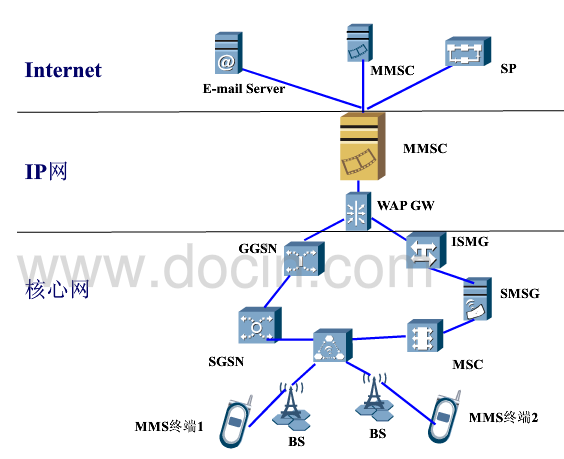
MMS服务器交互流程的抓包分析

Analysed WireShark

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comment** |
| 0.0.1 | 2014-7-9 | io | create this document. |
|  |  |  |  |

# 彩信的定义

彩信的英文名是MMS，它是Multimedia Messaging Service的缩写，意为多媒体信息服务，彩信的最大的特色就是支持多媒体功能，能够传递功能全面的内容和信息，包括文字、图像、声音、数据等各种多媒体格式的信息，彩信是在GPRS网络的支持下，以WAP无线应用协议为载体传送图片、声音和文字等信息，彩信业务可实现即时的手机端到端、手机终端到互联网或互联网到手机终端的多媒体信息传送。



## 发送方发送消息

1. 编辑欲发送的多媒体消息

2. 终端与MMSC建立WAP连接，通过WAP网关将信息内容发送至MMSC

3. MMSC存储信息并进行数据分析，同时对发送方做出响应，发送方终端显示“消息已发出”

## MMSC通知接收方

1. MMSC使用WAP PUSH 向接收方发送一条通知消息

2. 接收方与MMSC建立WAP连接，使用HTTP GET从MMSC取回MMS消息

3. 接收方终端仍通过同一个WAP连接用HTTP POST消息告知接收成功

4. MMSC使用WAP PUSH 告知发送方消息已送达，发送方终端显示“消息已送达”

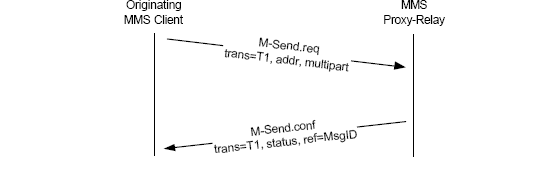
## 补充说明

* MMSC并不是直接将MMS消息发送给接收者，而是向其发送一个通知，告诉接收方有一条消息正在等待。
* 根据终端设置的不同，接收方的终端将尝试立即提取该消息，或者推迟一段时间提取，又或者仅仅将通知放在一边，不予理会。
* 由于某些原因信息中心无法通知到接收者时，信息中心将消息保存一定时间后再次发送。
* 若在一定时间内总是无法送达，就丢弃这条消息。

