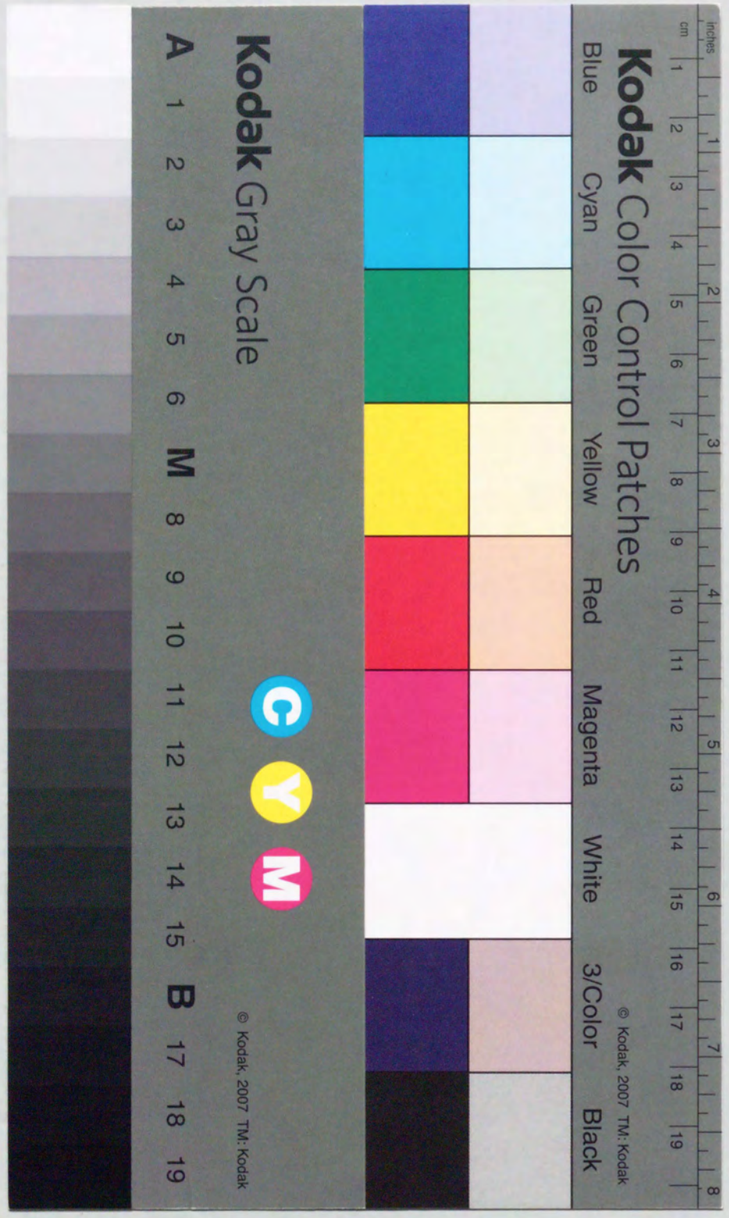
博 士 論 文

インターネットにおけるルータ管理の効率化に関する研究

2001 年 2 月

岡 山 聖 彦



博 士 論 文

インターネットにおけるルータ管理の効率化に関する研究

2001年2月

岡山 聖彦

内 容 梗 概

近年，インターネットにおいては，大規模化と高速性を要求するアプリケーションの増加とにより，通信のより一層の安定化と効率化が重要になっている．このためには，ネットワーク同士の接続点であるルータを如何にして安定に動作させ，ルータにおいて如何にして効率のよい通信を実現するかということが極めて重要である．

ルータを安定に動作させるためには，ルータで発生する障害から如何にして早期に復旧させるかということが基本的な課題として挙げられる．しかし，このための従来技術は，障害の発見の自動化に主眼が置かれており，現状では障害発見後の復旧作業は依然として管理者が手作業で行う必要がある．従って，従来技術のままでは管理者の不在などの理由によって障害が長期化してしまうという問題がある．他方，ルータにおいて効率のよい通信を提供する手段として，複数のバックボーンによりインターネットに接続するネットワーク，すなわち，マルチホームネットワークが注目されている．マルチホームネットワークにおいては，対外接続ルータが複数のバックボーンを効率良く使い分けることが重要である．しかし，従来技術では特定のバックボーンにトラヒックが集中して利用効率が低下するという問題がある．また，導入・管理のためのコストが高い，ユーザやサービスに対する透過性がないという問題もある．

上述したような背景のもとに，本研究では，ルータで発生する障害からの自動復旧法と，マルチホームネットワークにおける従来のトラヒック分散法のような問題を持たない新たなトラヒック分散法とを提案している．

障害からの自動復旧法では，ハードウェア故障を伴わない障害を対象としている．この方法では，管理者が端末を通じて行う復旧作業がプロ

グラミング言語によってスクリプト化できる点に着目し，管理者は想定する障害に対するスクリプトを記述してあらかじめ障害管理支援システムに登録するものとしている．そして，ルータの障害発生時には障害管理支援システムがスクリプトを自動実行し，ルータを遠隔操作することで復旧作業の自動化を図っている．これにより，同じ障害に対して管理者が手作業を繰り返す必要がなくなる．この方法に基づくプロトタイプシステムを実装して動作実験を行うことにより，ルータがプロトタイプシステムからの遠隔操作を受け付ける状態にあればスクリプトの記述に従って障害から自動的に復旧することも確認している．

マルチホームネットワークにおけるトラヒック分散法では，対外接続ルータが自らバックボーンの状態を常に監視することにより，通信先に関わらず適切なバックボーンを選択する方式を採用している．この方式によれば，ルータがバックボーンの状態を監視して適切なバックボーンを選択するので，通信が特定のバックボーンに偏ることはない．また，対外接続ルータのみでトラヒック分散を実現しているので，ユーザやサービスに対する透過性を備えており，従来方式に比して導入・管理も容易である．提案方式を実装した対外接続ルータを２つのプロバイダに接続して実験環境を構築し，実際のインターネット環境にて性能評価実験を実施することにより，提案方式の有効性も確認している．

本研究の自動復旧法とトラヒック分散法はいずれも，ネットワークの管理者がルータに対して行う作業を自動化および省力化するものである．従って，特に，管理者が不足しがちな中小規模の組織のネットワークにおいては，これらの技術が持つ意味は非常に大きいと考えられる．

目 次

図目次

表目次

1　序論

近年，インターネットに接続される計算機や通信回線の容量は急激に増加しており，インターネットを利用するユーザも多様化してきている．また，これに伴って，インターネットで提供されるサービスについても，これまでの学術的利用だけでなく，企業の広告や電子商取引に代表される商業的利用が増加し，多様化が進んできている．しかし，インターネットが次世代の情報通信基盤としての地位を確立するためには，インターネットにおける通信のより一層の安定化とより一層の効率化が重要であると考えられる．

現在のインターネットは TCP/IP プロトコル群 [1] を基礎としており，インターネットに接続する組織のネットワーク (以下，組織ネットワークという) やインターネットへの接続サービスを提供するプロバイダ (Internet Service Provider，以下，ISP という) のネットワークなど，数多くのネットワークを相互接続した巨大なネットワークとして構築されている．しかし，ネットワーク間の接続にはその規模や複雑さに関係なくルータ (router) が用いられ，インターネット上の計算機同士が通信を行う場合には，通信相手が属するネットワークに至るまでの経路上に位置する各ルータがパケットをバケツリレー方式で中継することにより，通信相手に届けるという手法が採られている．従って，上述した安定的かつ効率的な通信の実現のためには，インターネットの規模やサービスの形態に関わらず，ルータを如何にして安定にかつ効率よく運用するかということが極めて重要である．

ルータを安定にかつ効率よく運用するためには，一般に，ルータで使用されるハードウェアとソフトウェアの設定や保守だけでなく，ルータを含めたネットワークの性能管理や障害管理など，ネットワーク管理全

般に関する技術が必要である．特に，通信量の大きいルータに障害が発生すると，ネットワーク全体の安定性が著しく阻害されるので，そのようなルータの早期復旧法の開発は，現在，重要な課題となっている．また，対外接続ルータに2箇所以上の接続点を設け，これらを用いて複数のバックボーンネットワーク(以下，単にバックボーンという)と接続する組織ネットワーク，すなわち，マルチホームネットワークにおいては，外部ネットワークとの通信量が増大した場合，バックボーンのトラヒックに偏りが生じてパケット消失率の増加や遅延の増大が起こるので，この解決を目指した，ルータによる効率的なトラヒック分散手法の開発が重要な課題となっている．

ルータにおける障害からの早期復旧に関する従来技術としては，ネットワークの構成を自動的に把握･監視するためのDIET (Distributed Internet Exception Tracker) [2]，複数の管理者間で円滑な管理作業を行うためのトラブルチケットシステム[3]，ネットワーク管理に必要な情報を統一的に扱うためのプロトコルSNMP (Simple Network Management Protocol) [4][5][6][7]など，管理者の支援を行うシステムやプロトコルが考案されている．

DIETは，ICMP (Internet Control Message Protocol)[8][9]パケットを利用して一定の範囲(ホップ数)内に位置するルータを自動的に探索した後，各ルータに対する到達可能性を管理者が操作する端末上に自動表示するシステムである．これにより，管理者は端末上の画面に注目するだけで障害箇所が推定できる．

トラブルチケットシステムでは，複数の管理者が管理を行っているネットワークを前提としており，障害が発生した場合，最初に発見した管理者がその障害に対するチケット(障害を記録するための電子的な文書)をシステムに登録する．そして，複数の管理者のうち，この障害に対して何らかの復旧作業を行った管理者は，自らの作業内容をこのチケットに逐次記録する．最後に，障害が復旧した場合には，チケットを登録した管理者がチケットを閉じてシステムに保存する．これにより，すべての管理者は，現在発生している障害とその復旧状況をチケットの参照によって簡単に把握することができ，さらに，過去のチケットを参照することによって障害とその対処法などの知識を共有することが可能である．

上述した2つのシステムが管理者を支援するためのアプリケーションであるのに対して，SNMPは，このようなアプリケーションを構築するためのミドルウェアとして位置づけられ，インターネットの標準プロトコルとして採用されている．SNMPはマネージャ･エージェント方式のプロトコルであり，マネージャ･エージェント間の情報交換手順と，管理に必要な情報にアクセスするためのMIB (Management Information Base) [10][11][12]と呼ばれるデータベースを規定している．通常，エージェントはハブやルータなどの管理対象機器内に実装され，管理対象機器のファームウェアやOSとの間で独自の方法を用いて管理情報を交換する．マネージャは，通常，管理者が操作する計算機に実装され，エージェントによって蓄積された管理情報に対して，SNMPで規定された情報交換手順とMIBの構造に従ってアクセスすることにより，管理対象機器の機種やOSの種類に関わらず，管理に必要な情報を統一的な方法によって得ることができる．

しかし，上述した従来技術には以下のような問題がある．障害復旧の手順を時間の流れで捉えると，障害の発見，障害からの復旧，および，障害の記録という3つのフェーズに分けることができる．DIETは障害の発見，トラブルチケットは障害の記録を支援するためのアプリケーショ

ンであるが，その中間の部分，すなわち，障害からの復旧を自動化しようとする試みはまだ行われていない．また，SNMP はネットワーク管理のためのアプリケーションを構築するためのフレームワークであり，これを実装したネットワーク機器は数多いが，その上で構築されるアプリケーションについてはルータも含めたネットワークの自動監視システムが中心であり，障害からの復旧を自動的に行うためのアプリケーションは開発されていない．以上のことから，障害からの復旧は依然として管理者が手作業で行わなければならないという問題があり，障害からの早期復旧を実現する上での妨げとなっている．

他方，従来のマルチホームネットワークにおけるトラヒック分散法としては，自律システム [13] (Autonomous System，以下 AS という) 間経路制御プロトコル BGP4 (Border Gateway Protocol 4) [14] を用いる方式，ネットワークアドレス変換 (Network Address Translation，以下 NAT という) [15][16] を用いる方式 [17]，および，アプリケーションゲートウェイ (Application Level Gateway，以下 ALG という) を用いる方式 [18] が挙げられる．

BGP4 を用いる方式では，インターネット全体を AS と呼ばれる部分ネットワークの集まりとして扱い，AS 間で経路情報を交換することを前提としている．部分ネットワークは，AS 番号を取得した上で，そのネットワークに接続されているすべてのバックボーン[1] との間で経路情報を交換することにより，各バックボーンとの間のネットワークトポロジ，障害の有無，経路制御のポリシなどに基づいて適切なバックボーンをきめ細かく選択することが可能である．

NAT を用いる方式では，各バックボーンから個別アドレスの割り当てを受け，AS 番号のプライベート領域を用いて各バックボーンとの間で BGP4 による経路制御を行うことを前提としている．外部との通信の際には NAT により内部アドレスを個別アドレスに変換するので，自組織ネットワークの経路情報を外部に対してアナウンスする必要がなく，さらに，往路と復路のパケットが同一のバックボーンを通過するように制御できるという特徴を持つ．

