MySQL 内部交互协议分析（客户端到服务端的通讯协议）

1 典型的 MySql 会话过程

1.1 描述

一次正常的过程如下：

1） 三次握手建立 tcp 连接

2） 建立 MySql 连接

  a) 服务端往客户端发送握手初始化包(Handshake Initialization Packet)

  b) 客户端往服务端发送验证包(Client Authentication Packet)

  c) 服务端往客户端发送成功包

3） 客户端与服务端之间交互

  a) 客户端往服务端发送命令包（Command Packet）

  b) 服务端往客户端发送回应包（OK Packet, or Error Packet, or Result Set Packet）

4) 断开 MySql 连接

  a) 客户端往服务端发送退出命令包

5) 四次握手断开 tcp 连接

1.2 举例(使用 tcpdump 抓包)

客户端在命令行模式下使用命令：mysql –u root –pdbaudit –h 192.168.86.206 连上数据库抓取的数据包如下：

1.2.1 登陆

1)三次握手建立连接

```
19:00:22.534342 IP 192.168.86.101.59614 > localhost.localdomain.mysql: S
911022238:911022238(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
        0x0000:  4500 0034 043f 4000 4006 0801 c0a8 5665   E..4.?@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 189e 0000 0000   ..V.....6M......
        0x0020:  8002 2000 dbdd 0000 0204 05b4 0103 0302   ................
        0x0030:  0101 0402                                 ....
19:00:22.534390 IP localhost.localdomain.mysql > 192.168.86.101.59614: S
3302432077:3302432077(0) ack 911022239 win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 7>
        0x0000:  4500 0034 0000 4000 4006 0c40 c0a8 56ce   E..4..@.@..@..V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1d4d 364d 189f   ..Ve.......M6M..
        0x0020:  8012 16d0 02d3 0000 0204 05b4 0101 0402   ................
        0x0030:  0103 0307                                 ....
19:00:22.534916 IP 192.168.86.101.59614 > localhost.localdomain.mysql: . ack 1 win 4380
```

```
        0x0000:  4500 0028 0440 4000 4006 080c c0a8 5665   E..(.@@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 189f c4d7 1d4e   ..V.....6M.....N
        0x0020:  5010 111c 4959 0000 0000 0000 0000        P...IY........
```
2)服务端向客户端发送握手初始化包（Handshake Initialization Packet）
```
19:00:22.535632 IP localhost.localdomain.mysql > 192.168.86.101.59614: P 1:79(78) ack 1 win 46
        0x0000:  4508 0076 0d33 4000 4006 fec2 c0a8 56ce   E..v.3@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1d4e 364d 189f   ..Ve.......N6M..
        0x0020:  5018 002e 2eed 0000 4a00 0000 0a35 2e35   P.......J....5.5
        0x0030:  2e32 3100 8200 0000 2f75 2246 7b58 2652   .21...../u"F{X&R
        0x0040:  00ff f708 0200 0f80 1500 0000 0000 0000   ................
        0x0050:  0000 004b 6128 4049 2d46 565d 5366 2900   ...Ka(@I-FV]Sf).
        0x0060:  6d79 7371 6c5f 6e61 7469 7665 5f70 6173   mysql_native_pas
        0x0070:  7377 6f72 6400                             sword.
```
3)客户端向服务端发送包含用户名密码的验证包（Client Authentication Packet）
```
19:00:22.536678 IP 192.168.86.101.59614 > localhost.localdomain.mysql: P 1:63(62) ack 79 win
4360
        0x0000:  4500 0066 0441 4000 4006 07cd c0a8 5665   E..f.A@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 189f c4d7 1d9c   ..V.....6M......
        0x0020:  5018 1108 b2d0 0000 3a00 0001 85a6 0300   P.......:.......
        0x0030:  0000 0001 0800 0000 0000 0000 0000 0000   ................
        0x0040:  0000 0000 0000 0000 0000 0000 726f 6f74   ............root
        0x0050:  0014 ce03 1683 429e cae8 cb93 5435 71f2   ......B.....T5q.
        0x0060:  7439 d842 1922                             t9.B."
```
4)服务端向客户端发送一个空包（普通的 tcp 包，跟 mysql 无关）
```
19:00:22.536748 IP localhost.localdomain.mysql > 192.168.86.101.59614: . ack 63 win 46
        0x0000:  4508 0028 0d34 4000 4006 ff0f c0a8 56ce   E..(.4@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1d9c 364d 18dd   ..Ve........6M..
        0x0020:  5010 002e 59bb 0000                       P...Y...
```
5)服务端向客户端发送一个成功包（OK Packet）
```
19:00:22.536827 IP localhost.localdomain.mysql > 192.168.86.101.59614: P 79:90(11) ack 63 win
46
        0x0000:  4508 0033 0d35 4000 4006 ff03 c0a8 56ce   E..3.5@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1d9c 364d 18dd   ..Ve........6M..
        0x0020:  5018 002e 2eaa 0000 0700 0002 0000 0002   P...............
        0x0030:  0000 00                                    ...
```
6)客户端向服务端发送一个包（跟 mysql 似乎无关，包头不符合协议标准）
```
19:00:22.734205 IP 192.168.86.101.59614 > localhost.localdomain.mysql: . ack 90 win 4357
        0x0000:  4500 0028 0444 4000 4006 0808 c0a8 5665   E..(.D@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 18dd c4d7 1da7   ..V.....6M......
        0x0020:  5010 1105 48d9 0000 0000 0000 0000        P...H.........
```

