一种业务无关的通用协议①

贾欢欢，王 纯

(北京邮电大学 网络与交换技术国家重点实验室，北京 100876)
(东信北邮信息技术有限公司，北京 100191)

摘 要：在软件业高速发展的今天，不同系统间信息交换的重要性越来越突出。而信息交互协议的制定及与各自系统内部协议的转换不可避免，在这种情况下，一种与实际业务系统无关的通信协议的制定，可以使开发者只关注于业务部分，缩短了新系统开发的时间。而协议中不同的消息体格式的设计，也加大了协议的灵活性和适用性。

关键词：通信；XML；TLV

Service-Unrelated Communication Protocol

JIA Huan-Huan, WANG Chun

(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)
(EBUPT Information Technology Co. Ltd., Beijing 100191, China)

Abstract: Nowadays, with the rapid development of the software industry, the importance of exchange information among different systems has become more and more prominent. While the protocol formulation and the conversion from the exchange protocol to the systems' inner protocol is inevitable. At this point, the formulating of a service-unrelated communication-protocol, can solve all those troubles in the process of developing new systems, when the developers only have to concerns about the service-related parts, and thus the time for the system to be developed being shorten. The design of different message content formulates in the protocol, also brings flexibility and applicability to the protocol.

Keywords: communication; XML; TLV

随着计算机软件行业的发展，信息的作用越来越突出，信息交换的重要性也随之凸显。由于一般情况下，各系统都有内部数据信息格式，因此不同系统之间的信息交换，就涉及到系统间的协议制定以及与各自系统内的协议转换问题。

通常，当两个系统之间需要信息交换前，双方需要先商定一个新的协议来承载信息的交换，此协议会与他们之间的业务紧密相关。由于交换协议与系统之间业务的紧耦合性，一旦任何一方出现新的业务系统，原有的协议将不再适用，又需要有新的协议被制定，相应协议处理部分重新完成，既费时又费力，给业务的扩展增加了难度。

在这种情况下，为了新业务系统的快速开发及扩展，有必要制定一种与具体业务解藕的通用协议。协议与业务的分离，既可使业务系统开发人员只需关注于业务相关的部分，也可以在系统间的协议传递时加入更多业务无关的附加功能，如数据的安全性、快速性等。

① 基金项目:国家杰出青年科学基金(60525110);国家 973 计划(2007CB307100,2007CB307103);国家自然科学基金(60902051);中央高校基本科研业务费专项资金(BUPT2009RC0505);电子信息产业发展基金
收稿时间:2010-05-25;收到修改稿时间:2010-07-09

本文提出了一种可以完成相应功能的协议格式，并具体讨论了此协议内容的一些可能承载形式。

在实际应用中，各系统间协议的交换的具体过程虽然各不相同，但还是有很多共同的特点。

首先，因为信息基本是在网络上传递，基于系统的安全性考虑，系统之间并在物理连接建立后并不能立即进行业务间的通信，还需要进行鉴权，以确定连接的安全性和授权，然后才进行业务上的通信。一旦鉴权成功，有的连接是长连接类型，即一次鉴权成功后连接一直保存，一直可以收发信息；同时还有短连接类型，需要在每次通信前都先进行鉴权。同时，在通信过程中，对于长连接类型，在物理连接正常的情况下，为了确定对端系统的状态也是正常的，还会有一些心跳等其他连接状态验证机制。

由于系统间通信协议与系统内部协议不一定是同种协议(大多数情况下不一致)，各系统接收到发送端发送的消息后还会有协议从通信协议到内部协议的转换，当然，发送消息时也有从内部协议到通信协议的转换。

综上，系统间的通信涉及到的一般步骤为：1) 连接的建立与鉴权；2) 业务相关信息传递及连接维护信息的传递；3) 业务相关信息的协议转换。

当要开发的系统涉及到与其他系统之间的消息交互，或者需要扩展出与其他系统进行信息交互的功能时，如果没有已有的可用协议，则必须重新开发一个协议以及上面各步骤中涉及到的功能模块等，这会使开发人员在业务相关领域外的开发上花费大量的时间，加大了开发难度，尤其是需要快速开发的情况下。

为了解决这种问题，本文试着设计一种与具体业务解耦的系统间通信协议。因为与业务无关，一旦实现，任何新的系统在开发的时候都可以利用此协议作为信息交互协议。同时，合理选取适当的消息内容承载形式，步骤 3)中业务相关信息的协议转换步骤也会非常简单。在新开发系统的时候，利用此协议的编解码代理，用户只需要设置一些配置，就可以在各种情况下，从消息中取出自己想要的内容，做到只关注业务相关部分。

