

# 目 录

## [第1章 Sniffer软件简介](#)

### [1.1 概述](#)

### [1.2 功能简介](#)

## [第2章 报文捕获解析](#)

### [2.1 捕获面板](#)

### [2.2 捕获过程报文统计](#)

### [2.3 捕获报文查看](#)

### [2.4 设置捕获条件](#)

## [第3章 报文放送](#)

### [3.1 编辑报文发送](#)

### [3.2 捕获编辑报文发送](#)

## [第4章 网络监视功能](#)

### [4.1 Dashbord](#)

### [4.2 Application Response Time \(ART\)](#)

## [第5章 数据报文解码详解](#)

### [5.1 数据报文分层](#)

### [5.2 以太报文结构](#)

### [5.3 IP协议](#)

### [5.4 ARP协议](#)

### [5.5 PPP0E协议](#)

### [5.6 Radius协议](#)

**关键词：**Sniffer 协议分析

**摘 要：**本文对Sniffer软件的功能和使用作了简要的介绍，讲述了利用工具软件解决问题的思路和一些分析方法。

**缩略语清单：**

**参考资料清单：**

# 第1章 Sniffer软件简介

## 1.1 概述

Sniffer软件是NAI公司推出的功能强大的协议分析软件。本文针对用Sniffer Pro网络分析器进行故障解决。利用Sniffer Pro 网络分析器的强大功能和特征，解决网络问题，将介绍一套合理的故障解决方法。

与Netxray比较，Sniffer支持的协议更丰富，例如PPPOE协议等在Netxray并不支持，在Sniffer上能够进行快速解码分析。Netxray不能在Windows 2000和Windows XP上正常运行，Sniffer Pro 4.6可以运行在各种Windows平台上。

Sniffer软件比较大，运行时需要的计算机内存比较大，否则运行比较慢，这也是它与Netxray相比的一个缺点。

## 1.2 功能简介

下面列出了Sniffer软件的一些功能介绍，其功能的详细介绍可以参考Sniffer的在线帮助。

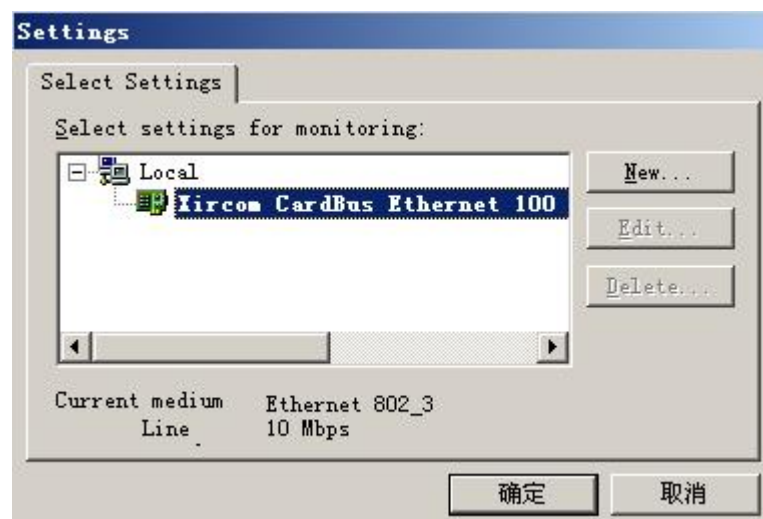
捕获网络流量进行详细分析

利用专家分析系统诊断问题

实时监控网络活动

收集网络利用率和错误等

在进行流量捕获之前首先选择网络适配器，确定从计算机的哪个网络适配器上接收数据。位置：File->select settings



选择网络适配器后才能正常工作。该软件安装在Windows 98操作系统上，Sniffer可以选择拨号适配器对窄带拨号进行操作。如果安装了EnterNet500等PPPOE软件还可以选择虚拟出的PPPOE网卡。对于安装在Windows 2000/XP上则无上述功能，这和操作系统有关。

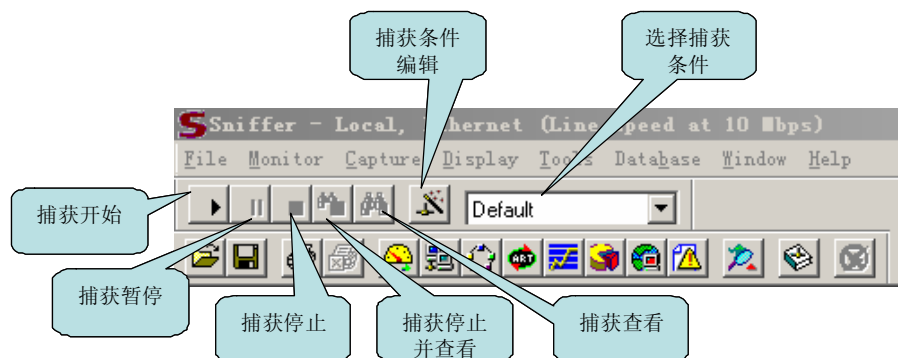
本文将对报文的捕获几网络性能监视等功能进行详细的介绍。下图为在软件中快捷键的位置。



