NMW System によるネットワーク管理とその評価

東京工業大学大学院
情報理工学研究科 数理・計算科学専攻
小野木 渡
(学籍番号 95M37052)

平成8年度修士論文

指導教官 大野浩之講師

平成9年1月24日

目 次

1 はじめに 4

2 研究の背景 5

2.1 ネットワーク管理 5

2.1.1 SNMP によるネットワーク管理 5

2.1.2 従来のネットワーク管理における問題点 6

2.2 ワームを用いたネットワーク管理 7

2.3 エージェントシステム 7

2.3.1 ネットワークエージェント 7

3 NMW System の設計と実装 10

3.1 NMW System の目標 10

3.1.1 NMW System の開発方針 10

3.1.2 管理者にかかる負担の軽減 12

3.2 NMW System のしくみ 12

3.2.1 NMW System の支援系 12

3.2.2 ワームの種類 13

3.2.3 WIDE/PhoneShell との協調 13

3.3 セキュリティの向上 14

3.3.1 ワームの暗号化 14

3.3.2 ワームの認証 17

3.3.3 アクセスコントロール 17

3.4 NMW System の API 18

3.4.1 ワームの作成 19

3.4.2 ワームの実行 20

3.4.3 API の今後 21

4 NMW System の実験と評価 22

4.1 実験環境 22

4.2 NMW System のインストール 23

4.3 NMW System の実行 23

4.4 ワームの実行例 23
4.4.1 情報収集ワーム 23
4.4.2 ファイル収集ワーム 24
4.5 一般ユーザによる利用 24
4.6 実行時間 25
4.6.1 暗号化時間 25
4.6.2 ワームの実行時間 26
4.7 実験評価 27

5 NMW System の利用例とその評価 34
5.1 公衆情報端末環境における利用 34
5.1.1 公衆情報端末の管理 34
5.1.2 暗号化ダンプ 35
5.2 NMW System の応用 36
5.2.1 NMW System を応用したサービス 36
5.2.2 サービスの実装 37
5.2.3 サービスの評価 ・考察 37

6 考察 38
6.1 セキュリティ 38
6.1.1 安全性についての議論 38
6.1.2 管理者のアカウントが盗まれた場合 38
6.2 他のエージェントシステムとの比較 39
6.2.1 ネットワーク管理 39
6.2.2 作業内容の制限 39
6.3 他のシステムとの協調 40
6.3.1 システム管理支援機構 magP 40
6.3.2 情報可聴化システム stetho 41
6.4 NMW System の今後 42
6.4.1 ネットワーク管理 42
6.4.2 エージェントシステム 42

7 結論 43

図の一覧

2.1 従来のネットワーク管理 6
2.2 ワームによるネットワーク管理 8

3.1 ネットワーク管理モデル (文献 11
3.2 NMW System 支援系 13
3.3 WIDE/PhoneShell との協調 14
3.4 記号の説明 15
3.5 nmw ユーザと wsd の鍵 16
3.6 NMW System の利用者と wsd の鍵 17
3.7 OWL を実行する環境の制限 18
3.8 ワームヘッダ 19
3.9 ワーム実行 API の一部 20

4.1 実験ネットワーク 22
4.2 暗号化ワームの送信 24
4.3 送信するワーム (ps-df-flood.owl) 25
4.4 暗号化されたワーム 26
4.5 ワームを送信したホストのログ 28
4.6 ワームを受け取ったホストのログ 29
4.7 情報収集ワームの結果 30
4.8 ファイル収集ワーム 31
4.9 暗号化され転送されたファイル 32
4.10 メール収集ワーム 33
4.11 wsd による owl の実行 33

5.1 NMW System を用いた暗号化ダンプ 35

6.1 リモートプログラミング言語との比較 40
6.2 magP、stetho との協調 41

第 1 章

はじめに

従来のコンピュータネットワークの管理手法には、管理対象ホストにログインして作業する方法や、SNMP を利用した管理アプリケーションを利用する方法などがある。これらの方法ではネットワークの規模が大きくなりその構成が複雑になると、管理者の負担が増大してしまう。したがってネットワークが現在のペースで発展を続けると、管理者の作業量や問題の複雑さなどの理由により、従来の管理手法ではネットワークを扱えなくなる恐れがある。また従来の管理手法では、ネットワークに障害が起きた場合、障害が起きたこと自体は検出できるが、その時点におけるネットワーク全体の状況を取得したり、ネットワークを制御したりすることが困難である。

これらの問題は、従来の管理手法そのものに原因がある。従来の管理手法では、管理を行う管理者や管理者を支援するシステムが各管理対象ホストに固定されおり、管理者はある特定のホストからリモートログインしたり、管理ソフトウエアを用いて通信したりすることで、ネットワークや管理対象の状態を調べている。

これに対して、管理エージェントが管理対象ホストに固定されずに、ネットワーク上を移動しながら自分自身の判断でネットワークを調べる方法がありうる。この移動する管理エージェントにはホスト間を移動するプログラム、すなわちネットワークワームを利用できる。

NMW System (Network Management Worm System) は、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。本システムは 1992 年度から研究が進められ、1994 年度までに基礎的な部分の開発を終えた。筆者は、1994 年度より研究に参加し、1995 年度からはその中心となり研究を続けてきた。

本論文では、第 2 章で NMW System の背景として従来のネットワーク管理について述べた上で、ネットワーク管理にワームを応用することを提案する。そして、NMW System の用いた手法が一般にネットワークエージェントと呼ばれることを述べ、その概念を説明し、関連研究について述べる。第 3 章では、NMW System の設計と実装について詳しく述べる。第 4 章では、NMW System の実行結果とその評価を行う。第 5 章では NMW System の利用例、応用例について述べる。第 6 章では考察として、エージェントシステムとしての NMW System、他のシステムとの比較、他のシステムとの協調、今後の課題について述べる。

第 2 章

研究の背景

本章では、NMW System の研究における背景として、最初に従来のネットワーク管理手法について述べ、その問題点を指摘する。そして、その問題点を解決する手法として、ネットワークワームを用いたネットワーク管理を提案する。

2.1 ネットワーク管理

コンピュータネットワーク管理手法としては、実際に管理対象機器にリモートログインして対話形式で操作する方法、SNMP(Simple Network Management Protocol)[6] によって管理情報の取得、設定、操作を行なう方法などがある。

この節では、従来のネットワーク管理システムの例として、SNMP によるネットワーク管理の仕組みについて述べ、従来のネットワーク管理における問題点について述べる。

2.1.1 SNMP によるネットワーク管理

現在利用されているネットワーク管理システムの多くは、SNMP を利用している。SNMP は、管理対象の機器に存在する管理エージェントと、管理者のいるホストで動く管理マネージャが通信をするためのプロトコルである。SNMP を利用した管理システムでは、このプロトコルにより管理エージェントと通信をして、管理情報の取得、設定、操作を行う。

これまで実装されている SNMP を用いたネットワーク管理システムでは、管理者が操作する管理マネージャが中心となり、各管理エージェントを監視する中央集中型 (スター型) が多い。管理マネージャがネットワークや管理対象機器の状態を把握し続けるためには、管理対象の状態を繰り返し調べている必要がある (これをポーリングと呼ぶ)。

しかし、この手法では、管理対象の数が増え範囲が広くなると、管理システムがネットワークへ負荷をかけることになり、その結果ネットワークに障害が生じることになると、管理システムの意味がなくなってしまう。これらの対策として、管理マネージャを多段階に配置し、ネットワークを木構造としてとらえ管理するる手法なども考えられている [19]。しかし、負荷は分散するものの、各枝葉のノードにおいては中央集中型と同じであるため、

根本的な解決にはなっていない。

また、管理者の負担を軽減する目的から、人工知能の導入による管理作業の自動化なども考えられている。しかし、人による管理作業に近付けると管理システム自体が大きくなってしまい処理が重くなってしまうため、実用に耐えないなどの問題がある。

2.1.2 従来のネットワーク管理における問題点

管理対象ホストにログインして管理作業する方法や、前節で述べたような SNMP を利用したネットワーク管理では、コンピュータネットワークの規模が大きくなりその構成が複雑になると、一般に管理者の負担も増大する。したがってネットワークが現在のペースで発展を続けると、管理者の作業量や問題の複雑さなどの理由により、従来の管理手法ではネットワークを扱えなくなる恐れがある (図 2.1)。

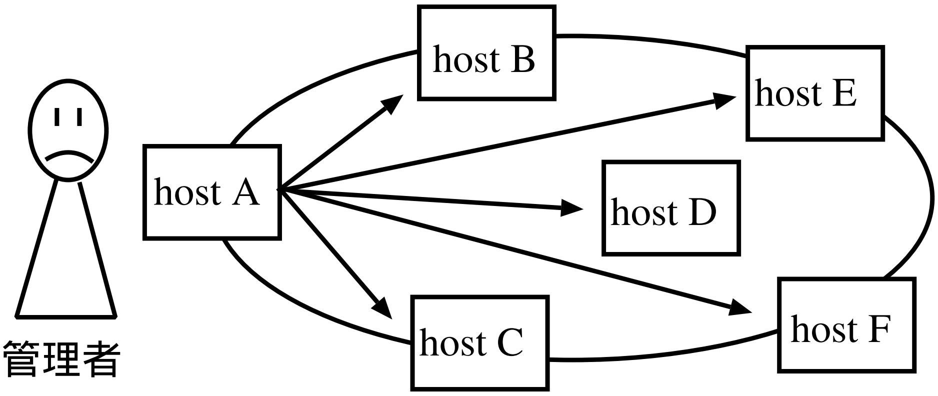

図 2.1 従来のネットワーク管理

また従来の管理手法では、ネットワークに障害が起きた場合、障害が起きたこと自体は検出できるが、その時点におけるネットワーク全体の状況を取得したり、ネットワークを制御したりすることが困難である。これは、ネットワーク管理モデル [16] で説明されているように、従来の手法では、コンピュータネットワークの管理をそれ自身を介して行っているためである。

これらの問題は、従来の管理手法そのものに原因がある。すなわち、管理を行う管理者やそれを援助する「エージェント」が各管理対象ホスト (機器) に固定されており、管理者がある特定のホストからリモートログインしたり、「管理マネージャ」を用いて「エージェント」と通信したりして、ネットワークや管理対象の状態を調べる方法だからである。

2.2 ワームを用いたネットワーク管理

従来の管理手法に対して、管理エージェントが管理対象ホストに固定されずに、ネットワーク上を移動しながら自分自身の判断でネットワークを調べる方法がありうる (図 2.2)。

この移動する管理エージェントにはホスト間を移動するプログラム、すなわちネットワークワーム [7](以下ワーム) を利用できる。

ワームをネットワーク管理に用いると、以下のような利点が生じる。

- 管理者の作業量を削減できる。

  ネットワーク上を移動しながらの調査、分析、制御が可能となるので、複数のホストでの作業を一つのワームにまかせることができる。

  また、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、複数のホストで各ホストの状況に応じた作業ができる。

- 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、作業に伴って生じる通信量が削減できる。また、一つのワームが複数のホストを周ることにより、管理者と管理対象の間の通信回数を削減することができる。

- ネットワークの障害時にも対応できる。

  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以遠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。

以上の検討にもとづき、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、NMW System(Network Management Worm System) の研究・開発を始めた。

2.3 エージェントシステム

近年、プログラムがネットワーク上を移動し各ホストで作業を行うという手法は、エージェントシステムと呼ばれるようなった。本節では、エージェントの概念について述べ、その例を挙げて説明する。

