ET Protocol Reporter

Deign Document 1.0

概要设计说明书

1 引言

1.1 编写目的

编写此设计说明书，首先是为了让大赛评委能更好的了解本系统的整体结构。其次，为了指导开发，让每一名开发者保持一致、明确的思路并保存开发经验。

1.2 背景

如今网络世界不再太平，各种各样的恶意程序无时无刻不在窥探着人们的电脑，企图盗走任何机密的信息，严重的威胁着网络用户的安全。计算机专家作了大量工作与网络入侵者展开较量。人们发现，将这些恶意程序据之局域网门外的最好方法就是在网关就将企图进入内网的数据包截获，再将收集到的信息送回到服务端作进一步分析。本软件就是为了实现以上所述的功能而设计的。

2 总体设计

2.1 需求分析

2.1.1 系统功能要求

本系统将实现以下功能：

网关

1. 收集流经的数据包
2. 分析流入局域网数据包的应用层协议类型
3. 分析流入局域网数据包传输的文件类型
4. 获得数据包的目标 ip 和源 ip
5. 统计一定时间段内各种协议下的数据流量
6. 统计一定时间段内所传输的各种文件类型的流量
7. 定时将统计结果发给服务器

服务器

1. 收集网关发来的统计信息数据包
2. 根据收集到的数据和用户的选择实时显示协议分布图、文件类型分布图、ip 地址分布图、拓扑图、攻击路径图，同时还能查看数据包的具体信息列表。

性能需求

要求能够截获数据包，并准确辨别数据包应用层协议类型和文件类型。保证网关与服务器之间数据的传输安全、快捷，使系统能在比较恶劣的网络环境下进行通信。能够自动维护系统运行过程中在内存和硬盘上建立的缓冲空间，节约系统资源，保证计算机能平稳运行本系统。

容错需求

有容错系统，当意外发生时能通知用户系统出错状况，并自动处理该错误。

能自动处理由于网络交通问题或其他 IO 问题造成的错误。

有日志系统，能自动记录系统运行时出现的错误以备检查。

2.2 运行环境

操作系统 ：Microsoft Windows NT\2000\XP

集成环境支持：Microsoft .NET Framework 2.0 Runtime

2.3 开发环境

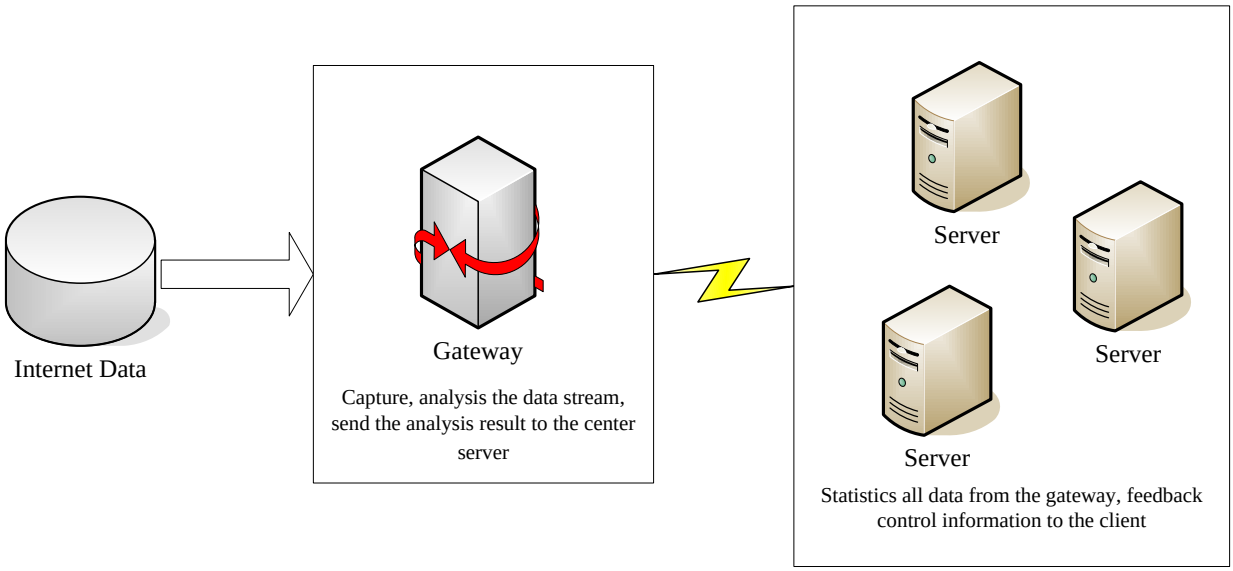
操作系统：Microsoft Windows NT\2000\XP

开发环境：Microsoft .NET Framework 2.0 SDK

Visual Studio.net 2005

开发语言：C# .net

2.4 系统结构



系统总体结构图

如图所示，本系统分为 Server 端和 Client 端两部分，系统将在 Client 端将收集到的数据包进行统计，然后将统计结果发到 Server 端，Server 端收到数据后，会根据收到的信息生成相应的统计图和表格。

