修士論文要旨 2002 年度 (平成 14 年度)

ネットワーク運用における DoS 攻撃に関する研究

本研究では，flooding 攻撃の情報を詳細に収集する機構である AGURI を作成し，運用ネットワーク上において flooding 攻撃のトラフィックを収集した。更に，収集した flooding 攻撃を分析することで，flooding 攻撃に対する効果的な対策の提言を行った。

収集データの解析により，flooding 攻撃は 1) 攻撃先のホスト，2) 攻撃手法，3) 時間的推移において特徴を持つことがわかった。これらの結果によって，ネットワーク運用者は警戒するポイントを大きく絞り込め，実際にある攻撃を受けた場合に次の攻撃を予測することができる。また，flooding 攻撃対策ソフトウェアの作成に関しても，ルールセットのグループ化といった，現状の flooding 攻撃の現状に即した設計を可能にした。

本論文の成果により，社会基盤となるインターネットを flooding 攻撃に対して強固にすることができる。通信経路上において flooding 攻撃によるネットワーク資源占有を効果的に防止し，もし攻撃による被害が出たとしても，迅速な対応と，予測による次の攻撃の対策によって，ユーザに対しより快適なネットワーク環境の提供を可能にした。

キーワード
1. flooding 攻撃, 2. ネットワーク運用, 3. トラフィック収集, 4. AGURI, 5. インターネット

慶應義塾大学 政策・メディア研究科

海崎 良

Abstract of Master's Thesis

Academic Year 2002

Detection and prevention of DoS attacks for network administratior

In this resesarch, analysis and countermeasure of flooding attacks using AGURI is shown. AGURI is a tool to detect, collect, and analyze network flows.

Analyzed data showed three prominent characteristics of flooding attacks : 1) victim host, 2) way of attack, and 3) uniqueness in time transition. Discovering those characteristics, AGURI gives alert to network administrators. Thus, quick response to the attack could be taken. Also, analyzed data shows that flooding attacks are sometimes predictable

By this research, prevention of network bandwidth exaustion caused by the flooding attacks has become possible. Also, this research proved that AGURI is a usefull tool supporting network administrators take quick contermeasures for flooding attacks.

Keywords :
1. flooding attack, 2. network management, 3. traffic mesurement, 4. AGURI,5. Internet

Keio University , Media and Governamce

Ryo Kaizaki

修士論文　　2002年度 (平成14年度)

ネットワーク運用におけるDoS攻撃に関する研究

慶應義塾大学 政策・メディア研究科

氏名：海崎 良

主査

慶應義塾大学 村井 純

副査

SonyCSL 長 健二朗

慶應義塾大学 中村 修

目 次

第 1 章　はじめに **1**
1.1 背景：DoS 攻撃とは 1
1.2 本研究の目的 2
1.3 本論文の構成 2

第 2 章　flooding 攻撃と対策 **3**
2.1 flooding 攻撃対応の流れ 3
2.2 flooding 攻撃を可能とする技術的背景 5
2.2.1 パケット交換網 5
2.2.2 ソースアドレス詐称 7
2.3 パケット交換網を補完する転送技術 7
2.3.1 End-End による輻輳回避 7
2.3.2 途中ルータによる輻輳回避 8
2.3.3 特定トラフィックの優先処理 8
2.4 ソースアドレス詐称への対策 9
2.4.1 ingress filter 9
2.4.2 IP traceback 10
2.5 トラフィック収集機構の必然性 13
2.5.1 トラフィック収集の技術的問題点 13
2.5.2 既存の手法 13
2.5.3 flooding 攻撃対応に適したトラフィック収集 18

第 3 章　AGURI の設計と実装 **21**
3.1 設計 21
3.1.1 データ収集手法 22
3.1.2 データ集約手法 24
3.1.3 データ蓄積手法 28
3.2 実装 29
3.2.1 データ収集部 29
3.2.2 データ集約部 32
3.2.3 データ蓄積部 34
3.3 ネットワーク運用への適応 36
3.3.1 長期にわたるトラフィック収集 36

3.3.2 トラフィックの視覚化 37
3.3.3 まとめ：AGURI の機能概要 41

第 4 章 実トラフィックの収集 42
4.1 実データの収集に関して 42
4.1.1 WIDE インターネット 43
4.1.2 データ収集ポイント 43
4.1.3 データ収集マシンの計算機環境 44
4.2 収集データ 45
4.2.1 期間 45
4.2.2 収集データの種別 46
4.2.3 収集データの長期的特徴 47

第 5 章 flooding 攻撃の傾向分析 52
5.1 収集したデータにみられる flooding 攻撃 52
5.1.1 flooding 攻撃の定義 52
5.1.2 flooding 攻撃の発生頻度 54
5.2 flooding 攻撃の特徴 55
5.2.1 攻撃先 IP アドレス 56
5.2.2 DDoS 攻撃の検出 58
5.2.3 攻撃手法 60
5.2.4 時間的推移 62
5.3 分析結果と提言 63

第 6 章 まとめ 64

図 目 次

2.1 flooding 攻撃への対応の流れ 3
2.2 回線交換網 6
2.3 パケット網 6
2.4 ingress filter 概念図 10
2.5 IP traceback 概要図 (DoS 攻撃) 11
2.6 IP traceback 概要図 (攻撃元追跡) 11
2.7 tcpdump 出力例 15
2.8 SNMP 概要図 16
2.9 cflowd 概要図 19

3.1 IPv4 ヘッダ 22
3.2 IPv6 ヘッダ 23
3.3 libpcap の機構概要 24
3.4 収集した IP アドレスの持つ階層的な意味 25
3.5 パトリシアツリー構造へのデータの追加 1 27
3.6 パトリシアツリー構造へのデータの追加 2 27
3.7 パトリシアツリーへの LRU を用いたデータの追加 1 27
3.8 パトリシアツリーへの LRU を用いたデータの追加 2 27
3.9 異なる時間粒度のサマリ作成 28
3.10 ネットワークインターフェースからのトラフィック収集 29
3.11 libpcap を利用したデータリンク層の隠蔽化関数 30
3.12 収集した第 3 層，第 4 層のデータを格納する構造体 30
3.13 tcpdump ファイルからのトラフィック収集 30
3.14 tcpdump ファイルから時間情報抽出と取り扱い 31
3.15 AGURI データからのトラフィック収集 32
3.16 AGURI データの切り分け 32
3.17 ツリー中のノード構造体 33
3.18 ツリー全体の構造体 33
3.19 パトリシアツリー構造を操作する関数 34
3.20 データ収集間隔の設定 35
3.21 AGURI のプロファイリング結果出力例 35
3.22 cron による定期的な AGURI データの蓄積 36

3.23 agurify.pl による AGURI データの階層的蓄積 37
3.24 AGURI による gnuplot サポート 38
3.25 AGURI データ参照用フォーム 39
3.26 web ブラウザ越しの AGURI データ参照 40

4.1 データ収集ポイント 43
4.2 AGURI データ（ポート番号）の出力例 47
4.3 データ A soure ip address 49
4.4 データ A destination ip address 49
4.5 データ A source port 49
4.6 データ A destination port 49
4.7 データ B source ip address 49
4.8 データ B destination ip address 49
4.9 データ B source port 50
4.10 データ B destination port 50
4.11 データ C (2001 年) source ip address 50
4.12 データ C (2001 年) destination ip address 50
4.13 データ C (2001 年) source ip address 50
4.14 データ C (2001 年) destination ip address 50
4.15 データ C (2002 年) source ip address 51
4.16 データ C (2002 年) destination ip addres 51
4.17 データ C (2002 年) source port 51
4.18 データ C (2002 年) destination port 51

5.1 2 回の flooding 攻撃例 54
5.2 分散した攻撃先 IP アドレスへの flooding 攻撃 58
5.3 データ A における DDoS 攻撃 59
5.4 データ C における DDoS 攻撃 59
5.5 TCP を用いた攻撃のポート番号の分散 61
5.6 flooding 攻撃対象の時間経過による推移 62

表 目 次

1.1 DoS 攻撃の分類 1

2.1 MIB アクセス項目数試算 17

4.1 収集マシンの計算機環境 44
4.2 データ A の AGURI データ量 45
4.3 データ B の AGURI データ量 46
4.4 データ C の AGURI データ量 46
4.5 長期 AGURI データ一覧表 48

5.1 データ A における flooding 攻撃発生件数 55
5.2 データ B における flooding 攻撃発生件数 55
5.3 データ C における flooding 攻撃発生件数 56
5.4 攻撃対象別 flooding 攻撃発生件数 57

第1章　はじめに

本章では，まず本研究の前提となる背景について述べ，次に，本研究の目的を述べる．最後に本論文の構成を述べる．

1.1　背景：DoS 攻撃とは

インターネットにおける Denial of Service(以下，DoS) 攻撃とは，大量のデータを送信することで，対象となるホストまたは途中経路のネットワーク資源を使い尽し，サービスを妨害および拒否する攻撃である．

DoS 攻撃は大きく分て，logic 攻撃と flooding 攻撃の 2 種類に分類される．(1)　この分類を以下の表 1.1に示す．

表 1.1: DoS 攻撃の分類

	攻撃手法	被害対象
DoS 攻撃	logic 攻撃	Software(分類 A)
		Operating System(分類 B)
	flooding 攻撃	ホストの計算機資源 (分類 C)
		ネットワーク資源 (分類 D)

- logic 攻撃

  Operating System やソフトウェアのセキュリティホールにつけこみ，システムの停止やサービスの低下を引き起こす攻撃である．この logic 攻撃は OS やソフトウェアをアップデートする，もしくは，特定パターンのパケットをフィルタリングすることによって防ぐことができる．

- flooding 攻撃

  大量のパケットを送信することによって，ネットワーク資源やホストの計算機資源を占有しサービス不能状態にする攻撃である．この flooding 攻撃は，一つ一つのパケットは通常のトラフィックと比較した場合に特徴がなく，ただパケット量が多いだけである．そのため，logic 攻撃と同等の手法で flooding 攻撃を防ぐこと

はできない．更に flooding 攻撃の被害を受ける対象で分類した場合，ホストの計算機資源とネットワーク資源の 2 つに分類することができる．

現在，logic 攻撃を検知，警告，対処する分野は，　攻撃パターンが出るたびにルールセットが追加されるといういたちごっこのような状態になっている．　更に，既にいつくかは商品として発売されていて，いかに迅速に攻撃パターンを見つけて，対策ルールセットをアップデートするかという局面にいたっている．

逆に，flooding 攻撃に対応する分野は，検知，対策ともまさにさまざな研究者が取り組んでいる問題である．更に，flooding 攻撃の検知を考慮した場合，ホストの計算機資源を対象にした攻撃の場合はホストにおけるデータ収集によって検知できる場合が多い，しかし，ネットワーク資源を対象にした攻撃の検知は，ネットワーク上におけるデータ収集とデータの多様性の問題を抱えているため非常に難しい．

1.2 本研究の目的

インターネットが日常生活のための情報基盤となった現在，flooding 攻撃への対策は非常に重要である．なぜならば，flooding 攻撃を受けてネットワーク資源が占有されてしまった場合，そのネットワークを経由する全ての通信が障害を受けてしまい，非常に広範囲の被害を被ってしまうからである．

そこで，本研究では，flooding 攻撃によってネットワーク資源が被害をうける場合を対象とする．(表 1.1中の分類 D)．

現状では flooding 攻撃の情報を詳細に取ることはできない，そのため，flooding 攻撃の実体には不明点が多く，現状では flooding 攻撃に対する効果的な対策が存在しない．

そこで，本研究では，1)flooding 攻撃の情報を詳細に収集する機構の作成, 2) 収集した flooding 攻撃の情報の分析, 3)flooding 攻撃に対する効果的な対策の提言の 3 点を目的とする．具体的には，ネットワーク上においてトラフィックを収集するソフトウェアの設計，実装を行い，実際のネットワークから長期に渡ってデータを収集し，収集したデータの分析を行い，分析によって見つけ出した flooding 攻撃の特徴から，効果的な対策を提言する．