1.2.2 客户端与服务端之间交互

客户端输入：use mysql
服务端返回：Database changed
1)客户端向服务端发送一个命令包（类型为 COM_QUERY）
```
19:07:56.352167 IP 192.168.86.101.59614 > localhost.localdomain.mysql: P 911022301:911022323(22) ack 3302432167 win 4357
        0x0000:  4500 003e 0450 4000 4006 07e6 c0a8 5665   E..>.P@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 18dd c4d7 1da7   ..V.....6M......
        0x0020:  5018 1105 fe85 0000 1200 0000 0353 454c   P...........SEL
        0x0030:  4543 5420 4441 5441 4241 5345 2829        ECT.DATABASE()
```
2)服务端向客户端发送一个结果包（ResultSet）
一个 ResultSet 包含了多个包，每个包都有自己的包头包体，
下面这个返回数据就包含了五个包(1 个 ResultSet Head Packet + 1 个 Field Packet + 1 个 EOF Packet + 1 个 Row Data Packet + 1 个 EOF Packet)
```
19:07:56.352413 IP localhost.localdomain.mysql > 192.168.86.101.59614: P 1:65(64) ack 22 win 46
        0x0000:  4508 0068 0d36 4000 4006 fecd c0a8 56ce   E..h.6@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1da7 364d 18f3   ..Ve........6M..
        0x0020:  5018 002e 2edf 0000 0100 0001 0120 0000   P..............
        0x0030:  0203 6465 6600 0000 0a44 4154 4142 4153   ..def....DATABAS
        0x0040:  4528 2900 0c08 0022 0000 00fd 0000 1f00   E()...."........
        0x0050:  0005 0000 03fe 0000 0200 0100 0004 fb05   ................
        0x0060:  0000 05fe 0000 0200                       ........
```
3)客户端向服务端发送一个命令包（类型为 COM_INIT_DB）
```
19:07:56.353134 IP 192.168.86.101.59614 > localhost.localdomain.mysql: P 22:32(10) ack 65 win 4341
        0x0000:  4500 0032 0451 4000 4006 07f1 c0a8 5665   E..2.Q@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 18f3 c4d7 1de7   ..V.....6M......
        0x0020:  5018 10f5 5534 0000 0600 0000 026d 7973   P...U4.......mys
        0x0030:  716c                                      ql
```
4)服务端向客户端发送一个成功包（OK Packet)
```
19:07:56.367217 IP localhost.localdomain.mysql > 192.168.86.101.59614: P 65:76(11) ack 32 win 46
        0x0000:  4508 0033 0d37 4000 4006 ff01 c0a8 56ce   E..3.7@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1de7 364d 18fd   ..Ve........6M..
        0x0020:  5018 002e 2eaa 0000 0700 0001 0000 0002   P..............
        0x0030:  0000 00                                   ...
```
5)客户端向服务端发送一个包（跟 mysql 没什么关系，包头为 0000 0000)
```
19:07:56.561717 IP 192.168.86.101.59614 > localhost.localdomain.mysql: . ack 76 win 4339
        0x0000:  4500 0028 0455 4000 4006 07f7 c0a8 5665   E..(.U@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 18fd c4d7 1df2   ..V.....6M......
        0x0020:  5010 10f3 4880 0000 0000 0000 0000        P...H.........
```

