NMWシステムを用いたネットワーク管理系の運用報告と、新たな応用分野に関する検討*

清水 亮博†
東京工業大学 理工学研究科 情報科学専攻‡
akihiro@is.titech.ac.jp

大野 浩之§
東京工業大学 Titanet 運用センター¶
hohno@is.titech.ac.jp

著者らは，1992年度よりネットワークワームを利用したネットワーク管理手法について研究を続けている．そして，ネットワークワームによるネットワーク管理モデルの提起や，そのモデルに基づく最初の実装（NMW (Network Management Worm) と命名）をすでに終え，ネットワーク間が障害によって寸断されたような状況であってもNMWがネットワーク管理に大きく貢献することを指摘した．ところで従来の報告では，NMWシステムの設計と実装上に問題点に関する議論が主であったが，今回の報告では，実際にネットワーク管理に投入して得られた運用結果とその評価を詳細に報告する．次にNMWシステムが，情報検索システムや，生成消滅を繰り返す系のシミュレーションなどに応用できることを指摘する．これらは，NMWの本来の用途ではないが，分散環境上のエージェントシステムのテストベッドとして利用可能であることを示唆するものである．

1　はじめに

著者らはネットワークワーム (以下ワーム) を利用したネットワーク管理手法について研究を続けている。これは、ワームをネットワーク管理に用いると、以下のような特徴が期待されたためである。

- ネットワーク上を移動しながらの調査、分析、制御が可能となる。

  ワームの移動の際に、ワームのプログラム本体のみに限らず、訪れたホストで得た情報も移動可能ならば、複数のホストの状況に応じた作業ができる (図1)。この特徴を用いて、ネットワーク上を移動しながらの調査、分析、制御等が可能となる。

- 管理作業に必要な通信量を削減できる。

  ワームのプログラムによって、必要な情報のみを選別できるので、作業に必要な通信量が削減できる。WIDE/PhoneShell[9] などの極端に回線容量が低い環境から管理者がネットワークを操作する場合や、ネットワークが輻輳を起こしている際に、この特徴は有効である。

- ネットワークの障害時にも対応できる。

*NMW System: It's current status and a study on its new application
†Akihiro SHIMIZU
‡Department of Information Science, Graduate School of Science and Engineering, Tokyo Institute of Technology
§Hiroyuki OHNO
¶Network Operation Center, Tokyo Institute of Technology

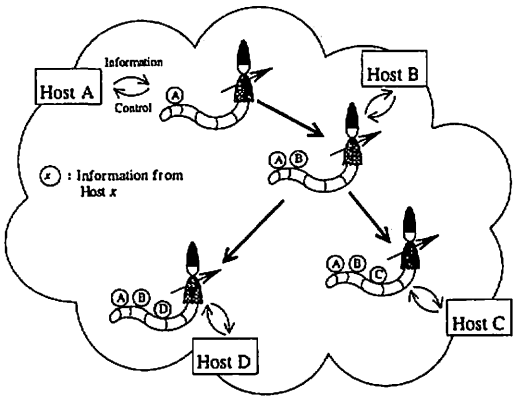


図 1: 計算機間での移動に伴う調査、分析、制御

ネットワークが障害を起こした場合、通常の方法では障害によって切断されたネットワークの状態を知ることはできない。しかしワームを用いれば、ワーム自身の判断で障害に対応することができる。対応の例としては、障害が復旧するまで待つ方法や、管理用の経路でワームの実行結果を管理者に送る方法などがある。

そこで、著者らは1992年度より NMW システム (Network Management Worm System) を開発中である [6]。現在では上記の特徴をほぼ確認できたと考えている。

