DOI:10.3969/j.issn.1006-6403.2020.04.018

一种基于 UDP 通信的远程过程调用方法

[吕晶 刘伟旻 全佳 黄昌金]

摘要 RPC 是一种有效的远程网络服务请求调用方法，针对主流 RPC 框架均为基于 TCP 协议，复杂的架构制约了其在轻量级应用程序的有效应用，为进一步简化系统复杂度、减少系统开销、提高传输效率，提出一种基于 UDP 通信的远程调用方法，采用 UDP 协议，利用负载均衡策略，基于服务进程向注册进程注册服务的方式，应用进程从注册进程获取服务进程的相关信息，实现应用与服务进程的直接 RPC 通信，有效地提高系统通信传输效率。

关键词： UDP RPC 负载均衡 C/S 模式

吕晶

广州汇智通信技术有限公司副总裁，主要研究方向为新一代移动通信技术、人工智能和大数据技术。

刘伟旻

广州汇智通信技术有限公司应用研究院科研专家，主要研究方向为人工智能、网络空间安全。

全佳

广州汇智通信技术有限公司互联网技术研究院系统架构师，主要研究方向为新一代移动通信技术、云计算和大数据技术。

黄昌金

广州汇智通信技术有限公司互联网技术研究院副院长，主要研究方向为新一代移动通信技术、云计算和大数据技术。

1 引言

远程过程调用（Remote Procedure Call，RPC）是一种通过网络从远程计算机程序上请求服务。通过 RPC，使用者无需了解底层网络技术，快速开发部署业务功能逻辑 [1]。因此，在 RPC 在分布式系统中的系统环境建设和应用程序设计中有着广泛的应用 [2]，如：分布式操作系统的进程间通讯、构造分布式计算的软件环境、远程数据库服务、分布式应用程序设计、分布式程序的调试等。

RPC 主流实现的方式 [3] 有：基于 TCP 协议、基于 HTTP 协议。基于 TCP 协议实现 RPC，由于处于协议栈底层，可更灵活地对协议字段进行定制，减少网络传输字节数，提高性能，但受所定义协议的局限，需要关注底层

实现细节，难以实现跨平台调用，不同的终端需要开发不同的工具包来进行请求发送和响应解析，代码量高，工作量大[4]；基于 HTTP 协议实现 RPC，作为通用的格式标准，使用 JSON 和 XML 格式开发相对成熟，但与 TCP 传输性能的存在较大差距[5]。

综上所述，基于上述协议的主流 RPC 框架，虽然有适用面广、功能强大的优点，但是也存在着代码量多、流程复杂、对系统开销较大的缺点，用于小型应用程序上过于庞大，因此，本文提出一种基于 UDP 通信的远程过程调用方法，采用对系统开销较小的 UDP 协议，利用负载均衡策略，基于服务进程向注册进程注册服务的方式，应用进程从注册进程获取服务进程的相关信息，实现应用与服务进程的直接 RPC 通信，具备构架轻便、系统开销小的特点，满足小型应用程序的轻量化要求，具有重要的意义。

2 基于 UDP 通信的远程调用方法

RPC 系统采用 C/S 模式，请求程序作为客户机，服务提供程序作为服务器，由请求程序向服务程序发送调用信息，能够获得答复信息并获得进程结果，交互流程如图 1 所示。

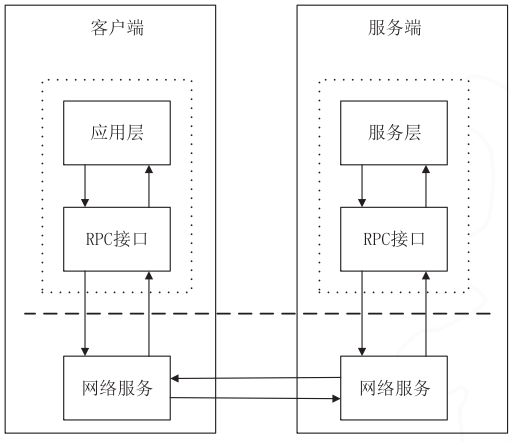

图 1　RPC 交互流程

基于 UDP 通信的远程调用方法，包含以下重要部分：①服务端服务进程注册模块、②客户端查询注册进程模块、③应用进程负载均衡策略、④客户端发送 RPC 服务请求模块、⑤服务端执行对应的业务流程模块、⑥服务端发送响应信息模块、⑦客户端接收响应信息模块，各部分对应的步骤如图 2 所示。

