卒業論文　　2003年度 (平成15年度)

トラブルシューティングの効率化に関する研究

指導教員
村井 純
徳田 英幸
楠本 博之
中村 修
南 政樹

慶應義塾大学 総合政策学部

氏名：高橋 宏明

s00556ht@sfc.keio.ac.jp

卒業論文要旨 2003 年度 (平成 15 年度)

トラブルシューティングの効率化に関する研究

本研究では, トラフィック量に影響を及ぼすトラブルのトラブルシューティングの効率化をはかるために, 新しいトラフィックモニタリングシステムの設計と実装を行なった.

現状のトラブルシューティングは, 1) トラブルの検知,2) トラブルの原因究明の 2 段階にモデル化できる. トラブルの検知のフェーズにおいては, 監視ツールでのトラブルの問題の切り分けが困難であるという問題がある. また, トラブルの原因究明のフェーズにおいては, 同一のトラブルに対しても同様なパラメータの選択の試行錯誤を繰り返すという問題がある.

監視ツールでのトラブルの問題の切り分けが困難であるという問題に対しては, 軸のパラメータの動的な組み合わせを動的に行なえるということで解決した. また, 同一のトラブルに対しても同様なパラメータの選択の試行錯誤を繰り返すという問題に対しては, 軸のパラメータの組み合わせを設定として保存できることで解決した.

本機構を用いることにより, 経験の浅いネットワーク管理者であっても, 熟練したネットワーク管理者と同じような見方で, 特徴的なトラフィックを効率的に見つけ出し, 原因を特定することが可能とし, トラブルシューティングの効率化を実現した.

キーワード : 1. トラブルシューティング効率, 2. トラフィックモニタリング, 3. パラメータ

慶應義塾大学 総合政策学部

高橋 宏明

Abstract of Bachelor's Thesis

Academic Year 2003

Design and Implementaion of traffic monitoring system in consideration of accumultion of network troubleshoot experience

In this research, the traffic monitoring system in consideration of accumulation of network troubleshoot experience is developed.

Current Troubleshooting is managed through following procedures. 1. Capturing traffic data 2. Understanding network condition. 3. Investigating causes of trouble. 4. Excute adequate responses to the trobule. In this case, the speed and the accuracy of trouble investigation largely depends on the experience of Network Administrator. However, in current situation, troubleshoot experience is accumulated only through individual bases. Also, As a number of nodes such as cars and information appliances, connected to Internet increases, a number of network and demand of network administrator will also increase.

In these cases, accumulation of troubleshoot experience is needed. In this system, accumulation of experience in traffic monitoring system is implemented by storing the parameter selection of a drawing axis as a configuration. Netflow of Cisco System is applied for the data needed in traffic monitoring.

Applying this system enabled accumulation of troubleshoot experiences.

Keywords : 1. Network Administration, 2. Traffic Monitoring, 3. accumulation of experience, 4. NetFlow

Keio University , Policy Management

Hiroaki Takahashi

目 次

図 目 次

表 目 次

第1章　序章

本章では, まず本研究の前提となる背景について述べ, 本研究の目的を述べる. 次に, 本研究における用語の定義を行なう. 最後に本論文の構成を述べる.

1.1　背景

インターネットの広帯域化とインターネット接続料金の低価格化により, インターネットユーザの数は急速に増加している. また, インターネットユーザの増加とともに, インターネットに接続するノードの数も増加している. ノードは計算機に限らず,PDA や携帯電話がインターネットに接続している. また, ルータやスイッチといったネットワーク機器は, 様々なベンダやプロダクトのものが存在する.

将来的に, 情報家電やマイクロノードや自動車といった様々なノードが, インターネットに接続することが可能になる. こういった状況下では,1 つの家庭や 1 台の車内といった単位において, ネットワークが形成されることとなる. ネットワークの数が増えることにより, ネットワークのユーザが増加する. それに伴い, 必要となるネットワークの管理者も増加する.

また,slammer ワームや Blaster ワームなどにより, ネットワークの安定供給を阻害する被害が頻発している. ネットワーク管理者は, ネットワークユーザに対して安定したネットワークを提供する必要がある. そのために, ネットワーク管理者はこういったトラブルに対して, 迅速に対処する必要がある. したがって, ネットワーク管理者は管理するネットワーク内のすべての機器を把握する必要がある. しかし, 先に述べたように, ネットワークに接続する機器は多岐に渡る. したがって, ネットワーク管理者の負担は大きくなっている.

1.2　目的

これまでは熟練したネットワーク管理者がネットワークを管理してきた.

これからは管理すべきネットワーク数と共に, 必要となるネットワーク管理者の数も増加する.

したがって, これからは経験の浅いネットワーク管理者がネットワークを管理する状況が発生する.

しかし, 管理すべき機器は多岐に渡り, 新種のワームによるネットワークトラブルの発生など, トラブルシューティングは困難なもとのなってきている.

経験の浅いネットワーク管理者が, ますます困難なものとなっているトラブルシューティングを行なう状況が増えることが想定される.

本研究では, 経験の浅いネットワーク管理者であっても, トラブルシューティングを効率的に行なうことができることを目的とする.

1.3 本研究における用語の定義

本節では, 本研究で用いられる用語の定義を行なう.

1.3.1 パラメータ

本研究において, パラメータとは送信元 IP アドレス, 宛先 IP アドレス, 送信元ポート番号, 宛先ポート番号,IP プロトコル番号など IP ヘッダに含まれる情報であると定義する.

1.3.2 トラブルシューティング経験

本研究におけるトラブルシューティング経験とは, トラブルの原因究明に役立つパラメータの組み合わせをいかに迅速に, また効率的に選択することができるかどうかであると定義する.

したがって, トラブルシューティング経験の蓄積とは, トラブルの原因の究明に役立つパラメータの組み合わせをネットワーク管理者の間で共有していくことである.

1.4 本論文の構成

本論文は全 7 章で構成される.

第 1 章は序章である. 第 2 章では現状のネットワーク管理, 第 3 章では問題点とその解決アプローチについて述べる. 次に, 本機構の設計, およびその実装について第 4,5 章で述べ, 第 6 章において, その正当性の評価について言及する. 最後に第 7 章で終る.

第2章　現状のトラブルシューティング

本章ではまず, 本研究で想定する環境を定義する. 次に, トラブルシューティングのモデルを示す. 最後に, ネットワークトラブルの実例をトラブルの事例ごとに見ていく.

2.1　想定する環境

本節では, まず想定するネットワーク環境を挙げ, ネットワーク管理の必要性を述べる. そして, 本研究が対象とするトラブルを定義する.

2.1.1　想定するネットワーク環境

将来的に, 情報家電やマイクロノード, 自動車といった様々なノードが, インターネットに接続することが可能となる. 家庭や車内といった単位において, ネットワークが形成されることとなる. ネットワークの数が増えることにより, ネットワークのユーザが増加する. それに伴い, 必要となるネットワークの管理者も増加する.

また, インターネットの使われ方も変化している. インターネットのトラフィックは,HTTP(Hyper Text Transfer Protocol) [2] のトラフィックが中心であったが, アプリケーションの多様化に伴い, トラフィック傾向も多様化している. WWW(World Wide Web) の閲覧と電子メールのやりとりが中心であったが, オンラインゲームやオンラインチャット, ファイル交換のように利用の仕方は多岐に渡る. インターネットの広帯域化とノードの多様化により, トラフィック傾向の多様化はますます促進されることが想定される.

2.1.2　ネットワーク管理とトラブルシューティングの必要性

ネットワークのユーザは, ネットワーク管理者に対して, 安定したネットワーク環境の提供を要求する. 要求に応えるためには, ネットワーク管理者はネットワークを構成する要素全てを完全に把握しなければならない. ネットワーク管理者は物理的な構成要素からアプリケーションだけでなく, ネットワークの運用体制に至るまですべての状態を常に監視しておく必要がある.

しかし, ネットワーク管理者が常に完璧に対応が可能であるとは限らない. 単一の人間が 24 時間ずっと万全の態勢でいることは難しく, 何らかのミスを犯すことも考えられる. また, これまでのネットワーク管理は, 熟練したネットワーク管理者によって行われてきた. 一方今後, 管理すべきネットワークが増加すると, 経験の浅い人でもネットワークを管理する必要がでてくる. こういった状況を考慮すると, ネットワーク管理者がネットワークを構成する要素すべての状態を常に監視しておくことは, 現実的に不可能である.

しかし, ネットワークに何らかのトラブルが起こった際には, 迅速に対処しなければならない. トラブルシューティングの際, ネットワーク管理者は, ネットワークに起こっている現象からト

ラブルの原因を推測する. 通常ネットワーク管理者はトラブルシューティングやメンテナンスのために, ネットワークのトラフィックやネットワークインターフェイスの情報を収集している. 実際のトラブルシューティングにおいて, ネットワークに起こっている現象を可視化するために, トラフィックモニタリングツールを用いることは必要不可欠である.

2.1.3 本研究が対象とするトラブル

ネットワーク管理者が対応しなければならないトラブルは多岐に渡る. ハードウエアの故障や,DoS(Denial of Service) 攻撃, ソフトウエアの不具合など様々なトラブルがある.

本研究が対象とするトラブルは, ネットワークに関するトラブルとする. 本研究において, ネットワークのトラブルとは, ネットワーク回線資源の枯渇を招く状態であると定義する. 例えば, 管理するネットワーク外部からの DoS 攻撃を受けている場合や, 内部の計算機がワームに感染して大量にパケットを送信している場合などが考えられる.

本研究において, トラブルといった場合, ネットワークのトラブルを中心に議論する.

2.2 トラブルシューティングのモデル

本節では, トラブルシューティングをモデル化し, 各フェーズごとに行っていることをみていく.

2.2.1 トラブルシューティングのモデル図

トラブルシューティングのモデル図を図 2.1に示す.

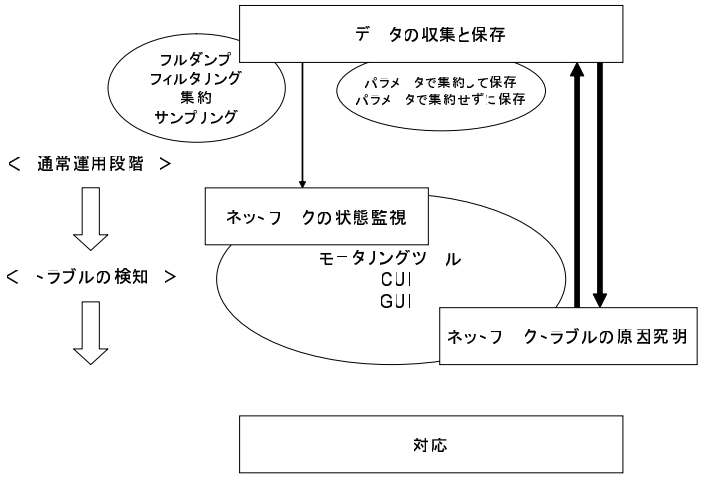

図 2.1: トラブルシューティングのモデル

トラブルシューティングの段階は大きく分けて次の 3 つに分類できる.

1. 通常運用
2. トラブルの検知

3. トラブルの原因究明

まず最初のネットワーク運用の段階では, ネットワークの状態監視に必要なデータの収集と保存を行い, ネットワークの状態を監視している段階である. データの収集と保存部分からネットワークの状態監視への矢印は, ネットワークの状態監視に必要なデータの流れを意味する. 矢印が片方向であるのは, ネットワークの状態監視側では, 動的にデータ要求の変更は行わないことを意味する.