## 第2章 报文捕获解析

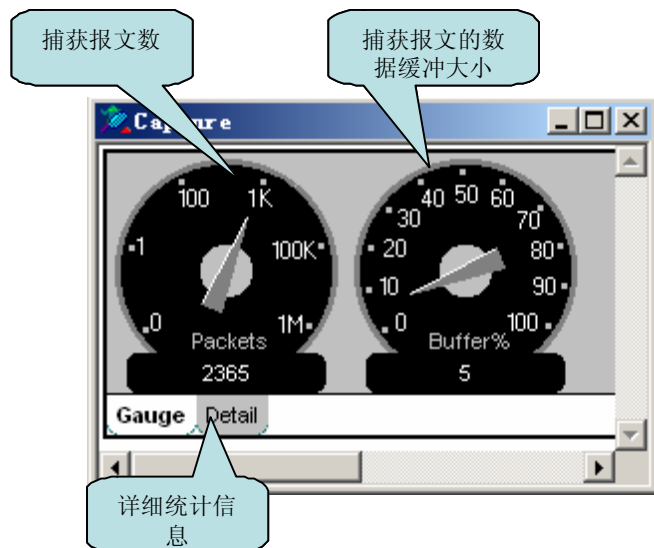
### 2.1 捕获面板

报文捕获功能可以在报文捕获面板中进行完成，如下是捕获面板的功能图：图中显示的是处于开始状态的面板



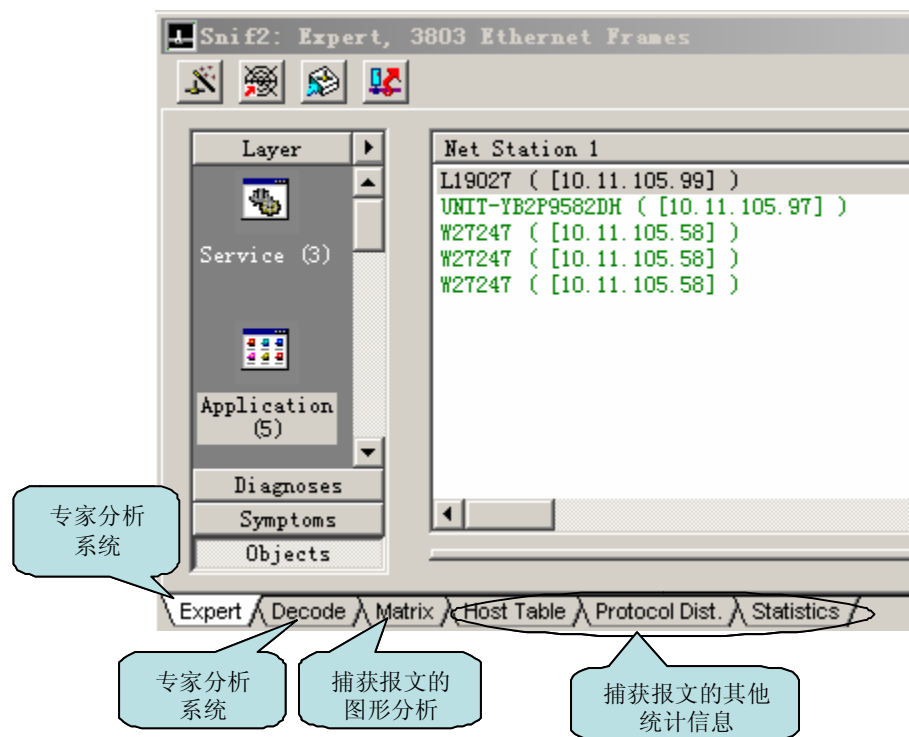
### 2.2 捕获过程报文统计

在捕获过程中可以通过查看下面面板查看捕获报文的数量和缓冲区的利用率。



### 2.3 捕获报文查看

Sniffer软件提供了强大的分析能力和解码功能。如下图所示，对于捕获的报文提供了一个Expert专家分析系统进行分析，还有解码选项及图形和表格的统计信息。

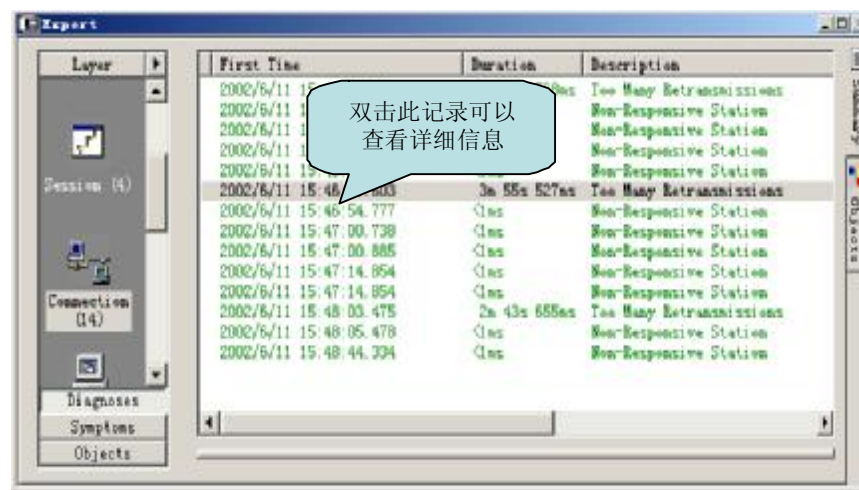


### 专家分析

专家分析系统提供了一个只能的分析平台，对网络上的流量进行了一些分析对于分析出的诊断结果可以查看在线帮助获得。

在下图中显示出在网络中WINS查询失败的次数及TCP重传的的次数统计等内容，可以方便了解网络中高层协议出现故障的可能点。

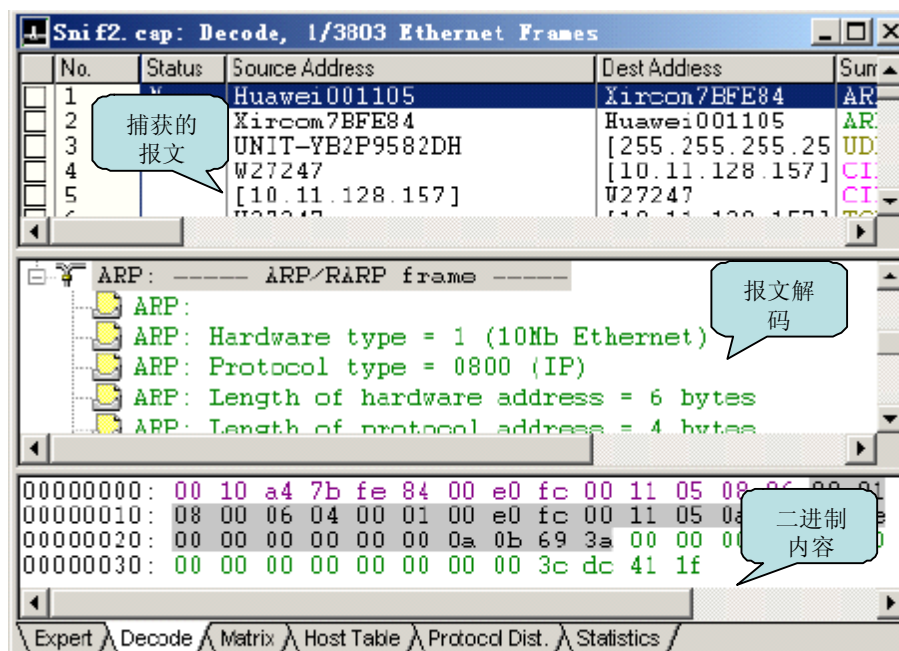
对于某项统计分析可以通过用鼠标双击此条记录可以查看详细统计信息且对于每一项都可以通过查看帮助来了解起产生的原因。



### 解码分析

下图是对捕获报文进行解码的显示，通常分为三部分，目前大部分此类软件结构都采用这种结构显示。对于解码主要要求分析人员对协议比较熟悉，这样才能看懂解析出来的报文。使用该软件是很简单的事情，要能够利用软件解码分析来解决问题关键是要对各种层次的协议了解的比较透彻。工具软件只是提供一个辅助的手段。因涉及的内容太多，这里不对协议进行过多讲解，请参阅其他相关资料。

对于MAC地址，Sniffier软件进行了头部的替换，如00e0fc开头的就替换成Huawei，这样有利于了解网络上各种相关设备的制造厂商信息。



功能是按照过滤器设置的过滤规则进行数据的捕获或显示。在菜单上的位置分别为 Capture->Define Filter和Display->Define Filter。

过滤器可以根据物理地址或IP地址和协议选择进行组合筛选。

### 统计分析

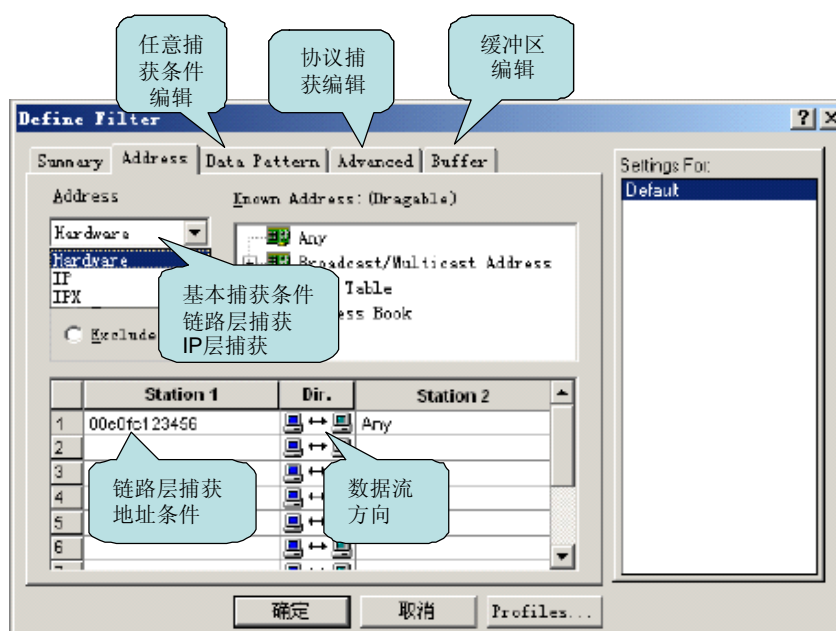
对于Matrix, Host Table, Portocol Dist. Statistics等提供了丰富的按照地址, 协议等内容做了丰富的组合统计, 比较简单, 可以通过操作很快掌握这里就不再详细介绍了。

