LOTRO Investigation v0.1

März 7

2012

This document should carry together what kind of information is delivered through the LOTRO UDP packets. Some sample packets are analyzed and explained.

Gathering information for building a private LOTRO server.

Page

Table of Contents

List of Authors

tA...tAmMo

[Please feel free to add your name if you added, changed something or corrected spelling and grammar]

List of Figures

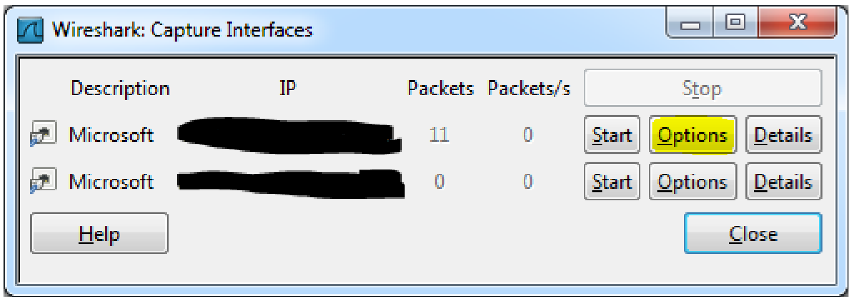
1. Tutorial for packet capturing with Wireshark

1.1 Workflow

Do the following preparations in Wireshark:

(1) Choose interface options

Please select the options for the relevant network interface.



Description	IP	Packets	Packets/s
Microsoft	[redacted]	11	0
Microsoft	[redacted]	0	0

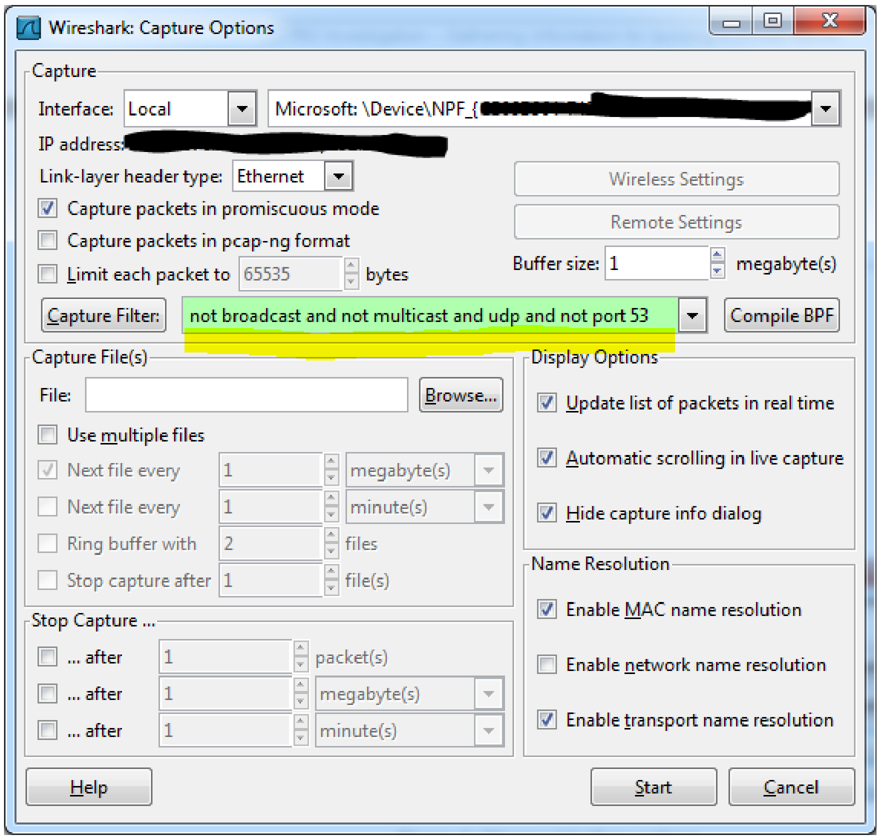
Figure 1: Choose interface options

(2) Apply capture filters

Use the following capture filters:

Not broadcast and not multicast and udp and not port 53

Click start.



