

多进程 DNS 服务器中通信方法的设计与实现

沈婷婷^{1,2}, 金 键¹, 毛 伟¹

(1 中国科学院 计算机网络信息中心, 北京 100190; 2 中国科学院 研究生院, 北京 100190)

摘 要: 多进程技术是目前大型软件开发中降低复杂性的有效途径. 针对 BIND10 服务器软件的开发提出一种基于安全 socket 的多进程通讯方法, 其中包括一种多进程通信协议 (MPCP), 通讯接口 CI 和消息路由进程 MRouter 的设计. CI 使用 SSL 协议来保障通信的安全性, MRouter 使用消息寻址转发机制实现进程之间的点对点 and 点对多点通信. 目前该设计已经在 BIND10 软件项目中应用并实现, 能够完成各个进程模块的实时通讯、支持分布式部署、保证容错性和健壮性、便于扩展功能的实现等.

关键词: 通讯; 多进程; 安全; 服务器; DNS

中图分类号: TP302

文献标识码: A

文章编号: 1000-7180(2011)04-0076-05

Design and Implementation of Communication Method for Multi-Process DNS Server

SHEN Ting-ting^{1,2}, JIN Jian¹, MAO Wei¹

(1 Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China;
2 Graduates School, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Multi-process technology is an effective method to deal with software complexity. This paper proposes a multi-process communication method, which includes a multi-process communication protocol and the design of a communication interface and a message router daemon. CI uses SSL protocol to keep security, MRouter uses message forwarding mechanism to achieve point to point and point to multi-point communication. This framework has already been used in BIND10 project. It can finish real-time communication between any processes, support distributable deploy, keep fault tolerance and robustness, be convenient to extend.

Key words: communication; multi-process; security; server; DNS

1 引言

域名系统 (Domain Name System, DNS) 是十分重要的 Internet 基础设施, 主要完成域名和 IP 地址之间的转换, 实现网络资源的精确定位. 众多的网络服务 (WWW, E-mail, FTP 等) 都建立在 DNS 体系基础之上.

域名服务器负责存储域名信息并完成域名请求解析任务. 随着网络环境的日益复杂、DNS 数据的迅速膨胀以及大量新的需求不断涌现, 对域名服务器软件的性能: 稳定性、安全性及可扩展性提出了更

高的要求.

广泛使用的 DNS 服务器是设计于 1998 年的开源的 BIND^[1], 目前已经发布多个版本. 从设计之初到现在, Internet 的操作环境、软件工程技术、计算机架构、用户需求等都发生重大变化, 设计人员很难提出向后兼容 15 年到 20 年的软件架构. BIND9 采用单进程多线程的架构^[2], 应对变化的策略是代码重构和附加工具软件, 这使得工作量剧增和软件松散度的提高, 同时维护工作也变得纷繁冗杂. 对此, ISC^[3] 基于 BIND9 的现有架构提出了一种基于多进程的服务器架构, 即下一代 DNS 服务器 BIND10.

收稿日期: 2010-05-27; **修回日期:** 2010-08-10

基金项目: 国家发改委 2009CNGI 项目 (CNGI-09-03-04)

主要设计思想是将各个部件耦合成一个整体,在提供高性能查询基础上增强交互性,提供实时在线监控、在线更新配置等功能.同时要求支持分布式部署、保证健壮性和安全性,容易扩展等.

本文针对 BIND10 服务器软件的需求提出一种基于安全 socket 的多进程通讯方法,其中包括一种多进程通信协议(MPCP),通讯接口 CI 和消息路由进程 MRouter 的设计.实现了进程之间高效、安全、灵活、快速的通信,同时实现了底层的封装,提供简洁的编程接口,方便工程人员设计开发.

2 BIND10 项目简介

BIND 10^[4]是 ISC 在 2009 年 4 月主导研发的下一代域名解析软件,针对 BIND 9 软件在安全方面的缺陷和下一代互联网而设计,进一步提升域名解析的安全性和查询的性能.

图 1 为 BIND10 的系统架构图.该架构采用功能分解的方法将系统划分成若干个进程,各进程完成独立的功能:命令检查、配置检查、用户查询、动态更新、状态统计等.