3 模块设计说明

3.1 截包模块设计说明

3.1.1 描述

本模块用于截获并保存数据包，为下一步分析做准备。

3.1.2 功能

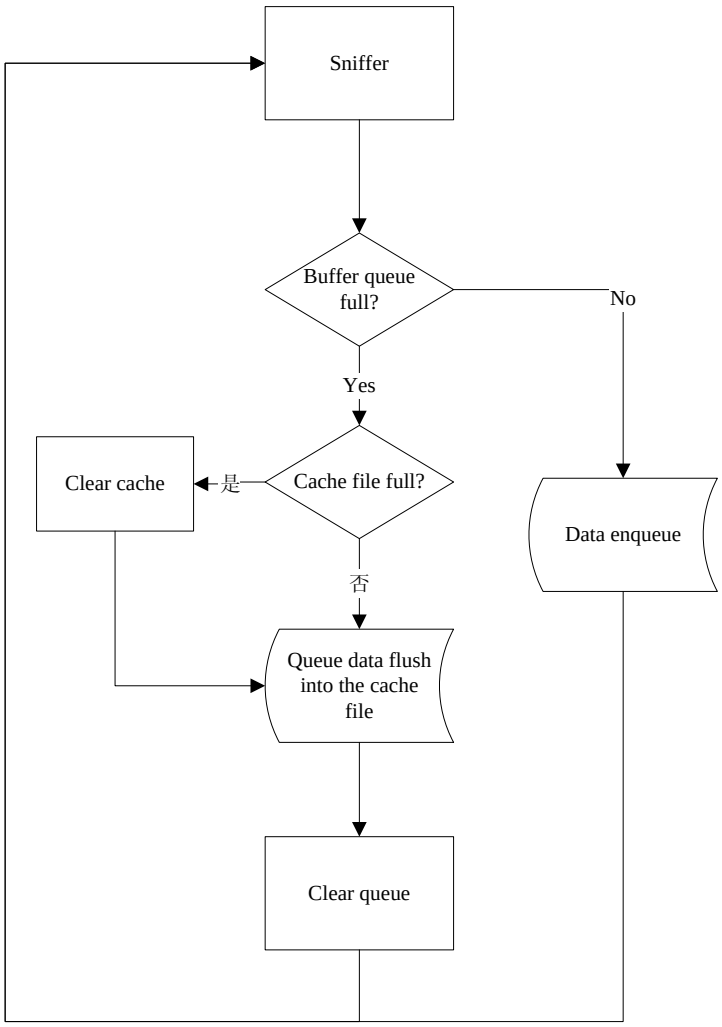
截获流入的数据包并保存在缓冲区，对缓冲区进行维护。以队列形式组织内存缓冲区中的数据，当缓冲区被写满时，自动将缓冲区中的数据写入文件（此时读取数据包的线程将转为读取文件）。

3.1.3 性能

接收并保存数据包时，如果数据包增加的速度大于处理的速度，导致队列满，能自动将旧数据包存入文件中，如果队列再一次加满，则应能再将队列中的旧包转到文件尾部，确保收到的每一个数据包都能得到分析。如果文件被写满（上限为 500m）的情况发生，则将文件清空，丢掉文件中待处理的数据，以免文件无限制扩大。

3.1.4 流程逻辑

用图表（例如流程图、判定表等）辅以必要的说明来表示本程序的逻辑流程。



流程图

该模块将完成类似 sniffer 的功能，截获每一个流入局域网的 ip 包，再将其保存在一个队列中。RawSocket 类 Start()函数被调用后，将调用私有的回调函数 CallReceive()不停收包，再将收到的包保存在队列中，如果队列已满，则将队列中的内容保存在 Cache 文件中，并清空队列。