Capture
Interface: Local | Microsoft: \Device\NPF_{[redacted]
IP address: [redacted]
Link-layer header type: Ethernet
[✓] Capture packets in promiscuous mode
[ ] Capture packets in pcap-ng format
[ ] Limit each packet to 65535 bytes
Buffer size: 1 megabyte(s)
Capture Filter: not broadcast and not multicast and udp and not port 53

Capture File(s)
File:
[ ] Use multiple files
[✓] Next file every 1 megabyte(s)
[ ] Next file every 1 minute(s)
[ ] Ring buffer with 2 files
[ ] Stop capture after 1 file(s)

Stop Capture ...
[ ] ... after 1 packet(s)
[ ] ... after 1 megabyte(s)
[ ] ... after 1 minute(s)

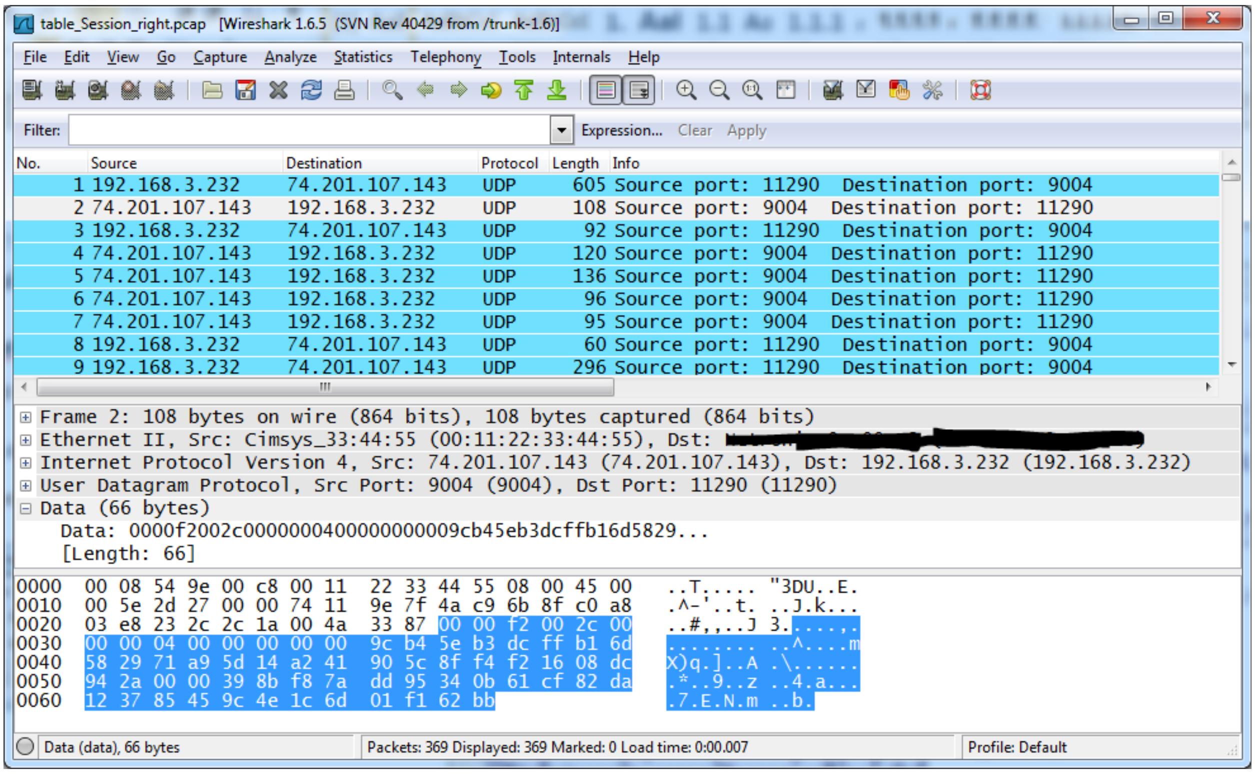
Display Options
[✓] Update list of packets in real time
[✓] Automatic scrolling in live capture
[✓] Hide capture info dialog

Name Resolution
[✓] Enable MAC name resolution
[ ] Enable network name resolution
[✓] Enable transport name resolution

Figure 2: Capture filter

(3) Stop the capture after a while

After some time stop capturing. This should somehow look like this.