次に, トラブルの発生段階では, ネットワークの状態監視から, ネットワークトラブルが発生したことを検知し, その原因究明を行う段階である. データの収集と保存からネットワークトラブルの原因究明への矢印は, 原因究明の際に必要なデータを意味する. 矢印が太い双方向になっているのは, 適切な原因究明を行うためには, 動的なデータ要求の変更を何度も行うことを意味する.

最後に, トラブルへの対応段階では, ネットワークトラブルの原因究明で特定したパラメータなどを基にフィルタリングを行うなどの対応を行う段階である.

また, 丸で囲んでいるものは, 四角で囲んだ各フェーズにおける具体的な実現手法の選択肢である. これについては, 次章以降で検討していく.

2.2.2 データの収集の前段階

データの収集の前段階として, 機器の設定や設置が必要となる. データの収集のためだけでなく, 普段からネットワーク管理者は, ハードウエアやソフトウエアのメンテナンスを行っている. ハードウエアとしては, ルータやスイッチ, ハブといったネットワーク機器や, サーバなどの管理を行う. また, ソフトウエアとしては, アプリケーションや OS(Operating System) のインストールやバージョンアップを行う.

2.2.3 データの収集と保存

データの収集は, ルータやサーバのように, トラブルが起こった場合にネットワークのユーザに対して安定したサービスの提供が困難になる機器に関するデータを取得する場合が多い.

ほとんどの場合, データは定常的に収集されるので, 保存の際のディスク容量が問題となる. データの集約が行われている場合が多いが, 長期にわたってデータを収集すると, データの保存に関する負荷が大きくなる. ルータやサーバのように, サービスを提供してる機器にデータを保存すると, 本来の役割であるサービスの提供自体に悪い影響を及ぼすことが考えられる. SNMP を用いた MRTG のように, 保存するデータ容量が一定である手法も存在する. しかし, データをただ収集し, 保存するだけでは価値がない. そのデータはそれを収集している人, ここではネットワーク管理者に提供されなければならない. したがって, 例え保存するデータ容量が一定であっても, データのグラフ化やデータを要求する問い合わせに対する処理をする必要が生じる. こういった処理は本来のサービスの提供に悪影響を及ぼしてしまう可能性がある. データはそれを収集している機器自体には保存せずに, データの export を行うことが多い. export されたデータの保存場所は, ネットワークの管理者により異なる.

2.2.4 ネットワークの状態監視

ネットワークの状態監視のフェーズにおいては, データの取得とその監視が行われる. ネットワーク管理者は収集したデータを用いて, 異常が起こっていないかネットワークの状態を監視する. ほとんどの場合, ネットワークの状態監視は定常的に行われるものなので, ネットワーク機器の構成などに変更がない限り, 一度監視の設定を行うと基本的にその変更は行わない. 例えば MRTG は, 最初に MIB のパラメータを設定すると, 頻繁に変更することはあまりない.

また,ping というネットワークの到達性の確認するプログラムを用いて, 管理するネットワークの外部への到達性を常に確認しておくことがある. この場合, 外部への到達性がなくなった時点で, ネットワークのトラブルを検知できる. トラブルの検知の後,ping を用いて具体的にどこまでの到達性がないかなどを特定することがあるが, この行為はすでにネットワークの状態監視のフェーズではなく, 次に挙げるネットワークトラブルの原因究明のフェーズであるといえる.

つまり, ネットワークの状態監視とは, 動的にデータを要求せずにデータの可視化を行うフェーズであるといえる. 多くのトラフィックモニタリングツールが動的にパラメータの変更ができない理由は, ネットワークの状態監視に特化して設計されているためであると考えれる.

ネットワークにトラブルが発生した場合, トラブルの検知が可能である. 多くの場合, 取得しているデータに変化が見られる. トラブルを検知すると, その原因を究明するフェーズに移行する.

2.2.5 ネットワークトラブルの原因究明

ネットワークトラブルの原因究明のフェーズにおいては, 可視化の際のパラメータを変更するといった動的な要求を行って, トラブルの原因を特定していく. ここでのデータは多くのパラメータを含んでいることが要求される. したがってデータの収集には,tcpdump のように細かい粒度でトラフィックを収集できるツールを用いる.

このフェーズでは, トラブルの要因となっているパラメータを絞り込めば絞り込むほど, 精度の高い対応が可能となる. 例えば, 特定の宛先ポートへのパケットが多いことが分かった場合を考える. この段階では, その特定の宛先ポートへのパケットは, 特定の宛先 IP アドレスへのものがほとんどを占めているか, ランダムな宛先 IP アドレスへのものなのかの切り分けは行っていない. 特定の宛先ポートへのパケットが多いということの検知だけに留まらず, どの宛先 IP アドレスへのものが多いか, さらにはどの送信元 IP アドレスからのものが多いかといった, 適切な問題の切り分けを行うことで, トラブルの原因となるパケットに対して効率的な対応が可能となる.

しかし, どのパラメータの組み合わせでデータを絞り込んでいくか, またどこまでのパラメータの絞り込みが適切であるか, といったパラメータの組み合わせの正確性や速度といったものは, ネットワーク管理者のトラブルシューティング経験に大きく依存する.

2.2.6 対応

ネットワークトラブルへの対応のフェーズでは, ネットワークトラブルの原因究明のフェーズで特定した, トラブルの原因となっているパラメータ情報を基に対応を行う.

トラブルの原因が管理するネットワークの外部からの DoS 攻撃である場合は, 境界ルータにて特定したパラメータをフィルタリングする. フィルタリングの例を, 図 2.2に示す.

```
cisco#configure terminal
cisco(config)#access-list 100 deny   53 any any
cisco(config)#access-list 100 deny   ip xxx.xxx.xxx.0 0.0.0.255 any
cisco(config)#access-list 100 deny   ip 10.0.0.0 0.255.255.255 any
cisco(config)#access-list 100 deny   ip 224.0.0.0 0.31.255.255 an(config)#access-list 100 deny   ip 127.0.0.0 0.255.255.255 any
cisco(config)#access-list 100 deny   udp any any eq sunrpc
cisco(config)#access-list 100 deny   icmp xxx.xxx.xxx.0 0.0.0.255 any echo
cisco(config)#access-list 100 deny   icmp any xxx.xxx.xxx.0 0.0.0.255 echo-reply
cisco(config)#access-list 100 deny   icmp any any timestamp-request
cisco(config)#access-list 100 deny   icmp any any mask-request
cisco(config)#access-list 100 deny   tcp any any eq telnet
cisco(config)#access-list 100 deny   udp any any eq snmp
cisco(config)#access-list 100 permit ip any any
```

図 2.2: ルータでのフィルタリングの例

また, 管理するネットワーク内部の計算機がワームなどに感染して, 大量のパケットを送信しており, 原因の究明でそのことが特定できた場合, その計算機をネットワークから切り離す.

また,DoS 攻撃やワームなど不具合をもたらすパケットがあるわけでなく, 正常に通信しているパケットによってネットワーク回線の使用率が高くなりすぎており, 原因の究明でそのことが判明した場合は, 回線の増強などの対策をとる.

ただ, ネットワークトラブルへの対応のフェーズは, ネットワーク運用の上でのポリシなどの人的要素も考慮した上でネットワーク管理者が行う. したがって, 本研究ではネットワークトラブルの原因となるパラメータの組み合わせの特定までは行うが, 対応はネットワーク管理者の判断に委ねる.

2.3 ネットワークトラブルの実例

本節では, 実際のネットワークトラブルに対して, どういった解決方法をとるのかを見ていく. ネットワークトラブルは,DoS 攻撃によるものと, ワームによるものに分類する.

2.3.1 DoS 攻撃によるもの

単純な DoS 攻撃

DoS 攻撃とは攻撃ターゲットのサーバに大量のパケットを送りつけて, サービスが提供できない状態に追い込む攻撃手法である. 単純な DoS 攻撃とは,1 台の計算機からサーバなどの特定の計算機への DoS 攻撃のことをいう. 単純な DoS 攻撃に対する, トラブルシューティングを流れを示す.

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.
- ネットワークの状態監視
  ルータのインターフェイスを流れるトラフィックの流量が異常に多くなったことを検知する.

- ネットワークトラブルの原因究明
  特定の送信元 IP アドレスから, 特定の宛先 IP アドレスへのパケットが多いことを突き止める.

- 対応
  境界ルータにて, 攻撃をしている送信元 IP アドレスからのパケットをフィルタリングする.

smurf 攻撃

smurf 攻撃とは送信元 IP アドレスを偽造した ICMP Echo Request パケットを送信することにより, ターゲットとなるホストに ICMP Echo Replay パケットを大量に送りつける攻撃方法である. smurf 攻撃の動作図を, 図 2.3に示す. 左上の攻撃者は,ICMP Echo Request パケットを送信元 IP アドレスをターゲットとなる左下の計算機の IP アドレスに偽造する. このパケットをターゲットとなるホストが属するブロードキャストアドレスに対して, 大量に送り続ける.

これにより, そのネットワーク内のすべてのコンピュータから ICMP Echo Replay パケットが, ターゲットとなる左下の計算機に対して大量に送りつけられることになる. その結果, ターゲットとなる左下の計算機は応答不能やサービス停止といった状態に陥ったり, ネットワーク内の負荷が増大するといった被害が生じる.

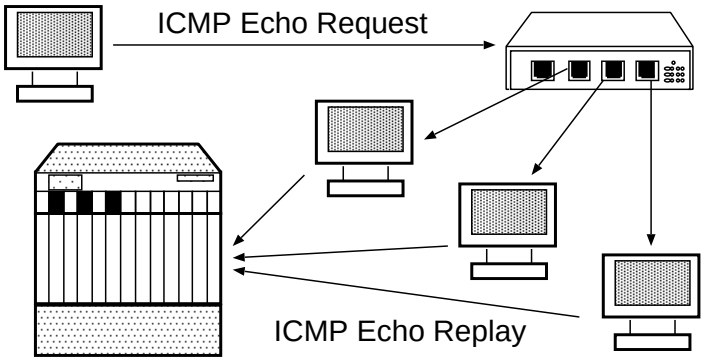

図 2.3: smurf 攻撃の動作図

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.

- ネットワークの状態監視
  ルータのインターフェイスを流れるトラフィックの流量が異常に多くなったことを検知する.

- ネットワークトラブルの原因究明
  ICMP パケットが多いことを検知する. 次に, 特定の宛先 IP アドレスへのパケットが多いことを検知する.

- 対応
  境界ルータにて, 攻撃を受けている宛先 IP アドレスとする ICMP パケットをフィルタリ

ングする. もしくは, ネットワーク内でブロードキャストアドレスに対する ICMP Echo Request をフィルタリングする.

2.3.2 SYN flooding 攻撃

SYN fooding 攻撃は WEB サーバなど TCP サービスに対して, 接続要求である SYN パケットを大量に送りつけ, そのサービスの処理能力を消費させる攻撃である. TCP はコネクション型のプロトコルであり, コネクションを確立するの際に，図 2.4で示すように，3 Way Handshake が用いられる. 右側のサーバ側は待ち受け状態にあり, そこに左側のクライアントは利用したいサービスのポートへ SYN パケットを送信する. サーバ側はそれを受け ACK パケットを送信, それと同時に SYN パケットも送信する. クライアントはそれを受けて,ACK パケットを送信してコネクションが確立されるというものである.