## 2.4 设置捕获条件

### 基本捕获条件

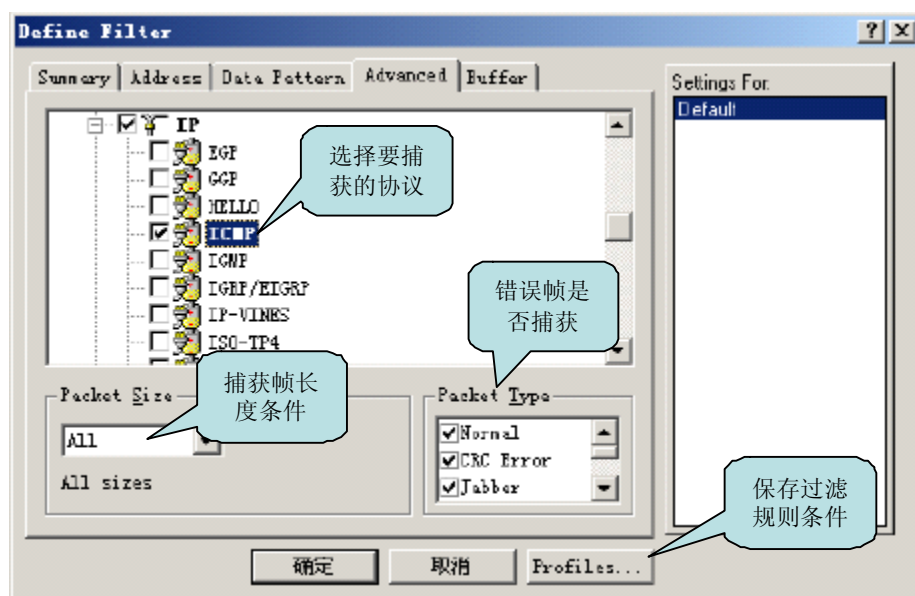
基本的捕获条件有两种：

- 1、链路层捕获, 按源MAC和目的MAC地址进行捕获, 输入方式为十六进制连续输入, 如: 00E0FC123456。
- 2、IP层捕获, 按源IP和目的IP进行捕获。输入方式为点间隔方式, 如: 10.107.1.1。如果选择IP层捕获条件则ARP等报文将被过滤掉。



## 高级捕获条件

在“Advance”页面下，你可以编辑你的协议捕获条件，如图：



### 高级捕获条件编辑图

在协议选择树中你可以选择你需要捕获的协议条件，如果什么都不选，则表示忽略该条件，捕获所有协议。

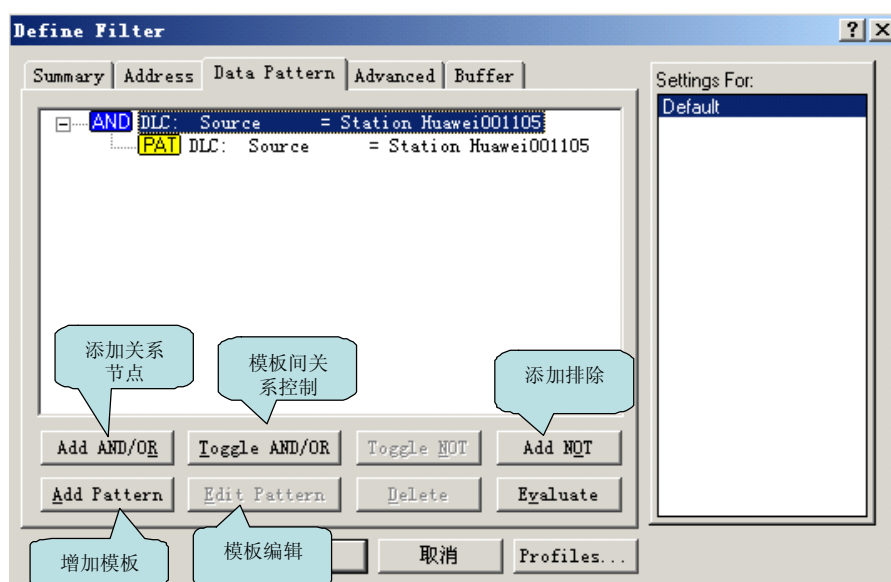
在捕获帧长度条件下，你可以捕获，等于、小于、大于某个值的报文。

在错误帧是否捕获栏，你可以选择当网络上有如下错误时是否捕获。

在保存过滤规则条件按钮“Profiles”，你可以将你当前设置的过滤规则，进行保存，在捕获主面板中，你可以选择你保存的捕获条件。

## 任意捕获条件

在Data Pattern下，你可以编辑任意捕获条件，如下图：



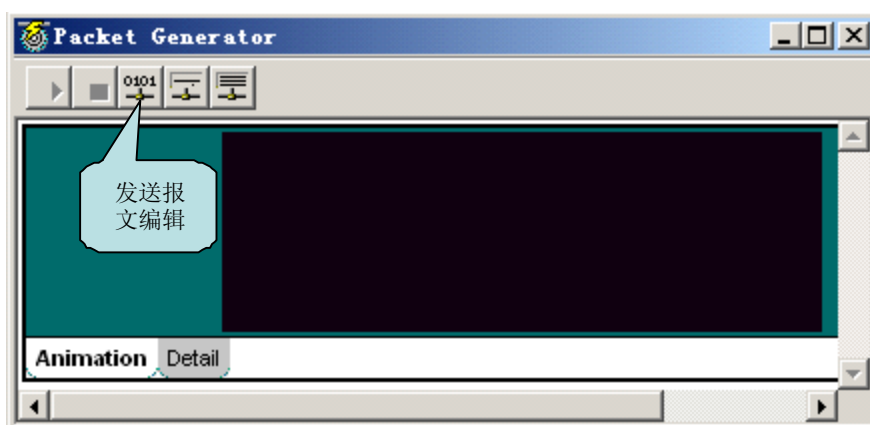
用这种方法可以实现复杂的报文过滤，但很多时候是得不偿失，有时截获的报文本就不多，还不如自己看看来得快。



