Improved Network Management Using NMW (Network Management Worm) System

Hiroyuki OHNO <hohno@is.titech.ac.jp>
Akihiro SHIMIZU <akihiro@is.titech.ac.jp>

Abstract

Network management worm (NMW) system is introduced in this paper. NMW system is a back-end part of our nation wide network management system and assists WIDE/PhoneShell environment. NMW is supported by three sub systems. They are WSD, OWL, and WORMC. These sub systems send, receive and execute NMW. Various information and command can be packed into NMW and processed on various hosts. We are testing NMW system in some networks to manage and monitor their networks and NMW system has been working fine. We think NMW system will be widely spread in the WIDE internet soon.

1 Introduction

We and the WIDE project have been developing various network management systems for the nation wide computer networks since 1988. WIDE/PhoneShell [1, 2] is one of the products of our research activities. It enables network managers to communicate with computers via telephone without using computer terminals. Under WIDE/PhoneShell environment, network managers can send command and data to computers by using DTMF signals and computer can send messages to managers by pager, telephone, and facsimile. (Fig. 1) We can regard WIDE/PhoneShell as a front-end part of the ubiquitous network management systems. Thus, we decided to design and implement the back-end part. NMW system (Network Management Worm System) is our experimental implementation of the back-end part of our network management systems.

2 Network management on the internet

We have some problems in the network management method currently used on the internet. The followings are two of them.

1. Almost all networks are managed using their networks themselves. Once a network has malfunctions, network managers who are apart from their "home network" can't manage the network, because they can't access their network via the internet. We should have another network only to manage and control our networks. Fig. 2 shows a two layered model which is proposed by us. The first layer is named "service layer". Our computer networks, which we usually use them, are belong to this layer. The second layer is named "management layer". This layer is for network managers and network management tools only. Thus, links between each networks should not be as same as the links of the service layer. If we have networks based on this two layered model, we might have robust computer networks. However, almost all of us have to manage and control our networks using the same networks.

2. Almost all network management tools, which are currently available on the internet, doesn't have enough ability to control the network, if the network itself is in trouble. And if there are many hosts over slow and intermittent links and managers have to control these hosts, almost network management tools are of no use. Managers have to endure bad response and spend long time.

Thus, we need a new approach to keep our computer networks stable. This is a reason we designed and implemented NMW system [3, 4, 5, 6, 7].

3 Network management using NMW System

3.1 NMW system

NMW system is supported by three sub systems. They exchange data and commands for each other to manage their networks stable. As the result, data and commands move around various hosts. They seem to visit many hosts as if they were network worms [8]. Thus, we named our system as "Network Management Worm (NMW) system." NMW system has ability to send, receive and execute NMW.