3.1.5 接口

截包模块中的 RawShocket 类里有两个 Public 级别的方法 Start() 和 Shutdown() 可供外部程序调用以启动或关闭回调函数。

3.1.6 存储分配

保存队列需要 50 兆的内存空间

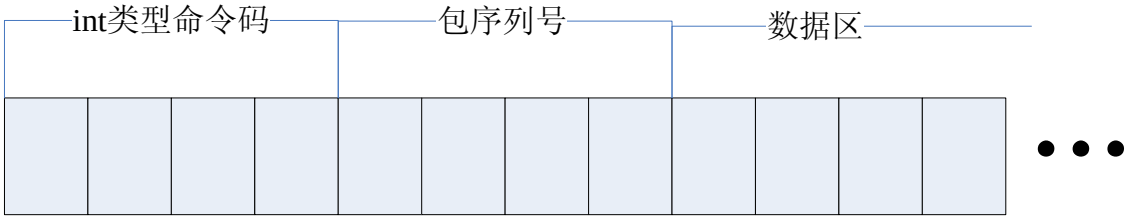
保存文件 500 兆的硬盘空间

3.2 传输模块设计说明

3.2.1 程序描述

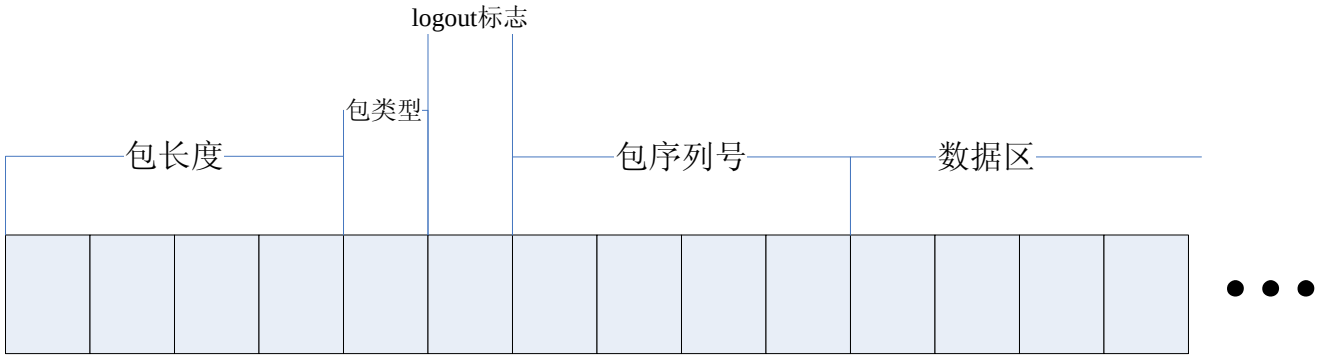
ExceedUDP 是本系统的传输模块，系统在 Client 端与 Server 端以 UDP 包进行数据传输。在 UDP 的上一层还有自定义的软件内部交流的协议。数据包包头格式定义如下：

server 发给 client 的包头里,前四位是一个 int 类型的命令码,后四位是一个序列号,表示 client 端发给 server 的有此序列号的包被收到。



client 发给 server 的包头里,前四位是包的长度,接着一位是包的类型,接

着一位表示该包是否为 logout 包,接着四位是在分包传输时该包的序列号。



给出对该程序的简要描述，主要说明安排设计本程序的目的意义，并且，还要说明本程序的特点（如 是常驻内存还是非常驻？是否子程序？是可重人的还是不可重人的？有无覆盖要求？是顺序处理还是并发处理等）。

3.2.2 功能

数据包传输有可靠传输和不可靠传输两种模式，在可靠传输模式下，接包的一端接到包后将自动向发送端发送确认信息，如果发包端在规定时间内没有收到确认，就会重发该包。在不可靠模式下，接收端收到包后不发回确认信息，发送端将包发出后，就不再重发该包。

3.2.3 性能

本模块使用 UDP 包是为了使数据传输更加灵活、快捷，同时尽可能确

保每一个 Client 发向 Server 端的包都能正常到达目的地，发生丢包时，应能够自动及时重发。

3.3 分析模块设计说明

3.3.1 定义

分词：将字符串中由空格、换行等分隔符隔开的字符串当成一个单词，并统计出相同单词出现的频率。

关键字权重表：在不同的协议或文件类型的数据包中，不同的单词的出现频率是不同的，比如在使用 http 协议传递文件的数据包中，出现 http 字样的可能性明显比其它字符串的出现频率要高。在建立分析模块的时候，我们通过实验的方法找出在某个协议或文件类型的传输中特定关键字的权重，每种支持的协议或文件类型都有一个关键字权重表。关于关键字权重表的获得方法将在算法一节中详细介绍。