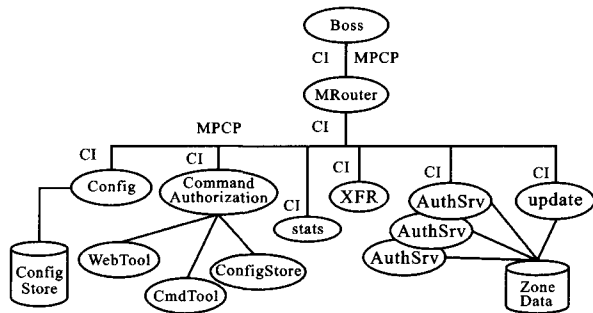


图 1 BIND10 服务器架构图

3 多进程通讯方法的设计与实现

开发 BIND10 的关键之一是为运行在相同主机或者不同主机的模块提供安全的通讯,实现跨平台资源的整合.从图 1 可以看出,通讯模块 MRoute 在 BIND10 架构占有重要的地位,为各模块提供了协作的平台,是实现其他功能的基础.各模块通过 MRouter 进行消息的转发实现控制信息和数据信息的传递.

通信性能的好坏在很大程度上决定着整个系统的性能,要求通信模块能提供正确、有序、安全和高效率的服务.对此本文设计并实现了一种基于安全 socket 的多进程通讯方法.该设计包含三个部分:

(1) 消息路由进程(Messages Router, MRouter):负责维护通信进程的安全连接和地址信息结

构,监听连接上发送的消息,根据地址信息转发到接收进程.其原理类似于路由器,实现消息的转发,支持消息的单播、多播和组播.

(2) 多进程通信协议 MPCP (Multi-Process Communication Protocol, MPCP):规定了消息转发的格式,各种命令的功能等.

(3) 通信接口 CI(communication interface):进行消息的编码、发送、接受、解码.

使用多进程通信协议 MPCP 规定进程通讯消息的格式,使用基于 OPENSsl^[5]库实现的 SSL 协议保证安全性、采用通用的通讯接口(CI, Communication Interface)和消息转发机制实现点对点通信(单播)和点对多点(多播)的安全通信.

3.1 MRouter 的工作原理

MRouter 工作流程见图 2.该进程维护 socket 事件^[6]队列和地址信息表两个数据结构,socket 事件队列存放监听 socket 和若干通讯 socket,地址信息表保存各通讯进程的路由信息.监听消息模块使用系统调用 select 监听事件队列中的 socket 是否有数据到达,根据 socket 类型将其分发到处理通讯 socket 模块和处理连接 socket 模块处理数据.处理通讯 socket 模块按照消息类型进行处理或者转发消息,并访问地址信息表验证源地址和确定目标进程地址;处理连接 socket 模块将生成连接 ID 号,存储连接信息到地址信息表中.

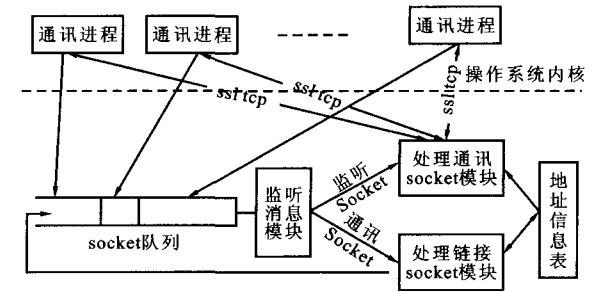


图 2 MRouter 工作流程图

消息转发凭借地址信息表(标识地址映射表,群地址映射表)确定目标进程.通讯进程与 MRouter 建立连接时生成标识地址映射表,注册时生成群地址映射表.当 MRouter 收到通讯消息时,首先对源 ID 进行验证,再读取 group、instance 和 to 字段的值,查找地址表确定目标进程的套接字集合,因此实现组播和单播.

3.2 MPCP 协议的设计

多进程通讯协议完成进程之间的点对点和点对多点通信,MPCP 是应用层协议.该协议中的消息

格式如图 3 所示.



图 3 数据包格式

数据包有头部和数据信息两部分,头部有版本号、数据包长度、头部长度、源 ID、目的 ID、组名、实例名、应答标志、序列号、目的进程名、TYPE.