# 交互过程

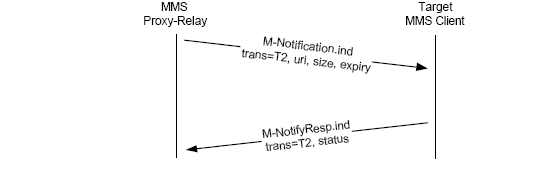
## 发送过程

彩信发送方把彩信发送给MMSC的过程，MMSC在收到彩信后会给发送方一个确认消息。



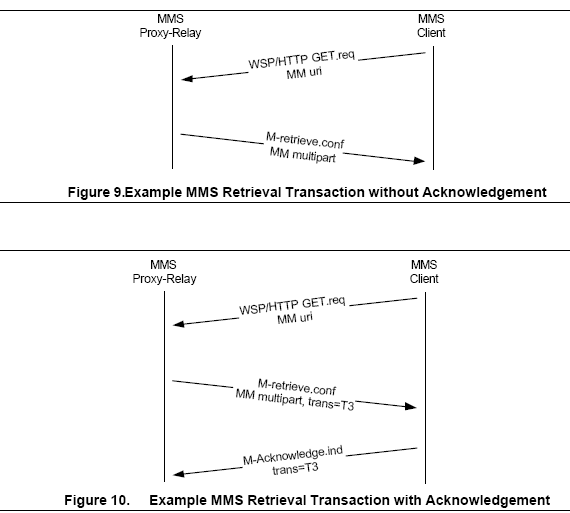
## 通知过程

为了把彩信投递给接收方，MMSC要通过PUSH协议给接收方发送一条彩信通知消息，这个消息通常是一条特殊短信，里面包含彩信的位置URL。



## 提取彩信

接收方收到彩信通知后，从中取出URL，然后通过标准的HTTP GET请求从MMSC上获取彩信。



## 彩信回执

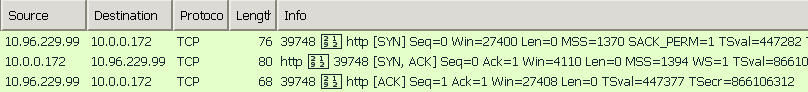
当MMSC成功的通知彩信接收方后，它会给彩信发送方发送一个消息表明彩信投递成功。



# WireShark抓包业务流程

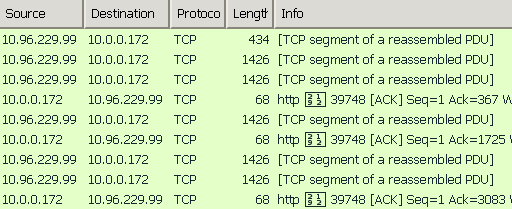
## 与WAP网关建立连接

发送端与WAP网关经过“三次握手”，建立TCP连接。



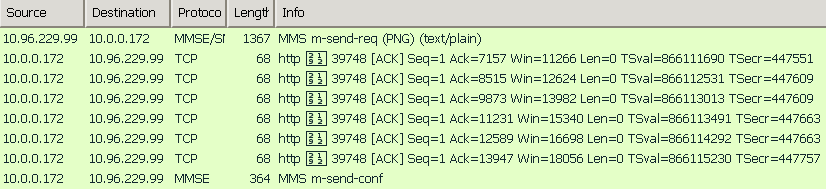
## 发送端向彩信中心上传彩信内容

数据传输过程中，直接和发送端通信的是WAP网关，所有数据都是通过WAP网关转发给彩信中心（MMSC）的，我们无法看到GPRS网内的设备，所以这里看到的接收端为WAP网关（10.0.0.172）。



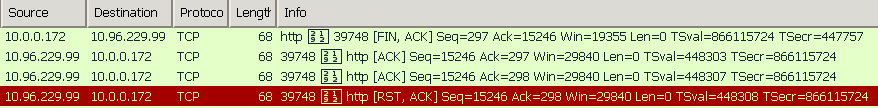
## MMSC发送成功响应

“M-Send.req”被视为发送端将彩信内容上传到MMSC，MMSC在接收到彩信全部内容后，通过WAP网关向发送端下发一条 “M-Send.conf”，表明彩信成功发送成功。



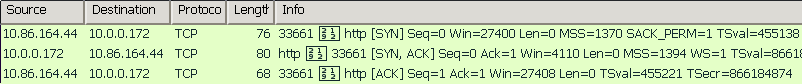
## 发送端断开与WAP网关的连接

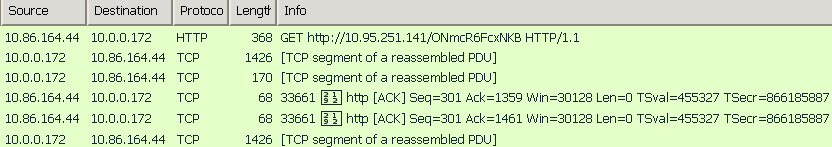
在收到上一步WAP网关返回的确认信息后，中断其与发送端的连接并告知（[FIN,ACK]）发送端，发送端收到信息后将最后的确认信息（[RST,ACK]）返回给WAP网关。至此，发送MMS的过程已经全部结束。