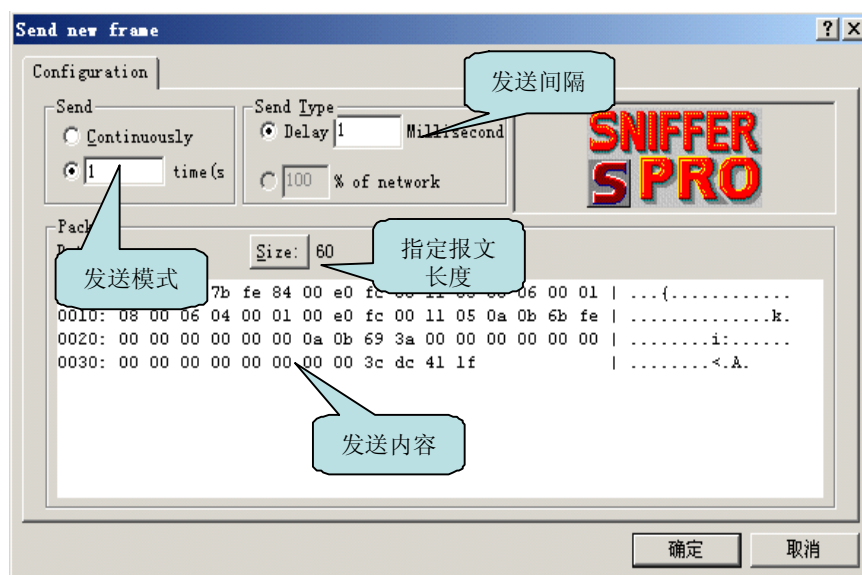
## 第3章 报文放送

### 3.1 编辑报文发送

Sniffer软件报文发送功能就比较弱，如下是发送的主面板图：



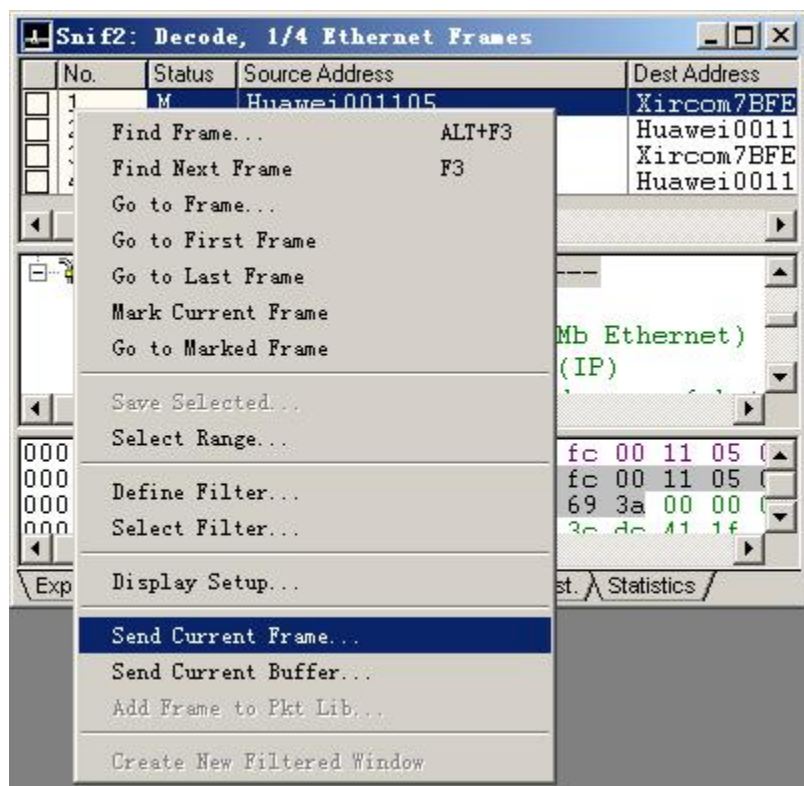
发送前，你需要先编辑报文发送的内容。点击发送报文编辑按钮。可得到如下的报文编辑窗口：



首先要指定数据帧发送的长度，然后从链路层开始，一个一个将报文填充完成，如果是NetXray支持可以解析的协议，从“Decode”页面中，可看见解析后的直观表示。

### 3.2 捕获编辑报文发送

将捕获到的报文直接转换成发送报文，然后修修改改可也。如下是一个捕获报文后的报文查看窗口：



选中某个捕获的报文，用鼠标右键激活菜单，选择“Send Current Packet”，这时你就会发现，该报文的内容已经被原封不动的送到“发送编辑窗口”中了。这时，你在修修改改，就比你全部填充报文省事多了。

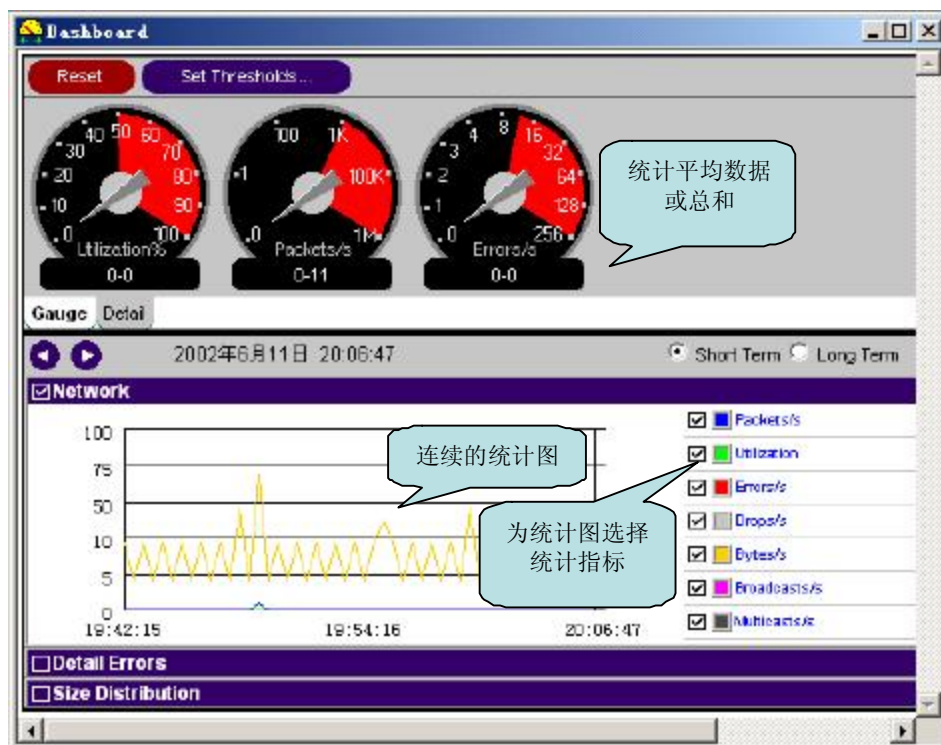
发送模式有两种：连续发送和定量发送。可以设置发送间隔，如果为0，则以最快的速度进行发送。

## 第4章 网络监视功能

网络监视功能能够时刻监视网络统计，网络上资源的利用率，并能够监视网络流量的异常状况，这里只介绍一下Dashbord和ART，其他功能可以参看在线帮助，或直接使用即可，比较简单。

### 4.1 Dashbord

Dashbord可以监控网络的利用率，流量及错误报文等内容。通过应用软件可以清楚看到此功能。



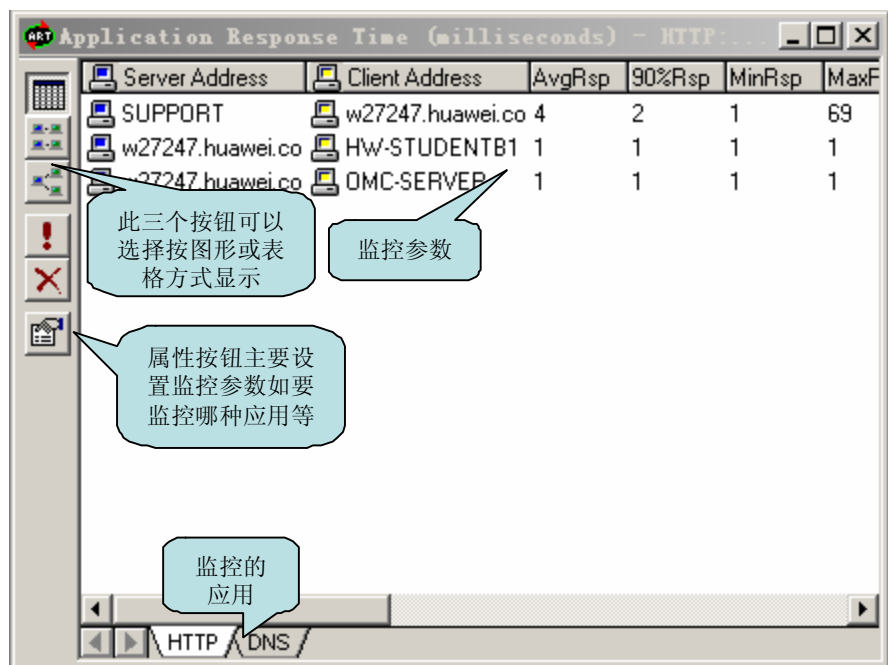
### 4.2 Application Response Time (ART)

Application Response Time (ART) 是可以监视TCP/UDP应用层程序在客户端和服务端响应时间，如HTTP,FTP,DNS等应用。

对与TCP/UDP响应时间的计算方法如下

TCP For each socket, ART stores the sequence numbers for packets sent by the client and waits for the corresponding ACK packets from the server. It then measures the time difference between the packet with the stored sequence number and the packet with the ACK to arrive at the response time.

UDP For each socket, ART measures the time between packets going from a client to a server and the next packet going from the server to the client.