1.3 本論文の構成

本論文では，第 2章にて，現状の問題点を述べ関連研究の整理を行う．第 3章にて，データ収集機構の設計と実装に関して述べる．第 4章にて，実データ収集に関して述べ，第 5章にて，収集したデータの分析を行う．最後に第 6章にて本研究のまとめを行う．

第2章　flooding攻撃と対策

本章においては，flooding攻撃が起こった場合のネットワーク運用者の流れと，その過程における対策に関して示す．また，トラフィック収集機構の必然性について示す．

2.1　flooding攻撃対応の流れ

現在，flooding攻撃によってネットワーク資源が占有されてしまった場合，基本的には，1) 攻撃の検知，2) 被害対象の特定，3) 攻撃特性の分析，4) 機器の設定変更の4つの手順が踏まれる．この手順の流れを図2.1に示す．

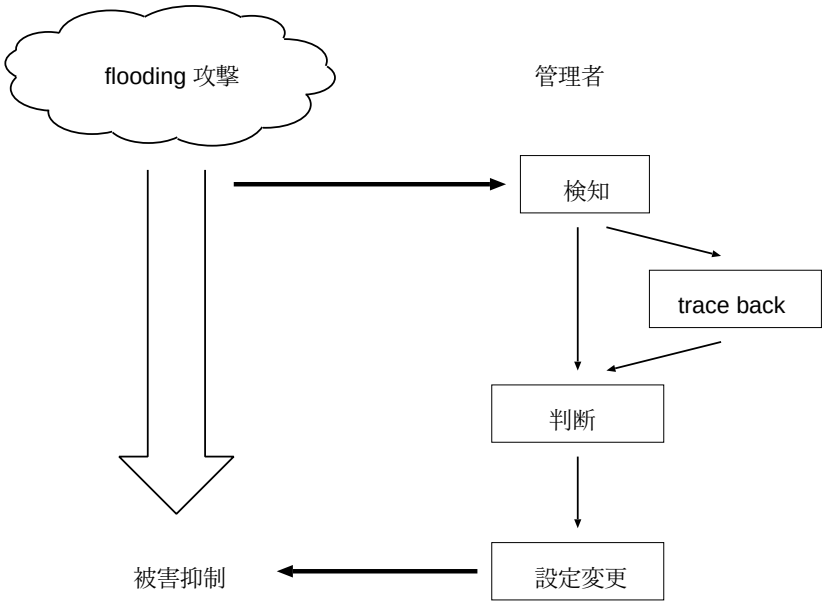

図 2.1: flooding攻撃への対応の流れ

図2.1に示した”traceback” は攻撃特性の分析の一部である，攻撃元ホストの追跡を表わしている．しかし，ネットワーク運用者にとっては，原因を追求するよりも，症状をできるだけ押さえる方が重要である場合が多いため，現状のネットワーク運用の現場では省かれる場合が多い．

これら4つの手順すべてを踏むことによって，flooding攻撃への対策を行うことができる．しかし，現状では，この対策はほとんど体系化できておらず，ネットワーク運用者によるその場しのぎの対策が取られる場合が多く，研究分野になっている．この現状を以下にまとめる．

- 攻撃の検知

  ネットワーク運用者の管理しているネットワークが flooding 攻撃による被害を受けた場合，ネットワーク運用者もしくはユーザからの連絡によって，特定のホストへの到達性が無いという症状が発見される．

  この時点では，ネットワーク運用者は，この症状が flooding 攻撃によって引き起こされたかどうか判断できない．同様の症状は，ルータやスイッチなどの中継器の故障や設定ミス，もしくは，回線障害でも十分起りうるからである．

  到達性が無いという症状に対して，ネットワーク運用者は，原因を追求する第一歩として，具体的な症状の洗いだしを行う．具体的には，traceroute(2) による症状が発生している個所の絞り混みに始まり，ping(3) による中継器への到達性の確認，中継器上で，接続拠点のステータスの確認，もしくは，経路情報の確認を行う．

  更に，既に当該中継器に MRTG(4) などのトラフィック量測定機構が備えてあった場合には，トラフィック流量の急激な増加から，備えて無い場合には，interface の流量を監視することによって，flooding 攻撃であると推測することができる．

- 被害対象の特定

  被害対象の特定は，攻撃の検知の際に行われた traceroute などと重複する場合が非常に多い．しかし，ネットワーク運用者が traceroute によって検知できる被害対象は，トポロジ的にもっとも手前の機器や回線であるため，その先に被害を受けている機器や回線が存在するかどうか不明である．

  つまり，現状では，被害対象を全て特定できるわけではないため，トポロジ的にもっとも手前の機器や回線の対策を行った場合でも，その先の機器や回線も被害を受けていてため，到達性が回復しない場合がある．

- 攻撃特性の分析

  攻撃特性の分析とは，攻撃先ホストの特定や，攻撃元ホストの特定，または，攻撃に使用されている手法などを指す．これらの情報は，中継器上では基本的に収集することができない．予め，データ収集用のマシンを設置していたり，中継機器において，ネットワーク運用者が設定しておいたデータを収集することはできるが，全ての中継器に予め想定される設定を行うことは非常にコストが高いため難しい．

  また，これらのデータを収集できたとしても，攻撃元ホストの情報は簡単に詐称することができるため，IP traceback などの現在，研究開発中の技術を用いる必要がある．（小節 2.4.2にて後述）

- 機器の設定変更

  機器の設定変更を行うためには，前提となる攻撃特性の分析が必要となる．現状では基本的に，機器の設定変更は flooding 攻撃に用いられている攻撃先ホストへ

のパケットを中継器でフィルタリングもしくは転送量を制限する手法を用いる．そのため，flooding 攻撃の特性が判明していない場合は，ネットワーク運用者としては手の打ちようが無い状況になる場合が多い．

更に，被害を受けている機器や回線が自組織の場合は，攻撃特性分析の後に機器設定を行うことができる．しかし，自組織の管理下に無い場合は，当該組織に連絡をとって対策してもらう必要がある．

これらの，パケットフィルタリングもしくは転送量制限を行うのとは異なる手法で flooding 攻撃の被害を抑制する研究も現在すすめられている．(2.3節にて後述)

この現状を踏まえ，本章では，まず，flooding 攻撃を可能とする技術的背景について触れ，その後に，flooding 攻撃への対策の手順となる，4 つの手順に関して議論を行う．

2.2 flooding 攻撃を可能とする技術的背景

現状のインターネットでは flooding 攻撃を容易に引き起こすことができる，それ加え flooding 攻撃への対策は非常に難しい．

その要因は，大きくわけて 2 つ存在する．第一に，ホスト毎やサービス毎のデータ送出量の制限がない点，第二に，データ転送に際してデータ送信元ホストを照会されない点である．

前者は，パケット交換網が本質的に内包している弱点である．後者は，データ転送の際に，中継器への負担を減らすというインターネットの設計思想に基づく弱点である．

2.2.1 パケット交換網

インターネットはパケット交換網の上に成り立っているため，基本的にネットワーク資源は共有されている．

図 2.2に回線交換網の概念図，図 2.3にパケット交換網の概念図を示す．

回線交換網では，データを送受信する場合，最初にエンドノード間の回線資源を確保する．そのため，確保した回線資源分しかデータを送信することができない．つまり，一つのホストがネットワーク資源を占有するためには，予め全てのネットワーク資源を確保しなければならず，大規模な帯域の確保は通常許されない．

一方，パケット交換網では，データ送受信を行う場合，エンドノード間の回線資源は共有される．そのため，比較的低コストでネットワーク資源を利用することができる．

言い換えると，データ送信の際にホスト毎に対するネットワーク資源割り当てを管理していない．

これは，データ転送時には宛先 IP アドレスのみ参照され，送信元 IP アドレスの情報は参照されない．そのため，ネットワーク資源を送信元 IP アドレスごとに管理することができないためである．

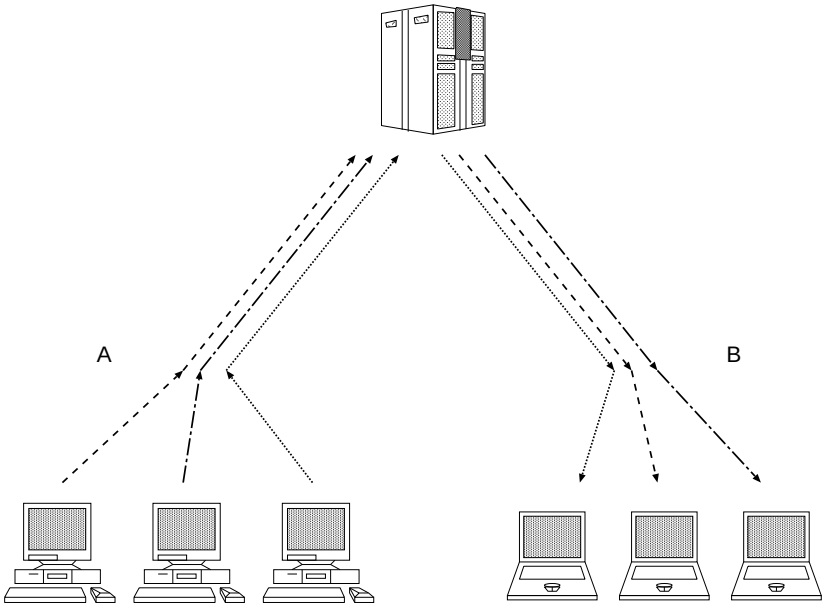

図 2.2: 回線交換網

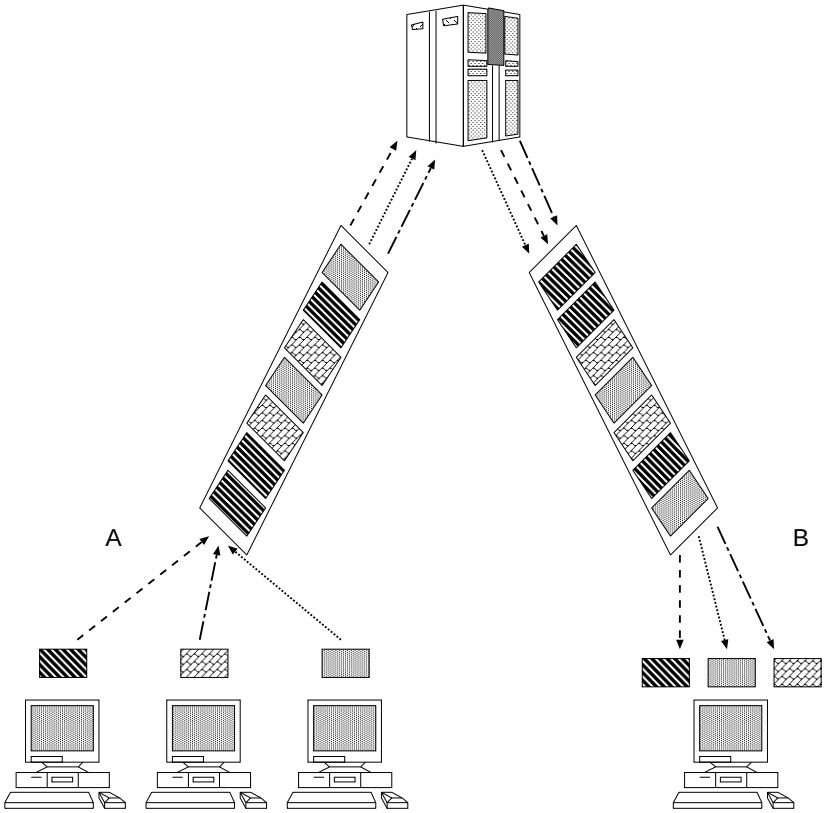

図 2.3: パケット網

つまり，インターネットに接続する全てのホストは，ホストの持つ最大値のデータ送出を行うことができる．言い換えれば，インターネットに接続する全てのホストは，ネットワーク資源を最大限に占有する可能性を持っていると言える．