客户端输入：show tables
服务端返回：查询结果，当前数据库中所有的表
1)客户端向服务端发送一个命令包（类型为 COM_QUERY)
```
19:22:17.971933 IP 192.168.86.101.59614 > localhost.localdomain.mysql: P 911022333:911022349(16) ack 3302432242 win 4339
        0x0000:  4500 0038 0466 4000 4006 07d6 c0a8 5665  E..8.f@.@.....Ve
        0x0010:  c0a8 56ce e8de 0cea 364d 18fd c4d7 1df2  ..V.....6M......
        0x0020:  5018 10f3 1d24 0000 0c00 0000 0373 686f  P....$.......sho
        0x0030:  7720 7461 626c 6573                      w.tables
```
2)服务端向客户端发送一个普通的 tcp 包
```
19:22:18.011368 IP localhost.localdomain.mysql > 192.168.86.101.59614: . ack 16 win 46
        0x0000:  4508 0028 0d38 4000 4006 ff0b c0a8 56ce  E..(.8@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1df2 364d 190d  ..Ve........6M..
        0x0020:  5010 002e 5935 0000                      P...Y5..
```
3)服务端向客户端发送一个响应结果包（Result Packets）
```
19:22:18.031320 IP localhost.localdomain.mysql > 192.168.86.101.59614: P 1:521(520) ack 16 win 46
        0x0000:  4508 0230 0d39 4000 4006 fd02 c0a8 56ce  E..0.9@.@.....V.
        0x0010:  c0a8 5665 0cea e8de c4d7 1df2 364d 190d  ..Ve........6M..
        0x0020:  5018 002e 30a7 0000 0100 0001 0157 0000  P...0........W..
        0x0030:  0203 6465 6612 696e 666f 726d 6174 696f  ..def.informatio
        0x0040:  6e5f 7363 6865 6d61 0b54 4142 4c45 5f4e  n_schema.TABLE_N
        0x0050:  414d 4553 0b54 4142 4c45 5f4e 414d 4553  AMES.TABLE_NAMES
        0x0060:  0f54 6162 6c65 735f 696e 5f6d 7973 716c  .Tables_in_mysql
        0x0070:  0a54 4142 4c45 5f4e 414d 450c 0800 4000  .TABLE_NAME...@.
        0x0080:  0000 fd01 0000 0000 0500 0003 fe00 0022  ..............."
        0x0090:  000d 0000 040c 636f 6c75 6d6e 735f 7072  ......columns_pr
        0x00a0:  6976 0300 0005 0264 620a 0000 0609 6462  iv.....db.....db
        0x00b0:  5f6f 705f 6c6f 6706 0000 0705 6576 656e  _op_log.....even
        0x00c0:  7405 0000 0804 6675 6e63 0c00 0009 0b67  t.....func.....g
        0x00d0:  656e 6572 616c 5f6c 6f67 0e00 000a 0d68  eneral_log.....h
        0x00e0:  656c 705f 6361 7465 676f 7279 0d00 000b  elp_category....
        0x00f0:  0c68 656c 705f 6b65 7977 6f72 640e 0000  .help_keyword...
        0x0100:  0c0d 6865 6c70 5f72 656c 6174 696f 6e0b  ..help_relation.
        0x0110:  0000 0d0a 6865 6c70 5f74 6f70 6963 0500  ....help_topic..
        0x0120:  000e 0468 6f73 7411 0000 0f10 6e64 625f  ...host.....ndb_
        0x0130:  6269 6e6c 6f67 5f69 6e64 6578 0700 0010  binlog_index....
        0x0140:  0670 6c75 6769 6e05 0000 1104 7072 6f63  .plugin.....proc
        0x0150:  0b00 0012 0a70 726f 6373 5f70 7269 760d  .....procs_priv.
        0x0160:  0000 130c 7072 6f78 6965 735f 7072 6976  ....proxies_priv
        0x0170:  0800 0014 0773 6572 7665 7273 0900 0015  .....servers....
        0x0180:  0873 6c6f 775f 6c6f 670c 0000 160b 7461  .slow_log.....ta
        0x0190:  626c 6573 5f70 7269 7605 0000 1704 7465  bles_priv.....te
        0x01a0:   7374 0600 0018 0574 6573 7431 0a00 0019   st.....test1....
        0x01b0:   0974 696d 655f 7a6f 6e65 1600 001a 1574   .time_zone.....t
        0x01c0:   696d 655f 7a6f 6e65 5f6c 6561 705f 7365   ime_zone_leap_se
        0x01d0:   636f 6e64 0f00 001b 0e74 696d 655f 7a6f   cond.....time_zo
        0x01e0:   6e65 5f6e 616d 6515 0000 1c14 7469 6d65   ne_name.....time
        0x01f0:   5f7a 6f6e 655f 7472 616e 7369 7469 6f6e   _zone_transition
        0x0200:   1a00 001d 1974 696d 655f 7a6f 6e65 5f74   .....time_zone_t
        0x0210:   7261 6e73 6974 696f 6e5f 7479 7065 0500   ransition_type..
        0x0220:   001e 0475 7365 7205 0000 1ffe 0000 2200   ...user.......".
```

