第 11 部

ネットワークマネージメント

第 1 章

序論

1.1 はじめに

計算機の進歩にともないコンピュータネットワークが脚光を浴びてきた。これはそもそも大型計算機ほどの処理能力はないが、システムを分散させて持ち、それぞれのシステムを各々がアクセスするという、ワークステーションの普及とともに生まれてきたものである。ワークステーションは処理速度の速いマシンとして利用するに当たっては、ユーザの要求を満たす場合が多い。ワークステーションを単なる処理速度の速いパソコンとして用いるとすると、これはそのユーザにとっては、大変便利でかつ効率的なマシンとなる（但し、価格の面を考えると個人購入は非常に難しい）。しかし、ワークステーションはパソコンではない。ワークステーションはシステムを分散させて持ち、それを共有して使うことをサポートしたマシンである。例えば、あるマシン上で JSERVER を走らせて、他のマシンはその JSERVER をアクセスする、といった具合である。従って、ワークステーションの利用を考えた時は、ネットワークを無視して考えることができなくなるのである。すなわち、ネットワークを介しての利用ができるという意味で、ワークステーションの価値があるとも言える。

さて、そのネットワークも、時代とともにどんどん広がっている。もともとは自分達で使うシステムを一つのネットワーク上に分散させて利用するという LAN(Local Area Network) が中心であった。それが、各々の LAN をつないで、さらに大きなネットワークができるようになった。そうすると、さらに多くのシステムを利用できるという点からは、ワークステーションをより有効に活用しているということになるが、逆に一つのシステムをより多くの人から使われているということにもなる。そのことが問題発生の原因となることもある。例えば、各々が同じシステムをアクセスすればそのシステムを載せているマシンの負荷は、自ずと高まる。また、様々なシステムを皆がアクセスしにいけば今度は網自体の負荷も上がってくる。これらの問題は LAN が中心の世界では、ユーザの数もおおよそ予

測ができたり、網としても効率の良いイーサネット等を用いていたことで、あまり気になることはなかった。ところが、現在では LAN をつなぎ合わせて大きなネットワークを作り、そのネットワークを一つのポリシによって管理し、そして、そのポリシを持ったネットワークをつなぎ合わせている状態である。そしてそのネットワークは、国家間をまたいでものアクセスが可能となるバックボーンを構築するまでに至っている。

このような広域のネットワークでは、網の種類も様々で、かつ経路に関しても一通りとは限らない。こうなると今までとは異なる新たな問題が生じる。例えば、どこかの網が切れてしまったとか、途中のどこかのゲートウェイが落ちてしまって突然遅い経路を通ることになってしまったという問題がある（但し、これらの問題は自分の組織内でよりも、自分の所属する組織を越えた相手と通信をする時、さらには他の組織を介してより遠くの組織と通信するような場合に特に問題となる）。また、組織の管理者にとって通信可能な他の組織の表玄関、すなわち外とつながっているゲートウェイの情報を認識していることは、自分の組織外で起きたことに対処する時に大変役に立つ。

さて、これらの問題を認識し、さらにはより良い方法で対処することは、今後のネットワーク運営にとって貢献すると考えられる。実際の通信はある経路を通って行なわれ、またどこかで起きた問題も経路情報を通して知ることが多い。しかし、現在の経路制御では、自分のマシンが、“どこかの経路が切れた”、“マシンが落ちて通信不可能になった”、ということを認識するまでには時間がかかる。そこで、本論文では、まず最初に自分からデスティネーションまでの経由したゲートウェイの情報を持つ。それから、そのデスティネーションが生きているかどうかの応答要求を出し続ける。そこで、もしその応答要求が返ってこなくなったらどこが切れて、あるいはどのゲートウェイが落ちて通信できなくなったのかを調べる。そのために途中の全てのゲートウェイに応答要求を出す。その結果どこの経路が切れたかを突き止める。そして、その故障箇所を知らせる。以上のことをする経路診断システムについて述べる。これにより、経路状態の遷移の早い認識を実現した。

1.2 背景

1.2.1 研究対象としてのネットワーク

本研究で対象とするネットワークは広域分散環境ネットワークである。広域分散環境ネットワークとは、autonomous system(自治システム) をつなぎ合わせたネットワークのことをいう。LAN をつなぎ合わせ大きなネット

ワークを作り、そのネットワークをあるポリシによって運営しているネットワークを autonomous system と呼ぶ。例えば慶応大学は一つの autonomous system と考えることができる。そのような autonomous system 内のネットワークや、LAN を対象にするわけでもない。ここでは、autonomous system 間を結んだネットワークを研究対象に考える。

autonomous system 内ネットワークの場合、管理者がそのネットワークの構成を理解していて、問題が起きた時にそれを調べて修復することができる。しかし、広域分散環境の場合、実際どこで問題が起きているのかを検知することが困難であり、さらに物理的にも遠隔地で存在しているので修復も困難である。

従って、ネットワークを介した問題が重要になってくる。つまり、自分で直すことができなかったり、自分でログインできなかったり、そういう状況で問題が起きた時も、管理者はできる限りの環境をエンドユーザに提供しなければならない。だからこそ、それぞれのネットワークの状況を知っておく必要がある。

1.2.2　ネットワーク管理の前提

ネットワークを運用する管理者の義務は、エンドユーザにとってネットワークを全く意識せずにネットワークを利用できるような環境を提供することである。どこかで問題が起きたということなど、エンドユーザはまったく気にしないでネットワークを利用できるということである。つまり、広域のネットワークを運用する場合に出てくる問題に速やかに対処して、問題を解決することとなる。広域ネットワークを運用する上で起こりうる問題には、

- どこかのゲートウェイが落ちた、
- どこかの網が切れた、

などがある。これらの問題が生じても、ネットワークを円滑に利用できるような環境を提供しなければならない。ユーザがネットワークを意識せずネットワークを利用できるようにシステム的に支えることがこのような広域ネットワークの理想的な運用である。

更に、その条件の下でできる限りの環境を提供しなければならない。広域ネットワークは状況毎に提供し得る環境が全く異なってくることもあり、もし提供できる環境が低性能なものだとしたらそのことをユーザに知らせなければならない。また、その状況下で最大の環境を提供しなければならない義務もある。つまり、異なった状況毎にそれに応じた機能が提供でき

なければならない。そうなってくると、ネットワーク管理者は自分の管理するネットワークだけなく、自分のネットワークとつながっているネットワーク、すなわち手の届かないネットワークの状態も常に注意して見ていなければならない。それは、直接つながっているネットワークだけでなく、ネットワークを介して到達可能なネットワークすべてである。

ユーザにその時の条件で実現できる最大の環境を提供することがネットワーク管理の最終目標である。ネットワーク管理者はこのことを常に意識してネットワークを運営する必要がある。

1.3 本研究の目的

一口にコンピュータネットワークといっても、その中で持ち上がる問題は様々ある。そこで、今それらを以下のように大別してみる。

- アプリケーション等の、ネットワークを利用する上の技術的な問題、
- 通信をささえるネットワーク管理の問題、
- ネットワークを作り上げている様々な網自体の問題、

本論文では、その数ある問題の中から、ネットワークが円滑に運営されかつ使用されるために、ネットワークの管理的側面から生じる問題を取り上げた。特に、経路情報に注目し、動的に変化する経路を動的に監視するシステムを提案する。

現在の経路は管理領域内、すなわち autonomous system 内だけを考えても、現在利用されている経路を調べることはできるが、どんな経路が利用可能なものとして潜在しているのかはわからない。もちろん、管理者は自分の autonomous system 内のネットワークの状態を把握し、autonomous system 内のネットワークの安定を保たなければならない。しかし、問題が生じた場合、それがどこで生じたのか、どのような問題が生じているのかを調べなければならないのである。autonomous system 内では、そのネットワークについての状態を把握しているという理由からその原因を突き止めやすい。ところが、それが広域なネットワークになるとその原因の究明が困難になる。そこで、その原因究明、及び、ネットワークの状態保持のために経路診断システムが必要なのである。

ユーザの立場から見るとどのような経路を通ろうとも、正しく通信できれば良いかも知れない。しかし、そのような通信サービスを提供するには、現在の経路、或は経路が変わってしまった原因を、そのサービスを提供する側が認識していなければならい。また、今後予想されるユーザ側か

らの要求として、自分自身で指定した経路を用いて通信したいという要求が生じるかも知れない。その時も、その要求に従った経路制御が必要になると考えられる。

更に、現在の経路制御は RIP [19] というプロトコルを中心に行なわれていることに注目する。このプロトコルを用いた経路制御を行なうと、実装上覆いきれないものがある。これは、経路制御の中でも経路が切れたという情報について、スポットを当てた場合である。RIP はその実装上、経路についての不利な情報はゆっくり伝播するという特性を持っている。しかし、ネットワークが広域化し、複雑化すると、ある経路は絶対切れては困るといった要求も出てくる。この時、RIP による情報の伝達をまって、それから対処するのでは遅過ぎることになる。経路が切れたら直ちに突き止めて修復しなければならないかも知れないし、バックアップ経路があればすぐつながなければならないかも知れない。未然に防げるものであれば防がなければならない。従って、ここで必要となるものは RIP による順次的な情報ではなく、即時調べるような積極的な検索によって獲得される情報である。これによって、事態の悪化を防ぐことができるかも知れないし、状態の早い立て直しにも役に立つかも知れない。

以上の理由から経路情報を認識すること、更には経路が変更された原因までを突き止められることが必要といえる。そこで、本論文では現存するプロトコルからどのような情報が得られるのか、それを利用した経路制御の現状、及び積極的な検索による故障部分の発見を含んだ、経路診断システムの設計と評価について述べる。

1.4　本論文の構成

本論文の構成は、第 2 章で現状のコンピュータネットワークと、ネットワークマネージメントについて述べる。現在のネットワークの状態や、現在用いられている経路制御のためのプロトコル、更には現存するマネージメントシステム、そして、それらの基に生じる問題点について述べる。第 3 章では、問題点に対するアプローチを述べる。第 4 章で、システムの設計について説明する。第 5 章でそのシステムの実装、及び評価を行ない、第 6 章で本研究の結論と、今後の課題について述べる。

第 2 章

コンピュータネットワークとネットワークマネージメント

ワークステーションの発展にともないコンピュータネットワークが盛んに論じられるようになってきた。コンピュータの世界にネットワークという新しい技術が導入され、そのことによって、エンドユーザ側、管理者側、等から様々な要求がなされるようになった。その要求に答えながらネットワークを運営していかなければならない。

ネットワークを運営するという問題を考えていく上で、ネットワークマネージメントという概念が生まれた。これは、広域ネットワークを円滑に運営するために人間の手によって行使している部分のバックアップを、少しでもコンピュータにサポートさせようとするものである。今後、更にネットワークが複雑化され、広域化されると、ユーザにとってはネットワークの利用の範囲が広がりとても便利になる。しかし、それだけネットワークユーザが多様化することに備えるため、管理者はより細かくネットワークを管理し、運営していかなければならなくなる。

この章では、まず現在の広域分散ネットワークについて、“今はどのような構図でできているか”、“管理の単位は何か”、“管理のあり方はどうか”について述べる。次に、ネットワークを管理するためのシステムとして、現在どのようなものがあるのかについて述べる。

2.1 現状のネットワーク

現状のネットワークはイーサネットを用いた LAN 中心のネットワークから、それらが複数つながった複雑なネットワークになっている。この複雑なネットワークは、一つのポリシに基づいて管理されている。このように、複雑につなぎ合わされたネットワークが、ある一つのポリシによって管理されている組織を、autonomous system (自治システム) と呼ぶ。更に、現在のネットワークをそのような autonomous system どうしを結び

付けて、より広域なネットワークが築き上げられている。現在広域なネットワークは、日本各地から、更には、国家間を結んだネットワークにまで至っている。

2.1.1　WIDE 広域分散環境

現在の広域分散環境は図 (**Figure 2.1**) のようになっている。現在の広域環境ネットワークの管理は、それぞれの autonomous system の管理者が広域ネットワークのよりよい利用環境を提供できるように、協力しながら思考錯誤で行なっている。その第一歩として、まずは確かなバックボーン作りをしている。なぜなら、日本における広域ネットワーク環境というものは後から作られたものだからである。

アメリカでは ARPAnet [1] が中心になって、コンピュータネットワークの要求、あるいは利用を予想してネットワークが構築された。従って、その発展のしかたは、まずバックボーンとなるべきネットワークがあって、そのネットワークを中心に広域ネットワークを構築した形になっている。ところが、日本の場合はそれぞれのネットワークが存在して、その上に、それらをつなぎ合わせる広域のネットワークを構築するという形になっている。このことによって、逆に現在のネットワークのような複雑な形になったともいえる。