2　业务无关的通信协议的设计

参照互联网及通信中消息传递特点[1]，本文设计了一种通用协议的整体结构，其各部分组成固定，但具体内容又可随系统需要修改，相对灵活。

由于此协议设定在网络中传递，因此涉及到的数字类型传递均为网络字节序顺序，在相应的主机中编解码的时候需要先做字节序的转换。

2.1 协议结构图

如图 1，消息分为同步头+消息头+消息体+长度验证字段。

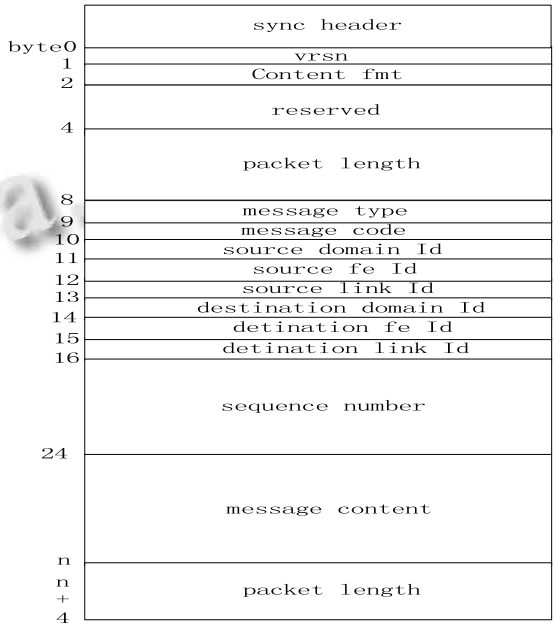

图 1　通用协议整体结构图

2.2 协议同步头及长度验证字段

这两部分保证了消息传递过程中的同步及纠错功能。

同步头主要用于消息错位时的恢复，它标志着一条消息的开始，长度验证字段则在消息末尾再一次注明了消息的长度，通过与所读的前面长度比较，可以判断此条消息是否在传送中出现了问题。

同步头可为 N 字节，前 N-1 个字节为 FF，最后一个字节为 00。软件应能识别大于 4 字节的同步头。当接收到连续 3 个以上的 FF 并紧接一个 00 的时候，就认为已经取得同步，并开始接收新的消息。

此协议中的同步头也可以自己定义，在协议编解码实现时会提供自定义的同步头的功能，以增加协议的灵活性。不定义时，默认为上面的说明。

2.3 协议头中各字段说明

消息头长度一共 24 个字节，是为协议在网络中的转发、接收而设定，含有路由等相关信息。其各部分

含义如下：

表 1 通用消息头中各字段长度及含义说明

名称	长度及类型	含义
vrsn	1 字节 无符号整数	协议版本号 初始为 1
Content fmt	1 字节 无符号整数	消息体格式，具体值参见 3.4
reserved	2 字节	目前无意义，留着作为扩展
packet length	4 字节 无符号整数	消息长度，不包括同步头、长度验证字段的长度
message type	1 字节 无符号整数	消息类型[可扩展] 1 控制类型 2 业务类型
message code	1 字节 无符号整数	具体消息类型 [自定义字段]
source domainId	1 字节 无符号整数	消息源地址三元组[可置空] 三元组标识着某一实际外部系统的具体链路 三元组与实际外部系统（及其各端口、链路）的映射关系，可由各系统灵活配置
source feId	1 字节 无符号整数	
source linkId	1 字节 无符号整数	
dest domainId	1 字节 无符号整数	消息目的地址三元组 [可置空]
dest feId	1 字节 无符号整数	
dest linkId	1 字节 无符号整数	
sequence number	8 字节 无符号整数	消息序列号，消息发送者应保证同一会话中的发送消息序列号从 1 开始逐一递增

2.4 协议体说明

协议体里面承载了消息的具体内容，其具体格式有几种可能形式，并且可扩展，在下一节中具体讨论。

3 通用协议中内容格式的设计

3.1 XML 文件格式

随着 XML(Extensible Markup Language)格式及其各种解析器的发展[2]，协议的消息体以 XML 格式为载体的传送成为一种很理想的形式[3]。

XML 的内容体里面可以根据不同业务需求灵活定制，接收方可以根据自己需要读出 XML 内容后按照系统的需要解出相应部分。数据的存储和表现形式完全分开，完全体现了 XML 文件的优点，也使协议体的解码非常简单。