4) 客户端向服务端发送一个普通的 tcp 包

```
19:22:18.232503 IP 192.168.86.101.59614 > localhost.localdomain.mysql: . ack 521 win 4209
        0x0000:   4500 0028 046b 4000 4006 07e1 c0a8 5665   E..(.k@.@.....Ve
        0x0010:   c0a8 56ce e8de 0cea 364d 190d c4d7 1ffa   ..V.....6M......
        0x0020:   5010 1071 46ea 0000 0000 0000 0000        P..qF.........
```

1.2.3 退出

客户端在命令行模式下输入命令：quit 退出数据库

1)客户端向服务端发送一个退出的命令包

```
15:50:46.533701 IP 192.168.86.101.58767 > localhost.localdomain.mysql: P 829834420:829834425(5) ack 3239997079 win 4357
        0x0000:   4500 002d 039d 4000 4006 08aa c0a8 5665   E..-..@.@.....Ve
        0x0010:   c0a8 56ce e58f 0cea 3176 44b4 c11e 6e97   ..V.....1vD...n.
        0x0020:   5018 1105 d5e3 0000 0100 0000 0100        P.............
```

2)三次握手断开连接（断开连接不是四次握手吗？但实际情况下测试如果是正常的退出只有三次握手的过程)

```
15:50:46.533733 IP 192.168.86.101.58767 > localhost.localdomain.mysql: F 5:5(0) ack 1 win 4357
        0x0000:   4500 0028 039e 4000 4006 08ae c0a8 5665   E..(..@.@.....Ve
        0x0010:   c0a8 56ce e58f 0cea 3176 44b9 c11e 6e97   ..V.....1vD...n.
        0x0020:   5011 1105 d7ea 0000 0000 0000 0000        P.............
15:50:46.533854 IP localhost.localdomain.mysql > 192.168.86.101.58767: F 1:1(0) ack 6 win 46
        0x0000:   4508 0028 648b 4000 4006 a7b8 c0a8 56ce   E..(d.@.@.....V.
        0x0010:   c0a8 5665 0cea e58f c11e 6e97 3176 44ba   ..Ve......n.1vD.
        0x0020:   5011 002e e8c0 0000                       P.......
15:50:46.534434 IP 192.168.86.101.58767 > localhost.localdomain.mysql: . ack 2 win 4357
        0x0000:   4500 0028 039f 4000 4006 08ad c0a8 5665   E..(..@.@.....Ve
        0x0010:   c0a8 56ce e58f 0cea 3176 44ba c11e 6e98   ..V.....1vD...n.
        0x0020:   5010 1105 d7e9 0000 0000 0000 0000        P.............
```

2.MySql 数据包结构的描述

2.1 包头(Packet Header)

每个数据包都有一个包头，具体格式如下 ：

```
Bytes                 Name
 -----                 ----
 3                     Packet Length
 1                     Packet Number

 Packet Length: The length, in bytes, of the packet
                that follows the Packet Header. There
                may be some special values in the most
                significant byte. The maximum packet
                length is (2**24 -1),about 16MB.

 Packet Number: A serial number which can be used to
                ensure that all packets are present
                and in order. The first packet of a
                client query will have Packet Number = 0
                Thus, when a new SQL statement starts,
                the packet number is re-initialised.
```