また、実際につながっている組織間を接続する網の能力に違いがある。更に、モデムを利用したダイアルアップによるアクセスや、ISDN [91] といった新しい媒体の加入を考えると、今後更に複雑なネットワーク様相を呈していくと予想される。

この広域ネットワークを運営していく上で、様々な要求、問題が生じる。例えば、

- 遅延、
- 輻輳、
- バンド幅、
- 優先権、
- 到達可能性、
- 信頼性、

などがある。これらの問題は、やはりネットワークの広域化によってクローズアップされている。なぜなら、LAN を中心としたネットワークでは、意

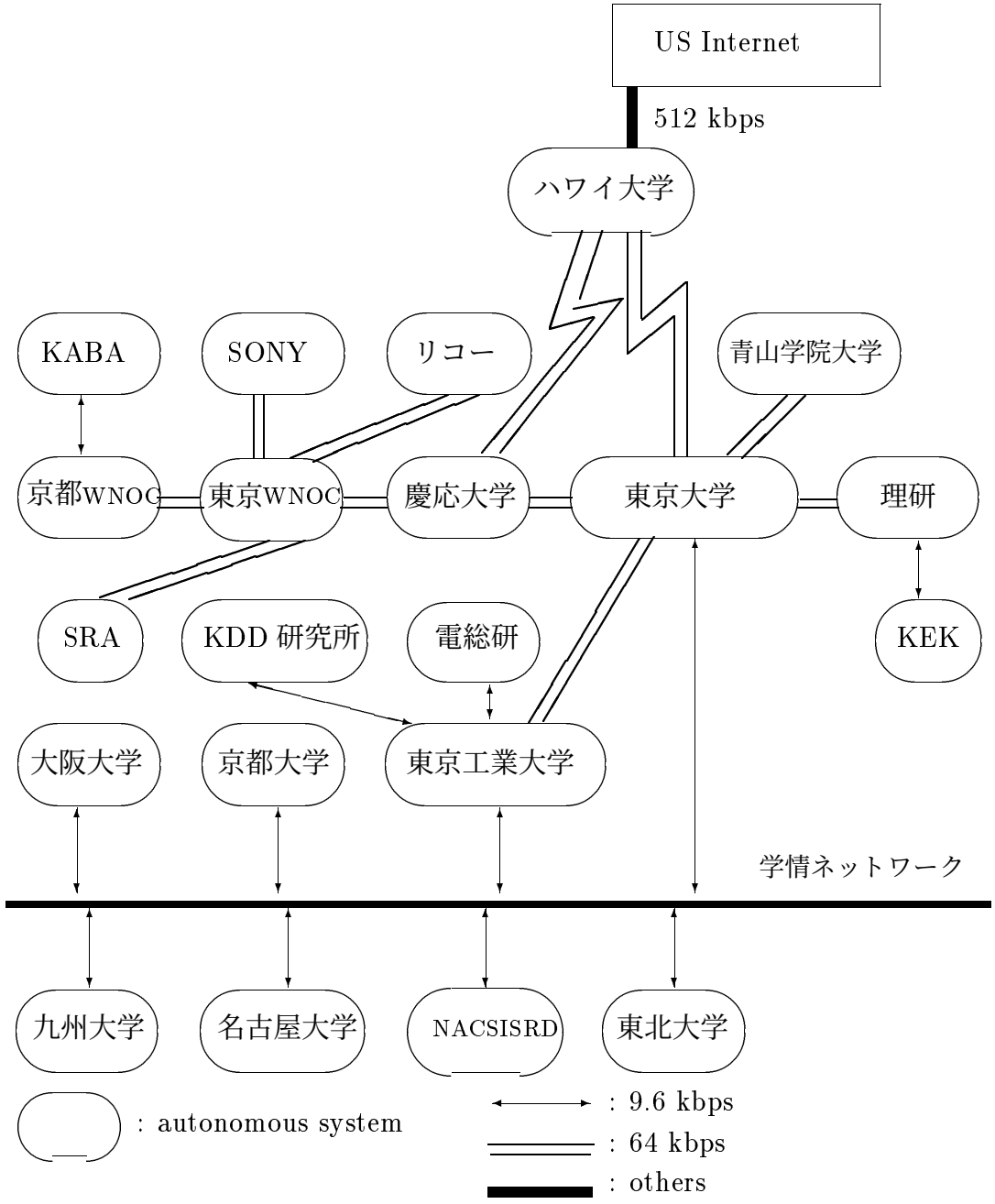


図 2.1: WIDE 広域分散環境

識する必要のない問題も含まれているからである。広域ネットワークをより便利かつ、有益なものにするには、これらの問題を解決しなければならない。

2.1.2　autonomous system(自治システム)

autonomous system とは、イーサネットによって結ばれたネットワークがいくつもつなぎ合わされてできた複雑なネットワークが、ある一つのポリシによって管理されている組織のことである。ここで、もう少し具体的に定義すると、autonomous system とは、経路制御の目的のために、単一の管理権限により制御されるネットワークとゲートウェイの集合のことをいう。従って、例えば、日本国内を考えてみれば、各大学 (慶応、東大、東工大、等) とか、一つの会社 (あるいは、その会社の中の研究所) 等が、その autonomous system と考えられる。その autonomous system 間を様々なポリシを反映させながら、つなぎ合わせることによって現在の広域分散環境が築き上げられている。

では、なぜ autonomous system という概念が必要かというと、それは各々ポリシの違った組織をつなぎ、その出来上がったネットワークを円滑に運営していくことを考えた時に必要となる。つまり、各々の autonomous system はあるポリシによって管理されていて、ネットワークを運営する側は、その autonomous system 間のネットワークの状態を保持できれば良い、ということになる。すなわち、ネットワークを階層的に管理できるのである (**Figure 2.2**)。この、ネットワークの階層的な管理を考えた時、autonomous system の概念が必要になる。

さて、現在の広域ネットワークを autonomous system の点から見ると、autonomous system の内のネットワークと、外のネットワークとに分けて考えられる。すると上述の WIDE 広域分散環境 [29] というものは “外” のネットワークである。内側は、単一の管理権限によって経路制御が行なわれているのに対し、外側は、それぞれの管理権限の集合によって経路制御がなされることになる。外側が様々な媒体によってつながっていて、その媒体によっても性能が異なる。

このように、autonomous system というのは一つのポリシによって管理されているのである。このポリシが、autonomous system 外のネットワークと通信を行なう時の、自分のネットワークを表わす大事な要因ともなるのである。

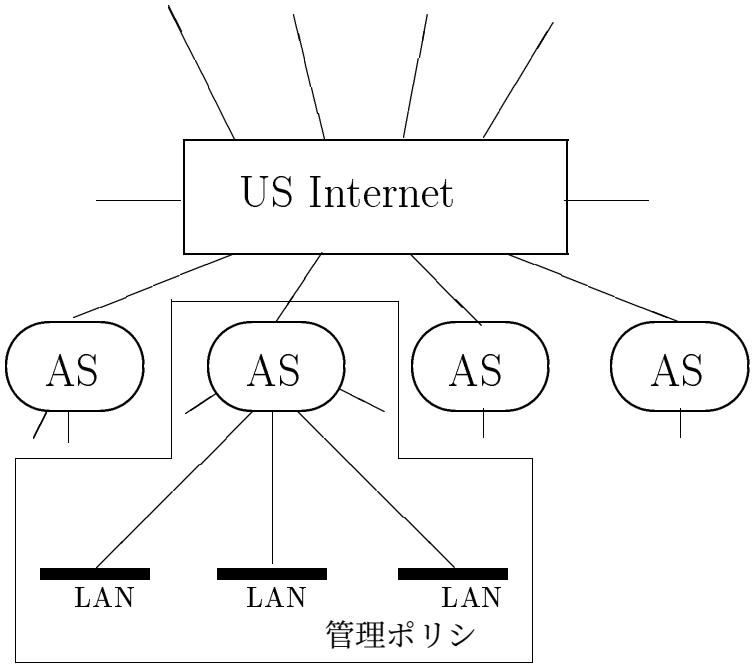


図 2.2: ネットワーク管理

2.1.3　管理ポリシ

広域分散環境に着目した場合、その autonomous system の管理ポリシがネットワークに大きな影響を与える。ここでは、管理ポリシとは何か、それがネットワークにどのように反映されているかを述べる。

管理ポリシは具体的に、

- autonomous system 内の情報で外に知らせるものに制限をつける、
- autonomous system 内の管理はその外とは独立に行なう、
- 送り出すパケットに優先権をつける、
- 入ってくるパケットに制限をつける、

等の自分の入ってくる、或は出ていく情報にその autonomous system の考え方を盛り込んだものである。また、このような政治的な要求だけでなく、性能的な要求もある。例えば、

- 速い通信が可能なら通信したい、
- 混んでなければ通信したい、

といったマシンや網自体に依存した要求である。これらも、体外的な要求と考えれば、その autonomous system のポリシの一つと考えることもできる。

このポリシは経路決定のシステムに反映されている。例えば、外に向かってアナウンスするとか、入ってくる情報に制限をつける、というのは gated [18] というシステムによって反映されている。また、性能的な要求をサポートするシステムはまだないが、結局それらが直接影響するものは経路決定である。つまり、ポリシを行使しながらネットワークを運営していき、かつネットワークを効率的に保持するために、経路決定に依存した要求を満たしていかなければならない。

以上のように、ポリシと経路が密接に関連している。

2.2　ネットワークマネージメント

現在の広域ネットワークを autonomous system のつながりと見ると、各 autonomous system 内ではその組織のポリシに基づいた管理がなされる必要がある。しかし、現状は、管理者が自分の “手” で作業をしなければならない。その管理をマシンに依存したシステムとして実現したものがネッ

トワークマネージメントシステムである。ネットワークが広域化していくことにより、自治システムのポリシに従った管理が必要となる。また、広域化することにより、エンドユーザ、或は、管理者からも様々な要求が出るようになった。例えば、

- ネットワークの安定性、
- 通信の高速性、反応性、
- 各 autonomous system の機密保護、
- 到達可能性、
- ネットワークの負荷 (網の負荷、マシンの負荷)、

などである。これらの、ポリシや、要求を一つ一つ管理者の手によって応えていたのではもう対応し切れなくなってきている。そこで、ネットワーク管理のためのシステムが必要なのである。

しかし、マネージメントシステムの開発はまだまだ発展途上で、十分な供給にまでは至っていない。既存のシステムはまだ広く利用されておらず、利用も複雑である。基本的には今ある情報を管理して、要求に応じてその情報を伝播するというアルゴリズムである。ネットワークの状態を自分から積極的に調べに行くわけではない。その意味では、まだ問題が残っている。しかし、マネージメントシステムに目を向け、管理に必要な情報をデータベースとして保持し、要求に答えるというだけでも、緊急時以外でのネットワーク管理には非常に役に立つ。本論文では、“できるだけ緊急” ということにも目を向け、経路情報に限ってはいるが、その要求に答えるシステムを構築する。

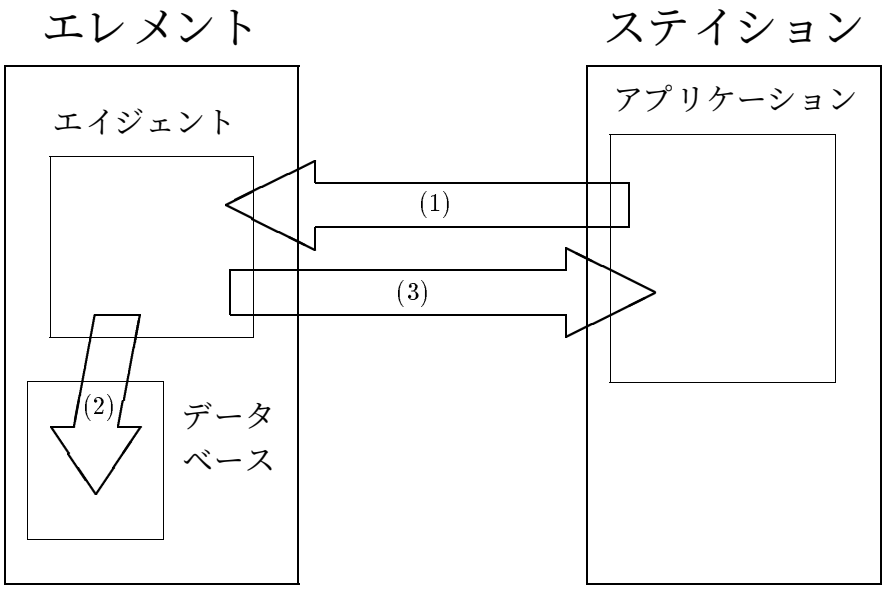
2.2.1 現存するマネージメントシステム

ここでは、SNMP [25] を中心に、更にその前任者である SGMP [26] をあげる。どちらも、基本的には同じアルゴリズムで作られている。あるゲートウェイ、あるいはホスト上にデーモンをおいて自分のデータベースを持つ、そして、権利のあるゲートウェイあるいはホストがその情報を得る。それによって、知りたい情報を伝播しようというアルゴリズムである。