[1]各バックボーンもそれぞれ 1 つの部分ネットワークとみなされる．

ALG を用いる方式は，WWW などの一部のアプリケーションを対象に開発されたシステムである．この方式では，各バックボーンに属するアドレスを持つ ALG と，これらの ALG にアクセス要求を転送するための ALG (首振りサーバ) とをそれぞれ導入し，首振りサーバに経路制御機能を付加する．そして，アプリケーションのクライアントが外部にアクセスする際には必ず首振りサーバを経由させることにより，マルチホームネットワークにおいて適切なバックボーンを選択する．経路制御をアプリケーションレベルで行うこと以外は，NAT を用いる方式と同じ特徴を持つ．

しかし，上述した従来方式には以下のような問題がある．BGP4 を運用するためには，ルータの管理者が経路制御技術に関する詳しい知識を持つ必要がある上に，各バックボーンの管理者と協調して管理を行う必要があるので，高い管理コストが要求されるという問題がある．また，BGP4 によって交換される経路情報には現在のトラヒック量などのバックボーンの利用状況が反映されないため，通信先に偏りがあると特定のバックボーンにトラヒックが集中して利用効率が低下する可能性がある．これを防ぐための方法として，往路のトラヒック分散に関してはいわゆる首振りルータ [19] によってある程度解決できるが，通常，インターネットでは往路と復路で独立した経路制御が行われるので，首振りルータを用いたと

しても，復路では特定のバックボーンにトラヒックが集中するという問題がある．NAT を用いた方式でも，バックボーン選択のためには BGP4 を用いてバックボーンから経路情報を取得する必要があるので，BGP4 を用いた方式と同様の問題が発生する．また，ALG を用いた方式を利用できるのはこれに対応した一部のアプリケーションに限られ，しかも，ALG に対応したアプリケーションであってもユーザが ALG の存在を意識する必要があるので，サービスに対する透過性が欠如しているという問題がある．

本研究の目的は，上述した管理者の手作業による障害の長期化という問題を解決するための，ルータで発生する障害からの自動復旧法の開発と，従来のマルチホームネットワークのトラヒック分散法における高い管理コスト，バックボーンの利用効率の低下，および，透過性の欠如という 3 つの問題を持たない，新たなトラヒック分散法の開発である．

ルータで発生する障害からの自動復旧法については，ハードウェア故障に起因しない障害を対象とし，管理者が手作業で行ってきた障害の診断と復旧作業をプログラミング言語を用いてスクリプト化し，DIET や SNMP に基づいて構築された既存のネットワーク監視システムと協調してルータの障害発生時にスクリプトを自動的に実行するための方式を提案する．この方式では，管理者がスクリプトに記述する管理コマンドを抽象化することにより，異なる機種や OS を持つルータを統一的に扱うことが可能であり，同じ障害に対しては管理者の手作業による復旧作業を繰り返す必要がない．さらに，この方式を実装したプロトタイプシステムを用いて動作確認実験を行うことにより，ルータが管理コマンドの遠隔実行を受け付ける状態において想定した障害から自動的に復旧することを確認した．

マルチホームネットワークにおける効率的なトラヒック分散法については，複数のバックボーンと自組織のネットワークとの接続を受け持つルータ (以下，対外接続ルータという) において，対外接続ルータが自ら各バックボーンの状態を常に監視して通信先に関わらずバックボーンを選択することにより，効率的なトラヒック分散を行う方式を提案する．提案方式では，対外接続ルータのみでトラヒック分散を実現しているので，従来の方式に比して運用・管理が容易であり，アプリケーションに対する透過性を備えている．しかも，バックボーンの選択にはコネクションの確立時間を利用して，コネクション確立時に応答の早いバックボーンを常に選択することにより，トラヒックが特定のバックボーンに集中して利用効率が低下するという問題を解消している．さらに，提案方式に基づいて実装した対外接続ルータと 2 つのプロバイダ (バックボーン) を接続して実際のインターネット環境にて性能評価実験を行うことにより，提案方式が複数のバックボーンを最も効率良く利用していることを確認した．

本研究の自動復旧法とトラヒック分散法はいずれも，組織ネットワークの管理者がルータに対して行う作業を自動化および省力化するものであり，サービスの多様化と通信量の増大に伴って重要性が大きくなり続けるルータ管理の効率化が期待できる．特に，管理者の不足しがちな中小規模の組織ネットワークにおいては，これらの技術が持つ意味は非常に大きいと考えられる．

本論文は上述した目的に従って実施した研究成果をまとめたものであり，本章を除いて 3 つの章で構成されている．

2 章では，ルータで発生する障害からの自動復旧法について述べる．まず，本研究で対象とする障害の定義と自動復旧を行うためのシステム

が前提とする条件を示す．次に，管理者が記述するスクリプトに用いるプログラミング言語や，スクリプトをルータに対して自動的に遠隔実行するための機構について述べた後，スクリプトに含まれる管理コマンドの抽象化法と，複数の管理サーバによる協調的管理機構について述べる．さらに，提案方式に基づいて実装したプロトタイプシステムと動作試験に基づく評価について述べる．最後に，残された課題について述べる．

3章では，マルチホームネットワークにおける効率的なトラヒック分散法について述べる．まず，NAT を利用したコネクション単位での経路制御法について説明し，次に，コネクション確立時のネットワークの状態に応じて適切なバックボーンを選択するための基準と，ルータにおけるコネクションの管理法について述べる．さらに，提案方式に基づいて試作したルータの実装法と性能評価実験について述べる．最後に，今後の課題について述べる．

4章では，本研究の成果をまとめるとともに，ルータ管理の効率化を進める上でさらに検討すべき課題や今後の展望に言及する．

2　作業のスクリプト記述に基づいたネットワークの障害管理支援システム

2.1　序言

障害からの自動復旧を実現するためには，復旧作業に必要な情報だけでなく，それぞれの管理者が実施する作業の手続きに関する情報をどのようにしてシステムに持たせるかが重要である．さらに，これらの情報を利用して，システムが管理対象となる機器を操作する仕組みが必要となる．

一般に，部品交換などを伴うハードウェア故障を除き，管理者の作業は管理者が操作する端末を用いて実施される．このとき，管理者は，管理対象機器から情報を取得したり，あるいは，管理対象機器を操作するための管理ツールを利用して，管理ツールの実行とその結果の評価を繰り返しながら作業を行う．例えば，管理対象機器が UNIX などの OS を搭載した計算機の場合，管理者は自分が操作する端末から管理対象機器に対してリモートログインした上で，まず，情報を収集するためのコマンドを実行し，その結果を分析して障害原因を特定する．そして，復旧に必要なコマンドの実行と結果の確認を障害から復旧するまで繰り返す．

本研究ではこの点に注目し，管理者が実施する作業内容をプログラミング言語を用いて記述する方式を提案する（以下，作業内容を記述したプログラムをスクリプトという）．管理者は想定される障害に応じたスクリプトをあらかじめ作成してシステムに登録し，障害発生時にはシステムが自動的に適切なスクリプトを選択して実行すれば，管理者が不在であっても復旧作業を実施することができる．この方式では，管理者があらかじめスクリプトを記述してシステムに登録する作業が必要になるが，

管理対象機器の機種や管理対象機器が接続されるネットワークの構成などに応じたスクリプトを作成することが可能であり，しかも，一度スクリプトをシステムに登録しておけば，同じ障害に対しては管理者の手作業による復旧作業を繰り返す必要がない．

また，ネットワークを介して管理対象となる機器を操作する場合，回線断などによってネットワークが分断されると，システムが管理対象にアクセス不能になる．従って，システムを分散配置して管理対象に対するアクセスの中継機構を持たせることにより，管理対象となる機器が接続されたネットワークが孤立した場合でも，そのネットワーク内で復旧作業を実施することが可能となる．

以下，本研究で提案する障害からの自動復旧法に基づいたネットワークの障害管理支援システムを，SPLICE[2] /NM（Network Manager）という．

2.2 障害の定義とシステム構成のための前提条件

ネットワーク管理の範囲は広く，ネットワーク機器の構成からユーザに対するアカウント作成や課金まで多岐に渡る．これらの項目は密接に関連するが，それぞれ異なる技術や考え方を要求されるため，ネットワーク管理全般について議論するのは困難である．

このため，OSI ではネットワーク管理を「構成管理」,「性能管理」,「障害管理」,「セキュリティ管理」,「アカウント管理」の 5 つに分類している[20]．これらのうち，本研究では特に障害管理に着目し，従来ネットワーク管理者が行ってきた障害からの復旧作業の自動化を目的としているが，ネットワーク上で発生する障害は多種多様であり，管理者によっても捉

[2]Supercomputer, Personal workstation and Local area network InterConnected Environment

え方が異なる．

例えば，ネットワーク性能の低下も障害の一つに挙げることができるが，それが一時的なものなのか，あるいは性能低下がどの程度見られると障害なのかといった分類は困難である．これを解決するためにはネットワークトラヒックを定常的に監視するなど，性能管理に要する技術が求められる．また，インターネットのように大規模なネットワークでは，通信のボトルネックとなる部分が組織外にあるなどの理由によって，問題の解決に時間とコストがかかる場合がある．

従って，本研究では障害の種類を限定し，ネットワークの接続性に重点を置く．すなわち,「あるホストまたはネットワークに到達不可能な状態」および「あるサービスが利用できない状態」を障害とみなす．また，本研究において障害復旧の対象となる機器は，ネットワーク同士の接続点であるルータと，ネットワークサービスを提供するサーバである．以下，管理対象という用語はこれらの機器を指す．

一方，障害からの自動復旧を達成するためには，1で述べた障害からの復旧作業の自動化だけでなく，障害の発見および作業の記録を自動化する機構も必要であるが，これらについてはネットワーク監視システムやトラブルチケットシステムなどの既存のシステムと，SPLICE/NM との連係動作により実現する．本研究で前提とする障害管理モデルを図 1に示す．

図 1の障害管理モデルにおいて，管理対象（managed object）はネットワーク監視システムによって常に監視される．管理対象において何らかの障害が発生すると，ネットワーク監視システムが検知して SPLICE/NM に障害発生を通知する．SPLICE/NM は通知のあった管理対象に対して障害の診断と復旧作業を行ない，その結果をトラブルチケットシステム

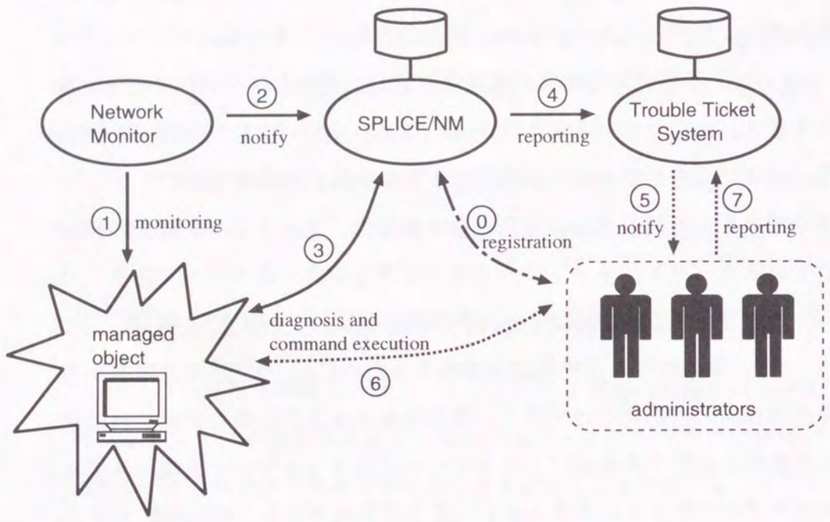


図 1: SPLICE/NM の管理モデル

に記録する．この際，障害が完全に復旧しなかった場合はトラブルチケットシステムが管理者に通知し，復旧作業は管理者に引き継がれる．最終的に障害が復旧すれば，管理者によって作業の結果などが記録される．

2.3 ネットワークの障害管理支援システムの設計

2.3.1 設計方針

SPLICE/NM を中心とした障害管理モデルを実現するためには，障害の診断および復旧作業に必要な情報を，どのようにしてシステムに持たせるかが重要である．

UNIX システムを中心としたネットワークでは，障害管理のために利用できる既存のツールやコマンドが数多く存在する．これらのツールやコマンドとの親和性が高ければ，障害管理システムの開発にかかるコストは低くなるが，インターネットのような異機種分散環境では，以下の2つの問題を解決しなければならない．

- 機種や OS による管理ツールやコマンドの差異

  同じ機能を持った管理ツールであっても，機種や OS によって名前や使用法が異なる．

- ネットワーク構成の差異

  ネットワークの構成はインターネットに参加する組織によってさまざまであり，あるネットワークにおける管理手法が別のネットワークに当てはまるとは限らない．

これらの問題により，SPLICE/NM があらかじめすべての機種や OS，さらには組織ごとのネットワーク構成にまで対応するのは難しい．

このため，本研究では，管理者が行なう診断および復旧作業をスクリプトとして表現する方式を考案した．管理者はこれまで行なってきた作業をスクリプトとして記述し，SPLICE/NM に登録する．あらかじめスクリプトを記述する作業が必要となる反面，それぞれのネットワーク構成や運用形態に合わせて記述することができる．

