An Object-Oriented Model of Network Performance Monitoring System*

WANG Ji-long, WU Jian-ping

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

E-mail: herb_sky@hotmail.com

http://www.tsinghua.edu.cn

Received May 10, 1999; accepted November 14, 1999

Abstract: The need to monitor the health of the global Internet has now become crucial. In this paper, an object-oriented model for network performance monitoring is discussed, which is adopted in the management of China Education & Research Network, a national information network of China. The current state of the network performance management model is discussed first, which is helpful in formulating the setting of goals and requirements which a large-scale performance monitoring model must meet. Then an architecture solution to systematize the implementation of Internet monitoring and how it attempts to address these goals and requirements are presented. Finally some examples about how to use this architecture model in real-life network monitoring are given.

Key words: Internet; performance; object-oriented; monitor

Historically, the Internet has been woefully under-measured and under-instrumented. The problem is only getting worse with the network's ever-increasing size. Consequently, the need to monitor the health of the global Internet has now become crucial. Several research efforts are tackling this problem by working on developing measurement models for small groups of sites, such as IPMA[1]; Surveyor[2]; Felix[3]; NIMI[4]. In this paper, we address and discuss an object-oriented model for network performance monitoring. We named our model here as CPMM (CERNET performance monitoring model) because it is used in CERNET performance monitoring system, a web-based network management system. CERNET (China education & research network) is a nation-wide information network composed of thousands of routers and hosts. The performance monitoring of such a huge and heterogeneous system is a very important and hard work. To automate the task of CERNET's performance monitoring, we designed and implemented an application based on CPMM model.

The key difference between CPMM and models mentioned above is that the foremost goal of CPMM is to devise an infrastructure capable of monitoring a very large network by high-level automation.

CPMM is required to cover the following needs:

(1) Problem-oriented monitoring. That is, CPMM must be able to locate a specific performance problem quickly and accurately. Suppose that a user called up with a complaint. A network administrator must get some idea of possible sources of the problem quickly to give proper reply. To achieve this, the administrator should be

* The project is supported by the National 'Ninth-Five' Sci-Tech Key Project of China under Grant No. 96-743 ("九五"国家重点科技攻关项目). **WANG Ji-long** was born in 1973. He got Ph.D. degree of computer science in Tsinghua University in 2000. Now he is a director of research department of CERNET Naitonal Center. His current research interests are Internet behavior theory, Internet measurement and analysis. **WU Jian-ping** was born in 1953. He is a professor of Tsinghua University. His current research interests are advanced computer networking, network management and network security.

able to order the monitoring of specific network resources at a high level of detail in order to ferret out the problem.

(2) General monitoring. To seek and forecast performance problems of the network in an active way, so that problems can be found out and solved before users complain.

(3) Historical analysis of trends and resource usage. It is desirable to be able to perform statistical analysis of various performance data over some period of time.

(4) Independent of the realization of management information collection. According to the specific condition, we have to employ a variety of methods to collect state information. For example, to a Cisco router with IOS 11.3, we can gather detailed traffic information by netflow mechanism, but sometimes to reduce the device's load, such as a backbone router, we may have to use Tcpdump to capture data.

1 Current State of the Network Performance Monitoring Model

Model for network management enables the designers to discuss management functions at a high level of abstraction and guide the design of management protocols and services. During the last decade people recognized the need for network performance management and developed models[5~14] to guide the design of network management services, protocols, and systems. Following are some of them.

ISO

The ISO management model is defined in the 'Management Framework'[9] and the 'Systems Management Overview' standards. As compared to other network management models, the ISO model received most attention within the research community.

ITU

The management model[13,14] of the ITU-T is known as the 'Telecommunications Management Network' (TMN). The name of this architecture already indicates that this architecture is primarily intended for management of telecommunications (e.g. telephony) networks. TMN in fact consists of multiple smaller architectures:

· a functional architecture

· a physical architecture

· an information architecture, which includes many ideas of ISO management

· a logical layered architecture, which includes a responsibility model.