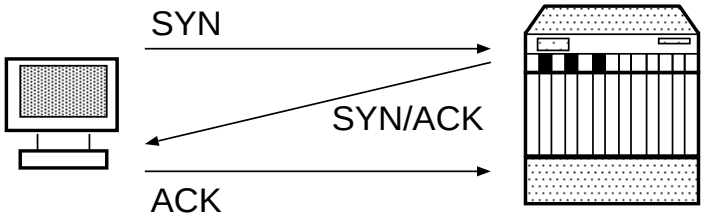

図 2.4: 3 Way Handshake

SYN flooding 攻撃は, 図 2.5で示すように，右側のターゲットに対して大量の SYN パケットを高速に送信し, サーバ側を待ち受け状態に陥れる. その間, サーバ側は送られてきた SYN パケットに対して,SYN と ACK のパケットを返し,ACK パケットを待ち受けるが, 送信された SYN パケットは IP アドレスが偽造されたパケットであり, サーバ側から SYN と ACK パケットを返す宛先は架空の IP アドレスであるため, サーバ側がいくら待っても ACK パケットは返ってこない. その間サーバ側のシステムは ACK パケットを待ち受けるために, バッファメモリ上にその情報を確保する. そのためサーバはメモリを消費し, 正常なアクセスを受けられなくなったり, システムクラッシュやサービスのダウンなどが発生する.

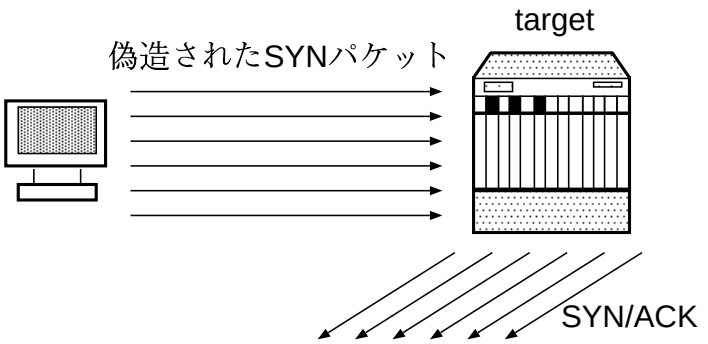

図 2.5: SYN Flooding 攻撃の概念図

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.

- ネットワークの状態監視
  攻撃を受けている WEB サーバのメモリ使用率が異常に高くなったことを検知する.

- ネットワークトラブルの原因究明
  特定の宛先 IP アドレスへのパケットが多いことを検知する. 次に, コネクションの途中である SYN_RECV という状態で止まっているものが多いことを検知する.

- 対応
  SYN flooding に対する完全な防御策はないが, 攻撃の被害の軽減は可能である. 同じ IP アドレスからの連続するある一定量以上の SYN 要求は受け付けないようにしたり, ファイアウォールで TCP 接続を代理受信して, 正常にオープンできた接続だけをサーバに渡すといった方法をとって対応する.

ポートスキャン攻撃

ポートスキャン攻撃とはネットワーク上のサーバのポートに順次アクセスを行う攻撃手法である. ネットワーク内部への侵入するためのセキュリティホールを探し出す行為であるが, ポートスキャン自体でもネットワークに負荷がかかる.

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.

- ネットワークの状態監視
  ルータのインターフェイスを流れるトラフィックの流量が異常に多くなったことを検知する.

- ネットワークトラブルの原因究明
  特定の送信元 IP アドレスからのパケットが多いことを検知する.

- 対応
  境界ルータにて, 攻撃をしている送信元 IP アドレスからのパケットをフィルタリングする.

2.3.3 ワームによるもの

SQL slammer ワーム

SQL slammer ワームとは感染したコンピュータの管理者権限を乗っ取り, ランダムな IP アドレスを宛先にして, ひたすら自分自身を送りつける. 2003 年 1 月にインターネット上で発見され, Microsoft 社の SQL Server と MSDN でバッファオーバフローを起こすために UDP ポート 1434 番を利用して感染する.

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.

- ネットワークの状態監視
  ルータのインターフェイスを流れるトラフィックの流量が異常に多くなったことを検知する.

- ネットワークトラブルの原因究明
  宛先ポート番号 1434 番への UDP パケットが多いことを検知する.

- 対応
  SQL サーバをインターネットに公開する必要がない場合は, UDP1434 ポート番をフィルタリングリングする.

Blaster ワーム

Blaster ワームの感染活動によりネットワークの帯域を消費し, 結果的にネットワーク全体のスループットが低下する. Blaster ワームはまず最初に, 任意の IP アドレスに対して, TCP の 135 番ポートをポートスキャンする. スキャン実施後, 応答があるホストに対して,DCOMRPC 脆弱点を利用したバッファオーバーフロー攻撃を試みる. 攻撃に成功すると,UDP69 番ポートを通じて,TFTP サーバを実行して, 被害ホストの TCP の 4444 番ポートを通じて msblast.exe をダウンロードするように命令を伝送する. そして msblast.exe が実行されると, それまでの過程を繰り返す.

- データの収集と保存
  ネットワークの状態監視のフェーズでは MRTG を用い, ネットワークトラブルの原因究明のフェーズでは tcpdump を用いる.

- ネットワークの状態監視
  ルータのインターフェイスを流れるトラフィックの流量が異常に多くなったことを検知する.

- ネットワークトラブルの原因究明
  管理するネットワークな内の特定の送信元 IP アドレスからのパケットが多いことを検知する.

- 対応
  対応としては, そのホストをネットワークから切り離すという手段をとる. SQL slammer ワームに感染したホストを自動的に切り離すために, 一定量の ICMP パケットを送信するホストについて, ネットワークから切り離すといった設定を行うこともある.

第3章　問題点と解決へのアプローチ

本章では, まずネットワークトラブルシューティングに用いるトラフィックモニタリングツールを機能ごとに分類する. 次に, 現状のトラブルシューティングにおける問題点を指摘する. 最後に, その問題をトラフィックモニタリングツールで解決する場合の解決アプローチを挙げる.

3.1　トラフィックモニタリングツールにおける機能

現状のネットワークトラブルに対するトラブルシューティングには, トラフィックモニタリングツールが用いられる. トラフィックモニタリングツールにおいて必要となる機能は, 以下の 3 つである.

- データの収集
- データの保存
- ユーザインターフェイス

3.1.1　データの収集

データの収集機能を実装するための手法は以下の 4 つである.

- フルダンプ
- フィルタリング
- 集約
- サンプリング

フルダンプ

フルダンプによるデータの収集は, インターフェイスで取得できるできる全てのパケットを収集できる. したがって, 送信先アドレス, 宛先アドレス, プロトコル, データ長など様々なパラメータを収集できる. しかし, 収集できるパラメータがあまりにも多いため, データの保存量が膨大になる.

フルダンプによるトラフィックモニタリングツールの一例として,tcpdump を紹介する. 図 3.1に tcpdump の出力図を示す.

```
17:47:57.645226 0800 62: 10.0.0.201.1967 > 192.168.0.130.21: S 192976724:192976
724(0) win 8192 < mss 1460,nop,nop,sackOK > (DF)
17:47:57.652837 0800 60: 192.168.0.130.21 > 10.0.0.201.1967: S 2786713294:27867
13294(0) ack 192976725 win 17520 < mss 1460 > (DF)
17:47:57.653074 0800 60: 10.0.0.201.1967 > 192.168.0.130.21: . ack 1 win 8760 (
DF)
17:47:57.712136 0800 113: 192.168.0.21 > 10.0.0.201.1967: P 1:60(59)ack 1 win 1
7520 (DF) [tos 0x10]
```

図 3.1: tcpdump 出力例

フィルタリング

フィルタリングによるデータの収集は,IP ヘッダの特定の領域のみのデータを収集したり, 特定のホスト宛てのデータのみを収集するなどといった, 収集するパラメータの数を制限した収集手法である. パラメータの数を制限しているため, フルダンプと比べてデータの保存量は削減される. しかし, ネットワークトラブルの原因となっているパラメータを含んだデータなど, 必要なデータを取りこぼしてしまう可能性が高くなる.

tcpdump には, 以下に示す項目でフィルタリングを行うことができる.

- パケット内の収集する byte 数
- 特定の IP アドレスまたは mac アドレスを送信元とするパケット
- 特定のプロトコルまたはポートを使用したパケット

集約

データを収集する際に, あるパラメータ単位で保存する手法である. 例えば, ある単位時間ごとにデータを収集し, その単位時間内の全データの時間的意味を同一化するなどといった包括関係にある情報を上位集合に組み込む手法をトラフィックの集約と呼ぶ.

集約手法では, 収集したデータの持つそれぞれのパラメータの粒度を落とすことによって, 細かな差異を隠蔽化し, 収集したデータの保存量を削除することができる.

集約手法を用いたモニタリングツールとして代表的なものに,MRTG がある. MRTG を紹介するとともに,MRTG がデータの収集に用いている,SNMP も一緒に紹介する.

SNMP(Simple Network Management Protocol) は，ネットワークの情報管理プロトコルである．本プロトコルによってネットワーク機器それぞれが MIB(Managemanet Infomation Base) の形式に基づいて収集したデータを収集，管理できる．図 3.2に SNMP の概要図を示す.

SNMP は，各機器ごとにデータを収集，保存する機構を持っている．SNMP を使用し情報の要求を行った場合，各機器それぞれが MIB からデータを抽出し収集したデータを要求元に返信する.

SNMP を使用して分散した各ネットワーク機器が保持するデータを，管理者が一元的に収集することで情報の収集，解析を行える．そのため，動的にネットワーク機器の情報を収集することが容易となる.

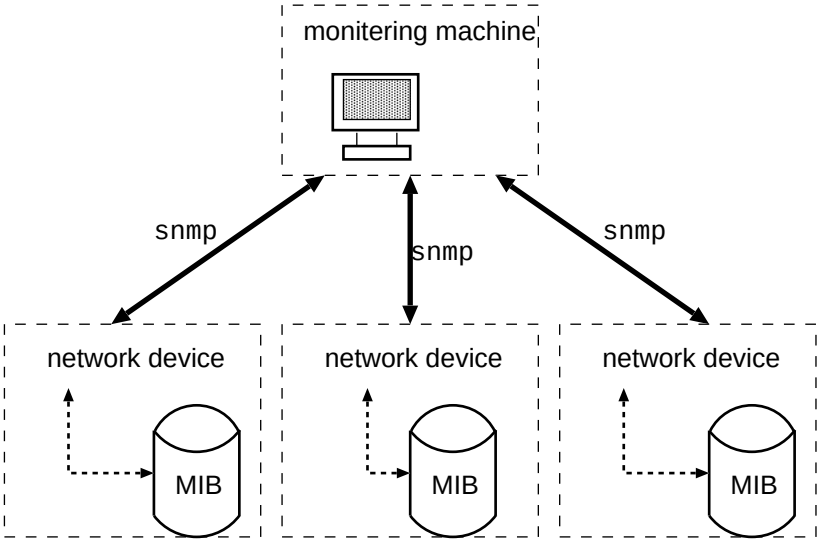

図 3.2: SNMP 概要図

サンプリング

サンプリングによるデータの収集とは,10000 パケット中 1 つのパケットのデータを収集したり, ランダムにパケットを収集するなどといった, すべてのパケットを母数とみなし, その中から標本抽出する収集手法である.