既存のツールやコマンド，および，SPLICE/NM が独自に定義するコマンドについてはライブラリとして提供する．管理者はこれらを用いてスクリプトを記述するが，機種や OS による差異を吸収するため，ライブラリで扱うツールやコマンドは抽象的に表現される．SPLICE/NM はツールやコマンドの変換テーブルを持ち，スクリプト中のコマンドは管理対象の機種や OS に合わせて変換してから実行される．ライブラリおよび変換テーブルを管理者が変更できるように設計すれば，新たなコマンドや機種の追加などに対して柔軟に対応することが可能である．

さらに，SPLICE/NM は管理対象に対して遠隔操作を行う必要があるため，サーバ・クライアント形式を採用する．以下，SPLICE/NM におけるサーバシステムを管理サーバ，管理対象内で動作するクライアントシステムを単にクライアントという．

管理サーバがクライアントに対してスクリプトに記述されたツールやコマンドを実行するには，障害に対する適切なスクリプトの自動選択機構と，クライアントに対するツールやコマンドの遠隔実行機構が必要である．また，障害の発見については DIET などの既存のネットワーク監視システムを利用するが，このようなシステムが利用できない場合や，システムによっては発見可能な障害が限定されている場合が考えられる．従って，管理サーバにもネットワーク監視機構を持たせ，既存のネットワーク監視システムの機能を補完する必要がある．

以上のことを考慮した結果，管理サーバを 3 つのモジュールで構成する．

- ネットワーク監視モジュール
- スクリプト管理モジュール
- スクリプト実行モジュール

ネットワーク監視モジュールは管理者の設定に従って定期的にクライアントを監視し，異常があった場合はスクリプト管理モジュールに通知する．ネットワークの監視と障害の通知は DIET との連係によって行われるが，DIET は SNMP をベースに管理情報を収集するため，アプリケーションレベルでの障害発見が困難である．ネットワーク監視モジュールは DIET に対する補完的な役割を果たす．

管理者は GUI などのユーザインターフェイスを通じてスクリプト管理モジュールにアクセスし（スクリプトの詳細については 2.3.2で述べる），管理作業の登録および変更などを行う他，管理作業を自動的に実行する際の条件などを登録する．ネットワーク監視モジュール，DIET，および，ユーザからのメールを通じて障害が通知されると，スクリプト管理モジュールは管理者の設定した条件に従ってスクリプトを選択してスクリプト実行モジュールに渡す．スクリプト実行モジュールは，スクリプトの内容に従って，クライアントに管理作業の実行を要求する．

一方，クライアントは管理サーバから送られる管理作業の実行要求をチェックするアクセス制御モジュールと，実際に管理作業を実行するコマンド実行モジュールから構成する．図 2に管理サーバ（NMS）とクライアントの構成を示す．

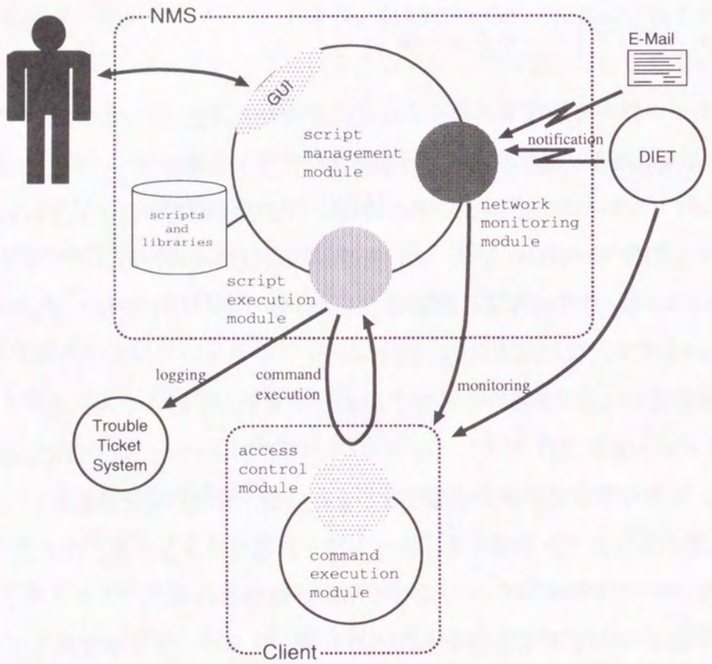


図 2: SPLICE/NM の構成

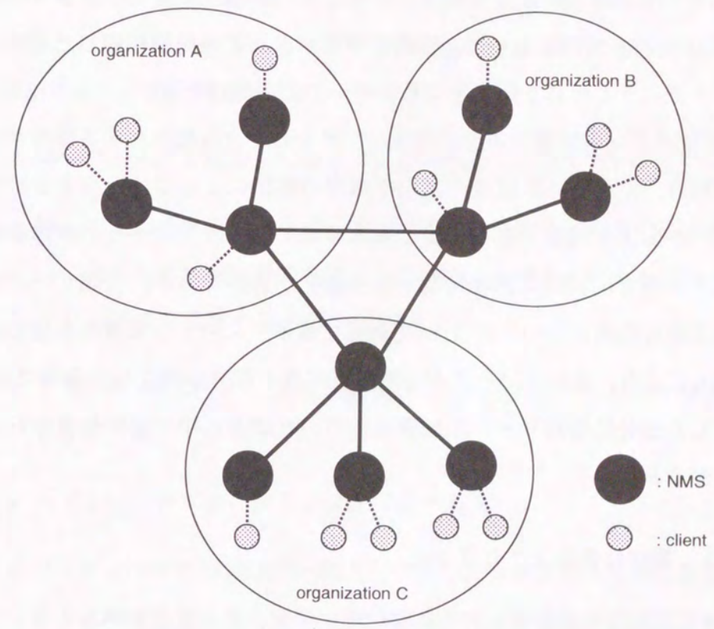


図 3: 管理サーバの配置例

また，SPLICE/NMはインターネットを対象としており，管理サーバを分散配置することによってネットワークの規模と組織の境界に対応している．管理者は隣接する管理サーバを登録することにより，複数の管理サーバをツリー状あるいはネットワーク状に配置することが可能である．例として，組織ネットワーク内では管理サーバをツリー状に，組織ネットワーク間では管理サーバをネットワーク状に配置した場合の構成を図3に示す．図3において，実線は管理サーバ間の接続を表し，点線はクライアントとそれを担当する管理サーバとの関係を表す．

クライアントに対する管理作業は，多くの場合直接担当する管理サーバが行う．ただし，管理者の設定や障害の種類によっては担当する管理サーバが障害を検知できない場合もあるので，他の管理サーバが作業を依頼する形で協調的に管理作業を行う機能が必要である．管理サーバ間の協調管理機構については2.3.5で詳述するが，このような機構を持たせることにより，ネットワークが分断されてある部分が孤立した場合でも，孤立した部分に管理サーバが配置されていればその中で復旧作業を行うことができる．

2.3.2 管理作業のスクリプト化

障害発生後の管理者が行う作業については，その原因がハードウェアによるものなのか，それともソフトウェアなのかによって異なる．原因がソフトウェアにある場合，診断と復旧に伴う作業はおおむねワークステーション上のコマンド実行とその結果の分析の繰り返しである．以下，管理者が端末上で実行するコマンドを管理コマンドと呼ぶ．

本研究ではこの点に注目し，管理コマンドの実行をプログラミング言語でいうところの関数呼び出し，実行結果の確認および分析を条件分岐

で表せると考えた．管理コマンドをライブラリ関数としてシステムが提供し，管理者はこれを利用してさまざまな管理作業をスクリプトとして記述し，管理サーバに登録することにより，管理に必要な情報をシステムに与える．

SPLICE/NMでは，管理作業を記述するためのプログラミング言語としてインタープリタ言語であるPerl[21]を採用した．実行速度の点から考えるとC言語に代表されるコンパイル言語の方が有利であるが，以下の理由によってPerlを導入した．

- 多くのOS上で稼働しており，特定の機種に依存しない．
- Perlで記述されたツールは多く，管理者や一般ユーザも含めて日常的によく使われる言語である．このため，管理者が新たなプログラミング言語を習得する必要がない．
- プログラムをコンパイルする必要がなく，実行までの処理が簡素化できる．
- ネットワークプログラミングが可能である．
- コンピュータの性能は年々向上しており，管理者が手で行う管理作業と比較すれば，インタープリタ言語でも十分対応できる．

言語の仕様はPerlに準ずるが，複数の管理コマンドを並列に実行したり，管理者が独自に作成したライブラリを実行時に取り込むなど，記述性を高めるために以下のようなマクロを用意した．

- INCLUDE "*library_name*" "*library_name*" ...

  スクリプト実行時にライブラリをリンクするためのマクロ．ライブラリが記述されたPerlスクリプトのファイルを指定する．

- PARALLEL *expr*; *expr*; ...

  評価式 *expr* を並列に実行する．

- SCRIPT "*script_name*"

  指定されたスクリプトを実行する．

- LOG "*comments*"

  コメントをログファイルに記録する．

スクリプトの実行時には，プリプロセッサを通じてこれらのマクロが展開された後，Perl インタープリタで処理される．

SPLICE/NM において，管理者が実行する管理コマンドは関数呼び出しの形で表現され，ライブラリとして登録される．ライブラリの一部を表 1に示す．管理者はこれらのライブラリを用いてスクリプトを記述し，管理サーバに登録する．例として，メールサーバの sendmail プログラムを再起動するスクリプトの記述例を図 4に示す．

スクリプトを用いてどのような作業が記述できるかは，ライブラリの種類と量に大きく依存する．しかし，インターネットは異機種分散環境であり，コンピュータのアーキテクチャや OS，周辺機器などもさまざまである．さらに，インターネットに参加する組織においては，組織内で独自のネットワーク運用を行う場合もあるため，すべての環境に対応したライブラリをあらかじめ提供するのは困難である．このため，SPLICE/NM ではシステムが提供するライブラリに加え，管理者がライブラリを作成して追加できるようなインターフェイスを提供している．

また，2.3.1で述べたように，SPLICE/NM の管理サーバにネットワーク監視モジュールを持たせることにより，DIET などの既存のネットワーク監視システムに対する補完的な障害発見機能を提供している．ネット

表 1: SPLICE/NM が提供するライブラリ（一部）

`ping_alive()`	指定されたホストに対して ICMP 応答要求パケットを送り，応答を調べる．
`ping_stat()`	指定されたホストに対して複数の ICMP 応答要求パケットを送り，RTT を測定する．
`get_route()`	経路テーブルを参照する．
`set_route()`	経路テーブルを変更する．
`get_if_stat()`	ネットワークインターフェイスの統計情報を参照する．
`get_if_conf()`	ネットワークインターフェイスの設定情報を参照する．
`set_if_conf()`	ネットワークインターフェイスの設定を変更する．
`get_pid()`	指定したプロセスの PID を参照する．
`ns_look()`	指定したホストのリソースレコードを参照する．
`r_exec()`	引数で与えた管理コマンドを，指定したホスト上で実行する．システムに登録すれば，任意の管理コマンドを実行できる．

```
#!/usr/local/bin/perl
INCLUDE "ping" "remote" "process";
$SUCCESS = 1;
$FAIL    = 0;
$status  = 0;
$ms= 'mailserv.domain.ac.jp';

unless(&ping_alive($ms)){
  LOG "not responding the host: $ms";
  print $FAIL;
  exit(0); }
if(&r_exec(*status,$ms,get_pid)==-1){
  LOG "sendmail on $ms not exists";
  @command = ("sendmail","-bd","-q1h");
  if(&r_exec(*status,$ms,@command)==\
$SUCCESS){
    LOG "sendmail on $ms restored";
    print $SUCCESS;
    exit(0);
  }else{
    LOG "can't execute sendmail";
    print $FAIL;
    exit(0); }
}else{
  LOG "sendmail already exits";
  print $FAIL;
  exit(0); }
```

図 4: 管理作業スクリプトの例

ワークの監視モジュールはアプリケーションレベルでの障害発見を主な役割としているが，ネットワークで提供されるアプリケーションは多種多様であり，さらに，アプリケーションサーバとして利用する計算機の機種やOS，設定などによって監視方法が異なるため，これらのすべてに対応することは困難である．従って，ネットワーク監視モジュールにおける管理対象の監視方法についても，障害からの復旧作業と同様に，管理者がスクリプトとして記述する方法を用いる．以下，障害からの復旧作業のためのスクリプトを「作業スクリプト」，ネットワーク監視モジュールが管理対象を監視するためのスクリプトを「監視スクリプト」と区別し，単に「スクリプト」と記した場合は両者を指すものとする．

スクリプトの作成時には，作成者や作成日時，内容に関するコメントなどの情報をスクリプトの属性として記録する．さらに，スクリプトを管理者が設定するカテゴリに分類して整理することにより，スクリプトの検索効率の向上を図っている．カテゴリは “mail” や “routing” など，管理者が任意に設定することができ，スクリプトの属性として登録する．図5にスクリプトの属性の例を示す．各項目の意味は以下の通りである．

- スクリプト名（SCRIPT）

  スクリプトの名前を表す．

- カテゴリ名（CATEGORY）

  スクリプトが属するカテゴリの名前を表す．

- コメントファイル名（HELP）

  スクリプトに関する簡単なコメントが入ったファイル名を表す．

- 作成者名（C_USR）

```
SCRIPT      : mail.pl
CATEGORY    : mail
HELP        : mail.help
C_USR       : Kiyohiko OKAYAMA
C_TIME      : Thu Feb 9 16:32:19 1997
INTERVAL    : 0
NEXT_SCRIPT : route/route.pl
TRIGGER     : DIET:bulls
```

図 5: スクリプトが持つ属性の例

スクリプトを作成した管理者の名前を表す.