2.3.1 ネットワークエージェント

利用者から仕事を受け取り、利用者に代わってその仕事を行うものをエージェントと呼ぶ。中でもコンピュータ上のソフトウェア環境での作業を支援するものをソフトウェアエージェントと呼ぶ。そして特にネットワーク環境を対象としたものをネットワークエージェントと呼ぶ [13]。

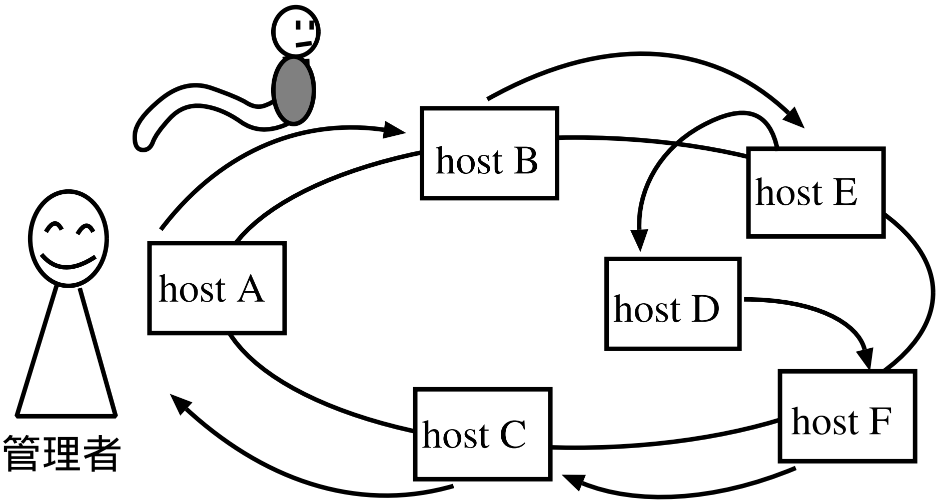

図 2.2 ワームによるネットワーク管理

ここで言うエージェントは、前で述べた SNMP における管理エージェントとは異なる。エージェントが利用者に代わり作業を行う点では同じであるが、SNMP の管理エージェントは各管理機器に固定されており移動することはないのに対し、この節で述べるネットワークエージェントはネットワーク上を移動するものである。

ネットワークエージェントでは、プログラム自体がネットワーク上を移動し、移動先のホストで処理をして、実行結果を元のホストへ返すといったリモートプログラミングと呼ばれる手法が用いられている。この手法は、大型計算機で古くから行われている RJE(Remote Job Entry) [1]と似てはいるが、処理を行うホストが一つでない、ネットワークを対象としているといった点で大きく異なる。

このリモートプログラミングを実現したものとして、Telescript[3][22] や Java[1] といった言語がある。

Telescript では、エージェントと呼ばれるプログラムが、目的のサービスを提供するホストへ移動し、そのホストにいるサービスエージェントと通信して作業を行う。

Java では、ネットワーク上のホストからアプレットと呼ばれるプログラムを取り寄せ、ローカルのホストで実行することができる。また Java を拡張し、リモートプログラミングや分散プログラミングを可能にするための研究が行われている。その例として HORB[2] や、Aglets Workbench[4] が挙げられる。

---

[1] RJE: リモートバッチ処理、遠隔一括処理ともいう。遠隔地の端末装置と中央の計算機システムを通信回線で接続し、遠隔地からジョブを中央の計算機に投入したり、処理結果を端末あるいは中央のに出力するようなデータ処理の形態を言う。

HORB

HORBは、分散環境でネットワークを意識することなく容易なネットワークプログラミングを可能にすることを目指して開発されている。HORBの特徴をまとめると以下のようになる。

- オブジェクト指向によるネットワークコンピューティングを実現
- オブジェクトをサーバー間で受け渡しすることが可能
- Javaの特徴（アーキテクチャ独立、ポータブル、インタオペラブルなど）をそのまま継承
- Javaの動作するマシンで動作可能
- 容易に分散オブジェクトを実現

Aglets Workbench

Aglets Workbenchは、移動エージェントを用いてネットワーク上で共有されたデータを検索、アクセス、管理するアプリケーションを作成するための視覚的環境である。Aglets Workbenchは、Javaで記述されており、そのクラスをAgletクラスと呼ぶ。このAgletクラスを用いることにより、移動エージェントを容易に作成することができる。また、移動エージェントを用いたアプリケーションを開発するための環境として、ビジュアルプログラミングの形でGUIを用意している。

以上のように、ネットワークエージェントやリモートプログラミングの手法を用いたネットワーク利用法の研究が盛んになりつつある[5]。

NMW Systemは、ワームと呼ばれるプログラムがネットワーク上を移動し、各ホストで管理作業を行う。このリモートプログラミング、移動エージェントの一種といえる。NMW Systemは、ネットワーク管理に特化したエージェントシステムである。

第 3 章

NMW System の設計と実装

NMW System は、1992 年度に清水亮博[1]により提案され [20]、研究・開発が始められたネットワーク管理システムである。筆者は 1994 年度より研究に参加し、1995 年度からはその中心となり研究を続けてきた。

本章では、この NMW System の設計と実装について述べる。

3.1 NMW System の目標

本節では、NMW System の目標について述べる。最初に開発方針を述べ、その後で管理者にかかる負担の軽減について述べる。

3.1.1 NMW System の開発方針

NMW System の開発目標は、以下の 3 点である。

- 新しい形態のネットワーク管理機構の提案
- サービス層 [16] の障害にかかわらず動作する管理層の実現
- ワームを実用に供する際の制御技術の確立

ここで言うサービス層、管理層とは、図 3.1のネットワーク管理モデル [16] で示される 2 つの層である。このモデルは、管理情報が流れる管理層 (Management Layer) と通常のサービスを提供するサービス層 (Service Layer) とが同一のネットワークを利用しているため、ネットワークに障害が発生してしまうとネットワークの管理が困難になることを指摘している。

これらの目標を達成するため、NMW System は以下の方針で設計した。

---

[1] 1995 年 3 月東京工業大学大学院情報科学専攻修了、現 NTT ソフトウェア研究所

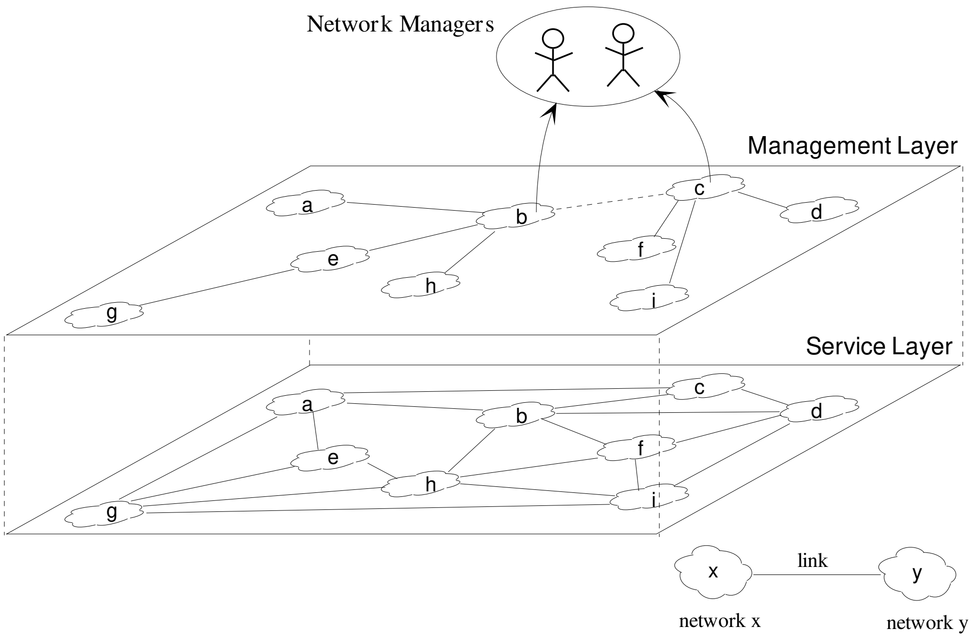

図 3.1 ネットワーク管理モデル (文献 [16] より転載)

1. NMW System のワームは、セキュリティホールを突いて侵入するのではなく、あらかじめ用意されたワーム支援系によって実行される。

   ネットワーク上のワームを実行するホストには、ワームを実行するための支援系が存在しなければならない。

2. ワームの実行はワーム支援系間のメッセージによって制御できる。

   万一、ワームが暴走した場合、ワーム支援系の制御によってワームは強制終了されなければならない。NMW System では、ワームのホスト間の移動回数を制限することなどで暴走に対応している。しかし暴走を完全に検知するのは困難であるので、万一の場合に備えて制御メッセージによってワームの実行を集中的に制御できるようにする。

3. 特定のネットワークに依存しない。

   NMW System は管理層のシステムであるので、必ずしもインターネットのみを用いるとは限らない。したがって、ワームや制御メッセージの転送、計算機の識別子等には特定のネットワークを前提とした方法を用いない。

4. ネットワークに負担をかけない。

   NMW System ではネットワークに障害が発生した場合、管理者にそのことを連絡するが、そのための連絡手段を探す必要がある。その時に大きな通信量を探索に使用す

ることは望ましくなく、さらにその通信量によって新たな障害を引き起こすことになりかねない。したがって通信量が大きくなる方法は使用しない。

5. 通常のサービスが停止していても動作する。

   ネットワーク上でホストやサービス、サーバ等の資源を探索する場合、システムが正常であればDNS等のディレクトリサービスを利用して、必要な資源を探すことができる。しかし、NMW SystemはネットワークK異常が発生した時にも資源探索を行なう必要があるので、通常のディレクトリサービスに依存できない。

3.1.2 管理者にかかる負担の軽減

管理者の立場から見たNMW Systemの特徴として以下の事柄を挙げることができる。

- 管理対象が増大しても容易に管理作業ができる。
- RTT(Round Trip Time)を気にせずに作業が可能。
- 利用者を端末の前に拘束する時間が少ない。

これらの特徴によりネットワーク管理において管理者の負担を軽減することが可能である。

3.2 NMW Systemのしくみ

3.2.1 NMW Systemの支援系

NMW Systemは、ワームとワーム支援系とから構成される。ワームとはプログラムの一種である。現在の実装では、`scm`[2]を使用している。NMW Systemにおいては、ワームはワーム支援系によりホスト間を移動し、作業を行なう。NMW Systemのワーム支援系は以下の3つのサブシステムから構成される。

OWL	ワーム実行インタプリタ (Original Worm descriptionLanguage) ワームを実行するインタプリタ。現在の実装では、`scm`の処理系を拡張して使用している。
wsd	ワーム支援デーモン (Worm Support Daemon) ワームの転送、制御メッセージの配送、結果の回収などのホスト計算機間の通信を行ない、owlを制御するサーバ。また、ワームが利用する情報を保持している。
wormc	ワーム制御インタフェース (Worm Controler) システムを制御するためのユーザインタフェース。管理者が操作するホストにのみ必要。

---

[2] Auberey Jafferによって作成されたschemeの実装。

これらのサブシステムの関係を図 3.2 に示す。ワーム支援系を導入することにより、ワーム実行の制御、実行時のアクセスコントロール、多種類の計算機を管理することなどが可能となる。