版本号是 2 个字节,当前值为 0x00001.

数据包长度字段是数据占 4 个字节.

头部长度占 2 个字节.

ID 唯一标识一个连接,可以称为地址,由进程 MRouter 自动生成和管理,在产生 ID 时必须保证其唯一性.当通讯进程与 MRouter 建立连接后,MRouter 给每个连接分配一个 ID 号.通讯进程使用 getid 消息获取该 ID,消息中使用源 ID 标识发送进程,目的 ID 字段标识目标进程.

系统中的进程分成若干组,每个进程是所在组中的一个实例.进程可以注册任何一个组中任何一个实例的消息.

序列号是一个 16 位的标识符,由通讯进程分配.这个标识符被拷贝到应答中,请求者用来匹配发出的查询.

消息的类型,包括 getid, subscribe, unsubscribe, send, stats 等.

getid:获取连接的 ID,是进程与 MRouter 建立连接之后发送的第一个消息,进程获取 ID 后可以在该连接上发送任意类型的消息,此消息中源 ID 不存在.

subscribe:进程启动后使用该消息进行注册,完成注册后进程能接收到发送给该组该实例的所有消息,若实例名为 *,则表示监听发送给该组的所有消息.

unsubscribe:进程取消对某个组和实例的消息的监听,但可以继续发送消息.

以上消息由 MRouter 进程直接处理.当 type 为 send 时,MRouter 进程会根据路由信息对消息进行转发到相应的通讯进程.

stats:系统进程使用该消息发送状态信息.

图 4 为使用该协议建立安全连接的顺序图.通讯进程首先使用 SSL 协议与 MRouter 建立安全连

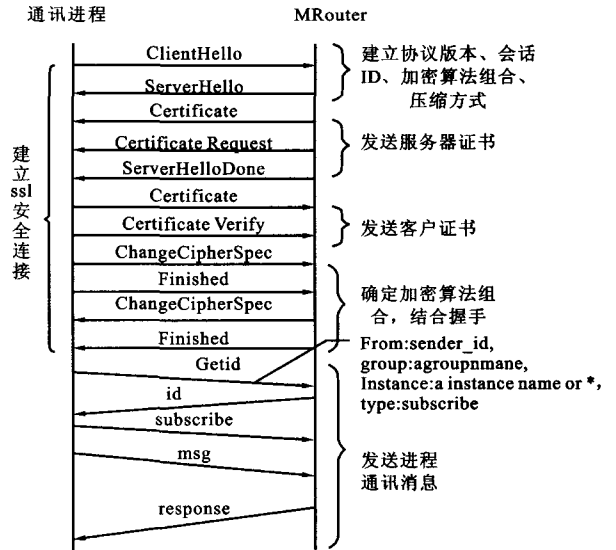


图 4 建立安全连接顺序图

接:完成相互之间的身份认证,确定加密算法组合,建立共享会话密钥;接下来,再使用 MPCP 协议进行通讯,用协商好的共享密钥加密和解密通讯消息.

3.3 通讯接口的设计

3.3.1 信息编码

头部信息和数据信息均编码成 Hash 映射(key,value).key 值编码成长度和数据;value 是简单数据类型(Data)、Hash 映射、列表(List)和 None 值中的任意一种,类型的编码格式见表 1.

表 1 项编码格式表

名称	长度	描述
类型长度	1 个字节	数据类型和长度占用位数
长度	不定	数据的长度
类型数据	不定	具体类型的编码数据

其中类型长度域包括两个内容:项的类型和长度所占位数,两部分的具体编码见表 2.

表 2 类型长度编码表

类型	类型编码	长度	长度编码
Data	0x01	Length_32	0x00
Hash	0x02	Length_16	0x10
List	0x03	Length_8	0x20
None	0x04		
Type_Mask	0x0f	Length_Mash	0xf0

编码的语法表达式如下:

<头部或数据信息编码> ::= <Hash 映射编码>
<Hash 编码> ::= <key 编码> <value 编码>.....
<key 编码><value 编码>

<key 编码> ::= <length(key)><string(key)>
<value 编码> ::= <Hash 编码> | <List 编码> |
<None 编码> | <Data 编码>
<List 编码> ::= <value 编码>.....<value 编码>
<None 编码> ::= 0x04
<Data 编码> ::= TyLen length(string(Data)) string
(Data)

其中 length(Data) 是 Data 数据的长度, string
(Data) 是将变量 Data 转化成 ASCII 码的字符串.