- 作成された時間（C_TIME）

  スクリプトが作成され，システムに登録された時間を表す.

- インターバル時間（INTERVAL）

  スクリプトが監視スクリプトである場合，その実行間隔を示す．作業スクリプトの場合には．図 5のように値として “0” を指定する.

- 次に実行するスクリプト名（NEXT_SCRIPT）

  このスクリプトを実行しても障害復旧が完了しなかった場合に，次に実行すべきスクリプト名を示す.「カテゴリ名/スクリプト名」の形で記述する．図 5の例では,「カテゴリ “route” の “route.pl” というスクリプト」を示している.

- スクリプトを選択するためのトリガ情報（TRIGGER）

  管理サーバが障害に対して適切な作業スクリプトを選択して実行するための検索キーであり,「トリガ名：キーワード」の形で表現される．トリガ名は障害を発見して管理サーバに通知するシステムやユーザの名称であり，キーワードは障害の発生した計算機の名前や障害の種類などを示す．図 5の例は,「“DIET System” から “bulls” という名前の計算機に関する障害発生報告があった時には，このスクリプトを起動する」ことを意味する.

なお，スクリプトおよびライブラリの蓄積には UNIX ファイルシステムを用いた．カテゴリはディレクトリ，スクリプトおよびライブラリはファイルにそれぞれ対応する.

2.3.3 スクリプトの自動実行機構

管理作業を自動化するには，作業に必要な情報を管理サーバに与えると共に，障害発生時にどの作業を行うのかという情報をあらかじめ登録しておかなければならない．SPLICE/NM において前者はスクリプトとして表現されるが，後者を実現するには障害とスクリプトの対応づけが必要である．

SPLICE/NM では，管理サーバに対して障害を通知するために以下の3つの方法を用いている．

1. DIET による障害の通知

   DIET は ICMP パケットを利用して，管理者が指定したルータの応答を定期的に監視する．ルータが応答しなくなった場合には，DIET はそのルータの IP アドレスとホスト名を管理サーバに通知する．

2. SPLICE/NM のネットワーク監視モジュールからの通知

   ネットワーク監視モジュールは，監視スクリプトの記述に従って定期的に管理対象を監視する．障害が発生した場合には，管理サーバ（スクリプト実行モジュール）に通知する．

3. ユーザからのメールによる通知

   上記2つの監視機能が動作していない場合や，管理者が想定していない（スクリプトに記述されていない）障害が発生した場合，ユーザからのメールによって管理サーバに障害を通知する．

障害の通知に含まれる情報としては，障害が発生した管理対象のホスト名と IP アドレス，障害の発生したサービス名（mail や news など），発生日時，報告者などが含まれる．これらはトリガ情報として管理サーバに

与えられ，管理サーバのスクリプト実行モジュールはトリガ情報をキーとして作業スクリプトを検索し，マッチした作業スクリプトを実行する．マッチする作業スクリプトが見つからない場合には，管理者にメールで通知する．

2.3.4 管理コマンドの遠隔実行機構

管理サーバが作業スクリプトを実行すると，スクリプトに記述された管理コマンドがクライアントに対して遠隔実行される．実際には，管理サーバからクライアントに対して管理コマンドの実行が依頼され，クライアントにおける実行結果が管理サーバに返される．異機種環境においては，同じ機能をもつ管理コマンドでも機種や OS によって名前や（オプションなどの）使用法が異なる場合があり，さらに管理者の設定よっても異なる場合がある．スクリプトを記述する管理者の立場から考えると，SPLICE/NM がこれらの差異を吸収し，管理コマンドを抽象的かつ統一的に表現できることが望ましい．

これを実現するため，抽象的な表現を実際の管理コマンドに変換するためのテーブルを SPLICE/NM に持たせる．この場合，どの部分にテーブルを持たせるかが問題となる．管理サーバに持たせた場合，変換テーブルを集中的に管理できる反面，機種ごとの変換テーブルを全て管理サーバが持つ必要があり，クライアントに変更があった場合は管理サーバに反映させなければならない．

従って，SPLICE/NM では変換テーブルをクライアントに持たせ，管理サーバから `r_exec()` ライブラリによる管理コマンドの実行依頼があった場合には，クライアント側で実際の管理コマンドに変換してから実行する．クライアントが持つ変換テーブルの例を表2に示す．

表 2: 変換テーブルの例

抽象的な管理コマンド	実際の管理コマンド
kill_hup $1	/bin/kill -HUP $1
reboot $1 $2	/sbin/shutdown -r $1 $2
route_delete $1	/sbin/route delete $1

注：$1，$2 は管理コマンドに与える引数

2.3.5 管理サーバ間の協調管理機構

SPLICE/NM では管理サーバを分散化することにより，ネットワークの規模と組織の境界への対応を図っている．各管理サーバは少なくとも 1 つ以上のクライアントを持ち，クライアントは管理サーバによって管理される．

管理サーバとクライアントが 1 対多の場合，障害によってクライアントが管理サーバと切り離されると，管理サーバはクライアントに対して復旧作業を行なうことができない．しかし，障害の種類によっては，同じネットワークに属する管理サーバとクライアントが分断されても，別のネットワークからはアクセス可能である状況が考えられる．さらに，監視スクリプトの記述内容や実行間隔によっては，クライアントで発生した障害を直接担当する管理サーバが検知できなかったり，他の管理サーバが先に検知する可能性もある．従って，管理サーバとクライアントの関係は多対多であるのが妥当であり，複数の管理サーバを介した作業依頼の中継機能が必要である．

SPLICE/NM では，作業依頼の中継は図 6のようにして行なう．管理サーバはあるクライアントに到達するための静的な経路テーブルを持ち，IP パケットがルータを介して配送されるような形で管理コマンドの実行要求が中継され，クライアントに送られる．クライアントで処理が行なわれた結果は，逆の順序で実行要求を行なった管理サーバへ返される．

ただし，インターネットにはさまざまな組織ネットワークが接続されており，組織ごとに独自のポリシを持って運用されている．SPLICE/NM でいうポリシとは「ある管理サーバがあるクライアントに対して，どの管理コマンドを実行できるか」という意味に置き換えられる．従って，クライアント側でアクセス制御を行ない，実行できる管理コマンドの集合

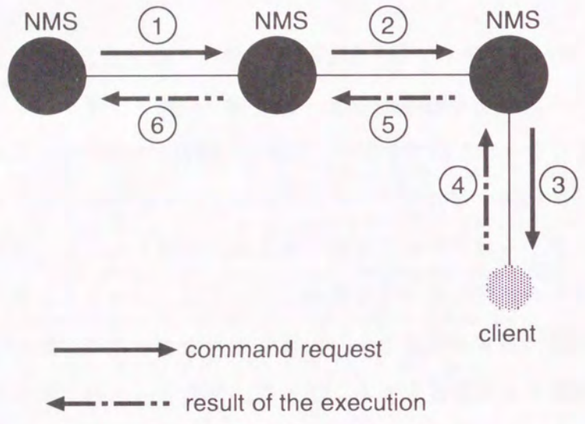


図 6: 管理コマンド実行要求の中継

を管理サーバによって変える必要がある．管理サーバごとに許可を行うと設定が煩雑になるため，SPLICE/NM では管理サーバとクライアントをグループ化し，クライアントが許可する管理コマンドの種類をグループ単位で設定できるようにした．以下，このグループのことをコミュニティという．あるコミュニティには，一つ以上の管理サーバとクライアントが含まれる．

管理サーバとクライアントをコミュニティでグループ化した例を図 7 に示す．図 7において，クライアントは “comA” および “comB” という二つのコミュニティに属する．クライアントは同じコミュニティに属する管理サーバからの管理コマンド実行要求しか受け付けないため，comC に属する管理サーバである nms4 からの要求は却下される．さらに，クライアントにおいて comA と comB に対して実行可能な管理コマンドの種類を変えることにより，コミュニティにおけるポリシの違いを反映させることができる．

コミュニティによるアクセス制御は，ポリシの違いを反映させるためだけではない．コミュニティを設定する際は，あるコミュニティに属する管理サーバを操作する管理者間でネゴシエーションが必要であるが，逆に言えば管理者が把握していない管理サーバからの要求を防ぐという意味を持つ．

2.4 プロトタイプシステムの実装と評価

2.3で述べた設計に基づき，管理サーバとクライアントを試作して動作実験を行った．実験には 4 台のワークステーションを用い，このうち 3 台で管理サーバを，残りの 1 台でクライアントを稼働させた．管理サーバを 3 台用意したのは，管理コマンドの中継動作を確認するためである．

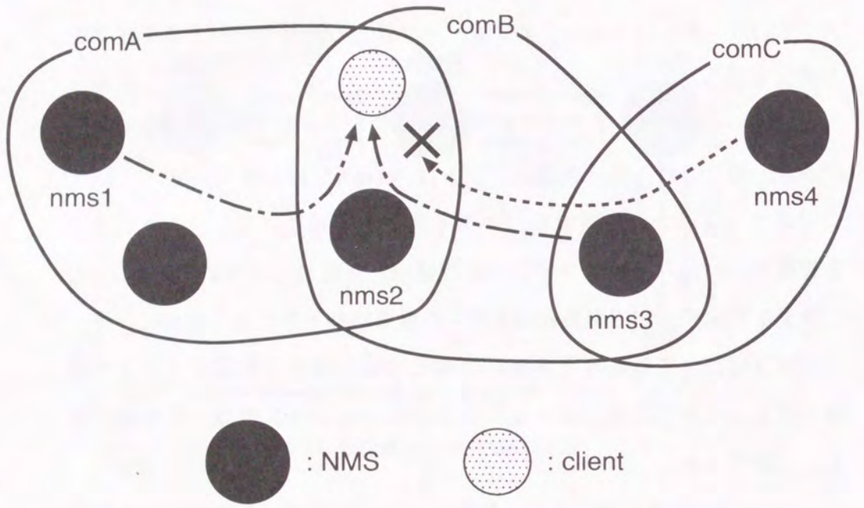


図 7: コミュニティの設定例

これ以上管理サーバの数を増やしても，その分中継役の管理サーバが増えるだけであるから，3台の管理サーバによる連係動作が成功すれば十分であると考えた．

管理者と管理サーバとのユーザインターフェイスは Tcl/Tk を用い，マウスの操作でスクリプトおよびライブラリの作成や変更，削除などを行なうことができる．管理サーバとクライアントの各モジュールは，C 言語および Perl を用いて実装を行った．

ただし，現時点では SPLICE/NM のみの実験であり，DIET との連携動作は実装されていない．ネットワーク監視モジュールによるクライアントの監視や，監視モジュールやユーザからのメールによって障害通知を受けたスクリプト管理モジュールが適当な作業スクリプトを起動するなど，モジュール間の連係動作が問題なく行われることが確認できた．また，3台の管理サーバ間でコミュニティの設定を変えることにより，管理コマンド実行要求の中継が設計通りに機能することも確認した．動作確認実験における管理サーバの実行画面を図 8に示す．

SPLICE/NM の導入による利点は，障害発生から復旧作業の実行までにかかる時間の短縮と，管理作業のスクリプト化による管理者の負担減である．ネットワーク監視モジュールによるクライアントの監視はポーリングベースであるため，最悪の場合ポーリング間隔の時間分だけ障害の発見が遅れる可能性があるが，逆に言えば管理者の設定したポーリング間隔内で確実に障害を発見できる．さらに，SPLICE/NM は電子メールによる障害通知のインターフェイスを持つため，より迅速な障害発見が期待できる．

障害発見後の復旧作業については，SPLICE/NM が提供するライブラリや，管理者が記述するスクリプトに依存する．作業スクリプトは管理者

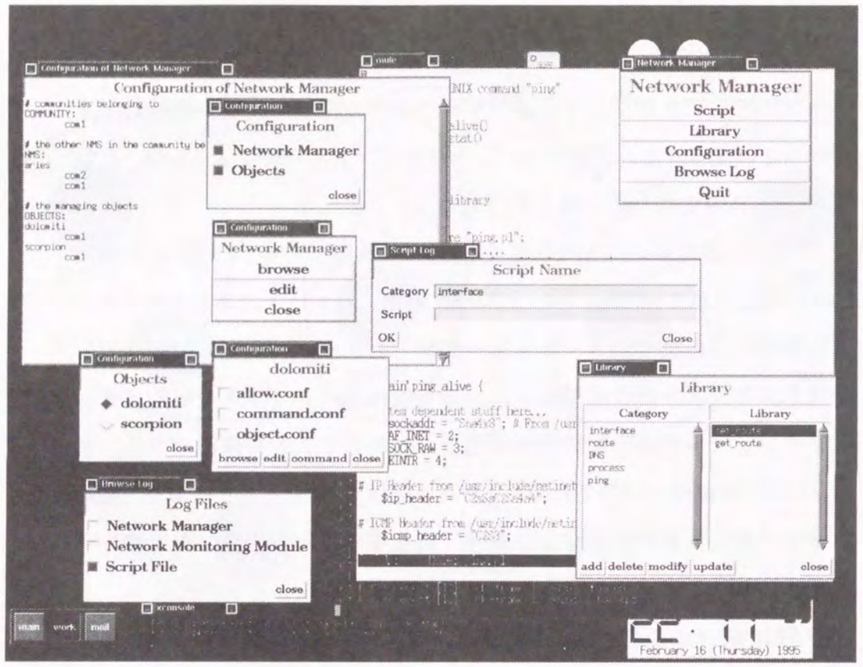


図 8: 管理サーバの実行画面

が記述しなければならないため，管理者が予想しない障害に対しては自動的に復旧作業を行なうことができない．しかし，一度スクリプトに作業を記述しておけば，同じ障害については管理サーバが自動的に対応できるので，管理者が同じ作業を繰り返す必要がなくなる．また，SPLICE/NM が完全に障害を復旧できなかった場合も，作業結果は管理者が引続き作業を行なう場合の判断材料になる．