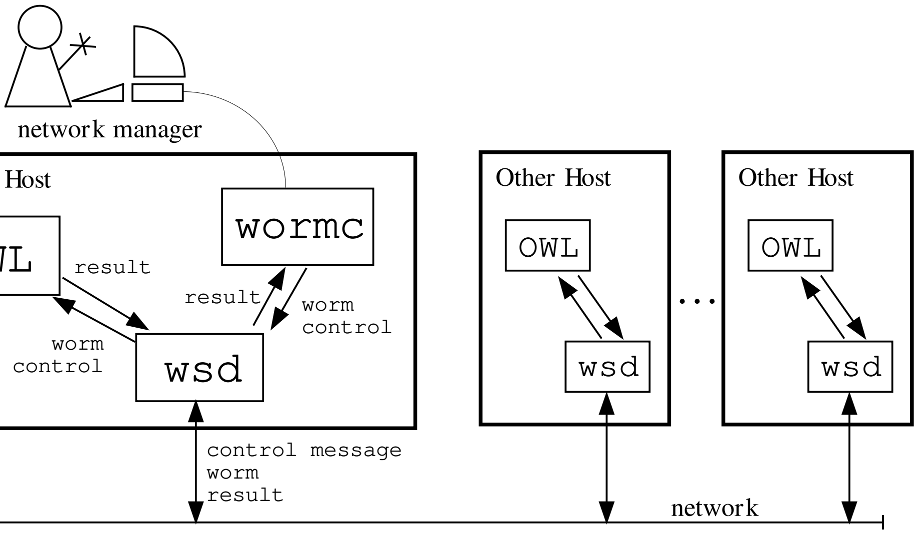

図 3.2 NMW System 支援系

3.2.2　ワームの種類

NMW System では 2 種類のワームを用意している。一つは全てのホストで実行される Flooding Worm で、もう一つが指定されたホストを順に周り実行される Free Worm である。Flooding Worm は、各ホストの状態を調査する場合などに利用するワームである。Free Worm は、一つあるいは複数のホストで作業を行う場合などに利用する。複数のホストを周る場合、その順番を指定することができる。NMW System では、ワームプログラムを「`*.owl`」と表す。実際のワーム (`*.owl`) の例は、第 4 章で説明する。

3.2.3　WIDE/PhoneShell との協調

ネットワーク管理システムにおいて、障害が発生したことを管理者に報告する機能は重要である。しかし、ネットワークを経由して報告する方法では、その通信路に障害が起きた場合に報告が困難となる。NMW System では WIDE/PhoneShell[16] と協調することによりその問題を解決した。

WIDE/PhoneShell は、ネットワークに直接アクセスできない環境にいる管理者が、公衆電話網における資源、例えば電話回線、FAX、ページャ等を利用することにより、ネットワーク管理を実施するための手法、およびそれを実現するための機構である。

NMW System は、障害が起きたことを報告する時にコンピュータネットワークが利用できない場合には、FAX やページャ等を利用して管理者に報告する (図 3.3)。

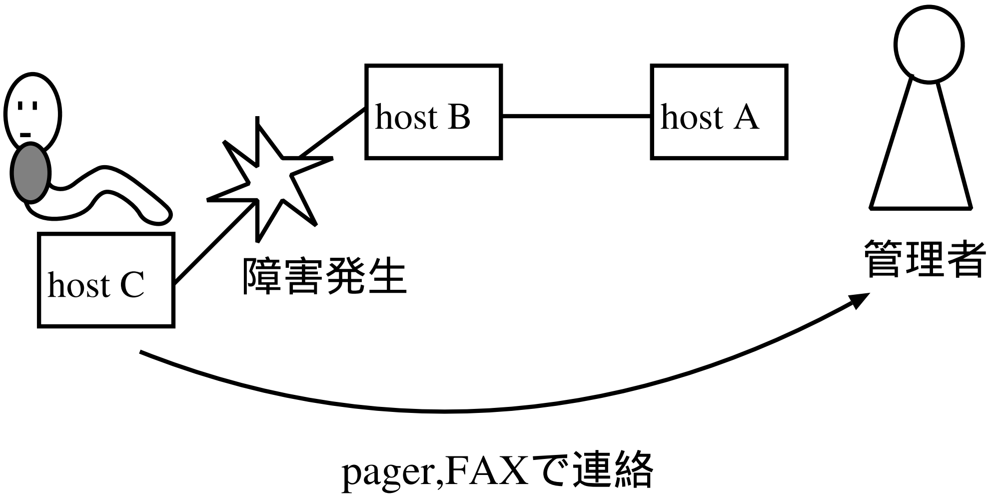

図 3.3 WIDE/PhoneShell との協調

3.3 セキュリティの向上

電子メールなどネットワーク通信を伴うアプリケーションのセキュリティとしては、通信路上での盗聴、通信内容の改竄、他人へのなりすましの 3 つへの対策が特に重要であるとされている。またその他にも、通信内容を記録して繰り返し送り付ける再送攻撃等もある。

NMW System もネットワーク介した通信を行うため、他のネットワーク通信を伴うアプリケーションと同様に、セキュリティを考慮する必要がある。NMW System において特にセキュリティの対象とするのは、次のものである。

- 通信路におけるワームの盗聴
- ワームの改竄
- 他人になりすましての利用

本節では、これらへの対策のため行った、ワームの暗号化、認証、アクセスコントロールについて述べる。

3.3.1 ワームの暗号化

NMW System の暗号化手法には、すでに提案された方法 [21] があるが、実装は行われていなかった。また、この方法では、公開鍵暗号方式を利用しているものの、秘密鍵、公

開鍵の2つとも秘密裏に持つと言う方法を用いているため、秘密鍵暗号方式を用いる方法と理論的に等しくなり、問題があった。

そこで、今回は公開鍵暗号方式を用いた新たな手法を提案し、実装を行った。

新暗号化手法

はじめに、この節で用いる記号を定義する(図3.4)。

W	ワーム本体（平文）
K_x^s	Xの秘密鍵
K_x^p	Xの公開鍵
$\xrightarrow{K_x^y}$	鍵 K_x^y を用いて暗号（復号）化する
$[W]K_x^y$	鍵 K_x^y で暗号化されたW

図3.4 記号の説明

NMW Systemは、次の3つの条件を満たして動作しているとする(図3.5)。

- 管理者はnmwユーザとなりNMW Systemを利用する
- wsdはnmwユーザの権限で実行されている
- wsdはnmwユーザの秘密鍵、公開鍵を利用できる

次にnmwユーザがワームをwsdに送るときの暗号化手順について述べる。

nmwユーザは、ワームを自分の秘密鍵で署名し、さらに自分の公開鍵で暗号化する。

$$W \xrightarrow{K_{usr}^s} [W]K_{usr}^s$$
$$[W]K_{usr}^s \xrightarrow{K_{usr}^p} [[W]K_{usr}^s]K_{usr}^p$$

この暗号化ワーム（$[[W]K_{usr}^s]K_{usr}^p$）は、nmwユーザの秘密鍵を利用できるものだけが作成することができ、復号化することができる。

wsdは、nmwユーザの秘密鍵、公開鍵を利用できるため、復号化し、認証することができる。

$$[[W]K_{usr}^s]K_{usr}^p \xrightarrow{[W]K_{usr}^s} [W]K_{usr}^s$$
$$[W]K_{usr}^s \xrightarrow{[W]K_{usr}^p} W$$

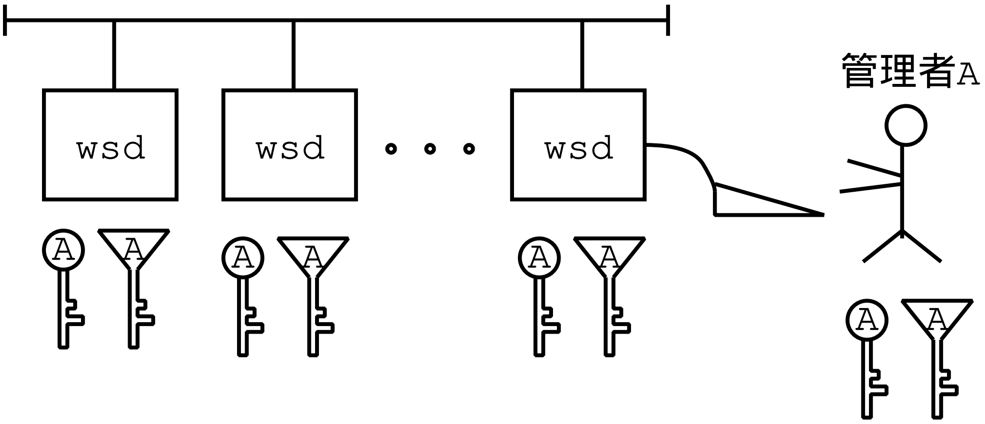

図 3.5 nmw ユーザと wsd の鍵

鍵の取扱

一般に、平文を暗号化、暗号文を復号化するためには、そのための鍵が必要である。公開鍵暗号方式では、通信相手に広く配布する公開鍵と、絶対に他人に知られてはならない秘密鍵とがある。今回は、OS 上の 1 プロセスである wsd が公開鍵暗号方式を使い暗号化、復号化を行う必要がある。そのため、秘密鍵をどのように扱うのか、という問題が生じた。

wsd が秘密鍵を利用するには 2 通りの方法が考えられる。一つは常に秘密鍵を保持して動作しているという方法であり、一つは秘密鍵をどこかに保管しておき、必要な時に取り出して利用するという方法である。しかし後者の場合には、秘密鍵を置くファイルもしくはディレクトリにアクセス制限を行うとしても、秘密鍵をそのまま保管するのは危険であるため、暗号化するなどの方法を用いて安全な状態で保管する必要がある。その結果 wsd は秘密鍵を取り出すための鍵を保持しなければならなくなり、2 つの方法は理論上同じであることになる。

暗号化の実装

今回は 2 つの方法のうち後者を選択し、前で述べた暗号化手法を実装するために、PGP(Pretty Good Privacy)[8] を利用した。そのため、wsd が秘密鍵を取り出すための鍵は PGP のパスフレーズになる。PGP を利用した理由は、以下の通りである。

1. 公開鍵暗号を提供するライブラリ

   公開鍵暗号方式のアルゴリズムとして、RSA 暗号、楕円暗号などいくつか存在する [23] が、それらのうち実用化されているものは少なく、その一つが RSAREF とよばれる実装である。しかし RSAREF は、米国の輸出規制法や特許法などの関係により、日本国内で個人的に使用することはできるが、広く配布することを考えている NMW System に用いるのは難しい。そこで、暗号化の部分を同じ RSAREF を用いたアプ

リケーションである PGP を利用することにした。それにより、PGP を利用できる環境でセキュアな NMW System を利用することができる。

2. ファイルによるメッセージ交換

   NMW System では、主にファイルを介してメッセージを交換する。そのため、暗号化アプリケーションを利用し暗号化を行うことが容易であった。

3.3.2 ワームの認証

暗号化手法の説明で述べたように、NMW System では、nmw ユーザから送られてきたワームを復号化し認証する手法を用いている。しかし、NMW System の運用方針によっては、複数の利用者をそれぞれ認証したい場合がある。

それを実現するため、wsd が nmw ユーザの公開鍵・秘密鍵の他に、NMW System の利用者の公開鍵の利用を可能にした (図 3.6)。利用者は自分の秘密鍵で署名した後、nmw ユーザの公開鍵で暗号化して wsd に送ることにより安全にワームを wsd へ送ることができ、wsd は利用者の公開鍵で認証することができる。

