THE GOBBLER

An Ethernet troubleshooter/protocol analyzer

TIRZA VAN RIJN and JAN VAN OORSCHOT

Delft University of Technology
Faculty of Electrical Engineering
Mekelweg 4, P.O. Box 5031, 2600 GA Delft, The Netherlands
E-mail: JPMvOorschot@et.tudelft.nl

ABSTRACT

This paper presents "The Gobbler", an Ethernet troubleshooter/protocol analyzer that runs on common PC, AT and PS/2 computers and can be operated from a remote central network management station. It features a packet capture program with extensive filtering capabilities for catching selected Ethernet packets and writing them to disk for later examination, and a dumpfile view and protocol analyzing program for examining the captured packets. "The Gobbler" is based on a event-driven multitasking operating system called the Network Packet Dispatcher, developed by the network performance group of the Delft University of Technology. Future enhancements for "The Gobbler" will include among others a packet generator and a cable fracture tester.

INTRODUCTION

Computer networks have become more and more popular over the years. Especially Local Area Networks (LANs) have been growing in number, in number of users and in topology complexity. Network managers are in search of flexible troubleshooting tools to trace the roots of network problems, while network protocol developers are in need of protocol analyzers to test their software.

In this paper we discuss "The Gobbler", an Ethernet troubleshooter and protocol analyzer that runs on common PC, AT and PS/2 computers and can be interrogated from a remote central network management station. In the following section "Context" the context of this project will be explained. The section "Results and applications" describes the capabilities of "The Gobbler" , while the section "Case studies" gives some examples of using "The Gobbler". The planned enhancements for "The Gobbler" are listed in the section "Future enhancements". Finally a summary of "The Gobbler" project is given in the section "Conclusions".

CONTEXT

Today it is possible to use common PC, AT or PS/2 computers with a network device that supports promiscuous mode as cheap network management stations. For these computers the network performance research group at the Delft University of Technology has developed the Network Packet Dispatcher, a special operating system with a Simple Network Management Protocol (SNMP) interface to build network management programs on. The Network Packet Dispatcher was the basis for amongst others the network monitor "The Beholder", the network packet capture program "NetCapt" and the network packet viewer and protocol analyzer "NetView", also developed at the Delft University of Technology.

The goal was to develop a flexible Ethernet troubleshooting and protocol analysis tool that would run on common PC, AT and PS/2 computers and could be operated from a remote central network management station using the SNMP protocol. It should be possible to capture selected Ethernet packets for storage on background memory, and to examine and analyse the captured packets afterwards. As basis the Network Packet Dispatcher operating system was used. In this section some of the elements of this objective are described: the Ethernet, the Network Packet Dispatcher and the SNMP protocol.

ETHERNET

The network troubleshooter/protocol analyzer was developed for an Ethernet network, one of the most popular LAN technologies [2]. This is a 10 Mbps broadcast bus topology in the form of a coaxial cable connecting the stations (Figure 1).

It uses the CSMA/CD (Carrier Sense Multiple Access with Collision Detect) access protocol, which means that when no carrier wave is present all stations are allowed to send, and that all stations stop transmitting on collision detection and wait for a random interval before retransmitting. The format of an Ethernet packet is depicted in Figure 2.

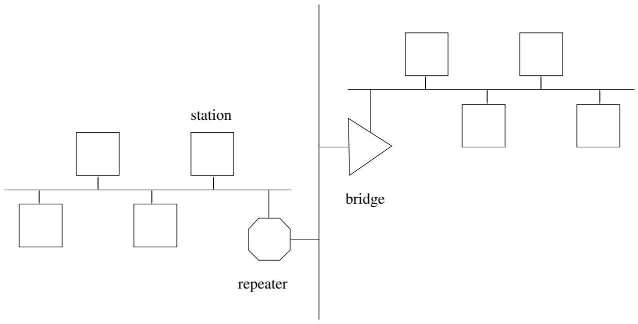

Figure 1. Schematic diagram of an Ethernet

Normally a station only receives packets that are addressed to him, but a special receive mode of the Ethernet network device, called 'promiscuous mode', allows the station to receive all the packets that appear on the net.