SPLICE/NM における実装上のポイントは，障害復旧に必要な機能を管理サーバとクライアントに分割したことと，管理サーバをネットワーク構成に合わせて分散化した点にある．機種や OS の違いによる管理コマンドの差異はクライアントが持つテーブルによって抽象化されるため，管理サーバから見ると，クライアントに対するすべての操作は統一されている．このため，管理サーバで特別な機構を導入することなく，機種や OS の異なる複数のクライアントを同時に管理することが可能である．また，管理サーバを分散配置して複数の管理サーバがクライアントを管理することにより，回線断などによってネットワークが孤立した場合でも管理作業を行なうことができる．

SPLICE/NM を稼働させるためには，少なくとも管理サーバおよびクライアントを実装するホストに Perl が必要である．Perl は無償で提供されるソフトウェアであり，ほとんど全ての UNIX 系 OS で動作するため，Perl によって SPLICE/NM の動作環境が限定されることはない．また，管理サーバでは Tcl/Tk を用いた GUI だけでなく，キャラクタベースのユーザインターフェイスも利用可能である．ライブラリやスクリプトは UNIX ファイルシステムにマッピングされているため，参照や変更に特殊なツールを用いる必要がない．

SPLICE/NM をインターネットで運用するには，スクリプトとライブ

ラリの共有やセキュリティ対策など，いくつかの点において機能強化や改善を行なう必要がある．最後に，検討課題や解決すべき問題をまとめる．

- スクリプトとライブラリの共有

  SPLICE/NM では管理者がスクリプトを記述し，必要ならばライブラリの追加を行なう．これらをネットワーク上で共有し，他の管理者が記述するスクリプトなどを参照できれば，新たにスクリプトを作成する際の助けになる．

  これを実現するには，分散ファイルシステムや分散データベースなどの技術を利用することができる．ネットワークにおけるスクリプトやライブラリの共有は SPLICE/NM の本質的な機能ではないため，既存の ftp や WWFS[22]，あるいは WWW などといった別の枠組で実現する方法もある．

- セキュリティ対策

  ネットワークを介して管理対象の操作を行なう場合，常になりすましや通信の妨害といったセキュリティ上の問題を考慮しなければならない．SPLICE/NM ではコミュニティという概念を用いたアクセス制御機構が実現されているが，管理サーバ間および管理サーバ・クライアント間の認証および暗号化通信機構は未実装である．

  しかし，管理サーバ間および管理サーバ・クライアント間の通信には TCP あるいは UDP を用いているので，認証および暗号化通信機構には既存のものを利用することができる．ネットワークにおける認証機構としては，Kerberos[23]，SPLICE/AS （Authenticaion System）[24][25]，および，証明書を用いた認証機構 [26][27] などが挙げられる．一方，ネットワークにおける暗号化通信機構としては，IPsec[28]～[37]，TLS （Transport Layer Security）[38]，および，Secure TCP[39] などが挙げられる．これらの各機構は，前提とするネットワークプロトコルの階層や適用範囲などが異なるので，SPLICE/NM を導入するネットワーク環境に応じて適切なものを選択する必要がある．

- 専用ルータの問題

  SPLICE/NM では管理サーバだけでなく，クライアントにもアクセス制御モジュールやリモート実行のモジュールを実装する必要がある．しかし，ネットワーク間の接続に用いるルータには専用の装置を用いることが多く，特に，UNIX が稼働していない機器については，クライアントに必要なモジュールの実装が困難である．

  これを解決するには，管理サーバからの管理コマンド実行要求をルータに独自のコマンドに変換する代理（proxy）クライアントの導入が必要である．また，専用ルータに関しては，電源の ON/OFF による再起動によって障害が解決される例も数多くみられる．我々の研究グループでは，ホームオートメーションの規格である X10 に SNMP を応用した制御システムを開発しており，このような技術を導入すれば，より多くの機器に対応可能となる．

- 管理コマンドの抽象化

  SPLICE/NM では異機種環境に対応するため，管理コマンドの抽象化を行なってスクリプトを記述する際の統一的な表現を可能にしている．実際に使われる管理コマンドへの変換にはクライアントが持つテーブルを利用するが，現状では抽象化の方法が規定されていないため，ネットワーク全体で表現の不整合が発生する恐れがある．

スクリプトによって管理コマンドの表現が統一されてなければ，スクリプトを共有した際，他者の記述したスクリプトの理解に余分な労力を費すことになる．

管理コマンドの統一的な抽象化については，クライアントが持つ変換テーブルをそのままネットワーク上で共有する方法や，SNMP の MIB のように，管理コマンドの実行によって変化するクライアントの状態をオブジェクト化し，これをカテゴリに分けてツリー構造を持たせる方法が考えられる．前者は実際に使われる管理コマンドを連想しやすいという点で有利であり，後者はツリー構造という枠の中でコマンドの拡張ができるという利点がある．

管理コマンドを抽象化する方法についてはスクリプト記述の容易さに直接影響を与えるため，記述性を損なわないことを念頭において検討する必要がある．

- 管理サーバと管理者間の協調管理

  作業スクリプトを実行しても障害が改善されなければ，スクリプトの記述方法によっては同じ作業スクリプトの実行が繰り返される可能性がある．また，ユーザからのメールによって障害の通知が行なわれた場合，管理者の作業中に作業スクリプトが実行される場合もある．従って，障害に対する作業状況を管理サーバおよび管理者が的確に把握し，作業の衝突などが発生しないような仕組みが必要である．

  複数の管理者による協調管理を支援するためのシステムとしては，トラブルチケットシステムの一つである T3[40] がある．管理サーバと T3 とのインターフェイスを規定し，管理サーバを T3 における管理者グループに組み込むことにより，作業の整合性を保つことができると考えられる．

2.5 結言

本章では，ネットワークにおける障害管理に注目し，障害からの自動復旧を実現するための方式と，これに基づいたネットワークの障害管理支援システム SPLICE/NM について述べた．従来管理者が端末の前で行なってきた障害の診断と復旧作業をスクリプトとして表現し，障害に応じて適切なスクリプトをシステムが起動することにより，障害復旧作業の自動化を実現している．作業のプリミティブとなる管理コマンドに関しては，システムがライブラリとして提供し，既存の管理コマンドを活用できるように設計を行なった．

現状では最小限のシステム構成で実験を行なっており，基本的な機能について十分な検討を行なった後，適用範囲を岡山大学の学内ネットワークへ拡大する．この段階で DIET や T3 などのシステムとの連携も含めた実験を行ない，インターネットでの実用化に向けて開発を進める予定である．

3　マルチホームネットワークにおける透過的な動的トラヒック分散

3.1　序言

1で述べた従来のマルチホームネットワークにおけるトラヒック分散法の問題のうち，管理コストが高いという問題は，経路制御を行うためにバックボーンなど他組織のネットワーク管理者と協調作業を行わなければならないために生じる．さらに，バックボーンの利用率が低下する問題は，バックボーンの選択基準に現在の各バックボーンの利用状況が反映されておらず，しかも，方式によっては往路，すなわち，組織ネットワーク内から外部へ送信されるパケットの制御しか行われない点に原因がある．また，透過性の欠如という問題は，ネットワークプロトコルの階層構造におけるアプリケーション層でトラヒック分散を実現していることに起因する．

従って，上述した問題を持たないトラヒック分散法を実現するためには，以下の３つの条件を満たす必要がある．

1. 自組織のみで運用・管理が可能であること
2. バックボーンの選択に現在のバックボーンの利用状況が反映され，かつ，往路のトラヒック分散が可能であること，
3. トラヒック分散機構をトランスポート層以下で実現すること

これらの条件を満たすため，本研究では，対外接続ルータが各バックボーンの利用状態を監視し，その結果のみに基づいてコネクション単位で適切なバックボーンを選択する方法を提案する．提案方式は，各バック

ボーンとの間で経路情報を交換する必要がないため，導入や管理が容易であり，しかも，各バックボーンの現在の利用状況に基づいてコネクション毎に各バックボーンを選択することにより，通信先に偏りが生じた場合でも用いて効率的にトラヒックを分散することが可能になる．さらに，提案方式はコネクション単位，すなわち，トランスポート層でのトラヒック分散法であるため，アプリケーションに対する透過性も備えている．

3.2 透過的な動的トラヒック分散法

3.2.1 トラヒック分散機構

本研究では比較的小規模かつインターネットの末端に位置する組織ネットワークを対象とし，図9に示すように，自組織のネットワーク（LAN）を一つのルータRにより二つのバックボーン（B1およびB2）に接続した構成を取るものとする[3]．また，自組織のネットワークにはB1から与えられたアドレスが割り当てられており，B2とはNATを経由してアクセスするように設定されているものとする．

このような構成のネットワークにおいて，提案方式では内部から外部へのTCPコネクションを対象とし，コネクション確立時にルータが適切なバックボーンを選択する．なお，外部から内部へのTCPコネクションにおけるトラヒック分散については，DNSラウンドロビン[42]など従来の方法で対処することができると思われるため．本研究ではこれ以上議論しない．以下では，往路と復路それぞれにおけるトラヒック分散について述べる．

往路のトラヒック分散は，ルータがそれぞれのバックボーンの状態を監視し，新たなコネクション確立を要求するパケットが来ると，その時

[3]3つ以上のバックボーンに接続することも可能であるが，説明の都合上省略する．

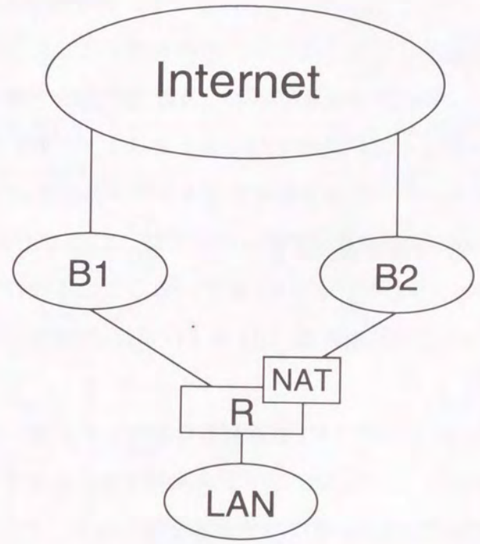


図 9: マルチホームネットワークの構成

点での各バックボーンの状態から適切なバックボーンを選択することにより行う．一度コネクションが確立されると，そのコネクションに属する以降のパケットは同一のバックボーンを利用する．

適切なバックボーンは，ルータの負荷やリンクの利用率など，その利用状態に応じて動的に変化するため，その選択にはコネクション確立時でのバックボーンの状態が反映されるような選択基準を用いる必要がある．また，このような選択基準を用いることにより，通信先が偏っている場合でもコネクション単位での適切なトラヒック分散が可能となる．

バックボーンの選択基準の詳細については，3.2.2で述べる．

復路のトラヒック分散は，NAT を用いることにより行う．この手順を，図 10において 2 台の計算機（H1 および H2）が通信を行う場合を例にとり説明する．

まず，ルータ R が往路で B1 を選択した場合を考える．この場合，H1 から送出されたパケットは R でアドレス変換することなく H2 にそのまま届き，復路では H2 は H1 宛にパケットを送り返す．ここで，H1 は B1 から割り当てられたアドレスを用いているため，このパケットは往路と同じ B1 を経由して H1 に届く．

一方，R が往路で B2 を選択した場合には，R は NAT を用いて通信元アドレスを H1 から R2 （B2 から割り当てられたアドレス）に変換するため，復路では H2 は R2 宛にパケットを送り返す．ここで，R2 は B2 から割り当てられたアドレスであるため，このパケットは往路と同じ B2 を経由して R に届き，発信先アドレスが R2 から H1 に変換されて最終的に H1 に届く．

