

# 短消息服务平台的设计与实现

## Short Message Service Platform Design

刘荣辉 刘光昌 (暨南大学电子信息工程系 510632)

黄浩文 (广州 广东省数据通信网络公司 510632)

**摘要:** 本文参考中国电信的SMGP协议和有关规范, 为服务提供商SP提出了一套完善的、支持大流量通信的、具有开放性、扩展性的短消息服务平台的设计方案和实现方法, 对平台的架构、通信处理、业务处理、测试等多方面进行了重点论述, 并提出了其他可行的参考解决方案。

**关键词:** 短消息网关 SMGP 小灵通 SP 多线程

### 1 前言

随着手机用户的剧增, 短消息通信成为一种时尚的沟通方式。本文从服务提供商SP/CP的角度设计了短消息服务平台, 为手机用户(以小灵通手机用户为例)和互联网用户提供短消息服务, 为互联网与移动网络提供了方便快捷的通信桥梁。

### 2 系统结构与模块

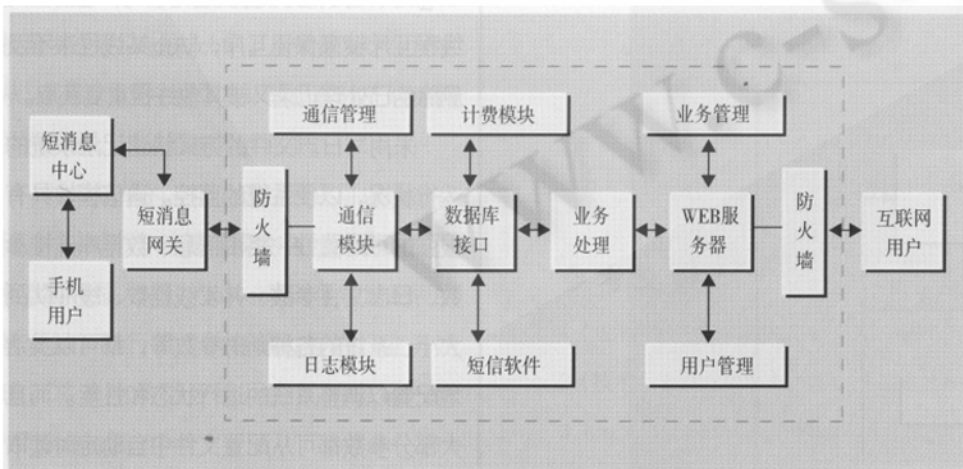


图 1

如图1所示, 虚线框内为服务提供商 (SP/ICP) 的短消息服务平台的结构和模块。

其中通信模块负责与短消息网关 (SMG) 通信; 通信管理模块用于对通信所需的路由信息和其他系统控制参数进行配置、管理和性能监控; 日志模块用于记录通信模块的动作, 为系统

监控、管理提供依据; 数据库接口模块提供了通用的数据库操作方法, 与数据库类型的选择无关, 方便系统的移植; 计费模块根据计费规则, 提取数据库的通信记录, 进行分析统计, 以形成话单与网关系统对帐, 同时提供自身业务量的统计分析功能; 业务处理模块主要通过数据库接口进行具体的短消息业务的处理; 业务管理模块为管理员提供了方便的业务维护管理功能, 方便了短消息业务的扩展; 用户管理模块主要是对短消息用户的信息、状态、积分等的维护管理以及平台管理员的管理; Web服务器为系统管理员和手机用户提供了基于浏览器的访问方式。通过开放的数据库接口, 服务平台还可以提供短消息客户端软件。

### 3 系统设计思想

系统采用多进程加多线程的方式, 基于SMGP/TCP/IP协议或HTTP协议, 采用流式套接字与短消息网关进行通信。不同的线程完成不同的功能模块。为了最大限度地提高系

统流量，并防止因来不及处理而丢失数据包，通信模块中的收发线程与上下行处理线程以内存中的消息缓冲队列作为数据接口，从而将收发和处理分开。

为了系统的开放性和扩展性，通信模块与业务模块必须分开，采用数据库作为这两者之间的接口，主要通过上行（用户发来的）短消息表MO\_Deliver和下行（发往用户的）短消息表MT\_Submit来实现。为了屏蔽数据库类型对系统的影响，采用面向对象的编程方法，构造统一的数据库接口模块。系统的各个模块通过统一的数据库接口进行耦合和交互。