2.2.2 ソースアドレス詐称

インターネットでは，全てのパケットは基本的に宛先 IP アドレスを使ってデータを転送する．そのため，データ転送時に送信元 IP アドレスは参照されない．

現在行われているほとんどの DoS 攻撃は source spoofing という手法を用い，攻撃元のソース IP アドレスを偽って大量のパケットを送信する．しかも，単一の IP アドレスを偽る場合は少なく，実際は少数のホストから送信されたと思われる DoS 攻撃パケットが，数百個のソース IP アドレスから送信されたように詐称される場合が多い．

そのため，DoS 攻撃を行っているパケットをインターネットの経路上において抽出できたとしても，その攻撃元を特定したり，症状を緩和することは非常に難しい．

2.3 パケット交換網を補完する転送技術

インターネットは転送するデータをパケットという単位に分割し，複数の通信が同時に行なえるパケット交換網である．そのため，多くのパケットが同一回線に集中し，輻輳が発生するとパケットの喪失や遅延が発生してしまう．輻輳による通信の品質低下を防ぐために様々な研究が行なわれている．

- End-End による輻輳回避
- 途中ルータによる輻輳回避
- 特定トラフィックの優先処理

本研究では，これらの技術を利用して flooding 攻撃の被害を抑制する方法について議論する．

2.3.1 End-End による輻輳回避

インターネットは，End-End での輻輳制御によりパケット交換網での帯域を共有している．インターネットトラフィックの大部分を占める TCP（Transmission Control Protocol）(6) トラフィックは，ネットワーク上の途中経路での輻輳を検知し，送信するパケットの量を調整する．

しかし，flooding 攻撃などはパケットを大量に送信して帯域を消費する事を目的としているため，End-End での輻輳制御は行なわれない．一方，TCP などの輻輳制御を行なうトラフィックは，flooding 攻撃による輻輳を検知して利用帯域を抑制してまう．そのため，End-End による輻輳制御機構は flooding 攻撃に影響されやすい．

本研究では，flooding 攻撃により他の TCP トラフィックが受ける影響を軽減する機構を提案する．

2.3.2 途中ルータによる輻輳回避

インターネットでは，ルータがパケットを転送する事により通信が行なわれる．ルータにはパケットを一時的に保持するパケット queueing バッファがある．一般的なルータのパケット queueing バッファは 1 つである．

途中ルータでの輻輳回避方法には，RED(Random Early Detection)(7) や FQ(Fair Queue)(8) などのがある．

RED は，ルータ内のパケット queueing バッファを監視し，パケット queuing バッファの平均値が大きくなるとランダムに選択したパケットを破棄する．パケットはランダムに破棄されるため，帯域を多く占有している通信でのパケットが破棄される可能性が高い．RED では，TCP などによる End-End での輻輳回避機構が前提になっている．TCP では，パケットが破棄されると送信するパケットの量を調整するため，RED によりパケットを破棄された TCP セッションはパケット送信量を抑制する．しかし，flooding 攻撃を行なっているトラフィックは End-End での輻輳制御を行なわないため，RED では効果的な輻輳制御を行なえない．

FQ では，トラフィック毎にパケット queueing バッファを用意し，それぞれのパケット queueing バッファから順番にパケットを転送する．それにより，各トラフィックは同等の帯域を利用できる．しかし，flooding 攻撃では，送信元の IP アドレスが偽造されている事が多く，flooding 攻撃によるトラフィックを特定して 1 つのパケット queueing バッファにまとめる事が困難である．

2.3.3 特定トラフィックの優先処理

IP では信頼性が保証されない通信しか提供されない．しかし，CBQ(Class Based Queue)(9) などのキューイング手法を利用する事により特定のトラフィックに対して優先的に帯域を割り当てる事ができる．

CBQ は，CBQ はリンク共有サービスや，リアルタイムアプリケーションのトラフィックに対するサービスを提供する queue である．クラス構造を基本として，階層的なクラス構造を作ることができ，クラスは独立した queue を持つ．ルートのクラスはリンクの利用可能な帯域全てを持ち，下位のクラスに帯域を割り当てる．下位のクラスは上位のクラスから帯域を借りることが可能で，上位のクラスの使用中の帯域に余裕がある場合は，下位のクラスは割り当てられた以上の帯域を利用することができる．しかし輻輳が起きた時，他のクラスが割り当てられた以上に帯域を使用しているために，割り当てられた以下の帯域しか使えないクラスがある場合は，割り当てられた以上に帯域を使用しているクラスは割り当てられた帯域まで調整される．

帯域割当の低いクラスの非優先 queue を作成し，flooding 攻撃などのトラフィックを

その queue に割り当てることにより，flooding 攻撃などの被害を軽減することができる．しかし，flooding 攻撃は送信元 IP アドレスを偽造している場合が多く，あらかじめトラフィックパターンを記述しておくのが困難である．本研究で提案する手法を利用する事により，flooding 攻撃のトラフィックパターンが発見できる．それにより，同一のトラフィックパターンを持つトラフィックを非優先 queue に割り当てる事ができる．flooding 攻撃のトラフィックを非優先 queue に割り当てる事により，flooding 攻撃による他のトラフィックへの影響は軽減される．

2.4 ソースアドレス詐称への対策

インターネットは中継器の処理をできるだけ削減しすることによって，規模性を獲得するという基本思想に基づいて設計されている．そのため，中継器におけるパケットの転送の際，転送に必要な宛先 IP アドレスは参照されるが，転送に必要のない送信元 IP アドレスは参照されない．つまり，中継器は送信元 IP アドレスを詐称しているパケットでも転送してしまう．

ソースアドレス詐称への対策として，現在では以下に示す 2 つの手法が存在する．

- ingresss filter
- IP traceback

前者は，ソースアドレス詐称された可能性の高いパケット組織外に転送しない技術であり，後者はソースアドレス詐称されたパケットを追跡してゆく技術である．

しかし，現状では両者とも完全な対応策としては機能していない．本節では，上記 2 つの技術について触れ，flooding 攻撃の被害を抑制するという視点から議論を行う．

2.4.1 ingress filter

インターネットは，その名が示すように様々なネットワークが相互接続したものである．それぞれのネットワークは割り当てられた IP アドレスブロックを自組織の中で用いる．そのため，自組織のネットワークから外部ネットワークへパケットを送信する場合，基本的に全てのパケットのソース IP アドレスは，自組織に対して割り当てられた IP アドレスブロック内の IP アドレスである．

つまり，外部ネットワークへの接続点において，自組織に割り当てられた IP アドレスブロック以外の IP アドレスがソース IP アドレスになっているパケットを filter することによって，不正なパケットが外部ネットワークへ転送されるのを防止することができる．

このネットワーク内部から外部へ転送されるパケットを防ぐ filter を ingress filter と呼ぶ．図 2.4に ingress filter の概念図を示す．

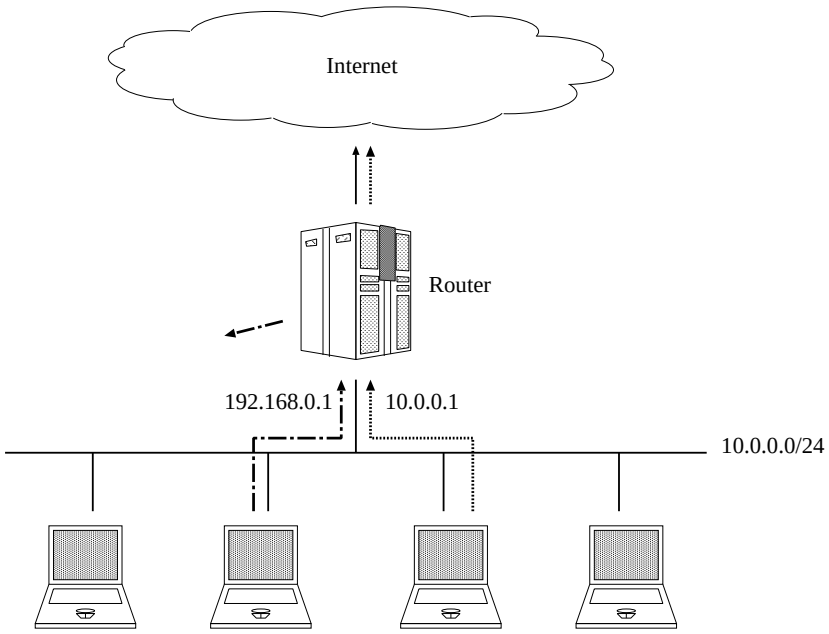

図 2.4: ingress filter 概念図

図2.4を例にとると，10.0.0.0/24 の IP アドレスブロックを保持するネットワークにおいて 192.168.0.1 のように，10.0.0.0/24 の IP アドレス空間に存在しない IP アドレスをソース IP アドレスとするパケットを Router において落す filter を ingress filter と呼ぶ．

この ingress filter が全てのネットワークの出口において記述されているならば，インターネット上をソース IP アドレス詐称したパケットが通過することはない．

しかし，この ingress filter は運用ポリシ上の問題，技術的な問題のため，全てのネットワークでは機能していない．これは，大手 ISP などのバックボーンでは，トポロジが複雑になる場合があり特定の IP アドレスブロックによってはどちらがネットワーク内部か判別しがたい場合があること．また，ingress filter を記述することに対して，なんら強制力がなく，ルータを設置した初期状態では記述されていないため，その初期状態のままになっている場合が多いこと．更に，IPv4 における MIP(Mobile Internet Protocol)(10) など，ingress filter が記述してあるネットワークでは動作しないソフトウェアがあることなどが原因である．

つまり，現状では source spoofing を防ぐために ingress filter を利用することは困難であると言える．

2.4.2 IP traceback

source spoofing によってソース IP アドレスを詐称されたパケットを追跡する技術が IP traceback である．IP traceback では，ルータ機能を追加することにより，パケットの通過した順路を作成し，これによって攻撃元の特定が可能となる．

図 2.5，図 2.6に IP traceback の概要図を示す．図 2.5は DoS 攻撃のパケットの経路

を，図 2.6は攻撃元を追跡する概要を示している．

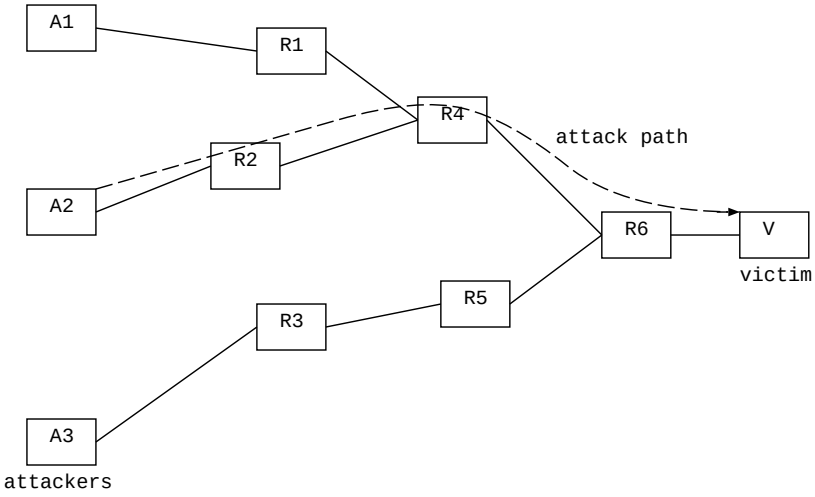

図 2.5: IP traceback 概要図 (DoS 攻撃)