## 接收端提取彩信内容

接收端下载彩信内容，首先需要与WAP网关建立连接。



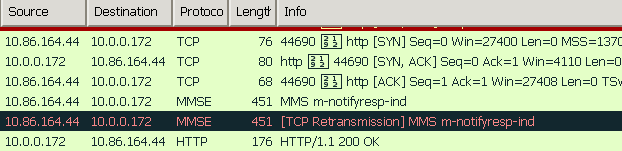
然后发出GET请求。在这个过程中，接收端需要与下载地址（10.95.251.141）进行通信，UE只和网关交互，从下载服务器过来的数据包通过WAP网关后变成了10.0.0.172的数据包。  


数据传输过程中，最后一个数据包下发的同时，MMSC还会通过WAP网关下发一条获取彩信回应（m-retrieve-conf），已经对全部彩信内容进行了下载。然后网关发送FIN包给UE，断开TCP连接。



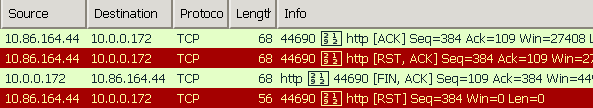
## 提取彩信确认

接收端通过WAP网关向MMSC发送回应信息（m-notifyresp-ind），表明已经成功接收收到MMSC下发PUSH消息并提取了彩信内容，彩信中心通过WAP网关返回确认信息（HTTP/1.1 200 ok），表明彩信提取成功。