以上のように，NAT を用いることにより往路と復路は同一のバックボーンを経由することになるため，往路でトラヒック分散を行うと復路

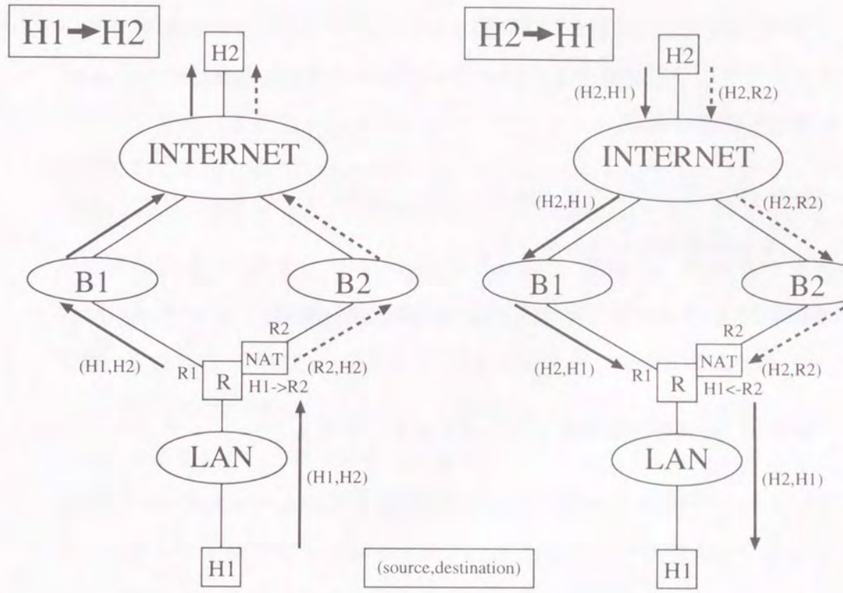


図 10: 往路および復路でのパケットの流れ

でも自動的にトラヒック分散が行われることになる．

3.2.2 バックボーンの選択基準

提案方式では，コネクション要求ごとにネットワークの状態を把握してバックボーンを選択するため，バックボーンの選択基準は以下の条件を満たす必要がある．

（条件 1） コネクション確立時点における通信先までのネットワーク状態を反映するものであること．

（条件 2） バックボーンの状態の評価を短時間かつ低コストで行えるものであること．

（条件 3） 通信先に関わらず適用できるものであること．

（条件 4） コネクション確立時に障害が発生しているバックボーンを選択しないものであること．

ネットワークの状態を測定するための既存の方法としては，ICMP パケットを用いる NEPRI[43], Bprobe[44] や UDP パケットを用いる pathchar[45], TReno[46] などが知られている．これらの方法はいずれも，測定のためのパケットをネットワークに送出し，その応答時間などを測定することによってネットワークの性能を実測するので，測定時における通信先までの経路の性能を的確に把握できるという特徴を持つ．

しかし，このような方法は，測定の精度を上げるために多数のパケットを連続してネットワークに送出するので，コネクション要求毎に測定を行おうとすると，測定によってコネクションが確立するまでの時間が大きくなると共に，ネットワークに多大の負荷がかかることになり，条

件 2 に反する．さらに，ICMP や UDP を用いた測定ではフィルタリングによって応答が返されない場合があるため，条件 3 を満たすことができない．

一方，ルータにおいて特定のサーバ群に対して負荷分散を行う方法として，LocalDirector[47] では各サーバへの新たなコネクションをラウンドロビン方式により割り当てる方法やコネクション数が均等になるように割り当てる方法が用いられている．しかし，これらの方法は各サーバの性能やルータと各サーバとの間のネットワーク特性が同一であることを前提としているため，このような前提が成り立たない一般的なマルチホームネットワーク環境では効果的なバックボーン選択が期待できない．また，これらの方法は上記の条件 1〜3 を満たすが，条件 4 は満たさない．

これらの問題を解決するため，本研究では，バックボーンの選択基準としてコネクションの確立時間を提案する．これはコネクション確立時に全てのバックボーンを用いて通信先とのコネクションの確立を試み，このうち最も早く確立できたバックボーンを選択する方法である．以下では，実際にルータで行われるバックボーン選択処理について述べる．

TCP コネクションの確立には，3 ウェイハンドシェイクが用いられる．3 ウェイハンドシェイクでは，H1 から H2 に対してコネクションを確立しようとする場合，実際に確立するまでに以下の 3 つのパケットの送受信を必要とする．

1. H1 は H2 に対して SYN フラグ付きのパケット（SYN パケット）を送る．
2. H2 はこのパケットを受け取ると H1 に対して SYN フラグと ACK フラグの両方が立ったパケット（SYN+ACK パケット）を送る．

3. H1 はこのパケットを受け取ると H2 に対して ACK フラグ付きのパケット（ACK パケット）を送る．

このとき，最後の ACK パケットの代わりに H1 が RST フラグ付きのパケット（RST パケット）を H2 に送ると，このコネクションは確立されず直ちに破棄される．この性質を利用して，ルータは最も早く SYN+ACK パケットを送り返したコネクションだけを選択する．すなわち，ルータは以下のように動作する．

1. ルータは内部から外部への SYN パケットを受け取ると，このパケットを複製して全てのバックボーンに送出する．このとき，アドレス変換が必要なバックボーンについては，送信元のアドレスをバックボーンのアドレスに変換する．

2. ルータはあるバックボーン B から上記の SYN パケットに対する最初の SYN+ACK パケットを受け取ると，そのパケットを（必要であればアドレス変換を行った上で）本来の送信先に中継する．また，今後このコネクションに対してバックボーン B を利用するように記録する．

3. ルータは他のバックボーンから 2 番目以降の SYN+ACK パケットを受け取ると，そのパケットを中継せずに破棄し，代わりにそのバックボーンを用いて RST パケットを送出する．

例として，図 10と同様のネットワーク構成において，B1 が選択される場合のパケットの送信手順を図 11に示す．

コネクション確立時間は通信先との RTT （Round Trip Time）の測定と同義であり，通信先との距離（ホップ数），H2 およびその経路上に

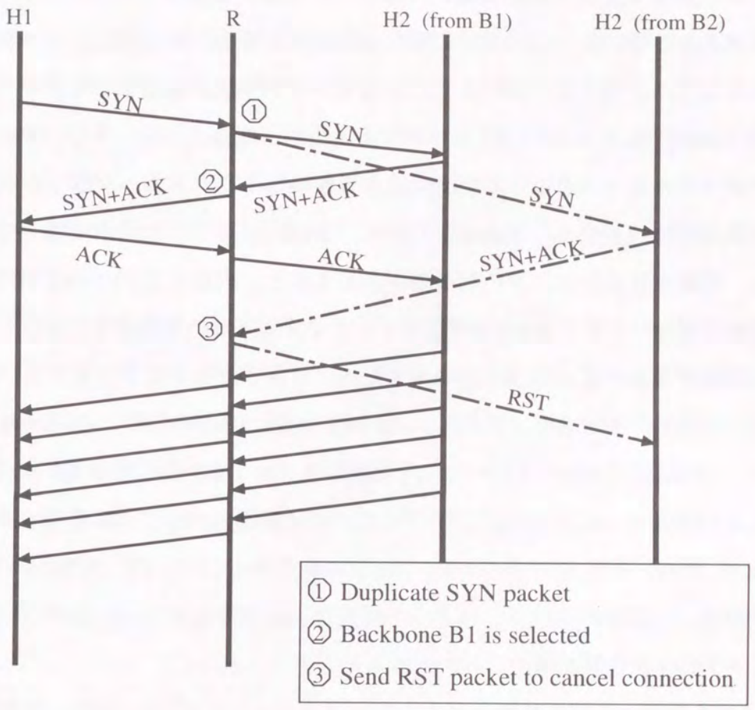


図 11: バックボーン選択時のパケットの送信手順

あるルータの性能や負荷，経路上の各リンクの容量や利用率などの要素により影響を受けて変化する．一般に，これらの要素はRTTとスループットとの間に負の相関関係をもたらす要因となる．例えば，ホップ数に関しては，他の条件が同じである場合，ホップ数が増加すると，ルータでの処理時間のためRTTが増加する一方で，経路上のリンクのいずれかがボトルネックとなって利用可能容量が小さくなりスループットが低下する可能性が高くなると考えられる．また，リンクの利用率に関しては，経路上のあるリンクの利用率が高くなると，当該リンクへの送出待ち時間増加のためにRTTが増加する一方で，利用率の増加は当該リンクを利用する通信量の増加を意味するため，コネクション当たりのスループットは減少すると考えられる．ただし，あるコネクションに限った場合，他組織間の通信などの一時的な外乱を受けてRTTの分散が大きくなることにより，必ずしも適切なバックボーンが選択されない可能性もある．しかし，提案方式ではコネクションを確立する毎にRTTの測定を行うので，長期的にみればこのような外乱はある程度吸収され，全体としてトラヒック分散が適切に行われるものと考えられる．

コネクション確立時間を利用した方法は，ルータがSYNパケットを複製し，これらの応答によってのみバックボーンを選択するので，条件1と2を満たしており，ICMPやUDPのようにフィルタリングされることがなく，あるバックボーンが故障しても応答があった他のバックボーンを選択するため，条件3と4も満たすことができる．

また，コネクション確立時間をバックボーンの選択基準として用いた場合，選択可能なすべてのバックボーンに対して，SYN，SYN+ACK，および，RSTの3つのパケットが流れることになる．従って，提案方式を実装したルータが経路上に複数存在すると，各ルータでの複製によって

通信量とH2の負荷が増加する可能性がある．しかし，本研究では，インターネットの末端に接続している組織が構築するマルチホームネットワークを対象としているので，組織の出口となるルータに対してのみ提案方式を適用すれば，選択されなかったすべてのバックボーンに対して上述した3つのパケットが増加するだけである．さらに，通信先の計算機でのSYNパケットの処理はカーネル内部のみで行われ，ユーザプロセスの起動やデータ授受は行われないので，H2に対する負荷も比較的小さいと考えられる．

なお，提案方式ではバックボーン選択にNATルータを用いており，これにSYNパケットの複製機能と選択されなかったバックボーンに対するコネクション破棄機能を追加している．このため，NATルータには追加された機能を処理するための負荷がコネクション要求に応じて加わるが，本研究では通信量の比較的小さい小規模な組織ネットワークを対象としているので，NATルータにかかる負荷も比較的小さく，十分実用に耐えられると思われる．

3.2.3 コネクションの管理法

通常，NATを経由して内部ネットワークから外部ネットワークへアクセスする場合，NATの内部ではどのようにアドレス変換を行えばよいかをコネクション単位で管理している．提案方式においてもNATの利用を前提としているためコネクション管理が必要であるが，NATを経由しないコネクションについてもどのバックボーンを選択したかを記録するため同様のコネクション管理を行う．

ルータには現在確立されているコネクションを管理するための表（以下，コネクション表という）を設け，通信元の内部アドレス及びポート

番号，アドレス変換後のアドレス及びポート番号，通信相手のアドレス及びポート番号，選択したバックボーンへのインタフェースなどを記録する．ルータはこのコネクション表を利用して以下のように動作する．

1. 内部から外部へのSYNパケットを受け取ると，前節で述べた方法によりバックボーンの選択を行い，このコネクションに関する新しいエントリを選択されたバックボーンと共にコネクション表に追加する．

2. RSTパケットやFINパケットを受け取るなどしてコネクションが解放される時には，当該コネクションに関するエントリをコネクション表から削除する．

3. それ以外のパケットを受け取った時には，このパケットがどのコネクションに属するかをコネクション表より求め，アドレス変換を行った上で，内部から外部へのパケットの場合にはコネクション表に登録されているバックボーンへ，外部から内部の場合には内部のネットワークへ中継する．

なお，上記の動作では，従来の経路情報に基づく経路制御が全く行われていないことに注意する．

3.3 実装と性能評価

3.2で述べたように，提案方式では複数のバックボーンに接続されたルータのみを用いて動的トラヒック分散を実現しているため，従来の方式に比して導入と管理が容易であり，かつ，透過的なトラヒック分散が行えることは明らかである．そこで，トラヒック分散の効率について提案方式の有効性を検証するため，動的トラヒック分散機能を持つルータ

を試作して性能評価を行った．本節では，試作ルータの実装方法と性能評価について述べる．なお，今回の実装ではコネクション確立時間をバックボーン選択基準としているので，UDPパケットは対象外とした．

3.3.1 実装法

試作ルータは，OSとしてFreeBSD 2.2.7Rを搭載したAT互換機（Gateway 2000社 GP6-450）に3枚のネットワークインタフェースを装着したものを用いた．本来，FreeBSDでは経路制御がカーネルで行われるが，試作ルータでは実装を容易にするため，カーネルを一切変更せずユーザプロセスで全ての経路制御処理を行った．このため，各バックボーンから到着するパケットは全てdivert機能を用いてカーネルが中継する前にユーザプロセスに渡されるようにした．逆に，バックボーンへのパケットの送出は，ソケットインタフェースを用いた通常の方法で送出するとカーネルで従来の経路制御が行われるため，BPF（Berkeley Packet Filter）[48]を用いてフレームを直接ネットワークインタフェースに書き出している．

3.3.2 性能評価

性能評価は，図12に示すようにクライアントとインターネットの間に試作ルータを配置し，クライアントからインターネット上のサーバと実際に通信を行うことによって実施した．試作ルータとインターネットの間は2種類のISPを経由してダイヤルアップ接続し，各モデムからISPまでのダイヤルアップ回線を対外接続用のバックボーンとみなした（以下，ISP1に接続されたバックボーンをB1，ISP2に接続されたバックボーンをB2という）．B1は56Kbpsあるいは28Kbpsの2種類の速度を切り替