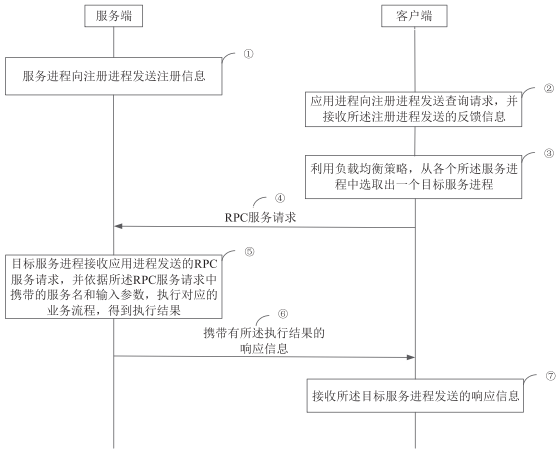

图 2　基于 UDP 通信的远程调用方法

系统处理流程如下：①首先，客户进程中，应用层调用 RPC 接口的 API 发起请求，调用系统网络接口发送请求，通过网络数据包从客户端发到服务端；②然后，服务进程中，RPC 请求从服务端的系统层传递到用户层的 RPC 接口，通过 RPC 请求解析后，找到对应的服务层，并将执行结果发回给 RPC 接口，调用系统网络接口回复执行结果；③最后，客户进程中，执行结果通过网络数据包从服务端发到客户端，从客户端的系统层传递到用户层的 RPC 接口，将执行结果转给对应的应用层，完成请求任务。

各部分详细介绍如下。

2.1 服务端服务进程注册模块

服务进程是提供 RPC 服务的应用程序，需要注册在服务器上，其对应信息包含：RPC 服务信息（服务名、服务说明等）以及相应的 UDP 端口号。服务信息的结构定义如下：

```
struct RpcServiceInfo
{
std::string name;          // 服务名
std::string ver; // 服务版本号
std::string help;          // 服务说明
}
```

2.2 客户端查询注册进程模块

客户端的应用进程是需要使用 RPC 服务的应用程序，当应用进程需要调用 RPC 服务时，可根据服务名向注册进程发送查询请求；注册进程可以依据查询请求中携带的

服务名，查找当前可以提供相应 RPC 服务的服务进程，并向应用进程发送反馈信息，反馈信息中可包含有提供 RPC 服务的各个服务进程的链接信息，包括：服务进程的 IP 地址、UDP 端口号等。反馈信息的结构定义如下：

```
struct GetServiceResMsg
{
std::string serviceName;              // 服务名
std::list<RpcService> serviceList;   // 服务进程信息列表
};
```

其中，RpcService 的定义如下：

```
struct RpcService
{
std::string ver; // 服务版本号
std::string ip;              //IP 地址
int port;                    //UDP 端口号
}
```

2.3 负载均衡策略

每个服务进程可以提供多个不同的 RPC 服务，因此，针对同一项 RPC 服务而言，可以提供该项 RPC 服务的服务进程可能有多个，即应用进程接收的反馈信息中可能包含有多个服务进程的相关信息。

系统应用进程设计按照可以依据负载均衡策略，选取出一个合适的服务进程作为目标服务进程，利用该目标服务进程提供相应的 RPC 服务。

2.4 客户端发送 RPC 服务请求模块

应用进程依据该链接信息，可向目标服务进程发送 RPC 服务请求。设计基于 UDP 通信的远程调用，将 RPC 服务请求以 UDP 数据包的形式发送到目标服务进程所在 IP 地址的 UDP 端口上，RPC 服务请求中可以包含请求号、服务名、请求参数等信息。RPC 服务请求的结构定义如下：

```
struct RpcReqMsg
{
std::string reqId;                   // 请求号，每次都不同
std::string serviceName; // 服务名
std::string textArg;                 // 参数（字符串）
std::vector<char> binArg;            // 参数 ( 二进制 )
}
```

为了可以一次发送多个 RPC 服务请求，在 RPC 服务请求中增加了参数 reqId，该参数在每个请求中都是不同了，唯一标识一个 RPC 服务请求消息。参数 serviceName 是请求的服务名。参数 textArg 和 binArg 是请求参数，格式是服务自定义的。为了方便使用，textArg 使用扩展性好的 JSON 格式，保存可以用字符串表达的参数，binArg 保存不能用字符串表达的参数，这样两种类型的参数已经可以覆盖业务需求，并且良好的扩展性，便于后期维护。

2.5 服务端执行对应的业务流程

目标服务进程提供的 RPC 服务可能有多个，依据 RPC 服务请求中携带的服务名，可以获知需要执行哪一项 RPC 服务，从而调用相应的业务流程。请求参数可以看做是该业务流程的输入参数，经过业务流程的处理，可以得到相应的执行结果。

