网络数据包捕获及协议分析系统的UML建模*)

蔡小华[1] 王汝传[1,2] 任勋益[1]

(南京邮电大学计算机学院 南京 210003)[1] (南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 为了使数据包捕获及协议分析系统易于从整体上进行功能分析并适应需求变化,本文将UML建模技术应用于网络数据包捕获及协议分析系统之中,首先提出该系统的一种体系结构,并对其部件功能进行了分析。接着运用UML技术建立系统的静态模型和动态模型,并实现了该系统。最后对所建立系统的优点进行了分析总结。

关键词 UML,捕获,动态建模,特征,数据包

UML Modeling of Packet Capture and Protocol Analysis System

CAI Xiao-Hua[1] WANG Ru-Chuan[1,2] REN Xun-Yi[1]

(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003)[1]

(State Key Laboratory for Novel Software Technology Nanjing University, Nanjing 210093)[2]

Abstract In order to system functional analysis easily and adapted to changing requirement, UML modeling technology is applied in Packet Capture and Protocol Analysis System. The system architecture is proposed first, functional analysis is given for its components. Static model and dynamic model are separately founded by UML and is implemented. An analysis of the advantages of the system is placed in the end.

Keywords UML, Capture, Dynamic modeling, Character, Packet

1 引言

互联网在为人们工作带来便利的同时,产生了日趋严重的安全问题。如何保护网络的信息安全,防范来自外部网络的黑客和非法人侵者的攻击,建立起强健的网络信息安全防范系统,是亟需解决的一个重要难题。防火墙、入侵检测系统是解决这类网络安全问题最重要手段。数据包捕获技术是这些网络安全产品的基础。因此,研究如何实现数据包捕获具有重要的意义。

UML是一种编制系统蓝图的标准化语言,可以对大型复杂系统做可视化的说明并构造系统模型,以及建立各种必要的文档。它提供了一套描述面向对象软件系统模型的概念和图形表示法,以支持面向对象的技术和方法,体现面向对象分析与设计风格。UML适用于以面向对象技术来描述任何类型的系统,而且适用于系统开发的不同阶段,从需求规格描述直到系统完成后的测试或维护的全过程,已经被工业标准组织OMG(Object Management Group)采纳为标准。UML通过三类图形建立系统模型:用例图、静态结构图(类图,对象图,组件图,配置图)和动态行为图(顺序图,协作图,状态图,活动图)。这些图可以从不同抽象角度使系统可视化,分别用于软件开发的各个阶段。

数据包捕获及协议分析系统的需求较为复杂,与操作系统底层结合较紧密。传统的方法一般采用面向数据流的结构化分析设计方法,设计时过多纠缠于数据处理,不能很好地把握系统整体的功能需求,也不能很好地适应系统的结构改动。而UML面向对象的方法从系统抽象的高度,很容易全面把握系统的总体需求,适应需求的变化。当系统需求发生变化时,只需要做很小的改动,系统即可完成相应的更新。

本文借助UML工具,用面向对象的方法完成以太网数据包捕获及分析系统的建模并具体实现。

2 数据包捕获系统设计

数据包捕获及协议分析系统主要完成网络数据包的捕获以及转储、对捕获的数据包进行流量特征分析统计和初步的协议分析,显示在用户界面。

系统体系结构如图1所示。

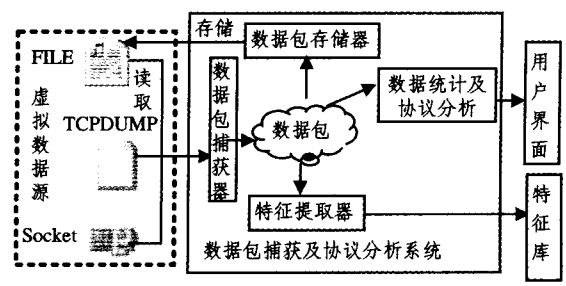

图1 数据包捕获及协议分析系统体系结构图

*)本课题得到国家自然科学基金(60573141和70271050)、江苏省自然科学基金(BK2005146)、江苏省高技术研究计划(BG2004004、BG2005037、BG2005038、BG2006001)、国家高科技863项目(2005AA775050)、南京市高科技项目(2006软资105)、现代通信国家重点实验室基金(9140C1101010603)、江苏省计算机信息处理技术重点实验室基金(kjs050001/kjs06)和江苏省高校自然科学研究计划(05KJB520092)资助。