この手法では, 多くのパラメータを含んだまま, データの保存量を削減することができる.

サンプリングによるパケットの収集手法として,NetFlow を紹介する. NetFlow は Cisco Systems 社製のルータを通過したパケットの収集を行い, 以下に示すデータを含んだトラフィック情報を生成する.

- データ量
- 送信元 IP アドレス, 宛先 IP アドレス
- 送信ポート番号, 受信ポート番号

NetFlow は,N 個に 1 個の割合でパケットを抽出し, 抽出したパケットの中から上記のパラメータなどを含んだ UDP パケットを送信する. 抽出の割合は設定可能であり, 取得できるパラメータは NetFlow のバージョンによって異なる.

この NetFow の送信したデータを利用したソフトウエアとして代表的なものに CAIDA の開発した cflowd がある. cflowd の概要図を, 図 3.3に示す.

データの収集手法の比較

データの収集手法を比較したものを, 表 3.1に示す.

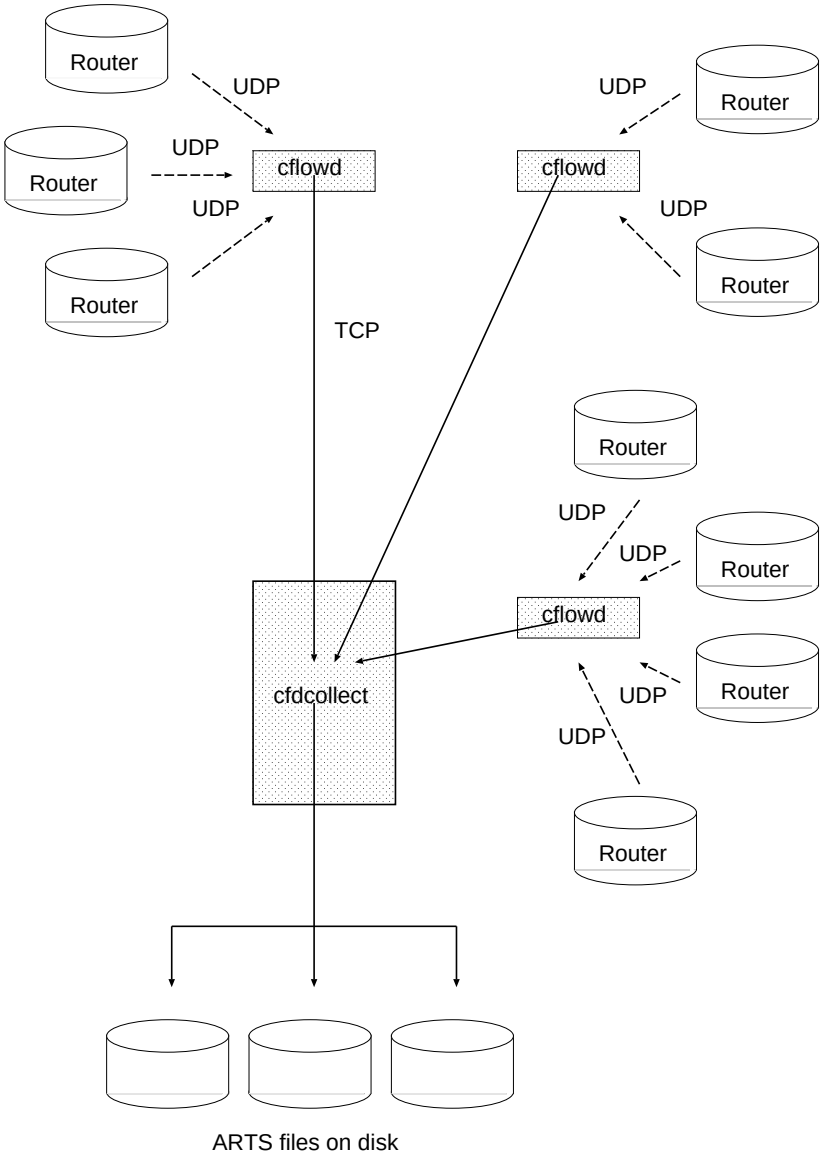

図 3.3: cflowd 概要図

表 3.1: データの収集手法の比較

	パラメータの数	データの保存量
フルダンプ	多い	多い
フィルタリング	少ない	少ない
集約	少ない	少ない
サンプリング	多い	少ない

3.1.2 データの保存

データの保存機能を実装するための手法は以下の 2 つである.

- パラメータで集約して保存
- パラメータで集約せずに保存

パラメータで集約して保存

基本的な動作概要は, データの収集の際のものと同じである. データの保存量は減るが, パラメータの組み合わせへの対応は柔軟でなくなる.

パラメータで集約せずに保存

収集したデータを, パラメータで集約することなく, そのままの形式で保存する手法である. データの保存量は多いが, パラメータの組み合わせへの対応は柔軟である.

データの保存手法の比較

データの保存手法を比較したものを, 表 3.2に示す.

表 3.2: データの保存手法の比較

	パラメータの組み合わせへの対応性	データの保存量
パラメータで集約して保存	柔軟でない	少ない
パラメータで集約せずに保存	柔軟	多い

3.1.3 ユーザインターフェイス

ユーザインターフェイスの実装手法は以下の 2 つである.

- CUI(Character User Interface)
- GUI(Graphical User Interface)

CUI

CUI はユーザに対する情報の表示を文字によって行う. CUI を用いたモニタリングツールとしては, tcpdump をあげることができる. CUI で見るデータは視覚的に分かりにくいが, パラメータの変更には柔軟に対応できる.

GUI

GUI はユーザに対して情報の表示をグラフ化によって行う. GUI を用いたモニタリングツールとしては, MRTG をあげることができる. GUI で見るデータは視覚的には分かりやすいが, 一般的にパラメータの変更に柔軟でない.

モニタリング手法の比較

モニタリング手法を比較したものを, 表 3.3に示す.

表 3.3: モニタリング手法の比較

	パラメータの変更への対応性	視覚的な見やすさ
CUI	柔軟	見にくい
GUI	一般的に柔軟でない	見やすい

3.2 現状のトラブルシューティングにおける問題点

現状のトラブルシューティングにおける問題点を挙げる.

1. 監視ツールでの問題の切り分けが困難
2. 同一のトラブルに対しても同様なパラメータの選択の試行錯誤を繰り返すこと

それぞれの問題点について, 図 3.4を見ながら説明していく.

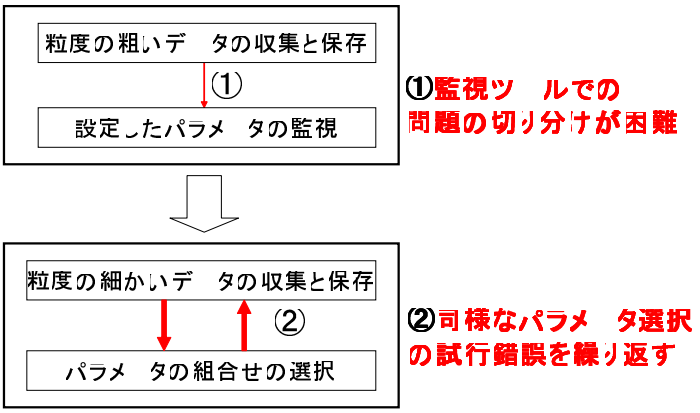

図 3.4: トラブルシューティングの問題点

3.2.1 監視ツールでの問題の切り分けが困難

1 つ目の問題点は, トラブルの検知のフェーズにおいて, 監視ツールでの問題の切り分けが困難であることである. 図 3.4では,「粒度の粗いデータの収集と保存」と「設定したパラメータの監視」の間の部分で表現した.

監視ツールでトラブルを検知し, そのままトラブルシューティングのフェーズに移行することが最も効率的であるといえる. トラブルの検知のフェーズで用いるツールとトラブルの原因究明のフェーズで用いるツールは, 共にデータの可視化を行う. しかし, 現状の監視ツールでは粒度の粗いデータしか必要としていない.

したがって, 共通の機能を持ち合わせているのにもかかわらず, 前章で見たネットワークトラブルの実例からも分かるように, 監視ツールと同じツールやデータを用いて問題の切り分けを行なうことは困難である.

3.2.2 同一のトラブルに対しても同様なパラメータ選択の試行錯誤を繰り返す

2 つ目の問題点は, トラブル原因究明のフェーズで, 同一のトラブルに対しても同様なパラメータ選択の試行錯誤を繰り返すことである. 図 3.4では, 「粒度の細かいデータの収集と保存」と「パラメータの組み合わせの選択」の間の部分にあたる. 同一のデータであっても, トラブルシューティングの際にトラブルの原因究明に役立つデータになるか否かは, 選択するパラメータの組み合わせによる. 前章で見たネットワークトラブルの実例では, ネットワークトラブルの原因を適切に検知した結果だけを記した. しかし実際には, ネットワーク管理者はトラブルが起こった際には, ネットワークに限らず様々な要素を原因として疑う. また, ネットワークトラブルであると検知できたとしても, データに対するパラメータの組み合わせの選択には試行錯誤を繰り返す. どのパラメータの組み合わせでデータを絞り込んでいくか, またどこまでのパラメータの絞り込みが適切であるか, といったパラメータの組み合わせの正確性や速度といったものは, ネットワーク管理者のトラブルシューティング経験に大きく依存する. しかし, 現状ではトラブルシューティング経験を蓄積する機構が存在しない. したがって, 同様なパラメータ選択の試行錯誤を繰り返されており, 効率的なトラブルシューティングが行なわれているとは言えない.

3.3 問題点に対する解決アプローチ

各問題点を解決するための必要要件を挙げる. 本研究では, ネットワークトラブルシューティングに用いるトラフィックモニタリングツールで, 2 つの問題点を解決する.

3.3.1 複数のパラメータ選択

監視ツールでの問題の切り分けが困難である問題への解決アプローチの 1 つ目である.

描画の際に, パラメータを複数選択できるようにする.

これにより, トラブルが起こった際に, どのパラメータがトラブルの原因となっているのか究明が可能となる.

3.3.2 特徴が目立つような描画

監視ツールでの問題の切り分けが困難である問題への解決アプローチの 2 つ目である.

描画の際, トラフィック異常などの特徴が目立つように描画できるようにする. 本研究の目的はトラブルシューティングの効率化である. 効率的に問題を切り分けるためには, 視覚的にかつ感覚的にトラブルシューティングを行なうことが好ましい.

3.3.3 トラブルシューティング経験の蓄積

同一のトラブルに対しても同様なパラメータ選択の試行錯誤を繰り返す問題の解決へのアプローチである.

パラメータ選択の試行錯誤の経験を, トラブルシューティング経験として蓄積できるようにする. トラブルシューティング経験の蓄積が可能になると, 過去に行なったトラブルシューティングと同じ見方で見ることができる. これにより, パラメータ選択の試行錯誤の繰り返しが軽減され, トラブルシューティングが効率化される.