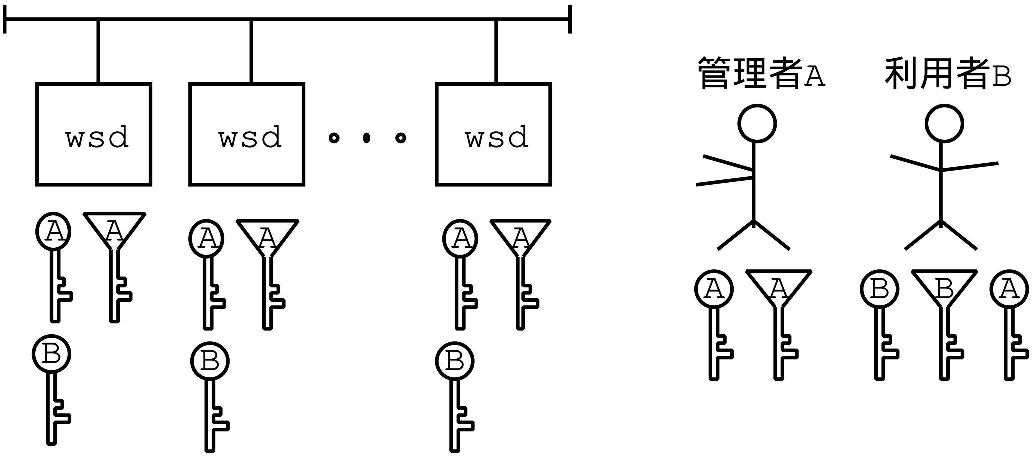

図 3.6 NMW System の利用者と wsd の鍵

3.3.3 アクセスコントロール

NMW System をより柔軟に利用するためには、複数の利用者を認証するだけではなく、利用者を信頼度などによって分類し、それに応じたサービスを提供することが必要である。

NMW System では、登録された利用者の公開鍵を信頼度により区別して保持する。NMW System の管理者は、信頼度により可能な作業を制限することが出来る。この制限は、管理者のポリシーにより自由に設定できる。

実装

PGP には、公開鍵をその信用度によって分類する機能がある。NMW System では、wsd がワームを認証するときに、公開鍵の信用度を調べ、それにより実行できるコマンドを制限することにした。

今回の実装では、wsd が OWL を実行する際、OWL が実行可能なコマンドへのパスを信用度により制限している (図 3.7)。そのパスの示すディレクトリに管理者が適切なコマン (サービス) を置くことによりコントロールする。

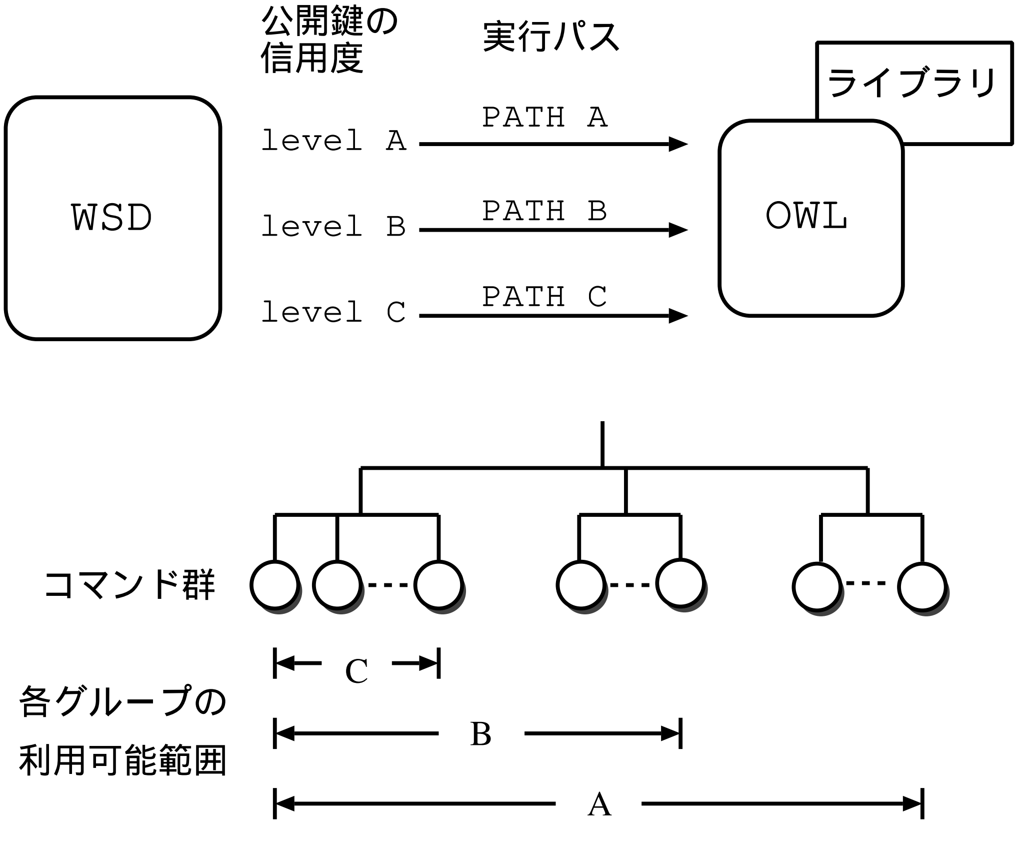

図 3.7 OWL を実行する環境の制限

また、OWL がライブラリ (実際には scm のライブラリ) を読み込む時に信用度により異なるライブラリを用いることができる。

3.4　NMW System の API

他のアプリケーションからワームを実行できるようにするため、NMW System の API(Application Proglaming Interface) を設計した。

3.4.1 ワームの作成

ワームの作成を行うための API を設計した。今回は、C 言語の関数の形で用意した。

ワームの構造

ワームは、ヘッダ部とボディ部からなる。ヘッダ部にはワームタイプ、ポスト時間、発信ホスト、生存時間 (Time To Live) が含まれる (図 3.8)。

```
typedef struct worm_header {
  char          w_type; /* worm type */
  long          w_time; /* posting time (in sec.) */
  u_long        w_addr; /* host addr */
  u_long        w_ttl;  /* TimeToLive */
} WormHeader;
```

図 3.8 ワームヘッダ

このうち、ポスト時間、発信ホストは自動的に付加されるので、ワーム作成時に必要な情報は、ワームタイプと生存時間である。ワームタイプには flood と free がある。flood 型は全ホストにワームを送る場合に用いる。特定のホストに送る場合には free 型となる。

ボディ部にはワーム本体が入る。free 型ワームの目的ホストなどはワーム本体に含まれている。

API の設計

ワームを作成するための関数として以下に挙げるものを作成した。

- `set_ttl(u_long w_time, char *wormname)`

  `TTL` の設定。

- `set_wormtype(char w_type, char *wormname)`

  ワームタイプの設定。

- `set_targethost(char hostname, char *wormname)`

  目的ホストの設定。繰り返しセットすることにより、複数のホストを設定できる。

- `set_worm_body(char *filename, char *wormname)`

  自分で用意したワーム (`filename`) をセットする。

- `set_worm_function(char *function, char *wormname)`

  あらかじめ用意されている関数 (`function`) をセットする。

3.4.2 ワームの実行

```
int sendworm(int wormtype, char *filename)
    wormtype:    ワームの種類
                    SECURE     1;
                    NONSECURE  2;
    filename:    ワーム書かれたファイル名
    返り値:       error  -1;
                  succeed int wormID;

int sendcontrol(int controltype)
    controltype: コントロールメッセージの種類
                    SHUTDOWN        1;
                    SHUTDOWN_LOCAL  2;
                    READ_CONF_FILE  3;
    返り値:        error  -1;
                   succeed 0;
```

図 3.9 ワーム実行 API の一部

図 3.9は、あらかじめ用意しておいたワーム (filename) を wsd に送る関数 sendworm(成功した場合の返り値はワームの識別子 wormID) と、コントロールメッセージを wsd に送る sendcontrol である。またその他に、次に挙げるワームの実行結果を取り扱う関数などを用意した。

- `int countresult(int wormID)`

  一つのワームの実行結果が複数存在する。`wormID` のワームの実行結果の数を返す関数。

- `bool resultup(int wormID)`

  `wormID` のワームの結果が帰ってきたか調べる関数。

- `int resultfilename(int wormID, char *result[] )`

  `wormID` の付く実行結果のファイル名を `result[]` に格納する関数。

- `int readresult(char *resultfilename, char buf)`

  ワームの実行結果（`resultfilename`）を `buf` に読み込む関数。

これらの関数により、他のアプリケーションから NMW System を利用し、wsd にワームを送る、ワームの実行結果を受け取る、wsd を制御を行うことが可能となった。

3.4.3 API の今後

今後は、NMW System をメッセージ交換の通信手段として利用できるよう、インタフェースを用意する予定である。NMW System では非同期な通信を、暗号化により安全に行うことができると予想できる。

第 4 章

NMW System の実験と評価

本章では、NMW System の運用とその評価について述べる。

4.1 実験環境

NMW System は現在、BSD/OS 2.x、NewsOS 4.2.x、SunOS 4.1.x、FreeBSD 2.1.x に置いて動作している。図 4.1に、今回実験を行ったネットワーク環境を示す。

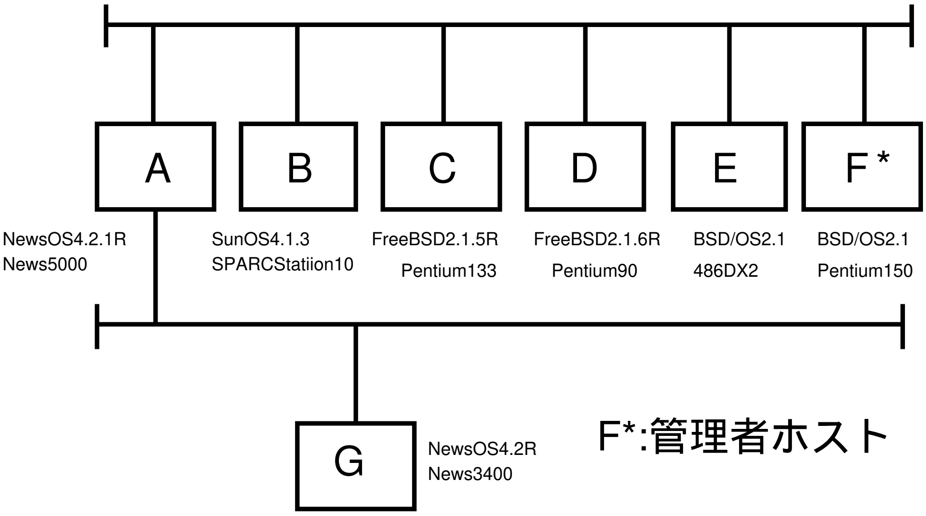

図 4.1 実験ネットワーク

4.2 NMW System のインストール

NMW System は、配布するためパッケージ化されている。そのため、インストールは、makefile の一部を利用環境に合わせて変更し、make するだけである[1]。