2.2 数据包

2.2.1 握手初始化包（Handshake Initialization Packet）

2.2.1.1 格式描述

```
Bytes                        Name
 -----                        ----
 1                            protocol_version
 n (Null-Terminated String)   server_version
 4                            thread_id
 8                            scramble_buff
 1                            (filler) always 0x00
 2                            server_capabilities
 1                            server_language
 2                             server_status
 2                             server capabilities (two upper bytes)
 1                             length of the scramble
10                             (filler)  always 0
 n                             rest of the plugin provided data (at least 12
bytes)
 1                             \0 byte, terminating the second part of a scramble

 protocol_version:    The server takes this from PROTOCOL_VERSION
                      in /include/mysql_version.h. Example value = 10.

 server_version:      The server takes this from MYSQL_SERVER_VERSION
                      in /include/mysql_version.h. Example value =
"4.1.1-alpha".

 thread_number:       ID of the server thread for this connection.

 scramble_buff:       The password mechanism uses this. The second part are the
                      last 13 bytes.
                      (See "Password functions" section elsewhere in this
document.)

 server_capabilities: CLIENT_XXX options. The possible flag values at time of
 writing (taken from  include/mysql_com.h):
  CLIENT_LONG_PASSWORD        1        /* new more secure passwords */
  CLIENT_FOUND_ROWS   2        /* Found instead of affected rows */
  CLIENT_LONG_FLAG    4        /* Get all column flags */
  CLIENT_CONNECT_WITH_DB      8        /* One can specify db on connect */
  CLIENT_NO_SCHEMA    16       /* Don't allow database.table.column */
  CLIENT_COMPRESS             32       /* Can use compression protocol */
  CLIENT_ODBC                 64       /* Odbc client */
  CLIENT_LOCAL_FILES  128      /* Can use LOAD DATA LOCAL */
  CLIENT_IGNORE_SPACE 256      /* Ignore spaces before '(' */
  CLIENT_PROTOCOL_41  512      /* New 4.1 protocol */
  CLIENT_INTERACTIVE  1024     /* This is an interactive client */
  CLIENT_SSL              2048 /* Switch to SSL after handshake */
  CLIENT_IGNORE_SIGPIPE   4096    /* IGNORE sigpipes */
  CLIENT_TRANSACTIONS 8192     /* Client knows about transactions */
  CLIENT_RESERVED         16384   /* Old flag for 4.1 protocol  */
  CLIENT_SECURE_CONNECTION 32768  /* New 4.1 authentication */
  CLIENT_MULTI_STATEMENTS 65536   /* Enable/disable multi-stmt support */
  CLIENT_MULTI_RESULTS    131072  /* Enable/disable multi-results */

 server_language:     current server character set number
 server_status:       SERVER_STATUS_xxx flags: e.g. SERVER_STATUS_AUTOCOMMIT
```

2.2.1.2 举例

```
Example Handshake Initialization Packet
                    Hexadecimal                ASCII
                    -----------                -----
protocol_version    0a                         .
server_version      34 2e 31 2e 31 2d 71 6c    4.1.1-al
                    70 68 61 2d 64 65 62 75    pha-debu
                    67 00                      g.
thread_number       01 00 00 00                ....
scramble_buff       3a 23 3d 4b 43 4a 2e 43    ........
(filler)            00                         .
server_capabilities 2c 82                      ..
server_language     08                         .
server_status       02 00                      ..
(filler)            00 00 00 00 00 00 00 00    ........
                    00 00 00 00 00
```

2.2.2 客户端验证包(Client Authentication Packet)

2.2.2.1 格式描述

```
VERSION 4.0
 Bytes                        Name
 -----                        ----
 2                            client_flags
 3                            max_packet_size
 n  (Null-Terminated String)  user
 8                            scramble_buff
 1                            (filler) always 0x00

 VERSION 4.1
 Bytes                        Name
 -----                        ----
 4                            client_flags
 4                            max_packet_size
 1                            charset_number
 23                             (filler) always 0x00...
 n (Null-Terminated String)     user
 n (Length Coded Binary)        scramble_buff (1 + x bytes)
 n (Null-Terminated String)     databasename (optional)

 client_flags:              CLIENT_xxx options. The list of possible flag
                            values is in the description of the Handshake
                            Initialisation Packet, for server_capabilities.
                            For some of the bits, the server passed "what
                            it's capable of". The client leaves some of the
                            bits on, adds others, and passes back to the server.
                            One important flag is: whether compression is desired.
                            Another interesting one is: CLIENT_CONNECT_WITH_DB,
                            which shows the presence of the optional databasename.

 max_packet_size:           the maximum number of bytes in a packet for the client

 charset_number:            in the same domain as the server_language field that
                            the server passes in the Handshake Initialization
packet.

 user:                      identification

 scramble_buff:             the password, after encrypting using the scramble_buff
                            contents passed by the server (see "Password
functions"
                            section elsewhere in this document)
                            if length is zero, no password was given

 databasename:              name of schema to use initially
```

2.2.2.2 举例

```
Example Client Authentication Packet
                    Hexadecimal                ASCII
                    -----------                -----
client_flags        85 a6 03 00                ....
max_packet_size     00 00 00 01                ....
charset_number      08                         .
(filler)            00 00 00 00 00 00 00 00    ........
                    00 00 00 00 00 00 00 00    ........
                    00 00 00 00 00 00 00       .......
```

```
user                70 67 75 6c 75 74 7a 61     pgulutza
                    6e 00                       n.
```

