**OSNIFFER 项目需求**

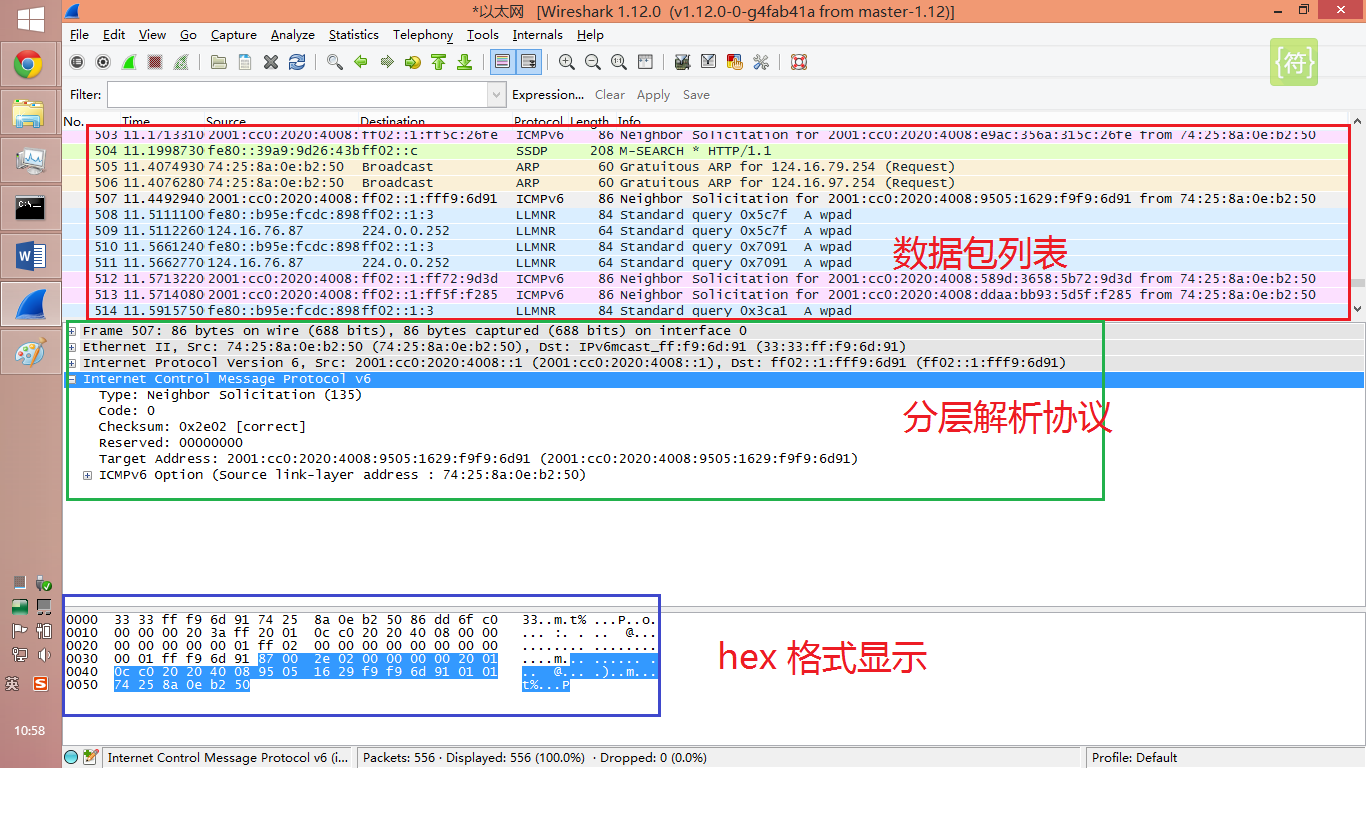
# FEATURES(功能)：

osniffer本质上讲就是web 版wireshark，所以在前期，主要以实现wireshark的功能为主，后期可以考虑加入一些其他特性(比如集成一些网络安全工具)。