4.3 NMW System の実行

この節では、NMW System の実行結果について、暗号化ワームの実行を例に説明する。

wsd の起動

wsd は、オプション無しで実行するとノーマルモード、-sec オプションを付けて実行するとセキュアモードで起動される。セキュアモードで起動した場合、パスフレーズを入力する必要がある。そのため、セキュアモードで起動するためにはパスフレーズを知っている管理者が直接実行する必要がある。

wormc の実行

ワームを暗号化して送信する方法は、wormc を -sworm オプションで起動する。実際にワームを送信した所を図 4.2に示す。

この結果、図 4.3に示すワームが、図 4.4のように暗号化され、wsd に送られる。

ワームの実行

wormc からワームを受け取った wsd は、自分宛でないワームの場合には該当するホストへ転送し、送る相手が無い場合には実行して結果を返す。図 4.5に wormc からワームを受け取り他のホストへ転送し、結果を受け取る wsd のログを、図 4.6にはワームを受け取り、実行して、結果を返す wsd のログを示す。

4.4 ワームの実行例

NMW System によるワームの実行例ををいくつか紹介する。

4.4.1 情報収集ワーム

前節において例として取り上げたワーム (図 4.3) は、各計算機のプロセス状態とディスク使用量を集めるワームである。このワームを例にして、ワームのスクリプトを説明する。

一行目の「`;; flood`」は、ワームを全てのホストで実行する flooding Worm であることを表している。この行が無い場合は、Free Worm である。2 行目以降の「;」はコメント

---

[1] パッケージに含まれているドキュメントに詳しく説明してある

```
nmw@taiko(7)% ./wormc -sworm ps-df-flood.owl

You need a pass phrase to unlock your RSA secret key.
Key for user ID: NMW User <nmw@is.titech.ac.jp>
1024-bit key, key ID 8CF5F955, created 1996/12/06

Enter pass phrase:

Transport armor file: ps-df-flood.owl.asc
filename.asc = ps-df-flood.owl.asc
The ID of posted worm is 1893388338.
nmw@taiko(8)%
```

図 4.2 暗号化ワームの送信

文を表す。7 行目で結果を報告するホストを得ている。8 行目ではプロセスの状態を 9 行目ではディスク使用量を調べ、結果を帰している。図 4.7に結果を示す。

4.4.2 ファイル収集ワーム

特定のホストからファイルを暗号化して転送するワーム (図 4.8) である。

4 行目で取ってくるファイルを設定し、5 行目でファイルのあるホスト、つまりワームの行き先を設定している。11 行目で目的のホストかどうかを調べ、目的のホストなら 12 行目を実行し目的のファイルを暗号化して帰す。目的ホストでない場合には、13 行目で目的ホストに移動する。

このワームの実行結果を図 4.9に示す。

4.5 一般ユーザによる利用

一般ユーザが NMW System を利用するには、2 通りの方法がある。一つは管理者が管理している NMW System を一般ユーザに開放する方法。もう一つは一般ユーザが自ら NMW System を動かす方法である。前者の場合には管理者によって適切に設定されていれば、後者の場合には元のままで一般ユーザの権限でできる作業が可能になる。

今回は、一般ユーザの権限で動く NMW System の実験として個人メールを取ってくる実験を前者の方法を用いて行った（図 4.10）。このように個人情報を扱う場合、その情報の

```
;; flood

;;
;; main
;;
(define (main)
  (send-result (getenv "HOST"))
  (send-process-status)
  (send-filesystem-status))
;;
;;
```

図 4.3 送信するワーム (ps-df-flood.owl)

所有者の権限で動いているプロセスかルート権限で動いているプロセスでなければファイルなどの情報にアクセスできないことがある。

特に設定を変えない限り、NMW System では、管理用の NMW User を作りその権限で動くように設定されている。今回、一般ユーザによる利用の実験においては、ルート権限で実行されるように設定した（図 4.11）。

4.6 実行時間

実行時間に影響するのは、ワームの転送にかかる時間、ワームの暗号化にかかる時間、ワームの実行時間の 3 つである。

4.6.1 暗号化時間

暗号化したワームの実行時間と暗号化しなかったワームの実行時間の差は、ほぼ実行過程における PGP の実行時間に等しい。

PGP では、平文を直接公開鍵暗号方式により暗号化するのではなく、ランダムに作成した秘密鍵暗号方式の鍵で暗号化し、その鍵を公開鍵で暗号化して、暗号文と暗号化した鍵とを一緒に送る方法を用いている。これは、平文を公開鍵暗号方式で暗号化するのにはかなり時間がかかるためにとられた方法である。また、PGP は平文を圧縮してから暗号化している。これは、暗号解読攻撃への安全性を高め、送信時間の短縮にもつながる。

図 4.1の環境において、管理者が図 4.3のワームを送信してから全ての結果が帰ってくるまでにかかった時間は暗号化した場合には 10 秒、しない場合には 6 秒であった。この結果

```
nmw@taiko(17)% cat ps-df-flood.owl.asc
-----BEGIN PGP MESSAGE-----
Version: 2.6.3ia

hIwD7XEP74z1+VUBA/kB7iXBd+pnFpN82ZG6PAp4a7uD3PaIj8H+GbynMsO3g92u
McEPv11tV0rX6FRiSv1HmT8lZyl17WT5chZjO4dZVTWOfbfIiaFvKn28XThVRHy0
+ih3Zc90LwwSW1y3JtV13RqNRUYjWyzsr679np7boRCh0fKZg8CTUB0VZFkFbKYA
AAFjMpYRPOo6F5zm10cAfPZO3qRHiobE4coihKgpgSYnLeoHJHvnP4aVloYrwNqo
ymz57keaE/CY2ANnroqIbbnA9jPnD/QxJ0dlMfyr8goCLbTZO+yhVyvkMo1Y6zA/
1ysDMNfsFk7MBpHa1irAMOxxV8jQAt1WkA4fb1UagyHd9OJjC3MbCgxJuZi8sk6v
wHWl8JbXcPnRBdxBnDyxTDm5E2pfvtsGMBVxz12nHjc0MRR5Eo238UlXAC3hmYDN
85GRq4/TCl6hHDUMZC1UskLMbtCL3+Utzs65Ze+MaLg3vXoV0gv01iyP5+Wy50Qx
+ULTToQ4dTn7HQCZflQgjqEzuGKLWWb3fSy5sEDgQVywwcl0PKhl0nR/XBnYAgdR
TFVfR8cqr6oEhg+p39O/gVBZ/cIaXhjee2SOuIgUUori8lNQyjlK60tIOzQgVRe2
Z0I6jbw3IDerP9nYwYNbVpxplE4Lng==
=KYo7
-----END PGP MESSAGE-----
nmw@taiko(18)%
```

図 4.4 暗号化されたワーム

ではおよそ倍の時間がかかったことになる。

ワームが大きくなるとそれに比例して時間がかかることが予想できる。しかし、実行時間は倍となる程度なので、どうしても実行時間を速くしたい場合を除けば、暗号化しても影響は少ないと考えられる。

4.6.2 ワームの実行時間

NMW System では、OWL として scm を利用している。scm は、scheme の実装の一つであるが、scheme はその他の lisp 処理系に比べシンプルに設計されているため、軽いことで知られている。また、その他のスクリプト言語に比べても実行速度は速い。

図 4.3のワームが各ホストで実行されるのにかかる時間は 1 ~ 4 秒であった。数値に幅があるのは各ホストの CPU の処理能力に差があるためである。

4.7 実験評価

セキュリティを向上したNMW Systemによって、安全にワームを送り、暗号化された結果を受け取ることが可能になった。そのため、安心して管理作業に利用することが可能になった。

```
:0:[Post Worm]:f:1893388338:131.112.57.2:taiko
> Type=f
> WormID=1893388338
> hostname=taiko
> hostaddr=131.112.57.2
> at Mon Jan  6 19:56:48 1997

:3:[Send Worm]:f:1893388338:131.112.57.2:taiko:nsk.is.titech.ac.jp
> Type=f
> WormID=1893388338
> wormhostaddr=131.112.57.2
> fromhost:taiko --> tohost:nsk.is.titech.ac.jp
> at Mon Jan  6 19:56:48 1997

:3:[Send Worm]:f:1893388338:131.112.57.2:taiko:piano.is.titech.ac.jp
> Type=f
> WormID=1893388338
> wormhostaddr=131.112.57.2
> fromhost:taiko --> tohost:piano.is.titech.ac.jp
> at Mon Jan  6 19:56:48 1997

～～略～～

:5:[Receive result]:f:1893388338:131.112.57.2:taiko:piano.is.titech.ac.jp
> Type=f
> WormID=1893388338
> wormhostaddr=131.112.57.2
> fromhost:piano.is.titech.ac.jp --> tohost:taiko
> at Mon Jan  6 19:56:55 1997

～～略～～
```

図 4.5 ワームを送信したホストのログ

```
:1:[Receive Worm]:f:1893388338:131.112.57.2:piano:taiko.is.titech.ac.jp
>       Type=f
>       WormID=1893388338
>       wormhostaddr=131.112.57.2
>       fromhost:taiko.is.titech.ac.jp --> tohost:piano
>       at Mon Jan  6 19:56:42 1997

Good signature from user "NMW User <nmw@is.titech.ac.jp>".
Signature made 1997/01/06 10:57 GMT using 1024-bit key, key ID 8CF5F955

～～略～～

:2:[Exec Worm]:f:1893388338:131.112.57.2:taiko.is.titech.ac.jp
>       Type=f
>       WormID=1893388338
>       hostname=taiko.is.titech.ac.jp
>       hostaddr=131.112.57.2
>       at Mon Jan  6 19:56:47 1997
:4:[Send result]:f:1893388338:131.112.57.2:piano:taiko.is.titech.ac.jp
>       Type=f
>       WormID=1893388338
>       wormhostaddr=131.112.57.2
>       fromhost:piano --> tohost:taiko.is.titech.ac.jp
>       at Mon Jan  6 19:56:49 1997
```