In 1988, network worms appeared on the in-

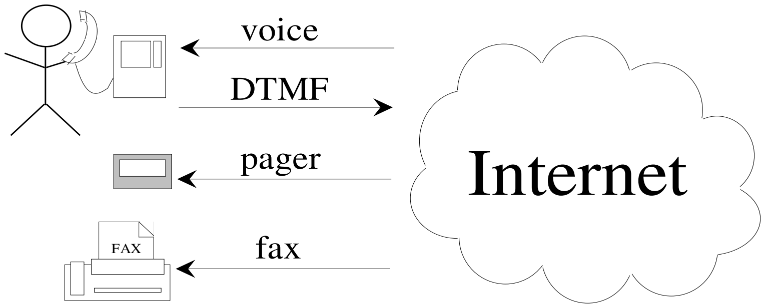

Figure 1: WIDE/PhoneShell

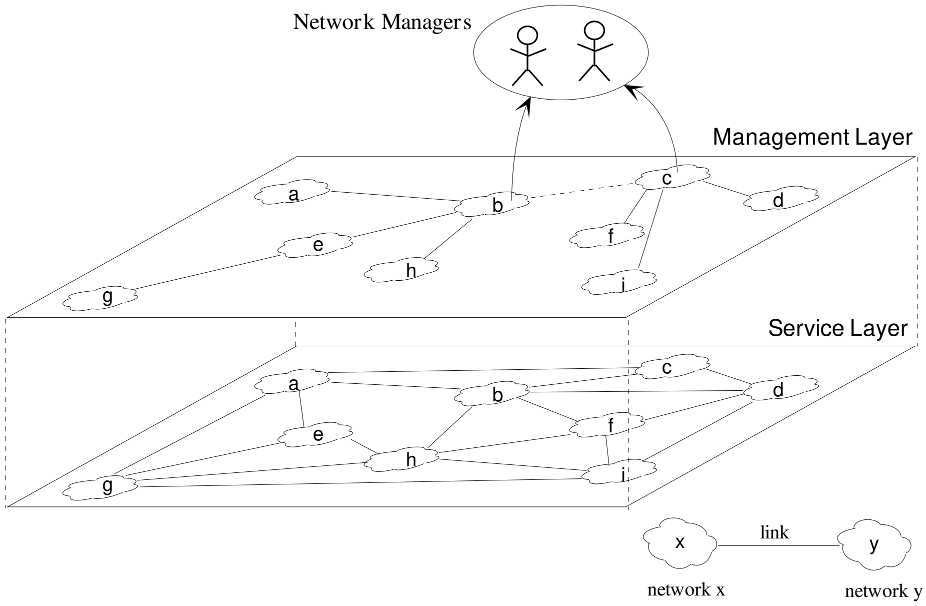

Figure 2: Two layers network model

ternet. They attacked and damaged many machines on the internet using several Unix security holes. These dangerous worms are known as the internet worm. In opposition to the internet worm, NMW is not dangerous, because the entity of NMW is just a simple list of data and commands. NMW can't move to other hosts by itself at all, but can move using previously installed programs that have authentication mechanism. NMW never uses security holes at all.

NMW System has been developing since 1992. The first version NMW was available at the end of 1992 in Tokyo Institute of Technology.

3.2 The features of NMW system

The features of NMW System are the following:

1. NMW system can exchange worms even if they don't have IP reachability. NMW system is designed to be able to exchange worms not only using IP but also using alternate ways such as E-mail, uucp and so on. This means NMW can move from network to network using these alternate ways. Even if some hosts are over the intermittent links, there is no problem. NMW can wait until the link to be established. (Fig. 3)

2. Once worms start from the origin host, they don't communicate the host until they go back to the host. Thus, network managers can use NMW system in high RTT environment where is traditional interactive operation hard to be used. We can also think that NMW system is somewhat improved remote job entry (RJE). (Fig. 4)

3. Worms can visit several hosts at one trip. When they move to other hosts, they don't have to go back to the origin host every time. This reduces network traffic.

