Dragonite Protocol Proposal 002 (zh-CN)

Dragonite 协议提案 002 简体中文

2017-06-27

## 简介

Dragonite是一个基于UDP的快速可靠数据传输协议。UDP自身是一种无连接，提供简单不可靠信息传送服务的协议，Dragonite在UDP的基础上主要实现了数据分组拆分、丢包重传、保证按序到达、速率控制等便于进行可靠数据传输的特性。Dragonite通过对UDP传输的封装能够提供类似TCP协议的功能与类似接口，但在重传机制、速率控制、连接管理上的内部实现有着不同的思路与方式。Dragonite旨在提供高度可定制的工作参数，在牺牲少量易用性的情况下在高延迟/高丢包等特殊网络中提供（相比TCP协议）更加稳定与快速的传输性能。

002在001的基础上对部分用词、含有歧义的变量名称进行了修改，同时引入了新的基于ACK计算延迟波动，辅助快速重发的机制。（由于未对协议数据包结构与基本逻辑行为进行变动，协议版本号仍未1，可兼容旧版程序）

## 概览

为了在无连接的UDP协议中实现有状态的连接机制，发起方每个连接使用一个独立的UDP端口，接收方可根据发起方IP:PORT的组合区分不同连接。监听端绑定一个固定UDP端口用于监听客户端发起的请求。Dragonite协议传输的基本单位为数据包，每个数据包有着不同类型与相应作用，例如携带内容，表示关闭连接，ACK等。在后文中这些数据包统称称为**消息**。

## 一些定义

Dragonite中的所有多字节数字均为**网络字节序（大端序）**

此文档中描述的Dragonite协议**版本号为1**

由于网络传输特点，Dragonite协议拥有可配置的MTU参数用于限制每个消息的最大包长度，实现中建议默认为1300-1400。**自动分割(auto-splitting)是****一个能个够将用户通过接口一次性发送的过大数据自动拆分为多个长度小于MTU消息进行发送的机制**，但这会导致在调用接收时无法保证一次调用即可接收到完整数据。实现应当允许用户关闭该特性，在关闭后若用户通过发送接口一次性发送超过MTU长度的数据应当**拒绝执行并返回错误信息**。

## 消息

每个通过Dragonite发送的有效UDP包均为一个消息。任何有效消息均有以下结构：

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| PAYLOAD  内容 | |

所有消息分为两大类——保证到达类与不保证到达类。

**保证到达类消息：由Dragonite协议机制在连接存活的情况下保证按序送达**

**不保证到达消息：类似未经封装的UDP包，不保证送达与顺序正常**

保证到达类消息使用一个**消息序号(sequence)**保证重传与排序等功能正常运行。该序号从连接建立后由0开始递增。发送方每发送一个保证到达类消息将sequence值加一。Sequence为4 bytes整型数字。因此保证到达类消息结构如下：

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| SEQUENCE 消息序号  **4 bytes** | |
| PAYLOAD  内容 | |

**一个此前未建立连接的目标地址向接收端发送一个Sequence=0的保证到达类消息后即视为申请建立连接。**

### 内容消息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型编号 | 保证到达类 | 长度 | 作用 |
| 0 | 是 | 不固定 | 携带内容数据 |

内容消息是用于携带用户发送内容数据的消息类型。一般而言是一条Dragonite连接中出现最频繁的消息类型。

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| SEQUENCE 消息序号  **4 bytes** | |
| DATA LENGTH 内容长度  **2 bytes** | |
| DATA  内容 | |

### 关闭消息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型编号 | 保证到达类 | 长度 | 作用 |
| 1 | 是 | 8 | 声明连接关闭 |

关闭消息用于声明一条连接的关闭。

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| SEQUENCE 消息序号  **4 bytes** | |
| STATUS CODE 状态代码  **2 bytes** | |

状态代码用于表示连接关闭的原因。正常情况下设置为0，如关闭是由于发生错误等特殊原因则可设为一个非0的错误代码。

### ACK消息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型编号 | 保证到达类 | 长度 | 作用 |
| 2 | 否 | 不固定 | 声明收到消息 |

ACK消息用于接收端向发送端告知收到某个（或多个）保证到达类的消息。ACK消息可以包含一个或多个Sequence编号，批量表示一段时间内收到的多个消息。因此为了节省流量、提高执行效率，建议Dragonite的实现不要每收到一个消息均返回一个ACK，而是以一定间隔（**建议5-20ms之间**）发送包含多个Sequence编号的ACK消息。