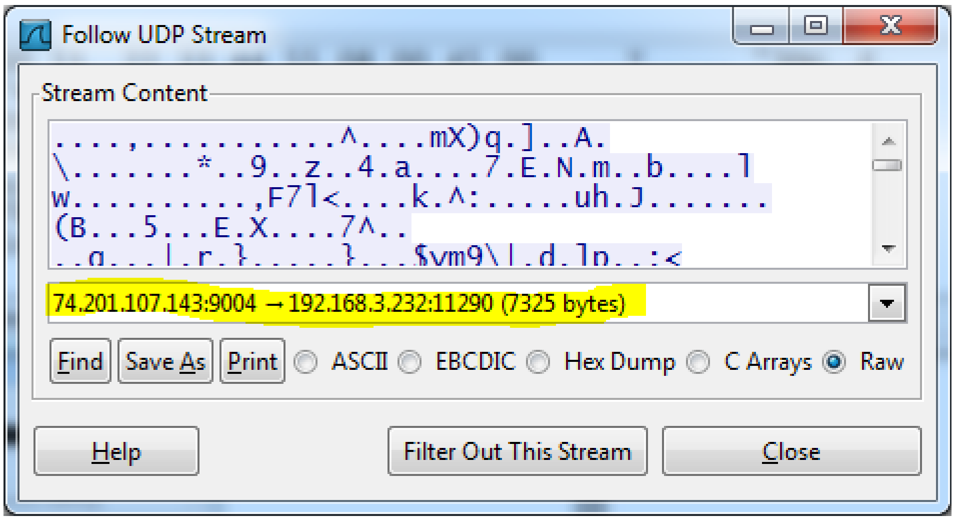

table_Session_right.pcap [Wireshark 1.6.5 (SVN Rev 40429 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
No.  Source          Destination     Protocol Length Info
   1 192.168.3.232   74.201.107.143  UDP       605 Source port: 11290  Destination port: 9004
   2 74.201.107.143  192.168.3.232   UDP       108 Source port: 9004  Destination port: 11290
   3 192.168.3.232   74.201.107.143  UDP        92 Source port: 11290  Destination port: 9004
   4 74.201.107.143  192.168.3.232   UDP       120 Source port: 9004  Destination port: 11290
   5 74.201.107.143  192.168.3.232   UDP       136 Source port: 9004  Destination port: 11290
   6 74.201.107.143  192.168.3.232   UDP        96 Source port: 9004  Destination port: 11290
   7 74.201.107.143  192.168.3.232   UDP        95 Source port: 9004  Destination port: 11290
   8 192.168.3.232   74.201.107.143  UDP        60 Source port: 11290  Destination port: 9004
   9 192.168.3.232   74.201.107.143  UDP       296 Source port: 11290  Destination port: 9004

⊞ Frame 2: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
⊞ Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: [redacted]
⊞ Internet Protocol Version 4, Src: 74.201.107.143 (74.201.107.143), Dst: 192.168.3.232 (192.168.3.232)
⊞ User Datagram Protocol, Src Port: 9004 (9004), Dst Port: 11290 (11290)
⊟ Data (66 bytes)
    Data: 0000f2002c000000040000000009cb45eb3dcffb16d5829...
    [Length: 66]

0000  00 08 54 9e 00 c8 00 11  22 33 44 55 08 00 45 00   ..T..... "3DU..E.
0010  00 5e 2d 27 00 00 74 11  9e 7f 4a c9 6b 8f c0 a8   .^-'..t. ..J.k...
0020  03 e8 23 2c 2c 1a 00 4a  33 87 00 00 f2 00 2c 00   ..#,,..J 3.....,.
0030  00 00 04 00 00 00 00 00  9c b4 5e b3 dc ff b1 6d   ........ ..^....m
0040  58 29 71 a9 5d 14 a2 41  90 5c 8f f4 f2 16 08 dc   X)q.]..A .\......
0050  94 2a 00 00 39 8b f8 7a  dd 95 34 0b 61 cf 82 da   .*..9..z ..4.a...
0060  12 37 85 45 9c 4e 1c 6d  01 f1 62 bb               .7.E.N.m ..b.