第4章　設計

本章では，まず本機構の動作概要図を示す．次に, 前章で示した必要要件ごとの機構とその機能を整理する．そして, それぞれの機能の実現を考慮して設計する．最後に, 各機構の機能と設計をまとめる．

4.1　動作概要図

本機構の機能とデータの流れからなる，システム動作概要図を図 4.1に示す．

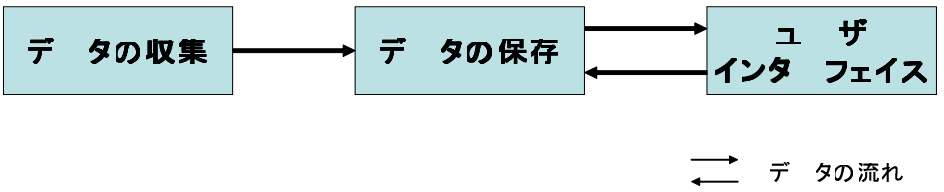

図 4.1: 動作概要図

データの収集場所からデータを収集する．収集されたデータは蓄積され，データを可視化する際のデータの要求に対して必要なデータを返答する．トラブルの原因究明に役立つパラメータの組合せをいかに迅速に, また効率的に選択することができるかがトラブルシューティング経験である．

4.2　複数のパラメータ選択機構

複数のパラメータ選択機構とは, データに対して絞り込みを行なうパラメータを複数選択できる機構である．複数のパラメータ選択機構の機能は, 次の 2 つである．

- 変更可能な数のパラメータを含んでいること
- パラメータで集約しないこと

変更可能なパラメータには少なくとも, 送信元 IP アドレス, 宛先 IP アドレス, 送信元ポート番号, 宛先ポート番号, IP プロトコル番号のいわゆる 5tuples を含んだデータであることが要求される．

また, 複数のパラメータで集約して, 描画する．その際, 何らかのパラメータで集約してしまうと, 集約したパラメータでの再描画が非常に困難となる．

したがって, 保存するデータは時間などで集約すべきでない．

4.3 特徴抽出描画機構

特徴抽出描画機構とは, 描画の際にネットワークトラブルなどの異常がみられる部分を, 特徴的に描画する機構である.

特徴が目立つような描画機構の機能は, 次の 2 つである.

- グラフ化できること
- 短時間で描画できること

本研究の目的は, 効率的なトラブルシューティングの実現である. 目的達成のためには, 視覚的かつ感覚的なトラブルシューティングが必要となる.GUI は CUI と比較して, 感覚的に把握しやすい. したがって,GUI を選択する.

次に, 画面の状態遷移である. 画面の状態遷移に関して, 次の 3 つのステップを設計する.

1. トラブルの検知
2. トラブル原因の絞り込み
3. トラブル原因であるかの検証

それぞれのステップの遷移は, グラフをクリックすることで行なう.

トラブルの検知を目的とした最初のステップでは, X 軸に時間をとり,Y 軸にバイト量もしくはパケット数をとる. このステップではトラブルが起こっていることを検知し, グラフの中で特徴のある部分のデータをクリックすることでパラメータで絞り込み, 特徴ある部分の原因を究明していく.

次の, トラブル原因の絞り込みを目的としたステップでは, X 軸のパラメータを動的に選択可能にする. クリックしたトラフィックの異常部分に対して, パラメータの組み合わせで絞り込みを行ない, トラフィック量の多い順に並べかえる. もし選択したパラメータの組み合わせがトラフィック異常の原因となるトラフィックを含んでいる場合, 上位のパラメータの組み合わせのグラフが大きく他を引き離す. 例えば,SYN flooding 攻撃が行なわれた場合, 宛先ポート番号と宛先 IP アドレスの組み合わせで絞り込みを行なった場合, トラフィックがランキング上位に偏る. 逆に, ランキングが上位に偏らない場合, パラメータの組み合わせは適切であるといえない. したがって, ネットワーク管理者が本機構を用いてトラブルシューティングをする際, ランキングが上位に偏るようなパラメータの組み合わせを見つけるために試行錯誤する.

また, 短時間で描画をするためには, データの問い合わせにかかる時間と描画にかかる時間を短縮する必要がある. 描画に必要なデータの要素数を大きく減らすことは困難であり, 極端に要素数を減らしてしまった場合, 見にくいグラフになってしまい, グラフ化の本来の目的である, 感覚的な把握というものが困難となる. したがって, 短時間で描画するためには, データの問い合わせにかかる時間を短縮する必要がある. データの問い合わせにかかる時間を短縮するためには, データの効率的な検索を考慮すべきである. しかし, 本機構では何らかのパラメータで集約して保存することはしない. したがって, データの収集時にデータ量を減らす必要がある. 候補に挙がっていた, フルダンプによるデータの収集とサンプリングによるデータの収集では, サンプリングの方がデータ量が少ない. したがって, 本機構ではサンプリングを用いる.

4.4 トラブルシューティング経験の蓄積機構

トラブルシューティング経験の蓄積とは, データの絞り込みを行なうパラメータの組み合わせを設定として保存する機構である. トラブルシューティング経験の蓄積機構の機能は, 以下の 2 種類の設定ファイルが追記可能な状態で存在することである.

- 既知のトラブルであるか検証するためのパラメータが記されたもの
- 過去の経験を基にして絞り込むためのパラメータが記されたもの

以上の 2 つの設定ファイルを考慮して, ユーザに対して次の 3 つのモードを設計する.

1. known mode
2. experience mode
3. custom mode

設定ファイルを用いるものは, パラメータの組み合わせを読み込む experience mode と, それに加えてパラメータの値まで読み込む known mode を設計する. known mode では既知のトラブルであるかの検証を行なう. 例えば, 特定のホストに対する DoS 攻撃が多い場合, そのホストへのトラフィック量の推移と, 総トラフィック量の比較を行なう. つまり,known mode では先に挙げた 3 つのステップの最後のステップのみを行なう.

experience mode では, トラブルの原因を絞り込む. トラブルシューティングの最初の手順を期待しており, それまでの経験をもとにトラブルの原因究明につながる, 有効なパラメータの見方であると判断されたパラメータの組み合わせでデータを見る.

custom mode では, 設定ファイルを用いず, 独自に描画の軸のパラメータの選択を行なうモードである. このモードで, 最終的な原因を究明する. 選択したパラメータは設定ファイルに保存可能である.

4.5 各機構の機能と設計のまとめ

本節では, 各機構の機能と設計をまとめる.

4.5.1 各機構の機能のまとめ

各機構の機能をまとめる.

- 変更可能な数のパラメータを含んでいること
- パラメータで集約しないこと
- グラフ化できること
- 短時間で描画できること
- 2 種類の設定ファイルの存在

4.5.2 設計のまとめ

上に挙げた機能を満たすトラフィックモニタリングシステムの設計をまとめる.

- サンプリングでのデータ収集
- パラメータで集約せずに保存
- GUI

データの収集は, サンプリングを用いる. これは, 「変更可能な数のパラメータを含んでいること」と「短時間で描画できること」の 2 つの機能を考慮した.
データの保存は, パラメータで集約せずに保存する. これは,「パラメータで集約しないこと」の機能を考慮した.
ユーザインターフェイスは,GUI を用いる. これは,「グラフ化できること」の機能を考慮した.
画面の遷移は,「2 種類の設定ファイルの存在」の機能を考慮した. 画面遷移図を図 4.2に示し, 各画面を説明する.

1. パラメータの組み合わせを設定ファイルから読み込むか, ネットワーク管理者がパラメータの組み合わせを選ぶかを選択
2. 既知のトラブルの検知用の設定ファイルを読み込むか, 原因究明のステップ用の設定ファイルを読み込むかを選択
3. パラメータの組み合わせを選択
4. 設定ファイルからパラメータの組み合わせを選択
5. 横軸に時間, 縦軸にバイト数をとる粒度の粗いグラフを描画
6. 選択したパラメータの組み合わせで, バイト数が大きいものから並べたグラフを描画
7. 選択したパラメータの値で絞ったグラフと, 粒度の粗いグラフを比較して描画
8. 選択したパラメータを設定として保存

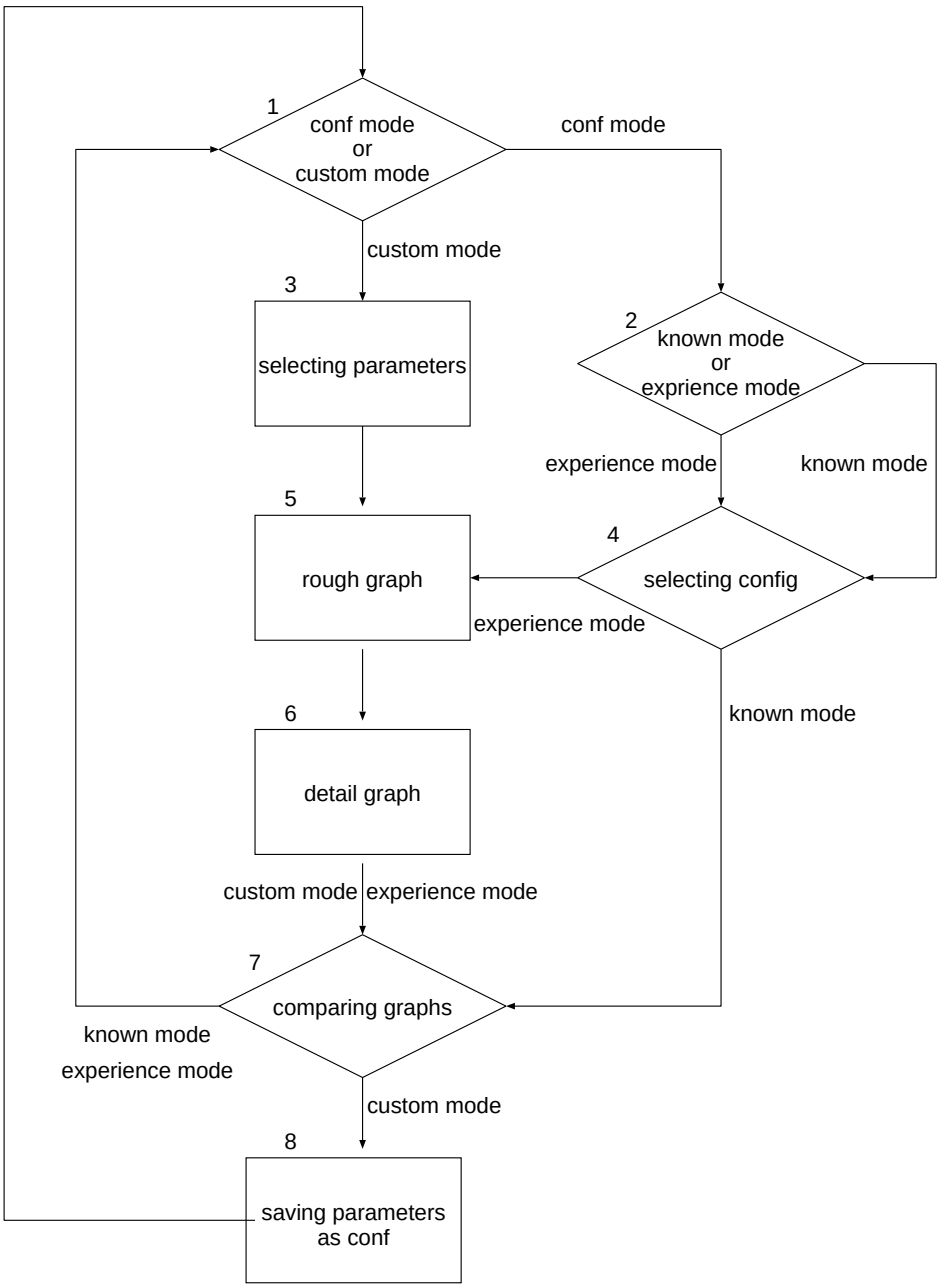
1
conf mode
or
custom mode
conf mode
custom mode
3
selecting parameters
2
known mode
or
exprience mode
experience mode
known mode
5
rough graph
4
selecting config
experience mode
known mode
6
detail graph
custom mode
experience mode
7
comparing graphs
known mode
experience mode
custom mode
8
saving parameters
as conf

図 4.2: 画面遷移図

第5章　実装

本章では, トラブルシューティング経験の蓄積を考慮したトラフィックモニタリングシステムの実装について述べる.