NMW システムは、次の3項目を目標に開発を進めている [5]。

- 新しい形態のネットワーク管理ツールの提案
- サービス層 [2] の障害にかかわらず動作する管理層の実現
- ワームを実用に用いる際の制御技術の確立

NMW システムは、ワームとワーム支援系から構成されている。ワーム支援系の概略を図 2に示す。

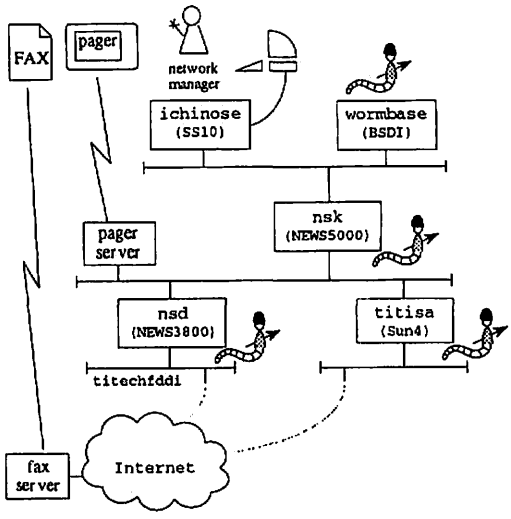


図 3: 実験ネットワークの構成

またNMW システムは WIDE/PhoneShell との連携等により、ネットワークが寸断されたような状況でも対応できることを示した [4]。

なお、現在の NMW システムは、文献 [4, 8] で示した暗号化手法等を実装中である。

2 運用報告

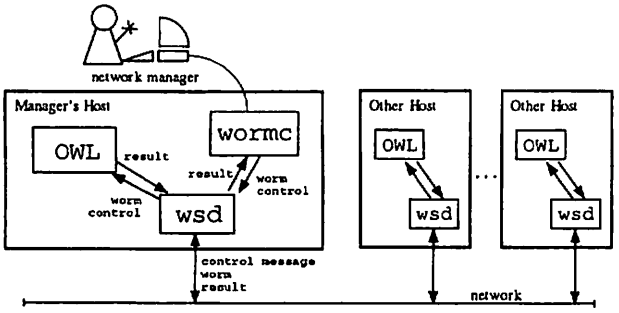
本節では NMW システムの運用例を 2 つ取り上げる。なお、これらの実験を行なったネットワーク構成の概略を、図 3に示す。

2.1 ログインホスト以外からの監視

UNIX の ping コマンドは ICMP パケットを目的のホストへ送り、その返事が返ってくることでホスト間の接続を調べている。

この ping を用いて他のホストを定期的に監視する場合、管理者は監視を行なうホストへ一旦ログインしなければならない。また、監視対象の変更の際も同様である。これを、NMW システムを用いて実装したのが図 4の pingd.owl[1]である。pingd.owl は図 3において、ホスト ichinose よりワームを実行させるとホスト nsk, nsg を経由して ksa に到着

[1]なお、これは文献 [4] のものを改良し、恒常監視向けにしたものである。



wsd　ワーム支援デーモン。システムの通信全てを扱う。
owl　ワーム実行インタプリタ。
wormc　ワーム制御ユーザインターフェース。

図 2: ワーム支援系の構造

する。ksa では一定時間毎[2]に、wormbase に対して ping パケットを送って wormbase - ksa 間の接続を調べている。接続が切れた状態が 3 回以上続くと、WIDE/PhoneShell を用いてページャによって切断したことを通知する。また接続が復旧し、3 回以上 ping パケットが返ってくると、ページャによって復帰したことを通知する。これは、短い周期で接続が切断と再接続を繰り返した時に、切断、再接続の度にページャにメッセージを送らないようにするためである。

pingd.owl はワームであるため、監視対象等を変更する際に ksa にログインする必要がない。これは、回線容量に余裕のないネットワークや、非常に遠いネットワーク間で調べる際に重要な特徴である。

図 5に pingd.owl が実行を開始した後、wormbase のネットワークケーブルが外れる事故が起こった場合の実行結果を示す。

なお、この pingd.owl を図 3のネットワークで実際に運用した時に、このワームがどの程度管理者に通知するかを調べた。ここでは 3 つのホストについて、wtmp ファイルの中身よりどの程度システムがダウンしていたかを推測した。その結果を表 1に示す。