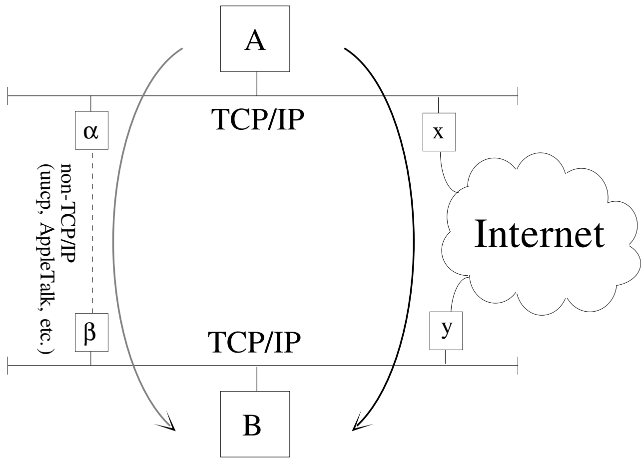

Figure 3: IP and non IP connections

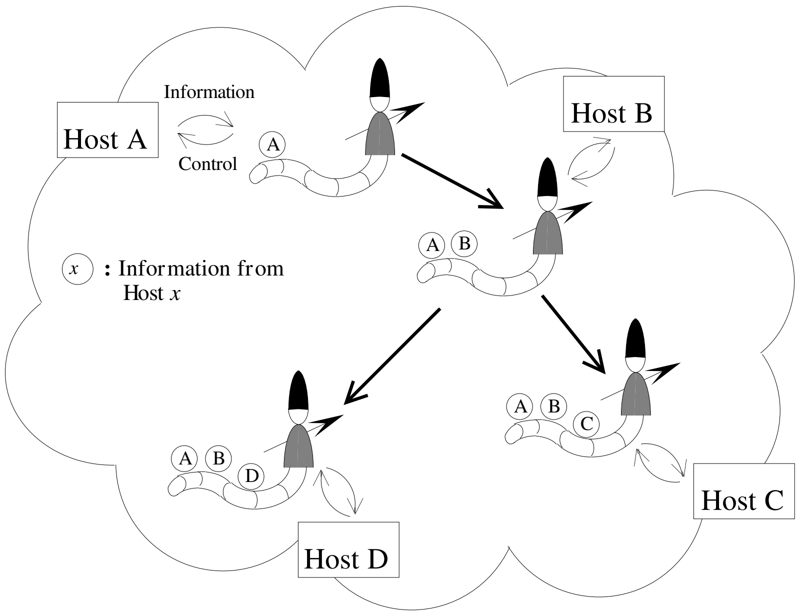

Figure 4: NMWs move around the network

3.3 Implementation of NMW

NMW system interprets and executes NMW using the following sub systems. They are WSD, OWL, and WORMC.

- WSD (Worm Support Daemon): Main part of NMW system. This sub system sends and receives NMWs. WSD also forwards received worms to OWL.

- OWL (Owl is Worm description Language): Interpreter of NMW. OWL read NMWs line by line and if each line is allowed to execute on the host, the line is executed. The current version of OWL is generated using scheme interpreter [9], because NMW has scheme based syntax. When they are sent to other hosts, they are ciphered by using PEM (Privacy Enhanced Mail) like mechanism to keep security and authentication [10, 11, 12].

- WORMC (WORM Controller): Controller of NMW system. WORMC is a user interface for the NMW system. Only network managers can control NMW using this system.

Fig. 5 shows relationship of the three sub systems. The three sub systems work cooperative to send, receive, control, and execute NMW.

3.4 Communications of NMW

In NMW system, only WSD can send and receive NMW. All hosts that can be reached directly from a certain host are called neighbor

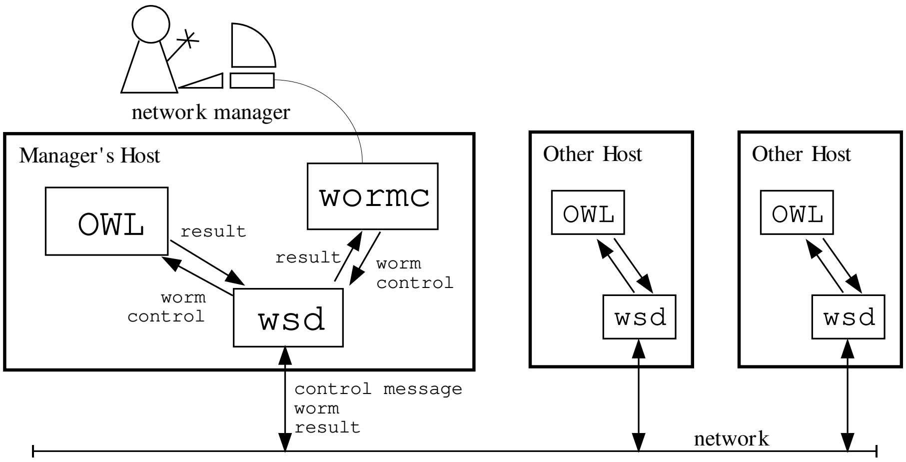

Figure 5: NMW system

host. A group of neighbor hosts that are connected to the same network is called neighbor host group.

In NMW system, there are three types of NMW. They are command worm, result worm and control worm. Command worm and control worm are always sent to neighbor host. This means command and control worm move in the network host by host. Result worm may be created when NMW is executed on certain host. This worm can go back to its origin host directly.