Data (data), 66 bytes | Packets: 369 Displayed: 369 Marked: 0 Load time: 0:00.007 | Profile: Default
```

Figure 3: Captured traffic

(4) Rightclick "Follow UDP Stream"

Choose the right direction and click "Save As".



```
....,...............^....mX)q.]..A.
\.......*..9..z..4.a....7.E.N.m..b....]
w...........,F7]<....k.^:......uh.J........
(B...5...E.X....7^..
..g...|.r.}......}...$ym9\|.d.]p..:<
```

74.201.107.143:9004 → 192.168.3.232:11290 (7325 bytes)

Find | Save As | Print | ASCII | EBCDIC | Hex Dump | C Arrays | [✓ Raw]

Help | Filter Out This Stream | Close

Figure 4: Follow UDP stream

(5) Use the "CreatePacketsFromWiresharkHexDump"-Tool to batch extract the packets

2.Client -> Server communication flow

Client sends session key and with this encrypted account and GLS-Ticket to server.

Server does something.

Client writes his okay for packet.

Server sends authorization and character data.

When user joins world server sends ip and port of world server.

Client makes new connection to other server where and is inside middle earth.

3. Packet analyze

3.1 Client packets

3.1.1 1st client packet (no encryption)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  00 00 00 00 EF 00 00 00 01 00 00 00 00 00 C0 AF  ....ï.........À¯
00000010  88 0E 00 00 00 00 3F 30 36 31 30 30 34 5F 6E 65  ˆ.....?061004_ne
00000020  74 76 65 72 3A 37 31 37 34 3B 20 64 69 64 76 65  tver:7174; didve
00000030  72 3A 58 58 58 58 58 58 58 58 2D 58 58 58 58 2D  r:XXXXXXXX-XXXX-
00000040  58 58 58 58 2D 58 58 58 58 2D 58 58 58 58 58 58  XXXX-XXXX-XXXXXX
00000050  58 58 58 58 58 58 AB 00 00 00 01 00 00 00 01 00  XXXXXX«.........
00000060  00 00 35 02 48 4F 8C 00 00 00 01 02 00 00 01 68  ..5.HOŒ........h
00000070  00 00 00 A4 00 00 35 1B 86 BA 08 27 A4 33 DC CB  ...¤..5.†º.'¤3ÜË
00000080  06 53 F0 AA 8F 6A 56 97 DC 35 54 0B 57 AD 9F 2A  .Sðª.jV—Ü5T.W.Ÿ*
00000090  6B 58 F6 98 89 19 7E 53 41 4F 75 D8 DB E3 97 10  kXö˜‰.~SAOuØÛã—.
000000A0  87 EC 2D 8C 03 3B 1D D9 FF 18 56 C4 6D AB 92 9D  ‡ì-Œ.;.Ùÿ.VÄm«’.
000000B0  8F 8E 2D 95 4E 32 0A 13 EA 5B 14 35 81 F4 E7 72  .Ž-•N2..ê[.5.ôçr
000000C0  66 45 0E 4A 07 C5 53 B5 DA E3 13 8F 62 4D FB A1  fE.J.ÅSµÚã..bMû¡
000000D0  F3 CF 05 BB 17 AF 4F 31 71 DD 65 97 91 9A F4 8B  óÏ.».¯O1qÝe—‘šô‹
000000E0  56 B9 F7 8B 99 B6 1A 30 E8 93 C5 26 7F 8A 4F F9  V¹÷‹™¶.0è“Å&.ŠOù
000000F0  00 8B 9F 40 44 23 0B 00 00 00 AB CD 75 16 8D 7D  .‹Ÿ@D#....«Íu..}
00000100  B0 7B B7 5A E1                                   °{·Zá
```

Size of Data-Block (starts at offset 00000016h == 3F)

Identifier for server what kind of packet he is getting (???)

Packet sequence number (in other packets yes, maybe not in the first)

First part maybe random, second part increasing ticks (???)