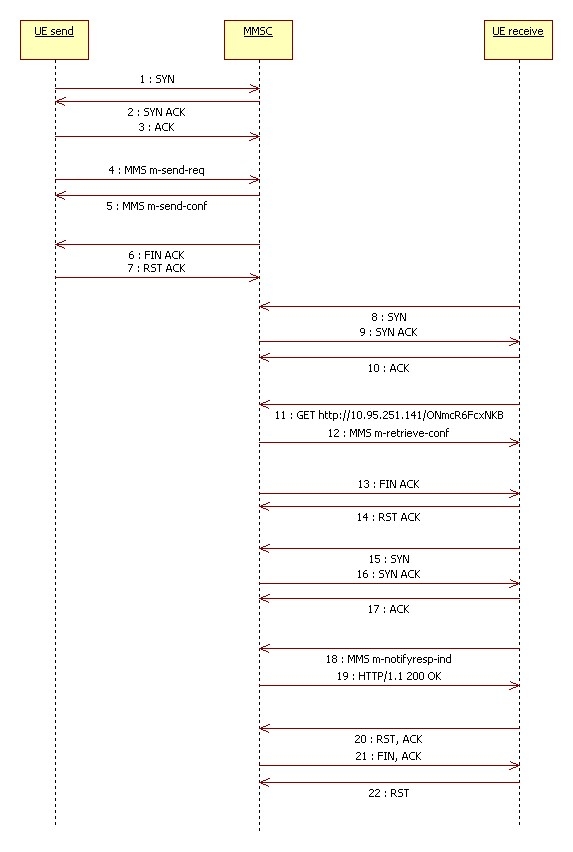


## 接收端端断开与WAP网关的连接

WAP网关与接收端先后中断与对方的连接。彩信接收完成。



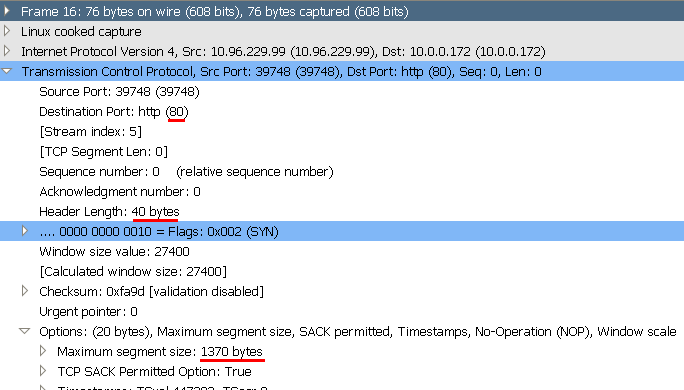
## 整体描述



# TCP层分析

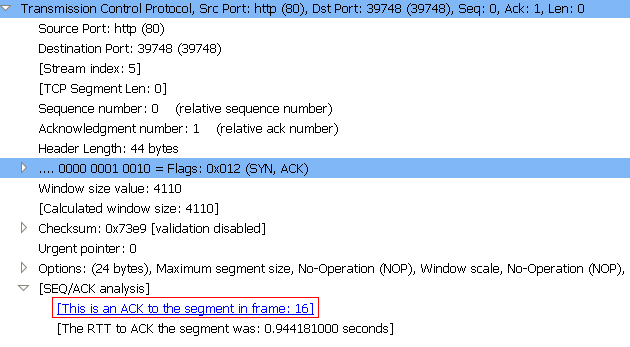
## 发送端与WAP网关建立连接

### SYN

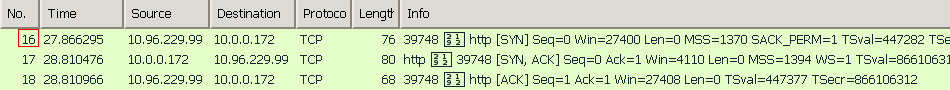


* Dst port（80）表示发送端请求使用80端口与WAP网关进行连接。
* Header Length：40表示原有数据在TCP层增加了40bytes的包头。
* Maximum segment size：1370bytes表示在TCP层，一个数据包最大为1370bytes。
* WAP网关接收到发送端发出的连接请求后，会返回一条以（SYN,ACK）标记的确认信息

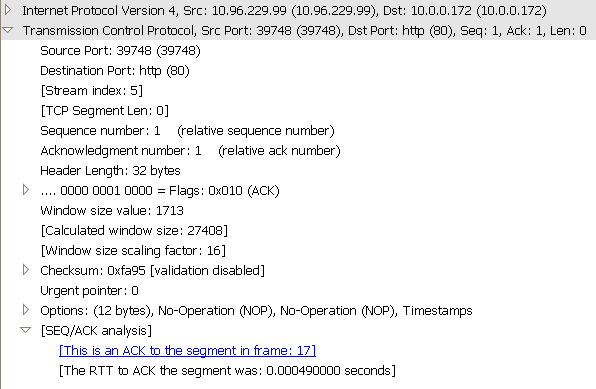
### SYN, ACK



“This is an ACK to the segment in frame：16”表明了这条信息是对第16帧信息的回复。即

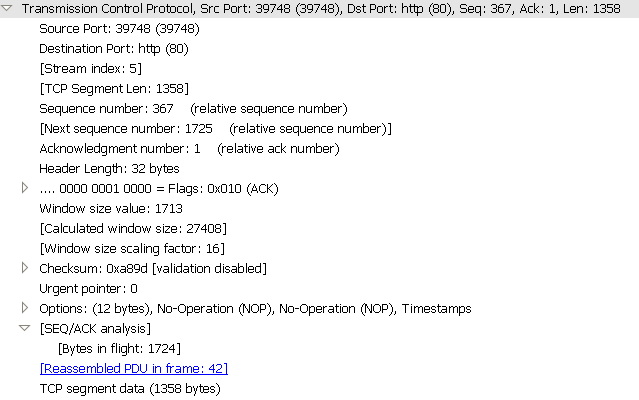


### ACK



上图是建立TCP连接的最后一步，发送端在收到WAP网关的回复后，再向其发送一条确认信息（ACK），表明通过80端口建立连接成功，接下来就要传输数据了。这条信息的TCP层包头大小为32bytes。

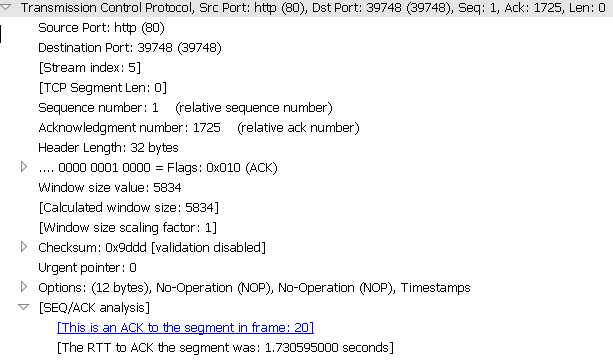
## 发送端向彩信中心上传彩信内容



彩信内容在TCP层经过打包（包头为20bytes）之后，通过HTTP的80端口向彩信中心传输。

* TCP segment data（1358bytes）表示当前传输的包的大小为1358bytes。
* “Reassembled PDU in frame：42”表示在第42帧被重新组合。

