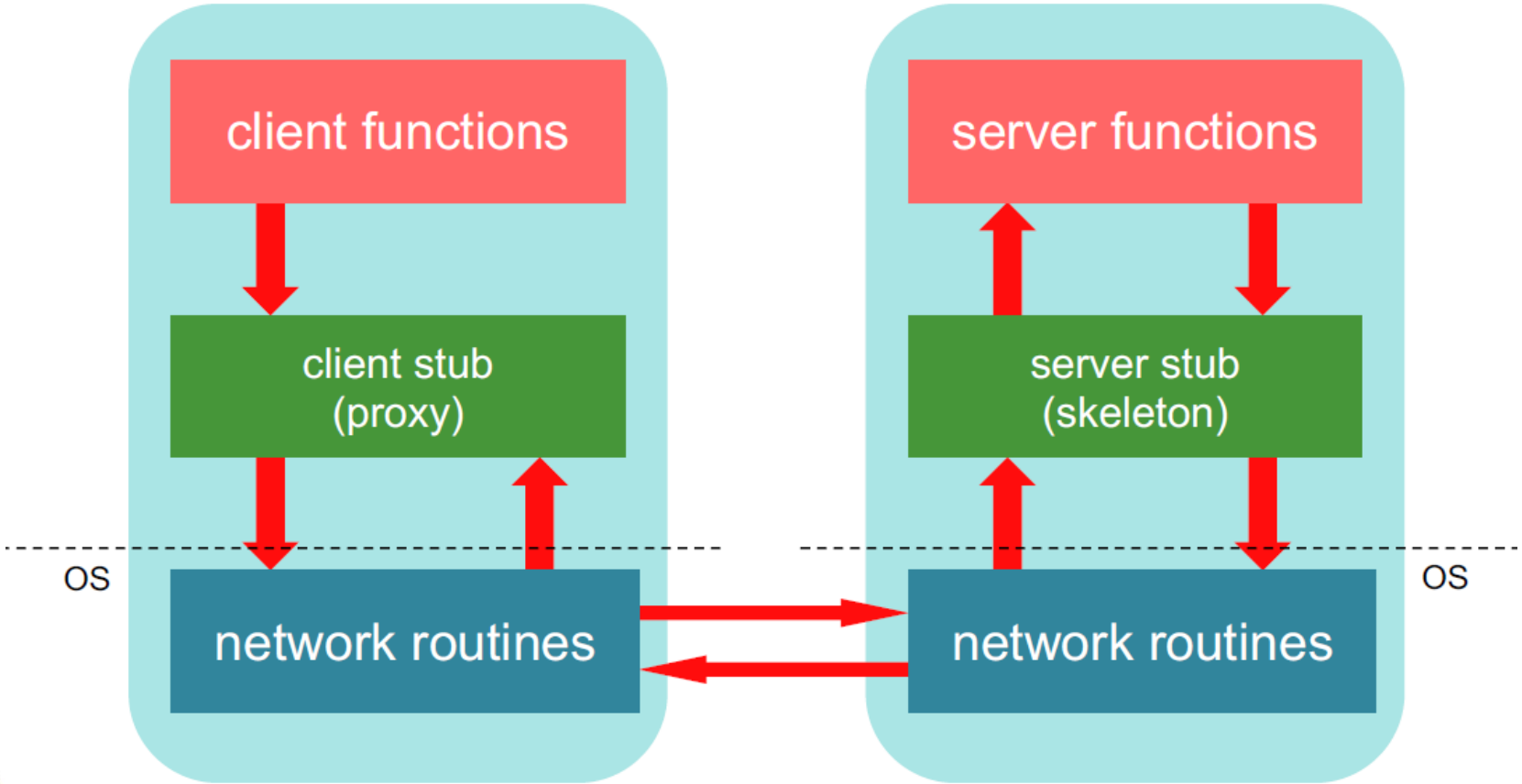
7.将返回值打包


client functions
client stub
(proxy)
OS
network routines
server functions
server stub
(skeleton)
OS
network routines