5.1　システム構成図

本機構のシステム構成を図 5.1に示す.

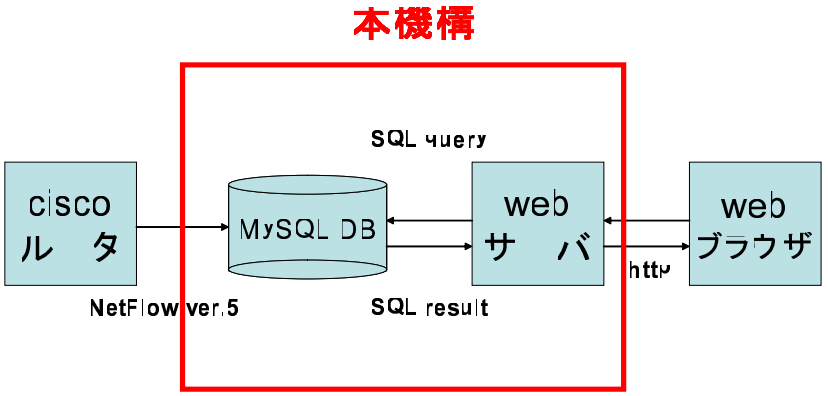

図 5.1: システム構成図

5.2　実装環境

実装環境について述べる. NetFlow データの export を行なう Cisco Systems 社のルータの IOS のバージョンは, 12.1(19)E1 である. また,NetFlow データを受信して保存し, 描画は同一の PC を用いた.

NetFlow パケットの受信部分のプログラム,MySQL データベースサーバ, SQL 結果の描画部分のプログラムは 1 つの計算機上で動作している. その計算機の構成を構成は, 表 5.1に示す.

5.3　収集するデータ

収集するデータとして,Cisco Systems 社製のルータに実装されている機能である, NetFlow version5 を用いた. 慶應義塾大学の境界ルータである cisco11.fujisawa.wide.ad.jp から NetFlow データの export を行なった. このルータの IOS のバージョンは 12.1(19)E1 である. NetFlow データは export を行なうインターフェイスにて, 図 5.2に示すコマンドにて設定する.

表 5.1: 動作プラットホーム

OS	FreeBSD 5.1-RELEASE
CPU	Intel Celeron 1.4GHz
Memory	256MB
Hard Disk	120GB
Database	MySQL version4.0.15
Web Server	apache version2.0.47
Language	PHP4, C 言語

```
#ip route-cache flow
#ip flow-export 203.178.143.104 version 5 peer-as
```

図 5.2: NetFlow version5 の flow export 設定例

5.4 NetFlow データ受信部分

NetFlow データ受信部分の実装は C 言語で行なった. 受信した NetFlow データは, header 部分と payload 部分のそれぞれを構造体に格納した. この 2 つの構造体を図 5.3と図 5.4に示す.

```
struct netflowv5_header{
        uint16_t  version;      /* flow-export version number */
        uint16_t  count;        /* number of flow entries */
        uint32_t  sysUptime;    /* system up time */
        uint32_t  unix_secs;    /* real time in unix time format */
        uint32_t  unix_nsecs;   /* real time in unix time format */
        uint32_t  flow_sequence;/* sequence number */
        uint8_t   engine_type;  /* no VIP = 0, VIP2 = 1 */
        uint8_t   engine_id;    /* VIP2 slot number */
        uint16_t  reserved;     /* unused */
}
```

図 5.3: NetFlow version 5 header 構造体

5.5 データベース部分

データベース部分は MySQL version4.0.15 を用いた. 受けとった NetFlow データを SQL の INSERT 文に変換して, データベースへの挿入を行なった. データベースは様々なパラメータを組み合わせた問い合わが想定される. したがって, データベースのテーブルにおいて, 特定のパラメータで集約を行なうべきではない. 本機構では, 表 5.2に示すフォーマットで, テーブルを作成した.

```
struct netflowv5_payload{
        uint32_t  srcaddr;      /* source IP address */
        uint32_t     dstaddr;   /* destination IP address */
        uint32_t     nexthop;   /* next hop router's IP address */
        uint16_t     input;     /* input interface index */
        uint16_t     output;    /* output interface index */
        uint32_t     pkts;      /* packets sent in duration */
        uint32_t     bytes;     /* octets sent in duration */
        uint32_t     first;     /* SysUptime at start of flow */
        uint32_t     last;      /* and of last packet of flow */
        uint16_t     srcport;   /* TCP/UDP source port number */
        uint16_t     dstport;   /* TCP/UDP destination port number */
        uint8_t      pad;
        uint8_t      tcp_flags; /* bitwise OR of all TCP flags in flow */
                                /*  for non-TCP flows */
        uint8_t      prot;      /* IP protocol, e.g., 6=TCP, 17=UDP */
        uint8_t      tos;       /* IP Type-of-Service */
        uint16_t     src_as;    /* originating AS of source address */
        uint16_t     dst_as;    /* originating AS of destination address */
        uint8_t      src_mask;  /* source address prefix mask bits */
        uint8_t      dst_mask;  /* destination address prefix mask bits */
        uint16_t     reserved;
}
```

図 5.4: NetFlow version 5 payload 構造体

表 5.2: 作成した MySQL のテーブル

fid	bigint (20) NOT NULL auto_increment
unix_secs	int(11) default NULL
srcaddr	int(10) unsigned default NULL
destaddr	int(10) unsigned default NULL
nexthop	int(10) unsigned default NULL
input	int(10) unsigned default NULL
output	int(10) unsigned default NULL
pkts	int(10) unsigned default NULL
bytes	int(10) unsigned default NULL
srcport	smallint(5) unsigned default NULL
dstport	smallint(5) unsigned default NULL

5.6 描画部分

WEB サーバを介して描画を行なう. WEB サーバは apache version2.0.47 を利用した. WEB クライアントとのデータのやりとりと, MySQL サーバへの SQL クエリの問い合わせには, PHP4 を用いた. 特に描画に関しては, グラフをクリックすることで新たに次のグラフを描画可能な, Advanced Software Engineering 社 [22] が提供している PHP 用のコンポーネントである, chart-director [23] を用いた.

5.7 ユーザインターフェイス

5.7.1 3 つのステップ

パラメータを選択した後のインターフェイスは, 次の 3 ステップを実装した.

- トラブルの検知
- トラブル原因の絞り込み.
- トラブルの原因であるかの検証

まず, トラブルの検知ステップのグラフを図 5.5に示す.

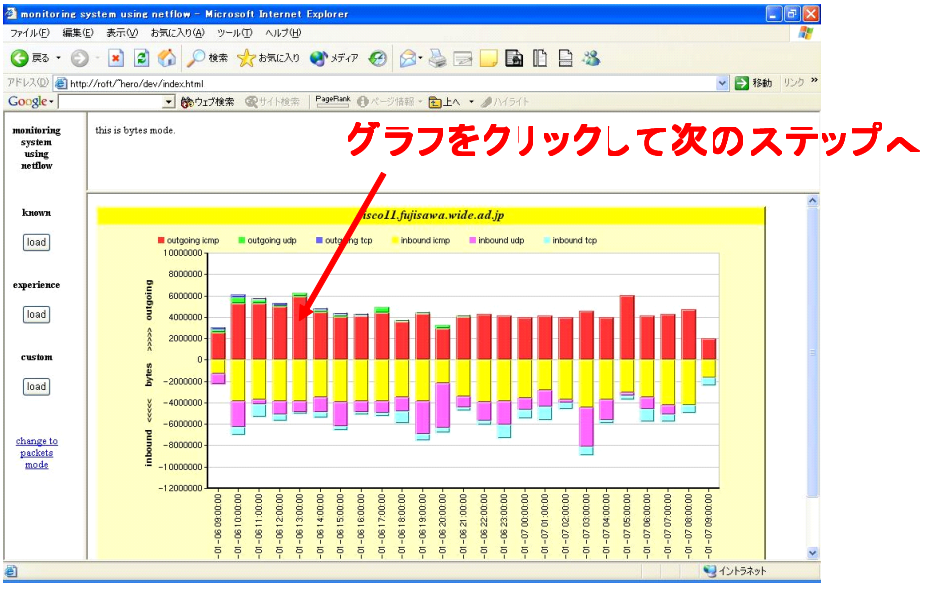

図 5.5: トラブルの検知ステップ

縦軸はトラフィックのバイト数であり, 横軸は最近 24 時間の時間である. 上に伸びている棒グラフは, 境界ルータの内側のネットワークから外側のネットワークへのトラフィックをプロトコル毎に積み上げたものである. 反対に, 下に伸びている棒グラフは, 境界ルータの外側のネットワークから内側のネットワークへのトラフィックをプロトコル毎に積み上げたものである. このグラフは主に, ネットワークトラブルの検知に用いる.

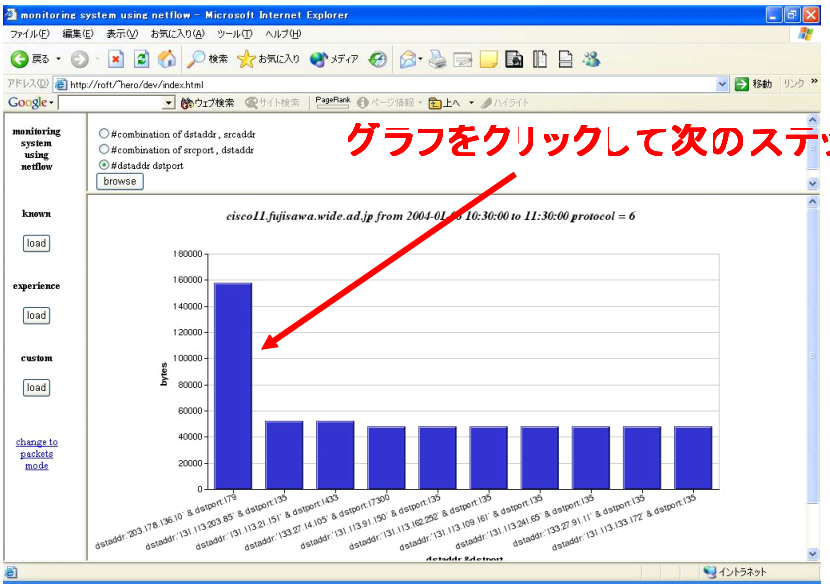

図 5.6: トラブル原因の絞り込みステップ

次に, トラブル原因の絞り込みステップのグラフを図 5.6に示す.