表 1より、このワームはシステムダウンし

[2]ここでは 10 秒毎

```
ksa
Mon Sep 19 04:44:21 JST 1994: wormbase is up.
Mon Sep 19 04:44:31 JST 1994: wormbase is up.
Mon Sep 19 04:44:41 JST 1994: wormbase is up.
Mon Sep 19 04:44:52 JST 1994: wormbase is up.
Mon Sep 19 04:45:02 JST 1994: wormbase is up.
Mon Sep 19 04:45:22 JST 1994: wormbase is down.
Mon Sep 19 04:45:42 JST 1994: wormbase is down.
Notify DOWN to manager.
Mon Sep 19 04:46:02 JST 1994: wormbase is down.
Mon Sep 19 04:46:22 JST 1994: wormbase is down.
Mon Sep 19 04:46:33 JST 1994: wormbase is up.
Mon Sep 19 04:46:53 JST 1994: wormbase is down.
Mon Sep 19 04:47:03 JST 1994: wormbase is up.
Mon Sep 19 04:47:13 JST 1994: wormbase is up.
Mon Sep 19 04:47:23 JST 1994: wormbase is up.
Notify UP to manager.
Mon Sep 19 04:47:34 JST 1994: wormbase is up.
Mon Sep 19 04:47:44 JST 1994: wormbase is up.
（以下略）
```

図 5: pingd.owl の実行結果

やすいホストに対しては有効であるが、それ以外のホストに対してはあまり有効ではないと考えられる。しかし、本来 pingd.owl は広域ネットワークでのネットワークの接続を調べることを目的としているので、この結果はやむを得ない。

2.2 ルーティングの確認

ルーティング情報の混乱によって、ホスト間で接続できないことがある。このような時通常は traceroute コマンドで調べるが、この方法は到達可能なホストはわかるものの、各

```
;;
;; pingd.owl Version 2.
;;    --- 恒常監視 version

;; globals
(define fail-count 0)
(define success-count 0)
(define loop-count 15)
(define loop-interval-sec 10) ;; 10sec
(define check-target "wormbase")
(define fail-notified nil)
(define hosts-on-path '("nsk" "nsg" "ksa"))

(define (ping-check target)
  (define (do-check)
    (if (ping target)
        (begin ;; ping成功
          (write-result
            (string-append
            (get-datetime)
            ": " target " is up."))
          (set! success-count
            (1+ success-count))
          (if (>= success-count 3)
            (if fail-notified
                (begin
                  (notify-up-by-pager target)
                  (write-result
                    "Notify UP to manager.")
                  (set! fail-notified nil)))
            (set! fail-count 0)))
        (begin ;; ping失敗
          (set! success-count 0)
          (set! fail-count (1+ fail-count))
          (if (and (not fail-notified)
                   (>= fail-count 3))
              (begin
                (notify-down-by-pager target)
                (write-result
                  "Notify DOWN to manager.")
                (set! fail-notified t)
                (set! fail-count 0)))
          (write-result
            (string-append
              (get-datetime)
              ": " target " is down."))))))
  (do-check)
  (loop (-1+ loop-count)
        (lambda ()
          (sleep loop-interval-sec)
          (do-check))))

(define (main)
  (if (not (null? hosts-on-path))
      ;; 監視するホストへ移動
      (let ((next-host (car hosts-on-path)))
        (set! hosts-on-path (cdr hosts-on-path))
        (add-variable-to-self 'hosts-on-path)
        (go-to-host next-host))
      ;; targetを監視
      (begin
        (write-result (hostname))
        (ping-check check-target))))
```

図 4: ワームの例 (1): pingd.owl

ホストのルーティングテーブルの内容まではわからない。そこでルーティングテーブルを調べながら、目的ネットワークへ到達可能かどうかを調べるワーム route.owl を作成した(図 6)。

図 3のネットワークで、 ichinose から titechfddi ネットワークへのルーティングを調べた結果を図 7、8に示す。図 7は正常な状態、図 8は実験中に nsd がシステムダウンした際の状態を示している。実際に図 8 のような実行結果が得られた場合には、結果より titisa または nsk のルーティングテーブルがおかしいことがわかるので、この結果を手がかりとしてさらに詳しく調べることができる。

3 ネットワーク管理以外への応用

NMW システムはネットワーク管理を第一目標として開発されているが、ネットワーク管理以外の目的にも応用できる。ここでは情報検索と、シミュレーション、さらに最近現れ出したエージェントシステムについて述べる。

3.1 情報検索システム

今後、Internet等のコンピュータネットワークが普及するにつれ、World Wide Web のような広域ネットワークに分散された情報検索システムが普及していくものと思われる。