えて用い，B2 は 56Kbps で固定した．この実験環境において，クライアントからインターネット上の HTTP サーバに対して同時に複数のデータ転送要求を発生させ，B1 を 56Kbps に設定して B1 と B2 でトラヒック分散を行う場合（以下，対称と呼ぶ），B1 を 28Kbps に設定して B1 と B2 でトラヒック分散を行う場合（以下，非対称と呼ぶ）の 2 種類の場合について平均伝送速度を求めた．また，提案方式のバックボーン選択基準はコネクション確立時間であるが，比較のためバックボーン選択基準をコネクション数およびラウンドロビンとした場合についても同様の計測を行った．選択基準がコネクション数の場合については，バックボーンの回線速度が不明である場合を想定し，対称および非対称のいずれにおいても B1 および B2 のコネクション数が均等になるようにバックボーン選択を行った．

なお，HTTP のデータ転送要求の発生には webjamma[49] を一部改造したものを用い，webjamma 実行時の URL リストには岡山大学総合情報処理センターに設置された HTTP 代理サーバのアクセスログから連続した 3600 個の URL を取り出して用いた．本来，このプログラムでは，親プロセスが一定数の子プロセスを生成し，各子プロセスにデータ転送要求を渡して HTTP プロトコルによりアクセスさせ，アクセスが完了すると直ちに親プロセスがその子プロセスに次のデータ転送要求を渡すような動作を行うため，常に子プロセスの数だけ並行してアクセスが発生する．しかし，このようなアクセスパターンは現実とは大きく異なるため，本実験では親プロセスでデータ転送要求の発生間隔が指数分布に従うように webjamma を改造した．転送要求の平均発生間隔については，小さすぎるとバックボーンが常に飽和状態になり，大きすぎると次のコネクション要求が発生するまでに既存のコネクションが終了してしまい，い

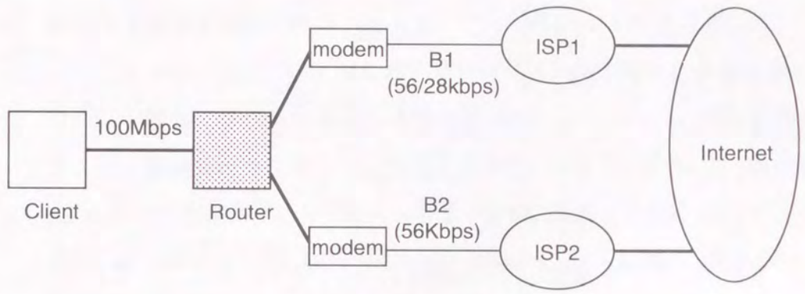


図 12: 実験環境の構成

ずれの場合も有意な計測結果を得ることができない．このため，予備実験を数回繰り返し，計測時の平均発生間隔として0.6秒，1.2秒，および，1.8秒の3種類の値を用いた．

実際の計測は，バックボーンの回線速度（対称および非対称），バックボーン選択基準（コネクション確立時間，コネクション数およびラウンドロビン），コネクション要求の平均発生間隔（0.6，1.2および1.8）のすべての組合わせ（18通り）について，ISPの利用状態が安定する夜間（午前2時から早朝にかけて）にそれぞれ2回ずつ行った．この際，いくつかのURLについては，URL自体が存在しなかったり，計測時に通信先のHTTPサーバがダウンするなどの理由により，データ転送の行えなかったURL（以下，無効なURLと呼ぶ）がいくつか存在した．しかし，無効なURLの数は，1回の計測における全URL数（3600個）の2%以下とごくわずかであったため，実験結果にはほとんど影響しないと考えられる．

実験結果として，対称および非対称の場合における平均伝送速度を表3，対称および非対称の場合において，選択基準としてコネクション確立時間，コネクション数，ラウンドロビンのそれぞれを用いたときのバックボーンの平均選択率を表4に示す．表3と表4については，無効なURLに対する結果を除いた上で算出を行った．また，非対称の場合に平均発生間隔を1.2として計測した際の各バックボーンにおけるコネクション数の時刻変化を図13～図15に示す．図13～図15については，実線（上半分）がB1，破線（下半分）がB2の時刻変化を表す．

まず，回線速度が対称の場合の実験結果について述べる．表4からわかるように，いずれの選択基準においてもB1とB2がほぼ均衡して使われているが，表3の平均伝送速度はコネクション確立時間が最も大きく，

表3: 平均伝送速度

平均発生間隔	選択基準	平均伝送速度 (Kbps) 対称	非対称
0.6	コネクション確立時間	6.4	5.0
	コネクション数	5.2	4.1
	ラウンドロビン	4.7	3.2
1.2	コネクション確立時間	13.8	9.3
	コネクション数	11.2	6.8
	ラウンドロビン	10.6	4.5
1.8	コネクション確立時間	24.7	16.0
	コネクション数	24.1	9.7
	ラウンドロビン	21.0	7.6

表 4: バックボーンの平均選択率

回線速度	選択基準	B1	B2
対称	コネクション確立時間	45.8%	54.2%
	コネクション数	46.3%	53.7%
	ラウンドロビン	50.0%	50.0%
非対称	コネクション確立時間	35.7%	64.3%
	コネクション数	41.7%	58.3%
	ラウンドロビン	50.0%	50.0%

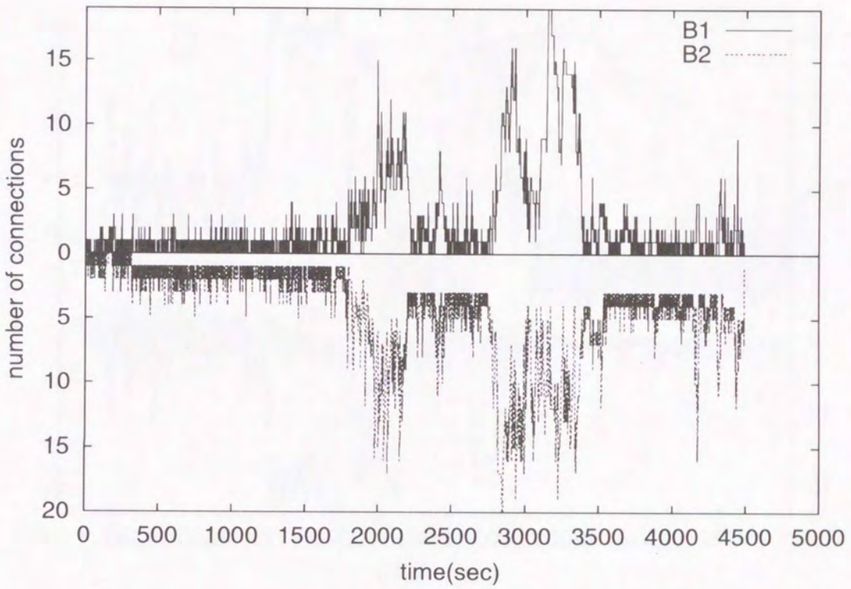


図 13: 回線速度非対称の場合におけるコネクション数の時刻変化 (選択基準をコネクション確立時間とした場合)

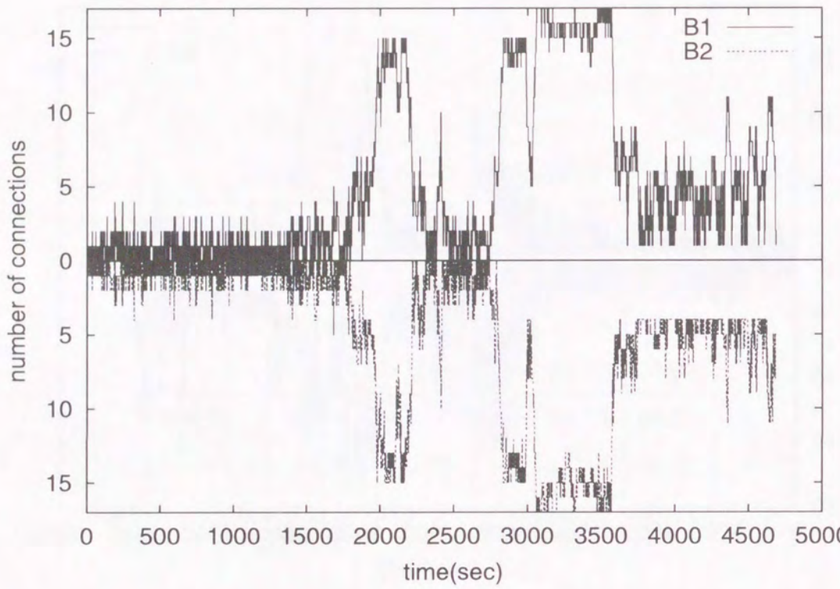


図 14: 回線速度非対称の場合におけるコネクション数の時刻変化 (選択基準をコネクション数均等とした場合)

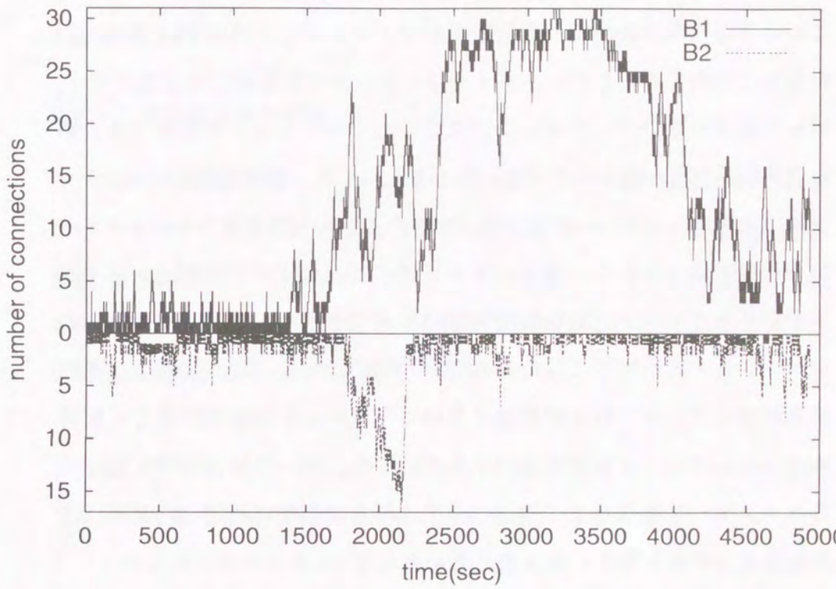


図 15: 回線速度非対称の場合におけるコネクション数の時刻変化 (選択基準をラウンドロビンとした場合)

コネクション数，ラウンドロビンの順に悪化している．B1 と B2 は同じ回線速度であるが，他の通信者間のトラヒックなどにより，計測中にも各バックボーンを経由する経路のネットワーク状態は常に変化している．このような状況において，選択基準がラウンドロビンの場合は B1 と B2 を交互に選択することから，ネットワーク状態の変化が全く反映されず，データ転送が遅いバックボーンに未完了のコネクションが集中するために平均伝送速度が最も小さくなったと考えられる．選択基準がコネクション数の場合はコネクション要求時点でコネクション数が少ないバックボーンを選択することから，ラウンドロビンに比べればデータ転送が早く完了するバックボーンをある程度有効に利用できるが，各バックボーンのコネクション数が均等に保たれるという制約がある．これに対し，選択基準がコネクション確立時間の場合はコネクションの数に関係なく応答の速いバックボーンを常に選択することから，データ転送が早く完了するバックボーンを他の 2 つの選択基準よりも積極的に利用した結果，平均伝送速度が最も大きくなったと考えられる．

次に，回線速度が非対称の場合の実験結果について述べる．表 3の平均伝送速度については対称の場合と同様であるが，非対称の場合は B1 と B2 の回線速度が異なるので，表 4から明らかなように，選択基準がコネクション確立時間の場合が回線速度の大きな B2 を選択する割合が最も高い．さらに，図 13～図 15を見ると，選択基準がコネクション確立時間の場合は回線速度の大きな B2 にコネクションが偏り，コネクション数の場合はほぼ均等にコネクション数が推移し，ラウンドロビンの場合は回線速度の小さな B1 に未完了のコネクションが集中していることがわかる．従って，回線速度が異なる場合においても，選択基準がコネクション確立時間の場合が回線速度の大きなバックボーンを最も有効に利用できていると言える．

以上の結果から，提案方式はネットワークの利用状況や回線容量に応じて適切にバックボーンを選択し，動的にトラヒックを分散していることが確認された．

3.3.3 現在の実装の問題点と対応策

今回の実験環境では，バックボーンに接続されるルータが 1 台であるため，ルータに故障が発生すると外部との通信が遮断されるという問題がある．この問題については，代替ルータを設け，ルータと代替ルータの間でコネクション表を同期させる機能を追加することにより，ルータの耐故障性を高めることができると考えられる．

また，今回の実装ではバックボーン選択基準にコネクション確立時間を採用したため，コネクションを持たない UDP パケットについては現状では負荷分散を行うことができない．しかし，これに関しては，バックボーンの状態に関わらず NAT を用いない側のバックボーン（図 9における B1）を選択し，その他のトラヒックに提案方式を適用することにより，トラヒック全体としてみれば負荷分散が実現できると考えられる．さらに，NAT では UDP パケットを送信元／送信先のアドレスとポート番号の組ごとにフローとして扱うので，例えばフロー数などを用いて適切なバックボーン選択基準を確立することができれば，トラヒック分散の仕組みについては提案方法がそのまま適用可能であると考えられる．