下面是osniffer最最核心的功能：

## Protocol Resolution(网络协议解析)：

选择某一个监听的网卡并开始监听，捕获到实时的数据包，选择对应的数据包可以按照协议栈分层解析协议，辅以直观的hex格式（最下方）展示，如图1所示：



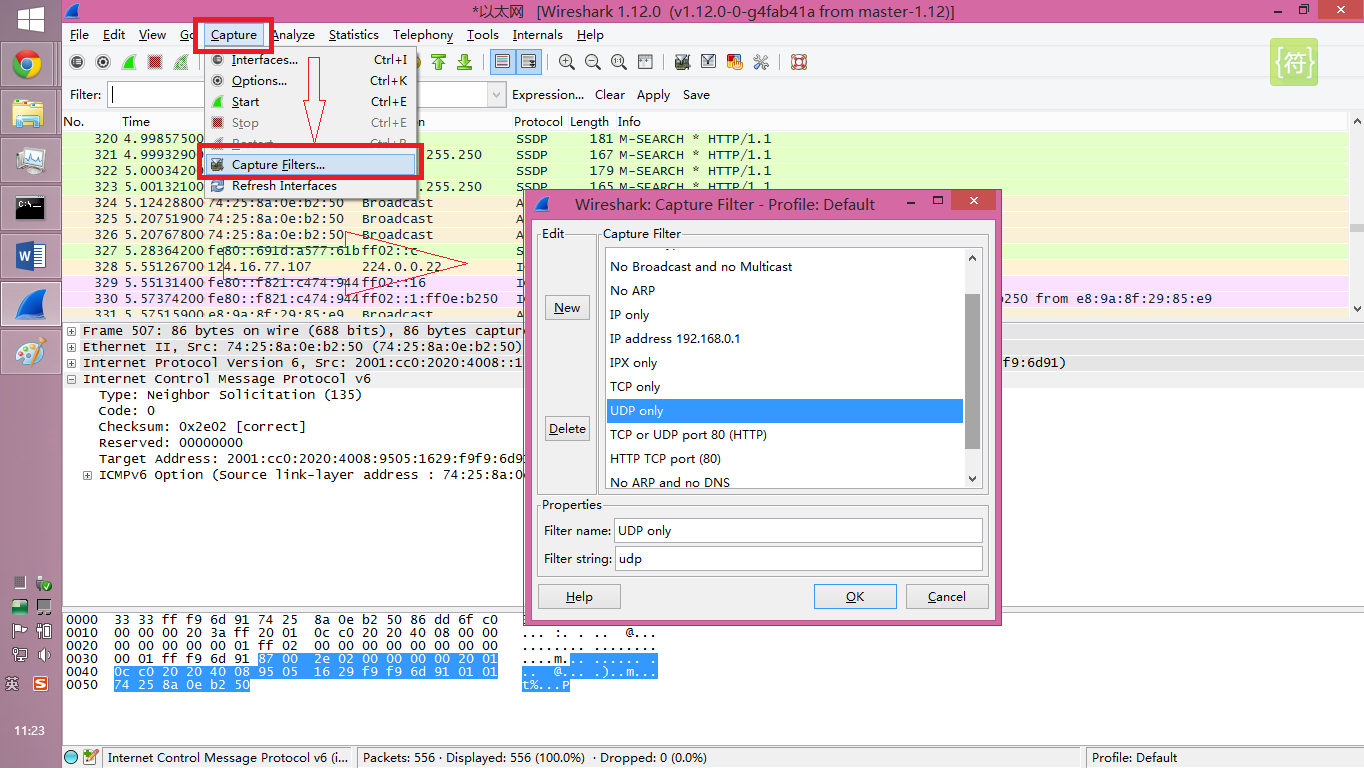