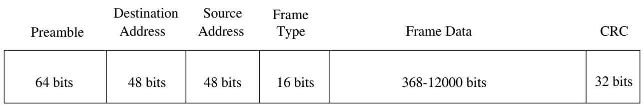

Figure 2. The Ethernet frame format

THE NETWORK PACKET DISPATCHER

The 'Network Packet Dispatcher' (NPD) is a small event-driven multitasking operating system on top of MS-DOS, developed by the network performance group of the Delft University of Technology especially as a programming basis for network management programs [4]. The NPD is equipped with a SNMP interface using an UDP/IP socket interface [6].

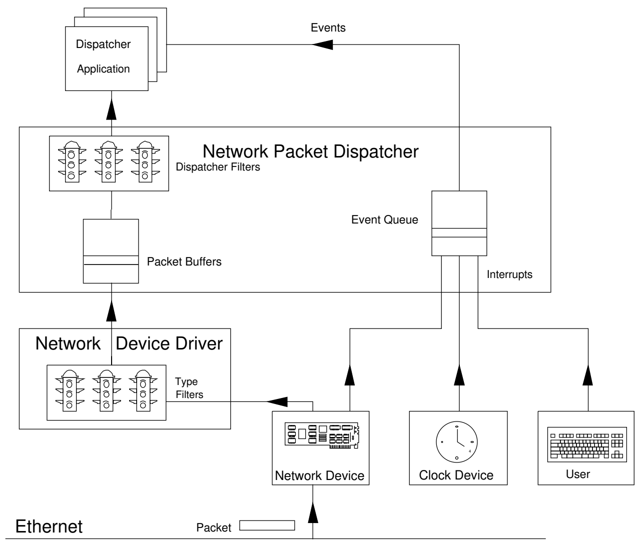

Figure 3. The Network Packet Dispatcher

The NPD is an event-driven operating system. It converts interrupts, generated by the network devices, the clock device, the NPD itself or the user typing on the keyboard, to 'Dispatcher Events' which are distributed to all the 'Dispatcher Application Programs' (see Table 1). It also receives and buffers incoming packets from the network devices to distribute them to the various Dispatcher Application Programs for further processing.

Event	Cause	Action
DPE_INIT	NPD	Initialize applications
DPE_END	NPD	Terminate applications
DPE_START	user	Start application
DPE_STOP	user	Stop application
DPE_RESET	user	Reset application
DPE_SHOW	user	Activate on-screen display
DPE_HIDE	user	De-activate on-screen display
DPE_EVERYSECOND	clock device	Time accounting
DPE_RECEIVEPKT	network device	Analyse packet
DPE_FREETIME	NPD	Perform non-essential actions
DPE_KEYPRESSED	user	Handle keyboard

Table 1

The 'Dispatcher Application Programs' are the actual network management tools, programs written in the C programming language that only need to process the NPD's Dispatcher Events. The NPD can run several Dispatcher Application Programs simultaneously.

Using the Ethernet network device in promiscuous mode allows the Dispatcher Application Programs to receive *all* of the passing packets. For some applications, however, it is desirable to receive only a selection of the packets. To filter out unwanted packets the NPD offers two filtering schemes: 'Type Filtering', performed by the network device driver, and 'Dispatcher Filtering', performed by the NPD itself.

'Type Filtering' is a built-in facility offered by the network device drivers. With Type Filters the NPD can instruct the network device driver to receive only packets of certain types, thus saving valuable interrupt time. Unfortunately, Type Filtering can only be done on the 2-bytes protocol type field of the Ethernet packet.