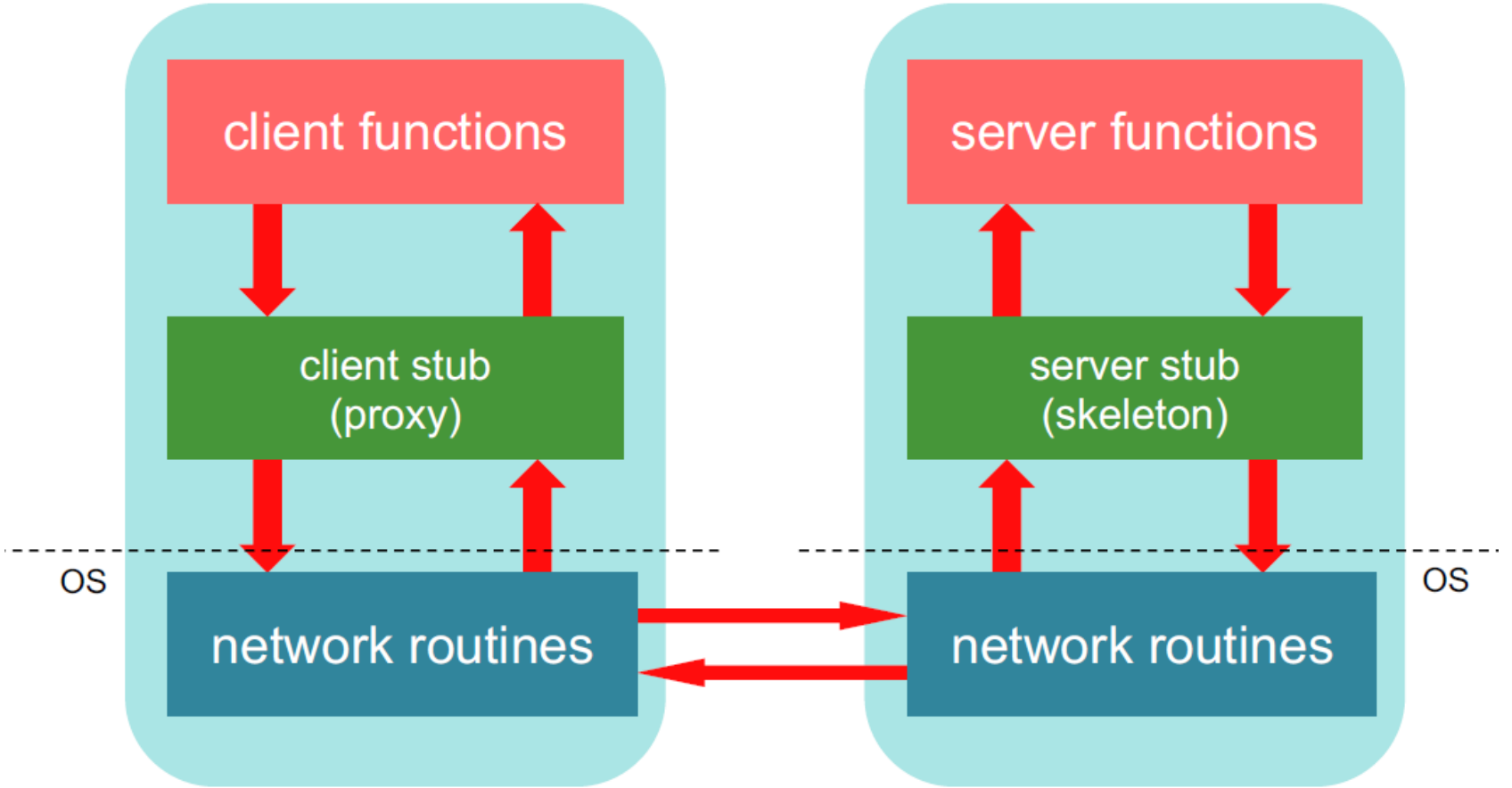
8.通过网络发送消息


client functions
client stub
(proxy)
server functions
server stub
(skeleton)
OS
OS
network routines
network routines