図 4.6 ワームを受け取ったホストのログ

```
piano
USER       PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED       TIME COMMAND
nmw       5352  3.0  1.6   656  484  ??  S      6:35PM    0:00.22 scm
nmw       5350  1.4  0.5   476  144  ??  S      6:35PM    0:00.04 ./wsd -sec
root         2  0.0  0.1     0   12  ??  DL   25Dec96    0:10.29 (pagedaemon)
root        12  0.0  0.1    68   44  ??  IWs  25Dec96    0:00.01 asyncd 2
root        14  0.0  0.1    68   44  ??  IWs  25Dec96    0:00.01 asyncd 2
root        20  0.0  0.4   748  112  ??  IWs  25Dec96    0:00.91 mfs -o rw -s 3
root        46  0.0  0.6   128  172  ??  Ss   25Dec96    0:10.19 syslogd
root        49  0.0  0.4   104  128  ??  IWs  25Dec96    0:00.06 portmap
root        89  0.0  0.2    72   60  ??  Ss   25Dec96    0:45.17 update
root        91  0.0  0.6   284  172  ??  Ss   25Dec96    0:13.74 cron
root        98  0.0  0.1    72   32  ??  IW   25Dec96    0:00.10 nfsiod 4
root        99  0.0  0.1    72   32  ??  S    25Dec96    0:11.59 nfsiod 4
root       100  0.0  0.1    72   32  ??  IW   25Dec96    0:00.98 nfsiod 4
root       101  0.0  0.1    72   32  ??  IW   25Dec96    0:00.14 nfsiod 4
root       103  0.0  0.2   112   72  co- IW   25Dec96    0:00.04 rstatd
root       106  0.0  0.3   128  108  ??  IWs  25Dec96    0:00.09 lpd
root       115  0.0  0.7   424  204  ??  Is   25Dec96    0:04.17 (sendmail)
root       121  0.0  1.1   204  348  ??  S<   25Dec96    0:55.21 /usr/sbin/amd
root       130  0.0  0.4   152  136  ??  IWs  25Dec96    0:00.95 inetd
nmw        399  0.0  0.1   472   36  p1- S    25Dec96    0:00.03 ./wsd -sec
root     29726  0.0  0.1    96   24  co  IWs+ Sat10PM    0:00.14 -xdmshell (xdm
root     29728  0.0  0.7   224  212  co  IW+  Sat10PM    0:00.10 /usr/X11/bin/x
root     29731  1.0 35.9  9824 11112  ??  S    Sat10PM    2:33.19 /usr/X11R6/lib
root     29741  0.0  1.4   312  440  co  IW   Sat10PM    0:00.14 -:0
root     29750  0.0  1.7   196  524  co  IW   Sat10PM    0:00.07 (xconsole)
root      4333  0.0  5.0  2360 1540  co  IW     2:26PM    0:00.65 (kterm)
onogi     4350  0.0  2.3   684  692  co  I      2:26PM    0:06.40 twm
onogi     4351  0.0  1.8   244  556  co  S      2:26PM    0:02.44 (xload)
onogi     4357  0.0  1.8   204  556  co  S      2:26PM    0:00.61 (dclock)
onogi     4373  0.0 16.1  5836 4980  co  S      2:26PM    2:32.88 mule-2.3.sj3 -
root      4375  0.0  9.5  4500 2940  co  I      2:26PM    0:06.91 (kterm)
onogi     4382  0.0  2.8   716  872  p1  Is+    2:26PM    0:03.45 -csh (tcsh)
onogi     4383  0.0  2.1   532  640  p2  IWs+   2:26PM    0:00.42 -tcsh (tcsh)
nmw       5353  0.0  0.6   212  176  ??  S      6:35PM    0:00.04 sh -c ps uax
root         0  0.0  0.0     0    0  ??  DLs  25Dec96    0:03.35 (swapper)
nmw       5354  0.0  0.8   160  232  ??  R      6:35PM    0:00.00 ps -uax
root         1  0.0  0.5   132  152  ??  IWs  25Dec96    0:00.39 /sbin/init --
Filesystem                     1K-blocks     Used    Avail Capacity  Mounted on
/dev/wd0a                          15743     6363     8592    43%    /
/dev/wd0h                         169220   150876     9883    94%    /usr
mfs:20                             15743        4    14951     0%    /tmp
amd:121                                0        0        0   100%    /home
amd:121                                0        0        0   100%    /auto
taiko:/export/local.i386-bsdos    934315   658412   229187    74%    /am/taiko/export/local.
i386-bsdos
nsk:/export/home.nsk             1961710  1474198   291340    83%    /am/nsk/export/home.nsk
/dev/sr0c                         166432   166432        0   100%    /cdrom
```

図 4.7 情報収集ワームの結果

```
;;
;; constant value
;;
(define targetfile "/var/log/httpd/access_log")
(define targethost "taiko")

;;
;; main
;;
(define (main)
  (if (= (hostname) targethost)
      (send-crypt-file targetfile)
      (go-to-host targethost)))
;;
;;
```

図 4.8 ファイル収集ワーム

```
taiko% cat ./result/taiko.97.1.13.22.55.58.-331425230
---BEGIN PGP MESSAGE-----
Version: 2.6.3ia

hIwD7XEP74z1+VUBBACRtyNqc854Rpo6iJtvVP/ejsW9FDIt/K97RB9oByNLacLM
h1pPoNwXvhPDukHzBWVF22ghWoBZndbk4jcK3SoWHdHaS0dorZjLlnhU8kY3c7Ho
h/yDw5Z9DA8f1CyzBCbJihXpfmmR9nXqhwqWMT+Ty0r+2HvQ6WhsoIeqg1IAfaYA
AAicYO5wLv59GnS6IlzmaV7304LUeBqXKGr72lxJbuyRUEG4lE4HEO8MTP0ZusSD
jDpQB1KvpN60f6+XTGmcqPQEar0OseWdViMTsIQt7ZYLWq0nF1t+mycR8Rn4W2ze
n53U1veCYlPhV//5CCuqt1ZFYm1bVuXC9BLiCw+L7cETnCbp90y3hT8Muvkckmf0
IO9r05vUpnMfo0v2pDBaiu+JOu0Og0zBYVm0JQGOx68D7SKu5muh61IiFwmgslcN
5IrEt1Cc+5aHwPKQOhjmSCvCjFbK2DuuHTnCf2sEAJSnT2WugMC7nphiGmrHe2TG
q/I3WPSI38FD7elpxaT5tWuDDfHOmb8NGw66XeqIAxm+3mndj5AAghLsCHH1idqx
gxqlV63YJVYmK8oOMKk2Hae3AHDHnfSynQLJ9lpRlxINwaH5Id3IuxiE8Mf4RhOV
p7uTrXtGar+GDHSeeT2OPhEKPo8S4n+aQSbVsFoTxuokHheceTw5iYXeSMFrJmwI
KPLO3aNBz2ZjyKOnvOvxhXmDMfVFh2HNNphVKpYxm8yioCGtH3UDM4tqBvLYi3sZ
gqf0xsLAaOxmlHj/9b/1rZUlLTe7PHZuio5KkRh+WxDeFmZj8vbg8JizoKMQo2pR
euv8k8eRtq3CY+Ap6OD87DYCk7D1m1YNRXRXwb5YZLkZZAg01SKitIp5kLEoIX6d
JqCbOMXUI/NGJ2iujpkF4T7ycEadZZoUZAFXKEp0Cnk1WquhYVkeqeTbuGg+UIYc
YFupSOpOWIPmLRHea9eOB8AnAPpHfh+sJJSPsTVNPjX1qqyIMHCC/zKGoHQhGwI8
8ycdAt9ZLBbWbJUPLRZR9OM7Y24ui2BI4ZPsVXO3ZolRVmj+SuIrpH/vzTHXlk+P
fkg4chK1aVE7xKJSF7siB3cz6VKL7D3Y2MQwc5OktoRb8F7gmm/VUlWtnZCwQmZU
uyKSypHWY+idc/IkegvrjdKQhrIOMJ86S1dpYSC+HgBULdArYJjtzmD+tg28EcrF
qzY9zDMamrvB6ORNLKtUlBWXQEYs6xLESsc4uKD+AAMju5xhxGvBsjQRv1JdeU8D
Ht6XuOYoOYJbR22U8EXSZtRR/ik0vrMDlWTy4HY131AG9wOQWZJLOT5Mwg9XEa6e
mo82AVrqr1L7x7GIFApamEQkipZtoBCRkyDLFpmSGA9n03eUEb5itmn6OxrS3QiI
ulKDuL5IvUIdhmdKQoTuw1LJMZmfiCPc+PvbX1TCCry6N4p+TEU5BBCmynR8QbKd
cVH9OQRCMHIfvdbhhqIt4HO/q2wTbtSiu2RNPbsKPkTo9frEEREZ/ckpWMctaPgM
VVKcn08/4a5p0iBwp9oLFgBtqtKIZsZXV6uZrWDVJmR+xzTgaa7+snc/JWML3eLz
Jue2DL1IEwm6KGvrIqFOBC+Mi7WCZcMqpdxMJrKoXcHAIsTpugc34WQPHnNSSy/t
zQET/r29O8nTTAdLP72kOFSqpL6GHQJrVPymKh6jEikEbhN9hxA0WgPGkOB2OJ5C
AtgRteLHLpZN6FyOKB3HN6OF2inXnWjOANp1Wd137cc86XgVGQKlhlGwH4bfp+A5
HZgGBq2OeDBp2n7O6eici7kvQJOiSheyOQBQ2yfTNOpZLy5e9xnpbEuKMrjgFuEI
UHubl6bdF5KcC+Q7nosSBGVPHSbJqILiBT2Lz3kzDe4tSr1YGicIIGIGPZgxhQAD
RMq3ajergPlOztiaASiFNti4uVd1mRiXSjzJf1A7WotVNWVDTy709vZHmQiJIOi3
WV+OClneG3wlj+EWsiQOJVLCuWgsGC+PpFndPXlHtbpxD3uqLkC6wKwA+C/ffCq4
73NijCEBHv5V6BADwGSMuOwzav9eLfuWBZGsiWBahLBUh32/93PI37iJrH/Tg1XT
lU4EutIPxn1qED6fTAy7sST+R1aXzWIlrvCd6KqD3oooQsRCR6MT6JdxGRfr4MQr
g2gp2b/DBNbLcNiqTCfg6P/sNnCmPQOsgH7tUVzhdyxhcwjT7ggtg9ewTfPZ8KmG
j/Sg+jqxFFigOKmeZE8QA7dlCYkrhU7YqOs5jTa8zRUS/jfzigu3WUPrLsHlmY2B
OA/VTHyfGvsDJcI/9hfbz15WKlYsQ7/H/3qJMkkW64HV8kuPhWLWTv2Y9xptroDS
+iNRyRqVib1fR2B7Sz87LlSifvL42OYj74dHUbuph+yEJ/gM5auJCcDEAOQlkRe4
ECRrrijY4brSeJtzWkdqJh55gOxi97LbxdN4NI8lA7tO3Vfq5AlLZtornGmNCwEg
bSBqHOnTCHongb+4uMjc5/xplTRLByJXFAyNrPX5kEOFwH9Sk9ZzPtxdylgHXAd7
l/TbdXWnctRphVbmxaA5eREIzn1YNBg6gf6WiZq7l04JqheOtJhwVN1znQxRrFCI
41hJ2Ka/hii2JIIW3F6NskzTl+mVpB+v7uOvtOuBYBmghs23X4kesgK3zuPnnm6b
HkKTnXNw5qCnePOWt2VRdcn08yEm6WRkS7noB6ng+RYDcLhJ/mfCMBG9xDJJA3IW
VuDKZO3j933v+vUl66vjdflQh92ZE3OPis/Xh8S2R3FcIuX5HayZ+evN3gNYAHFT
+hNJbFZBF95+rAOrOEdo3URtoZuBur8hOLBWdy4IpTpiKMVRsANUkNoaBvDnBqLP
vtsfkv+SEqlFTo5qxdCLAd9on5D8gflUUK1F89EXmIWS5pouH6OWZG/KHXUOrUiy
6FUaLoTsdWL+OiIiqPDw/zzcJUoWco/BJkROG/nOXCHPK+Gsjji/cTelavZ43OOP
vfyXhNbdsGNKr4NXOkUN5NPucD9THWsMk2OqqTOHD66myCpa3KKxmQ28sR3F/4Vq
BDa5sjg5P1kOLn1AIUSkFcLpKOohxx1GlPsOI4Pbf/j/C4NkJVodEPA5j18pg+g=
=1kw7
-----END PGP MESSAGE-----
```

図 4.9 暗号化され転送されたファイル

```
;;
;; constant value
;;
(define targetfile "/var/mail/onogi")
(define targethost "nsb")

;;
;; main
;;
(define (main)
  (if (= (hostname) targethost)
      (send-crypt-file targetfile)
      (go-to-host targethost)))
;;
;;
```