(1) SNMP(Simple Network Management Protocol)

SNMP アーキテクチャモデルはネットワークマネージメントステイションとネットワークエレメントの集まりからなっている (**Figure 2.3**)。一つ

のエレメントに対し、複数のステイションがエレメントの中の情報をアクセスしにいっている。



(1) 要求
(2) 検索
(3) 応答

図 2.3: SNMP アーキテクチャ

- ネットワークマネージメントステイション ··· エレメントを監視したり制御したりするマネージメントアプリケーションを実行する。
- ネットワークエレメント ··· ホスト、ゲートウェイ、ターミナルサーバ、等その他同種類のもので、マネージメントステイションからの要求によってマネージメント機能を実行するよう、その責任を負わされたエイジェントを持ったデバイスのこと。
- エイジェント ··· 実際に SNMP を実装する snmpd が動いていて、自分のデータベースを獲得、保持して、ステイションの要求に応じてそのデータベースを伝播する。

SNMP はこのマネージメントステイションとエイジェントの間のマネージメントインフォメーションを伝達するために用いられているプロトコルであ

る。マネージメントインフォメーションの伝達はプロトコルメッセージの交換によって行なわれる。エイジェントは、MIB(Management Information Base) [30] によって定義された変数を持っており、この変数はマネージメント情報をオブジェクトの形で持っているものである。この変数がステイションに伝達されSNMPアプリケーションによって表現される。尚、この変数はフリーアクセスではなく、コミュニティ名というものによって、そのアクセスを制限している。

- **MIB(Management Information Base)**

  管理に必要な情報を ASN.1 言語 [57] に従って定義づけしたものである。そのオブジェクトのリストを定義したもの。組織的な関心事や管理的なポリシに大いに関係していて、管理されているオブジェクトを特定している。例えば、経路情報のための変数としては、Interface Table や、IP Routing Table という変数のグループがある。また、それぞれの変数は、単一の変数の場合や、複数の変数のエントリを持った変数の場合もある。

- **SMI(Structures and Identification of Management Information)**

  SMI [36] とは、MIBで定義されたデータを構造化して持っているものである。つまり、データを階層化された、変数木を構成している。このことによって、変数の意味付けがされ、ある変数がどこの部分に管理されているかが分かる。また、ある変数をアクセスしたいときは、この変数木の構造に基づいてなされなければならない。しかし、この管理された変数木が非常に多種多様であることから、変数をアクセスする際の複雑さが生じている。

(2) SGMP(Simple Gateway Monitering Protocol)

基本的な考え方はSNMPと同じである。情報を変数として持っておき、その変数をアクセスすることによって情報が伝播できるものである。もともと、急場の要求を満たすために作られたもので、将来的にもっと汎用的なものが現れることは予想されていた。

変数の管理はCCITTのX.409言語 [11] に基づいてなされている。欠点はプロトコル実装上の複雑性が増すこと。利点は、プロトコルに従ったデータの表現がマシンに対して独立しているので、広域のネットワークに対応しやすいこと。

SNMP と同様ゲートウェイにかかる負担を最小限にしようという目標を持って実装されてはいるが、共にその負荷はまだまだ大きく、これからこれらのマネージメントシステムはかなり発展していくことが予想される。

第 3 章

経路制御

マネージメントシステムを経路制御だけに注目して扱う。そこで、経路制御の現状を把握するために、この章で、

1. 現存する経路制御プロトコル、
2. 経路制御の問題点、
3. 問題点に対するアプローチ、

の順に述べる。まず、現存するプロトコルを列挙する。そのプロトコルには利点もあり、欠点もある。例えば、誤情報を避けるために悪い情報はその情報の真偽性を確認しながら伝播するということと、悪くなる可能性があるなら真偽に関わらずすぐに知りたいという概念は相反する。しかし、これは一つのプロトコルによる実現はできないのである。この原因はプロトコルの性能が悪いからではない。ローカルネットワークのために作られたプロトコルを、広域ネットワークにも用いているために生じた問題とも考えられる。このような矛盾を解決するためのアプローチを最後に述べている。

3.1 現存する経路制御プロトコル

ここで、実際にシステムを設計する上での足掛かりとなる、現在の経路制御に関係したプロトコルを紹介する。実際に経路に関する情報を提供したり更新したりするプロトコル (あるいは、プログラム自身) として以下のようなものがある。

(1) RIP(Routing Information Protocol)

RIP とは、routed [80] と呼ばれるプログラムで実装されている。もともとカリフォルニア大学バークレイ校でローカルネットワーク上のマシン

間で矛盾のないルーティングと到達可能性の情報を提供するために設計された。

これは、各デスティネーションをゲートウェイのホップカウントで計った距離にしたがった経路情報を伝達する。普通 routed はルーティング情報のパケットを待っている状態になっている。ネットワーク間のルータになっているマシンは、定期的に自分のルーティングテーブルのコピーを直接接続しているホストあるいはネットワークに送る。routed によって受け取ったパケットがルーティングテーブルの更新に利用されるのは次の場合である。

1. ホップカウントが 15 以下で (16 以上は無限大と定義されている)、そのデスティネーションのエントリがルーティングテーブルの中に存在しなかった場合。

2. 送られてきたメッセージが直接接続しているゲートウェイ (ルータ) からのものである場合。

3. ルーティングテーブルのエントリが 90 秒間更新されておらず、更にその経路の方が現在の経路より少なくともコスト的に効率的な場合 (つまり、ホップカウントが同じか小さい場合)。

4. デスティネーションに対する経路がホップカウントをメトリックスとして、今の経路より短い場合。

更新されると、routed は自分のマシンのルーティングテーブルを書き換え、直接接続しているホストあるいはネットワークにパケットを送る。routed はルーティングテーブルを変更する前に、その情報の安定さを確かめるために 30 秒程度待つ。また、routed はルーティングテーブルのエントリが 3 分間更新されてないと、そのエントリのメトリックスを無限大を表す 16 にセットし、そのエントリを抹消する。抹消は、そのエントリの無効がインタネットに伝播するのを保証するためにさらに 60 秒遅らせて行なう。

このように直接ルーティングテーブルを書き換えられるので、現在の経路制御に関して、最も影響を与えている。

(2) HELLO Protocol

RIP での経路制御方法は、最小のホップカウントを基に経路を計算するという方法である。ホップカウントではネットワークの応答の生の測定や最適形を作り出せない。従って、ネットワークの負荷に応じて経路を変

化させることができない。結果として、経路制御を相対的に静的なものにしている。

そこで、HELLO Protocol [37] は、デスティネーションに対するメトリックスとして、ホップカウントの代わりに遅延を基に距離を計る基準を用いたプロトコルである。

HELLO Protocol は以下の基本的な二つの機能を提供する。

1. 経路のトポロジの中でクロックを同期させ、各マシンがデスティネーションへの最短遅延経路を計算する、
2. RIP と同様に、新しい経路を計算するすることにより、経路情報を更新できる、

最短遅延経路を計算するためには、HELLO メッセージの交換に関係するマシンは近接するマシンでのクロックの最良と思われるデータを管理することが必要である。また、経路情報の更新に関しては、RIP と同様 HELLO Protocol 用のルーティングテーブル、つまり遅延時間によって定められたルーティングテーブルを伝播し合うことによって行なわれる。しかし、そこにはまた RIP と同じ様に、経路更新に関しては、良い更新は即座になされ、良くない更新は冗長的に行なわれるという問題は残っている。

しかし、この HELLO Protocol は現在、ほとんどサポートされていない。

(3) EGP(Exterier Gateway Protocol)

他の autonomous system につながっているゲートウェイどうしが自分の autonomous system の到達可能性情報を知らせるために使われるのが EGP [38] である。

EGP は以下の三つの特徴を持っている。

1. 二つのゲートウェイが到達可能性の情報を通信するべきであることを合意するための neighbor acquisition 機構を保持している。
2. 合意したゲートウェイに対し、継続的に応答しているかどうかを試験する。
3. ルーティング更新メッセージを渡すことにより、ネットワークの到達可能性情報を定期的に交換する。

この neighbor acquisition 機構によって合意を得ると、ゲートウェイは他のゲートウェイにルーティング情報の交換を要求することができる。

しかし、このプロトコルもほとんどサポートされていない。

(4) GGP(Gateway-to-Gateway Protocol)

コアゲートウェイシステムという、全てのデスティネーションに対して信頼性のある矛盾のない経路を提供するよう設計されたシステムがある。このコアゲートウェイシステム内でルーティング情報を交換するためのプロトコルが GGP [58] である。

GGP によってゲートウェイが交換するルーティング情報は、(N,D) という組から構成されている。N はネットワークで、D はそのネットワークに到達するためのコストを与える距離 (distance) である。距離を比べることによって、ネットワークに対する最短経路を計算する。また、経路の更新が行なわれると、そのコストはシステム内の全てに伝播される。距離は通常ゲートウェイホップ、即ちホップカウントを用いる。

このプロトコルも、ほとんどサポートされてない。

(5) gated

コーネル大学で書かれた、autonomous system 外のゲートウェイへの経路情報をどのように知らせるかを制限した規則の集合を持った RIP と HELLO と、EGP を組み合わせたものである。

gated は RIP や HELLO のメッセージを受け付け、ローカルマシンのルーティングテーブルを routed のように変更する。autonomous system 内からの経路を EGP を使って外のゲートウェイに知らせる。これより、gated の重要性は、内側のルーティング情報と外側へのルーティング情報を自動的に結び付けることが可能であることである。

実際は HELLO も、EGP もサポートされていないので、現在は外側に対しても、routed を用いている。

3.2　経路制御の問題点

経路制御上の問題点は autonomous system 内、autonomous system 外に分けて考えるべきである。しかし、実際はプロトコルとしては、内も外も同じプロトコルを用いているのが現状である。逆に、このことが問題の原因になっている可能性もある。autonomous system 内で用いているものをその外でも使うことによって矛盾する要求が生じることがある。そこで、プロトコル実装上の問題と、プロトコルによって覆いきれない問題についてを原因とともに述べてみる。

3.2.1 プロトコル実装上の問題点

ここでは、経路制御で用いられているプロトコルの実装上の問題点を述べる。autonomous system 内の経路情報交換には RIP を用いる。その際に生じる問題として、

1. autonomous system 内に存在する全てのトポロジを扱えるわけではない、
2. もしある一つのゲートウェイが落ちたとする (あるいは、網が切れているかも知れない) と、それに代わる置換経路があればそれを使って通信が行なわれる。その際、置換経路の伝播が完了してない時の経路がループを起こすかも知れない、
3. もし経路が更新される時、更新の情報が全てのゲートウェイに伝播されるまでの間は元の経路が保存されており、その間の経路は矛盾を招くおそれがある、
4. RIP 自身のパケットによってネットワークの負荷を上げ、ネットワークを非効率的なものにする恐れがある、
5. RIP を用いる上で有効なホップカウントは 15 までなので、それ以上のホップカウントが必要なネットワークとは情報交換ができない、

などがある。現在の経路制御では今使っている経路だけ (次にどのゲートウェイに対して送るか) を知っていて、存在する全ての経路を認識しているわけではない。もちろん、いつもそのことを知っている必要があるわけではないが、今の経路が切れて置換経路が必要な時に分かっていた方が対応が早いことがある。この意味で、管理者がトポロジを理解していることが必要な場合もある。

置換経路の伝播は誤情報の伝播を避けるために、冗長的に伝わっていく。その間にも通信が行なわれる可能性があり、経路が更新されていればその経路に従ってデータは送られる。しかし、どこか途中がまだ更新されていない時、このデータがデスティネーションまで到達する保証がなくなる。

RIP は 30 秒毎に自分のルーティングテーブルの情報を伝達する。従って、常にネットワーク上に RIP のパケットが流れているといえる。このことが網自身、或はホストの負荷を増すことにもなる。しかし、これは RIP に限ったことではなく、このようにある期間毎にパケットを送るプロトコルの一般的な問題ともいえる。以上のように、プロトコルとして特性を持っているが、その特性が逆に広域ネットワークの利用を考えた時に問題の原因になることもある。

ホップカウントの最大が 15 という実装の仕方は、もともとローカルネットワークを対象にして考えられたプロトコルであることから理解できる。そのプロトコルをそのままインターネットに利用していることから、この問題は生じてきている。