IETF

In 1988 the 'Simple Network Management Protocol' (SNMP)[6,9] was defined by the IETF to meet the immediate management needs of the Internet. As opposed to the ISO and ITU-T the IETF did not define a separate model standard to describe the concepts behind SNMP. The reason for this is that these concepts resembled those already described in drafts of the OSI Management Framework and were considered to be obvious. In 1992 the IETF started the development of a second version of SNMP (SNMPv2)[6,8]. Although the concepts behind SNMPv2 are more difficult to understand and so should be defined in a separate standard, such a definition has not been produced.

Although these models proved their applicability in many cases, they still suffer from a number of problems. All current management models, notably the ISO, ITU-T (the former CCITT) and the IETF models, have been developed after the design of the normal network functions is completed. Also such approach indicates a specific conceptual view on the role of management functions they did not apply different architectural concepts for the design of management functions. Thus implementers may not always obtain a sufficient understanding of these standards.

There are still several research efforts from non-standardized organizations that are tackling this problem by working on developing measurement models. However these models either were designed mainly for small groups of sites, such as IPMA[1]; Surveyor[2]; Felix[3], or considered little about automatic addressing of performance problem (NIMI[4]) thus are unsuitable for the overall Internet monitoring.

2 Overview of the CPMM Model

CPMM is an object-oriented model. It defines the "managed objects" representing the resources to be monitored, defines "history data" objects encapsulating the processing of history data related to "managed objects", defines "data formatting" objects performing data formatting tasks, defines "threshold" objects supporting passive monitoring and the information exchanging mechanism related to objects. Commonly, network management system works in C/S mode with a manager running on the management server and agents running on devices and monitoring stations. The manager sends messages to manipulate agents. Agents notify the manager of various events that occur within agents or that agents detect. In CPMM, between manager and agents, we defined "managed objects" which represent agents and deputize the information exchange between manager and agents. This scenario is depicted in Fig. 1.

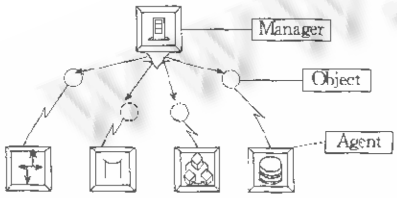

Fig. 1 Architecture of CPMM

This section gives an overview of the objects, the relationships between the objects, and the behavior of the objects in the model. There are many details and options that are not described in this article due to space limitations.

Managed Object

At the most elementary level performance monitoring ultimately depends on measuring information about resources that are either with in agents or that agents monitor. We call each agent a managed object and define a class called managed object class. A managed object class is a collection of performance measurements and relative methods such as data collection method and data retrieval method that are used to monitor agents. The managed object class defines methods to get data from agents and methods to put formatted data to server. By these methods, the server can manipulate agents just as they are the same. There are a variety of performance measurements to describe network devices' performance. However any given piece of equipment may only be able to provide certain part of these measurements. In fact, in general only some of these measurements will make sense when monitoring certain types of resources. For example, it does not make sense to try to collect laser bias current measurements on a non-optical link. For each type of resource being monitored, there are various measurements that are applicable. Each type of equipment may well have a distinct set of measurements that apply only to that type of resource. Furthermore, for different agents we may employ different data collection methods, such as SNMP, Tcpdump, Tcl and so forth.

The need to collect different information for different resources and to collect data using different methods for different agents asks us to define managed object class as a virtual class and derive subclasses from managed object class for different agents. Each subclass corresponds to the type of resource being monitored. The subclasses define the particular set of measurements that are applicable to a particular type of resource. In fact, the managed object class will usually not be instantiated; only its subclasses will be instantiated. In this article, we will use the term "managed object" generically to refer to the managed object class or any of its subclasses, since all the subclasses behave essentially the same.

For each monitored resource if there is a managed object instance, it specifies the various measurements that are to be collected about the monitored resource. The various measurements are represented as attributes in the managed object instance and can be accessed via data retrieval methods of managed object.

Generally, there are only two types of performance measurements: gauge and counter. Either gauge or counter is only an integer.

Although the data structure of performance measurements is very simple, the measurements are often time-related. The counters give the number of events that occur within a time interval, and they are reset at the end of each interval. It is also applicable for gauges. So a measurement in a managed object represents a group of data collected over a long time instead of a single integer. We use another object to deal with the data management of measurement.