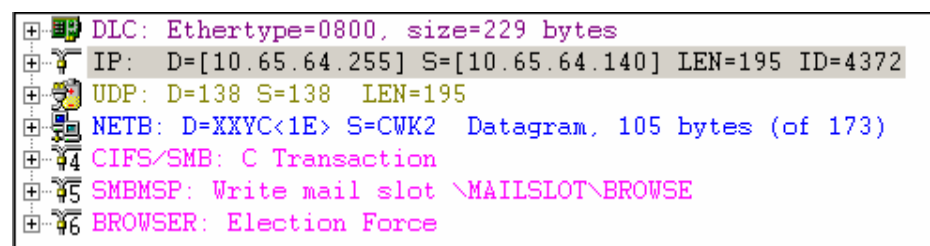
# 第5章 数据报文解码详解

本章主要对：数据报文分层、以太报文结构、IP协议、ARP协议、PPPOE协议、Radius协议等的解码分析做了简单的描述，目的在于介绍Sniffer软件在协议分析中的功能作用并通过解码分析对协议进一步了解。对其其他协议读者可以通过协议文档和Sniffer捕获的报文对比分析。

## 5.1 数据报文分层

如下图所示，对于四层网络结构，其不同层次完成不同功能。每一层次有众多协议组成。

应用层	Telnet FTP和e-mail等
传输层	TCP 和 UDP
网络层	IP ICMP IGMP
链路层	设备驱动程序及接口卡



如上图所示在Sniffer的解码表中分别对每一个层次协议进行解码分析。链路层对应“DLC”；网络层对应“IP”；传输层对应“UDP”；应用层对对应的是“NETB”等高层协议。Sniffer可以针对众多协议进行详细结构化解码分析。并利用树形结构良好的表现出来。

## 5.2 以太报文结构

EthernetII以太网帧结构

Ethernet\_II

DMAC	SMAC	Type	DATA/PAD	FCS
------	------	------	----------	-----

Ethernet\_II以太网帧类型报文结构为：目的MAC地址（6bytes）+源MAC地址+（6bytes）上层协议类型（2bytes）+数据字段（46-1500bytes）+校验（4bytes）。

Wireshark packet capture analysis of an FTP connection. The packet list shows an Ethernet II frame (No. 1) and an IP packet (No. 2). The packet details pane shows the Ethernet II frame structure with fields like Destination MAC (Station Xircom7BFE84), Source MAC (Station Huawei001105), and EtherType (0800). The packet bytes pane shows the raw data of the frame. Annotations highlight the 'Sniffer 自动添加时间戳' (Sniffer automatically adds timestamp), '目的MAC地址' (Destination MAC address), '源MAC地址' (Source MAC address), and '上层协议类型' (Upper layer protocol type).

总长度：单位字节，最大65535字节

标识：IP报文标识字段

标志：占3比特，只用到低位的两个比特

MF（More Fragment）

MF=1，后面还有分片的数据包

MF=0，分片数据包的最后一个

DF（Don't Fragment）

DF=1，不允许分片

DF=0，允许分片

段偏移：分片后的分组在原分组中的相对位置，总共13比特，单位为8字节

寿命：TTL（Time To Live）丢弃TTL=0的报文

协议：携带的是何种协议报文

1：ICMP

6：TCP

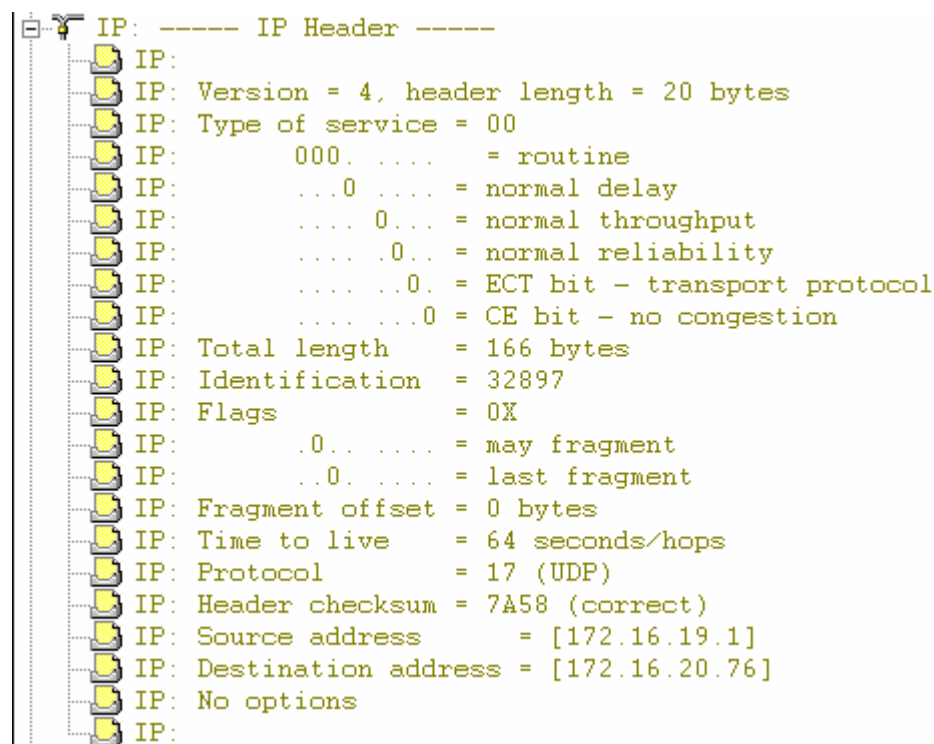
17：UDP

89：OSPF

头部检验和：对IP协议首部的校验和

源IP地址：IP报文的源地址

目的IP地址：IP报文的目的地址



上图为Sniffer对IP协议首部的解码分析结构，和IP首部各个字段相对应，并给出了各个字段值所表示含义的英文解释。如上图报文协议（Protocol）字段的编码为0x11，通过Sniffer解码分析转换为十进制的17，代表UDP协议。其他字段的解码含义可以与此类似，只要对协议理解的比较清楚对解码内容的理解将会变的很容易。

## 5.4 ARP协议

以下为ARP报文结构

硬件类型		协议类型
硬件长度	协议长度	操作 请求1, 回答2
发送站 硬件地址 (例如, 对以太网是6字节)		
发送站 协议地址 (例如, 对IP是4字节)		
目标 硬件地址 (例如, 对以太网是6字节)		
目标 协议地址 (例如, 对IP是4字节)		

ARP分组具有如下的一些字段:

**HTYPE (硬件类型)**。这是一个16比特字段, 用来定义运行ARP的网络的类型。每一个局域网基于其类型被指派给一个整数。例如, 以太网是类型1。ARP可使用在任何网络上。

**PTYPE (协议类型)**。这是一个16比特字段, 用来定义协议的类型。例如, 对IPv4协议, 这个字段的值是0800。ARP可用于任何高层协议。

**HLEN (硬件长度)**。这是一个8比特字段, 用来定义以字节为单位的物理地址的长度。例如, 对以太网这个值是6。

**PLEN (协议长度)**。这是一个8比特字段, 用来定义以字节为单位的逻辑地址的长度。例如, 对IPv4协议这个值是4。

**OPER (操作)**。这是一个16比特字段, 用来定义分组的类型。已定义了两种类型: ARP请求(1), ARP回答(2)。

**SHA (发送站硬件地址)**。这是一个可变长度字段, 用来定义发送站的物理地址的长度。例如, 对以太网这个字段是6字节长。

**SPA (发送站协议地址)**。这是一个可变长度字段, 用来定义发送站的逻辑(例如, IP)地址的长度。对于IP协议, 这个字段是4字节长。

**THA (目标硬件地址)**。这是一个可变长度字段, 用来定义目标的物理地址的长度。例如, 对以太网这个字段是6字节长。对于ARP请求报文, 这个字段是全0, 因为发送站不知道目标的物理地址。

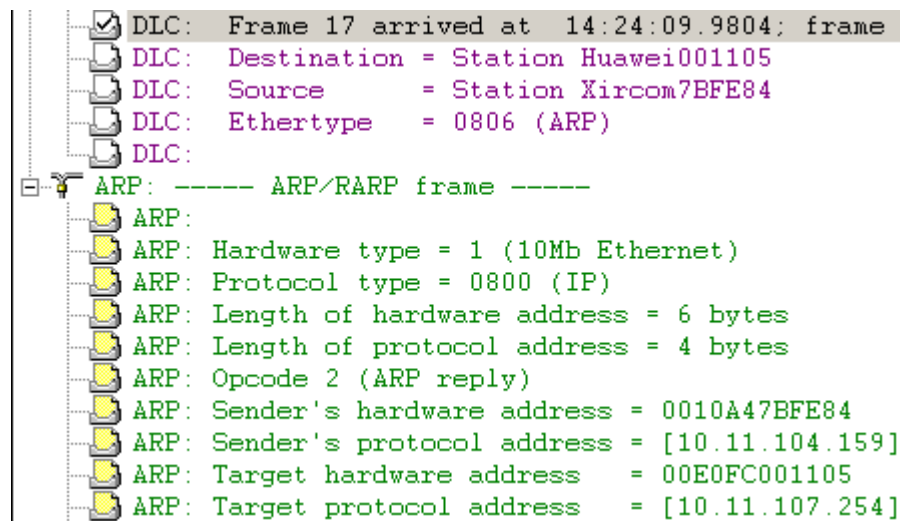
**TPA (目标协议地址)**。这是一个可变长度字段, 用来定义目标的逻辑地址(例如, IP地址)的长度。对于IPv4协议, 这个字段是4字节长。

```

DLC: Frame 16 arrived at 14:24:09 9803: frame
DLC: Destination = Station Xircom7BFE84
DLC: Source       = Station Huawei001105
DLC: Ethertype    = 0806 (ARP)
DLC:
ARP: ----- ARP/RARP frame -----
ARP:
ARP: Hardware type = 1 (10Mb Ethernet)
ARP: Protocol type = 0800 (IP)
ARP: Length of hardware address = 6 bytes
ARP: Length of protocol address = 4 bytes
ARP: Opcode 1 (ARP request)
ARP: Sender's hardware address = 00E0FC001105
ARP: Sender's protocol address = [10.11.107.254]
ARP: Target hardware address  = 000000000000
ARP: Target protocol address  = [10.11.104.159]

```





上面为通过Sniffer解码的ARP请求和应答报文的结构。

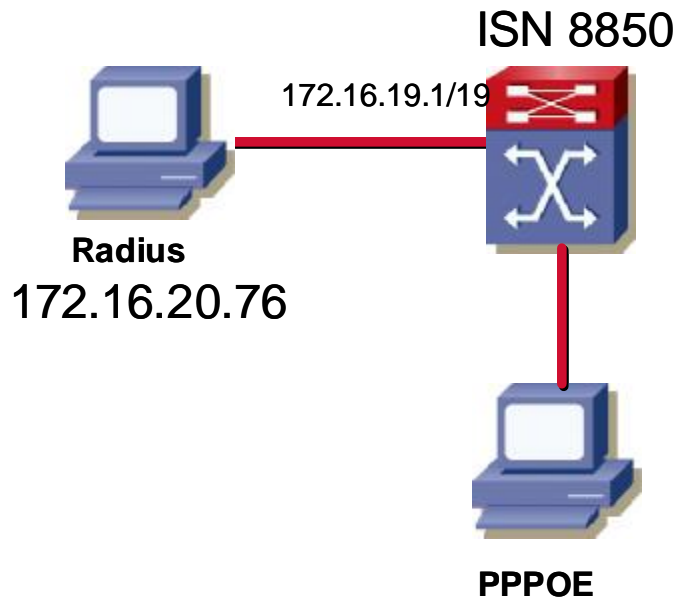
## 5.5 PPPOE协议

### PPPOE简介

简单来说我们可能把PPPOE报文分成两大块，一大块是PPPOE的数据报头，另一块则是PPPOE的净载荷（数据域），对于PPPOE报文数据域中的内容会随着会话过程的进行而不断改变。下图为PPPOE的报文的格式：

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
版 本				类 型				代 码				会 话 ID																			
长 度 域																净 载 荷															

- Y 数据报文最开始的4位为版本域，协议中给出了明确的规定，这个域的内容填充0x01。
- Y 紧接在版本域后的4位是类型域，协议中同样规定，这个域的内容填充为0x01。
- Y 代码域占用1个字节，对于PPPOE的不同阶段这个域内的内容也是不一样的。
- Y 会话ID占用2个字节，当访问集中器还未分配唯一的会话ID给用户主机时，则该域内的内容必须填充为0x0000，一旦主机获取了会话ID后，那么在后续的所有报文中该域必须填充那个唯一的会话ID值。
- Y 长度域为2个字节，用来指示PPPOE数据报文中净载荷的长度。
- Y 数据域，有时也称之为净载荷域，在PPPOE的不同阶段该域内的数据内容会有很大的不同。在PPPOE的发现阶段时，该域内会填充一些Tag（标记）；而在PPPOE的会话阶段，该域则携带的是PPP的报文。