More extensive filtering possibilities are offered by the 'Dispatcher Filters'. Each Dispatcher Application Program can select several Dispatcher Filters to get the desired selection of packets. The Dispatcher Filters, like Dispatcher Application Programs programs written in C, analyse the incoming Ethernet packets and either pass them on to the attached applications or discard them.

THE SNMP PROTOCOL

To allow operation from a central network management station, the NDP is provided with a SNMP interface. The Simple Network Management Protocol (SNMP) [1] is the most widely used protocol for TCP/IP-based networks to communicate network management information. A SNMP server (or agent) maintains a set of network variables. SNMP messages from a SNMP client (or proxy agent) either instruct the server to fetch values from variables (with a *get-request*) or to store values in variables (with a *set-request*), and the server translates these requests to equivalent operations on local data structures.

All the network variables and the operations allowed on them are defined in the agent's Management Information Base (MIB) [5]. SNMP uses the Abstract Syntax Notation One (ASN.1) [3] to define names and types for variables in the MIB. ASN.1 defines a hierarchical namespace and a lexicographical ordering among the names.

RESULTS AND APPLICATIONS

The result of this project is "The Gobbler" troubleshooter/ protocol analyzer, which can be seen as the successor of "NetCapt" and "NetView".

It takes time to write the packets to disk, and in the meantime the packet catcher may have missed some packets. The usefulness of the "Gobbler" packet capture application depends on the network load and the filters present. Without filters, the number of missed packets lies between 10% and 50% for a network load between 5% and 15%. When for instance only packets from a single host are captured, the number of missed packets drops to about 0,1% when the network load is 10%. Whether or not a certain loss of packets is acceptable depends on the purpose of the capturing.

Although "The Gobbler" was designed to be user-friendly and easy to use even for a layman, the interpretation of the packets' contents remains the task for an experienced network manager or protocol developer.

"The Gobbler" consists in fact of two separate programs: a local "Gobbler" to be operated from the local network management station, and a remote "Gobbler" to be operated from a remote central network management station. Both "Gobblers" run on PC, AT and PS/2 computers with a network device that supports promiscuous mode. Their capabilities are described in the next sections.

THE LOCAL "GOBBLER"

The local "Gobbler" is meant for use on a local network management station. It is therefore provided with a menu-driven user interface, but lacks a SNMP interface. It features two Dispatcher Application Programs: a packet capture program with extensive filtering capabilities for catching selected Ethernet packets and writing them to disk for later examination, and a dumpfile view and protocol analyzing program for examining the captured packets.

The packet catcher

The packet capture program writes the packets that pass the filters to disk. The user can set the name of the output dumpfile and its maximum size, the maximum runtime of the program and the maximum number of packets that may be captured. A status window keeps the user informed about the selected dumpfile name, the current and maximum number of captured packets, the current and maximum dumpfile size, the current and maximum runtime, the number of selected filters and the total received and missed packets. It is also possible to open a window displaying the source and destination address and protocol type of the captured packets. The program stops automatically on exceeding one of the limits, but can also be stopped by the user.

The filtering capabilities

For the packet capture program a universal Dispatcher Filter was developed that maintains three collections of filters:

1. *Start filters*, triggering the start of letting the packets through to the application. As long as no packet has passed the start filter collection, no packet is let through to the application.
2. *Packet filters*, selecting which packets are let through to the application. After a packet has passed the start filter collection and as long as no packet has passed the stop filter collection, the packet filter collection analyses the incoming Ethernet packets and either rejects or accepts them. If a packet is accepted, it is passed on to the application.
3. *Stop filters*, triggering the end of letting the packets through to the application. After a packet has passed the stop filter collection, no packet is let through to the application anymore until another packet passes the start filter collection.