Length of Client-ID String, here starts the Data-Block

Client-ID String

Length of remaining data inside the packet

Date & Time when packet was generated [24.02.2012 21:33:41]

Length of Simple Key Blob [Blob Header + Session Key Block]

Blob Header

Session Key Block

Length of Account name + GLS Ticket Block

Account name + GLS Ticket Block

Figure 5: First client packet

Investigations:

- The first part of the Client-ID String changes after Software update.
- Date & Time as 4 Byte unsigned Integer value is the time in seconds since 01.01.1970 (reverse it to 0x4F480235 = 1330119221 seconds. Add the seconds to 01.01.1970 and you get 24.02.2012 21:33:41).

- The Session Key Block contains the 128 Bit **Session Key**. The Session Key Block is encrypted with the Clients Public Key. The Public Key size is 1024 Bit and he is known. The Session Key Block can be decrypted with the Servers **Private Key** to gain the **Session Key**. The **Private Key** is only known at the Game Server and can't be brute forced. It would take ages. The Account name + GLS Ticket Block is **RC4** encrypted with the **Session Key**. In this Account name + GLS Ticket Block only the Unicode string of fictional account name "tammo" is encrypted (Unicode: 2 bytes per character plus the leading length specification of the string = 11 Bytes or 0x0B in Hex).

The other values in the first client packet are unknown. Maybe at offset 000006h to 000009h there are the identification bytes, what kind of action the packet does at the server side.
I think that the 4 Byte words, e.g. for the packet sequence number could only be 2 bytes long, the other two zero values could be used as separator of the segments (inside a data-block).
[tA]

3.1.2 2nd client packet (no encryption)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  00 00 94 2A 1C 00 00 00 08 00 00 00 00 00 19 2D  .."*...........-
00000010  5A 21 01 00 B1 6D 90 5C 8F F4 F2 16 08 DC 1F 6A  Z!..±m.\.ôò..Ü.j
00000020  20 1F 2B C7 82 47 AB 64 25 39 FD 54 17 88 33 33   .+Ç,G«d%9ýT.ˆ33
00000030  00 00                                            ..

Start bytes / identifier for this client session
Length of Data-Block
Identifier for client what kind of packet he is getting (???)
Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)
First 4 bytes some kind of checksum, next 4 bytes could be ticks because
they increase by time (???)
Data-Block
```

Figure 6: Second client packet

Investigations:

- No.

[tA]

3.1.3 Ping packet (unencrypted 3rd client packet)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  94 2A 06 00 02 00 00 08 02 00 00 00 24 C7 57 7C  ”*..........$ÇW|
00000010  03 00 B1 6D 00 00 00 00 DC FF                    ..±m....Üÿ

Start bytes / identifier for this client session
Length of Data-Block
Identifier for client what kind of packet he is getting (???)
Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)
First 4 bytes some kind of checksum, next 4 bytes could be ticks because
they increase by time (???)
Data-Block
```

Figure 7: Client ping packet

Investigations:

- This seems to be a ping packet. There exists more versions with other indentifiers.

[tA]

3.2 Server packets

3.2.1 1st server packet (no encryption)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  00 00 F2 00 2C 00 00 00 04 00 00 00 00 00 9C B4  ..ò.,.........œ´
00000010  5E B3 DC FF B1 6D 58 29 71 A9 5D 14 A2 41 90 5C  ^³Üÿ±mX)q©].¢A.\
00000020  8F F4 F2 16 08 DC 94 2A 00 00 39 8B F8 7A DD 95  .ôò..Ü”*..9‹øzÝ•
00000030  34 0B 61 CF 82 DA 12 37 85 45 9C 4E 1C 6D 01 F1  4.aÏ‚Ú.7…EœN.m.ñ
00000040  62 BB                                            b»

Start bytes / identifier for this server session
Length of Data-Block
Identifier for client what kind of packet he is getting (???)
Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)
First 4 bytes some kind of checksum, next 4 bytes could be ticks because
they increase by time (???)
Data-Block