縦軸はトラフィックのバイト数である. 横軸は設定ファイルから読み込んだ, もしくはネットワーク管理者が選択したパラメータの組み合わせでグループ化した場合の, トラフィック量が多いものから並べたものである. ネットワークトラブルが起こっている際に, 適切なパラメータの組み合わせを選択した場合, トラフィック量が多いものと少ないものが極端であるといったグラフが描画される. 図 5.6で用いた SQL のクエリ文の 1 つを図??に示す.

```
SELECT sum(bytes) ,
DATE_FORMAT(from_unixtime(unix_secs),
'%y-%m-%d%%H:00:00') as time , unix_secs
FROM flow Where srcaddr > inet_aton('131.113.0.0')
and srcaddr < inet_aton('131.113.255.255') and prot = 6
and unix_secs >=unix_timestamp() - 60*60*24
or srcaddr > inet_aton('133.27.0.0')
and srcaddr < inet_aton('133.27.255.255') and prot = 6
and unix_secs >=unix_timestamp() - 60*60*24 GROUP BY time
```

図 5.7: SQL のクエリ文

最後に, トラブルの原因であるかの検証ステップのグラフを図 5.8に示す.

縦軸はトラフィックのバイト数であり, 横軸は最近 24 時間の時間である. このグラフでは, 粒度の細かいデータを用いたグラフが特定のパラメータの値をとるものに偏っている場合などに, それがネットワークトラブルの原因であるかの検証を行なう. 選択したパラメータの組み合わせと値で全体のトラフィックをグループ化した時のグラフと, 全体のトラフィックのグラフとの

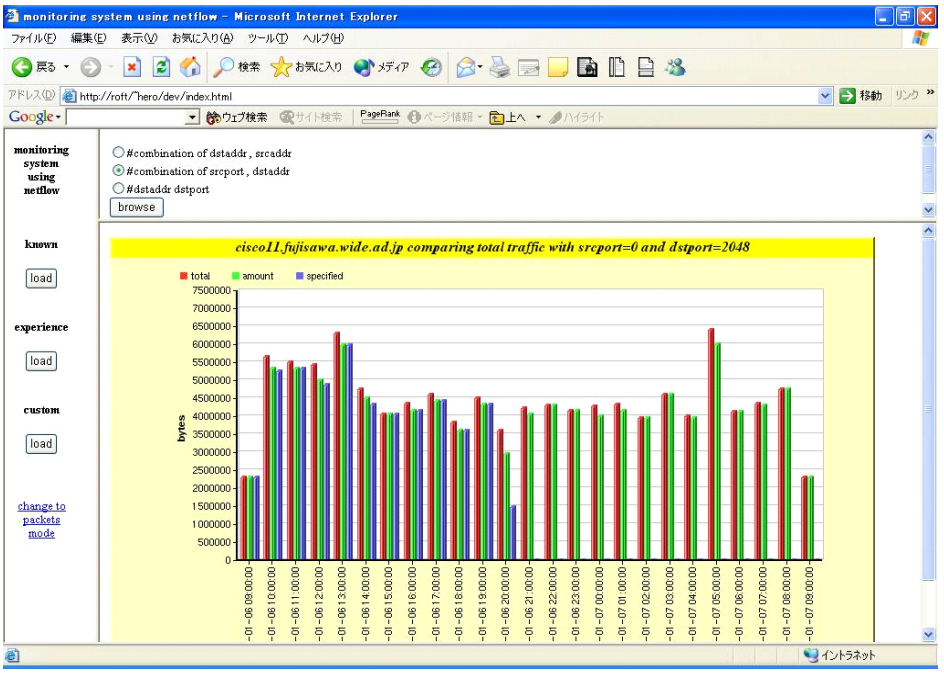

図 5.8: トラブルの原因であるかの検証ステップ

比較を行なうことができる. もし 2 つのグラフが同一の形状をとり, トラフィックの量も近い場合は, 選択したパラメータの組み合わせと値がネットワークトラブルの原因を有している可能性が高い. 縦軸はトラフィックのバイト数であり, 横軸は最近 24 時間の時間である. このグラフでは, 粒度の細かいデータを用いたグラフが特定のパラメータの値をとるものに偏っている場合などに, それがネットワークトラブルの原因であるかの検証を行なう. 選択したパラメータの組み合わせと値で全体のトラフィックをグループ化した時のグラフと, 全体のトラフィックのグラフとの比較を行なうことができる. もし 2 つのグラフが同一の形状をとり, トラフィックの量も近い場合は, 選択したパラメータの組み合わせと値がネットワークトラブルの原因を有している可能性が高い.

5.7.2 Y 軸の選択

本研究で対象とするトラブルを検知するためには, バイト量もしくはパケット数の変化をみていればよい. 本機構では,Y 軸をバイト量とパケット数のいずれかを選択できる. Y 軸の変更は, 図 5.9に示すように, 画面の左下で変更する. Y 軸を変更しても, 他のステップは同様である.

5.7.3 3 つのモード

選択するパラメータによって, 次の 3 つのモードを実装した.

- known mode
- experience mode

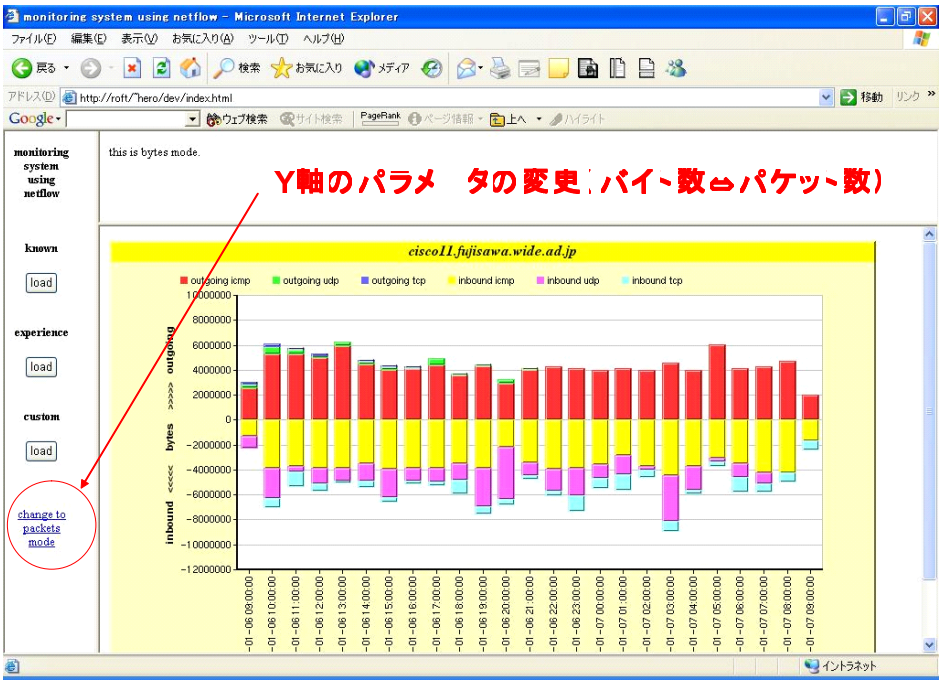

図 5.9: Y 軸の変更

- cusctom mode

1 つ目の known mode では, 既知のトラブルであるかの検証を行なう. パラメータで絞り込んだトラフィック量と全体のトラフィック量を比較する. したがって,known mode 用の設定ファイルには, パラメータの値まで指定してある SQL クエリが保存されている.

2 つ目の experience mode では, 過去の経験を基に, トラブルの絞り込みに有効なパラメータの組み合わせで, データを絞り込む. したがって,experience mode 用の設定ファイルには, パラメータの要素しか記述されていない.

3 つ目の custom mode では, 様々なパラメータの組み合わせの選択が可能である. また, パラメータの組み合わせを設定として保存可能である. パラメータの保存のインターフェイスは図 5.10である.

図 5.10の画面で, 本機構のユーザであるネットワーク管理者が有益であると判断した場合, 設定を保存する. その際,known mode として設定するか, experience mode として設定するかを選択することができる. また, 設定ファイルは複数人のネットワーク管理者で共有されることを想定している. したがって, 保存するパラメータの組み合わせによって, どういったグラフが得られるか, また誰が保存するかといった情報をコメントとして書き込む.

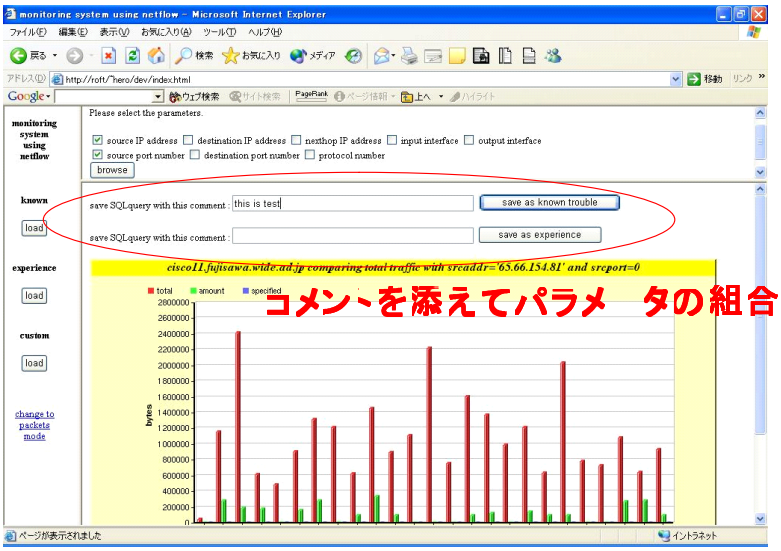

図 5.10: パラメータの組み合わせの保存画面

第6章　評価

本章では, 本機構の評価について述べる.

6.1　実データの収集に関して

図 6.1に示すような構成で, 本機構の評価実験を行った. cisco11.fujisawa.wide.ad.jp という, WIDE バックボーンと慶應義塾大学の学内ネットワークの間の境界ルータから,NetFlow version5 のデータを export した.

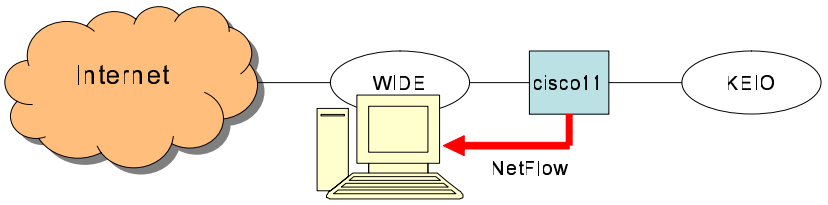

図 6.1: 実験構成図

本研究では, 機能要件にしたがって, 評価項目を以下のように定めた.

- 少ないディスク保存容量
- 選択したパラメータの設定保存可能
- パラメータの動的変更可能
- 複数のパラメータを選択して描画可能
- 描画に要する時間が短い

6.2　少ないディスク保存容量

保存容量の使用容量を計測する.

MySQL では, データベース中にテーブルを 1 つ作るたびに, 3 つのファイルが作成される. frm という拡張子のテーブル構造ファイル, MYD という拡張子のデータファイル, MYI という拡張子のインデックスデータファイルの 3 つである. 2004 年 1 月 5 日 5 時から 2004 年 1 月 13 日 17 時までの保存容量は 3 つのファイルを合計して 23M バイトであった. したがって,1 日あたりに使用する保存容量は約 2.7M バイトである. 一方,cisco11.fujisawa.wide.ad.jp を通過する総トラフィック量は, 約 135G バイトである. データ量は 5 万分の 1 になっており, ディスク保存容量は少ないといえる.