History Data Object

Generally, we need not only the current state data to evaluate agents' performance, but also the history data over a certain interval. To support data logging function, we define history data class. The history data class includes the database of the performance measurement to be logged and the methods to access the database.

Problems concerned with data logging are disk usage and access speed. A large scale network like CERNET may contain thousands of agents. An agent may have dozens of measurements valuable to resource monitoring. And commonly we need to keep the logging data for a long time, maybe several months or even several years. In fact, if we could we would like to keep some measurements forever, for example, traffic data. However, keeping history data for a large number of measurements over a long time will challenge greatly our storage capacity and database accessing technology.

To avoid the bad performance of the network management system itself, we define a policy for data management. The database of a history data object contains several tables. Different tables represent different sampling intervals. The row number of each table is fixed, therefore the maximum size of the database is a constant. When the size of a tables is fixed, the longer its sampling interval, the longer its coverage. Thus different tables present different coverages. For example, a database has 4 tables. The table with a sampling interval of 5 minutes contains the data of the current day, while the table with a sampling interval of half an hour contains the data of the current week. By this way, we can compress the history data greatly. For example, we define the maximum row number of a table as 1000, then a table with sampling interval of 24 hours can keep the measurement over 2 years. This policy is reasonable because when we are consider a performance measurement such as traffic over a period, we are mainly concerned with its varying trend rather than accurate values. So to track a year's traffic, 24 hours sampling is enough. One thing must be addressed here: a sample of some interval is not the instantaneous values at some point but the average of the values at the ends of the intervals.

Data Formatting Object

We have introduced the mechanism of collection and storage of performance data in CPMM. However, a complete performance monitoring model must also deal with how the data are formatted for different usages. To be clear and function independent, we define a set of objects to provide the capability of information format. These objects are called "formatter". The function of a formatter is to scan a set of history data objects and formats information under some rule and then forward the result to any application calling them. The parameter of the formatter is the set of objects to be scanned.

We define the following formatters in CPMM:

(1) BasicFormatter: Defining basic fixed scanning behavior. It collects information from all objects scanned.

(2) StatisticFormatter: A subclass of BasicFormatter. It provides a calculation to be performed across the scanned managed objects.

(3) MeanFormatter: A subclass of StatisticFormatter that computes the mean values of one or more attributes across the set of scanned managed object instances.

(4) VarianceFormatter: A subclass of StatisticFormatter that computes the variance of one or more attributes

across the set of scanned managed object instances.

(5) Min-maxFormatter: A subclass of StatisticFormatter that computes the minimum and maximum values of one or more attributes across the set of scanned managed object instances.

(6) ReportFormatter: A subclass of BasicFormatter. It provides a mechanism for efficient packaging of the scanning report and also permits reports to be initiated on demand.

New formatter can be defined conveniently by deriving a subclass of any formatter described above.

Threshold Control Object

Besides general performance analyzing functions, in many cases it is desirable to notify the administrator immediately when some event occurs. If there is a catastrophic network failure at 3:00 p.m., the administrator probably does not want to wait until 5:00 p.m. to find out about it. Spontaneous reporting is accomplished using a class called "threshold". Threshold objects contain the thresholds and hysteresis levels associated with the measurements of managed objects. If a counter or gauge in the managed object passes a threshold, a notification is issued. This notification may be sent to an active window of network monitor or stored in a log database as a performance spontaneous event record, or both.

The managed object instances and threshold object instances are related by a pointer from the managed object instances to the threshold data instances. This relationship is used instead of containment to allow for different ways of using thresholds. It is sometimes desirable to have one set of thresholds apply to many resources of the same type. In other cases, it may be desirable to set up specific thresholds for an individual resource. This may occur, for example, if the administrator is attempting to trace a problem on an individual circuit. The pointer relationship allows this flexibility.

Other Objects

To build a complete performance monitor model we still need some other objects. In many cases, an event won't be processed at once. Only a few of the events will be given real-time response according to some event filter rules. Other events will be logged and may be viewed later when some problems are analyzed. A log object is up to event logging function and providing log access.