本論文では、これらの問題点をプロトコルの実装を変更することによる解決を目的にしているのではない。プロトコルの情報を取り入れ、更に機能を付加することにより問題点を解決することを目的としている。

3.2.2　プロトコルでサポートされていない問題点

autonomous system の外の広域ネットワークに注目すると、外側は複数の管理権限が共存しているので、それらを侵してはならない。また自分自身も外向きのポリシを持って経路制御をしていかなければならない。

ゲートウェイが、各々の autonomous system の制御情報を知っていると、通信における性能が分かったり、経路の動的な変化が分かったり、という広域ネットワークを安定的に保持するために有用なことが多い。実際広域ネットワークでは、経路の動的な変化は頻繁に起こり得る。

そのような広域分散環境をネットワーク管理者の立場から、更に経路に焦点を絞って考えた時の問題点を述べる。ここで問題になるのは、どこかの故障によって今ある経路が使用不可能になり、置換経路によってカバーしようとした時、どのような環境が提供できるかということである。なぜなら、ネットワーク管理の大前提は、エンドユーザがネットワークを意識せずネットワークを利用できる環境を提供することだからである。従って、問題は動的に変わる可能性のある経路の情報を、いかに動的に監視するかということである。

動的な監視によって管理者が知りたいことは、やはり経路が切れたという情報である。もちろん、伝播される情報にはメトリックスとしてのホップカウントが減り、到達が可能であるという情報もある。このような喜ばしい情報を知ることもトポロジをとらえる意味で大切である。しかし、経路がより良くなってエンドユーザに環境が提供できなくなることはないが、経路が切れると環境を提供できなることは大いにありうる。従って、ここで知りたい情報は特に経路が切れるという情報になる。

更に、この切れたことの伝播についてもう少し堀下げる。RIP による伝播は冗長的に伝播される。従って、その経路が切れたかどうかはしばらく立ってからわかることになる。しかし、場合によっては、絶対に切れては困るとか、切れたらすぐ知りたい、という要求がある。これは、RIP によって実現することはできない。そこで、本論文でこのできるだけ早く切れた情報を検知するシステムを提案する。

この問題を解決するには、ある経路が使用できなくなったという情報を受け取り、環境が変わってしまった場合に、

- エンドユーザ、に提供可能な環境を必要に応じて知らせる、
- その置換経路の容量がどうなのか、その経路と容量の情報を持つ、
- 実際にどこで故障が起きたのかを調べる、

といったの機能が必要になる。

これらの機能を備えて、動的に変わる経路に対し動的に対応し、更にユーザにできる限りの環境を提供しなければならない。このようなプロトコルによってサポートされていない問題点についてのアプローチを次に述べる。

3.3 問題点に対するアプローチ

そこで、具体的なアプローチを考える。上述した三つの問題点をそれぞれ解決することがこの問題解決のアプローチと考える。その中でも本論文では特に、三番目の故障の検知に着目する。前の二つの問題については今後の課題にする。特に二番目については、その情報をどのように保持し伝播するかというまさしくマネージメントプロトコルを必要とすると考えらる。

具体的に以下のようなアプローチを考える。それは、

1. ユーザインタフェース、
2. カレント経路検索システム、
3. 故障部分検索システム、
4. 重要リンク決定システム、
5. 経路診断システム、

このように分類される。ここで、経路診断システムが、ユーザインタフェース、カレント経路検索システム、故障部分検索システム、重要リンク決定システムをサポートする。インタフェース部分は、システムが把握した情報をユーザにより視覚的に提供する。重要リンク決定システムによってデスティネーションを定める。そして、そのデスティネーション–自分間の経路をカレント経路検索システムによって把握する。その経路情報をデータ

ベースにして故障部分検索システムで故障を検知しさらには場所の究明をする。つまり、このシステムは RIP による冗長的な情報の伝播を補うために、自分から積極的に応答要求を出すことで経路の生の状態を把握しようというものである。

　このような経路診断システムを設計することによって、問題を解決させる。次節で、その具体的な設計について述べる。

第 4 章

経路診断システムの設計

この章では、経路診断システムとは何か、何を目的としたシステムか、をまず述べる。そこで、目的を達成するためにどのような設計が必要かを述べる。更に、その設計ではこのシステムはどんなものから構成されるかを述べる。その構成要素がそれぞれどのような役割をしていて、どんな設計で、何をするのかを述べる。

4.1 経路診断システム

個々のシステムの具体的な設計をする前に、経路診断システムがどういうものかについてを述べる。

エンドユーザにとってはネットワークのことを気にせず、ネットワークを利用できることがもっとも望ましい。それを提供する側の管理者にとっては、ユーザが気にせず利用できるような環境を築くことが目標である。そこでは、動的な経路制御を動的に把握することが最終目標である。このシステムの実現のためにはいくつかのステップを踏む必要がある。この経路診断システムは経路情報を利用して経路を動的に管理するために、以下のような経緯を踏まえる。

第一は全てのトポロジの理解である。しかし、全てのトポロジの理解をインターネットで実現することは非常に困難である。なぜなら、そのトポロジは複雑、雑多だからである。しかし、これが動的に把握できるということは経路制御する上で、とても大切なことである。なぜなら、経路を動的にとらえていればネットワークの故障に気がつきやすかったり、その箇所が調べやすかったりするからである。更に、ある所からある所までで到達可能な経路の情報をとらえられることも大切である。そこで、このトポロジの理解は、全ての経路を把握することは困難なので、現在の経路を検索することによって補う。さらには世の中に存在する全てのデスティネーションに対して現在の経路を把握するというのは、莫大なデータベースを持つことになり有用でないし、また広域ネットワークを利用する上でもそ

のデータベースのうちの一握りが有用なデータになるであろうと予想されるので、デスティネーションを自分で特定して、そのデスティネーションに対しての経路を把握する。

次は、どこかの場所が落ちたり網が切れたりしたとき、その場所と原因を究明することである。原因の究明に関しては、ここでは触れない。将来的に、切れた経路の原因を自動的に究明でき、更にそれを自動的に修復できるようなシステムは非常に重宝するであろう。ここでは、経路が切れた場所を発見、するシステムを構築する。

この時、そのネットワークの状態を判断するメトリックスとして、

- それぞれの網にかかっている負荷、
- ゲートウェイにかかっている負荷、
- その網を使ったときのラウンドトリップタイム、
- 通信パケットの応答率、

などがわかると、その判断の目安になる。その中でも、本システムではパケットの応答率を経路が切れたことの判断基準にした。ネットワークの網自身の負荷や、ゲートウェイの CPU にかかる負荷、というのは確かに経路が切れる直接の原因となる。マシンの CPU にかかる負荷を耐えず調べるということは可能かもしれないが、いくつものゲートウェイを介していたりすることを踏まえると実際的ではない。ましてや、網自身にかかる負荷を監視するということは実現しにくい。そこで、その直接の原因によって影響が及ぼされる二次的な結果としての、ラウンドトリップタイムや、パケットの応答率 (あるいは、到着率) を用いる方が、そのデータの収集を考えた時に実際的である。また、ラウンドトリップタイムと応答率というのは独立したものではなく相関しているものなのでどちらをパラメータにしても同じであるので、最終的な判断としてパケットの応答率を選んだ。ここで、最終的なと言ったのは実装に関してはラウンドトリップタイムを基に応答率を導いているからである。

このように経路診断システムではまず、デスティネーションを決め、そのデスティネーションまでの経路調べ、そのデスティネーションに対して定期的に応答要求をすることで経路状態を捉えている。さらに、その応答率によって経路の異変を察知した時は、経路のデータベースから中間ゲートウェイに対して応答要求をして、どこの経路が切れたかを調べている。

ここまでが分かることで、一つの経路を動的に捉えたことにする。これが、システムの目的と概要である。残された問題は、このシステムを構

築する際に限って問題になるのではなく、ネットワークを利用したシステムを構築する際に必ず問題になる、“そのシステムによるネットワークへの影響である”。特に、動的に何かをする時は気をつけなければならない。このことをあまり気にしない短絡的な実現方法は、もし静的な情報を捉えることができているのなら、耐えずその情報を得ることができるような要求パケットを送り続けるという方法である。しかし、経路情報のように耐えず変わる可能性があり、また安定しているとほとんど代わらないといった情報を得るために、耐えず要求パケットを送り続けることはネットワークの負荷を非常に増大させる可能性があることを予想しなければならない。

このことを考慮すると、ネットワーク上のアーキテクチャとしては次の二つが考えられる。

1. 自分のマシンにあるシステムを載せ、他のマシンには負荷をかけずに実現する (**Figure 4.1**)。

2. お互いのマシン毎に自分の情報を保持し、それらを規則にしたがって交換することにより実現する (**Figure 4.2**)。

前者は、自分が望む要求に対する行動を全て自分で行なう。従って、他のマシンには負荷をかけることはなく、他のマシンの持っている情報を自分が得るだけである。この方法では、他のマシンに負荷をかけないことが利点である。しかし、もし必要な情報が非常に多く、その情報を保持するだけでマシンの負荷になり、それを処理するのにまたマシン大きな負荷がかかってしまう場合はこの方法による実装は効率的でない。

逆に、後者はお互いで “このような情報をこのように管理しておこう” という合意をする。その合意の基に、自分の保持している情報を交換し合う。この方法は前者の欠点である、非常に多いデータ量という問題を解決する。その意味で、広域分散環境のネットワークを管理するには一般的である。しかし、互いにデータを交換し合う規則を構築しなければならない。これは、ネットワークのアーキテクチャに大いに関わることで簡単なことではない。