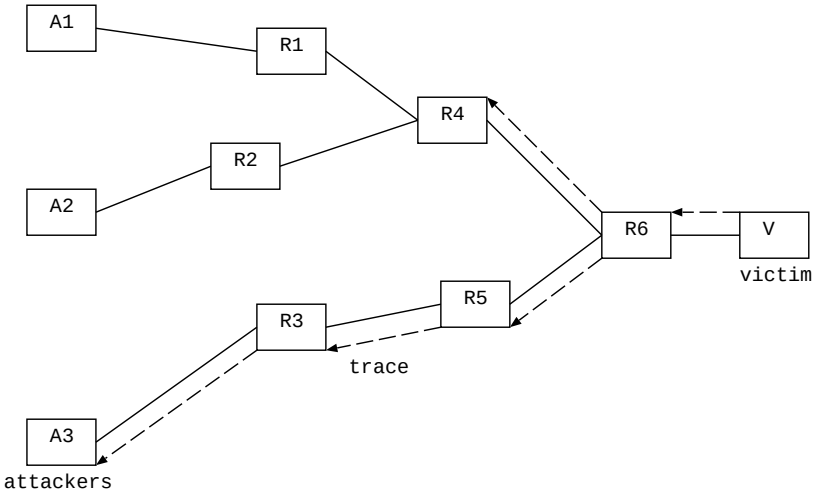

図 2.6: IP traceback 概要図 (攻撃元追跡)

図 2.6に示したように，ルータが協調することによって，攻撃元を追跡する機構が IP traceback である．

現在，IP traceback を行う際に，ルータに追加する機能は大きくわけて以下の 3 つの手法に分類できる．

- IP 系 traceback

  中継したルータの IP アドレスをパケットにマークする手法が IP 系 traceback 技術である．IPv4 には，IP Record Route オプションが存在し，ルータの IP アド

レスを順次オプションヘッダに追加することができる．しかし，オプションヘッダには 20bytes という上限があり，IPv4 アドレスを最大 9 こまでしか記述することができない．

この問題を改良する手法として，確率的パケットマーク方式 (12) が提案されている．この手法では，各ルータが一定の確率に従って，自律的にパケットオプションを付加するため，この確率を低下させることによって，20bytes という上限ないでも記述することができる．また，確率を低下させることによって，flooding 攻撃パケット自体へマークする確率も同時に低下させてしまうが，flooding 攻撃はデータ量の多い攻撃なので，サンプリングの標本数を小さく設定した場合でも，問題は発生しない．

- ICMP 系 traceback

  ICMP を用いた traceback 方式には，iTrace(13) がある．iTrace では，ルータがパケットを中継した際に，低い確率で送信先ホストに ICMP メッセージを送信する．flooding 攻撃のように，攻撃パケットを大量に中継する場合には，全ての中継したルータから送信先ホストに対して ICMP メッセージが到着する．この到着した ICMP メッセージをホストにおいて再構成することによって，攻撃元を追跡することができる．

  しかし，iTrace の ICMP メッセージの中継は，ルータの入力デバッグの性能に依存するため，一部のルータで負荷が高い場合に正常処理されない場合がある．また，iTrace では送信する ICMP メッセージ分のトラフィックを増加させてしまうという欠点がある．

- Hash 系 traceback

  Hash 系 traceback と呼ばれる手法は，ルータにログ記録機能を追加する手法である．この手法では，攻撃 packet のログを隣接する上位ルータから再帰的に探索することによって，攻撃元を逆探知することができる．しかし，ルータで全てのパケットをログに記録するとディスク空間が不足するため，Hash テーブルを利用することによって解決している．

  Hash を利用したログ手法としては，Source Path Isolation Engine(11) が提案されている．この手法では，パケットに一意な Hash ID を付加することで，ディスク空間を大幅に削減している．Hash 関数の入力として，IP 層の発信元や送信先だけでなく，TTL, オフセットフラグ，ID フィールドも用いる．従って，IP 層より上位層のデータが等しいパケットでも，Hash ID をつけるルータによって Hash ID が異なる．この結果，DoS 攻撃のトラフィック量が非常に小さい場合でも，細かい粒度で逆探知することができる．

  また，現在では，高速な記録を実現するために横河電気 (14) により Hash traceback 用のハードウェアの開発が進んおり，近い将来，実現の度合いが高い．

これらの技術は，現在 IETF(15) の IP Path Tracing ワーキンググループにおいて標準化に向けた話し合いが進められている．この状況を考えた場合，その攻撃元を追跡することが可能になるであろう．その際，IP traceback の出発点となる，flooding 攻撃の検知は非常に重要である．

2.5 トラフィック収集機構の必然性

ネットワーク運用者が，flooding 攻撃に対応するためには，2.1節で述べたように，1) 攻撃の検知，2) 被害対象の特定，3) 攻撃特性の分析，4) 機器の設定変更 の 4 つ手順が必要である．これらの手順のうち，最後の手順である機器の設定変更は 2.3節で述べたように，細かい粒度でのトラフィックエンジニアリングを実現する可能性を秘めている．したがって，その前提となる攻撃の検知，特性分析は現状の技術が必要である．

攻撃特性の分析の一部である攻撃元ホストの追跡は，2.4.2節で述べたように IP traceback 技術の開発が進んでいるが，攻撃元ホストの追跡に目的を絞っており，DoS 攻撃の情報を汎用的に収集する機構ではない点，また，攻撃の検知自体を行う機構を持っていない点の 2 つの観点から，ネットワーク資源を占有する flooding 攻撃の情報を収集するために最適であるとはいえない．

つまり，ネットワーク運用者が flooding 攻撃に対応するためには，それに特化したトラフィック収集機構が必要である．このトラフィック収集機構は，機器の設定変更のために必要な情報をより詳細に収集する必要がある．

本節では，flooding 攻撃に適したトラフィック収集機構を作成する際の技術的問題について触れる．その上で，その問題の関連技術による解決手法を紹介し，本研究で作成する機構に必要な要件を議論する．

2.5.1 トラフィック収集の技術的問題点

flooding 攻撃を検知するためには，前提として，ネットワークからトラフィック情報を抽出する必要がある．しかし，全てのトラフィック情報を抽出することは不可能に近い．また，もし抽出できたとしても，その情報を保存したり解析したりすることは一段と難しい．

例えば 100Mbps の回線のトラフィックを収集する場合，単純に計算しても 1 秒間で 12Mbyte,1 分間で 715Mbyte,1 時間で 42Gbyte,1 日で 1Tbyte のデータ量になる．このため，単純に全てのパケットを収集する手法では，長期に渡ってトラフィック情報を収集するはできない．

2.5.2 既存の手法

ネットワーク上におけるトラフィック収集を行う場合，単純に全てのパケットを収集することはディスク容量の関係上困難である．また，ディスクに保存できたとしても，

大容量のデータは実際に利用するための加工コストが高くなるため現実的ではない．

そのため，ネットワーク上におけるトラフィック収集では，一般的に以下に示す 3 つの手法を用いることが多い．

- フィルタリング

  IP ヘッダの特定の領域のみのデータを収集したり，特定のホスト宛のデータのみを収集するなどといった，予め設定したルールセットに適合する情報のみを収集する手法をフィルタリングによるトラフィック収集と呼ぶ．

  この手法では，既定のトラフィックのみを収集することができるため，収集するデータ量を削減することができる．しかし，ルールセットを設定する際に考慮されていなかったデータはまったく収集することができない．つまり，この手法を用いた場合，収集するデータ量を削減するためには，より厳密なルールセットが必要となるが，ルールセット厳密にすればするほど収集したいデータを取りこぼしてしまう可能性が高くなる．このため，適切なルールセットの記述が非常に難しい．

- サンプリング

  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，全てのパケットを母数とみなし，その中から標本抽出する手法をサンプリングによるトラフィック収集と呼ぶ．

  この手法では，抽出したデータのみを収集するため，母数である全てのデータと比較した場合に収集するデータ量を削減することができる．しかし，抽出する標本数が少ない場合は母集団である全てのパケットの傾向を表わすことができない．一方，サンプリングの信頼度を上げるために標本数を増加させた場合は，収集するデータ量が大きくなりすぎてしまう可能性がある．また，ネットワーク上におけるトラフィック収集の場合，データを収集する時間や場所によって，母数である総トラフィック量が大きく増減するため，適切な標本数の設定が難しい．

- 集約

  データを収集する際に IP アドレスを AS(Autonomous System)(16) 番号単位でデータを収集したり，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含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．

  この手法では，収集したデータの持つそれぞれの要素の粒度を落すことによって，細かな差異を隠蔽化することによって，収集したデータの保存量を削減することができる．しかし，データの保存量を削減しすぎると，データの持つ意味が大きく失われることとなり，トラフィック収集の本来の目的を果たせない可能性が発生する．

  そのため，この手法ではトラフィック収集の目的に応じて，集約する粒度の適切な設定を考慮する必要がある．

現在よく使用されているトラフィック収集ソフトウェアは使用目的に応じて上記の 3 つの手法のいづれか，もしくは，組み合わせを使用しているものが多い．以下に代表的なトラフィック収集ソフトウェアについて述べ，本研究の目的である flooding 攻撃情報の収集という観点から議論を行う．