9.客户端桩函数接受消息



10.解包返回值，返回给客户端



客户端桩函数

- 客户端桩函数的接口和远程主机上的函数相同，可以视作远程函数
- 实际功能：
  - 打包参数
  - 发送信息
  - 等待服务器回应
  - 解包返回信息
  - 返回给调用者

服务器桩函数

- 实际功能：
  - 接受信息
  - 解包参数
  - 调用本地过程
  - 打包返回值
  - 发送信息

RPCs优势

- 提供给用户调用远程过程的接口

- 简化了实现互操作的难度和复杂度:
  - 所以网络相关代码都是由桩函数实现
  - 用户不用操心网络传输、网络字节序等底层实现
    - IP地址，端口号设置转换
    - 运输协议选择

数据表示

- 远程过程调用经常需要在异构系统中通信
- 远程主机的数据标识形式可能不同：
  - 不同的字节序：大端模式，小端模式
  - Integer，float等的字节数不同
- 因此对于所以能当作参数传递的数据结构，RPCs需要一个统一的标准编码方式以实现异构系统中的通信

数据表示

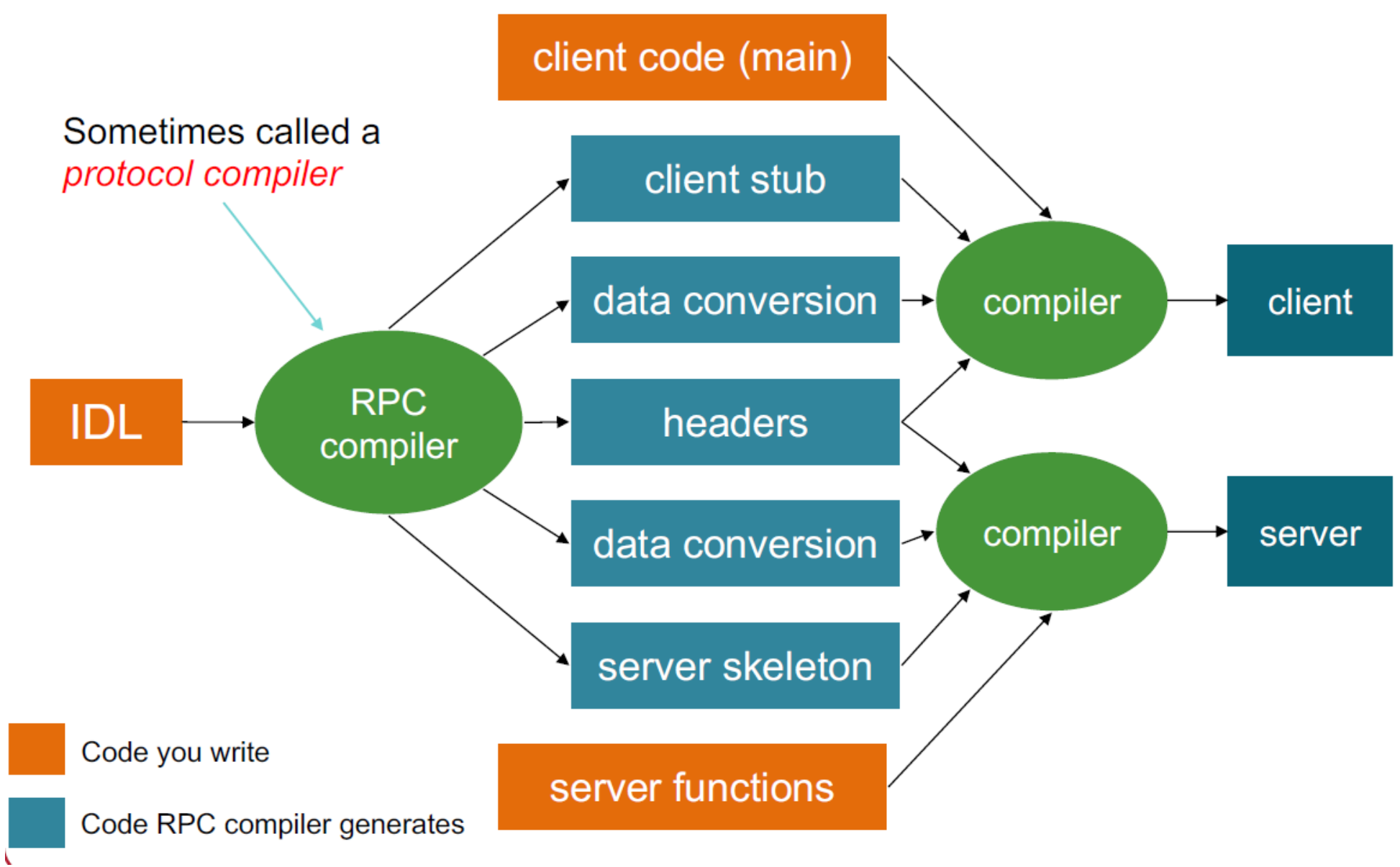
- Implicit typing（隐式类型）： 只保留数据的值不保留数据类型
  - ONC RPC‘s XDR (eXternal Data Representation)
  - DEC RPC‘s NDR