そこで、本論文では前者を用いた実現を考えてみた。このような方法で経路を動的に管理することを実現する。

以上のように、経路診断システム (**Figure 4.3**) は、

1. ユーザインタフェース、

2. カレント経路検索システム、

3. 故障部分検索システム、

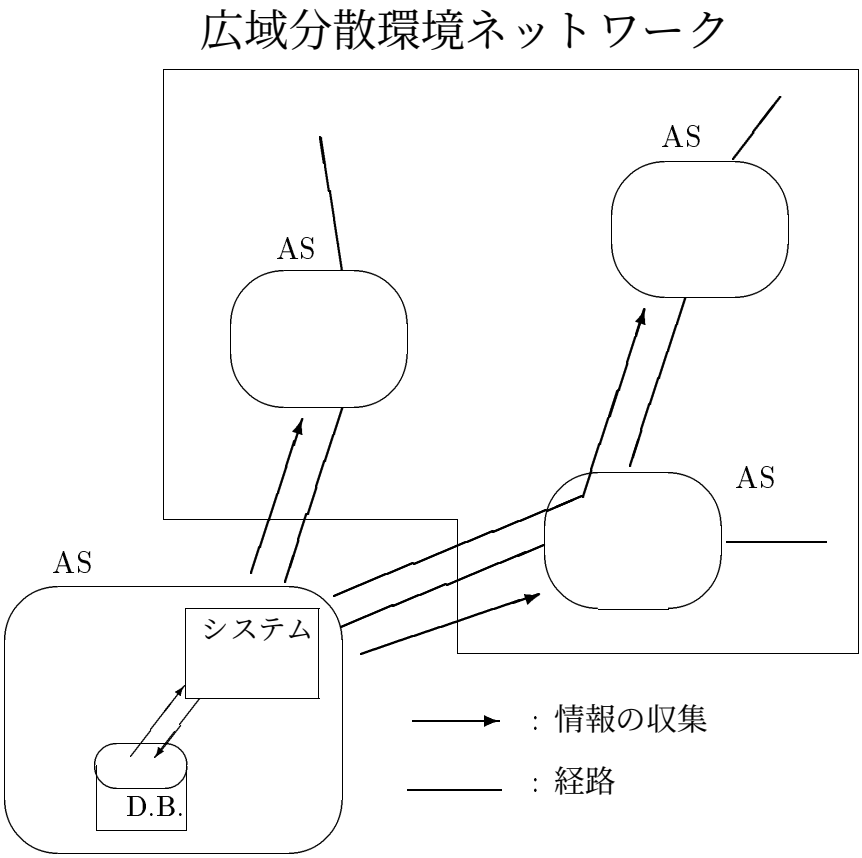


図 4.1: アーキテクチャ1

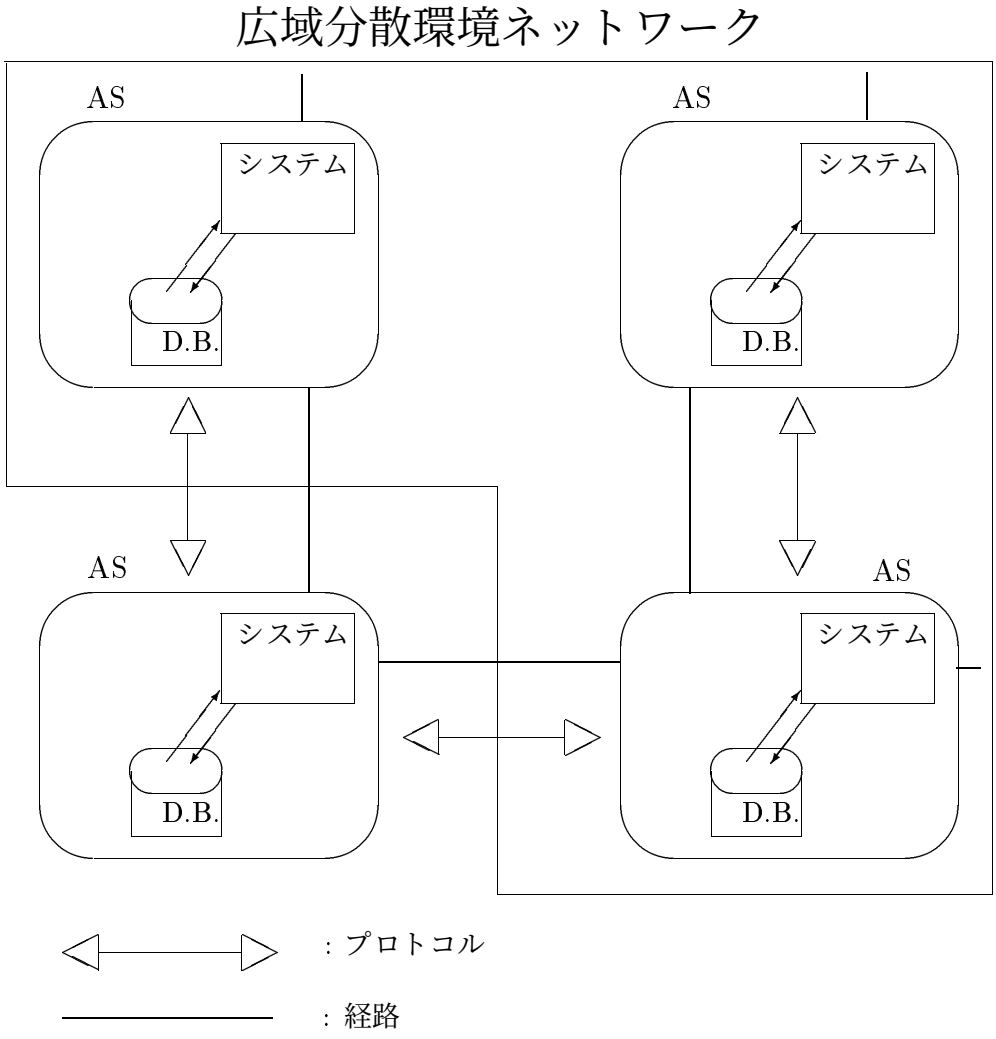


図 4.2: アーキテクチャ2

4. 重要リンク決定システム、

の四つのシステムを統合的にサポートして、経路を診断し、経路状況の情報を提供する。

4.2　ユーザインタフェース

経路診断システムのユーザインタフェースとは、システムからの情報をユーザに見せるための部分である。本システムの中のメインとなる部分を占めるシステムではないが、システムの診断結果を直接表す部分という意味から、本システムの成果を映し出しているといえる。ここでは、データベースを基に診断した結果をディスプレイ上にアピールしているので、デスティネーションまでの経路と、もし異変があった時どこで起きたかということが分かる。従って、その結果から今までのホップカウントだけによる経路制御に、ラウンドトリップタイムを基にした応答率による経路状態が分かる。このことで、経路の動的な変化に対応している。具体的には、まずデスティネーションまでの経路が決定したらそれを表示する。そして、何か状態変化を知らせる信号を受け取るたびにその情報を表示する。これを繰り返している。しかし、前にも述べたが将来的にはもっと様々なメトリックスを反映させて経路状態を診断するのが望ましい。そうすれば、動的な経路変化をより正確に監視できるようになる。

また、広域ネットワークを考える時、経路情報に加えて次のような情報が得られることも有用となるかも知れない。

1. autonomous system の外とつながっているゲートウェイの情報。

2. ある autonomous system の管理者の名前。

autonomous system 外とつながっているゲートウェイが落ちるとそのネットワーク、あるいはそのゲートウェイを介して通信するネットワークへの到達が不可能になる。それにともない、その autonomous system の管理者名や、もし管理者がログインしてたら、そのマシン名の情報が出ると便利である。また、autonomous system 外とつながっているゲートウェイが落ちそうになったら、落ちるのをまって落ちたら RIP を送ることを待たずに、もし可能なら、先に置換経路に切替えてしまう機能を持っていると役立つ場合もある。

本論文では、このユーザインタフェースに焦点を当てるのではなく、そのためのデータベースの保持、データベースを作るためのシステムの実現を第一目標にしている。そして、その情報を利用したユーザインタフェースを構築する。

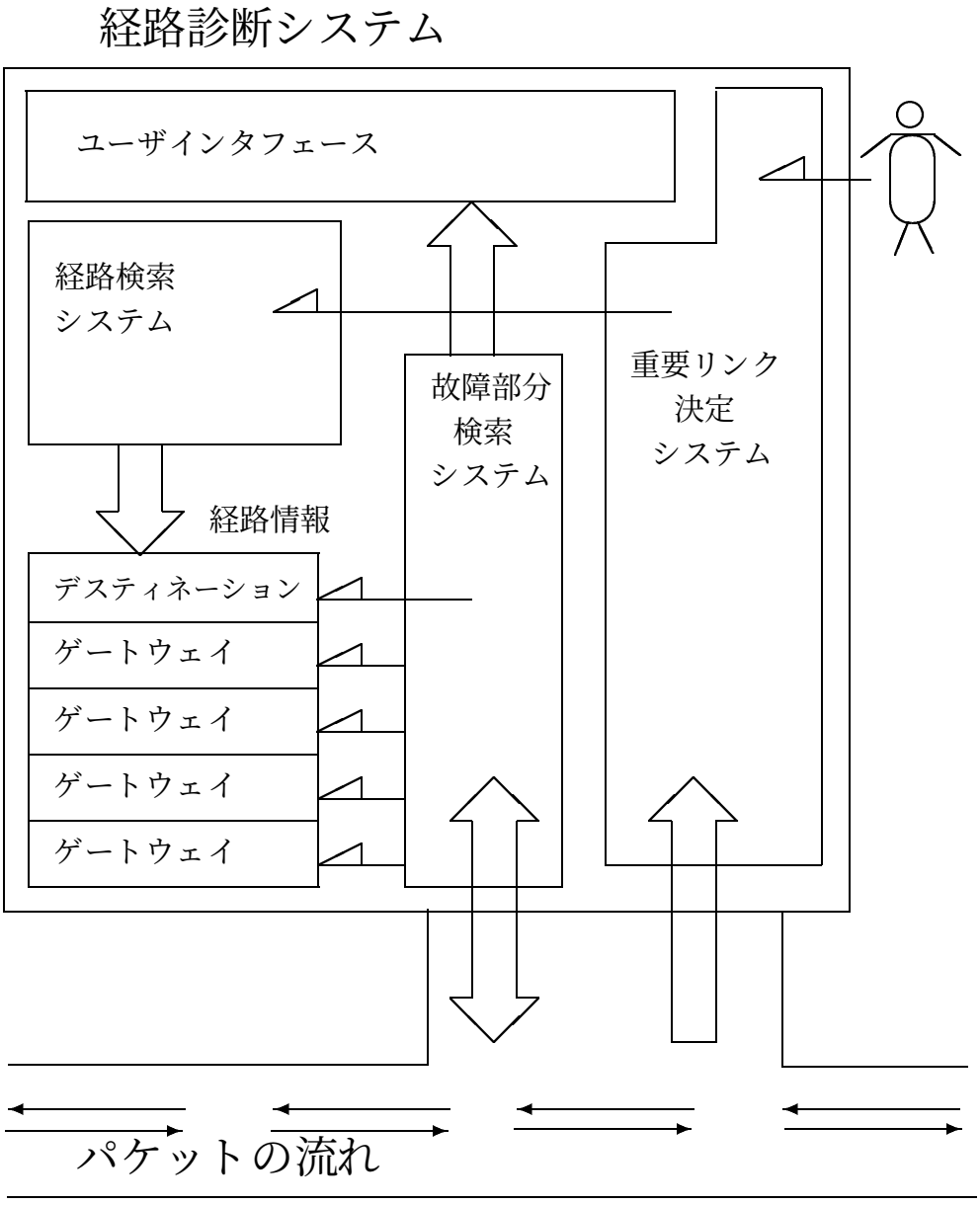


図 4.3: 経路診断システム

4.3　カレント経路検索システム

カレント経路検索システムでは、文字通りデスティネーションまでどのような経路をたどっているかその経路を検索する。その際検索される経路は、現在のホップカウントを用いて定められている、ルーティングテーブルに従った経路である。さらに、その調べた情報をいつでも引き出せるようにデータベースとして保持する。

カレント経路システムの実現は、以下のものから成り立っている。

1. 保持する情報となるデータベース。
2. その情報を得るためのシステム。

カレント経路検索システムの設計をこの二つから述べる。

4.3.1　データベース

データベースはシステムというよりは、システムによって生じた結果であるが、実際に参照したり、更新したりという意味からは逆に一番大切ともいえる。このデータベースの中の中心は介している中間ゲートウェイであるが、付帯情報としてラウンドトリップタイムも有している。

4.3.2　データベースを得るためのシステム

データベースを得るとは、デスティネーションまでの経路を検索し道順を築き上げることである。その経路の決定の仕方は現在の経路制御においては一意である。そこでまず経路決定のアルゴリズムを説明する。そしてその後、そのアルゴリズムに則ったデータベースの得方について述べる。

(1) 経路決定のアルゴリズム

経路の決定のアルゴリズムは現在は一つしか存在せず、いかなる実現方法を用いても結果は一つである。その決定方法とは、メトリックスとしてホップカウントを用い、デスティネーションに対しホップカウントの少ないゲートウェイへ向かう経路をとるというものである。従って、各マシンのルーティングテーブルにはデスティネーションネットワークやデスティネーションへ送るための次のゲートウェイの情報が記述してある。そのデスティネーションへのゲートウェイは一意である。この情報は 30 秒毎に各ゲートウェイが RIP をブロードキャストすることにより、今の経路がホップカウントをメトリックスとしたときに最短であるかどうかが決められる。よって、経路はデスティネーションが与えられた時そのデスティネーショ

ンネットワークに対するゲートウェイが決まり、その経路をとる。そして、そのゲートウェイがまた自分のルーティングテーブルに従って経路をとる。このように、デスティネーションまでの経路が決定されるのである。言い換えると、経路を決定する際自分は次のゲートウェイまでの責任しか持たず、それ以降はそのゲートウェイに任せている。

さてそのデータベースの得方、すなわちデスティネーションまでの経路の検索の仕方であるが、大きく分けて二種類ある。

- 情報伝播プロトコルを用いてルーティングテーブルからのデータを集積する、という各マシンに依存した方法。
- 自分から送るパケットの IP [56] データグラムの time-to-live フィールドを利用する、という自分のマシンにのみ依存した方法。

前者の方法を取り入れるものとして、二通り考えてみた。

1. SNMP を用いたもの。
2. RIP query を用いたもの。

(1) SNMP を用いた方法

現在一口に SNMP といっても様々のものが広まっているが、一番機能の整っているものとしては製品化しているものがある。また、MIT や CMU で作られたものもある。

製品化している SNMP を用いるためには各自治システムがその製品を購入しなければならないということになり、ここでは考慮にいれない。では、MIT や CMU で作られた SNMP を用いたとすると、これはコマンドによってデスティネーションのルーティングテーブルが取ってくることができる。従って、必要な時にこのコマンドを使うようなシステムにすれば、そのホストのルーティングテーブルを取り出すことができる。しかし、この SNMP がそれぞれの自治システムのゲートウェイ上で動いている保証はない。

(2) RIP query を用いた方法

こちらは逆に、今ある RIP というプロトコルを利用して他のホストのルーティングテーブルを取り出そう、というものである。ここでは、RIP query という機能をあるゲートウェイに出すことによって、そのゲートウェ

イが送っている RIP の情報を聞き出すことができる。このことによってそのホストのルーティングテーブルを取り出すことができる。

以上の通りであるが、SNMP はマネージメントシステムとして開発されたものであるがまだまだ発展途中であり、また、各 autonomous system で取り入れられているかという問題もある。RIP query を用いた場合ルーティングテーブルの冗長的な更新の問題点は解決されない。いずれにしても、プロトコルによって情報を伝播し、その情報を基に経路を構築するということでパフォーマンスがかかる。