このようなシステムにおいて、情報を検索

ホスト	主な用途	調査期間	ダウン回数	月毎の平均
A	NIS、Mail	2/1/94 ～ 9/18/94	4 回	0.5 回
B	NetNews, router(FDDI x 1, Ether x 2)	2/1/94 ～ 9/18/94	38 回	5 回
C	router(Ether x 2)	8/19/94 ～ 9/18/94	0 回	0 回

表 1: ホスト毎のシステムダウン回数

```
;;
;; route.owl
;;
;;  globals
(define travel-path ())
(define target-network "titechfddi")
;;
;; main routine
(define (main)
 (write-result (hostname))
 ;; check loop
 (if (not (null? travel-path))
   (if (in-string-list? (hostname)
                        travel-path)
     (begin
      (write-result "Routing is looped.")
      (quit))))
 (set! travel-path (cons (hostname)
                         travel-path))
 (add-variable-to-self 'travel-path)
 (write-result travel-path)
 ;; check next host
 (let ((next-host (find-route
                    target-network)))
  (if (string? next-host)
    (begin
     (write-result next-host)
     (if (this-host? next-host)
       (begin
        (write-result
          "Next host is this host.")
        (write-result
          "trace is completed.")
        (quit)))
     (if (not (ping next-host))
       (begin
        (send-result "Next host is down.")
        (quit)))
     (if (has-wsd? next-host)
       (go-to-host next-host)
       (write-result
         "Next host don't have wsd.")))
    (write-result
      "Oops. there is no route!"))))
```

図 6: ワームの例 (2): route.owl

する方法は 2 つ考えられる。一つは関係のあるデータを全て一箇所に集め、そこで検索を行なうものである。例としては、archie などがある。

これとは逆に、検索を分散して行なう方法が考えられる。分散して検索する方法は、さらに以下の 2 つに分類できる。

すべてのホストで同時に検索する方法

検索要求に対して全ての情報検索サーバで同時に検索を行なうものである。この方法は全数検索が必要な場合には効率的であるが、そうでない場合は非効率である。

関連するデータを辿って検索する方法

全数検索でない場合は、こちらがの方法が有効である。関連するデータを辿りながら目的のデータへ到達する際に、検索命令を出すオブジェクトは特定のホストから動かないものと、ホスト間を移動するもの、すなわちワームを用いるものが考えられる。

この場合、検索命令とその結果のデータ量が少ない場合は前者の方が有利であるが、これらが多くなると検索自体ではネットワークを使用しないワームを用いる方法が有効である。

情報検索をワームによって行なうアルゴリズムは、次のようになる。

```
1: nsd
2: (nsd nsk ichinose)
3: nsd-f
4: Next host is this host.
5: trace is completed.
```

nsd におけるワームの実行結果

行数	意味
1	ホスト名
2	途中経路（逆順）
3	ルーティングテーブルが指しているホスト
4, 5	ルーティングテーブルが nsd を指しているので終了

図 7: route.owl の実行例 (1): 正常終了時

```
1: nsk
2: (nsk ichinose)
3: titisa
```

nsk におけるワームの実行結果

本来は、3 行目は nsd を指しているはずだが、nsd が落ちているため titisa を指している。そのため、

```
1: titisa
2: (titisa nsk ichinose)
3: Oops. there is no route!
```

titisa におけるワームの実行結果

titisa ではルーティングがないので、異常終了する。

図 8: route.owl の実行例 (2): 異常終了時

あるホストにおいて、

1. データがループしていないか、これまで辿ってきたデータと照合し、ループしていたら終了。
2. そのホスト内でデータを検索し、
   - 目的のデータであれば、ユーザに送り終了する。
   - 関連するデータを持つホストへ、ワーム自身のコピーを送る。

3.2 シミュレーション

NMW システムをシミュレーションに使用する場合の、シミュレートではない実際のネットワークの挙動を簡単に得ることができる利点がある。例えば、分散処理における負荷分散のアルゴリズムをシミュレーションで検証する際、一般的なシミュレーション手法ではネットワークの負荷等を乱数を用いて生成する。しかし、この方法は生成する値が妥当なものであるかの検証が必要で、その上時刻によって変動する要因などを見落す可能性がある。