**注意由于ACK消息长度不固定，理论上在接收速率极高的情况下需要发送的ACK消息可能超过MTU大小，此时需要根据设置的MTU拆分为多个ACK消息！**

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| CONSUMED SEQUENCE 上层应用已读走的数据的最大编号  **4 bytes** | |
| SEQUENCE COUNT 编号个数  **2 bytes** | |
| SEQUENCE 1 消息序号1  **4 bytes** | |
| SEQUENCE 2 消息序号2  **4 bytes** | |
| SEQUENCE N 消息序号N  **4 bytes** | |
| …… | |

### 心跳消息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型编号 | 保证到达类 | 长度 | 作用 |
| 3 | 是 | 6 | 保持NAT映射  检测连接存活 |

心跳是连接建立后每隔一段时间都会发送的一个用于保持NAT与检测连接是否存活的消息。建议间隔时间**2-20s**

|  |  |
| --- | --- |
| VERSION 协议版本  **1 byte** | MSG TYPE 消息类型  **1 byte** |
| SEQUENCE 消息序号  **4 bytes** | |

接收方在收到心跳消息后按照正常其他处理保证到达类消息的方式返回ACK即可。发送方根据是否在超时时间内收到接收方的ACK判断连接是否仍然存活。

## 关于重传

连接双方可以根据己方一个可靠消息发出后多长时间收到对方对应的ACK消息来判断线路延迟大小。**在一般情况下，应只利用没有经过重发的数据包的ACK计算延迟，以避免错误的判断出过高延迟。**延迟变量RTT的计算应符合以实现渐进式的变化，减小偶然性网络波动的影响。延迟波动变量devRTT的计算为

重发机制在进行重发前等待的时间应为RTT+devRTT，注意当devRTT小于一次批量ACK的时间间隔时，应以ACK间隔为准，以避免误因远端ACK的延迟发送而提前判断出数据包已经丢失。

**由于在特殊网络环境下可能存在延迟突然大幅增长的情况，会导致所有数据包若根据此前RTT进行重发判断，则至少重发一次，无法再仅依据没有经过重发的数据包刷新RTT数值。因此应加入延迟矫正机制，即若两秒之内收到的所有ACK消息均来自重发过的消息，则用此经过重发的消息计算出的延迟进行一次RTT更新。加入此机制后，可保证在两秒或数个两秒后将RTT纠正到实际大小。**

## 参数概念

一个标准的Dragonite协议实现应允许用户配置以下内部参数。

1. **发送速率**-为每个连接调整发送数据的最大速率，应允许调用者在连接过程中**随时调整**。
2. **MTU**-每个消息的最大包长度。建议默认1300-1400。**连接双方必须有相同MTU大小**，以免其中一方接收的数据缓存越界。
3. **自动分割**-一个能个够将用户通过接口一次性发送的过大数据自动拆分为多个长度小于MTU消息进行发送的机制，但这会导致在调用接收时无法保证一次调用即可接收到完整数据。**实现应当允许用户关闭该特性，在关闭后若用户通过发送接口一次性发送超过MTU长度的数据应当拒绝执行并返回错误信息。**
4. **接收缓存大小**-为防止一端发送速率过高超出接收方处理能力而被缓存的数据包不断占用内存，可设置最大接收缓存大小，最多允许在内存中缓存多少个待处理的数据包。**注意服务端默认值通常应大于客户端数倍**，因服务端通常需要同时与多个客户端连接。
5. **发送窗口倍数**-在收到对应ACK之前发送方最多可发送多少消息。注意发送窗口本身必须由计算得出（其中为根据发送速率与MTU计算出的每秒包数量，为秒单位的与目标网络延迟）而不得允许自行设置。**这个值是在延迟完全稳定而网络完全不丢包的理想情况下正好达到目标速率所需要的最小窗口大小。但由于实际的网络情况，此值需要进行一定程度倍数放大。**而倍数分为aggressive激进与passive保守两个值——**前者根据目前从ACK获得的目标接受到最大Sequence判断是否可以继续发送，后者通过目前对方收到的最后一个连续Sequence（即若收到1/2/3/6/7，则最大连续Sequence为3而非7）判断。**经过测试，倍数上推荐aggressive设置为5，passive设置为2。
6. **ACK发送频率**-多久发送一次批量ACK消息。推荐默认为5-20ms之间。
7. **快速重传次数**-在发生偶然丢包后，默认重传等待时间为上文所述计算方法，**此阶段被称为快速重传**。但若一个消息发生多次丢失，**可认为网络出现临时中断，应逐渐成倍提高等待时间至4倍（或更大），此时为慢速重传阶段。**此选项为一个消息在进入慢速重传前最多被快速重传的次数，推荐默认为5。
8. **心跳频率**-发送心跳包的频率。推荐为2-20s。
9. **接收超时**-多久没有从对方收到任何消息后判断此连接已经作废。推荐为心跳包频率的两倍。**至少需要大于心跳频率。**