蔡小华 硕士研究生,主要研究方向为计算机软件技术、信息安全;**王汝传** 教授,博士生导师,主要研究方向是计算机软件、计算机网络和网格、信息安全、无线传感器网络、移动代理和虚拟现实技术等;**任勋益** 博士研究生,主要研究方向为计算机网络和网格计算、移动代理和信息安全技术等。

网络数据包捕获器负责从数据源中捕获数据包，为网络协议分析器提供数据来源。特征提取器从捕获的数据包中提取用于分析网络流量异常的数据包特征，如数据包的到达时间等信息。捕获的数据包需要存储到磁盘文件中，以便管理人员做离线分析或再现当时的网络流量状况。同时接收到的数据包数量及包头信息等，需要直观地显示在用户界面上。

通过以上分析可以知道，待开发系统有如下功能需求：

①以混杂/非混杂模式实现网络数据包的侦听与捕获；

②对已存储的数据文件进行回放，数据仿真网络数据流量；

③进行数据包存储，接收到的数据包可以存储到数据库中或以文件的形式存储于磁盘中；

④提取数据流量特征，存储于专门的数据结构中；

⑤对捕获的数据包进行统计分析；

⑥单个数据包进行简单的协议分析，供用户界面显示。

数据包的捕获方式因不同的操作系统而异。但不管采用哪种捕获方式，都必须有很高的实时性。因为网络中的数据包很多，如果捕获不及时，就会有漏包的情况出现，这样就有可能让一些可疑的数据包有机可乘了，因此还要求系统对于数据包捕获具有较高的实时性和准确性[1]。

3 基于 UML 的捕获及协议系统建模

对数据包捕获及协议分析系统进行 UML 建模，首先要确定系统的边界，即系统将与外界的哪些人或物交互；为了完成这些交互，系统需要提供哪些功能，确定系统的用例，建立系统的用例图；根据系统的用例，细化各个业务流程，理清交互次序，建立用例的顺序图和协作图。完成这些步骤后，在完成的分析中提取已经明确的对象，抽象成类，并完成各类之间的交互，根据功能的需要，再补充一些类，构成系统整体的类图设计；最后回溯回去，修改用例图及顺序图中一些地方，使之与类图完整一致，再借助代码自动生成技术生成系统的源代码。

3.1 系统边界界定

执行者(Actor)是指在系统边界以外，透过系统边界与系统交换信息，即与系统进行交互的任何事物。由问题域研究可知，系统需要与以下实体进行交互：

A. 数据源(Data Source)。数据源包括网卡及存储有历史数据包的数据文件，是系统数据的来源，提供给数据包捕获器分析用的数据。

B. 特征库(ChrtLib)。特征库用于存放从数据包中提取出的数据流量信息。

C. 存储介质文件(Storage)。用于存储截取到的数据包。

D. 用户界面(UI)。用户界面实时显示系统截取数据包的情况及数据包的概要信息，包括捕获的数据包统计及数据包协议内容。

这样，确定了系统的 Actor，也就确定了系统的边界。

3.2 系统用例图模型

用例(Use Case)是对一个 Actor 使用系统的一项功能时所进行交互过程的一个文字描述序列，描述定义了当执行用例时系统将发生什么事。一个用例的实体是一系列的活动，对双方的行为描述力求准确、清晰。

在数据包捕获及协议分析系统中，数据源提供系统数据来源，与系统主要有两种交互：一是让系统从数据源中捕获数据包，包括实时数据(网卡数据)和文件数据。另一方面，要把离线的数据文件恢复成在线数据状态，模拟在线网络数据流量环境，需要数据源提供数据回放功能。因此，数据源有两个用例：数据包捕获用例和数据回放用例。