次に後者による方法である。

(3) IP データグラムを用いた方法

この方法は IP データグラムの time-to-live フィールドの特性を使うものである。time-to-live フィールドとは文字通り生存時間を表している。すなわち、IP データグラムを受け取ったゲートウェイはそのパケットが自分宛でなければその time-to-live フィールドの値を 1 引いて次に送る。その時その値が 0 であるとそれ以上送ることはせず、“0 になって目的のデスティネーションに届きませんでしたよ” という返事を送り主に返すのである。

この特性を利用して最終デスティネーションまでの経路を構築する。

SNMP はまだあまり一般に広まっていないし、また、RIP query を用いる場合も経路を構築するまでに探索コストがかかる。そこで、本論文では三番目の IP データグラム (**FIgure 4.4**) を用いた方法を採用した。

4.4　故障部分検索システム

故障部分の発見についても次の二通りが考えられる。

1. RIP に応じた検索による発見。
2. 積極的な検索による発見。

能動的、受動的の違いはあるが、いずれにしてもこのシステムでは自分の持っている経路情報を基にどこで故障がおきているのかを発見するものである。

4.4.1　RIP に応じた検索

RIP に応じた検索をする場合は、カレント経路を検索する時にも RIP に基づいたデータベースを持っていなければならない。具体的には、自分

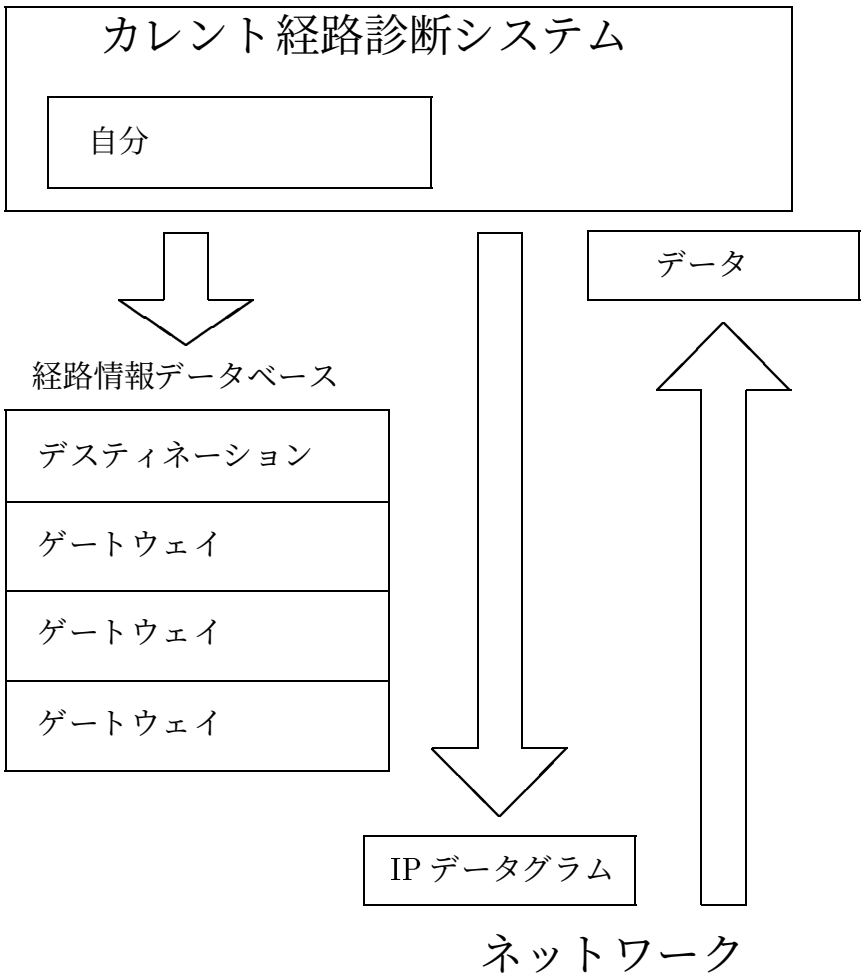


図 4.4: カレント経路診断システム

の持っている経路情報と RIP によって 30 秒毎に伝わってくる情報を見比べる。RIP による伝播で切れたことが分かるのは、ルーティングテーブルが更新されないことによってルーティングテーブルの中からそのエントリが消えることによって認識できる。その後、各中間ゲートウェイに RIP query を出してどこから情報が途切れていいるのかを発見する。

この場合、RIP からの情報を待っていて、それが伝わったらデータベースを基に切れた場所の究明をする。従って、情報を待つことにより切れたことの認識に時間がかかることが予想され、総じて故障箇所の発見にも時間がかかる。

4.4.2　積極的な検索

RIP に応じた検索が、RIP による情報の伝播から切れた場所の発見をするという受動的な検索であるのに対し、この積極的な検索では、自分からデスティネーションに対し、定期的に応答要求のパケットを送ることによって経路の状態を確認するという能動的な検索をする。

具体的には、デスティネーションに対して定期的に応答要求パケットを送る。そこで、その応答率がなくなったことで切れたことを判断する。その後、自分のデータベースの中の中間ゲートウェイに応答要求を出す。これによってどこが切れたかが分かる。

このように、RIP に応じた検索では、切れた場所が発見できるという機能が加わるだけで、切れた場所、あるいは切れそうな場所の早期発見にはならない。これに対し、積極的な検索では RIP による情報の伝播よりも早く発見できる可能性がある。従って、本研究では故障部分の発見に、RIP に応じた検索ではなく、積極的な検索による方法を用いる (**Figure 4.5**)。

4.5　重要リンク決定システム

このシステムで、ターゲットとするデスティネーションを定める。WIDE 広域分散環境ネットワークの中で、どのリンクが重要なのかを定めることは非常に困難なことである。そこで、本論文ではその重要さをエンドユーザの主観的な立場から見た定め方と、実験から定める方法とを用いた。

4.5.1　主観的な定義

これは、そのエンドユーザの主観によって重要なリンクを決めるものである。この主観にもここでは次の二通りがあると考える。

1. 経験による主観。

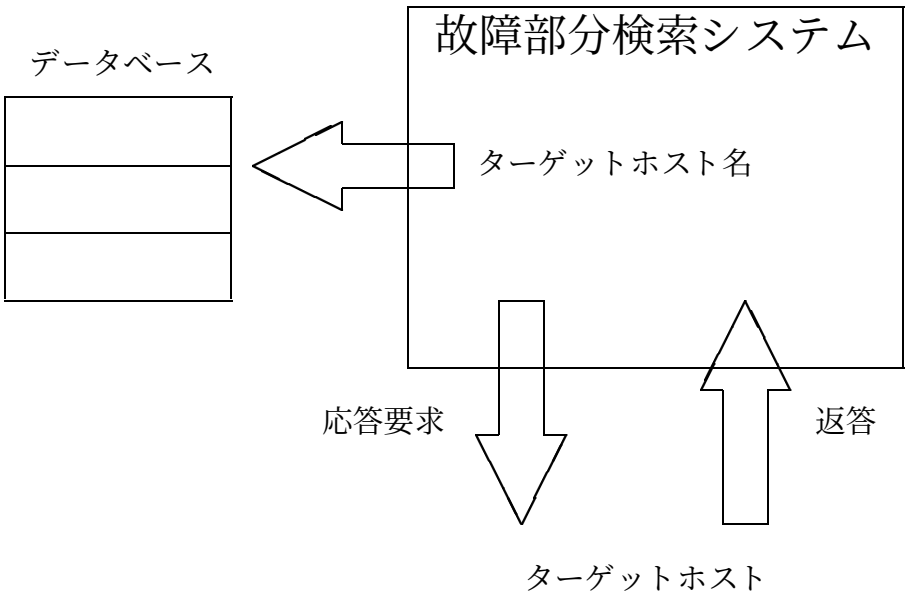


図 4.5: 故障部分診断システム

2. ポリシによる主観。

経験による主観とは、文字通り、今までの経験からここのリンクは多用しているのでどうしても切れては困る、というものである。これに対し、ポリシによる主観とは、たとえあまり使われなくともポリシ上どうしても切れては困る、というものである。

いずれにしても、ここでの定義では自分で経路を診断したいデスティネーションを定めることになる。

4.5.2 実験的な定義

これはあるホストに対し、そのホストに出入りする IP データグラムの統計を定期的に取ることにより、そのホストにとってどのリンクの使用頻度が高いかを調べる。具体的には、IP データグラムのソースフィールドと、デスティネーションフィールドの統計を取ることで、どのホストとの通信が多いかが分かる。その結果から、使用頻度の高い経路をそのホストにとって重要な経路とするものである。そこで、その重要度を以下のように定義する。

A 指定: 1 時間毎に 10 分間データグラムを取り出入りする IP データグラムのうちの 50%以上を占めるリンク。

B 指定: それ以外のリンク。

ここで、A 指定のリンクは自動的にそのリンクを監視する。B 指定のリンクについては何もしない。その理由は、多くのホストに向けて経路診断をすることはネットワーク中に多くの応答パケットを投げることになるので、そのことによって逆にネットワークの網の負荷を上げてしまい効率が悪くなりかねないからである。

これらのように、この重要リンク決定システムでは、経路診断を実際にどのデスティネーションに対して行なうか、それを決定するためのものである。

第 5 章

実装と評価

ここでは、経路診断システムの実装についてと、実装に当たり、本システムと RIP による実装との比較、システムが与えるネットワークへの負荷について述べる。

5.1 実装

前節の設計に従って具体的にどのように実装したかを述べる。このシステムの中心部の流れは次ページに示す。

5.1.1 ユーザインタフェース

ここでは、デスティネーションの状態を画面上に表わすことで経路の診断結果としている。前述のアルゴリズムの中の信号部分の時、デスティネーション、あるいは中間ゲートウェイの状態を表示する。

具体的には、

- 危険信号 ··· ホスト名の点滅、ブザー、
- 通信不能信号 ··· ホスト名の反転、経路への印づけ、ブザー、
- 通常時 ··· ホスト名、経路、

とすることによって状態の遷移を表わしている。

このようなディスプレイ制御のシステムを、付加機能を持った便利なものになるよう構築すると管理者だけではなく、エンドユーザにとっても意義のあるアプリケーションになる。これは、今後の課題にする。

```
main(){
    /* デスティネーションまでの経路を探索 */
    /* 探索の結果をデータベースとして保持 */
    カレント経路検索;
    for (;;) {    /* デスティネーションと通信可能の時 */
          if (デスティネーションが生きている場合) {
                /* デスティネーションに対し応答要求を開始 */
                応答要求 (デスティネーション);
                if (パケット喪失率 != 100%) {
                      if (パケット喪失率 >= 90%) {
                            危険信号;
                      }
                      else {
                            for (データベースのエントリがなくな
るまで) {
                                  応答要求 (ゲートウェイ);
                                  if (パケット喪失率 == 100%) {
                                        ゲートウェイへの通信不
能信号;
                                  }
                            }
                            デスティネーションへの通信不能信号;
                      }
                }
          }
          else {    /* デスティネーションと通信が不能の時 */
                応答要求 (デスティネーション);
                /* 再び通信可能になる */
                if (パケット喪失率 < 90%) {
                      通信復活信号;
                      カレント経路検索;
                }
                else {
                      死んだまま;
                }
          }
    }
```

}

5.1.2 カレント経路検索システム

カレント経路検索には IP データグラムの time-to-live フィールドの特性を利用した。現在の IP の実装は、送られた IP データグラムが自分宛なら取り込み、そうでなければルーティングテーブルに従って次のゲートウェイにそのデータを転送する。その際 time-to-live フィールドの値を 1 減らしてから転送する。そこで、もし time-to-live フィールドが 0 になったらそのパケットを捨て、そのことを送り主に伝える (これは、ICMP(Intenet Control Message Protocol) [55] の TIME_EXCEEDED という機能によって行なわれる)。従って、time-to-live フィールドが 1 である IP データグラムを受け取ると転送する時に 1 を引くと 0 になってしまうのでそこでそのデータグラムを捨て、そのことを送り主に対して伝える。その時の、ICMP データグラムの送り主は元の IP データグラムを捨てたホストになる。

以上のことより、time-to-live フィールドを 1 から一つずつ増やしてデスティネーション宛に IP データグラムを送るとその返ってくるデータグラムによって途中のゲートウェイが分かるのである。

5.1.3 故障部分検索システム

故障部分検索には応答要求を利用した。これは ICMP の ECHO_REQUEST 機能を利用している。この機能は、ECHO_REQUEST パケットを受け取るとエコバックしなければならないというものである。ECHO_REQUEST パケットを受け取るとエコバックのために ECHO_REPLY パケットを送り返す。これによって経路及び、ホストの生存が確認できる。

