云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« 抵抗组织：阿瓦隆及兰斯洛特扩充 | 返回首页 | 上次提到的阿瓦隆辅助工具 »

skynet 近期更新及 sproto 若干 bug 的修复

skynet 的 1.0 版已经发布了 3 个 alpha 版，等稳定以后将发布 beta 版本。

最近的问题主要集中在一些我们在老项目中没有使用到的特性上面。尤其是 sproto 这个模块，我希望它将来作为 skynet 推荐的通讯协议，但我们老的项目开始的比 sproto 的项目早，所以早期项目全部使用的是 google protocol buffers （以及我自己做的实现）。 随着新项目的开展，我们公司内部开始大面积使用 sproto ， 也就发现了一些 bug ，在最近集中修复。

由于 skynet 使用多 lua VM 结构，为了不在每个 VM 里重复加载 sproto 协议，最近增加了 sproto 协议对象的共享。这个作为未写入 sproto 文档的特性提供，当然也不影响 sproto 在其它领域的使用。不过加这个特性比较匆忙，第一次提交时在 gc 方面遗留了一个 bug ， 有可能导致多个 VM 重复释放 C 对象，问题已经在仓库最新的提交中修复。

另一个为了 skynet 的应用而特别加上的特性是让 sproto 的 decode 可以接收指针（lightuserdata）。固然只接受 string 会让实现更稳固一些，不过在 skynet 里很多地方 string 和 lightuserdata + size 是通用的，所以就顺带支持了。这样可以减少一次内存拷贝。

根据使用的同学的需求，在 sproto 的 lua bingding 里增加了更为详细的出错提示，这可以帮助实际使用时的错误定位。另外，还增加更为严格的类型检查。缺少这些检查应该算是 bug ， 因为使用 sproto 而不是 json 这种的无格式的协议，就是为了可以多做一些类型检查的。复杂类型（在 lua 里用 table 实现）不检查还会导致进程挂掉，这是绝对不可以接受的。

最后一个严重的 bug 是设计上的。

sproto 的 encode C API 采用的是 callback 的方式。由使用者（通常是其它语言的 binding ）提供一个 callback 函数，C 核心根据 sproto 协议，每个字段调用一次这个函数。

如果它返回 -1 表示编码错误（一般是 buffer 不够大）， 会让 C 核心的编码过程错误返回。

如果它返回 0 表示这个字段不存在。这是因为 sproto 是允许字段不存在的，不存在的字段不会被编码进最终的串。另外，对数组的编码也依赖它。如果在编码一个数组时返回 0 ， 表示数组结束。

其它情况应返回一个正数，表示当前需要编码的对象的长度。

对于简单类型，如 boolean ， integer ， 一般返回的是固定值。boolean 返回 4 ， integer 返回 4 或 8 （提示 C 核心这个整数是 32bit 还是 64bit 的）。最终编码不一定按这个数字来，且 callback 函数得到的写入地址也并非最终 buffer 的地址。C 核心会提供一个地址对齐的地址，然后根据 sproto 的编码协议来转换到最终 buffer 中，同时还要处理大小端问题。

对于不定长类型，如 string 或自定义类型。这个长度会帮助 C 核心了解应该将 buffer 指针后移多少字节。callback 函数将直接把数据写入最终的 buffer 。而问题就出在这里。

当 string 是一个空串时，由于空串的长度为 0 ， 会让 C 核心误会这个字段并不存在，这导致所有的空串无法编码。更严重的是，如果是字符串数组，碰到空串就会停止编码这个数组。最终的修补方案是，约定在编码 string 的时候，应该返回字符串长度 + 1 。这属于一个设计问题，所以除了 lua binding 之外，别的语言的 binding 也需要修改。好在目前已知的 python binding 也是我们公司的同学实现的，应该马上能改过来。

对于用户类型，没有 string 这个问题。即使是空的对象，也有一个数据头。所以不可能为 0 。对于空对象，不在数组中时，目前的 lua binding 会返回 0 ， 让 C 核心跳过这个字段，而在数组中时，则会返回一个空的数据头。

利用 sproto 实现的 sharemap 也被查出一个 bug ， 不过这个 bug 不属于 sproto 。它的 metatable 被不小心循环引用了，如果一个字段不存在会导致 lua 检测出 metatable 循环引用而出错。

还有一个问题是在使用 httpc 时发现的，虽然已经知道，但因为用的不多也没有特别在意。这次在正式版发布前，还是给出解决方案：

skynet 的 socket 层在处理域名的时候直接调用了系统 api getaddrinfo ，这会阻塞住线程。由于 skynet 的 socket 是单线程的，所以一旦做域名查询，会导致 skynet 所有的 socket 消息处理阻塞。一般我们不会使用域名，即使用，也是在数据库第一次连接的时候，通常发生在 skynet 进程启动的时候，所以影响不大。但一旦使用 httpc 模块，就很容易向外连接一个域名了。

由于系统并不提供异步的域名解析方法，很多其它网络库的做法是使用额外的线程去查询域名。我并不想针对这个需求而大幅度修改已经稳定了的 skynet socket 层，所以提供了独立的解决方案：那就是在上层自己使用 dns 协议发送 udp 包查询。为了让 httpc 模块可以使用它，对其也做了一点改变，允许用户连接一个 IP 地址，而自己在 http header 里填写 host 字段。

另外，在充当 http 客户端时，http 服务器往往会在返回的 header 中填写多个 Set-Cookie 字段，之前对同名的 header 中字段没有正确的处理，现在做了修正。（多个同名字段会生成一个 table ）

云风 提交于 April 7, 2015 10:11 AM | 固定链接

POST A COMMENT

非这个主题相关的留言请到：留言本

名字：

Email 地址：

为了验证您是人类，请将六加一的结果（阿拉伯数字七）填写在下面：

URL:

记住我的信息?

留言：
（不欢迎在留言中粘贴程序代码）

提交