なお，提案方式はトラヒック分散に NAT を用いているため，NAT の利用できないアプリケーションにはそのまま適用することができない．しかし，このようなアプリケーションのトラヒックについては，ポート番号で識別することによって NAT を用いない側のバックボーンを選択すれ

ばよく，その他のトラヒックに提案方式を適用することにより，トラヒック全体としてみれば負荷分散が実現できると考えられる．

3.4 結言

本章では，マルチホームネットワークにおいて，ルータが各バックボーンの状態を自らが判断して，コネクション単位で適切なバックボーンを選択する方式について述べた．この方式により，マルチホームネットワークにおける透過的かつ効率的なトラヒック分散が高い技術レベルや運用コストを必要とせずに可能となり，マルチホームネットワークの普及に貢献できると考えられる．今後の課題としては，高速なバックボーンを用いた場合の通信先のサーバやルータの負荷も含めたより詳細な性能評価や，UDP パケットに対するバックボーン選択基準の検討も含めたより効果的なバックボーン選択方法の確立などが挙げられる．

4 結論

近年のインターネットの急激な成長と，そこで提供されるサービスおよびそれらの利用者の多様化などにより，インターネットを構成する組織ネットワークの接続点であるルータを如何にして安定にかつ効率よく運用するかが極めて重要になっている．特に，ルータの安定運用のためには，ルータで発生する障害からの早期復旧が必要であり，これを達成するための自動復旧法の開発が重要な課題となっている．また，複数のバックボーンネットワークと接続する組織ネットワーク，すなわち，マルチホームネットワークにおいては，その対外接続ルータでの効率的なトラヒック分散法の開発が重要な課題となっている．

しかし，障害からの自動復旧法については，ネットワークを自動的に監視して障害を管理者に報告するシステムは数多く存在するが，障害からの復旧作業自体を自動化するためのシステムは開発されていないため，管理者の手作業の介在によって障害が長期化するという問題がある．また，従来のマルチホームネットワークにおけるトラヒック分散法は，高い管理コスト，バックボーンの利用効率の低下，および，サービスに対する透過性の欠如という問題のいずれかを持つ．

そこで，本研究では，従来管理者が手作業で行っていた復旧作業を自動化するための方式と，マルチホームネットワークにおいて上述した問題の発生しない新たなトラヒック分散法を提案し，その有効性を確認した．

2章では，管理者が端末を通じて行う復旧作業をスクリプトとして表現可能であることを示し，管理者が作成したスクリプトを自動的に実行するための方式と，これに基づいたネットワークの障害管理支援システムを提案した．管理者は障害に対するスクリプトをあらかじめ作成してシステムに登録しておけば，同様の障害が発生した場合にはシステムが

自動的に復旧作業を行うので，繰り返し発生する障害に対しては非常に有効であると考えられる．

3章では，マルチホームネットワークにおいて複数のバックボーンと自組織のネットワークとの接続を受け持つ対外接続ルータが，TCP のコネクション確立時間を利用して適切なバックボーンを選択し，さらに，NAT を組み合わせることによって復路においても効率的なトラヒック分散が可能な方式を提案した．提案方式は対外接続ルータのみで実現されており，しかも，組織ネットワーク内の計算機がコネクションを確立する時点での各バックボーンの利用状況を考慮した選択を行うので，特別な設定や各バックボーンの管理者との連携を行う必要がない．

以上，本研究の成果をまとめたが，これらをより有効に活用するためには，本研究で提案した障害からの自動復旧法とマルチホームネットワークにおけるトラヒック分散法を組み合わせ，より効率的なルータ管理を実現する必要がある．通常，中小規模の組織ネットワークが持つ対外接続回線は一つである場合が多いので，バックボーン側のルータで障害が発生した場合には組織ネットワークが孤立し，バックボーン側の復旧作業を待つ必要があった．これに対し，マルチホームネットワークでは複数の対外接続回線を持つので，本研究で提案するトラヒック分散法を適用すれば，あるバックボーンに障害が発生した場合でも，自動的に別のバックボーンが選択されてインターネットとの接続を保つことができる．従って，組織ネットワーク内に SPLICE/NM の管理サーバを置き，バックボーン側のルータを管理対象とすれば，そのルータで障害が発生しても，自動的に他の正常なバックボーンを経由してそのルータに対する復旧作業を実施することができると考えられる．さらに，バックボーン側のルータで発生する障害は，組織ネットワークの対外接続ルータが検出することができるので，対外接続ルータが管理サーバに対して障害を通知するための仕組みを実現すれば，ネットワーク監視システムを用いたポーリング方式による障害発見に比してより迅速な障害復旧が期待できる．

謝辞

本研究について，深いご配慮のもとに，終始ご懇篤なるご教示を賜った大阪大学大学院基礎工学研究科宮原秀夫教授に心から感謝申し上げます．また，数々の有益なご助言を賜った大阪大学大学院基礎工学研究科谷口健一教授，東野輝夫教授，村田正幸教授に深く感謝申し上げます．

本研究の全過程を通じて，ご指導ご鞭撻を賜った岡山大学工学部岡本卓爾教授に深く感謝申し上げます．度々有益なご助言を頂きました岡山大学工学部杉山裕二教授に深謝致します．また，研究の詳細に至るまで熱心なご指導とご助言を頂いた奈良先端科学技術大学院大学情報科学研究科山口英教授，岡山大学工学部横平徳美助教授，岡山大学総合情報処理センター山井成良助教授に心から感謝申し上げます．

本研究の途上において，種々ご討論頂いた岡山大学工学部籠谷裕人講師，中西透助手に厚くお礼申し上げます．また，大阪大学工学部，奈良先端科学技術大学院大学情報科学研究科にてお世話になりました大阪大学大学院工学研究科西尾章治郎教授，大阪大学サイバーメディアセンター下條真司教授，奈良先端科学技術大学院大学情報科学研究科山本平一教授，九州工業大学情報工学部尾家祐二教授に厚くお礼申し上げます．さらに，本研究の途上において種々ご協力を頂いた，奈良先端科学技術大学院大学情報科学研究科情報ネットワーク講座，岡山大学工学部情報工学科電子情報工学講座の卒業生，在学生諸氏にお礼申し上げます．

参考文献

[1] J. Reynolds and R. Braden. Internet Official Protocol Standards. *RFC2700*, 2000.

[2] S. Yamamoto, K. Okayama, S. Yamaguchi, and H. Miyahara. Design and Implementation of Distributed Internet Exception Tracker. In *Proceedings of the 8th International Joint Workshop on Computer Communication*, pp. F2–3–1–F2–3–7, Taipei, Taiwan, 1993.

[3] D. Johnson. NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist. *RFC1297*, 1992.

[4] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction to Version 3 of the Internet-standard Network Management Framework. *RFC2570*, 1999.

[5] B. Wijnen, D. Harrington, and R. Presuhn. An Architecture for Describing SNMP Management Frameworks. *RFC2571*, 1999.

[6] J. Case, D. Harrington, R. Presuhn, and B. Wijnen. Message Processing and Dispatching for the Simple Network Management Protocol(SNMP). *RFC2572*, 1999.

[7] D. Levi, P. Meyer, and B. Stewart. SNMP Applications. *RFC2573*, 1999.

[8] J. Postel. Internet Control Message Protocol. *RFC0792*, 1981.

[9] S. Deering. ICMP Router Discovery Messages. *RFC1256*, 1991.

[10] K. McCloghrie. SNMPv2 Management Information Base for the Internet Protocol Using SMIv2. *RFC2011*, 1996.

[11] K. McCloghrie. SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2. *RFC2012*, 1996.

[12] K. McCloghrie. SNMPv2 Management Information Base for the User Datagram Protocol Using SMIv2. *RFC2013*, 1996.

[13] J. Hawkinson and T. Bates. Guidelines for Creation, Selection, and Registration of an Autonomous System(AS). *RFC1930*, 1996.

[14] Y. Rekhter and T. Li. A Border Gateway Protocol 4. *RFC1771*, 1995.

[15] K. Egevang and P. Francis. The IP Network Address Translator(NAT). *RFC1631*, 1994.

[16] P. Srisuresh and M. Holdrege. The IP Network Address Translator(NAT) Terminology and Considerations. *RFC2663*, 1999.

[17] 梶田将司, 結縁祥治. NAT によるプライベートネットワークの準マルチホーム化技法. 情報処理学会分散システム/インターネット運用技術研究報告, No. 16, pp. 73–78, 1999.

[18] 中川郁夫, 上谷一, 鍋島公章, 樋地正浩, 今野幸典. マルチホーム環境におけるアプリケーションルーティング技術の提案. 情報処理学会分散システム運用技術研究報告, No. 12, pp. 37–42, 1998.

[19] 小巻賢二郎, 所真理雄. フローを考慮した経路制御機構. 電子情報通信学会技術研究報告 IN98-27, pp. 25–32, 1998.

[20] Marshall T. Rose. *The Simple Book*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[21] L. Wall and R. Schwartz. *Programming Perl*. O'Reilly & Associates, Sebastopol, CA, 1990.

[22] Y. Kadobayashi, S. Yamaguchi, and H. Miyahara. WWFS: a Framework for Distributing Information in the Internet Environment. In *Proceedings of the 7th IEEE Region 10 International Conference*, pp. 227–231, 1992.

[23] J. Steiner, C. Neuman, and J. Schiller. Kerberos: an Authentication Service for Open Network Systems. In *Proceedings of the USENIX 1988 Winter Conference*, pp. 191–202, Dallas, USA, 1988.

[24] S. Yamaguchi, K. Okayama, and H. Miyahara. The Design and Implementation of an Authentication System for the Wide Area Distributed Environment. *IEICE Trans. Inf. & Syst.*, Vol. E74, No. 11, pp. 3902–3909, 1991.

[25] 濱口伸, 岡山聖彦, 山口英, 尾家祐二. スケーラビリティを考慮したインターネット環境における個人認証システムの開発. 情処学 DPS 研報 98-DPS-87, pp. 233–238, 1998.

[26] C. Adams and S. Farrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. *RFC2510*, 1999.

[27] M. Myers, C. Adams, D. Solo, and D. Kemp. Internet X.509 Certificate Request Message Format. *RFC2511*, 1999.

[28] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC2401*, 1998.

[29] S. Kent and R. Atkinson. IP Authentication Header. *RFC2402*, 1998.

[30] C. Madson and R. Glenn. The Use of HMAC-MD5-96 within ESP and AH. *RFC2403*, 1998.

[31] C. Madson and R. Glenn. The Use of HMAC-SHA-1-96 within ESP and AH. *RFC2404*, 1998.

[32] C. Madson and N. Doraswamy. The ESP DES-CBC Cipher Algorithm With Explicit IV. *RFC2405*, 1998.

[33] S. Kent and R. Atkinson. IP Encapsulating Security Payload(ESP). *RFC2406*, 1998.

[34] D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. *RFC2407*, 1998.

[35] M. Schneider D. Maughan, M. Schertler and J. Turner. Internet Security Association and Key Management Protocol(ISAKMP). *RFC2408*, 1998.

[36] D. Harkins and D. Carrel. The Internet Key Exchange(IKE). *RFC2409*, 1998.

[37] R. Glenn and S. Kent. The NULL Encryption Algorithm and Its Use with IPsec. *RFC2410*, 1998.

[38] T. Dierks and C. Allen. The TLS Protocol Version 1.0. *RFC2246*, 1999.

[39] T. Tsutsumi and S. Yamaguchi. SecureTCP — Providing Security Functions in TCP Layer. In *Proceedings of the INET'95*, No. T3, pp. 905–913, Honolulu, USA, 1995.

[40] 泉裕, 山本茂, 山口英, 山本平一. LAN 環境における Trouble Tracking Ticket System(T3) の構築. 情処学 DPS 研報 95-DPS-68, pp. 1–6, 1995.

[41] 中川郁夫, 上谷一, 鍋島公章, 樋地正浩, 今野幸典. マルチホーム環境におけるアプリケーションルーティング技術の提案. 情報処理学会分散システム運用技術研究報告, pp. 37–42, 1998.

[42] T. Brisco. DNS Support for Load Balancing. *RFC1794*, 1995.

[43] 青木武司, 菊池慎司, 高橋英一, 岡野哲也, 安達基光, 勝山恒男. IP ネットワークの性能測定技術. 電子情報通信学会技術研究報告 IN98-90, pp. 9–16, 1998.

[44] R. L. Carter and M. E. Crovella. Measuring Bottleneck Line Speed in Packet-Switched Networks. *Tech. Rep. of Boston University*, No. BU-CS-96-006, 1996.

[45] V. Jacobson. pathchar — A Tool to Infer Charasteristics of Internet Paths, 1997. ftp://ftp.ee.lbl.gov/pathchar/msri-talk.pdf.

[46] M. Mathis and J. Mahdavi. Diagnosing Internet Congestion with a Transport Layer Performance Tool. In *Proceedings of INET'96*, 1996.

[47] Cisco Systems Inc. LocalDirector in the Data Center. http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/ldir_wp.htm.

[48] S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-Level Packet Capture. In *Proceedings of the 1993 Winter USENIX Conference*, pp. 259–269, 1993.

[49] R. P. Wooster and M. Abrams. Proxy Caching that Estimates Page Load Delays. In *Proceedings of the 6th World Wide Web Conference*, pp. 325–334, 1997.