3.5 Related Approach

General Magic Corp. has been developing MagicCap and Telescript [13]. Their approach may be similar to our system. However, they are general purpose commercial products, but ours is specialized for the wide area network management. Thus, they are different. Note that all source codes will be available by anonymous FTP. Everyone can compare our NMW system with other related systems.

4 Experiments

We are testing NMW System in some networks that are belonging to the Department of Information Science, Tokyo Institute of Technology. Worms are patrolling many hosts and come back to the origin host with precise information when they find malfunctions of the network. They sometimes beep pagers of network managers with warning messages using WIDE/PCS (Pager Control System) (Fig. 6), if they cannot go back to the origin host because of network malfunctions.

Fig. 7 shows contents of patrol.nmw, a sample NMW. Fig. 8 and Fig. 9 show result. These result created by WSD and OWL at certain host. WSD at the host pushed result into NMW's body then send the NMW to next host. A sample NMW patrol.nmw returned with this result. The body of NMW is almost scheme, thus, WSD can put data and information into body of NMW with ease. If the next host is not alive (as shown in Fig. 9), NMW beeps managers' pager using WIDE/PCS to notify the trouble.

5 Conclusion

NMW system is designed and implemented for the nation wide network management. It is tested in some local area networks in Tokyo Institute of Technology and works well. We think it will be widely used in WIDE internet soon.

6 Acknowledgment

This research is a part of the research activity of WIDE project. We would like to thank all of the members of the WIDE project for their active discussions and suggestions.

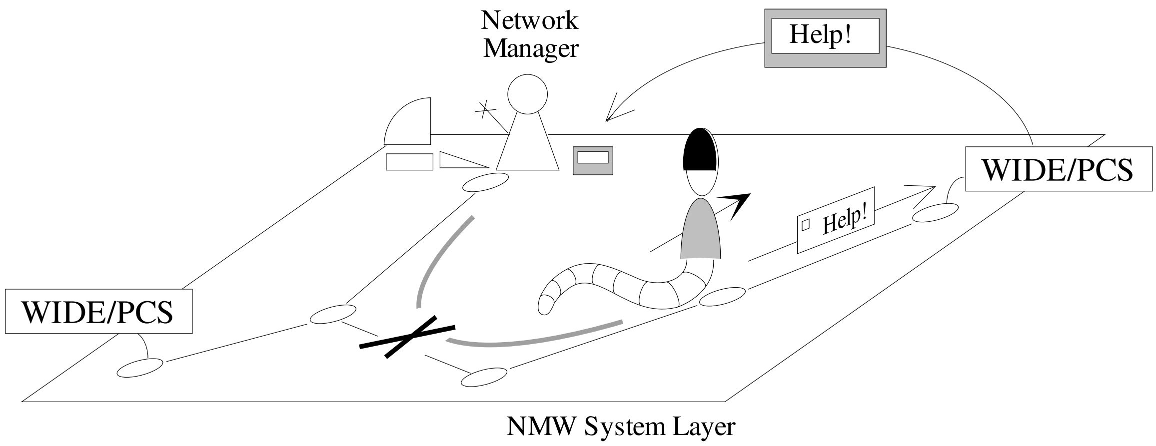

Figure 6: Linkage with WIDE/PCS

```
;;
;; patrol.nmw -- patrol hosts
;;
(define (get-next-host)
  (if (null? next-host-list)
      (begin
        (set! next-host-list travel-path)
        (if (and (> loop-upper-bound 0)
                 (>= loop-count loop-upper-bound))
            (begin
              (write-result "My work is finished.")
              (write-result "bye.")
              (quit)))
        (set! loop-count (1+ loop-count))
        (add-variable-to-self 'loop-count)))
  (let ((next-host (car next-host-list)))
    (set! next-host-list (cdr next-host-list))
    (add-variable-to-self 'next-host-list)
    next-host))
(define (main)
  (sleep interval-seconds)
  (write-result (hostname))
  (write-result (get-datetime))
  (write-result (get-load-average))
  (write-result (get-users-stat))
  (write-result (get-process-stat-byname "afsd"))
  (let ((next-host (get-next-host)))
    (if (ping next-host)
        (begin ;; ping OK
          (write-result (string-append
                         "next host: " next-host " is OK."))
          (write-result (string-append "I'm leaving "
                                       (hostname)))
          (go-to-host next-host))
        (begin ;; ping fail
          (write-result (string-append
                         "next host: " next-host " is down."))
          (notify-down-by-pager next-host)))))
;;
;; initialize globals
;;
(define travel-path '("host1" "host2" "host3" "host4"))
(define next-host-list travel-path)
(define loop-upper-bound 2)
(define loop-count 0)
(define interval-seconds (* 60 60)) ;; 1 hour
```

