The YS protocol

Vincent A.
vincentweb31@gmail.com

25 décembre 2011

Table des matières

1 Introduction

When I started hacking with the YS protocol, I knew nothing about sockets, internet protocols, and serialization, but I had great ambitions. The quest was huge for the ignorant knight I was. After years of patience, reading, experimenting, asking questions, sharing code, ... I ended up to get a ”little” idea of the YS protocol. But even with this knowledge, the quest will be huge for you.

2 The YS protocol and its links with TCP

In the OSI model, the YS protocol belongs to the application layer, just above TCP. The choice made by Soji Yamakawa of choosing TCP in the transport layer has the following connections :

- more data is sent since the TCP header is quite big compared to other protocols of the transport layer
- if a packet was lost, TCP waits for the server server to send it again although the following packets were successfully received which leads to phenomenons were you see your opponent flying backward during a network play.
- necessity of separating the YS messages since contrary to the UDP protocol, TCP concatenate all the messages to send in its buffer and send them when the buffer is full enough or old enough. That is why all the YS messages start with an integer giving the size of the message. This issue can be avoided by the use of the TCP PUSH flag.
- the OS implementation of the TCP keep-alive is not mandatory, that is why it is done in the application layer.
- you are certain all you messages were received, however the YS protocol force the client to send a copy of the received message to check it received the same thing, which I think is useless.

3 Prerequisites

3.1 Sockets

3.2 Serialisation

4 The specifications of the YS protocol

4.1 The header

The YS messages have the following shape :

Length (int)	Type (int)	Data

Every YS message start with :

- a length information = data size + type size = data size + 4
- the type of the packet (the purpose of its content)

For example let’s decode the following message :

```
18:00:00:00:01:00:00:00:64:6f:69:6e:67:5f:74:65:73:74:73:00:00:00:00:00:7f:db:32:01
```

18 : 00 : 00 : 00 = (int 24) is the size of the message
01 : 00 : 00 : 00 = (int 1) is the type of the message, 1 means it’s a login message
$64 : 6f : 69 : 6e : 67 : 5f : 74 : 65 : 73 : 74 : 73 : 00 : 00 : 00 : 00 : 00 : 7f : db : 32 : 01$ is the data

4.2 The different types

4.2.1 Login (type=1, 0x1)

CHAR[16]	the user name (the 16^e bit is the null character)
INT	the size

4.2.2 Map (type=4, 0x4)

The client must reply the received message.

CHAR[60]	the name of the map

4.2.3 Entity joined (type=5, 0x5)

The client must reply the received message.

4.2.4 Acknowledgement (type=6, 0x6)

4.2.5 Flight data (type=11, 0xb)

INT	timer which is incremented in an odd way
INT	ID of the pilot flying
SHORT	info1 (5=the lives of the player are coded on 1 char (strength ¡ 256 most of the cases), 3=the lives are coded on a short)
2 OCTETS	unknown (WARNING : these two octets are only present when info1=3)
FLOAT	x position of the aircraft in meters
FLOAT	z altitude of the aircraft in meters (y axis of scenedit)
FLOAT	y position of the aircraft in meters (z axis of scenedit)
SHORT	heading
SHORT	AOA
SHORT	bank
SHORT	xSpeed
SHORT	ySpeed
SHORT	zSpeed
8 OCTETS	unknown
SHORT	fuel
6 OCTETS	unknown
CHAR	spoilerBrake
CHAR	flapsGear
CHAR	afterburnerSmokeTrailsGunfire (convert in binary)
4 OCTETS	unknown
SHORT	gunAmmo
CHAR	rockets
CHAR	unknown
CHAR	AAM
CHAR	AGM
CHAR	bombs
CHAR/SHORT	lives
2 OCTETS	unknown
CHAR	elevator
CHAR	aileron
2 OCTETS	unknown
CHAR	trim

4.2.6 Player left (type=13, 0xd)

4.2.7 Keep-alive (type=17, 0x11)

Empty message which must be sent from time to time to avoid being disconnected by the server.

4.2.8 Object left (type=19, 0x13)

4.2.9 Damages (type=22, 0x16)

int	kind of victim entity (0=the victim is ground object, 1=the victim is a player)
int	victim ID, (you get the ID of an entity with the messages of type 5)
int	kind of killer entity (0=the killer is ground object, 1=the killer is a player)
short	power of the damage If you want to kill an object of strength 3 in one shot, this value must be 3.
short	shot (10=missile/rocket hit its target, 11 gun bullet/bomb hit its target, 12 bomb/rocket explosion (not hit directly))
short	weapon (gun=0, aim9=1, AGM=2, bomb500=3, rocket=4, AIM120=6, bomb250=7, bomb500HD=9, AIM9X=10 ; nothing sent for kamikaze kills !)
4 octects	unknown

4.2.10 YSFlight version (type=29, 0x1d)

4.2.11 Missile allowed option (type=31, 0x1f)

int	missile option (1=missile allowed by the server). This message can be sent to the clients at any moment, allowing a proxy such as YSPS to change options on the fly !

4.2.12 Chat message (type=32, 0x20)

4.2.13 Weather and server options (type=33, 0x21)

The client must reply the received message.

4.2.14 User data (type=37, 0x25)

4.2.15 Weapon allowed option (type=39, 0x27)

The client must reply the received message.

int	weapon option (1=weapons allowed by the server). This message can be sent to the clients at any moment.

4.2.16 Show username option (type=41, 0x29)

The client must reply the received message.

4.2.17 Other server options (type=43, 0x2b)

The client must reply the received message.

4.2.18 Aircraft list (type=44, 0x2c)

This message is use to send to the client the list of the aircraft installed on the server. The client must reply the received packet.

1 octet	unknown
char	Number of aircraft sent
2 octets	unknown
char[][]	The concatenation of the aircraft identifier installed on the server.

4.3 Message order

5 Filling the holes