特征库用于存储从数据包中提取出的数据流量，供检测分析。它主要和系统的数据包特征提取功能相关，需要与系统进行特征提取方面的交互，所以需要一个特征提取用例。

存储介质用来存储捕获的数据包，只需要与系统的数据包存储模块交互，需要一个数据包存储用例即可。

用户界面主要用来显示系统的实时状态，如数据包的捕获情况与各数据包的包头信息等。它需要与系统的两个模块进行交互：数据包统计器模块和数据包协议分析器模块。数据包统计模块负责统计数据包数量及丢包等数据并显示在用户界面上；数据包协议分析器负责对捕获的数据包进行包头协议分析，提取出 IP、数据长度等信息，并显示在用户界面上。所以，用户界面需要两个用例：数据包统计用例和数据包协议分析用例。

综上所述，系统的用例图如图 2 所示。

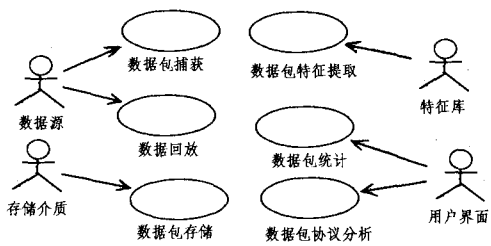

图 2 数据包捕获与协议分析系统用例图

3.3 系统的动态模型

UML 的顺序图表示以时间安排的对象交互。它与协作图是同构的，都是属于交互图。在顺序图中，我们能够清晰、明确地显示控制流。顺序图描述了对象之间动态的交互关系，着重体现了对象间消息传递的时间顺序[2]。

接下来对已确定的 Use Case 做增量开发和迭代规划，建立相应的顺序图和合作图。

3.3.1 数据包捕获用例模型

系统在运行时是从网络上实时捕获数据的，但系统应该也可以分析离线的数据，主要是转储的数据包文件，从这些文件中读取数据，进行分析。

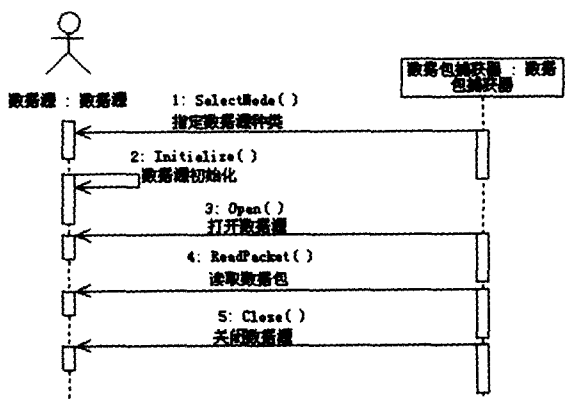

图 3 数据包捕获用例顺序图

网卡是系统的硬件设备，而数据文件是存储在硬盘上的文件，两者有差异。但在 Linux 下，一切硬件设备也都抽象成文件。借助这种思想，在本系统设计中，将网卡与数据文件抽

象成统一的数据源，数据包捕获器只需要指定数据源的种类，不需要考虑数据源具体类型操作，在统一的接口下操作网卡与数据文件。这样，捕获器从数据源捕获数据包时，首先需要给数据源一些初始化参数，如数据源种类、捕获的时间、数量等。数据源按参数进行初始化并准备就绪，捕获器打开数据源开始捕获数据包，完成工作后关闭数据源。

本用例的 UML 顺序图描述如图 3 所示。

图 3 中 SelectMode()用于选定数据源的种类，指定网卡数据还是文件数据；Initialize()执行数据源的打开前初始化工作；Open()用于打开数据源，准备开始捕获数据包；ReadPacket()用于指定数据源开始数据包捕获工作，向捕获器提供数据包；Close()用于捕获完毕后关闭数据源。