2.2.3 命令包

2.2.3.1 格式描述

```
Bytes                        Name
 -----                        ----
 1                            command
 n                            arg

 command:      The most common value is 03 COM_QUERY, because
               INSERT UPDATE DELETE SELECT etc. have this code.
               The possible values at time of writing (taken
               from /include/mysql_com.h for enum_server_command) are:

               #      Name                Associated client function
               -      ----                --------------------------


               0x00   COM_SLEEP           (none, this is an internal thread
state)
               0x01   COM_QUIT            mysql_close
               0x02   COM_INIT_DB         mysql_select_db
               0x03   COM_QUERY           mysql_real_query
               0x04   COM_FIELD_LIST      mysql_list_fields
               0x05   COM_CREATE_DB       mysql_create_db (deprecated)
               0x06   COM_DROP_DB         mysql_drop_db (deprecated)
               0x07   COM_REFRESH         mysql_refresh
               0x08   COM_SHUTDOWN        mysql_shutdown
               0x09   COM_STATISTICS      mysql_stat
               0x0a   COM_PROCESS_INFO    mysql_list_processes
               0x0b   COM_CONNECT         (none, this is an internal thread
state)
               0x0c   COM_PROCESS_KILL    mysql_kill
               0x0d   COM_DEBUG           mysql_dump_debug_info
               0x0e   COM_PING            mysql_ping
               0x0f   COM_TIME            (none, this is an internal thread
state)
               0x10   COM_DELAYED_INSERT  (none, this is an internal thread
state)
```

```
               0x11   COM_CHANGE_USER     mysql_change_user
               0x12   COM_BINLOG_DUMP     sent by the slave IO thread to request
a binlog
               0x13   COM_TABLE_DUMP      LOAD TABLE ... FROM MASTER
(deprecated)
               0x14   COM_CONNECT_OUT     (none, this is an internal thread
state)
               0x15   COM_REGISTER_SLAVE  sent by the slave to register with the
master (optional)
               0x16   COM_STMT_PREPARE    mysql_stmt_prepare
               0x17   COM_STMT_EXECUTE    mysql_stmt_execute
               0x18   COM_STMT_SEND_LONG_DATA mysql_stmt_send_long_data
               0x19   COM_STMT_CLOSE      mysql_stmt_close
               0x1a   COM_STMT_RESET      mysql_stmt_reset
               0x1b   COM_SET_OPTION      mysql_set_server_option
               0x1c   COM_STMT_FETCH      mysql_stmt_fetch

 arg:           The text of the command is just the way the user typed it, there
is no processing
                by the client (except removal of the final ';').
                This field is not a null-terminated string; however,
                the size can be calculated from the packet size,
                and the MySQL client appends '\0' when receiving.
```

2.2.3.2 举例

```
Example Command Packet
                    Hexadecimal                ASCII
                    -----------                -----
command             02                         .
arg                 74 65 73 74                test
```

2.2.4 响应包

2.2.4.1 成功包

2.2.4.1.1 格式描述

```
VERSION 4.0
 Bytes                       Name
 -----                       ----
 1   (Length Coded Binary)   field_count, always = 0
 1-9 (Length Coded Binary)   affected_rows
 1-9 (Length Coded Binary)   insert_id
 2                           server_status
 n   (until end of packet)   message

 VERSION 4.1
 Bytes                       Name
 -----                       ----
 1   (Length Coded Binary)   field_count, always = 0
 1-9 (Length Coded Binary)   affected_rows
 1-9 (Length Coded Binary)   insert_id
 2                           server_status
 2                           warning_count
 n   (until end of packet)   message

 field_count:     always = 0

 affected_rows:   = number of rows affected by INSERT/UPDATE/DELETE

 insert_id:       If the statement generated any AUTO_INCREMENT number,
                  the number is returned here. Otherwise this field contains 0.
                  Note: when using for example a multiple row INSERT the
                  insert_id will be from the first row inserted, not from
                  last.

 server_status:   = The client can use this to check if the
                  command was inside a transaction.

 warning_count:   number of warnings

 message:         For example, after a multi-line INSERT, message might be
```

```
                    ″Records: 3 Duplicates: 0 Warnings: 0″
```

2.2.4.1.2 举例

```
Example OK Packet
                    Hexadecimal               ASCII
                    -----------               -----
field_count         00                        .
affected_rows       01                        .
insert_id           00                        .
server_status       02 00                     ..
warning_count       00 00                     ..
```

2.2.4.2 错误包

2.2.4.2.1 格式描述

```
VERSION 4.0
 Bytes                       Name
 -----                       ----
 1                           field_count, always = 0xff
 2                           errno (little endian)
 n                           message

 VERSION 4.1
 Bytes                       Name
 -----                       ----
 1                           field_count, always = 0xff
 2                           errno
 1                           (sqlstate marker), always '#'
 5                           sqlstate (5 characters)
 n                           message

 field_count:       Always 0xff (255 decimal).

 errno:             The possible values are listed in the manual, and in
                    the MySQL source code file /include/mysqld_error.h.
```

```
sqlstate marker:   This is always '#'. It is necessary for distinguishing
                   version-4.1 messages.

sqlstate:          The server translates errno values to sqlstate values
                   with a function named mysql_errno_to_sqlstate(). The
                   possible values are listed in the manual, and in the
                   MySQL source code file /include/sql_state.h.

message:           The error message is a string which ends at the end of
                   the packet, that is, its length can be determined from
                   the packet header. The MySQL client (in the my_net_read()
                   function) always adds '\0' to a packet, so the message
                   may appear to be a Null-Terminated String.
                   Expect the message to be between 0 and 512 bytes long.
```