## 4 通信模块的设计

### 4.1 HTTP 协议方式

UTStarcom的短消息网关支持SP以HTTP协议方式接入，在80端口上采用短连接进行通信（包括授权连接、发送一条请求、接收一个应答、关闭连接的过程）。SP和SMG互为HTTP的客户端和服务端，当SP向用户发送MO短消息时，以客户端的身份向SMG提交POST消息并接收应答；当SMG向SP发送源于手机的MT短消息时，SP作为Server端接收SMG提交的POST请求并返回应答。可以用SP实现服务器端，用java的URL类实现客户端。HTTP方式实现起来较简单，适合于短消息流量较小的时候。

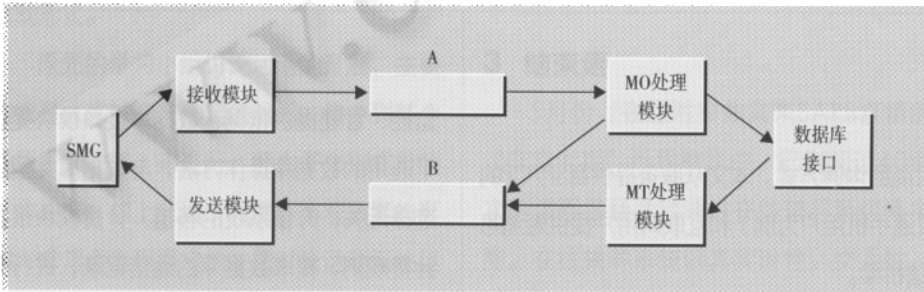


图 2

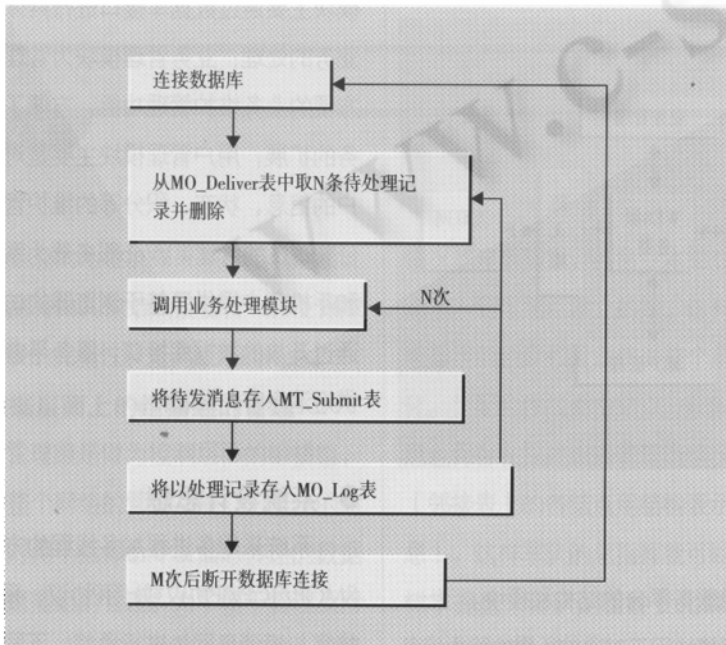


图 3

### 4.2 SMGP 协议方式

SMGP 是基于 TCP/IP 的短消息网关协议。如图 2，A 为接收缓冲队列，B 为发送缓冲队列。接收线程以长连接方式进行循环接收，将收到的数据包放入内存中的接收队列；发送线程组（多个相同功能的线程）从发送队列中取出数据包，以短连接方式进行异步的并行发送。收发过程通过调用中兴的 API 实现。MO 处理线程组从接收队列中取出数据包进行处理、写日志文件、调用数据库接口存储状态报告包到 MT\_Log 表中。MT 处理线程组轮流周期性地从数据库中读取一定量的记录，形成 Submit 数据包后放入发送队列。

缓冲队列的实质是一个指针数组，为了让多个线程可以互斥地从收发缓冲队列中存取数据包，在构造缓冲队列时采用了基于内存的两个无名二值信号灯和两个无名计数信号灯（semaphore），两个二值信号灯控制队列读写的互斥性，两个计数信号灯防止队列的存取过界。多个线程对公共的系统日志文件进行写操作时使用两个基于内存的二值信号灯便可保证写文件时的正确互斥，也可以使用线程互斥锁 Pthread\_Mutex\_Lock 来实现。当多个 MT 处理线程同时从数据库中的 MT\_Submit 表中读取待发记录时，也需要用线程互斥锁来保证互斥，防止某线程来不及删除的已处理记录又被其他线程重复获取。