**图1 解析网络协议**

## Filter[capture\display](过滤[捕获\显示])：

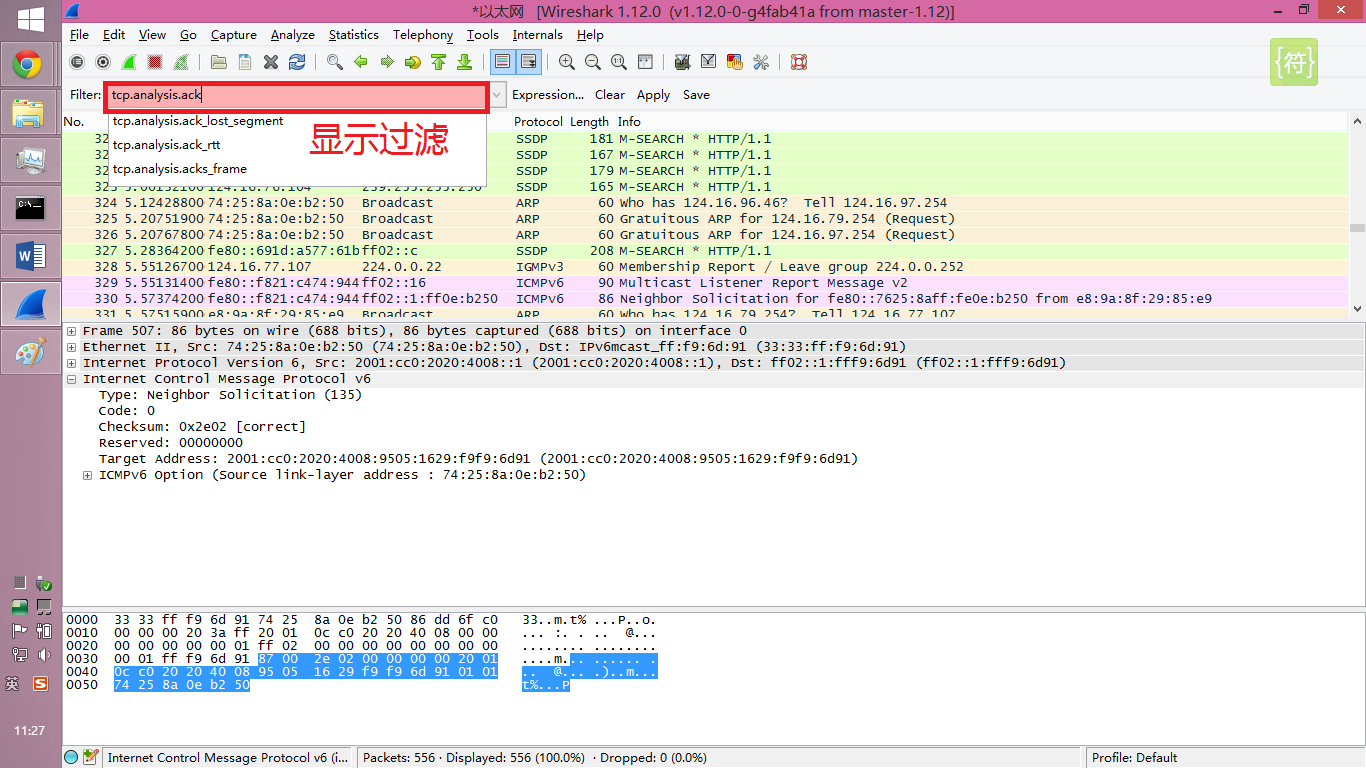
捕获过滤：在底层驱动的时候就进行过滤，这个驱动有相应的支持。一次性过滤(过滤了之后就捕获不到了)。