例如，当多个系统进行配置同步的时候，消息体中可以直接传送配置文件的内容。

这种方法在消息解析时非常灵活、简单。当然，它也有其缺点，例如，XML 格式的内容的生成对格式要求很高，同时，XML 文件中各字段的定义、XML 文件格式的设计，也可能会增加额外工作量。但在特定的场合，比如上面的配置同步，还是比较适用的。

这种方法适合多个系统间信息的同步，如集群软件间配置信息的同步，当有第三方保证可以生成合格的 XML 格式的内容的时候，也可适用此方法。

3.2 TLV 格式

TLV 指 Tag, Length, Value[4,5]，即[名称，长度，值]。在消息体中，Tag 字段被定义为 4 字节的无符号整数，Length 定义为 4 字节的无符号整数，value 则定义为各种类型值对应的字符串表示。解码时通过某一需要字段对应的 Tag 及其后长度即可得到其值。

此格式既减少了传递的信息量，又可以使协议的消息体中各字段值不必有序，大大减少了及编、解码的复杂度。

但协议双方必须事先将所有可能出现的 Tag 值协商好，各自编解码时严格对照此表。而每次出现新的协议字段的时候，tag 值映射表需要相应更新，会带来一定复杂度，此时可用下面方法来增加 Tag 制定的灵活性：

所有协议字段名称双方事先约定，而 Tag 值与名称的含义对照则通过 3.1 的方式在连接建立前进行确定，并定期更新，然后传送时按照对方的映射关系将 Tag 翻译成对应的字段。一种极端情况是每次发送前都进行此操作。

3.3 TLV 格式的变体

针对上面 Tag 是整型，使用起来很不形象很不方便的缺点，这里提出两种 TLV 格式的变种：

第一种，N=V。即在消息体中，传输 “字段名”=“值”的方式，不同字段间用分号格开。消息体中同样可以没有顺序，字段名可以使用双方都比较熟悉的名称，而值则表示为值的具体字符串表示。在处理消息的时候与实际逻辑相关，尤其方便。在消息出错的

时候也便于跟踪和修改，避免了 TLV 中整型的 tag 值事先需要约定的缺点。

当然，这种方式也有其缺点：加大了流量，减慢了速度。字符串的匹配速度，一般情况下都会比整型要慢，信息长度也会增大。

第二种：LNLV 格式，即 (name_length, name,value_length, value)。这种情况下，名称还是按照一种字符串传递，因为前面加上了名称的长度，长度可以为任意值。

它可以针对任何形式、任何长度的名称，即使名称中包含双引号等，更加灵活，但同时也存在着上面的缺点：字符串形式的传递，加大了流量，减慢了速度。

3.4 内容格式实际应用

鉴于以上各种内容格式都有其优缺点，结合系统间业务的实际特点，建立在不同情况下，应用不同的内容格式。而利用协议同步头中的 content fmt 字段来区别各格式。

例如，将 content fmt 字段进一步定义为：

表 2 通用消息头中 content fmt 具体值定义

值	含义
0	其他
1	XML 格式
2	TLV 格式
3	TLV 变体 1 “name”=”value”格式
4	TLV 变体 2 [name_length, name,value_length, value]格式

4 通用协议的应用

在我们的项目前，旧有平台与其外部实现有的接口包括 CMPP、SMPP、SGIP 以及其内部的各种协议。而我们的项目，在支持旧有平台的各种协议外，还支持了上述的通用协议，此时通用协议采用 3.2 的 TLV 格式做为内容格式。

在实际应用前，我们先解决了各不同协议的信息在平台中的内容存储格式问题，如下：

4.1 内部承载 PDU 的设计

在我们平台内部，所有短信相关信息都以内部承载 PDU 形式存储。内部承载 PDU 的结构以通用协议为基础，其内部各成员为通用协议头的各部分内容，加上与路由相关的信息的源、目的号码以及消息内容，而与路由无关的其他协议字段以 TAG 为主键，以 string 类型存储在其 map 中单独存储。

TAG 各值在定义时平台内部统一，包含了所有协议的所有字段，并且在编号时为各协议留下了可扩展的余地。

这种协议无关的内部承载 PDU 的设计，使信息在平台其他部分传递时与接口的协议实现了解藕，使其在业务相关处理部分完全脱离协议的限制。

由内部承载 PDU 各字段的定义和特点可知，内部 PDU 就是通用协议在平台中的体现。

4.2 内部承载 PDU 与通用协议之间的转换

通用协议在编码时，分为两部分：协议头及协议体。协议头中各部分与 PDU 中各成员一一对应，直接提取即可。而除去路由部分外，内容部分通过遍历存储 map,逐一加入各字段，字段长度可通过 string 长度确定。