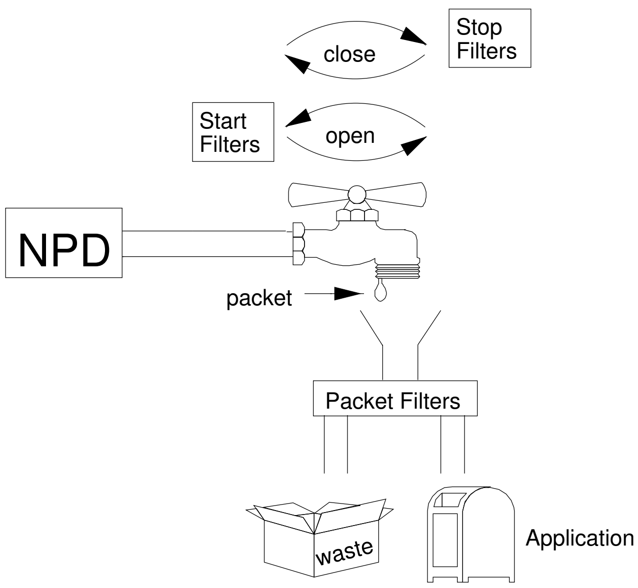

Figure 4. Visualization of the filters

The filters can check an Ethernet packet for six criteria: destination address, source address, protocol type, packet contents, device number and packet size.

Of all the above mentioned filter types there exists both a 'positive' variant, that *accepts* packets conforming to its criteria, and a 'negative' variant, that *rejects* packets conforming to its criteria.

A packet passes a filter collection if it conforms to at least *one* of the positive filters and to *none* of the negative filters in that collection. If a filter collection contains *n* positive filters F_i consisting of criteria $f_{i1}...f_{ik}$ and *m* negative filters G_i consisting of criteria $g_{i1}...g_{ik}$ and the packets are tested on equalness to all the criteria, then the packet's passing P of the filter collection is given by the formula:

$$P(packet) = (F_1 + F_2 + ... + F_n) \bullet (\neg G_1 \bullet \neg G_2 \bullet ... \bullet \neg G_m)$$
$$= A(F_1, F_2, ..., F_n) \bullet \neg A(G_1, G_2, ..., G_m)$$

where $A(X_1, ..., X_n) = (X_1 + ... + X_n)$ and $X_i = (x_{i1} \bullet x_{i2} \bullet ... \bullet x_{ik})$.

Filters can be added, edited and deleted, and it is possible to assign hostnames and protocol names to Ethernet addresses and protocol numbers to make the setting of the source and destination address and protocol type easier. Filter configurations can be read from and written to a file. A status window displays the contents of the filters.

The dumpfile viewer/protocol analyzer

The dumpfile viewing and protocol analyzing program can examine and analyse the packets in the dumpfile created by the packet capture program. It can handle the protocols IP, ARP, RARP, Novell IPX and several DEC protocols like Phase IV, LAT and LAVC. The data contents of the packets are displayed in both hexadecimal and ASCII format. Of other protocols the whole Ethernet data field is taken as the packet's data contents.

THE REMOTE "GOBBLER"

The remote "Gobbler" is meant to be operated from a remote central network management station using SNMP. Its variables can therefore not be set from the local network management station, nor does it display its results on the local screen. It features five Dispatcher Application Programs, a packet catcher with filtering capabilities, and four others (among which a SNMP agent and a tftp server) to make the control by SNMP and the transfer of the dumpfile from the local station to the remote station possible. The dumpfile viewer in this case is a separate program to be run on the remote station itself, not on the local station.

The packet catcher

The packet capture program is the same as in the local "Gobbler", but this version can be started and stopped from a remote central network management station. Of course, here the name of the output dumpfile and its maximum size, the maximum runtime of the program and the maximum number of packets that may be captured are set using the SNMP set-request. The current number of captured packets, the current dumpfile size, the current runtime, and the total received and missed packets can be queried using the SNMP get-request.

The filtering capabilities

The Dispatcher Filter for the remote "Gobbler" has the same capabilities as the one for the local "Gobbler", except for the possibility of reading filter configurations from a file or writing them to a file on the local station. This is not a problem, because it is possible to use "Sage" programs ("The Sage" is a SCHEME interpreter with a SNMP library, also developed at the Delft University of Technology) for setting standard filter configurations from the remote station. All the filters' criteria can be set using SNMP.