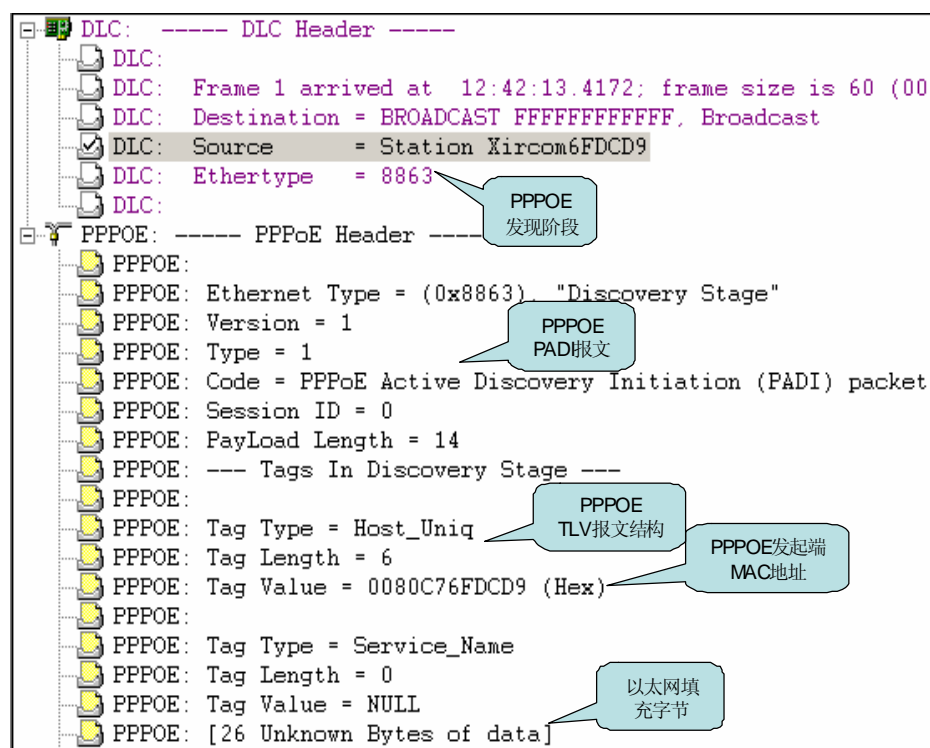
捕获报文测试用例图

如图所示，Radius Server IP地址为172.16.20.76。PPPOE用户Radius报文交互过程分析如下。

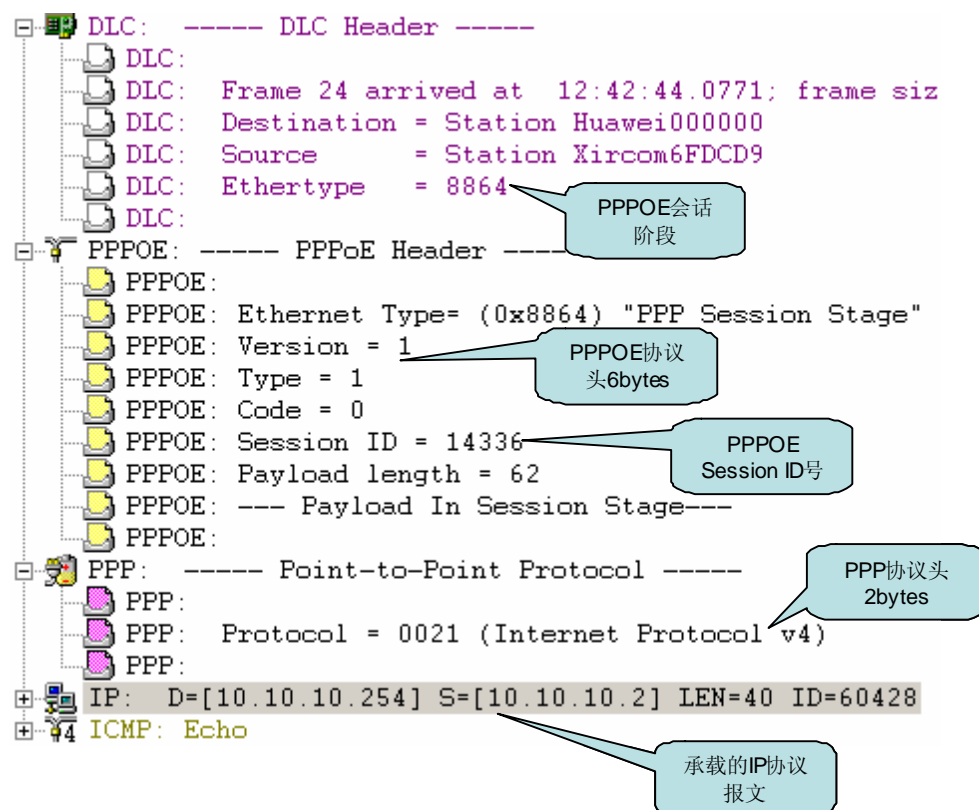
No.	SH	Source Address	Dest Address	Summary	
1	M	Xircom6FDCD9	Broadcast	PPPoE: : Discovery Stage	PPPOE发现阶段报文交互过程
2		Huawei000000	Xircom6FDCD9	PPPoE: : Discovery Stage	
3		Xircom6FDCD9	Huawei000000	PPPoE: : Discovery Stage	
4		Huawei000000	Xircom6FDCD9	PPPoE: : Discovery Stage	
5		Huawei000000	Xircom6FDCD9	PPP: LCP Configure Request	
6		Xircom6FDCD9	Huawei000000	PPP: LCP Configure Request	PPPOE LCP协商过程报文
7		Huawei000000	Xircom6FDCD9	PPP: LCP Configure Ack	
8		Xircom6FDCD9	Huawei000000	PPP: LCP Configure Request	
9		Huawei000000	Xircom6FDCD9	PPP: LCP Configure Ack	
10		Huawei000000	Xircom6FDCD9	PPP: LCP Configure Request	
11		Xircom6FDCD9	Huawei000000	PPP: LCP Configure Ack	PPPOE CHAP认证过程报文
12		Huawei000000	Xircom6FDCD9	CHAP: MESSAGE TYPE = Challenge	
13		Xircom6FDCD9	Huawei000000	CHAP: MESSAGE TYPE = Response	
14		Huawei000000	Xircom6FDCD9	CHAP: MESSAGE TYPE = Success	
15		Huawei000000	Xircom6FDCD9	PPP: IPCP Configure Request	
16		Xircom6FDCD9	Huawei000000	PPP: IPCP Configure Request	PPPOE IPCP协商过程报文
17		Xircom6FDCD9	Huawei000000	PPP: IPCP Configure Ack	
18		Huawei000000	Xircom6FDCD9	PPP: IPCP Configure Reject	
19		Xircom6FDCD9	Huawei000000	PPP: IPCP Configure Request	
20		Huawei000000	Xircom6FDCD9	PPP: IPCP Configure Nak	
21		Xircom6FDCD9	Huawei000000	PPP: IPCP Configure Request	
22		Huawei000000	Xircom6FDCD9	PPP: IPCP Configure Ack	
23		H28899	[10.10.10.255]	BROWSER: Announce Host H28899	
24		H28899	[10.10.10.254]	ICMP: Echo	
25		[10.10.10.254]	H28899	ICMP: Echo reply	

上图为PPPOE从发现阶段到PPP LCP协商，认证IPCP协商阶段和PPPOE会话阶段交互过程。

PPPOE发现阶段，PADI报文，Sniffer解码结构如下所示。



PPPOE会话阶段，Sniffer解码结构如下所示。



## 5.6 Radius协议

Radius报文简介

标准Radius协议包结构

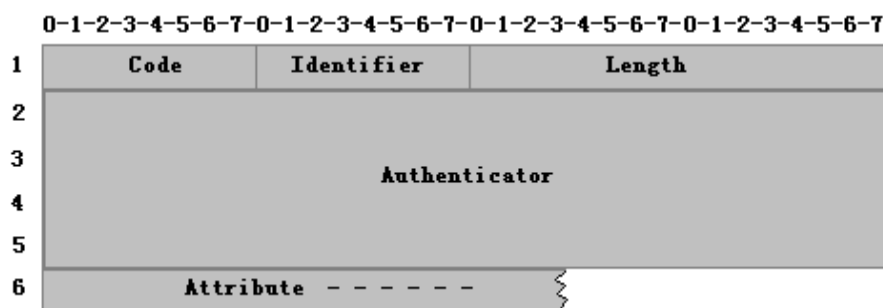


图9 Radius包格式

**Code:** 包类型；1字节；指示RADIUS包的类型。

- |     |                     |      |
|-----|---------------------|------|
| 1   | Access- request     | 认证请求 |
| 2   | Access- accept      | 认证响应 |
| 3   | Access- reject      | 认证拒绝 |
| 4   | Accounting-request  | 计费请求 |
| 5   | Accounting-response | 计费响应 |
| *11 | Access-challenge    | 认证挑战 |

**Identifier:** 包标识；1字节，取值范围为0 ~255；用于匹配请求包和响应包，同一组请求包和响应包的Identifier应相同。

**Length:** 包长度；2字节；整个包中所有域的长度。

**Authenticator:** 16 字节长；用于验证RADIUS服务器传回来的请求以及密码隐藏算法上。

该验证字分为两种：

1、请求验证字---Request Authenticator

用在请求报文中,必须为全局唯一的随机值。

2、响应验证字---Response Authenticator

用在响应报文中，用于鉴别响应报文的合法性。

响应验证字=MD5(Code+ID+Length+请求验证字+Attributes+Key)

**Attributes:** 属性

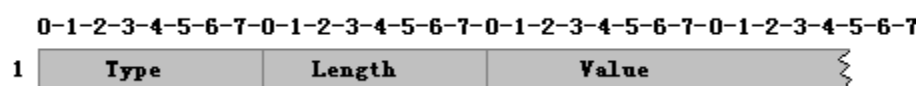
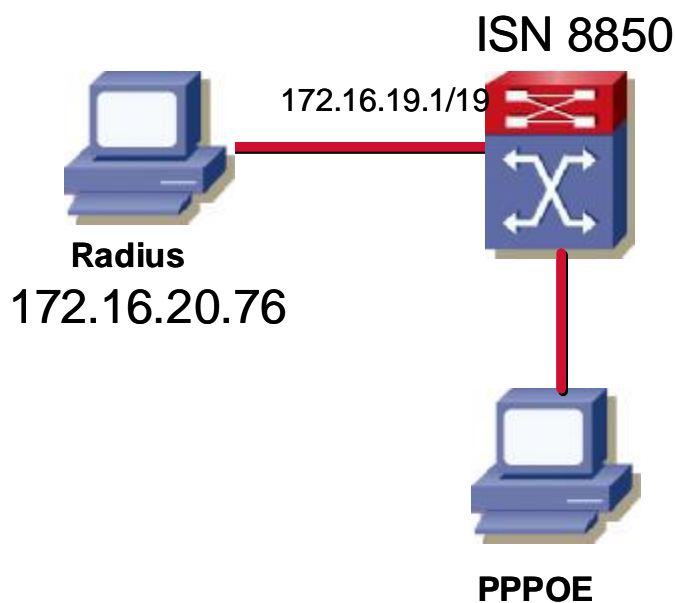


图10 属性格式

属性域是TLV结构编码。



测试用例图

下图为用户端PPPOE，Radius Server和BAS交互的认证上线和下线的过程。

No.	Seq	Source Address	Dest Address	Summary
1	M	[172.16.19.1]	[172.16.20.76]	RADIUS: Access-Request Id = 25
2		[172.16.20.76]	[172.16.19.1]	RADIUS: Access-Accept Id = 25
3		[172.16.19.1]	[172.16.20.76]	RADIUS: Accounting-Request Id = 26
4		[172.16.20.76]	[172.16.19.1]	RADIUS: Accounting-Response Id = 26
5		[172.16.19.1]	[172.16.20.76]	RADIUS: Accounting-Request Id = 27
6		[172.16.20.76]	[172.16.19.1]	RADIUS: Accounting-Response Id = 27