3.3.2 类型数据编码

Hash 映射中每个项是 (key, value) 对格式, 项
的顺序不定, 其编码格式见表 3, 其中 key 为小于
256 位的非空字符串, 数据项再按其具体类型进行
编码, 因此数据项的编码是递归的.

表 3 Hash 数据编码表

名称	长度	描述
键值长度	1 个字节	键值的长度信息
键值(key)	不定	键值的名称
数据(value)	不定	具体类型的编码数据

List 是一个确定顺序的数据项的集合, 每个数
据项按照其类型编码方式进行编码; Data 类型是简
单的数据类型(如整形、布尔类型、字串等), 编码成
字符串; None 表示 List 某个位置或者 Hash 的 val-
ue 没有数据.

例子:

Hash 数据类型编码:

```
{  
  "from" => "sender@host",  
  "to" => "recipient@host",  
  "seq" => 1234,  
  "data" => {  
    "list" => [ 1, 2, nil, "this" ],  
    "description" => "Fun for all",  
  },  
}
```

编码后的格式为:

```
长度: 0x60(96 字节)  
0x04 "from" 0x21 0x0b "sender@host"  
0x02 "to" 0x21 0x0e "recipient@host"  
0x03 "seq" 0x21 0x04 "1234"  
0x04 "data" 0x22 0x2d  
  0x04 "list" 0x23 0x0c  
    0x21 0x01 "1"  
    0x21 0x01 "2"  
  0x04
```

```
0x21 0x04 "this"  
0x0b "description" 0x21 0x0b "Fun for all"  
解码是编码的逆过程, 这里不再详述.
```

3.3.3 接口 CI

CI 通信接口完成通信功能, 是通信结构的底
层, 提供进程接收和发送消息的 API, 实现底层通信
细节的封装. 主要功能是:

- (1) 保持与 MRouter 进程的安全 TCP 连接.
- (2) 接收本地进程传送的消息, 将消息编码成
网络传输格式并发送给 MRouter.
- (3) 接收 MRouter 发送的消息, 并把消息解码
传送给本地进程处理.

目前该接口将所有的信息都编码成 ASCII 码
进行传输, 不支持其他类型编码.

进程使用该接口进行通讯. 进程启动时使用 CI
接口将自动连接 MRouter 进程, 同时取得连接的
ID, 此后调用 CI 的 API 进行发送和接收消息. 通信
架构见图 5.

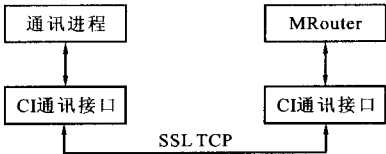


图 5 进程通信架构

4 本文通讯方法的优缺点分析

DNS 是大型分布式系统, 提供重要的基础服
务. DNS 服务器是极其重要的 Internet 基础设施,
对性能、稳定性、安全性及可扩展性有较高的要求,
将该通信设计运用到 DNS 系统开发中, 具有如下优
点: (1) 支持分布式部署. 分布式部署可以解决性能
问题, 对于 DNS 服务器, 查询处理进程每秒钟要接
收大量的查询, 通过检索数据库给出应答, 系统的
CPU、内存、硬盘、带宽的要求都很高, 可以将这个
模块放到单独的机器上工作. 也支持多个查询模块
的同时部署, 实现 DNS 集群. (2) 提高系统的健壮性
和容错性. 一个进程的故障不会影响其他进程的
正常工作. (3) 实现跨语言编程. 每个进程可以根据其
质量需求和业务逻辑等选择不同的语言. (4) 保证分
布式部署进程之间的安全性. 采用 SSL 协议和证书
分发范围的限制保证安全性, 进程分布在不同的主
机上, 拥有证书的进程才能与 MRouter 建立连接,
连接上传送加密的机密数据. (5) 支持点对点通讯
(单播), 点对多点通讯(组播), 还可以很容易的扩展
支持广播方式.