tcpdump

tcpdump は libpcap(17) を使用したパケット収集ツールである．図 2.7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
192976724:192976724(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
2786713294:2786713294(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 17520 (DF) [tos 0x10]
```

図 2.7: tcpdump 出力例

tcpdump は，起動したホストが受け取れる全てのパケットを収集できる．そのため, 送信先アドレス, 宛先アドレス，プロトコル，データ長などさまざまな情報を収集できる．

しかし，収集できる情報が多いにも関わらず，収集した情報を集約する機能が付加されていない．そのため情報量が膨大になってしまう．

この問題点を補完するために tcpdump には，以下に示す項目を設定することのできる機能が付加されている．

- パケット内の収集する byte 数
- 特定の IP アドレスまたは mac アドレス宛のパケット
- 特定の IP アドレスまたは mac アドレス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

上記の機能は全て，フィルタリング手法に分類される．

つまり，特に事前の設定をせずに tcpdump を使用した場合は，全てのパケットを収集することになってしまう．しかし，事前に適切なフィルタを記述することによって収集するデータ量を削減することができる．

SNMP

SNMP(Simple Network Management Protocol)(18) 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)(19) の形式に基づいて収集したデータを収集，管理できる．図 2.5.2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．

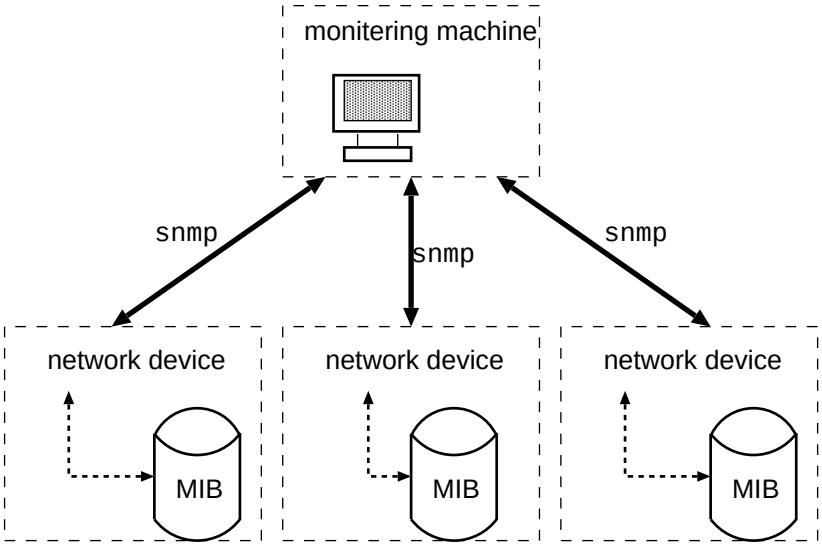

図 2.8: SNMP 概要図

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.

この SNMP を使用した可視化ツールである，MRTG，RRDtool(20)，cricket(21) などは収集した情報を可視化を実現している．これらの SNMP を用いたツールは，定期的に SNMP クライアントを用いてネットワーク機器から情報を収集している．ここで収集される情報には timestamp が付加されていないため，SNMP を用いてデータを収集する時点で既に時間的な集約を行っていることになる．この時間的集約によって SNMP を用いたツールはデータ収集量を抑制し，長期に渡るトラフィック収集を実現している.

しかし，SNMP を用いて収集することのできる MIB 形式のデータ保持機構は，各 IP アドレス毎の流量や各プロトコル毎の流量を収集する機構をもともと備えていない．これらの情報は拡張 MIB に記述することで可能であるが，全ての項目を予め設定しておくことは項目数が膨大であるため不可能である．この項目数の概算を表 2.1に示す.

表 2.1に示した項目を計算すると以下のようになる.

表 2.1: MIB アクセス項目数試算

データ項目	項目数	説明
データの通過した時間	なし	データを取得した時間を保持するため必要なし
パケット数	1	
総データ長	1	
送信元 IP アドレス	2^{32}	IPv4 に限った場合
宛先 IP アドレス	2^{32}	IPv4 に限った場合
送信ポート	2^{16}	
宛先ポート	2^{16}	
IP プロトコル	4	最低でも tcp，udp，icmp，egp の 4 つは必要
IP version	1	IPv4 のみの試算のため

$$
\begin{aligned}
&Total \\
&= (Packet + Length) \times (protocol + src\_addr + dst\_addr + src\_\text{ポート} + dst\_port) \\
&= (1 + 1) \times (4 + 2^{32} + 2^{32} + 2^{16} + 2^{16}) \\
&= 2 \times (4 + 4294967296 + 4294967296 + 65536 + 65536) = 17180131336
\end{aligned}
$$

- Total： 総 MIB アクセス項目数 (IPv4 のみの場合)
- Packet： パケット数
- Length： データ長
- protocol： プロトコル
- src_addr： 送信元 IP アドレス
- dst_addr： 宛先 IP アドレス
- src_port： 送信元ポート
- dst_port： 宛先ポート

上記の試算式に示したように SNMP を使ってトラフィックの情報を収集した場合，膨大なデータを収集しなければならない．これは，” 値が 0” のデータも全て収集しなければならないためである．

しかし SNMP を用いて収集することのできる MIB 形式のデータ保持機構は，各 IP アドレス毎の流量や各プロトコル毎の流量を収集する機構をもともと備えていない．そのため，MIB 形式で保存されたデータを収集する SNMP は，パケットの詳細な情報を必要とする flooding 攻撃の情報収集には適応しない．

netflow

netflow は Cisco システムズ社 (22) 製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
- 送信元アドレス，宛先アドレス
- 送信ポート，受信ポート

netflow は，データ収集量の削減を実現するためサンプリング手法を採用している．つまり，N 個（設定可能）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情報のみを UDP を用いて送信する．

この netflow の送信したデータを利用したソフトウェアとして代表的なものに CAIDA(23) の開発した cflowd(24) がある．cflowd の概要図を，以下の図 2.9に示す．

図 2.9に示されているようように，cflowd は netflow 機能を備えたルータからトラフィック情報を収集する．この際，トラフィック情報の送信には UDP が用いられる．

cflowd は各ルータから送信された情報をフローの単位に集約することによって，更に保持するデータ量を削減している．また，これらのデータを多地点から収集，集約する機構も備えている．

尚，この cflowd においては，フローは送信元アドレス，宛先アドレス，送信ポート，受信ポートが同一のデータの集合と定義されている．

しかし，netflow 機能は Cisco システムズ社の特定の機器，特定の IOS(25) のみで動作が保証されており，さまざまなベンダの通信機器によって構成されるネットワークには対応できない．

また，netflow 機能を使用した場合，パケットの収集よりもルータのパケット転送を優先するため，特に flooding 攻撃を受けている場合には，標本数が減少してしまうというため，flooding 攻撃の情報を詳細に収集するのは難しい．

2.5.3 flooding 攻撃対応に適したトラフィック収集

2.5.2節で述べてきたように，ネットワーク上においてトラフィックを収集する場合，基本的に全てのデータを収集することはできないため，1) フィルタリング,2) サンプリング,3) 集約 の 3 つの手法を用いることによって，収集するデータ量を削減する．

しかし，これらの手法には一長一短があるため，目的に応じた手法を選択，もしくは，組み合わせる必要がある．更に，本研究の目的の一つである flooding 攻撃の情報を詳細に収集することは，既存のソフトウェアで実現することは難しい．

そこで，本小節においては，ネットワーク運用における flooding 攻撃への対応の現状を踏まえ，flooding 攻撃の情報を収集するために必要な要件を議論する．

ネットワーク運用を行う上で重要なことは，より多くのユーザに対してより快適なネットワーク環境を提供することである．flooding 攻撃によって，ネットワーク資源が

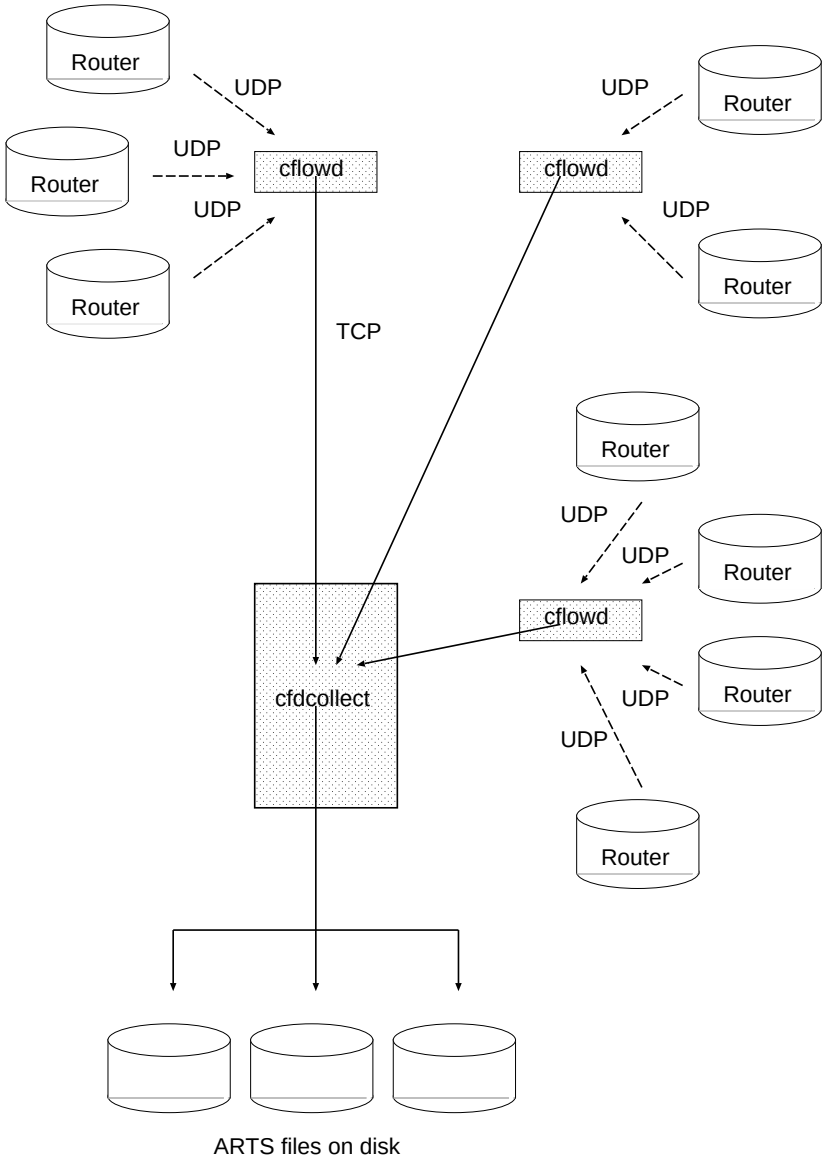

図 2.9: cflowd 概要図

占有されてしまった場合，当該ネットワークを経由するほぼ全ての通信が被害を受けてしまう．これの被害を最小限に抑えるために，以下に示す 3 つの方針が考えられる．

方針 1. 攻撃の被害を抑える

flooding 攻撃の対象となっている宛先 IP アドレスパケットを全て転送しないもしくは転送量を制限する．

これによって，flooding 攻撃の対象となっている IP アドレスを持つホストを除く全てのホストに対し，flooding 攻撃の影響を抑えることができる．

方針 2. 攻撃パケットのみを排除する

flooding 攻撃の対象となっている宛先 IP アドレスに加え，攻撃に使用されているプロトコルや送信元 IP アドレスを含むパケットのみを転送制限する．

これによって，攻撃の対象となっているホストに対しても，flooding 攻撃の影響を抑えることができる．

方針 3. 攻撃を予測して被害を防止する

“昨日のインターネットは今日のインターネットではない” と言われるように，インターネットの特性が変化し続けるならば,「方針 3」を実現することは不可能である．しかし，flooding 攻撃の特性が常に変化しているかどうか現在では不明である．

flooding 攻撃がなんらかの特性をもっていると仮定するならば，過去の flooding 攻撃の情報を蓄積し，攻撃手法のパターンを見つけ出すことによって，次の攻撃を予測して，flooding 攻撃による被害をあらかじめ防止する可能性を秘めている．

これら 3 つの方針を比較した場合，” 方針 3” が最も被害を最小限に抑えることのでき，その次が” 方針 2” である．しかし，” 方針 2” を実現するためには” 方針 1” にくらべ，より粒度の細かいトラフィック収集が必要である．また，” 方針 3” を実現するためには，” 方針 2” に加え，長期にわたるトラフィック収集が必要である．

本研究では，上記 3 つの方針すべてに対応することのできるトラフィック収集機構を作成する．このためには，まず flooding 攻撃パケットの詳細な情報を収集する必要がある．しかも，ネットワークに接続されたどのホストでも flooding 攻撃の加害者にも被害者にもなり得る現状，どのようなパケットでも攻撃パケットになり得る現状を踏まえた場合，あらかじめ特定の情報だけを収集するだけでは不十分である．そのため，あらかじめルールセットを記述し，特定のパケットだけを抽出する手法を取ることはできない．更に，過去の攻撃特徴から，次の攻撃を予測するためには長期にわたるトラフィック収集が必要である．

第3章　AGURIの設計と実装

本章では，flooding攻撃の情報を収集することのできるトラフィック収集機構aggregtion based traffic profiler(以下，AGURI)(27) の設計と実装について述べる．

3.1　設計

flooding攻撃の情報を詳細に収集するためには，新たに独自の機構を作成する必要がある．本研究では，この機構をAGURIと呼ぶ．本節ではAGURIの設計について述べる．

2.5節で述べた設計要件を以下にまとめる．

要件1. あらかじめルールセットを記述する必要がない

ネットワーク資源を占有するflooding攻撃は，ネットワークに接続する全てのホストが加害者または被害者になれる．また，この攻撃は大量にデータを送信することによって引き起こすことができるため，特定のポートや特定のプロトコルに攻撃手法を限定されることがない．

このため，あらかじめ設定したルールセットに適合したパケットのみを収集するという手法では，全てのflooding攻撃のトラフィックを収集することはできない．

要件2. トラフィックの詳細な情報を収集することができる

flooding攻撃への対応である中継器上におけるフィルタ記述を考慮した場合，IP層に記述してある情報をトラフィック収集によって明らかにする必要がある．

例えば，攻撃先ホストのIPアドレスが判明した場合には，中継器上において当該IPアドレスを宛先とするパケット転送を制限することができる．更に，攻撃に使用されているプロトコルやポート番号が判明した場合には，パケットの転送制限をより詳細に記述することができる．

要件3. 短期から長期に渡って利用できる

現在被害を受けているflooding攻撃に対処するためには，時間的に細かい粒度のトラフィックが必要になる．

それに加え，flooding攻撃の予測による被害の防止を考えた場合，攻撃手法のパターンや攻撃周期などの情報が有用である．そのため，長期的なトラフィック収集が必要である．つまり，収集するデータ量をできるだけ削減する必要がある．

本節では，これら 3 つの要件をみたす AGURI の設計を行う上で，トラフィックデータの収集部と収集したデータの集約部，集約したデータの蓄積部に関する議論を行う．