このシミュレーションを NMW システムによって行なえば、乱数による疑似的な環境ではなく、実際の環境でのシミュレーションが行なえる。具体的には、対象のアルゴリズムを実装したワームを用意し、必要な数だけ実行するだけでよい。あとは、ワームの挙動を観察すれば、それがシミュレーションの結果となっている。

このようなシミュレーションを行なう場合、各ワームがホストにかける負荷と、実際のプロセスがかける負荷は当然違うので、負荷を換算して各ワームに伝える必要がある。そのため、この違いを換算する仮想的な計算機の負荷を表す機構が必要である。この機構を実装するには NMW システムの wsd の機能を拡張する方法と、ワーム間通信[3]を用いて負荷を計算するワームを用意する方法が考えられる。

3.3 エージェントシステムのテストベッド

最近 General Magic 社の Telescript[1] 等に代表される、エージェントシステムと呼ばれる分散システムが商用のシステムとして出始めている。エージェントシステムの定義は未だ定まっていないが、通信時に純粋なデータではなくプログラムを送り、送り先で必要な動作を行なってからその結果を得るシステムのようである。

これは、筆者らが採用 [6] したネットワークワームの定義と本質的に変わるところがない。したがって、NMW システムはエージェ

[3] 4.1節参照

ントシステムのテストベッドとして用いることも可能である。

4 今後の方針

4.1 ワーム間通信

NMW システムの次の目標の一つに、ワーム間通信の設計および実装がある。現在のところ、NMW システムにはワーム間で直接通信を行なう方法はない。しかし複数のワーム間での情報交換、ワームによる他のワームの起動等を行なうために、ワーム間通信が必要であると判断した。

ここで、ワーム間通信を行なうより、ワーム自身が相手のワームがいるホストまで移動して情報交換を行なった方が良いのではとの意見もありうる。しかしワームの転送は、TCP 等の転送を保証されるプロトコルで行なわれるので、次のような問題点がある。

不必要な通信が行なわれる。 通信には、信頼性を要求しないものもありうる。例えば、あるワームが近くにいるワームのどれかと連絡を取ろうとした時、そのメッセージは近くの全てのワームに届く必要は無く、どれか一つに届けば良い。

またネットワークが輻輳を起こしそうな時にはなるべく通信量を減らしたい。このような場合には TCP 等の通信量が多いプロトコルより、UDP 等の比較的通信量が少ないプロトコルを使いたい。

なお通信を確実に行ないたい場合でも、通信コストが無視できないほどワーム本体が大きい場合には、ワーム間通信で済めばネットワークにかける負担が小さくなる。

必ずパケットが双方向に送られる。 TCP 等の信頼性のあるプロトコルでは、何らかの応答メッセージが必ず受け側から送り側に送られる。したがって、例えばルーティングの調査において、あるホスト A から ホスト B へのパケット到達可能性と、逆に B から A へのパケット到達可能性を別々に調べたい場合など、パケットが一方にのみ送られなければならない場合には、ワーム自身が転送されるかどうかで調べることはできない。

通信相手の指定方法は、以下のような要素について指定できるようにする。

- 相手のワーム ID
  ワーム ID には、ワームの発信ホスト、ユーザ名を含む。
- 相手のワームがいるホスト
  具体的なホストだけでなく、いまいるホストから N ホップ以内といった指定も含む。
- その他

なお、通信相手のワームがいるホストがわからない場合には、各ホストの wsd が持っているワームのログに、そのホストへのワームの到着/出発情報が含まれているので、これを調べることで、相手のワームがいるホストを調べることができる。

4.2 ネットワークの管理自動化

ネットワーク管理の究極の姿は、ネットワークを構成しているコンピュータ自身による自立的な管理であると、筆者らは考えている。このようなネットワーク管理をワームを用いて実現するために、次のようなモデルを考えた。

「ネットワーク内で生活しているワームが、ワーム自身が生きのびるために環境としてネットワークを制御する。ユーザはその副産物としてネットワークサービスを受けることが可能となる。」

このようなワームを実現する際の一般的な問題点は、ネットワーク上の現象やオブジェクトが正常であるか、異常であるかを見分けるための手法である。例えば、人間の管理者がネットワーク上のあるサービスの動作が異