采用写日志文件的方式快速记录系统的运行状况，以便系统的监控。通信模块具有很强的可配置性，路由信息、数据库连接参数、日志文件参数、并发线程数、缓冲队列大小、系统的各种延时参数等，都可以灵活地配置以调整系统的运行状态和性能。而且大部分参数都可从配置文件中自动定时读取并更新，而不需要重新启动程序。

## 5 业务处理模块的设计

业务处理进程可在专门的服务器上运行，编程语言和操作系统都无限制。采用线

程组进行并发的业务处理,业务处理线程的流程如图3所示。每个线程使用独立的数据库连接,为了防止数据库长连接异常,隔一定时间重连一次数据库。各个线程从MO\_Deliver表中提取/删除记录的操作由线程锁保证互斥,防止重复处理。处理后的业务记录存入MO\_Log表中以便进行计费统计。

对于实时互动的短消息业务,可以在Submit包的源号码(手机所收短信的发送方号码)中填写“SP代码+业务代码+问题代码”的长号码(总长不超过21位),用户回复后我们就可以在Deliver包中获取同样的长号码,然后就可以根据长号码中的业务代码和问题代码调用业务处理模块和进行对应的操作。如果某些网关不支持在Submit包的源号码中填写长号码,那么可以根据用户的会话状态表Session中的历史记录来调用不同的业务处理模块。用户的Session记录在每次业务处理中都进行更新,每次增加用户Session记录时将已过期活期的Session记录删除。

对于订阅类短消息业务,由于网关厂商一般都不提倡在Submit包中填写多个目的手机号,因为这样不便于确定状态报告的对应关系,不便于计费和对帐,故本平台采用逐一发送的方式。需要大批量发送的短消息放在批量发送表SendBatch中,用批量发送线程组从SendBatch表中提取并以较低的优先级进行并行发送,以免影响其他业务。

相对简单的业务处理过程还可以通过存储过程来实现。当通信模块中的MO处理线程将数据包插入MO\_Deliver表时便触发此表的触发器,执行相应的存储过程进行业务处理,将产生的下发给用户的消息插入到MT\_Submit表。

## 6 计费模块的设计

计费统计模块可以在一台Windows平台的服务器上运行,用C++Builder进行开发。计费时,对数据库中的MO\_Log和MT\_Log进行

分析,提取出收到成功状态报告的消息(用户手机成功接收)记录,根据计费规则表中不同业务的收费标准进行计算、汇总,形成话单,以便与运营商进行对帐。同时定时(每天)对各种业务进行统计,形成报表,为业务、管理提供决策支持。

## 7 管理模块与其他设计

平台采用了B/S架构,用Apache提供Web服务,用Jsp进行设计。系统管理员可以通过浏览器登陆系统进行业务的生成、业务规则的修改、计费规则的修改、以及业务信息的更新等业务维护管理,手机用户可以在网站上进行业务注册、业务信息修改、发送短消息、话单查询等操作。由于系统采用了数据库作为开放的接口,外部程序可以很方便地与短消息服务平台相交互,利用C++Builder的数据库操作控件便可轻松设计一个客户端软件,为用户或企业提供短消息接入服务。

## 8 通信模块的测试

测试条件:平台的通信处理机通过宽带网络与中兴的小灵通短消息网关相连,采用Linux9.0操作系统+MySQL数据库。用一个线程进行循环接收,用三个发送线程进行异步的并行发送。向广东的30万小灵通手机用户逐一发送短消息,并请求回送状态报告。

测试结果:发送速率达到150条/秒左右,接收速率120条/秒左右,CPU占有率小于10%,内存占用率极低。

结果分析:接收线程所收到的数据包是网关返回的发送状态报告,而状态报告是短消息中心将短消息发送出去之后才返回给网关,然后再转发给通信处理机。手机用户由于种种原因(关机,网络外)无法实时接收到短消息,造成网关转发给通信处理机的状态报告数小于发送的短消息数,这就是收发速度看起来“不均等”的原因。

如果增加发送线程的数目,调整线程的延时参数,在网关不进行流量限制的前提下,通信处理机的收发速度还可以大幅度提高,因此通信速率的瓶颈在于网关的流量控制,而不在于本平台的通信处理机。

## 参考文献

- 1 《短消息网关(SMG)协议 v1.3》, 中国电信集团公司, 2002年4月。
- 2 《技术规范 v1.3 版补充修订》, 中国电信集团上海研发中心, 2002年9月。
- 3 <http://skybird.9s9s.com/bbs/index.asp>。
- 4 《UNIX网络编程》,[美]W. Richard Stevens著,施振川等译,清华大学出版社,1999年7月第1版。
- 5 《短消息网关客户端开发包说明》,深圳中兴通信有限公司,2003年4月。