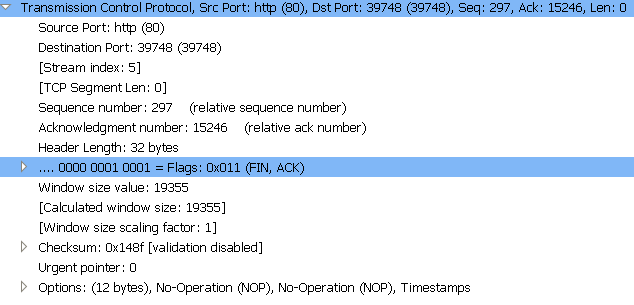
对于发送端上传到彩信中心的所有数据包，彩信中心再收到后会回复一条确认信息，表明该数据包已经成功接收。



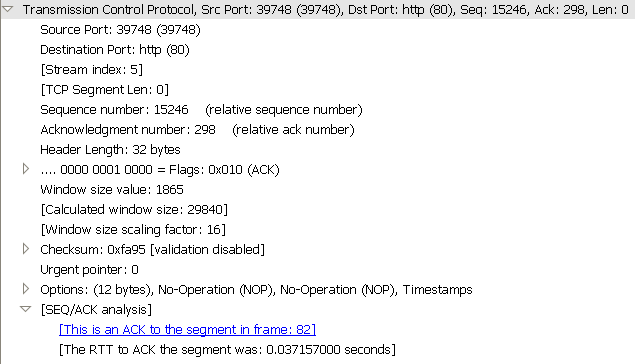
* “This is an ACK to the segment in frame：20”表明该条确认信息是对第20帧的回复
* “The RTT to ACK the segment was：1.730595000 senconds”表明距发送时间相隔了1.730595000秒。

## 发送端断开与WAP网关的连接

WAP网关发送TCP层数据包，断开与发送端连接。



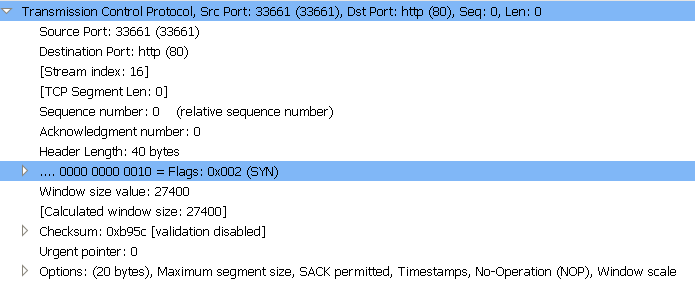
在接收到WAP网关断开连接的确认后，发送端最后会像WAP网关发送一条ACK确认信息，表明连接已经被中断。