缺点: (1) 证书文件的管理和分发. 使用 OPENSSL 的 API 很容易生成自认证证书和密钥, 但是有一定的使用期限, 这就要求管理员定期对证书文件进行更新再分发. 另外没有解决证书的分发问题, 目前需要管理手动分发到各通讯进程的主机上. (2) 信息编码的支持有一定的局限性. 目前只支持 ANSI 中的 ASCII 编码. (3) 对于分布式部署, 应该对负载均衡算法进行进一步研究, 保证分布在不同主机上的相同功能模块的负载均衡.

5 结束语

目前该通讯方法已经应用在 BIND10 服务器软件开发中, 能够有效协调进程间通信. 同时, 该方法的应用不局限于 DNS 服务器, 对于基于多进程的任何项目都可以直接使用. 对于支持多种信息编码、以及多种数据类型方面应有所扩展; 应考虑使用更智能的方式分发证书文件以及分布式部署的负载均衡问题.

参考文献:

- [1] Paul Albitz, Cricket Liu. DNS and BIND[M]. 5th Edition. America: O'Reilly, 2006.

- [2] Tatuya Jinmei, Paul Vixie, Implementation and evaluation of moderate parallelism in the BIND9 DNS server [C]// Proceedings of the annual conference on USENIX '06 Annual Technical Conference. Boston, MA: ACM, 2006: 12-20.
- [3] Internet Systems Consortium (ISC) BIND [CP/OL]. [2010-05-27]. <http://www.isc.org/>.
- [4] BIND10 project[EB/OL]. [2010-05-27]. <http://bind10.isc.org/>.
- [5] OpenSSL project[EB/OL]. (2004) [2010-05-27]. 2004. <http://www.openssl.org/>.
- [6] Stevens W R, Fenner B, Rndoff A M. Unix 网络编程 第 1 卷: 套接口 API[M]. 杨继张, 译. 3 版. 北京: 清华大学出版社, 2006.

作者简介:

沈婷婷 女, (1985-), 硕士研究生. 研究方向为计算机网络、DNS 服务器的安全与设计开发.

金 键 男, (1976-), 硕士. 研究方向为计算机网络系统、互联网寻址技术.

毛 伟 男, (1968-), 研究员, 博士生导师. 研究方向为下一代互联网技术、资源定位与寻址技术、网络安全技术.

(上接第 75 页)

功耗报告分析结果, 证明了本文所采用方法的有效性, 最终实现 45.7 万门、内核功耗 199 mW、时钟频率 105 MHz 的 SoC.

参考文献:

- [1] Payman ZarkeshHa, Meindi James D. Optimum on chip power distribution networks for gigascale integration [C]// Proceedings of the IEEE 2001 International Interconnect Technology Conference. Burlingame, CA, USA: IEEE, 2001(6): 125-127.
- [2] 田志新, 刘勇攀, 杨华中, 等. 基于马尔可夫-蒙特卡洛采样的电源网络分析[J]. 微电子学与计算机, 2008(12): 1-4.
- [3] 严晓浪, 杨垠丹. 超深亚微米集成电路 IR-DROP 快速论证分析的研究[D]. 杭州: 浙江大学, 2004.

- [4] 高海霞, 张弘. 低功耗测试向量产生技术的研究[D]. 微电子学与计算机, 2009, 26(1): 213-216.
- [5] Chen Hongyu, Wang Q, Mori M, et. al. Optimal Planning for Mesh-Based Power Distribution [C]// Proceedings of the ASP-DAC 2004. Asia and South Pacific Design Automation Conference. N J, USA: IEEE, 2004 (1): 444-449.
- [6] 陈春章, 艾霞, 王国雄. 数字集成电路物理设计[M]. 北京: 科学出版社, 2008.
- [7] 曾晓洋, 郭小川. 低功耗物理设计[D]. 北京: 中国学位论文全文数据库, 2007.

作者简介:

桑红石 女, (1970-), 博士, 副教授, 硕士生导师. 研究方向为集成电路.