ここでは、この ECHO_REQUEST パケット 10 個を 1 秒毎に送り、それに対してエコバックされた ECHO_REPLY パケットの個数をカウントし、その割合によってホスト、及び経路の状態を認識している。具体的には、パケットの喪失率を以下のように定義する。

$$lostpacketslate = \frac{sendingpackets - receivingpackets}{sendingpackets} \times 100$$

この値をもとに次のように経路の状態を定義づける。

1. パケット喪失率 100% $\cdots$ どこかの経路が切れている。
2. パケット喪失率 90%以上 $\cdots$ 効率が悪い。
3. パケット喪失率 90%未満 $\cdots$ 正常。

このパケット喪失率によってデスティネーションへの経路が切れていると診断されると、経路情報のデータベースからホストを一つずつ取り出してそれらに対して同様に応答要求を出す。その結果、どこの経路が切れていたのかを発見する。

5.1.4　重要リンク決定システム

重要リンクの決め方のうち、経験的に決める方は自分の主観でターゲットとなるデスティネーションを定めるので、この場では割愛する。さて、IP データグラムの統計のとり方であるが、ここでは NIT(Network Interface Tap) [67] を利用して、イーサネット上にながれている IP データグラムを全て拾い上げる。そのデータグラムのソース、デスティネーションの統計を調べてどのホストとの通信が多いかを定め、そのホスト経路を診断するためのターゲットデスティネーションにする。

しかし、この経路診断システムで自動的にデスティネーションを決定して自動的に診断を始めるという実装はしていない。IP データグラムの統計をとってその結果からターゲットデスティネーションを決める。そして、そのデスティネーションに対して経路の診断をするようにした。

5.2　評価

ここで、RIP の実装と、本システムの実装を比較することにより、本システムの評価をする。RIP によって切れたリンクを検知するには、RIP による情報の伝播が冗長的であるために時間がかかる。これは、RIP が悪いことを表しているわけではない。RIP のアルゴリズムとして、良い情報 (つまり、少ないホップカウントで到達できるという情報) はいちはやく伝播される。すなわち、すぐルーティングテーブルが更新される。しかし、悪い情報 (ホップカウントが増えたという情報) は、その情報の信用性を確かめるために冗長的に伝播するのである。すなわち、悪い情報がルーティングテーブルの新たなエントリになることはない。従って、何かの間違いで悪い情報が生まれても、冗長的に伝播している間に正しい情報が伝わると、悪い情報は捨てられるのである。

このことから、RIP による経路制御では補えない問題を本システムで補佐してみた。そこで、RIP を用いた実装の結果と、本システムの実装の結果との比較を行ない、評価とする。

5.2.1 RIP の実装の結果

RIP を用いて経路制御を行なった場合の、切れたリンクの検知と対応の仕方は、

1. RIP のメトリックス (すなわち、ホップカウント) が徐々に増える、
2. メトリックスが 16 以上になった時、切れたことを認識する、
3. 置換経路が存在する時、より有用な置換経路を選択する、
4. 存在しない時は、人間が修復するかもしくは、新しいリンクを作る、

このように行なわれる。

そこで、例えば、koch–ccut(東大) 間の経路を考えてみる (**Figure 5.1**)。

ここで、ccut が落ちて koch–ccut 間のリンクが切れたとすると、ルーティングテーブルに default というエントリがないと仮定すれば、RIP による経路状況の伝播、変遷は以下のようになる (**Table 5.1**)。数字は、経路制御のためのメトリックスであるホップカウントを指している。

マシン	時間 (sec)	180s	420s	660s
ccut	loopback	Down	Down	Down
relay	ccut 1	ccut 16	消滅	—
jp-gate	relay 2	relay 2	relay 16	消滅
koch	jp-gate 3	jp-gate 3	jp-gate 3	jp-gate 16

表 5.1: ccut 宛のルーティングテーブルの変遷

これは ccut がダウンしたため RIP を伝播しなくなる。従って、直接接続している relay は ccut 宛の情報が伝わらず、180 秒後にメトリックスが無限大にセットされ、更に 60 秒後にルーティングテーブルの中から ccut のエントリが抹消される。これによって、relay が伝播する RIP の情報に ccut 宛の情報がなくなる。同様にして、更に 180 秒後に jp-gate が ccut のメトリックスを無限大にセットする。そしてその 60 秒後にエントリが抹消される。以下同様に koch にも ccut のダウンが 660 秒後に伝わる。そこで、koch は ccut 到達不可能になったことを知るのである。

しかし実際にはルーティングテーブルは default というエントリを持っている (**Table 5.2**)。

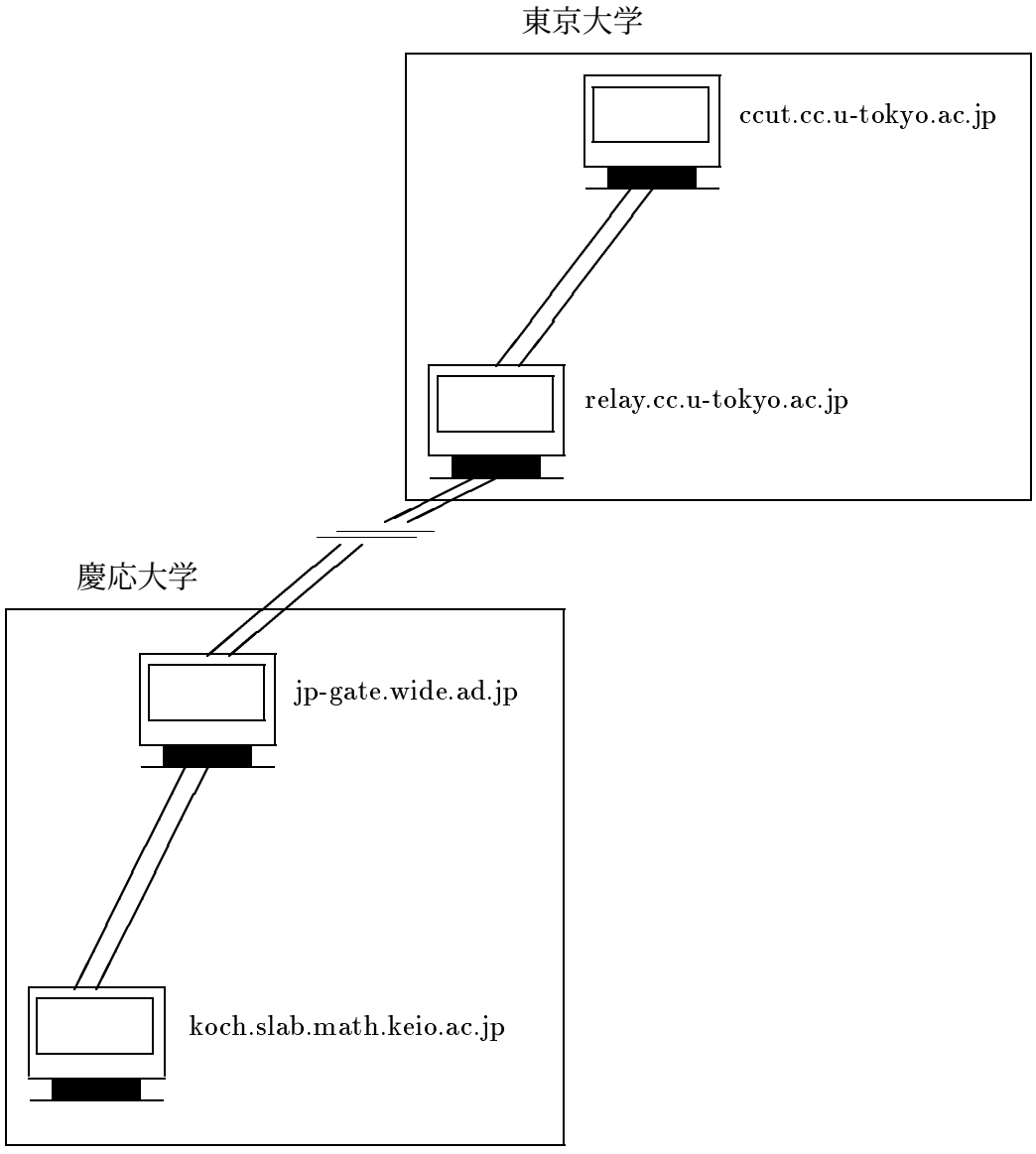


図 5.1: koch-ccut 間経路

Routing tables

Destination	Gateway	Flags	Refcnt	Use	Interface
radish.nakanish	kossvax.slab.ma	UGH	2	3859	ie0
keio-math-slab-	koch.slab.math.	U	26	1941748	ie0
default	jp-gate.wide.ad	UG	12	118593	ie0
loopback	localhost	U	40	293941	lo0

表 5.2: koch のルーティングテーブルの一部

このように、ccut 宛のパケットは defaults である jp-gate に送られてしまう。すると koch は何秒間たっても自分のルーティングテーブルにないデスティネーションへのパケットはデスティネーションを default として次のゲートウェイを選ぶ。従って、koch は ccut がダウンしたということを認識することはなくなるのである。これは実際に通信はできないのに、経路制御は自分のルーティングテーブルに従って行なうことを表わしている。このことが、RIP によって覆いきれない部分なのである。

5.2.2 経路診断システムの評価

RIP に対し、経路診断システムの場合の切れたリンクの検知の仕方は、

1. 重要なリンク (デスティネーション) を考え、そのリンク (デスティネーション) に対したえず応答要求パケット (ICMP ECHO_REQUEST) を送り、その応答 (ICMP ECHO_REPLY) を調べる、
2. そのラウンドトリップタイムが著しく遅れ最後に一パケットも返ってこなくなったことでそのリンクが切れたと仮定し、認識する、
3. その後、途中の全てのノードに対し監視パケットを送り、実際にどこのリンクが切れて、その目的デスティネーションに到達できないかを確認する、

となる。

そこで、同様に koch–ccut 間の経路を考えてみる。ここで、ccut が落ちてその間のリンクが切れると以下のようになる (**Table 5.3**)。

これより、約 30 秒で ccut–relay 間の経路が切れたことが確認できる。この 30 という値は初期設定によって変わる値である。この結果はあるターゲットに対して 1 秒毎に 10 個応答要求を出した時の値である。この応答要

マシン	時間 (sec)	10s	20s	30s
ccut	喪失率　100%	Down	Down	Down
relay	—	—	—	喪失率　0%
jp-gate	—	—	喪失率　0%	—
koch	—	喪失率　0%	—	—

表 5.3: 経路診断システムによる変遷

求を 2 秒毎に送ったり、また 30 個送り続けたりという設定をすることも可能である。もしあるホストに対して応答要求を 30 個送ったとすると、その情報の信頼性は高まる。つまり、10 個の応答要求に対しての応答率より、30 個の応答要求に対する応答率の方が精度は良くなるという意味である。しかし、どこかのマシンが落ちて通信不能になった時、そのそれぞれのホストに対して 30 個ずつの応答要求をすると、それだけ故障箇所の決定までに時間がかかることになる。また、間隔をあけることはネットワークへの負荷を与えないという点から有意義ではあるが、信頼性のあるデータを提供するためには間隔は疎でも長いあいだ送らなければならないので、早い発見は実現できないというジレンマもある。これらから、1 秒毎に 10 個送ることを選んだ。

また、今後の課題としては、例えば重要リンク決定システムに応じて要求パケットの送る個数、送る間隔、データのサイズを決めるような実現をするものが考えられる。

更には、より凝ったユーザインタフェースを作ることによりネットワークのトポロジと状態の理解を実現できる。例えば、ある程度のネットワークの状態図を表わし、その中のどれかを対象に選んでやるとその間のネットワークの状態を検索しに行くようにする。

いずれにしても、今後の発展はまだまだ考えられるが、とりあえず第一目標である故障箇所の早期発見は達成されたといえる。

5.2.3　実装によるネットワークへの負荷

設計のところでも述べているように、ネットワークを利用するアーキテクチャをこのシステムのように、一つのマシンが情報を一人で集めに行くという方法をとると、多数のパケットをネットワークに送り込まないとその実現はなかなか困難になる。そこで、その網の負荷をなるべく少なくするためにこのシステムでは、送るデータグラムをなるべく小さく必要最小限 (例えばヘッダ部分だけ) を送るように実装している。

具体的に、1 回の応答要求で 64 バイトを送る場合を考える。これによってうける、CPU の負荷と、ネットワークの網自体にかかる負荷を考える。

(1) CPU の負荷

これは、CPU のロードアベレージを xload で見ることによってその様子を調べてみた。この章の最後に、システムを動かした時とそうでない時の結果を示す。

これより、直接 CPU に与える負荷はそれほどないといえる (尚、この結果は任意にマシン kitasun のロードアベレージをとりそのときにシステムも動かしてみたものである)。