The dumpfile viewer

Since it is not possible to make a dumpfile viewer or protocol analyser using SNMP, the dumpfile viewer is not part of the remote "Gobbler", but supplied as a separate UNIX program to be run on the remote central network management station itself. First the dumpfile created by the packet capture program has to be transferred from the local network management station to the remote central network management station using the tftp server, and then the "rdunix" program can be used to view the dumpfile. This program is only a dumpfile viewer, it lacks the protocol analyzing features of the local "Gobbler" version.

CASE STUDIES

Case 1

At a certain moment the 'hall of fame' of the network monitor "The Beholder" showed that a significant portion of the network load was caused by packets that had destination address ff:ff:ff:ff:19:89. Using "The Gobbler" packet catcher with the destination address of a positive packet filter set to ff:ff:ff:ff:19:89 these packets were captured. Analysis showed that all these packets had length 166 and used the illegal protocol number 1989. The contents of the packets even revealed the hostnames of the machines where these packets originated (this can also be gathered from the source address if you have a way of mapping Ethernet addresses to hostnames).

Case 2

A bridge was having problems in getting through its startup sequence using the BOOTP protocol. "The Gobbler" packet catcher was used to capture the packets from and to the bridge. The dumpfile viewer and protocol analyzer made it possible to follow the whole startup sequence and to track down the cause of the problem.

FUTURE ENHANCEMENTS

The network performance research group is planning to extend the troubleshooting capabilities of "The Gobbler" with a ping and an echo application, a universal packet generator and a cable fracture tester. It is also the intention to develop a Protocol Definition Language that would allow users of "The Gobbler" to define their own protocols. The protocol analyzer can then use these definitions to analyse packets that use these protocols. Furthermore, the dumpfile viewer for the remote central network management station will be extended with protocol analyzing capabilities and a UNIX window interface.

CONCLUSIONS

To manage complex network structures troubleshooting tools are needed to determine the origin of network problems. Debugging of network protocol software requires protocol analyzers. "The Gobbler" was developed to provide these features.

"The Gobbler" is an Ethernet troubleshooter and protocol analyzer that runs on common PC, AT and PS/2 computers and is equipped with SNMP provisions to allow operation from a remote central network management station. It features a packet capture program with extensive filtering capabilities to catch selected Ethernet packets and write them to disk for later examination, and a dumpfile view and protocol analyzing program to examine the captured packets.

ACKNOWLEDGEMENTS

This paper was written in partial fulfilment of the requirements for the Master's degree in Electrical Engineering at the Delft University of Technology. At the Department of Computer Architecture and Digital Techniques, the Data Network Performance Analysis Project (DNPAP) research project was initiated by J.P.M. Van Oorschot and is headed by Prof. G.L. Reijns. The main interests of this research group are on network performance analysis, the modelling of interconnected networks and centralized network management.

REFERENCES

[1] Case, J.D. et al., *A Simple Network Management Protocol (SNMP)*, RFC 1157, May 1990.

[2] Comer, D.E., *Internetworking with TCP/IP - Volume I: Principles, Protocols, and Architecture*, Second Edition, Prentice-Hall International Inc., Englewood Cliffs, 1991.

[3] ISO 8824, *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*, 1987.

[4] Oorschot, J.P.M van, and Thio, L.H., *The Network Packet Dispatcher. An Event-driven Multi-tasking Operating System for Network Monitors*, 2nd International Conference on Local Communication Systems: LAN and PBX, page 343-355, Palma de Mallorca, 1991.

[5] Rose, M., *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, RFC 1158, May 1990.

[6] Wisse, D.M., *Simple Task Report - A Simple Network Management Protocol*, Delft University of Technology internal report, 1990.

Table of Contents