显示过滤：只显示部分数据包，需要我们自己编写。可任意次过滤（数据包已经被捕获到，只是显不显示的问题）。

如图2、3所示：



**图2 捕获过滤**

****

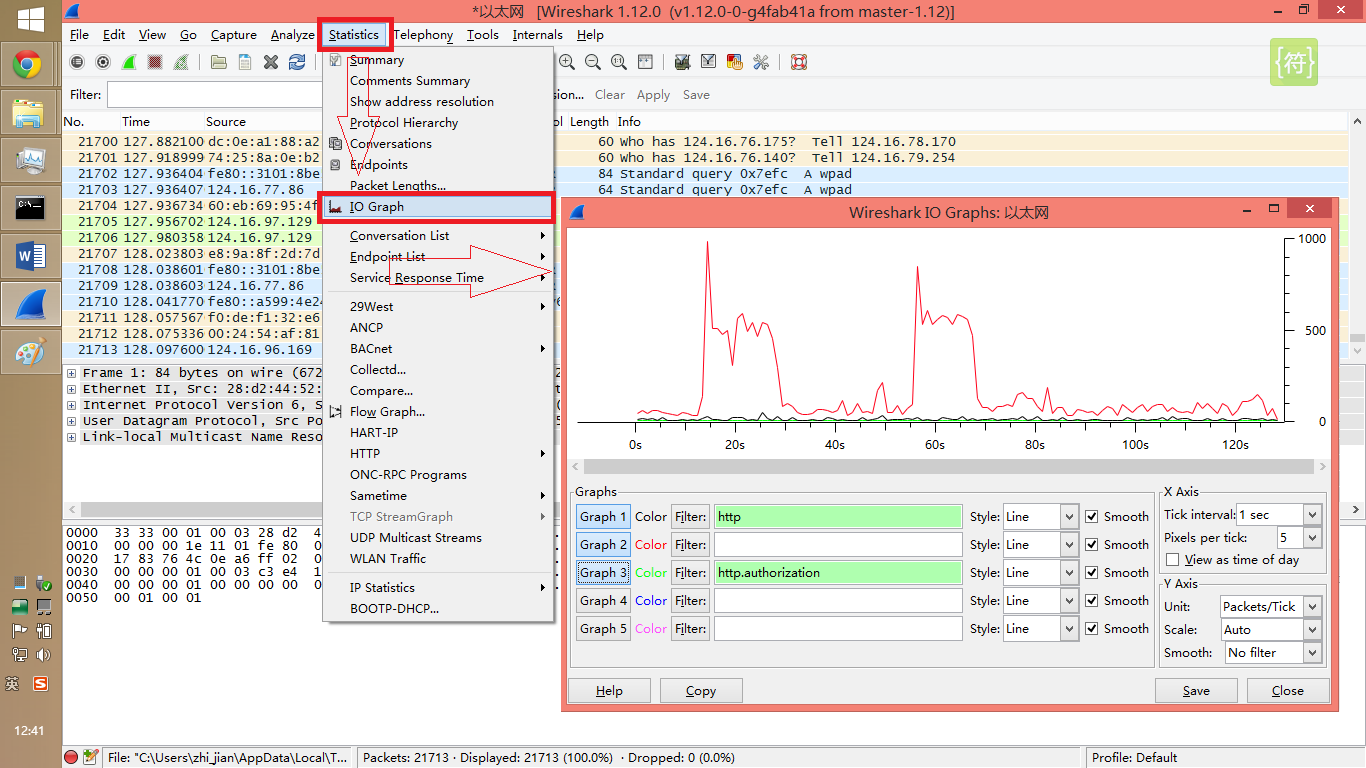
**图3 显示过滤**

## Statistics(统计数据)：

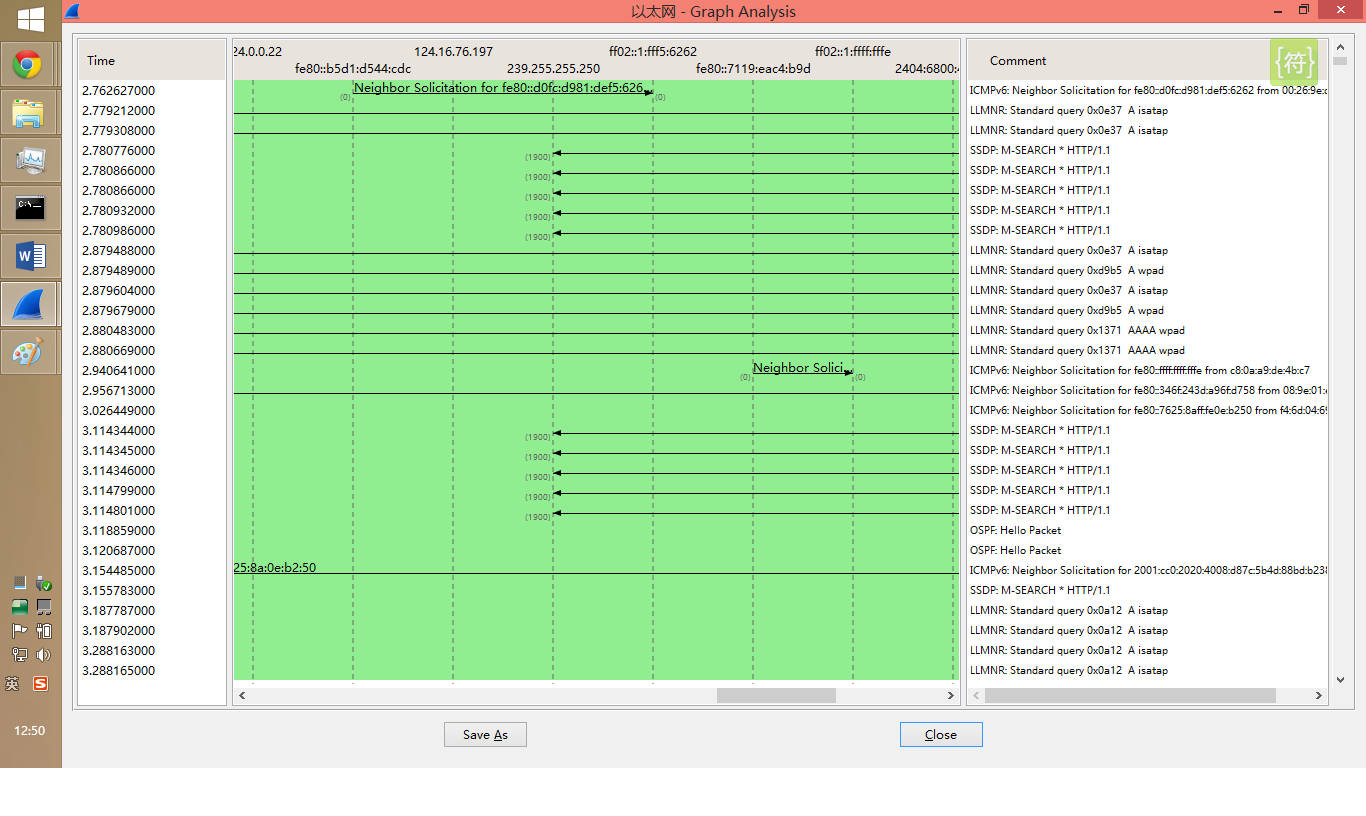
统计数据功能最需要的就是直观的展示一个捕获过程的一些统计信息，如网络IO统计图(IO graph)、数据流图(Flow graph)，如图4、5所示：

IO统计图以折线统计图的形式显示单位时间内某一类型(可指定5个不同的过滤)的数据包的个数；

Flow Graph 则是以序列图的形式，直观的显示了时间轴、不同主机的请求序列关系，相当赞的功能。



**图4 IO graph**

**图5 Flow Graph**

## Save(保存文件)：

把捕获到的数据以文件的形式保存下来，而且是以wireshark的数据文件.pcapng的格式保存(良好的兼容性)。

# CHALLENGES(难点)：

## analyze(分析)：

**大量** osniffer最最基本的就是需要解析各种各样的网络协议

**重组** 大数据的数据通常会分多个包传输，为了便于分析，需要将这些多个包重组，比如TCP报文重组的功能，HTTP报文中一幅图片的重组。

**流程** Flow graph中需要了解网络的各个协议的流程

## abstract(抽象)：

由于网络协议非常非常多，在解析协议的时候必须使用面向对象的方式设计这些协议，务必做到数据解析和UI完全分开。需要一个好的设计、高度抽象的设计，开发方便、维护方便、可重用度高。而且对于“显示过滤”这也是必须的。

## compiler(编译器)：

在实现显示过滤的时候，需要提供给用户：支持逻辑运算(and or not xor like等)、支持嵌套的过滤表达式，需要设计一个语法分析器（可以理解为一个稍微复杂一点的计算器），还必须进行语法检查。

## format(格式)：

保存文件的格式为.pcapng，需要阅读相关的文档，了解具体的格式。