- Explicit typing(显式类型）： 在保持值的同时保存数据类型
  - JSON（JavaScript Object Notation)
  - Google Protocol Buffers
  - XML

RPCs 编程实现

- 大部分编程语言并没有在语言层面上支持远程过程调用
  - C，C++

- 解决方案：
  - Interface Definition Language（IDL）接口描述语言：用来描述远程过程
  - 使用专门的编译器来产生桩函数
  - IDL和函数原型类似，包含函数名，参数，返回值


client code (main)
Sometimes called a
protocol compiler
client stub
data conversion
compiler
client
IDL
RPC
compiler
headers
compiler
server
data conversion
server skeleton
server functions
Code you write
Code RPC compiler generates

RPCs 编程实现

- 客户端需要设定和RPC相关的参数:
  - 传输类型（TCP or UDP）
  - 服务器地址
- 服务器端不需要做大的更改

ONC RPC（Sun RPC）

- ONC RPC是第一个商用的RPC库以及RPC编译器
  - Mid 1980s 由Sun 公司提供，后来成为Open Network Computing协会的标准

- ONC RPC支持绝大多数POSIX和POSIX-like操作系统，比如Linux，BSD，SunOS等等

ONC RPC（Sun RPC）

- ONC RPC提供名为rpcgen的编译器来处理远程过程接口定义，并产生客户端和服务器端的桩函数
- ONC RPC一系列库函数来支持数据传输打包等工作
- ONC RPC使用XDR（eXternal Data Representation)这种数据序列化格式，使得参数能够在各种异构系统中传输。

ONC RPC 例子

- RPC接口定义（文件名后缀为.x，比如data.x)
  - 函数声明、版本号以及一个唯一的程序号

ONC RPC 例子

- Rpcgen在处理data.x会生成以下文件：
- Data.h
  - 包含函数的声明等，客户端和服务器端都需要包含该文件
- Data_svc.c：
  - 包含服务器端桩函数的代码
- Data_clnt.c:
  - 包含客户端桩函数的代码
- Data_xdr.c
  - 包含将数据转换为XDR格式的代码，客户端和服务器端都需要包含该文件

ONC RPC 例子

- 用户需要编写客户端程序（client.c), 服务器程序（server.c)，RPC接口定义（data.x)

- 客户端和服务器端分别先编译data.x，生成桩函数等对应的文件，然后客户端编译client.c和对应的桩函数连接在一起，得到客户端程序；服务器端同理。

服务器端运行流程

- 服务器上程序启动后，运行服务器端桩函数，将进程调入后台运行，桩函数建立一个socket并且和一个本地接口绑定在一起
- 接着桩函数调用svc_register，注册程序号和版本号，同时将程序号、版本号、端口号发送给port_mapper。
- Port mapper是一个系统启动时就运行的进程，用来追踪程序号、版本号、端口号。

客户端运行流程

- 客户端先调用cint_create，依据远程系统的名字、程序号、版本、协议，与远程系统的port mapper通信，来找到正确的端口号
- 客户端收到端口号后，调用RPC 桩函数，给服务器发送一条消息，等待服务器的回应。收到服务器回应后，客户端桩函数将返回值返回给调用程序。