3.3.2 数据包存储用例模型

数据包存储用例主要完成数据包捕获后存储到本机磁盘上的工作。为了提高效率，减少读写磁盘的次数，采用缓冲存储技术。在内存中开辟一块缓存池，当捕获数据包时，先存储到缓存中。如果缓存已满，则将缓存一次性写到磁盘中，然后清空缓存，再进行存储。考虑到效率的问题，在此可以开两个线程：一个负责缓冲区管理，一个负责将缓冲区数据存储到磁盘中。

其具体流程描述如图 4 所示。

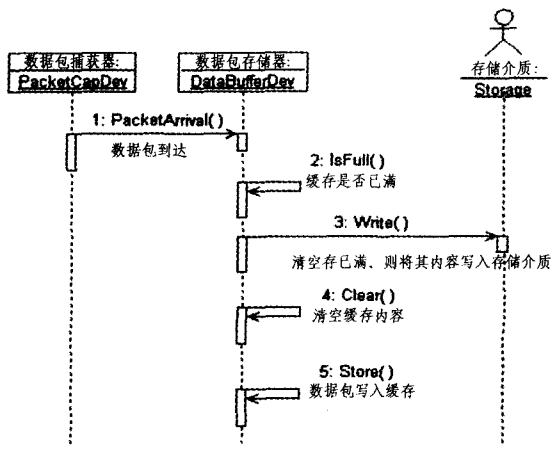

图 4 数据包存储用例顺序图

图 4 中 PacketArrival()是系统消息响应函数，通知各个相关模块有新数据包到达；IsFull()用于判断数据缓冲区是否已满；Write()用于将数据缓冲区数据存储到磁盘中；Clear()用来清空缓冲区数据；Store()用于将数据包写入缓冲区。

根据作出的顺序图还可以生成对应用例的协作图。图 5 是根据数据包存储用例顺序图生成的协作图。

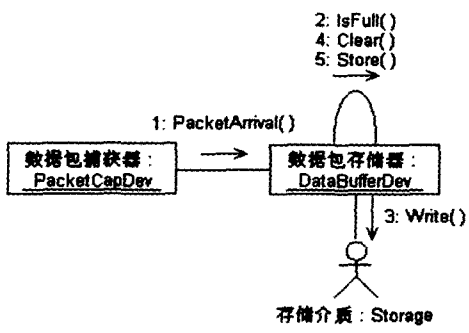

图 5 数据包存储用例协作图

其他的用例都做类似的增量开发，在此不赘述。

3.3.3 类图

综合以上各 Use Case，外部对象主要有数据源、用户界面、特征库、存储介质。系统内部对象主要有数据包捕获器、数据包协议分析器、数据缓冲器、数据包统计器、数据包流量特征分析器、数据回放器等。考虑其中的消息关联，将各个对象抽象为类，并加入各个类之间的可能关系，建立系统的类图，如图 6 所示。

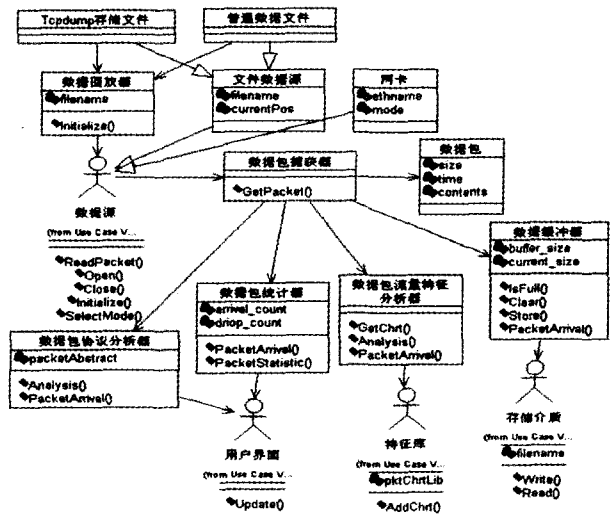

图 6 数据包捕获与协议分析系统类图

4 系统实现及主要优点

4.1 系统实现

通过以上步骤，借助 UML 完成系统的分析与设计，建立系统的模型，并实现将现有的系统模型映射为具体的实现代码。这个过程中主要使用前向工程，借助于目前许多的 UML 建模工具，如 Rational Rose，可以提供多种编程语言以供映射[3]。UML 中的大部分图包括类图、构件图和状态图，都可以在工程中选用。顺序图中每个对象都由一个类来实现。顺序图抽取出了每个类具有的功能，通过将类向实际编程语言的映射，可以得到具体的实现代码。尽管其中大部分代码不能成为最终的源程序，但对编码实现有重要的参考价值。