归一化：将大于一的数组化为一组大于 0 小于 1 的小数。

3.3.2 程序描述

分析模块用于分析截获的数据包中包含的信息，以确定传输协议和文件类型等。这个是最难完成的模块。由于题目规定不可以使用端口号去猜

测协议类型，所以我们最终决定对每个数据包包头前若干个字节进行分析再用分析结果进行判断以确定数据包传输的协议和文件类型。

分析模块的主要部分是协议分析。

3.3.3 功能

通过对包头提取数据的分析，确定确定传输协议、文件类型等并将数据进行统计。

3.3.4 性能

尽量精确地确定协议类型和文件类型，应尽量减少误判。如果待处理的数据过多，能自动调整处理粒度，保证数据处理的速度，尽量避免缓冲区被写满的情况发生

3.3.5 算法

先介绍生成分析模块所需的关键字权重表的原理。我们使用的是用计算机对数据进行训练的方法。

首先从协议标准文档（RFC）里提取协议关键字（如命令格式，回复代码等出现在协议头的关键字），并计算出每个关键字的内熵 InEntropy（可以描述同一关键字在不同协议中出现频率），如果该值越小，表示它所在的协议数越小，则对协议的分辨能力越强，由内熵值可以计算关键词

的初始权重如下：

$$W_{i0} = e^{-\mathrm{InEntropy}(i)-0.5}$$

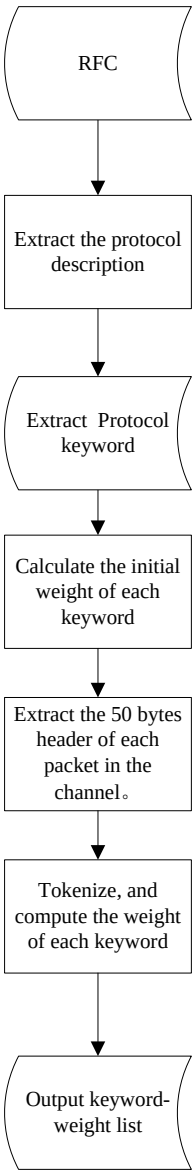
其次，针对每一种协议 T 收集大量的数据包(通过自己开发的采集器收集)，并对从数据包中截取的 50 字节进行分词处理，计算每个词的 TF（词频），为了减少错误数据的影响，我们将对 TF 求正弦值，则关键词 i 能证明数据包是该协议的权重为：