3.1.1 データ収集手法

flooding 攻撃の情報を収集する場合，ネットワーク運用者の対策を考慮する必要がある．ネットワーク運用上の対策として，一般的なものに中継器上における転送量制限のフィルタ記述が存在する．また，研究中の技術としては 2.3節で述べた技術も存在する．

これらの手法を利用する場合，基本的に IP ヘッダに記述されている情報が必要となる．現在，インターネット上では IPv4 と IPv6 の両者が用いられている．図 3.1に IPv4 のヘッダを，図 3.2に IPv6 のヘッダを示す．

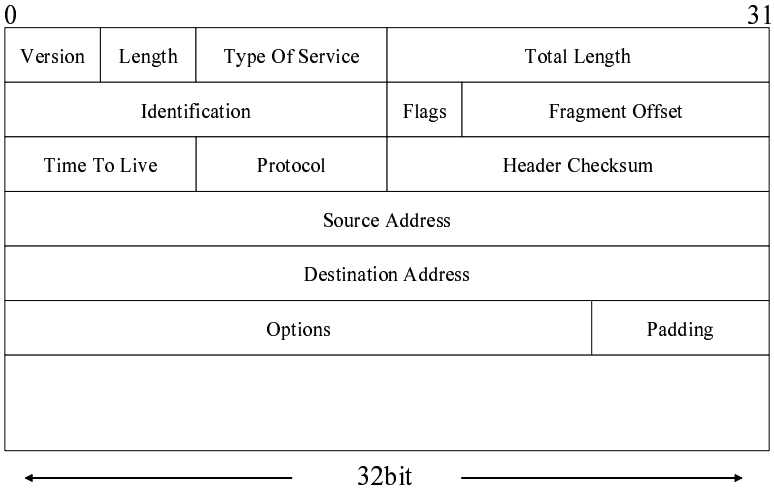

図 3.1: IPv4 ヘッダ

これら IPv4 ヘッダ，IPv6 ヘッダの中で収集すべき要素は，flooding 攻撃への対応の際に必要となる情報である．IP ヘッダ内で最低限収集すべき要素を以下に示す．

- Version
- Source Address
- Destination Address
- Length
- Protocol(IPv6 では Extension Header に含まれる)

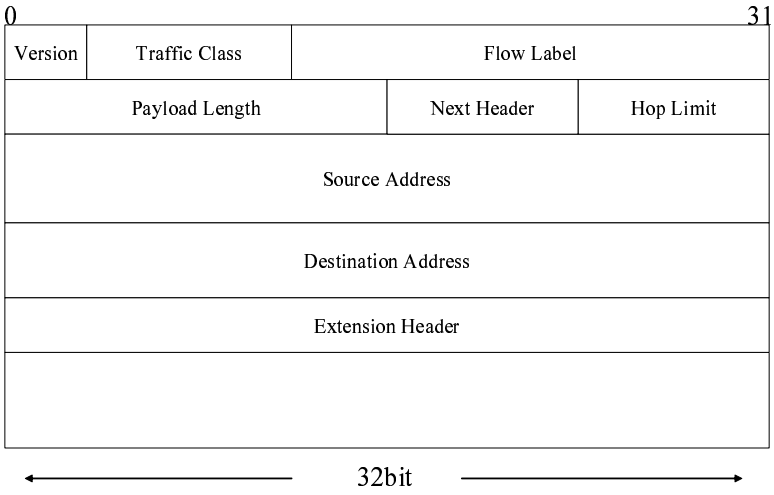

図 3.2: IPv6 ヘッダ

また，IP 層の情報では内が，中継器におけるフィルタリングにおいては，一般的に送信元ポート番号，宛先ポート番号によるフィルタの記述を行うことが可能であるため，第 4 層のポート番号情報も収集した方が良い．

これらの情報を収集する既存のトラフィックを収集手法として，1)SNMP を利用,2)netflow を利用,3)libpcap を利用の 3 つが上げられる．2.5.2節で述べたように，SNMP や netflow を用いることは適切ではない．そこで，AGURI では libpcap を利用する手法を採用した．

しかし， 2.5.2節で述べたように，libpcap を利用した場合，基本的に全てのパケット情報を収集してしまうため，データ収集量を削減するためには，適切なフィルタ記述が必要である．

libpcap はパケットを取り込むためのライブラリである．libpcap を使用することで OS が提供する下位 (データリンク層) のパケット取り込み機構を隠蔽できる．図 3.3に libpcap の概要を示す．

libpcap の機構の中で特に注目するべき点は，パケットを可変長でキャプチャーできることである．

この機能を用いることによって，Ethernet で IPv4 のデータを収集する場合，データリンクヘッダ (Ethernet ヘッダは 14 バイト)，IP ヘッダ (IPv4 の場合 20 バイト)，トランスポートヘッダ (TCP ヘッダは 20 バイト) の合計 54 バイト以上の長さでデータを収集すれば，設計要件をみたすパケット情報を収集することができる．

また，libpcap は root 権限のある単一ホストで利用できる．つまり，通信機器から直接収集することはできない．しかし，データを収集したい通信機器のポートミラーリングを行うことによって，データ収集の対象となる機器を通過するデータと全く同じ

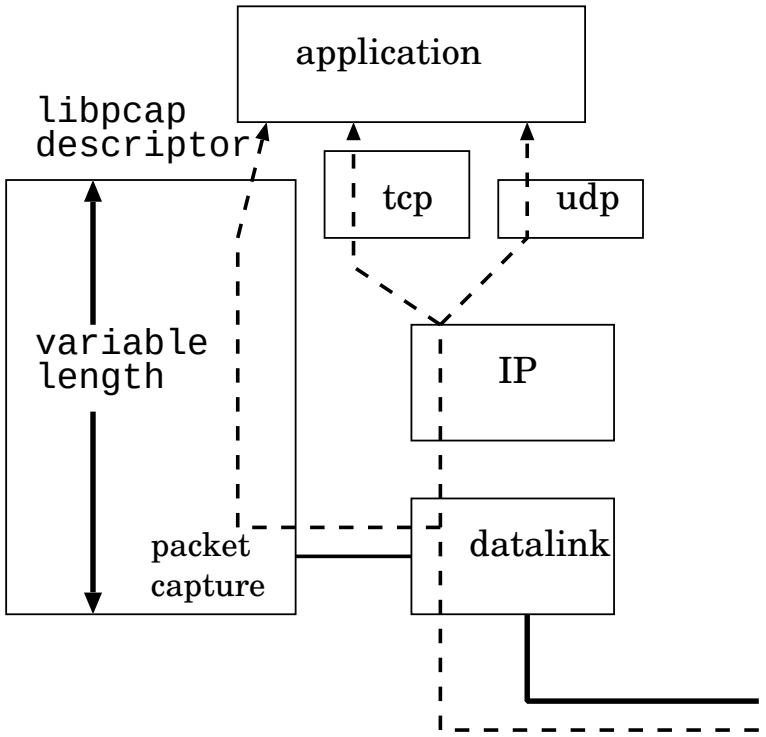

図 3.3: libpcap の機構概要

ものを収集できる.

このため，実際にネットワーク上でデータを収集する際にも，特定のベンダに縛られることがない.

3.1.2 データ集約手法

libpcap を用いて収集したデータは，以下の 3 つに分類することができる.

1. 階層的に複数の意味を持つもの

   Source Address や Destination Address，または，Source Port や Destination Port などの情報は階層的に複数の意味を持つ.

2. 他のデータと関連付けられて意味を持つもの

   IPv4 の Total Length や IPv6 の Payload Length など，一つのパケットにおける意味はあまりないが，他の情報と関連付けることによって，情報を付加することができる.

   例えば，Source Address の情報と関連付けることによって，特定の Source Address からのトラフィック量を算出することができる.

3. flooding 攻撃への対応にはあまり効果的でないもの

   libpcap を使用してデータを収集した場合，パケットの先頭からバイト単位で表現した情報を全て抽出する．

   そのため，IPv4 の場合，Source Address の情報を収集するためには，Idenfication や Fragment Offset といった，ネットワーク運用者の対応にとって効果的ではない情報も同時に収集してしまう．

libpcap の性質のため，flooding 攻撃の対応にあまり効果的でないが収集してしまった情報は破棄することによって，保持するデータ量を削減することができる．

しかし，階層的に複数の意味つものは，その意味を考慮した集約を行わなければならない．

2 分木構造によるトラフィックの格納

libpcap を用いて収集した情報には IP アドレスやプロトコル番号など階層的に複数の意味合いを持つものがある．

図 3.4に，”203.178.143.54” を送信元 IP アドレスとする 80 バイトのパケットを収集した例を示して説明する．

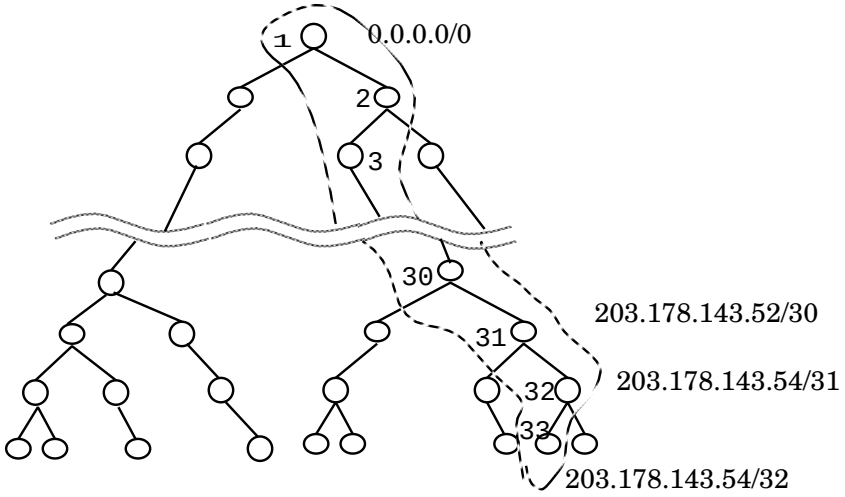

図 3.4: 収集した IP アドレスの持つ階層的な意味

この場合，203.178.143.54 を送信元とする 80 バイトのデータを収集しただけではなく，以下に示す IP アドレス空間を送信元とする 80 バイトのデータを収集したことにもなる．

1. 0.0.0.0/0
2. 128.0.0.0/1

: :

32. 203.178.143.54/31

33. 203.178.143.54/32

このように，収集したパケットは 2 分木構造で表現することができる．つまり，収集された IP アドレスはそのアドレスの所属する IP アドレス空間に含まれる要素である．

flooding 攻撃は単一のホスト宛のデータのみで行われる場合もあるが，同じ組織に割り当てられた IP アドレス空間に対して，攻撃を行うことによって同一経路上のネットワーク資源を占有することができる．このため，収集した IP アドレスはそのアドレスの持つ階層的な意味も考慮する必要がある．

パトリシアツリー構造と LRU による集約

このように，2 分木構造に収集したデータを格納することにより，flooding 攻撃に適した情報の意味を抽出することができる．このように木構造中の親ノードが必ず 2 つの子ノードを保持する場合，木構造中の全てのノード数 N は以下の式で表わすことができる．この式中におけるリーフノードとは，子ノードを持たない末端のノードを指す．

$$N(\text{全ノード数}) = n(\text{リーフノード数}) \times 2 - 1$$

つまり，IPv4 のアドレス構造を純粋な 2 分木で表現した場合，$n = 2^{32}$ となるので，$2^{33} - 1$ 個のノード数分のデータ領域を確保する必要がある．更に IPv6 の場合は $n = 2^{128}$ となるので，純粋な 2 分木構造に IP アドレス情報を格納するのは不可能である．

そこで，本研究ではパトリシアツリー構造を用いることによって，保持する情報量を削減する．パトリシアツリー構造とは，分岐点にのみ中間ノードを作成することによって，保持するノード数を小さく保つ構造である．

更に，LRU(Least Resent Used) アルゴリズムを新規ノードの追加に対して用いることにより，保持するノード数を一定以下に保つことができる．

この AGURI 独自の集約機構を図 3.5，図 3.6，図 3.7，図 3.8 を例にとり解説する．

この例におけるリーフノード数を 5 とする．

また，図中におけるリーフノードを”1,2,3,....”，中間ノードを”A,B,C,....” と記述する．

図 3.5のツリー構造に新たにリーフノード 5 を追加した状態を図 3.6に示す．ツリー構造の作成アルゴリズムにパトリシアツリーを用いているため，リーフノード 5 の作成と同時に，リーフノード 5 と中間ノード C の分岐に中間ノード D が作成される．

図 3.6のツリー構造に更にリーフノード 6 を追加する状態を図 3.7に示す．ツリー構造の作成アルゴリズムにパトリシアツリーを用いているため，リーフノード 6 の作成と同時に，リーフノード 6 とリーフノード 2 の分岐に中間ノード E が作成される．

更に，AGURI ではツリー中のノード数を一定に保つために LRU を用いているため最も早く追加されたリーフノード 1 をツリー構造から削除する．

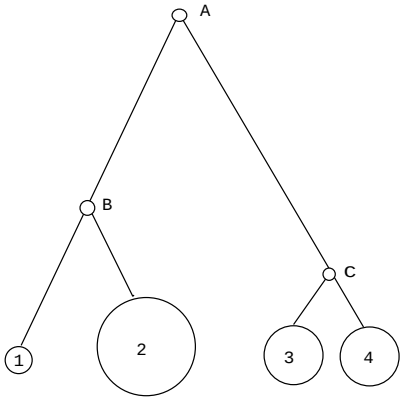

図 3.5: パトリシアツリー構造へのデータの追加 1

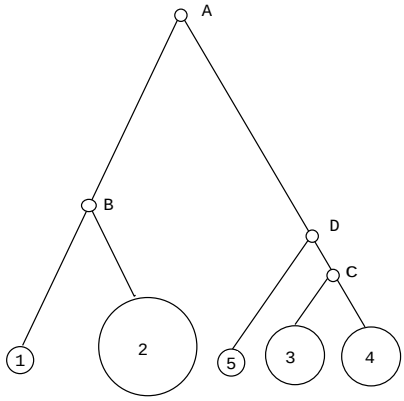

図 3.6: パトリシアツリー構造へのデータの追加 2

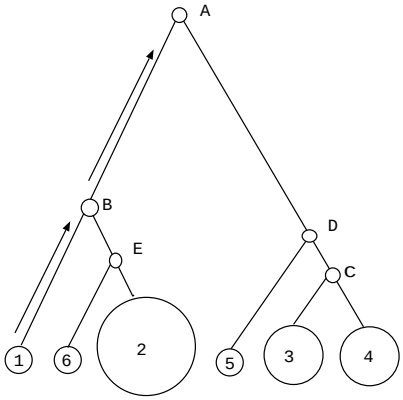

図 3.7: パトリシアツリーへの LRU を用いたデータの追加 1

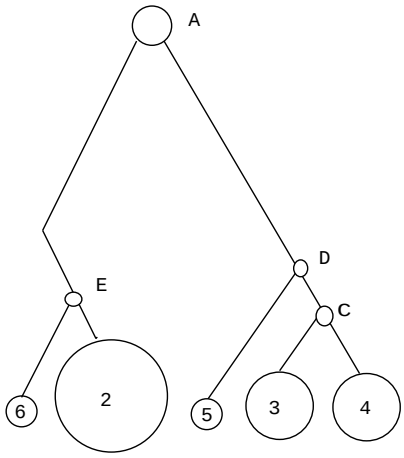

図 3.8: パトリシアツリーへの LRU を用いたデータの追加 2

ツリー構造から削除する際に，リーフノード 1 と分岐である中間ノード B の保持していた値をノード A に集約する．

この基本動作に加え，ある閾値以上のデータを保持しているノードを LRU の刈込み対象から除外することによって，トラフィック量の多いデータを保持し，かつ，ツリー構造の大きさを一定に保つことができる．

つまり AGURI では，IP アドレスやポート番号の階層的に複数の意味をもつデータとトラフィック量を関連付けることによって，収集したパケットの持つ要素ごとの流量を保持することを可能にした．

更に，データ保持構造にパトリシアツリーを用いることによって，2 分木構造にくらべ大幅にデータ保持量を削減した．また，LRU アルゴリズムを採用することにより，トラフィック量の少ない要素を集約することを可能にした．

3.1.3 データ蓄積手法

パトリシアツリー構造に格納したトラフィックは，時間情報を無視している．これは，収集したデータをツリー構造に追加する際に，データ収集時間を考慮せず同一のツリーにデータを追加するためである．例えば，AGURI で 1 時間トラフィックを収集した場合，収集を始めた直後のトラフィックと収集終了直前のトラフィックは，約 1 時間の時間的差異があるにも関わらず，同一のツリー構造に格納される．これにより，AGURI で長期に渡ってデータを収集した場合には，データ量を抑制するのと引き替えに，時間粒度が粗くなってしまう．

そこで，AGURI で収集した情報を一定間隔で切り出すことによって，収集したデータの時間粒度を細かくする必要がある．収集データの最適な切り出し間隔は，トラフィックを収集するポイント，収集する計算機資源によって異なるため，AGURI では，このデータ切り出し間隔をユーザが自由に設定できる必要がある．

さらに，もっとも時間的に細かい粒度のデータだけでは，長期にわたるトラフィック特性を見つけ出すことは難しい．そこで，AGURI では複数の細かい時間粒度のデータを集約することにより，長期にわたるデータを作成できる機能を付加する必要がある．

一定間隔で切り出されたデータを，その時間のトラフィック情報を示したサマリと定義し，図 3.9を用いて，サマリが更新される時間を”5 分” と仮定した場合の例を示す．

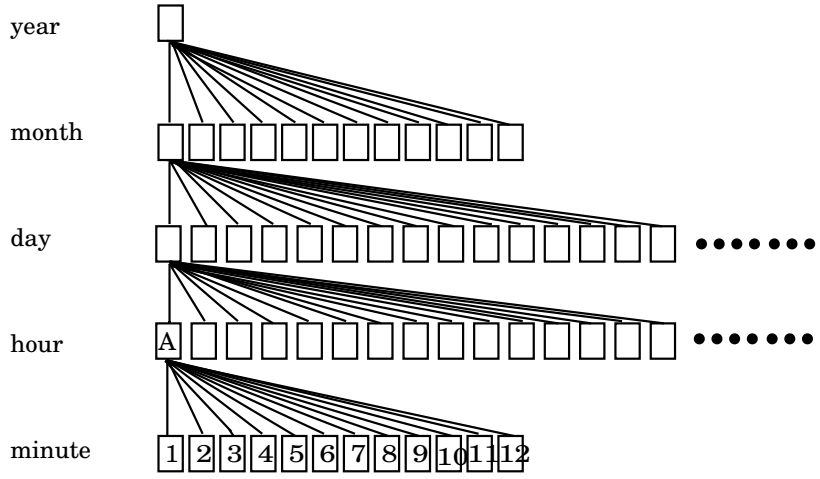

図 3.9: 異なる時間粒度のサマリ作成

図 3.9中のサマリ A は最近 1 時間のネットワーク状況を表すサマリである．このサマリ A は，サマリ 1 からサマリ 12 まで 12 個のサマリから作成される．

同様の手法によって，1 日分のサマリ,1ヵ月間のサマリ，1 年間のサマリも作成する．

このように時間粒度の異なるサマリを作成することで，短期的なネットワークの状況から長期的なネットワークの状況まで把握できる．

3.2 実装

AGURI は実装において，大きく分て以下の 3 つに分類することができる．

- データ収集部
- データ集約部
- データ蓄積部

本節では，上記の 3 項目について，実行コマンドとソースコードを交え AGURI の実装について述べる．

3.2.1 データ収集部

AGURI は libpcap を利用することにより，計算機のネットワークインターフェースからだけでなく，tcpdump を用いたデータからのトラフィック収集を実現している．

また，短期間のサマリを複数組み合わせることにより，長期サマリを作成する機能を実現するために，AGURI 自身の出力するサマリファイルからも収集することができる．

これら 3 つの収集対象の判別は AGURI を実行する際の引数で判別される．以下，これら 3 つ収集対象について個別に述べる．

ネットワークインターフェースからの収集

ネットワークインターフェースからデータを収集する際の実行コマンドを，以下の図 3.10に示す．