The result of performance monitoring is ultimately provided in the form of performance monitoring report. There are commonly two types of performance monitoring report: graph and table. A variety of reporter objects are needed to generate performance monitoring report. A virtual class "Reporter" is defined to describe the common interface of reporter objects. Each type reporter objects, such as a bar, a pie, a line or a plain table, is derived from a certain subclass of "Reporter".

The Performance Monitor Session

The performance monitoring is done by exchanging information between manager and managed objects. We call the process of information exchanging a session. Because to different objects and in different cases the information exchanged between manager and managed objects may be of great difference, we define 16 types of sessions.

(1) New: Allows the manager to activate a managed object instance.

(2) Delete: Allows the manager to deactivate a managed object instance.

(3) Retrieve: Allows the manager to read the performance data in a managed object instance.

(4) Initialize: Initializes the measurements in a managed object.

(5) Threshold Edit: Allows the manager to set or retrieve threshold settings.

(6) Threshold Control: Allows the manager to define or reconfigure which thresholds apply to which monitored resources.

(7) Data-collection on: Allows the manager to enable the periodic data collection of performance measurements.

(8) Data-collection off: Allows the manager to suppress the periodic data collection of performance measurements.

(9) Data collection control: Allows the manager to configure the contents and schedule of data collection.

(10) Data formatting: Allows the manager to request a data formatting on demand.

(11) Statistical data formatting: Allows the manager to request a statistical formatting.

(12) Event notify: The object sends spontaneous notifications to the manager.

(13) New log: Allows the manager to create new logs.

(14) Delete log: Allows the manager to delete old logs.

(15) Logging control: Allows the manager to activate or deactivate a log or change the parameters controlling the log.

(16) Log retrieval: Allows the manager to read the contents of the log directly.

3 Examples of Using the Model in CERNET Performance Monitoring

Monitoring

In order to give some ideas about the power and flexibility of the model, several examples are presented that show how the model may be tailored to fit specific needs.

Data Collection

The basic function of performance monitoring is to collect the state information of the monitored network. We use managed objects to represent network management agents. A managed object is an instance of a subclass of "managed object class". To get periodic state data, the manager enables the periodic data collection of performance measurements of managed objects, which support the scheduled data collection function, by session "data collection on". The managed objects would begin to collect the data using proper method defined by the corresponding managed object subclass. For each measurement to be monitored, a history data object is created.

Data Formatting and Report Generation

To be convenient for performance analyzing, the "raw data" must be formatted to suit all kinds of analyzing requirements. The formatter object instance then comes into play. It scans the logging data looking for all the records that meet certain criteria, such as having a specific time stamp. From each of the records that meet the selection criteria, it retrieves a given set of attributes. The set of attributes to be retrieved is part of the formatter configuration information. This information is then formatted into a report. The formatter report is issued as a notification and will be forwarded to the waiting manager. This scenario is depicted in Fig. 2.

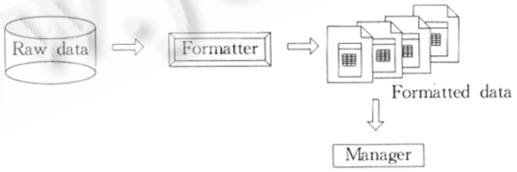

Fig. 2 Data formatting

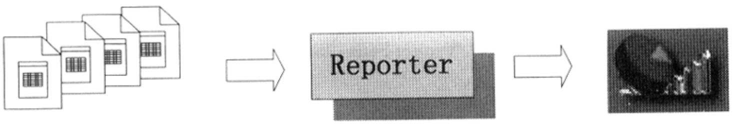

Fig. 3 Report generation

The formatted data are served as the input of performance reporter. A performance reporter is an instance of report subclass. The reporter eats a set of formatted data and generates curves, bars, pies or tables.

Events Monitoring

The manager enables agents' threshold control function by "threshold control" session. Agents notify the manager whenever a significant degradation of performance is detected by session "event notify". The threshold data object is required when agents' threshold control is enabled. This scenario is depicted in Fig. 4.