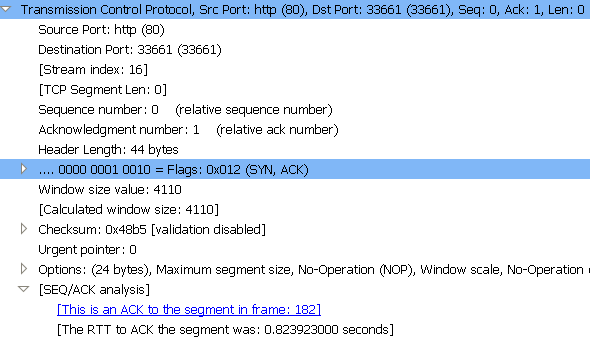
第82帧是WAP网关向发送端发送的断开连接的信息。

## 接收端与WAP网关建立连接

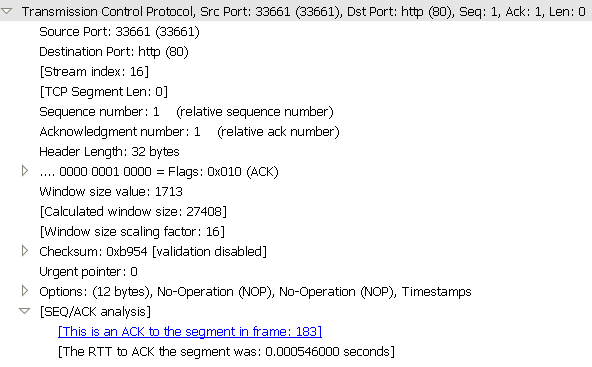
### SYN



### SYN,ACK



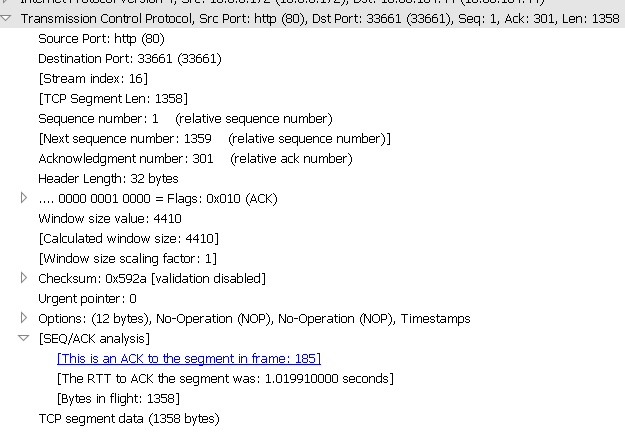
### ACK



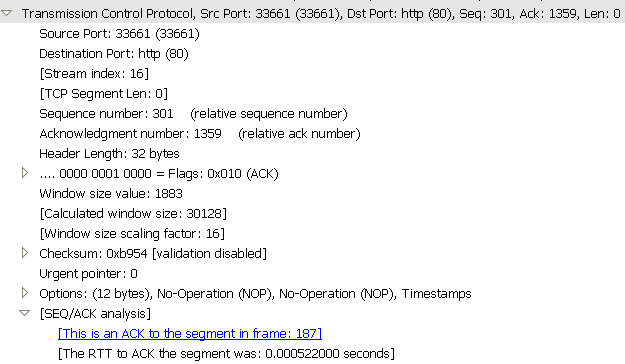
上图为接收端与WAP网关建立连接的过程，请求首先由接收端发起，WAP网关接收到请求后会返回一条确认信息（SYN,ACK），最后接收端向WAP网关发送一条ACK信息，表明连接已经建立。

## 接收端下载彩信内容

接收端下载彩信的过程中，彩信内容在TCP层仍然被打包（包头为32bytes），数据的流向是从彩信中心到接收终端，当接收端成功接收一个TCP包后，会向彩信中心发送一条确认信息。



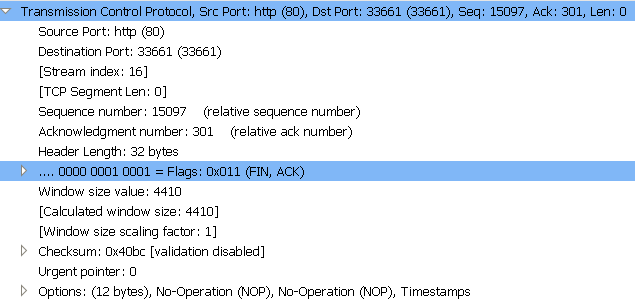
* “Reassembled PDU in frame：185”表示下载的彩信数据包在第185帧会被重新组合。
* 1358 bytes是TCP包的大小。



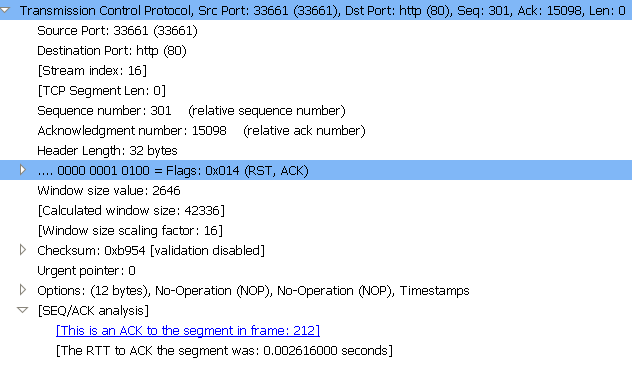
* 该条确认信息是对第187帧传输的数据包的确认。

## WAP网关断开与接收端的连接

接收端成功下载全部数据包后，WAP网关会主动向接收端发送一条由（FIN,ACK）标记的信息，表明传输结束，连接已从WAP网关一侧中断。这条信息同样被打上32bytes的包头发送出去。