Figure 7: Example of NMW (patrol.nmw)

```
host1 ;; <-- hostname
Mon Jan 30 04:37:38 JST 1995 ;; <-- date
  4:37am  up 1 day, 10:52,  16 users,  load average: 1.23, 1.05, 1.00 ;; <-- load average
((xdm      console  Sat 6pm 34:17      6      6  sh /home/host1/kusano/.xsession )
 (akihiro  ttyp0    10:17pm  1:30                rlogin host2 )
 (m-oota   ttyp1    Sat 6pm 33:49                -tcsh )
 (akihiro  ttyp2    10:30pm     3     45      5  -tcsh )
 (onogi    ttyp3     1:52am  2:25                -tcsh )
 (yoshiaki ttyp4    11:25pm  4:47                -csh )
 (kusano   ttyp5    Sat 6pm  5:38   7:59   2:20  xload )
 (onogi    ttyp6     1:52am    34   2:16   1:57  /usr/local/lib/mule-2.2/19.28/mi)
 (m-oota   ttyp7    Sat 6pm    17      1         xcover )
 (tatsuya  ttyp8     3:29am    11      2      1  -csh )
 (kusano   ttypa     7:55pm    31   9:10   4:40  xload )
 (kusano   ttypb     7:55pm     1                perl /usr/local/bin/jisho2 )
 (izumi    ttypc    Sat 7pm 32:08      2         -csh )
 (izumi    ttypd    Sat 7pm 32:08      2         less /tmp/show005205_nakamori_20)
 (izumi    ttype    Sat 7pm 30:39                -csh )
 (kimoto   ttyq0    Sat 8pm 23:08                kinput2 -canna )) ;; <-- status of users
((root        39  0.0  0.7   468   276 ?  S    0:02 /usr/sony/etc/afsd -C -l 1)
 (akihiro  17455  0.0  0.8   528   336 ?  S    0:00 /usr/sony/etc/afsd -C -l 1))
;; <-- status of process "afsd"
next host: host2 is OK.
I'm leaving host1 ;; <-- move to next host
```

Figure 8: Result of patrol.nmw (1)

```
host3 ;;  <-- hostname
Mon Jan 30 04:18:16 JST 1995 ;; <-- date
  4:18am  up 34 days,  7:44,  14 users,  load average: 1.29, 1.23, 1.00 ;; <-- load average
((xdm      console  Wed 3pm 5days     13     13  xbiff )
 (hohno    ttyp0    12Jan95  3:24  14:55     46  -tcsh[hohno from nocn2] )
 (hohno    ttyp1    12Jan95  1:22  25:32   1:06  -tcsh[hohno from nocn2] )
 (hohno    ttyp3    12Jan95 28:08   1:32      9  -tcsh[hohno from nocn2] )
 (hohno    ttyp5    13Jan95 28:03   1:06      5  -bin/tcsh )
 (ookubo   ttyp6    Thu11am 3days      2         -csh )
 (ookubo   ttyp7     6:30pm  1:38      1         -csh )
 (mori     ttyp8    Sun12am  1:46  30:36  16:36  dclock )
 (hohno    ttyp9    Wed 2pm 31:20                -csh )
 (tatsuya  ttypb    Sat 2pm 2days     29      2  -csh )
 (mori     ttypc     1:04pm    37 150:05      9  -csh )
 (akihiro  ttypf    10:17pm     7     50     25  xdvi -paper a4 main.dvi )
 (akihiro  ttyq0     2:21am    36                -tcsh )
 (mori     ttyq2    10:37pm  1:46                -tcsh[mori from host3] )) ;; <-- status of users
((root        31  0.0  0.2   468    56 ?  S    0:11 /usr/sony/etc/afsd -n host3 Sony NEWS -l 2))
;; <-- status of process "afsd"
next host: host4 is down.  ;; <-- can't move to the next host.
```

Figure 9: Result of patrol.nmw (2)

References

[1] Hiroyuki Ohno. Improved Network Management using WIDE/PhoneShell. In *Proceedings of INET '93*, 1993.