2.2.4.2.2 举例

```
Example of Error Packet
                    Hexadecimal              ASCII
                    -----------              -----
field_count         ff                       .
errno               1b 04                    ..
(sqlstate marker)   23                       #
sqlstate            34 32 53 30 32           42S02
message             55 63 6b 6e 6f 77 6e 20  Unknown
                    74 61 62 6c 6c 65 20 27  table '
                    71 27                    q'
```

2.2.4.3 结果集包

一个结果集包由多个数据包组成，数据结构如下(按顺序列出)：

1. 结果集头部包（Result Set Header Packet）（1 个） the number of columns
2. 字段包(Field Packet)(n 个) column descriptors
3. 分隔包(EOF Packet)(1 个) marker: end of Field Packets
4. 行数据包(Row Data Packets)(n 个) row contents
5. 分隔包(EOF Packet)(1 个) marker: end of Field Packets

如，下面是当客户端使用 select count(*) from user 语句查询时 tcpdump 抓取的数据包：

客户端输入：select count(*) from user

服务端返回：查询结果

```
mysql> select count(*) from user;
+----------+
| count(*) |
+----------+
|          11 |
+----------+
1 row in set (0.00 sec)
```

1）客户端向服务端发送命令包

```
18:26:36.054761 IP 192.168.86.101.61382 > localhost.localdomain.mysql: P 454811053:454811083(30) ack 3069793847 win 4209
        0x0000:  4500 0046 04bd 4000 4006 0771 c0a8 5665  E..F..@.@..q..Ve
        0x0010:  c0a8 56ce efc6 0cea 1b1b ddad b6f9 5637  ..V..........V7
        0x0020:  5018 1071 bf9f 0000 1a00 0000 0373 656c  P..q.........sel
        0x0030:  6563 7420 636f 756e 7428 2a29 2066 726f  ect.count(*).fro
        0x0040:  6d20 7573 6572                               m.user
```

2)服务端向客户端发送一个普通 tcp 包

```
18:26:36.054762 IP localhost.localdomain.mysql > 192.168.86.101.61382: . ack 30 win 46
        0x0000:  4508 0028 594f 4000 4006 b2f4 c0a8 56ce  E..(YO@.@.....V.
        0x0010:  c0a8 5665 0cea efc6 b6f9 5637 1b1b ddcb  ..Ve......V7....
        0x0020:  5010 002e 7e59 0000                         P...~Y..
```

3)服务端向客户端发送一个结果集包

```
18:26:36.054980 IP localhost.localdomain.mysql > 192.168.86.101.61382: P 1:65(64) ack 30 win 46
        0x0000:  4508 0068 5950 4000 4006 b2b3 c0a8 56ce  E..hYP@.@.....V.
        0x0010:  c0a8 5665 0cea efc6 b6f9 5637 1b1b ddcb  ..Ve......V7....
        0x0020:  5018 002e 2edf 0000 0100 0001 011e 0000  P..............
        0x0030:  0203 6465 6600 0000 0863 6f75 6e74 282a  ..def....count(*
        0x0040:  2900 0c3f 0015 0000 0008 8100 0000 0005  )..?............
        0x0050:  0000 03fe 0000 0200 0300 0004 0231 3105  .............11.
        0x0060:  0000 05fe 0000 0200                         ........
```

4)客户端向客户端发送一个普通 tcp 包

```
18:26:36.254624 IP 192.168.86.101.61382 > localhost.localdomain.mysql: . ack 65 win 4193
        0x0000:  4500 0028 04bf 4000 4006 078d c0a8 5665  E..(..@.@.....Ve
        0x0010:  c0a8 56ce efc6 0cea 1b1b ddcb b6f9 5677  ..V..........Vw
        0x0020:  5010 1061 6de6 0000 0000 0000 0000          P..am.........
```

其中第三个包是结果集包，去除 ip 头和 tcp 头后红色部分即为真正的数据包：

```
4508 0068 5950 4000 4006 b2b3 c0a8 56ce  E..hYP@.@.....V.
c0a8 5665 0cea efc6 b6f9 5637 1b1b ddcb  ..Ve......V7....
5018 002e 2edf 0000 0100 0001 011e 0000  P..............
0203 6465 6600 0000 0863 6f75 6e74 282a  ..def....count(*
2900 0c3f 0015 0000 0008 8100 0000 0005  )..?............
0000 03fe 0000 0200 0300 0004 0231 3105  .............11.
0000 05fe 0000 0200
```