接收端收到WAP网关断开连接的信息后，立即发送一条ACK信息，给予WAP网关回应，同时也向WAP网关发送一条由（RST,ACK）标记的信息，表明连接已中断。



* “This is an ACK to the segment in frame：212”表明该条信息是对第212帧的确认回复。

# HTTP层分析

## 发送端上传彩信内容

在发送端上传彩信内容的过程中，彩信内容加上发送终端的HTTP包头信息形成了HTTP层的数据包。



在HTTP包头中：

* “Method”指传输方法。
* “URI”是目标地址。
* “Version”是HTTP协议版本号。
* “content-length”是HTTP层的数据总大小。
* “content-type”是传输的内容类型。

## 彩信中心回复确认信息

接收完发送端上传的全部彩信内容后，彩信中心会通过WAP网关向发送端下发一条确认信息。



其中HTTP部分包含了：

* HTTP协议版本号。
* 返回的状态码。
* x-mmsc-tid 彩信编号。
* x-mmsc-from, x-mmsc-to 发送及接收方号码。
* Date: 日期时间等内容。

## 接收端下载彩信内容

接收端在接收到彩信中心下发的PUSH消息后，会根据PUSH消息中的信息去指定地点提取彩信内容。

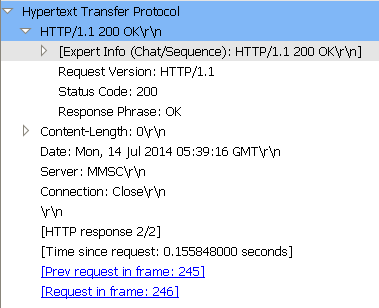


在接收端发送的get请求中，包含了含有必要信息的HTTP包头，其中包括：

* 下载地址
* 传输方法
* HTTP协议版本号
* 终端型号
* 终端支持的内容类型等信息。

## 接收端下载彩信内容

当接收端下载完所有的彩信内容后，彩信中心会通过WAP网关向接收端发送一条信息，告知彩信内容已全部提取。



在这条信息的HTTP包头中，包含了：

* HTTP协议版本号。
* 返回的状态码。

对应接收端提出的get请求，在提取彩信内容后，彩信中心还会向接收端发送一条HTTP响应信息，内容与上图基本一致。

# MMSE层分析 -- MMS PDU

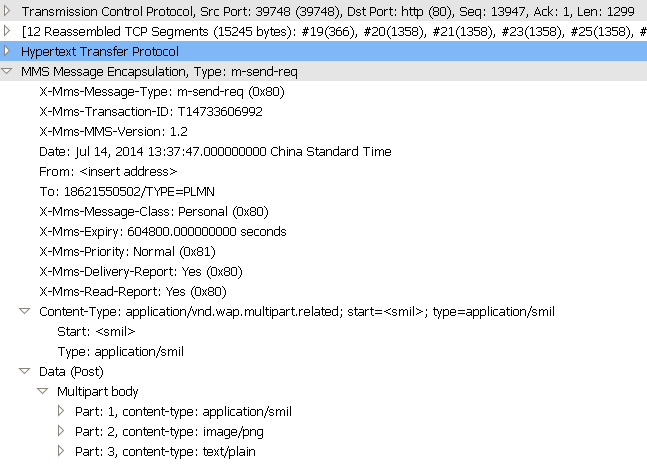
在MMSE层，被传送的是MMS PDU（协议数据单元），个MMS PDU 由MMS头和MMS体构成。

只在m-send-req和m-retrieve-conf中MMS PDU才包含了MMS体，其它部分只对MMS头进行传送MMS PDU被依次传递给WSP或者HTTP消息的内容部分（取决于使用哪种传输协议），这些消息的content-type被设置为application/vnd.wap.mms-message

每个MMS PDU的头三个参数依次为X-Mms-Message-Type, X-Mms-Transaction-ID 和X-Mms-MMS-Version。不同类型的PDU对应不同的角色，由X-Mms-Message-Type来标识。

## 发送彩信：m-send-req

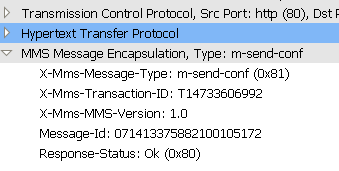
m-send-req标识了该条PDU是指彩信的发送



* MMS Message Encapsulation是MMS头。
* X-Mms-Message-Type: m-send-req
* X-Mms-Transaction-ID:T14733606992 指定了该条PDU的传输号。
* X-Mms-MMS-Version:1.2 是彩信协议的版本号。
* X-Mms-Delivery-Report指彩信的信息回复。
* Data是彩信的原有内容，即MMS体。