[2] WIDE Project. WIDE Project Annual Report 1992 (Part.16: WIDE/PhoneShell). Technical report, WIDE Project, 1993.

[3] WIDE Project. WIDE Project Annual Report 1993 (Part.16: WIDE/PhoneShell). Technical report, WIDE Project, 1994.

[4] Akihiro SHIMIZU and Hiroyuki OHNO. Network management using network worm. In *Proceedings of 47th annual conference (1)*, pages 1–299. IPSJ, 1993 (In Japanese).

[5] Akihiro SHIMIZU and Hiroyuki OHNO. Network management system using network management worm. In *Proceedings of distributed system management research group*, pages 101–109. IPSJ, July 1994 (In Japanese). DSM-9407012.

[6] Akihiro SHIMIZU and Hiroyuki OHNO. Network management using nmw and new application of nmw. In *Proceedings of disributed system and multi media work shop*, pages 101–109. IPSJ, October 1994 (In Japanese).

[7] Akihiro SHIMIZU and Hiroyuki OHNO. Resource finding using nmw system. In *Proceedings of 49th annual conference (1)*. IPSJ, September 1994 (In Japanese).

[8] John F. Schoch and Jon A. Hupp. The 'Worm' Programs — Early Experience with a Distributed Computation. *Communications of the ACM*, 25(3):172–180, March 1982.

[9] H. Abelson, R. K. Dybvig, C. T. Haynes, G. J. Rozas, N. I. Adams IV, D. P. Friedman, E. Kohlbecker, G. L. Steele Jr., D. H. Bartley, R. Halstead, D. Oxley, G. J. Sussman, G. Brooks, C. Hanson, K. M. Pitman, and M. Wand. Revised^4 report on the algorithmic language scheme, November 1991.

[10] WIDE Project. WIDE Project Annual Report 1993 (Part.10: Security). Technical report, WIDE Project, 1994.

[11] Akihiro SHIMIZU and Hiroyuki OHNO. Security improvement of nmw system. In *Proceedings of 49th annual conference (1)*. IPSJ, September 1994 (In Japanese).

[12] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. Request for Comment 1155, May 1990.

[13] Barbara Knaster. *Presenting MAGIC CAP*. Addison Wesley, 1994.

[14] R. Droms. The Knowbot Information Service. Internet Draft, December 1989.

[15] Jeffrey D. Case, Mark S. Fedor, Martin L. Schoffstall, and James R. Davin. A Simple Network Management Protocol (SNMP). Request for Comment 1157, May 1990.

[16] Keith McCloghrie and Marshall T. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. Request for Comment 1213, March 1991.

[17] Marshall T. Rose. *The Simple Book*. Prentice-Hall, 1991.

[18] Kornel Terplan. *Communication Networks Management*. Prentice-Hall, second edition, 1992.

Author Information

Hiroyuki Ohno recevied the Ph.D. degree in computer science from Tokyo Institute of Technology in 1993. He is a assistant professor of the Network Operation Center of Tokyo Institute of Technology. His current interests include high-speed wide area networks and network management technology on wide area computer networks. He is a member of WIDE project, Japan Society for Software Science and Technology(JSSST), Information Processing Society Japan(IPSJ), and a Pioneer Member of the Internet Society.

Akihiro Shimizu received the MS degree in computer science from Tokyo Institute of Technology. He graduated the graduate school of Tokyo Institute of Technology in 1995 and he is working for NTT. His main interest is wide area computer network management using agent systems. He is a member of IPSJ (Information Processing Society Japan).