図 4.10 メール収集ワーム

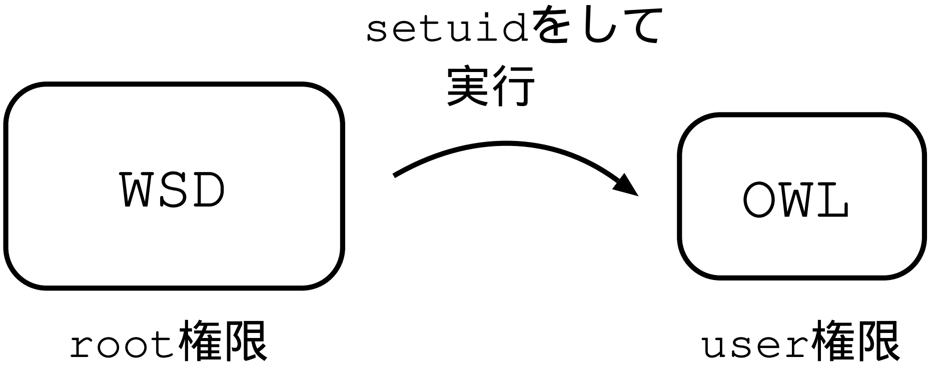

図 4.11 wsd による owl の実行

第 5 章

NMW System の利用例とその評価

この章では、NMW System の利用例を挙げ、その評価および考察を行う。

5.1　公衆情報端末環境における利用

現在大野研究室において、公衆情報端末計画 (PICKLES プロジェクト) [11] と学内情報システム (CITRUS システム) [9] の研究・開発が木本[1]を中心として進められている [12]。PICKLES プロジェクトは、あらゆる場所に公衆情報端末が設置され、あらゆる人がインターネットを利用する環境の構築を目指すものであり、CITRUS は、公衆情報端末と携帯端末により、大学内の情報をサービスをより快適に利用できる環境を目指すものである

本節では、これらのプロジェクトへ NMW System を応用することを述べる。

5.1.1　公衆情報端末の管理

公衆情報端末計画ではたくさんのコンピュータが広く点在することになるので、それらのマシンが設置されている場所へ行き管理作業を行うことになると管理者に大きな負担がかかる。また、ネットワークを利用し全てのマシンにログインして作業を行ったとしても、管理対象のコンピュータの数が多いので負担が大きいことにかわりはない。

しかし、事故が起きたとき等の特殊な状況を除けば、各マシンにおける管理作業の内容は、各種ログファイルの収集やシステムの状況の把握等のように、いずれのマシンでも同じであることが予想される。そこで、あらかじめ用意した管理作業内容を、各マシンを周りながら実行するワームとして用意し管理を行うことで、作業量を軽減することが可能である [18]。

---

[1] 東京工業大学情報理工学研究科数理計算科学 専攻

5.1.2 暗号化ダンプ

学内情報システムでは情報の提供者側にも端末が必要である。そのため、実際に図書館や事務室に公衆情報端末や情報提供用端末を設置し試験運用を始めている [10]。

このような端末では、実際に利用してデータ等を蓄積していく情報提供者と、その端末の管理者とが異なることがある。この場合、管理作業にはディスクのバックアップが含まれるが、利用者側としては管理者にその内容を知られたくない。そこで、暗号化ダンプ[2]を導入する方針である (図 5.1)。暗号化ダンプを NMW System が行うには、2 通りの方法がある。

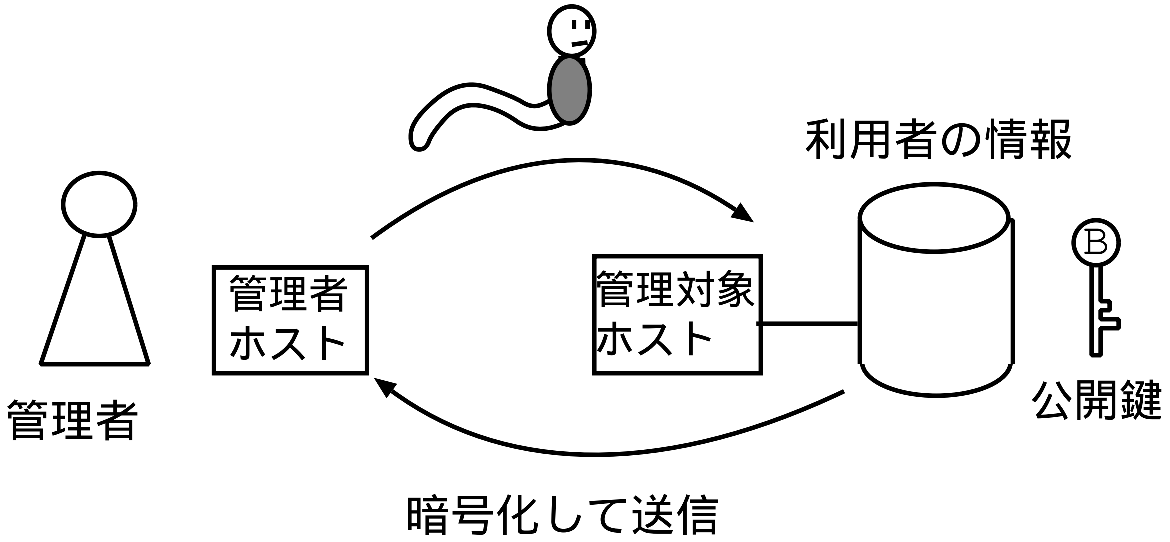

図 5.1 NMW System を用いた暗号化ダンプ

1 つは、まずワームが情報提供用端末に行き情報提供者にディスクのダンプを行うことを報告する。その時、情報提供者は自分の持っている暗号化用の鍵 (秘密鍵方式) の利用を許可する。ワームはその鍵で暗号化したダンプファイルを持ち帰る。この方法では、情報提供者は必ず事前にバックアップを行う事を知ることになる。

2 つ目は、ワームが情報提供者の公開鍵で暗号化する方法である。この場合、情報提供者への連絡はバックアップを行う前後のどちらでも可能である。

いずれの方法によっても、情報提供者は端末の管理者に自分の持っている情報の内容を知られることなく、バックアップをすることが可能になる。

---

[2] 暗号化ダンプは PICKLES プロジェクトの中で考案された

5.2 NMW System の応用

コンピュータネットワークを利用し、遠隔地と通信する場合には，途中のネットワークが混雑していたり回線が細かったりすると満足な応答時間が得られないことがある。そのような状況下では、通常の対話的に行う作業を快適に行うことは難しい。近ごろではネットワークプロバイダを利用し、電話回線を用いて自宅からインターネットへ接続するケースが増えてくるなど、その傾向がさらに顕著になってきている。

そこで、作業の内容をローカル側で作成し、それを作業を行いたい離れたホストへ送り実行をさせるという方式を検討した。この方式では、ネットワークの状況に影響されずに作業を行うことができる。さらに、このサービスにエージェントの概念を用いると、複数のホストを巡っての作業を行うことも可能となる。

そこで、NMW System をエージェントシステムの実装としてとらえ、一般ユーザが利用できるサービスを提供し、通信速度を気にせずに作業できるようなシステムの設計、実装を行った [17][18]。

5.2.1 NMW System を応用したサービス

ネットワークを介したサービスとして、telnet, rlogin 等がある。前で述べたようにこれらのサービスは、ネットワークを介して対話的に作業を行うため、満足な通信速度が得られない環境では快適に作業することが困難になる。そこで、以下のようなサービスを考える。

- ユーザが作業内容をローカル側で作成（命令書）
- 作成した命令書をローカルホストのデーモンに送る
- デーモンは命令書を解釈・実行し、必要があれば他のホストのデーモンへ送る
- 命令書に書かれている作業が終了するまでホストを周りつつ作業を行う
- 作業の終了を知らせ、結果を帰すか、ユーザからの指示を待つ

このように一つのリモートホストへローカルの端末から命令を送り実行させる方法として、昔から行われている RJE (remote job entry) と呼ばれる処理がある。しかし、ここで述べたサービスでは、NMW System を応用することで、RJE では考えられていなかったネットワーク上の複数のホストを巡回しながら作業を続けることを可能にした。また、WIDE/PhoneShell を利用することで、作業の終了したときや作業の途中で事故が起こったとき等に、ページャ等によってユーザに連絡を取ることができる。そのため、ユーザは、作業を行っている間に端末に向かっている必要はなく、回線を接続し続けておく必要もない。終了の連絡が来てから再び回線をつないで結果を受け取ることが可能である。

5.2.2　サービスの実装

このサービスでは、リモートホストの計算機資源を telnet 等と同様に利用できるため、パスワードに代わる認証方法が必要である。そこで、ユーザが命令書に電子署名を行うことで認証を行う。

また、作業結果はユーザの公開鍵により暗号化しユーザに送り返すことで、メールなど秘密裏に扱いたいデータを扱うことも可能となる。

5.2.3　サービスの評価・考察

通信速度が遅い環境として、電話回線を利用し、データ転送速度を遅く設定した環境で実験した。このような環境で通常の方法で対話的な作業を行おうとすると、応答時間が大きく、作業を行うことは困難であったが、今回提案したシステムを用いると、ネットワークを介した対話的な作業はほとんど行う必要がないので、目的の作業をきちんと行うことができた。

ユーザが各ホストで行う作業として、telnet 等で行える命令と同じだけのものを用意することは可能であるが、実際に作業に必要な命令を整理し制限していくことがセキュリティ上重要である。

第 6 章

考察

6.1 セキュリティ

本節では、今回実装を行ったセキュリティ向上手法に関する考察を行う。

6.1.1 安全性についての議論

今回提案した暗号化手法は、公開鍵暗号方式を利用するアプリケーションと同様、公開鍵を公開するものであるので、秘密鍵、公開鍵の両方を秘密裏に持つという、以前に提案された手法 [21] よりも安全になったと言える。
今回の実装では PGP を利用したが、PGP ではプロセス (アプリケーション) による利用は考えられていない。NMW System では、管理者が利用するシステムであるので、管理者は信頼できるという仮定の元に設計されている。そのため、以下に述べる PGP の主張する特長を放棄している。

PGP は、パスフレーズは利用者の頭の中のみに存在する、と仮定している。そのため、たとえ計算機の管理者 (UNIX ではルート) が利用者の暗号文を覗こうとした場合でも、暗号文を解読することは難しい。

6.1.2 管理者のアカウントが盗まれた場合

前節で述べたように NMW System のセキュリティ向上手法では管理者が信頼できるという仮定の元に設計されている。この場合の信頼できるというのは、一般ユーザにとって正規の管理者が悪いことをしないと言う意味である。管理者のアカウントが盗まれた場合の安全性は考えられている。
悪意のある第三者が管理者のアカウントを盗んだ場合でも、NMW System を利用することはできない。それは、アカウントを盗み管理者になりすましたとしても、パスフレーズが判らないと次の 2 つをすることが不可能だからである。

- 新しいwsdをセキュアモードで実行する
- ワームに署名する

このため、あるホストの管理者のアカウントを盗まれた場合、NMW Systemを用いて他のホストを攻撃することは困難である。

6.2 他のエージェントシステムとの比較

本節では、NMW Systemと他のエージェントシステムとの比較をし、考察する。

6.2.1 ネットワーク管理

NMW Systemは、ネットワーク管理システムである。そのため、管理者の作業は、各ホストで行う作業を決めることである。

それに対し、Telescript、HORB、Aglets Workbenchなどは、ネットワークエージェントを利用するサービスを作成するためのプログラミング言語、またはプログラミング環境である。そのため、エージェントを扱う枠組みを用意しているが、利用者はそれらを用いて自分の要求するサービスやアプリケーションを記述することになる。これらを用いてネットワーク管理を行う場合、当然のことながら、管理者は管理システムを作成する必要がある。管理作業の内容について考えるのはその後となる。