## 发送确认：m-send-conf

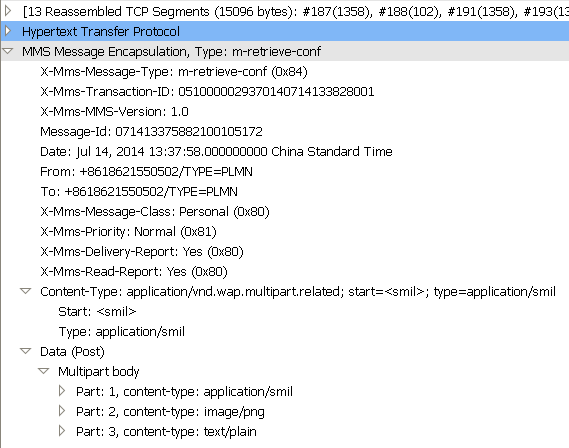
m-send-conf由彩信中心发送到彩信的发送端，是对m-send-req的确认，表明彩信中心已经成功接收发送端上传的彩信内容。



* X-Mms-Transaction-ID与m-send-req步骤一致。
* Response-Status返回的状态码为ok说明发送彩信成功。

## 提取彩信回应：m-retrieve-conf

与发送彩信步骤相同，该条MMS PDU包含了MMS头和MMS体，使用m-retrieve-conf标识，表示该条信息是成功提取彩信后的确认，由彩信接收端发送到彩信中心。

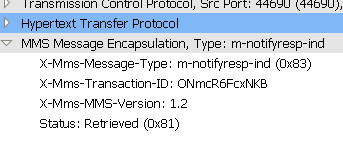


MMS头记录了：

* 该条PDU的编号。
* 接收彩信的时间。
* 彩信的发送方与接收方号码。
* 彩信主题。
* 是否阅读彩信回执等信息。

## 彩信通知回应：m-notifyresp-ind

接收端收到彩信中心下发的PUSH消息后，会回应一条由m-notifyresp-ind标识的信息，表示已经接收到彩信通知。



Retrieved表示已经接收到彩信通知。

# 问题总结

## 彩信发送失败

发送端与彩信中心建立连接失败

发送端上传彩信过程中连接被断开

彩信发送时延过大

发送端上传彩信过程中出现大量丢包

## 下载失败

接收端没有接收到彩信中心下发的PUSH消息

接收端彩信中心建立连接失败

接收端下载彩信过程中连接被断开

## 彩信接收时延过大

接收端下载彩信过程中出现大量丢包

# 附录A：名词解释

|  |  |
| --- | --- |
| MMS | mms是英文缩写，它可以是Membership Management System的缩写，中文译名为会员管理系统。也可以是Multimedia Messaging Service的缩写，中文译为多媒体短信服务。 |
| MMSC | MMS中继服务器/多媒体消息业务中心(MMSC)：MMSC是整个多媒体消息系统的核心，对多媒体消息进行存储和处理，包括消息的输入输出、地址解析、通知、报告等。同时，负责多媒体消息在不同MMSC之间的传递等操作。MMSC还产生CDR话单用于计费。另外，MMSC需要很多到其他网络的连接：并开展各种增值服务。MMS中继正是系统的IP接口。系统通过它与各种网络相连，支持多种协议。 |
| PDU | 协议数据单元，是指在分层网络结构，例如在开放式系统互联(OSI)模型中，在传输系统的每一层都将建立协议数据单元(PDU)。 |
| WAP网关 | WAP网关在技术上遵循WAP论坛定义的MMS规范，以支持多媒体消息业务。通过WAP网关建立MMS用户代理与MMS中继服务器的数据访问通道，从而支持多媒体信息的发送、接收、通知等操作。对于MMS用户代理与WAP网关的连接，则采用WAPStack实现在不同无线网络平台上的承载。针对MMS应用而言，采用无线会话协议（WSP）传递数据。 |
| WireShark | Wireshark（前称Ethereal）是一个网络封包分析软件。网络封包分析软件的功能是撷取网络封包，并尽可能显示出最为详细的网络封包资料。Wireshark使用WinPCAP作为接口，直接与网卡进行数据报文交换。 |

# 附录B：WireShark文件