$$W_i = W_{i0} + \mathrm{Sin}(TF_i / M)$$

对于没有在协议标准文档出现的关键字，其初始权重为 0：

$$W_{i0} = 0$$。

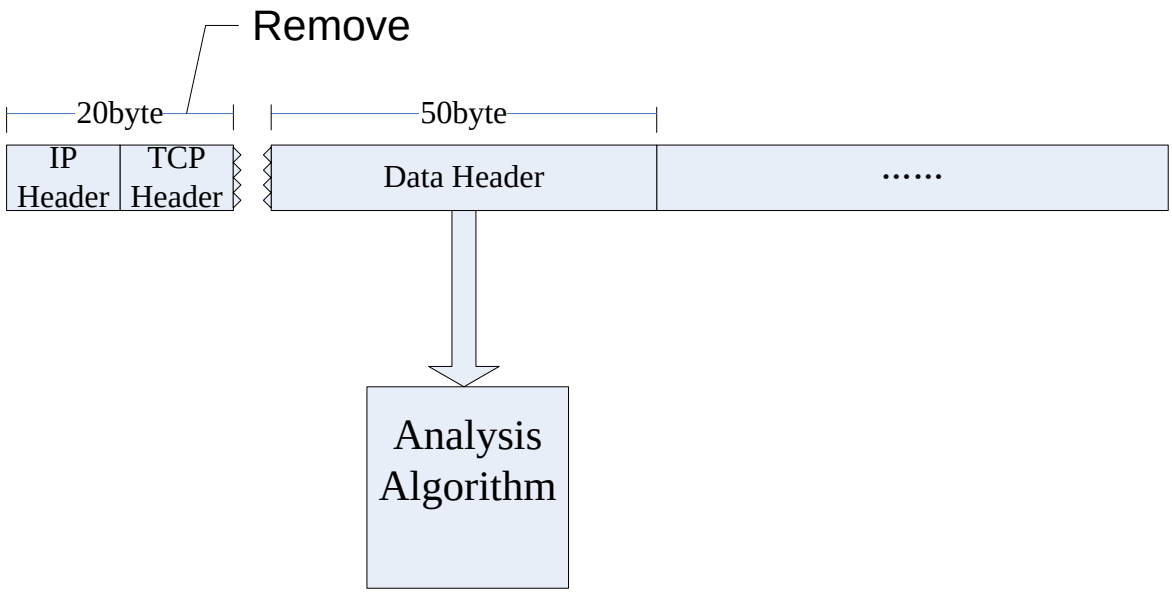
由以上得到协议所有关键字的权重，排序后取前 100 位为该协议的判断关键字。



关键字权重表生成示意图

收到数据包后系统会先判断是否为 TCP 包，系统只对 TCP 包进行分析。不是 TCP 包的将被丢弃。之后再判断传输该 TCP 包的会话类型是否确定，如果确定则无需进一步分析。如果 session 类型不确定，则对该包进行如下处理。

我们先将数据包前面的 IP 和 TCP 头剥除，再提取出剩余部分的前 50 个字节数据，数据包处理如下图所示：



数据包处理示例

将提取出的数据送入分词模块，对提取到的关键字进行统计，得出在包中提取出来的数据中各个关键字出现的频率（以下简称词频）。

然后在各个协议和文件类型的关键字列表中查找相应关键字，把关键字权重和词频相乘之后累加得到该协议或文件类型的得分。最后将得分归一化。

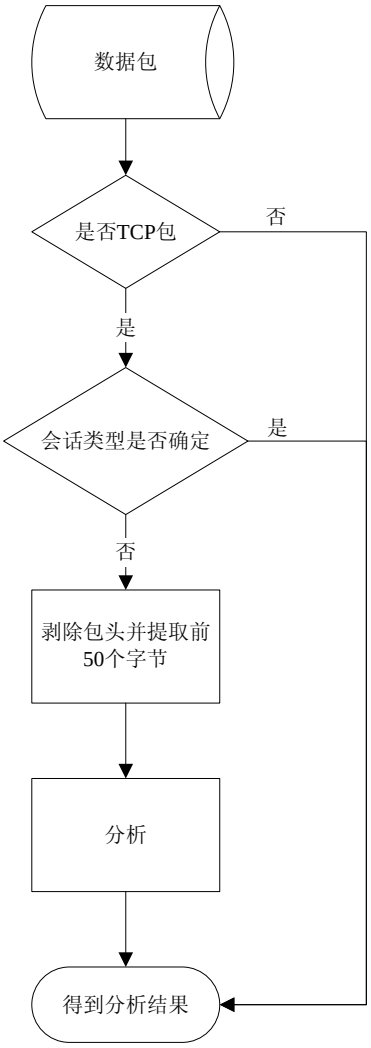
在计算文件或协议类型得分的时候，我们并不单单采用上述线性累加的方法，我们还引入一种相关字机制。这个机制在协议分析和文件类型分析中有所不同。在协议分析中，只要发现一些特定的关键字相邻出现，则认为该数据包使用某种协议的可能性会大大增加，因此相关字的分值（或者说权重）会比原来两个关键字单独出现的分值（或权重）之和加倍。而

在文件类型分析中，分析模块只要发现一些特定的命令关键字（如 “get”），就会在往下 50 个字节里搜索点号 ”.”，再查看点号后面的关键字是否为文件类型的关键字，如果是的话，则认为该数据包传输该类型文件的可能性极大，因此认定该类型为该包传输的文件类型。

通过上述方法获得协议或文件类型得分后，将分数归一化。对于文件类型分析来说，只需向 Sever 端发送已认定的或得分最高的类型即可。但对于协议类型分析来说还不能这么简单就结束。为了得到更高的正确率，我们在这里又引入另一种机制。在这里 Client 端选出得分最高的三个协议类型发给 Sever。Sever 中有个小的分析模块将对收集到的数据进行分析，对于通过同一个端口进入 Client 的所有数据包得分前三名的协议类型的分数进行累加，累加后分数最高的协议才能被认定为是这些包的传输协议。