报文1: BAS请求Radius Server认证报文。

报文2: Radius Server回应BAS认证通过报文。

报文3: BAS计费请求报文。

报文4: Radius Server计费响应报文。

报文5: BAS计费结束报文。

报文6: Radius Server计费结束响应报文。

从中可以看出对于报文请求和响应是通过IP地址+Radius 协议域中ID号进行配对识别的。

UDP: D=1645 S=1812 LEN=146

RADIUS: ----- RADIUS HEADER -----

- RADIUS: Code = 1 (Access Request)
- RADIUS: Identifier = 25
- RADIUS: Length = 133
- RADIUS: Authentication = 26fe8768341de68a72a1276771e1c1ca
- RADIUS: Attributes follow
- RADIUS: User-Name = 'test'
- RADIUS: CHAP Identifier = 1
- RADIUS: CHAP Response = 'Hx. |. Œý.Vúd" |Hc>>"
- RADIUS: CHAP Challenge = '&þ|h4.æ|ri'gqáÁÊ"
- RADIUS: Service-Type = 2 (Framed)
- RADIUS: Framed-Protocol = 1 (PPP)
- RADIUS: NAS-IP-Address = [172.16.19.1]
- RADIUS: NAS-Identifier = 'ESR'
- RADIUS: NAS-Port = (72447644), should be between 0 and 65535
- RADIUS: NAS-Port-Type = 16 (XDSL - Digital Subscriber Line)
- RADIUS: Unknown attribute 07

BAS发起 Radius认证 请求报文

Radius属性字段TLV结构, Sniffer只在此对属性类型和Value进行了解码分析, 其TLV十六进制数值可在下面源码中获得

对于Sniffer不能解析的属性类型可以同构源码对应 ASCII进行翻译, 如此属性可以从ASCII码翻译中获得该用户端口信息

```

00000070: 05 00 00 00 01 04 06 a2 01 13 01 20 05 4b 5a 62 71 71 4a
00000080: 05 06 34 00 7e 9c 3d 06 00 00 00 10 57 28 73 6c 74 3d
00000090: 6f 74 3d 31 33 3b 73 75 62 73 6c 6f 74 3d 30 3b
000000a0: 71 6f 72 74 3c 31 3b 3b 4c 41 4e 44 44 3d 32 33 3b
000000b0: 33 3c 3b 3b
  
```

上图显示了BAS发起的Radius认证请求（Code=1）报文的结构。Radius报文是承载在UPD协议之上的，这里我没不关注上层报文的结构。

下图为PPPOE CHAP 认证过程的Radius认证请求报文和PPPOE中CHAP认证的Challenge 报文。通过比较可以方便看出 BAS 发出的 Challenge 值为“26fe8768341de68a72a1276771e1c1ca”与PPPOE中CHAP认证过程中BAS发给PPPOE用户的Challenge值是一致的。

RADIUS: Attributes follow

- RADIUS: User-Name = "test"
- RADIUS: CHAP Identifier = 1
- RADIUS: CHAP Response = "Hx. |. Œý.Vúd" |Hc>>"
- RADIUS: CHAP-Challenge = "&þ|h4.æ|ri'gqáÁÊ"
- RADIUS: Service-Type = 2 (Framed)
- RADIUS: Framed-Protocol = 1 (PPP)
- RADIUS: NAS-IP-Address = [172.16.19.1]
- RADIUS: NAS-Identifier = "ESR"

BAS发给Radius的 CHAP-Challenge属性

```

00000020: 14 4c 07 14 06 6d 00 92 b7 df 01 19 00 8a 26 fe
00000030: 87 68 34 1d e6 8a 72 a1 27 67 71 e1 c1 ca 01 06
00000040: 74 65 73 74 03 13 01 48 78 2e 8b 1d d5 fd 03 56
00000050: fa 64 a8 80 48 a2 bb 3c 12 26 fe 87 68 34 1d e6
00000060: 8a 72 a1 27 67 71 e1 c1 ca 06 06 00 00 00 02 07
00000070: 06 00 00 00 01 04 06 ac 10 13 01 20 05 4b 5a 62
00000080: 05 06 34 00 7e 9c 3d 06 00 00 00 10 57 28 73 6c
00000090: 6f 74 3d 31 33 3b 73 75 62 73 6c 6f 74 3d 30 3b
000000a0: 71 6f 72 74 3c 31 3b 3b 4c 41 4e 44 44 3d 32 33
  
```

CHAP: ----- CHAP Header -----

CHAP:

CHAP: Code = 1 (Challenge Packet)

CHAP: Identifier = 1

CHAP: Packet length = 21, DOES NOT EQUAL ACTUAL 42

CHAP:

CHAP: ----- CHAP data -----

CHAP:

CHAP: Value Size = 16

CHAP: Challenge Value = 26FE8768341DE68A72A1276771E1C1CA

CHAP: Name = ""

BAS发给PPP用户的Challenge的值

```

00000000: 00 80 c7 6f dc d9 00 e0 fc 00 00 00 88 64 11 00 .e莖莖.!
00000010: 00 38 00 17 c2 23 01 01 00 15 10 26 fe 87 68 34 .8..?...
00000020: 1d e6 8a 72 a1 27 67 71 e1 c1 ca 00 00 00 00 00 .莖r?gq
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

下图为PPPOE用户发给BAS的经过CHAP加密后的用户密码和BAS发给Radius Server中认证请求报文用户秘密属性域的比较。可以看出在Radius 认证过程中，BAS设备将Challenge属性和用户加密后的密码发给Radius进行验证。

CHAP: ----- CHAP Header -----

CHAP:

CHAP: Code = 2 (Response Packet)

CHAP: Identifier = 1

CHAP: Packet length = 25, DOES NOT EQUAL ACTUAL 38

CHAP:

CHAP: ----- CHAP data -----

CHAP:

CHAP: Value Size = 16

CHAP: Response Value = 48782E8B1DD5FD0356FA64A88048A2BB

CHAP: Name = "test"

PPPOE用户发给BAS的经过CHAP加密后的用户密码

```

00000000: 00 e0 fc 00 00 00 00 80 c7 6f dc d9 88 64 11 00 .莖...
00000010: 00 38 00 1b c2 23 02 01 00 19 10 48 78 2e 8b 1d .8..?...
00000020: d5 fd 03 56 fa 64 a8 80 48 a2 bb 74 65 73 74 00 .莖.V莖
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

RADIUS: Attributes follow

RADIUS: User-Name = "test"

RADIUS: CHAP Identifier = 1

RADIUS: CHAP Response = "Hx. .Öý.Vúd" [Hx. .Öý.Vúd]

RADIUS: CHAP-Challenge = "&p|h4.æ|ri'gqáÁE"

RADIUS: Service-Type = 2 (Framed)


BAS发给Radius的经过CHAP加密后的用户密码  
(将用户端PPP过程发出密码直接添加到Radius该属性中)

```


00000020: 14 4c 07 14 06 6d 00 92 b7 df 01 19 00 8a 26 fe
00000030: 87 68 34 1d e6 8a 72 a1 27 67 71 e1 c1 ca 01 06
00000040: 74 65 73 74 03 13 01 48 78 2e 8b 1d d5 fd 03 56
00000050: fa 64 a8 80 48 a2 bb 3c 12 26 fe 87 68 34 1d e6
00000060: 8a 72 a1 27 67 71 e1 c1 ca 06 06 00 00 00 02 07
00000070: 06 00 00 00 01 04 06 ac 10 13 01 20 05 45 53 52
00000080: 05 06 34 00 7e 9c 3d 06 00 00 00 10 57 28 73 6c

```

通过比较可以清楚了解协议各字段含义相互关系，为问题处理提供有效的手段。  
下面为PPPOE用户Radius认证的Sniffer捕获的报文。



chap-client.cap



RadiusServer.cap