常だと判断するのは、サービスを提供しているホストがネットワークに接続されて稼働中のはずであり、自分はサービスを利用する権限があること、および正常なサービスの動作と、異常なものを区別できるためである。これと同じことをワームに行なわせようとすると、サービスを提供しているホストの存在、稼働状況、利用権限等は対応する情報をワームによって収集し、判断することで比較的容易に調べられる。しかし、サービス自身の動作が正常であるかを一般的に判断することは困難である。例えば、各サービスの動作状況を何らかの方法で収集、蓄積しておき、その蓄積された情報と状態を比較することで正常、異常を見分けるような方法が考えられる。

このようなシステムでの、ワームの形態を主にワーム自身の規模によって分類すると、以下の3種類に分類できる。

白血球ワーム このワームは規模がもっとも小さいものである。ワームはあたかも人体の白血球のようにネットワーク上に多数が同時に存在し、常にネットワークを監視する。このため障害発生時等に素早く対応できると期待される。ネットワークに対する負荷を考えると、この方法は資源を多く必要とする手法を用いるワームには使えない。

定期点検ワーム このワームはネットワークの各部分を定期的に訪れ、その一部分の範囲内の状況を調べ、必要な作業を行なうものである。現在のInternetのようにネットワークの構造に階層構造が前提とされており、ネットワークを部分[4]に分けて考えられるなら、この手法は使用できる。また、前述の白血球ワームよりネットワークにかける負荷は小さいが、障害等発生後即時に対応することはできない。

受動的情報収集/緊急出動ワーム このワームは 通常時には受動的な情報収集とその判断のみを行ない、能動的な活動はしない。情報を収集する方法としては、文献[7]で提案した他のワームからの情報を収集する方法等が考えられる。能動的な活動を始めるのは、収集した情報からワーム自身が動作すべき事態[5]、すなわち緊急事態と判断した場合である。この形態をとるワームは、ネットワークの障害に自動的に対処するもので、知識ベース等を備えたために、実行コストがかなり大きくなったものなどが考えられる。

5 おわりに

本論文では、まずネットワークワームを利用したネットワーク管理システムであるNMWシステムの運用に関して報告した。

次に、ネットワーク管理以外の新しい応用分野を検討し、NMWシステムが他の分野にも応用できることを指摘した。

なお、現在のNMWシステムに欠けているワーム間通信についての検討、将来のネットワーク管理モデルに関する展望も行ない、現在のアプローチの有効性を確認した。

謝辞

本研究について、WIDEプロジェクトの研究者からさまざまな助言を得た。ここに記して感謝する。

参考文献

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

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

[3] John F. Schoch and Jon A. Hupp. The 'Worm' Programs — Early Experience with

[4] 例：一つのサブネット、同じネットワークアドレスのネットワーク等。

[5] この中には他のワーム等から通報を受けた場合等も含まれる

a Distributed Computation. *Communications of the ACM*, Vol. 25, No. 3, pp. 172–180, March 1982.

[4] 清水亮博, 大野浩之. ネットワークワームを利用したネットワーク管理機構. 分散システム運用技術研究グループ資料, pp. 101–109. 情報処理学会, July 1993. 資料番号 DSM-9407012.

[5] 清水亮博, 大野浩之. ネットワークワームを利用したネットワーク管理手法. 第 47 回 (平成 5 年度後期) 全国大会 講演論文集 (1), pp. 1–299. 情報処理学会, 1993.

[6] 清水亮博. 寄生プログラム (ネットワークワーム) を用いたコンピュータネットワークの管理方式. 平成 4 年度卒業論文, 東京工業大学理学部情報科学科, February 1993.

[7] 清水亮博, 大野浩之. ネットワークワームを用いたネットワーク管理システムの資源探索に関する一考察. 第 49 回 (平成 6 年度後期) 全国大会 講演論文集 (1). 情報処理学会, September 1994.

[8] 清水亮博, 大野浩之. ネットワークワームを利用したネットワーク管理システムのためのセキュリティ向上手法. 第 49 回 (平成 6 年度後期) 全国大会 講演論文集 (1). 情報処理学会, September 1994.

[9] ＷＩＤＥプロジェクト. 1992 年度 ＷＩＤＥプロジェクト研究報告書 (第 16 部 WIDE/PhoneShell). Technical report, ＷＩＤＥプロジェクト, 1993.