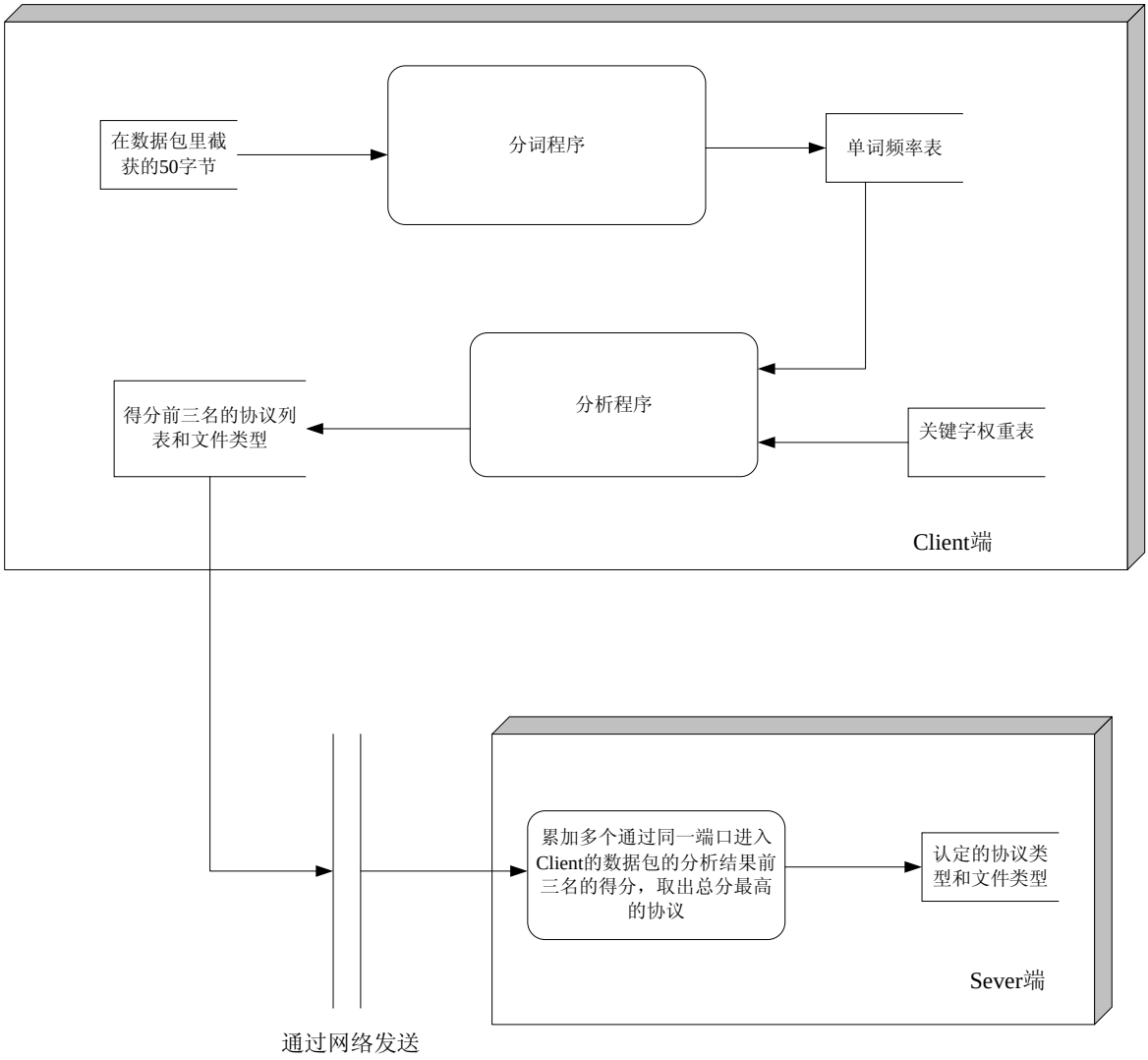
上述协议分析机制已被测试结果证实为有效，在我们自己的测试中，协议的分辨率达到 98%以上。

3.3.6 流程逻辑



流程图

如上图所示，分析模块会先查看 TCPflag，若为 false 则将该包丢弃，直接返回读取下一个数据包，如果为 TCP 包，则进一步判断会话（session）类型。如果会话类型确定，则返回该类型；如果会话类型不确定，则将数据包处理后传给分词程序作进一步分析。数据传给分词程序后的流程如下图所示：


在数据包里截获的50字节
分词程序
单词频率表
分析程序
得分前三名的协议列表和文件类型
关键字权重表
Client端
通过网络发送
累加多个通过同一端口进入Client的数据包的分析结果前三名的得分，取出总分最高的协议
认定的协议类型和文件类型
Sever端

数据流程图