(2) ネットワークの負荷

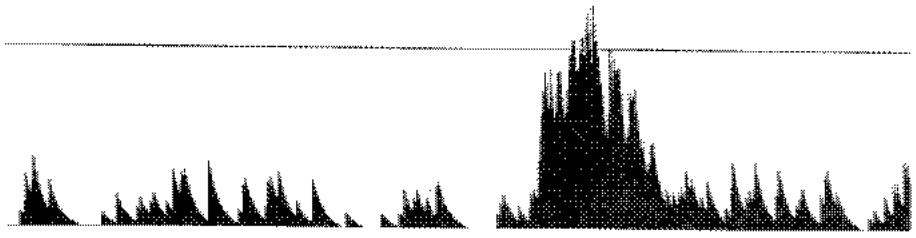
例えば jp-gate–relay 間の経路を考えてみる。ここはバンド幅が 64kbps(bit per second) の専用回線を用いている。この間にながれている IP データグラムの統計をしらべてみる。ここで、このデータを 10 分程度収集してみると 660byte/s 程度のデータグラムが流れている。従ってこれは約 6600bps のデータが流れていると考えられる。一方、このシステムが送っている IP データグラムは 1 秒間に 64 バイトであるから約 640bps である。以上より、もともと最大の 10%程度の使用量の所へ、さらにこの 10 分の 1 のデータを流してもそれほどの影響はないといえる。

もちろん、この 10 分間の統計は前後するが、そもそも最大容量の 100 分の 1 であるのでそれほどの負荷を与えていないと判断できる。また、回線に関しても 64kbps の専用回線ばかりではい。例えば、イーサネットは 10mbps であり、9.6kbps の回線、等様々である。結局、ネットワークに与える負荷とはその時使っている容量に対してまだどれだけの容量が余っていてそこにどれだけの容量を送るかという相対的な割合が重要になる。

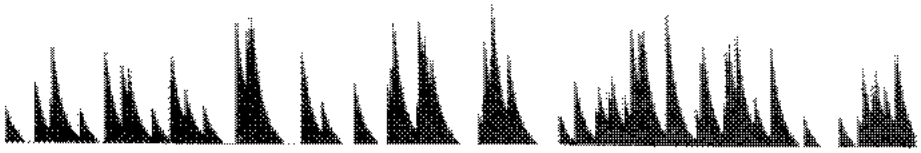
このように考えても 64 バイトが与える影響はそれほど大きくないといえる。

以上よりネットワークに与える負荷それほどないといえる。しかし、ネットワークを介したアーキテクチャでも述べた通り、一つのマシンが全ての情報を得に行くといった実現の方法を取ると、そのマシンにかかる負荷が大きくなる可能性はある。このシステムでも、これを 10 個のデスティネーションに対して行なえばそのネットワークに与える負荷は単純に 10 倍になる。また様々なマシンでこのように実現したシステムを動かせばネットワークの負荷にもなりかねない。従って、このシステムが残す課題もある。

しかし、第一目標を達成していることと、当座はそれほどネットワークに負荷は与えないという実験結果からこのシステムは成功と評価できる。



システムを走らせた場合



システムを走らせてない場合

第 6 章

結論

6.1 結果

目的でも述べた通り、このシステムの目標は動的に変化する経路状態を動的に把握することであった。そのために、実際には切れていなくても切れそうと判断して対処をしている。その理由は、ネットワークが広域になってくると、“どうしても切れては困るリンク”といった要求が出てくる。このような要求に応じるためにも、危険な状態はいちはやく発見して適切な対処をしてネットワークの安定性を保持しなければならないからであった。

そこで、本システムではその切れそうであるか否かの判断のために、応答要求パケットの喪失率を用いた。100%返ってこなければもちろん切れたと判断し、さらには 90%以上返ってこない場合にはそれを危険と判断している。このような実現により、危険箇所の早期発見をしている。そこには、そのパケットの喪失率の信頼性という問題もある。例えば、パケットは多く送ればそれだけその間の経路の応答性は正確に伝わる。しかし、多く送ることによって時間をかけてしまうと早期発見というテーマに対してジレンマが起こる。そこで、本システムでは 1 秒毎を 10 回というのを一つのサイクルにしている。もし一つもパケットを喪失していなければ約 10 秒後に応答要求が終る。もし一つでも喪失していると 10 個送った後、今までのパケットの返ってきた時間 (つまり、ラウンドトリップタイム) の最大値の 2 倍だけ待って終了する。また、一つも返ってこなかったら 10 秒間だけまって終了する。このように 10 個のパケットの応答要求に結論を導ている。“10 個では信頼性がないのでは”、という意見に対しては、各マシンはそもそも落ちるために作られているのではなく、たまたま落ちてしまうのである。あるいは、経路にしても切れるためにあるのではなく、たまたま切れてしまうのである。だとすれば、この応答要求は 10 回をサイクルに無限に行なうよう実装されている、従って、そこにひっかかった時が落ちたことを表し、その時までには何サイクルもの応答要求を行なっているか

も知れない。そう考えれば、たとえ 10 回でもそれに 1 回も応答がなければ異常といえる。そこで、異常を検知した後は各デスティネーションに対し 10 回ずつの応答要求をしてパケット喪失率が 100%であった所を切れた箇所と判断している。

さらに、このシステムではその切れたデスティネーションに対し応答要求を出し続けている。もちろん、落ちている間のパケット喪失率は 100%である。そこで、もしマシンが復活することによって経路が復旧したり、置換経路を取ったことによって経路が復旧するとこの応答要求に対してデスティネーションは返事を出す。これによってこのシステムでは復旧したこともすぐに分かることになる。続けて、もし復旧したらすぐにカレント経路の検索を始める。そうすると新しくできた経路が分かる。このことから、もし新しい経路が元の経路と同じであれば、経路が切れたことの原因がそのマシンが落ちた等の理由であるかも知れない。逆に経路が変わっていれば、経路が何かの原因で落ち、すぐに置換経路によってつながったとわかる。

このように、切れた後もデスティネーションに対して応答要求をすることによってその切れたことの原因についての検査にもなり、また、ユーザにとっても復旧がすぐに伝わり便利である。このことによって、本システムの目的であった刻々と変わる可能性のある経路を、動的に監視することの実現がなされたといえる。もちろん、この経路決定の基になっているのは現在の経路制御プロトコルであるので、経路決定に用いられているのはホップカウントだけである。今後は他のメトリックス (例えば、ラウンドトリップタイムなど) を基にした経路制御も考えられ、本システムがそれらに対する汎用性もあるといえる。

また、ユーザインタフェースで少しでもユーザの目に移るような実現をすることでその後の早い対処に役立ている。さらに、ユーザインタフェースを便利にすることで、システムの応用性が出てくる。このユーザインタフェースをより充実させて一つのシステムのより有用な実現をしたい。

また、重要リンク決定システムの導入がなされていない。これは、やたらに診断システムを動かさないためである。これをしっかり導入するためにはその経路の重要さの定義づけが必要になってくる。つまり、流れているデータグラムの統計から、その統計をどのように判断するかである。ここでは最も簡単な、宛先として流れている量の多いソースとデスティネーションを重要であると定義している。その宛先が重要であることは正しいと思うが、さらに細かい定義付けをして、それに応じて診断を始めるというのが望ましい。そのためにも、さらにデータ分析システムを導入した診断ができると有用である。しかし、このシステムで第一目標は達成されている。

以上の意味において、このシステムは故障部分の早期発見、及び経路情報の動的な監視に対し有用である。今後の課題もあり、システムをさらに発展させることにより、より正確な情報の捉え方が可能になる。いずれにしても、ひとつの目標を達成した点がこのシステムの成果である。

6.2　今後の発展

6.2.1　問題点

このシステムの最大の問題点は、自分のホストからの検索しかできないことである。従ってネットワーク管理の立場からの全てのトポロジの把握という目標を達成するまでには至ってない。そこで、自分が直接制御することのできない経路の情報を把握するという新たな目標が生じてくる。

6.2.2　課題

このシステムでは、RIP によって覆いきれない部分を支援した。しかし、ネットワーク管理を考えた場合、さらなる要求が存在する。このシステムでは、自分のホストが中心で自分のホストにかかわる経路に関する情報だけを調べることができる。従って、自分が直接関係してない経路の情報を調べられない場合がある。例えば、次のような経路のトポロジの場合を考える (**Figure 6.1**)。

ここでは、A–B 間、A–C 間の経路情報は分かるが、B–C 間を経由していないかぎり、B–C 間の経路情報は分からない。しかし、ネットワークを考えた場合ホスト A にとっても、B–C 間の情報を知っていた方がいいときもある。このような時、お互いに持っている情報を交換するようなプロトコルが必要になってくる。

このように、今後は情報交換プロトコルを利用するにしろ、任意の経路の状態が把握できるということが課題となる。

6.2.3　応用

このような経路情報管理システムの利用は様々考えられる。例えば、置換経路への早い移行ができる。つまり、今までのアルゴリズムでは切れてから、もし置換経路があればそれに変更するというものだった。そこで、切れる前に先につないでしまって、現在の経路が切れても通信は途切れないようにすることができる。あるいは、新しい通信媒体である ISDN の接続の目安になることもある。

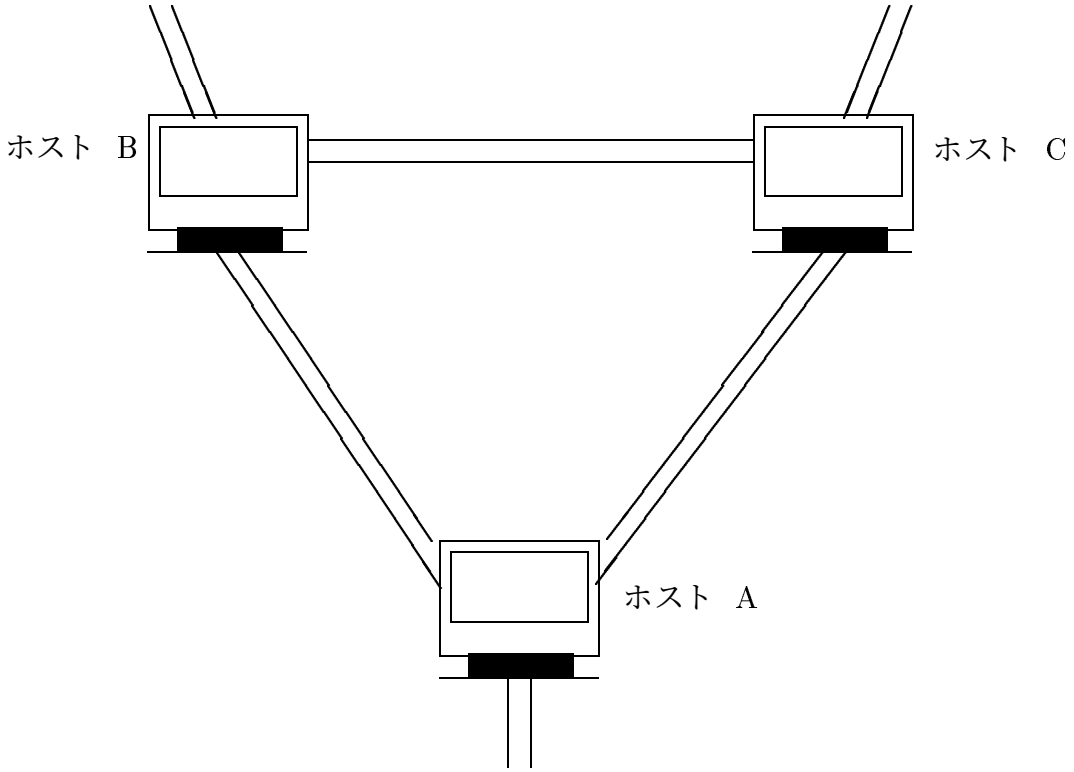


図 6.1: 例

さらには、経路情報に他のメトリックスを導入するとより正確なネットワークの状態を把握できる。例えば、ネットワーク利用の要求として次のようなものがある。

- そんな経路が存在するんだったら必ずそこから送って、
- このパケットは速い所から送って、
- このパケットは安い回線で送って、

などである。

このような、新しい要求に答えることで、ネットワークの安定した管理、そしてネットワークの効率のよい利用が実現する。

以上のように、本論文で述べた経路診断システムはこれからのネットワークマネージメントの一歩としては、その役割を果たせた。新しい課題を一つずつ克服していくことにより理想的なネットワーク運営を実現していきたい。