借助 Rose 的代码自动生成技术，系统数据缓冲器类映射的程序代码片段如下所示：

```
class PacketBuffer
{
private:
    u_char * m_pBase;      // 缓冲区首地址
    int m_bufferSize;      // 缓冲区大小
    int m_bufferCurrentSize;// 缓冲区剩余空间大小
public:
    bool IsFull();
    bool Clear();
    bool Store();
    void PacketArrival(u_char * pPacket);
    ……
};
```

其他类的映射代码类似，在此不一一赘述。

4.2 系统优点

在本系统实现中，考虑到系统对精度与时间方面的要求，应用了一些关键技术，使系统具有如下优点：第一，在设计时运用 GOF 设计模式中的抽象工厂模式(Abstract Factory)[4,5]，对数据源做了抽象，建立虚拟数据源的抽象类，运用

(下转第 75 页)

为了弥补上述三类算法各自的缺点，一些研究试图将它们结合使用：文[6]首先通过测度将流分类，然后在每一类中寻找代表端口以标识协议。虽然该方法可以提高基于端口识别协议的准确性，但是如果端口的选择完全随机，方法将失效。文[8]和[16]结合了端口和负载的算法以提高识别的准确性。特别是文[8]中的算法迭代地使用九个子方法，识别的准确性达到99%，但是其时间和空间复杂度过高，不可能应用于实际。

小结 新的应用层协议识别算法的研究是现在计算机网络的一个热点研究方向，其必须有高准确性和高效率以克服由于新复杂协议及高速网络给协议识别所带来的困难。本文将目前正在使用或研究的协议识别算法归纳为三类并分析了它们的优缺点和难点：传统的基于端口识别协议的算法简单，但是随着网络的不断发展其准确性已经低于50%，不可用；基于负载识别协议的算法具有90%以上的准确性，但开发和维护需要很大的工作量，且该类算法的时空复杂度过高，限制了其在实际环境中的使用；刚出现的基于测度识别协议的算法不需要知道协议的具体细节，易于扩展与维护，时空复杂度都远低于基于负载的算法，但是目前算法还不成熟，识别不细，且准确性只达到70%左右。鉴于三类算法各有优缺点，未来的研究可以向两个方向发展：一、研究基于测度的算法中如何挑选测度及判别方程并证明算法的最优；二、进一步研究如何更好地结合各类算法，使其优势互补。例如，可以结合基于测度和负载的算法：先使用基于测度的算法对流进行粗分，然后在每一类中使用基于负载的算法标识出协议。这样既可以避免在大范围中使用基于负载的算法以提高速度，又可以增加基于测度的算法的精度和准确性。

参 考 文 献

1 Sen S, Wang J. Analyzing Peer-to-Peer Traffic across Large Networks[C]. IEEE/ACM Transactions on Networking. NJ: IEEE Press, 2004. 219～232

2 Plissonneau L, Costeux J L, Brown P. Analysis of Peer-to-Peer Traffic on ADSL[J]. In PAM 2005, volume 3431 of LNCS Springer, 2005. 69～82

3 RFC3971. Requirements for IP Flow Information Export (IPFIX)[S]

4 Hifn, Inc. Why You Need Flow Classification, Technical White Paper[EB/OL]. http://www.hifn.com/docs/a/WP-0001-00-Why-You-Need-Flow-Classification.pdf. September 2001

5 IANA. http://www.iana.org/assignments/port-numbers[S]

6 Kim M S, Won Y J, Hong J W K. Application-Level Traffic Monitoring and an Analysis on IP Networks[J]. ETRI journal, 2005, 27(11): 22～42