示意图

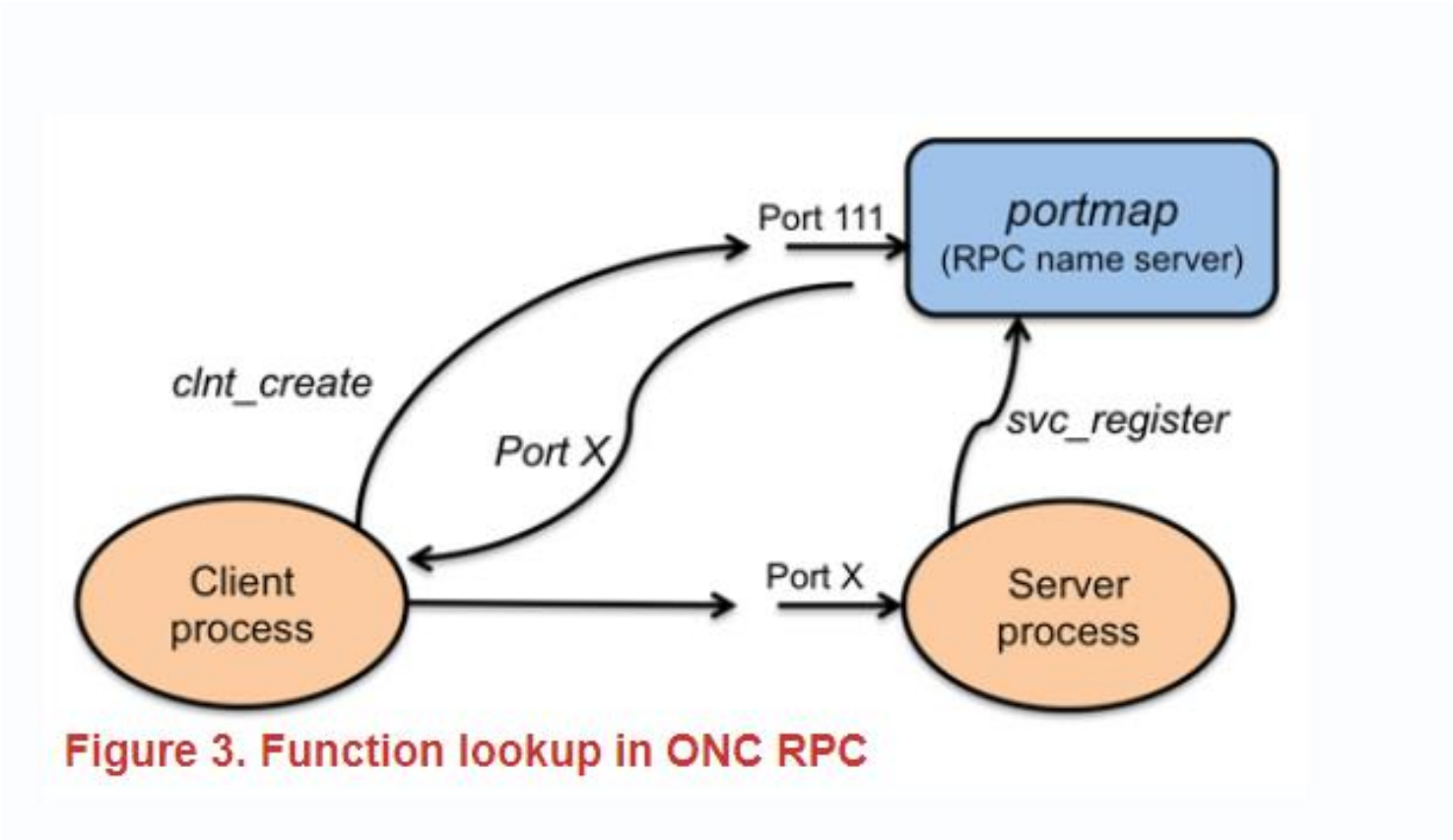


Figure 3. Function lookup in ONC RPC

答疑