进一步对其分析如下（蓝色字体为包头）：

Result Set Header Packet
0100 0001 01 .....
Field Packet
1e 0000   ....
0203 6465 6600 0000 0863 6f75 6e74 282a   ..def....count(*
2900 0c3f 0015 0000 0008 8100 0000 00   )..?...........
EOF Packet
05 0000 03fe 0000 0200 ....
Row Data Packet
0300 0004 0231 31 11
EOF Packet
05 0000 05fe 0000 0200 ....

2.2.4.3.1 结果集头部包（Result Set Header Packet）

2.2.4.3.1.1 格式描述

```
Bytes                          Name
 -----                          ----
 1-9    (Length-Coded-Binary)   field_count
 1-9    (Length-Coded-Binary)   extra

 field_count: See the section "Types Of Result Packets"
              to see how one can distinguish the
              first byte of field_count from the first
              byte of an OK Packet, or other packet types.

 extra:       For example, SHOW COLUMNS uses this to send
              the number of rows in the table.
```

2.2.4.3.1.2 举例

```
Example of Result Set Header Packet
                    Hexadecimal                ASCII
                    -----------                -----
field_count         03
```

2.2.4.3.2 字段包(Field Packet)

2.2.4.3.2.1 格式描述

```
VERSION 4.0
 Bytes                      Name
 -----                      ----
 n (Length Coded String)    table
 n (Length Coded String)    name
 4 (Length Coded Binary)    length
 2 (Length Coded Binary)    type
 2 (Length Coded Binary)    flags
 1                          decimals
 n (Length Coded Binary)    default

 VERSION 4.1
 Bytes                      Name
 -----                      ----
 n (Length Coded String)    catalog
 n (Length Coded String)    db
 n (Length Coded String)    table
 n (Length Coded String)    org_table
 n (Length Coded String)    name
 n (Length Coded String)    org_name
 1                          (filler)
 2                          charsetnr
 4                          length
 1                          type
 2                          flags
 1                          decimals
 2                          (filler), always 0x00
 n (Length Coded Binary)    default
```

2.2.4.3.2.2 举例

```
Example of Field Packet
                    Hexadecimal               ASCII
                    -----------               -----
catalog             03 73 74 64               .std
db                  03 64 62 31               .db1
table               02 54 37                  .T7
org_table           02 74 37                  .t7
name                02 53 31                  .S1
```

```
org_name            02 73 31                  .s1
(filler)            0c                        .
charsetnr           08 00                     ..
length              01 00 00 00               ....
type                fe                        .
flags               00 00                     ..
decimals            00                        .
(filler)            00 00                     ..
```

2.2.4.3.3 分隔包(EOF Packet)

2.2.4.3.3.1 格式描述

```
VERSION 4.0
 Bytes                 Name
 -----                 ----
 1                     field_count, always = 0xfe

 VERSION 4.1
 Bytes                 Name
 -----                 ----
 1                     field_count, always = 0xfe
 2                     warning_count
 2                     Status Flags

 field_count:          The value is always 0xfe (decimal 254).
                       However ... recall (from the
                       section "Elements", above) that the value 254 can begin
                       a Length-Encoded-Binary value which contains an 8-byte
                       integer. So, to ensure that a packet is really an EOF
                       Packet: (a) check that first byte in packet = 0xfe, (b)
                       check that size of packet < 9.

 warning_count:        Number of warnings. Sent after all data has been sent
                       to the client.

 server_status:        Contains flags like SERVER_MORE_RESULTS_EXISTS
```

2.2.4.3.3.2 举例

```
Example of EOF Packet
                    Hexadecimal                ASCII
                    -----------                -----
field_count         fe                         .
warning_count       00 00                      ..
server_status       00 00                      ..
```

2.2.4.3.4 行数据包

2.2.4.3.4.1 格式描述

```
Bytes                   Name
 -----                   ----
 n (Length Coded String) (column value)
 ...

 (column value):       The data in the column, as a character string.
                       If a column is defined as non-character, the
                       server converts the value into a character
                       before sending it. Since the value is a Length
                       Coded String, a NULL can be represented with a
                       single byte containing 251(see the description
                       of Length Coded Strings in section "Elements" above).
```

2.2.4.3.4.2 举例

```
Example of Row Data Packet
                    Hexadecimal                ASCII
                    -----------                -----
(first column)      01 58                      .X
(second column)     02 35 35                   .55
```