7 Roughan M, Sen S, Spatscheck O, Duffield N. Class-of-service mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. In: Proc. ACM. SIGCOMM IMC 2004, Taormina, Italy, Oct. 2004. 135～148

8 Moore A W, Papagiannaki K. Toward the Accurate Identification of Network Applications[C]. In PAM2005. Boston, MA, 2005. 41～54

9 Kang H J, Kim M S, Hong J W-K. A Method on Multimedia Service Traffic Monitoring and Analysis[J]. In DSOM 2003. Lecture Notes in Computer Science 2867. Heidelberg, Germany, Oct. 2003. 93～105

10 van der Merwe J, Caceres R, Chu Y-H, Sreenan C. mmdump-A Tool for Monitoring Internet Multimedia Traffic[C]. ACM Computer Communication Review, Oct. 2000, 30: 48～59

11 Sen S, Spatscheck O, Wang D. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures[C]. In: *Proceedings of the 13th international conference on* World Wide Web. NY: ACM Press, 2004. 512～521

12 Zander S, Nguyen T, Armitage G J. Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics[C]. In: PAM2005. Boston, MA, 2005. 325～328

13 Zuev D, Moore A W. Traffic Classification Using a Statistical Approach[C]. In: PAM2005. Boston, MA, 2005. 321～324

14 McGregor A, Hall M, Lorier P, Brunskill J. Flow Clustering Using Machine Learning. Techniques. In: PAM2004, France, April 2004. 205～214

15 Moore A W, Zuev D, Crogan M. Discriminators for use in flow-based classification[R]. Department of Computer Science, Queen Mary, University of London, August 2005

16 Choi T S, Kim C H, Yoon S H, Park J S, Lee B J, Kim H H, Chung H S, Jeong T S. Content-aware Internet Application Traffic Measurement and Analysis[C]. In: Proc. NOMS, 2004. 511～524

（上接第72页）

抽象接口，将网卡与文件数据封装在一起，对外提供统一接口，实现无差别访问。另外，借鉴Windows消息机制思想，在数据包到达后，由数据包捕获器向各个处理模块发送消息，通知各模块处理新到的数据包。运用设计模式及消息传递机制，系统的结构简单清晰，易于维护。第二，读写磁盘的速度系统，处理的速度存在较大差异，而本系统对时间要求较高。为解决数据包存储的速度瓶颈问题，本系统采用缓存技术。数据包到达时，不是立即存储到磁盘中，而是先存储在内存里事先开辟出来的缓冲区中。当缓冲区数据满时，再一次性写入磁盘中，这能显著地减少磁盘读写次数。另外，结合多线程技术，另开一线程，专门负责数据写操作。缓冲技术结合多线程技术的运用，使得系统实时性得到显著的改善[6]。

结束语 网络数据包捕获及协议分析技术是防范网络攻击的基础，良好的底层捕获实现为网络入侵检测奠定坚实的基础[7]。本文运用UML建模技术，设计数据包捕获及协议分析系统的总体结构和功能模块，将UML应用到数据包捕获及协议分析系统的设计与开发中，使开发者能够准确理解系统各部分之间的内在联系，加快软件开发过程，提高软件开发水平和软件质量，对网络攻击防范软件开发具有借鉴意义。

参 考 文 献

1 周治平，夏娟，纪志成. 基于UML的实时系统设计方法的分析与比较[J]. 计算机工程，2005，31(13)：99～101

2 王强，贾素玲，等. UML系统分析设计[M]. 北京：高等教育出版社，2005

3 马蔷，杨南海，等. UML软件开发[M]. 北京：电子工业出版社，2005

4 姚淑珍. UML和模式应用一面向对象分析与设计导论[M]. 北京：机械工业出版社，2002

5 Alexandrescu A. Modern C++ Design(影印版)[M]. 北京：中国电力出版社，2003

6 高光勇，谢志恒. 网络入侵检测系统中的包截获和报文解析[J]. 齐齐哈尔大学学报，2004，20(4)：47～50

7 刘文涛. Linux网络入侵检测系统[M]. 北京：电子工业出版社，2004