6.3 選択したパラメータの設定保存可能

選択したパラメータの設定保存可能であることを検証する.

本機構ではパラメータの組み合わせを設定ファイルに保存している. 設定ファイルはネットワーク管理者ごとに分けるのではなく, 共通のファイルに追記している. したがって, 本機構において複数人でのトラブルシューティング経験の蓄積が可能であるといえる.

6.4 パラメータの動的変更可能

パラメータの動的変更が可能であることを検証する.

本機構では, パラメータをあらかじめ設定しておく必要がない. パラメータの組み合わせをweb ページの遷移ごとに変更することができる.

したがって, 本機構ではパラメータの動的変更可能であるといえる.

6.5 複数のパラメータを選択して描画可能

複数のパラメータを選択して描画可能であるという機能要件に対する評価としては, 何通りのパラメータの取り方が可能であるかを検証する. X 軸のパラメータの要素としては, 以下の 8 つが挙げられる.

1. 送信元 IP アドレス
2. 宛先 IP アドレス
3. nexthopIP アドレス
4. intput インターフェイス
5. output インターフェイス
6. 送信元ポート番号
7. 宛先ポート番号
8. IP プロトコル番号

Y 軸のパラメータの要素としては, 以下の 8 つが挙げられる.

1. バイト量
2. パケット数

X 軸のパラメータの要素の組み合わせの取り方は,

$$2^8 - 1 = 255 \tag{6.1}$$

である.

次に,Y 軸のパラメータの要素の組み合わせの取り方は, 2 通りである.

したがって, 描画の際のパラメータの組み合わせの選択の仕方は,

$$255 \times 2 = 510 \tag{6.2}$$

で 510 通りである.

それに対して,SNMP を用いて収集することができる MIB 形式のデータを MTRG で描画する場合と比較する. X 軸のパラメータの要素としては, 以下の 5 つが挙げられる.

1. 送信元 IP アドレス
2. 宛先 IP アドレス
3. 送信元ポート番号
4. 宛先ポート番号
5. IP プロトコル番号

MRTG ではパラメータを組み合わせての描画はできないので, X 軸として取り得る軸の数は 5 通りである.

Y 軸のパラメータの要素としては, 以下の 2 つが挙げられる.

1. バイト量
2. パケット数

したがって, Y 軸として取り得る軸の数は 2 通りである.

したがって, 描画の際のパラメータの組み合わせの選択の仕方は,

$$5 \times 2 = 10 \tag{6.3}$$

で 10 通りである.

本機構では複数のパラメータを組み合わせて選択することができる. また,MTRG の描画の際のパラメータの組み合わせの選択の仕方が 10 通りであるのに対して, 本機構のパラメータの組み合わせの数は 510 通りであり, 非常に柔軟な描画が可能であるといえる.

6.6 描画に要する時間が短い

データベースへの問い合わせ時間の計測を行なった. 本機構のユーザが最初に見る画面である, X 軸に時間,Y 軸にバイト量をとる画面を表示するのに問い合わせる SQL クエリの応答時間を計測した. 100 回計測を行ない, 平均時間, 最長時間, 最短時間のぞれぞれを算出して, 表 6.1に示した.

6.7 有効なトラフィックの見方

本機構で有効であると考えられるパラメータの組み合わせを考察する. 以下に挙げるものは,known mode もしくは experience mode としての保存が考えられる.

表 6.1: データベースへの問い合わせ時間

average	1.12sec
max	2.71sec
min	0.25sec

6.7.1 ICMP を用いるワーム

MS blast や nachi のようなワームは, 感染したホストから特定のネットワークアドレス内の宛先 IP アドレスに対して,ICMP パケットを送信する. MS blast や nachi のようなワームを発見するのに有効なパラメータの組み合わせを挙げる.

- IP プロトコル番号
- 送信元 IP アドレス

この 2 つのパラメータの組み合わせでトラフィックをみることによって, MS blast や nachi のようなワームの影響がみられる場合, 特定の送信元 IP アドレスからのトラフィックが増加する.

6.7.2 TCP SYN flooding 攻撃

SYN flooding のような攻撃は, 攻撃対象となりうるホストが web サーバに限定される. したがって, 以下のようなパラメータの組み合わせが有効である.

- IP プロトコル番号
- 宛先ポート番号
- 宛先 IP アドレス
- パケット数

この 4 つのパラメータの組み合わせでトラフィックをみることによって, SYN flooding 攻撃が起こった場合,web サーバの TCP ポート 80 番宛へのパケット数が非常に多くなる.

6.7.3 UDP slammer ワーム

slammer ワームは, 感染したホストからランダムな宛先 IP アドレスの宛先ポート番号 1434 番に対して,UDP パケットを大量に送信する. slammer ワームを発見するのに有効なパラメータの組み合わせを挙げる.

- IP プロトコル番号
- 宛先ポート番号
- 送信元 IP アドレス

この 3 つのパラメータの組み合わせでトラフィックをみることによって, slammer ワームに感染したホストがネットワークに影響を及ぼしている場合, 特徴的な形状のグラフとなる.

6.8 まとめ

本研究のそれぞれの評価項目についてみていく.

- 少ないディスク容量

  MySQL データベースへの保存量は, フルダンプする時のトラフィックと比較して 5 万分の 1 になっており, ディスク保存容量は少ないといえる.

- 選択したパラメータの設定保存可能

  known mode 用と experience mode 用のそれぞれの共通の設定ファイルを追記可能な状態にしており, 選択したパラメータの設定保存可能であるといえる.

- パラメータの動的変更

  web ページの画面遷移ごとに, 動的に選択されたパラメータの組み合わせを用いて描画している. したがって, パラメータの動的変更可能であるといえる.

- 複数のパラメータを選択して描画可能

  本機構では複数のパラメータを組み合わせて選択することができる. また,MTRG の描画の際のパラメータの組み合わせの選択の仕方が 10 通りであるのに対して, 本機構のパラメータの組み合わせの数は 510 通りであり, 非常に柔軟な描画が可能であるといえる.

- 描画に要する時間が短い

  データベースへのクエリの応答時間の平均は 1.12 秒であり, 描画に要する時間は短いといえる.

本機構では, 評価項目をすべて満たしていることが確認できた. また, いくつかのネットワークトラブル原因についての検知手法を示した.

第7章　結論

7.1　まとめ

様々なノードがインターネットに接続し, トラフィック傾向が多様化した現在, トラブルシューティングにおいてトラフィックモニタリングツールは非常に重要である. なぜならネットワークのトラブルが発生した場合, ネットワーク管理者はモニタリングツールを用いて, ネットワークに起こっている現象を把握し, トラブルの原因を究明するからである. 本研究では, トラフィック量に影響を及ぼすトラブルのトラブルシューティングの効率化をはかるために, 新しいトラフィックモニタリングシステムの設計と実装を行なった.

現状のトラブルシューティングは, 1) トラブルの検知,2) トラブルの原因究明の 2 段階にモデル化できる. トラブルの検知のフェーズにおいては, 監視ツールでのトラブルの問題の切り分けが困難であるという問題がある. また, トラブルの原因究明のフェーズにおいては, 同一のトラブルに対しても同様なパラメータの選択の試行錯誤を繰り返すという問題がある.

監視ツールでのトラブルの問題の切り分けが困難であるという問題に対しては, 軸のパラメータの動的な組み合わせを動的に行なえるということで解決した. また, 同一のトラブルに対しても同様なパラメータの選択の試行錯誤を繰り返すという問題に対しては, 軸のパラメータの組み合わせを設定として保存できることで解決した.

本機構を用いることにより, 経験の浅いネットワーク管理者であっても, 熟練したネットワーク管理者と同じような見方で, 特徴的なトラフィックを効率的に見つけ出し, 原因を特定することが可能とし, トラブルシューティングの効率化を実現した.

7.2　今後の展望

本機構では独自にトラブルシューティング経験の蓄積を行なった. しかし, 経験の蓄積を行なっているトラブルチケットや snort などと共通のインターフェイスを用いて連係することで, より効率的に経験が蓄積されることが想定される.

また, 多地点でのデータを収集することで, 各地点でのトラフィック傾向を比較できるであろう.

謝辞

本研究を進めるにあたり，御指導を頂きました，慶応義塾大学環境情報学部教授の村井純博士に感謝致します．また，絶えず御助言と御指導を頂きました，同学部の南政樹氏，インターネット総合研究所の許先明氏，奈良先端科学技術大学院大学情報科学センターの中村豊氏に深い感謝の意を表します．

海崎良氏，小原泰弘氏，三屋光史朗氏，日野哲志氏，岡田耕司氏をはじめとする慶應義塾大学政策メディア研究科の方々，久松剛氏をはじめとする慶應義塾大学環境情報学部の村井・徳田・中村・楠本・南研究室の諸氏，並びに中尾嘉宏氏，藤井聖氏をはじめとする奈良先端科学技術大学院大学情報科学研究科の方々には，本論文執筆の上で様々な議論をして頂き，また温かい励ましを頂きました．ここに，深い感謝の意を表します．

最後に，2 年半の長きに渡って苦楽を共にした，清水崇史氏，谷岡洋平氏，廣瀬峻氏，橋本和樹氏，成瀬大亮氏をはじめとする同期諸君に最大限の感謝の意を表します．

以上を持って謝辞と致します．

参考文献

[1] AT&T Research, Coffman, Odlyzko 2000

[2] R. Fielding, et al. “Hypertext Transfer Protocol – HTTP/1.1”, RFC2616, June 1999

J.D. Case, M. Fedor, M.L. Schoffstall, J. Davin. ”Simple Network Management Protocol(SNMP)”, RFC 1157, May 1990

[3] H. Zimmermann ”OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection,” IEEE Transactions on communications COM-28, No. 4, April 1980

[4] International Organization for Standardization, http://www.iso.ch/

[5] The Multi Router Traffic Grapher, http://people.ee.ethz.ch/ oetiker/webtools/mrtg/

[6] tcpdump, http://www.tcpdump.org/

[7] NetFlow, http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml

[8] sFlow, http://www.sflow.org/

[9] J.D. Case, M. Fedor, M.L. Schoffstall, J. Davin. ”Simple Network Management Protocol(SNMP)”, RFC 1157, May 1990

[10] M.T. Rose, K. McCloghrie. ”Concise MIB definitions”, RFC 1212, Mar 1991

[11] InMon Corporation, http://www.inmon.com/

[12] Cisco Systems, http://www.cisco.com/

[13] Juniper Networks, http://www.juniper.net/

[14] Extreme Networks, http://www.extremenetworks.com/

[15] Hewlett-Packard Development Company, http://www.hp.com/

[16] Hitachi Ltd, http://www.hitachi.co.jp/

[17] snort, http://www.snort.org/

[18] Symantec Corporation, http://www.symantec.com/

[19] CAIDA, http://www.caida.org/

[20] R.Needham "Denail of Service: An Example", Communications of the CACM volume 37, November 1994

[21] Crannog, http://www.crannog-software.com/

[22] Advanced Software Engineering Limited, http://www.advsofteng.com/

[23] chartdirector, http://www.advsofteng.com/product.html