また、これらのシステムを使ってネットワーク管理アプリケーションを作成する場合、NMW Systemの構成をそのまま応用することができる。つまり、NMW Systemは、エージェントシステムを用いたネットワーク管理の指針となる。

6.2.2 作業内容の制限

TelescriptやJava(HORB、Aglets Workbench)では、一般に広いネットワーク上でのサービスを前提としているため、実行されるホストの安全を考え、エージェントプログラムのできる作業が制限されている。

一方、NMW Systemでは、ネットワーク管理が目的であるため、管理者の行いたい作業が自由にできなければならない。そこで、ワームの移動を管理対象のネットワーク内のホストに限定し、そのかわりに当該ホスト上では自由に管理作業を行えるようにした。

このためNMW Systemでは、ワームを実行するユーザ(管理者)の認証が特に重要になる。

また、NMW Systemでは、OWLに適当な制限を加えられる。そのため、同一のリクエストに対して、各ホストが各々のポリシーの下に異なる回答をすることもできる。例えば、ワームが各ホストにログインしているユーザを調べようとしたとき、ホストによって、UNIXのコマンドのwhoの結果をそのまま返したり、人数のみを返したり、回答を拒否したりできる。

このようにワームの作業を制限することもできるため、広域ネットワークに置けるネットワークサービスを提供するエージェントシステムとしての応用も可能である。

NMW System と Telescript、Aglets Workbench との比較を図 6.1 にまとめる。この表から NMW System は、管理作業に適している、自由度が高いため応用性がある、などが読み取れる。

	NMW System	Telescript	Aglets Workbench
管理作業	◎	△	△
セキュリティ	○	○	○
他のメディアとの連携 (pager,FAX 等)	○	○	△
リモートホストに送って実行する	○	○	○
ローカルホストに取り寄せて実行する	○	○	○
複数のホストを回って実行する	○	○	×
機能の拡張 ･制限	○	×	○
一般への普及度	×	△	△

図 6.1 リモートプログラミング言語との比較

6.3 他のシステムとの協調

NMW System は、OWL やワームを自由に作成できることにより柔軟な作業を可能とした。しかし、複雑な作業を行うためにはワームの作成や OWL の設定などが難しくなってしまう。

そこで、他のシステムを利用することによって、NMW System をより使いやすくすることが考えられる。以下、magP、stetho との協調ついて述べる (図 6.2)。

6.3.1 システム管理支援機構 magP

magP は、中嶋[1]により研究されているシステム管理支援機構である [14]。magP は、過去の管理作業履歴を蓄え知識として利用する、情報の収集を自動的に行い管理者に報告する、と言った方法で管理者の負担を軽減することを目的として開発されている。現在では、複数のホストからの情報収集なども可能となっている。この節では magP と NMW System を比較した上で、magP との協調について考察する。

---

[1] 東京工業大学情報理工学研究科数理計算科学専攻

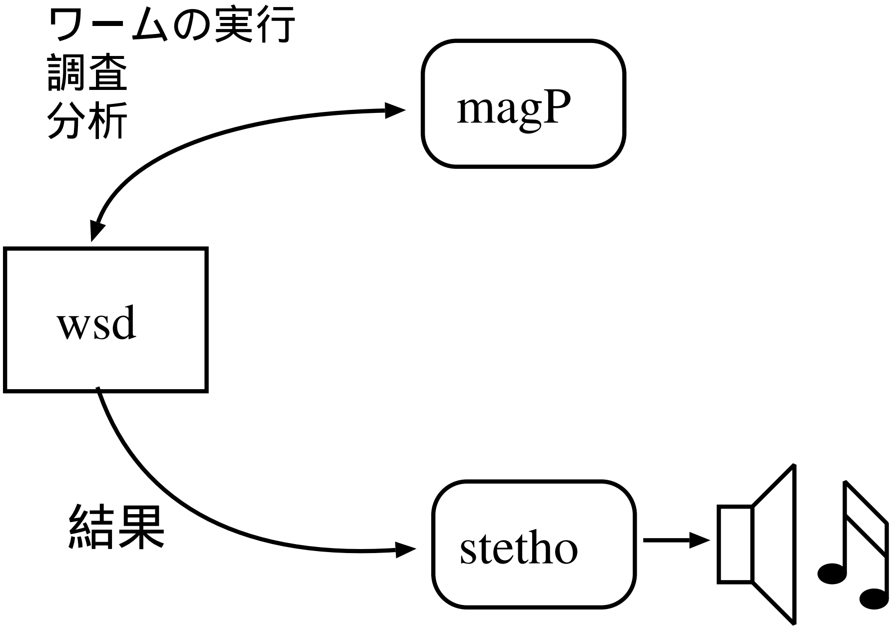

図 6.2 magP、stetho との協調

magP との比較

現在の magP は、ネットワーク上の複数のホストから情報を収集することが可能となっており、NMW System に似た側面を持つ。しかし、管理ホストを中心としたスター型のシステムである、知識ライブラリ記述言語に java を用いているため NMW System に比べると処理が重い、等の違いがある。

magP との協調

magP との協調には 2 通りの方法が考えられる。一つ目は、NMW System では、各ホストでの作業を OWL によって行っているが、この OWL をそのまま magP に置き換える方法である。2 つ目は、OWL から magP を利用する方法である。

magP を利用することにより、情報分析を容易に行うことが出来る。

6.3.2 情報可聴化システム stetho

stetho は、成田[2]により研究されている情報可聴化システムである [15]。

---

[2] 東京工業大学情報理工学研究科数理計算科学専攻

stetho システム

stetho の利用例に、tcpdump の可聴化がある。これは、tcpdump の出力からパケットを種類別に分類し、それぞれ異なる音を割り当て、その発生間隔を再現しながら音で表現するものである。この時、発生量は音量で表現される。その結果ネットワークの状態の把握が容易になった。

これらを実現するため、stetho システムは情報をフィルタリングするための分析部、音で表すための表現部にわかれている。

stetho システムとの協調

NMW System が stetho を利用する場面が 2 通り考えられる。一つ目は magP の場合と同様に、OWL が情報分析のために stetho の分析部を利用することである。2 つめは NMW System の結果報告や管理者への異常発生の通知に音を利用することである。

6.4 NMW System の今後

6.4.1 ネットワーク管理

NMW System が、公衆情報端末の管理システムに優位性を示すことはすでに明らかになった。公衆情報端末は、今後より広く普及していくことになる。そこで NMW System を、その広く大きなネットワークにおいて運用することにより、さらにシステムの安定性を上げ、信頼できる管理システムにしていく。

また、他の管理システムと積極的に協調していくことによりその長所を取り入れ、さらに進んだネットワーク管理システムにしていく。

6.4.2 エージェントシステム

NMW System がエージェントシステムの一面を持っていることは述べた。今後は、そのエージェントシステムとしての側面を応用し、ネットワーク上に存在するサービスの利用や、ネットワーク上の機器操作などに応用していくことが考えられる。

第7章

結論

ネットワークの規模が大きくなりその構成が複雑になると、管理にかかる作業量が増大し、管理者にかかる負担が増える。そのため従来のネットワーク管理手法では管理できなくなる恐れがある。

これらの問題を解決するために、ネットワークワームをネットワーク管理に応用する手法がを提案した。ワームを用いることにより、管理者の作業量の削減、管理作業に必要な通信量の削減、ネットワーク障害への対処が可能となる。これらの検討にもとづき、NMW System(Network Management Worm System) の研究・開発が始められた。

NMW System は、ワームとワームを支援する3つの支援系で構成され、ワームは支援系によってネットワーク上を移動する。そのため、支援系によりワームのコントロールが可能である。さらに、セキュリティ向上のため、ワームの暗号化、電子署名を行い、認証、アクセスコントロールを可能とした。また、API を設計することにより、NMW System を他のシステムなどから利用できるようにした。

NMW System の実験として、いくつかのワームを実行し、その結果について評価を行った。さらに、NMW System を実際の管理作業への利用や、管理作業以外へ応用について述べた。これらにより、NMW System のネットワーク管理への有用性を示した。

また、セキュリティ向上についての考察、他の関連研究との比較、他のシステムとの協調、今後の課題ついて述べた。

謝辞

学部4年、修士1年の2年間、さまざまな面でご指導を頂いた木村泉教授に感謝します。

多くのアドバイスを頂いた慶応SFCの新美さん、Titanet運用センターの飯島さん、応用物理学科技官の山下さん、WIDEプロジェクトの皆さんに感謝します。

また、普段からいろいろとお世話になりました大野研究室、西七号館の皆さんに感謝します。特にこの3年間、ともに研究してきた木本君、中嶋君、成田君の3人に感謝します。

そしてなにより、3年間、本当に熱心に指導していただいた大野浩之講師に心から感謝いたします。

参考文献

[1] Url: http://java.sun.com/.

[2] Url: http://ring.etl.go.jp/openlab/horb/.

[3] URL: http://www.genmagic.com/Telescript/.

[4] Url: http://www.trl.ibm.co.jp/aglets/.

[5] Bharat. K. A. and Cardelli. L. Migratory Applications. Draft of paper for publication.

[6] 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.

[7] John F.Schoch and Jon A.Hupp. The ‘Worm’ Programs – Early Experience with a Distributed Computation. March 1982.

[8] Simon Grfinkel. *PGP: Pretty Good Privacy.* O’Reilly & Associates, Inc., 1995.

[9] 木本雅彦, 大野浩之. 学内情報システム ～citrusの概要～. 第52回全国大会 講演論文集(1), p. 277, March 1996.

[10] 木本雅彦, 大野浩之. 公衆情報検索端末の図書館における利用例. 第53回全国大会 講演論文集(3), p. 475, September 1996.

[11] 木本雅彦, 大野浩之. 公衆情報端末計画 ～picklesの概要～. 第52回全国大会 講演論文集(6), p. 381, March 1996.

[12] 木本雅彦. 大学内公衆情報端末(PICKELS)を用いたネットワーク利用環境の構築に関する研究. 平成8年度修士論文, 東京工業大学大学院 情報理工学研究科数理・計算科学専攻, January 1997.

[13] 木下哲男, 菅原研次. エージェント指向コンピューティング. 株式会社ソフト・リサーチセンター, 1995.

[14] 中嶋良彰. システム管理に必要な情報を自動的に収集・分析するための機構(magP)に関する研究. 平成8年度修士論文, 東京工業大学大学院 情報理工学研究科数理・計算科学専攻, January 1997.

[15] 成田哲也. ネットワークトラフィック可聴化システム stetho のネットワーク管理支援システムへの拡張. 平成 8 年度修士論文, 東京工業大学大学院 情報理工学研究科数理・計算科学専攻, January 1997.

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

[17] 小野木渡. Nmw システムの安全性強化とその広域ネットワークサービスへの応用. 平成 6 年度卒業論文, 東京工業大学情報理工学部数理・計算科学専攻, February 1995.

[18] 小野木渡, 大野浩之. エージェントを利用したネットワークサービス. 第 52 回全国大会 講演論文集 (1). 情報処理学会, March 1996.

[19] R.Kooijman. Divide and conquer in network management using event-driven network area agents. http://wwwsnmp.cs.utwente.nl/research/ARTICLES/GENERAL/koo9505.ps, May 1995.

[20] 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.

[21] 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.

[22] J. E. White. Telescript Technology. General Magic White Paper, 1996.

[23] 池野信一, 小山謙二. 現代暗号理論. 電子情報通信学会, 1986.