2.6 服务端发送响应信息

业务进程可以一次发送多个 RPC 服务请求，为了便于区分每个 RPC 服务请求的执行结果，在响应信息中可以携带请求号和服务名，其中，该请求号和 RPC 服务请求中的请求号相同，该服务名是和 RPC 服务请求中的服务名相同。响应信息的结构定义如下，

```
struct RpcResMsg
{
std::string reqId;                   // 请求号，和 RPC 服务请求中一致
std::string serviceName;  // 服务名，和 RPC 服务请求中一致
std::string textResult;              // 执行结果（字符串）
std::vector<char> binResult;         // 执行结果（二进制）
}
```

响应消息中的参数 reqId、serviceName 和 RPC 服务请求中的参数是一致的，textResult 保存可以用字符串表达的结果，binResult 保存不能用字符串表达的结果。

2.7 客户端接收响应信息

如果响应信息的数据量很大，可以分为多个 UDP 包发送，在消息的包头中可以用消息的分片序号来标识每个数据包，在客户端的 RPC 接口层中进行重组后提交给上层。

3 结论

本文所提的基于 UDP 通信的远程过程调用方法，采用了对系统开销较小的 UDP 协议，设计了简单快捷的处理流程，快速的配置下发和状态查询功能既能满足轻量级应用程序的需求，又能减轻系统负担，在 2/3/4/5G 通信网业务功能设计、DPI 设备系统、协议解析设备系统中有广泛的应用，特别适用于功能简单、网络情况良好的应用场景。

参考文献

1 Lahman A,Shaaban Y,Fransazov M,et al.Object-oriented remote procedure calls for browser applications:U.S.Patent 10,223,181[P].2019-3-5.

2 Mauroner F,Baunach M.Remote instruction call:An RPC approach on instructions for embedded multi-core systems[C]//2018 IEEE International Conference on Industrial Technology (ICIT).IEEE,2018:1442-1446.

3 郑馥薇，沈卓炜．基于数据分发服务的远程过程调用系统 [J]. 计算机应用 ,2018,38(S2):239-242.

4 于天，黄昶．一种高性能异步 RPC 框架的设计与实现 [J]. 信息通信 ,2018(03):127-129.

5 刘小舟，龙辛，刘智磊，等．半实物仿真平台中 RPC 的设计实现 [J]. 机械工程与自动化 ,2016(06):61-62.

（收稿日期：2020-01-10）

（上接第 66 页）

某些应用中，B 设备的单应用解析识别能力要强于 A。因此，我们可以根据具体的需求，通过这些测试结果多方面来评定 DPI 设备的解析识别能力强弱。

3.5 测试方法改进及展望

目前有关 DPI 的测试历时不长，相关经验不多，DPI 解析识别能力方法仍然还有不少提升和改进的空间，比如：

（1）业务种类的时效性和覆盖的全面性

互联网的业务日新月异，测试的流量库一定要及时更新，能反映现网的中最具有代表性的业务。

（2）业务流量的比例和吞吐量

目前测试中，由于测试手段的限制，业务的流量与现网差别较大，更多的是以海量的背景流量情况下产生少量的带有有效测试信息的测试业务流量。这种方法满足了测试流量的需求，但对真正现网运维提供的参考相对有限，未来也期望能够从流量和业务吞吐量上越来越接近真实现网。

（3）加密业务的识别和处理

现在互联网业务中，以 SSL/TLS 为代表的加密流量的比重越来越高，在这种情况下，DPI 设备要进行业务识别的难度也越来越大，从测试的角度也需要考虑适应这种变化的趋势。

4 结束语

DPI 解析识别能力的测试是评估 DPI 设备整体功能性能的最重要一环，同时也是 DPI 设备最本质的特点——深度报文检测的最直接体现。本文探讨的 DPI 解析识别能力的测试方法，为当前 DPI 设备解析识别能力测试提供一种有效可行的方案，提升了 DPI 测试的科学性与合理性。由于互联网业务变化快、应用新增多、应用更新快等特点，DPI 解析识别能力的测试需要实时跟进业务应用的变化、增加测试的业务样包以及更新新的业务应用回放技术，并不断完善整体测试技术方案。

（收稿日期：2020-01-03）

勘误启事

《广东通信技术》2020 年第 3 期封面及书脊，“第 42 卷”有误，应为“第 40 卷”；“第四十二卷”有误，应为“第四十卷”。

《广东通信技术》2020 年第 3 期目次页，“第 42 卷”有误，应为“第 40 卷”。

特此说明。

本刊编辑部

2020 年 4 月 12 日