```
root#aguri -i interface_name
```

図 3.10: ネットワークインターフェースからのトラフィック収集

インターネットは Ethernet や ATM などさまざまな物理媒体を利用することが可能である．ネットワークを流れるパケットにはその回線特有のデータリンク情報が付加されている．そのため，ネットワークインターフェースから情報を収集する場合，libpcap を用いてデータリンク層を隠蔽化する．AGURI において，データリンク層を隠蔽化する関数のプロトタイプ宣言を図 3.11に示す．

AGURI では，図 3.11に示された関数を用いることにより，複数の通信媒体から同一にデータを収集することができる．

更に，収集したデータは以下の図 3.12に示す構造体に格納することにより，IP ヘッダやトランスポートヘッダから必要な情報を抽出することができる．

```
static void ether_if_read(u_char *user, const struct pcap_pkthdr *h,
              const u_char *p);      /* Ehternet */
static void fddi_if_read(u_char *user, const struct pcap_pkthdr *h,
              const u_char *p); /* Fiber Distributed Data Interface */
static void atm_if_read(u_char *user, const struct pcap_pkthdr *h,
              const u_char *p); /* Asynchronous Trasfer Mode  */
static void ppp_if_read(u_char *user, const struct pcap_pkthdr *h,
              const u_char *p); /* Point-to-Point Protocol*/
```

図 3.11: libpcap を利用したデータリンク層の隠蔽化関数

```
#include<netinet/ip.h>
struct ip;      /* IPv4 header 構造体 */
#include<netinet/ip6.h>
struct ip6_hdr; /* IPv6 header 構造体 */
#include<netinet/tcp.h>
struct tcphdr; /* TCP header 構造体 */
#include<netinet/udp.h>
struct udphdr; /* UDP header 構造体 */
```

図 3.12: 収集した第 3 層，第 4 層のデータを格納する構造体

図 3.12に示した構造体に抽出したパケット情報を格納し，この構造体のメンバであるパケットのソース IP アドレスやソースポート番号などの情報を任意に抽出することができる．

tcpdump データからの収集

tcpdump データからデータを収集する際の実行コマンドを，以下の図 3.13に示す．

```
root#aguri -r dump_file
```

図 3.13: tcpdump ファイルからのトラフィック収集

図 3.13に示した dump_file は，“tcpdump -w” コマンドによって収集することのできるトラフィック情報である．

つまり，AGURI では libpcap を用いることにより，ネットワークインターフェースから直接トラフィック情報を収集するだけでなく，事前に tcpdump によって収集した情報からもトラフィック情報を収集することができる．これにより，リアルタイムな情報収集でなはく，過去のトラフィック情報の収集も行うことができる．

しかし，tcpdump ファイルからデータを収集する場合，パケットを収集した時間を考慮する必要がある．なぜなら，tcpdump ファイルに含まれた時間情報を無視した場合，一定間隔のサマリ情報の切り出しができないからである．

この問題を解決するために，AGURI では tcpdump ファイルに含まれる時間情報を抽出し，定期的に一定間隔の時間が経過したのと同じ挙動を模倣する．この，時間情報の抽出と，経過時間の模倣を行う機能を，以下の図 3.14に示す．