解码时，协议头中各部分通过解析直接存入承载 PDU 中各成员中，而在解析内容时，为编码时的逆过程：对各[T,L,V]三元组，Tag 和 Length 长度固定，而 Value 的长度通过其之前的 Length 值即可确定，然后根据其 Tag 将对应值存入 map 中，各字段的顺序可以任意排列而不影响承载 PDU 中各字段的值。

4.3 内部承载 PDU 与其他协议之间的转换

由于 Tag 值是按照平台支持的所有协议需要制定，其与各协议的各字段都有一个映射关系。

在编码时，目的协议从内部承载 PDU 的 map 中按照对应的各位置需要的值的 TAG 取出其值，转换成需要的类型，然后编进码流中;解码时将各个位置的值取出，按其对应的 TAG 存进内部 PDU 中。

仅管这些大多为字段顺序固定、长度特定[不一定都相同]，需要完全理解协议的协议，但无论其编码还是解码，以内部承载 PDU 为信息载体，对编、解码的准确性没有任何影响。同时，这种承载 PDU 还可以做为不同协议之间转换的媒介，这会在 4.4 中介绍。

4.4 通用协议在实际系统中的应用

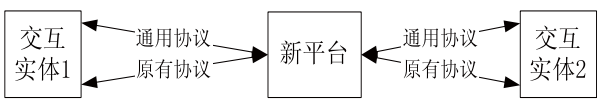

图 2 平台与交互实体连接图

实际应用中，与平台连接的实体可能有很多个，如上图所示。而其与各交互实体可能的通信协议也不一定只有一种。因此，平台以内部承载 PDU 为信息的载体，同时支持了通过协议及平台与旧有接口需要的各种协议。这样，对旧有的各实体，平台与其交互协议保持不变。

而对各新的接口，平台在协议制定上如下：

当平台只需要与一个实体交互，且之间接口需要新开发时，我们采用本文中提出的通用协议做为交互协议，则只需用到 4.2 中内部承载 PDU 与通用协议之间的转换过程，无论是平台还是新的实体在这方面的实现上都非常简单。

当平台原来已经和其他实体有了交互后，又扩展出新的交互时，则可以在新的接口上采用通用协议，与原有接口不变，平台内部采用 4.1 中内部承载 PDU 做为信息载体，如上 4.2 与 4.3 的过程都会被用到。而且接口之间互相没有影响。

当平台同时与原有实体和新的实体交互，同时还涉及到通过平台进行原有实体和新的实体之间的信息交互时，则原有实体接口不变，与新的实体通过通用协议进行通信。从原实体 1 到目的实体 2 中的协议转换通过 4.3 中的解码与 4.2 中的编码组合实现，从新的实体 2 到原实体 1 之间的协议转换则通过 4.3 中的解码与 4.3 中的编码组合实现。按照这种原理，平台还可以实现任何已支持的旧有协议之间的转换。在这里，平台以内部承载 PDU 为载体，实现了不同实体协议间的转换。

综上可见，协议无关的通信协议的引入及其在平台内部的表现形式内部承载 PDU，不但使平台内部信息的载体与协议完全解除耦合，使其业务处理与编解码部分相分离，也使平台更容易扩展出与其他实体的交互而不影响已有各接口。更能在平台原有功能上，扩展出协议转换等更多功能。

5 结束语

在系统信息交互日趋频繁的今天，一种与业务无关的消息交协议的设计及其编解码的实现，可以大大减少新业务开发时交互协议的制定及实现过程，使业务开发人员的注意力可以集中在业务相关领域上。既缩短了新业务系统的开发时间，节省开发成本，又可减少相关的错误的出现。实践证明了其可行性。

参考文献

1 冯怀迪,周亚军. 通信协议设计与实现.机电工程,2009, 26(11):91－93.

2 丁跃潮,张涛. XML 实用教程.北京:北京大学出版社, 2008.

3 张奇支,朱晓民,廖建新. XML 在短消息协议中的应用.中国无线电,2004,(2):20－24 .

4 Zhang YT, Liao JX, Zhang TY, Zhu XM. A novel method for the short message or multimedia message synchronization. Second International Conference on Wireless and Mobile Communications. 2006.10.

5 王沁,许娜,张燕,张晓彤.优化 TLV 编码规则.计算机科学, 2008,35(11):104－107.