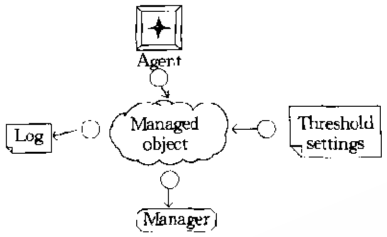

Fig. 4 Reporting of performance events

In this scenario, the managed object instances are created by the manager. For each of the resources being monitored, the manager would instantiate an appropriate managed object. The instances of managed object represent the resource. The managed object selects the options such as the performance summary interval and set of counters and gauges that it is willing to monitor.

The managed object instances would point to one or more threshold data objects that define the conditions causing the performance events to be issued. The manager can alter the threshold settings by "threshold control" session. If we create and properly configure the log objects, each of these notifications will be converted into performance events and stored into the log. A "logging control" session is needed in this case if the manager needs to be able to change any log configuration information.

4 Strengths and Weaknesses of the Model

This model covers almost all of the key aspects in building a network performance management system. It is powerful. However it is not perfect.

The most outstanding feature of our model is its scalability which is fatal to the generic performance management model. A generic model should work across different technologies, including yet-to be-developed technologies. It is obviously not possible to specify the attributes to be kept for an undiscovered technology. It is not even possible to specify the attributes for all the existing technologies. The best that can be done is to make the model extensible enough to cover new types of measurements. We use object-oriented method to do this. When new technologies are standardized, we can define new subclasses of managed object class that contain the attributes appropriate for monitoring the resources associated with the new technology. The primary drawback of this approach is that it may lead to a combinatorial explosion of managed object subclasses. This can lead to difficulties in interoperability since one system may not be familiar with the other system's current data subclasses. Furthermore, the approach of creating new subclasses of managed object for new technologies can be expected to be administratively cumbersome.

References:

[1] Almes, G. IP Performance Metrics: Metrics, Tools, and Infrastructure. http://io.advanced.org/surveyor/, 1997.

[2] Craig, L., Farnam, J., Scott, J., *et al.* The Internet Performance and Analysis Project. http://www.merit.edu/ipma/docs/team.html, 1997.

[3] Christian, H., Mark, W.G., Joe, D., *et al.* Project Felix: Independent Monitoring for Network Survivability. ftp://ftp.bellcore.com/pub/mwg/felix/index.html, 1997.

[4] Vern P., Jamshid, M., Andrew, A., *et al.* An architecture for large-scale Internet measurement. IEEE Communications, 1998,36(8):48~54.

[5] Claffy, K., Tracie, M. What's next for Internet data analysis? stutus and challenges facing the community. Proceedings of

the IEEE, 1997,85(10):1563~1571.
[6] RFC 1157. A Simple Network Managment Protocol.
[7] RFC 1441. Introduction to SNMP v2.
[8] RFC 1445. Administrative Model for SNMP v2.
[9] RFC 2261. An Architecture for Describing SNMP Management Frameworks.
[10] Nilotpal, M., Subrata, M. ROS-to-CORBA Mappings: First Step towards Intelligent Networking using CORBA. In: Proceedings of the 4th International Conference on Intelligence in Services and Networks. Como, Italy, May 27~29, 1997.
[11] Malan G. R., Jahanian, F. An extensible probe architecture for network protocol performance measurement. Computer Communication, 1998,28(4):215~217.
[12] ISO/TC 97/SC 21 N975. OSI Management Framework-Seventh Working Draft. November 1985.
[13] CCITT: Recommendation M.3010, Principles for a Telecommunications Management Network. Geneva 1992.
[14] CCITT X.738, ISO/IEC 10164—13. Open Systems Interconnection— -Systems Management —Part 13: Summarization Function. DIS Text, 1992.

一个面向对象的 Internet 性能监控模型

王继龙, 吴建平

(清华大学 计算机科学与技术系, 北京 100084)

摘要: 性能监控模型和技术是当前 Internet 运行和管理所面临的关键问题之一. 提出了一个面向对象的 Internet 性能监控模型. 该模型已成功运用于 CERNET 全国主干网络的性能管理. 首先综述了当前的各类网络性能管理技术及主要问题, 进而抽象出大型计算机互联网络性能管理的主要需求和目标. 针对这些问题和需求, 提出了系统的网络性能监控和体系结构模型. 最后, 通过应用实例分析了模型的主要特点.

关键词: 互联网; 性能; 面向对象; 监控

中图法分类号: TP393 **文献标识码**: A