[Inside the Data-Block it seems to be that a double zero (0x00 0x00)
separates the segments]
```

Figure 8: First server packet

Investigations:

- The first server packet is unencrypted. If a "join-world" server connects to the client, a somehow similar packet is received.

[tA]

3.2.2 2nd server packet (unencrypted)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  F2 00 5F 00 06 00 00 01 02 00 00 00 7E DE 1F AF  ò._.........~Þ.¯
00000010  DD FF B1 6D CA 1B D9 A9 5D 14 A2 41 01 02 02 04  Ýÿ±mÊ.Ù©].¢A....
00000020  00 00 00 00 00 06 81 49 24 00 00 08 2D 01 0D 58  .......I$...-..X
00000030  00 58 00 2E 00 58 00 58 00 58 00 2E 00 58 00 30  .X...X.X.X...X.X
00000040  00 2E 00 58 00 58 00 58 00 01 00 00 00 00 01 02  ...X.X.X........
00000050  02 04 00 00 00 00 00 06 80 AA 17 00 00 08 00 00  ........€ª......
00000060  00 00 00 00 00 00 9E 81 E5 0A FE 01 00 25 01 00  ......ž.å.þ..%..
00000070  00 00 00                                         ...

Start bytes / identifier for this server session
Length of Data-Block
Identifier for client what kind of packet he is getting (???)
Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)
First 4 bytes some kind of checksum, next 4 bytes could be ticks because
they increase by time (???)
Data-Block
```

[Changed my IP-Address inside the packet to XX.XXX.XX.XXX]

Figure 9: Second server packet

Investigations:

- Inside this packet you will find your IP-Address
- The other values are unknown to me

[tA]

3.2.3 3rd server packet (unencrypted)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  F2 00 87 00 06 00 00 00 03 00 00 00 C7 41 D8 0E  ò.‡.........ÇAØ.
00000010  DD FF B1 6D 01 02 02 04 00 00 00 00 00 06 81 49  Ýÿ±m...........I
00000020  19 00 00 01 D6 80 9A 10 00 00 00 00 00 00 00 00  ....Ö€š.........
00000030  00 00 00 00 00 00 00 00 00 00 02 01 02 02 04 00  ................
00000040  00 00 00 00 06 81 49 20 00 00 03 F7 01 0B 74 00  ......I ...÷..X.
00000050  6F 00 78 00 78 00 69 00 70 00 6C 00 61 00 73 00  X.X.X.X.X.X.X.X.
00000060  6D 00 61 00 01 00 00 00 00 01 02 02 04 00 00 00  X.X.............
00000070  00 00 06 81 49 05 00 00 78 66 00 01 01 02 02 04  ....I...xf......
00000080  00 00 00 00 00 06 81 49 11 00 00 06 6B 01 00 00  .......I....k...
00000090  00 06 80 02 01 A6 02 06 2F 00 02                 ..€..¦../..
```

Start bytes / identifier for this server session

Length of Data-Block

Identifier for client what kind of packet he is getting (???)

Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)

First 4 bytes some kind of checksum, next 4 bytes could be ticks because they increase by time (???)

Data-Block

[Changed my Login-Account inside the packet to XXX]

Figure 10: Third server packet

Investigations:

- Inside this packet you will find your Login-Account
- The other values are unknown to me

[tA]

3.2.4 Other packets were unencrypted too

3.2.5 Server pong packet (unencrypted)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  F2 00 0A 00 02 40 00 08 12 00 00 00 88 BB 79 C9  ò....@......ˆ»yÉ
00000010  E0 FF B1 6D 09 00 00 00 B2 08 00 00 03 00        àÿ±m....².....

Start bytes / identifier for this server session
Length of Data-Block
Identifier for client what kind of packet he is getting (???)
Packet number (increases, next server packet got value 0x02 0x00 0x00 0x00)
First 4 bytes some kind of checksum, next 4 bytes could be ticks because
they increase by time (???)
Data-Block
```

Figure 11: Server pong packet

Investigations:

- This seems to be a server pong packet, or a confirm packet
- The other values are unknown to me

[tA]

List of Links

Document Attachments

Source code for:

- Packet extraction from Wireshark stream
- Client packet de- and encryption
- Server packet de- and encryption
- Simple server-test program for your own logged packets(no multithread, not thread save, not performance optimized)

Please feel free to mod!