```
/* パケットに付加された timestamp 情報は struct timeval *ts に格納されて
いる */
do_pkthdr(const struct timeval *ts, u_int caplen, u_int len)
{
        if (caplen > caplen_max)
                caplen_max = caplen;
        caplen_total += caplen;
        if (start_time.tv_sec == 0) {
                 if (interval != 0 && ts->tv_sec % interval != 0)
                        return (0);
                start_time = *ts; /* tcpdump データの開始時間を格納 */
        }
        end_time = *ts;
/* 設定した一定間隔の時間を経過したかどうかチェック */
        if (interval != 0 &&
            end_time.tv_sec - start_time.tv_sec >= interval) {
print_summry();         /* 現在のサマリを出力*/
             tree_resetcount();     /* ツリー構造に格納したデータの消
去  */
                start_time = *ts;
        }
        packet_length = len;
        return (1);
}
```

図 3.14: tcpdump ファイルから時間情報抽出と取り扱い

tcpdump ファイルから抽出された時間情報は，struct timeval 構造体に格納される．AGURI は格納された時間情報を順次比較し，設定した一定間隔を経過したかどうか確

認する．一定時間経過した場合は，現在のサマリを出力し，ツリー構造に格納したトラフィックデータを消去する．

AGURI データからの収集

複数の AGURI データからデータを収集する際の実行コマンドを，以下の図 3.15に示す．

```
root#aguri file1 file2 ...
```

図 3.15: AGURI データからのトラフィック収集

図 3.15に示した，”file1”,”file2” はそれぞれ AGURI データを表わしている．AGURI は，複数の AGURI データを入力することにより， 短期データから長期データの生成を可能する．例えば，10 時から 10 時 20 分までのデータ，10 時 20 分から 10 時 40 分までのデータ，10 時 40 分から 11 時まで 3 つの 20 分間の AGURI データを入力することにより，10 時から 11 時までの 1 時間のデータを作成することができる．

この機能を実現するためには，AGURI データを構成要素に分解する必要がある．この機能を実現する関数のプロトタイプ宣言を，以下の図 3.16に示す．

```
static int read_file(FILE *fp); /* AGURI ファイルの parse */
```

図 3.16: AGURI データの切り分け

AGURI ファイルは大きく分類し，”%” から始まるヘッダ部分とそれ以外のボディ部分の 2 つに分類できる．ヘッダ部分に記述されている要素は，AGURI のバージョン，トラフィックを収集した時間などであり，ボディ部には特定のアドレスやポートごとのトラフィック流量が記述してある．

これらの構成要素を分解，抽出することによって，ネットワークインターフェースから直接データを収集した場合と同様に取り扱うことを可能にした．

3.2.2 データ集約部

AGURI において，収集されたデータは以下に示す 4 つのツリー構造に格納される．

- source ip address
- destination ip address
- source protocol

- destination protocol

このツリー構造を構築するために，AGURI ではツリー中のノードとツリー構造全体を構造体で表現している．この 2 つの構造体を以下の図 3.17図 3.18に示す．

```
struct tree_node {
        struct  tree_node *tn_parent;     /* 親ノード */
        struct  tree_node *tn_left;       /* 左の子ノード */
        struct  tree_node *tn_right;      /* 右の子ノード */
        struct  tree *tn_tree;            /* ツリー構造全体を示すポイ
ンタ */
        TAILQ_ENTRY(tree_node) tn_chain;  /* LRU リストのエントリ */
        size_t  tn_prefixlen;             /* prefix */
        u_char  tn_key[16];               /* 流量 */
        u_char          tn_intree;
        short           tn_depth;         /* ルートノードからのツリー
の深さ */
        u_int64_t       tn_count;
};
```

図 3.17: ツリー中のノード構造体

AGURI ではパトリシアツリー構造を採用しているため，リーフノード以外の全てのノードは必ず左右に子ノードを保持する．また，ツリー中のノードは流量，prefix，ツリーの深さを値として保持する．

このように，各ノードが親ノード，子ノードに連結することによって，全体としてツリー構造を表現することができる．

```
struct tree {
        struct  tree_node *tr_top;              /* ルートノードへのポ
インタ */
        size_t  tr_keylen;                      /* 流量 */
        u_int   tr_nfree;                       /* free ノード数 */
        u_int64_t       tr_count;
        TAILQ_HEAD(_lru, tree_node)     tr_lru; /* LRU リスト */
};
```

図 3.18: ツリー全体の構造体

更に，図3.18に示したように，各ノードの連結から表現されるルートノードへのポインタを保持することによって，ツリー構造全体を指し示すことができる．

AGURIは収集したパケット毎に，これらのツリー構造へ流量の追加，ノードの追加と削除などを行うことによって，流量の多いトラフィックのみの格納をおこなっている．このツリー構造の操作に関連する関数のプロトタイプ宣言を以下の図3.19に示す．

```
/* ノードを追加する際のメモリ領域確保 */
static struct tree_node *leaf_alloc(struct tree *tp, const void *key,
                                struct tree_node *np);
/* ノード削除の際のメモリ領域解放  */
static void leaf_free(struct tree *tp, struct tree_node *leaf);
/*  リーフノードの追加  */
static void leaf_reclaim(struct tree *tp, int n);
/* ツリー構造を全て消去  */
static void tnode_reset(struct tree_node *np);
/* ツリー構造に保持された値の消去 */
static int tnode_resetcount(struct tree_node *np, void *arg);
 /* 子ノードに含まれる値を親ノードに加算  */
static int tnode_sum(struct tree_node *np, void *arg);
/* ノードの検索  */
static struct tree_node *tnode_find(struct tree *tp,
            const void *key, size_t len);
/* ノードの集約 */
static int tnode_aggregate(struct tree_node *np, void *arg);
 /* ツリー構造のデータを出力  */
static int tnode_print(struct tree_node *np, void *arg);
/* ノードの保持する値を出力 */
static int key_print(u_char *key, size_t len, size_t prefixlen);
```

図 3.19: パトリシアツリー構造を操作する関数

AGURIは，上記の図3.19中に示した関数を用いることにより，パケットを収集する度に，ツリー構造へのデータの追加，集約を行うことができる．

3.2.3 データ蓄積部

ツリー構造に格納されたデータは一定間隔ごとに保存されている．この間隔を設定するAGURI際の実行コマンドを，以下の図3.20に示す．

図3.20中の，”time_interval” が設定された一定間隔を示している．この間隔は秒単

```
root#aguri -s time_interval
```

図 3.20: データ収集間隔の設定

位で記述することができる．例えば”aguri -s 5” を実行した場合，5 秒毎にツリー構造のデータを保存し，あたらにツリー構造に収集データを格納する．

AGURI は”HUP シグナル” を受け取ることにより，一定間隔ごとに保存されたデータを全て出力する．この AGURI データの出力例を以下の図 3.21に出力例を示す．

```
%%!AGURI-1.0
%%StartTime: Thu Sep 01 00:00:00 2002 (2002/9/01 00:00:00)
%%EndTime:   Sun Oct 01 00:00:00 2002 (2002/10/01 00:00:00)
%AvgRate: 14.91Mbps

[src address] 4992392109177 (100.00%)
0.0.0.0/0       87902964189 (1.76%/100.00%)
 0.0.0.0/1      206637364377 (4.14%/14.78%)
 0.0.0.0/2      205796877844 (4.12%/7.12%)
   60.0.0.0/6   97928228974 (1.96%/3.00%)
        62.52.0.0/16    51875058871 (1.04%/1.04%)
    64.0.0.0/8  100831910967 (2.02%/3.51%)
     64.0.0.0/9 74610984109 (1.49%/1.49%)
 128.0.0.0/2    142349668983 (2.85%/13.33%)
  128.0.0.0/3   197067746696 (3.95%/10.48%)
   128.0.0.0/5  202911635757 (4.06%/5.45%)
    133.0.0.0/8 69142535628 (1.38%/1.38%)
                150.65.136.91   54123094932 (1.08%)
  192.0.0.0/4   212653628837 (4.26%/38.41%)
   192.0.0.0/6  88855538654 (1.78%/1.78%)
    202.0.0.0/7 235853368912 (4.72%/14.70%)
     202.0.0.0/9        117196493427 (2.35%/6.77%)
                202.12.27.33    160473669718 (3.21%)
             202.30.143.128/25  60239291958 (1.21%/1.21%)
                203.178.143.127 94031811680 (1.88%)
   204.0.0.0/6  228960094456 (4.59%/17.68%)
    204.0.0.0/8 125458765333 (2.51%/7.58%)
                204.123.7.2     87103414877 (1.74%)
                204.152.184.75  165733431144 (3.32%)
    206.0.0.0/7 164036959478 (3.29%/5.51%)
     206.128.0.0/9      53526598302 (1.07%/1.07%)
    207.0.0.0/8 57628266965 (1.15%/1.15%)
  208.0.0.0/4   282590640975 (5.66%/31.72%)
   208.0.0.0/6  116047154301 (2.32%/22.20%)
    209.0.0.0/8 140888988219 (2.82%/11.78%)
                209.1.225.217   238192306019 (4.77%)
                209.1.225.218   209160635530 (4.19%)
    210.0.0.0/7 154008321340 (3.08%/3.08%)
     216.0.0.0/9        192899750315 (3.86%/3.86%)
%LRU hits: 86.82% (1021/1176)
```

図 3.21: AGURI のプロファイリング結果出力例

図 3.21に出力例は，2002 年 9 月に収集した soure IP address ごとのトラフィック傾向を示している．例えば，203.178.143.127 を発信元とするパケットが 94031811680bytes 流れていることを意味しており，その値が総流量の約 1.88%を占めることを表している．更に，この 203.178.143.127 を含む IP アドレス空間 202.30.143.128/25 には 60239291958bytes

の流量があることを示している．

このように，AGURI に対して定期的に”HUP シグナル” を送信し，出力結果を順次蓄積することによって，長期にわたるトラフィック収集が可能である．

3.3 ネットワーク運用への適応

本章の設計，実装の節で述べてきたように，AGURI は短期から長期に渡るトラフィックの傾向抽出を可能にした．

しかし，実際のネットワーク運用に AGURI を用いる場合，この AGURI データを管理者が直感的に理解しやすい名前で，直感的に理解しやすい場所に蓄積するのが望ましい．また，AGURI の出力はテキスト形式であり，大量のデータを直感的に理解することは難しい．そのため，AGURI の出力結果を視覚化することが望ましい．

そこで AGURI では，ソースコードと同時に複数のスクリプトを提供することによって，AGURI をネットワーク運用においてより便利に使用できるようにしている．本節では，これらのスクリプトの説明を行った上で，AGURI の機能概要をまとめる．

3.3.1 長期にわたるトラフィック収集

AGURI に対して定期的に”HUP シグナル” を送信しすることにより，それまで AGURI で収集したデータを出力することができる．この出力結果を順次蓄積することによって，複数の短期間トラフィックデータを収集することが可能である．

また，AGURI には複数の AGURI データを組み合わせることによって，再帰的に長時間の AGURI データを生成することが可能である．そのため，最も時間粒度の細かいデータのみを収集し，定期的にそのデータから時間粒度の大きい AGURI データを作成ことによって，短期から長期にわたるトラフィック収集を可能にする．

AGURI と同時に配布している”agurify.pl” というスクリプトによって，これらの機能を実現している．図 3.22に agurify.pl の crontab への記述例を示す．

```
# run agurify.pl every 2 minutes
*/2 * * * *     /usr/local/lib/aguri/agurify.pl -i fxp0
```

図 3.22: cron による定期的な AGURI データの蓄積

このように，agurify.pl を crontab に記述することによって，定期的に AGURI が作成したサマリをファイルに書き込むことができる．更に，そのファイル名にトラフィック収集開始時間を記述することができる．また，agurify.pl は unix のディレクトリ構造を活用することによって，年，月，日，時に応じて階層的なディレクトリ構造を構成し，ファイル名に適合するディレクトリにデータを保存する．2003 年 1 月 1 日 1 時 10

分から 2 分に渡って収集されたデータを例にとって説明する．この例を以下の図 3.23に示す．

```
root# pwd
/aguri.log/2003/01/01/01
root#ls
20030101.0100.agr       20030101.0120.agr       20030101.0140.agr
20030101.0102.agr       20030101.0122.agr       20030101.0142.agr
20030101.0104.agr       20030101.0124.agr       20030101.0144.agr
20030101.0106.agr       20030101.0126.agr       20030101.0146.agr
20030101.0108.agr       20030101.0128.agr       20030101.0148.agr
20030101.0110.agr       20030101.0130.agr       20030101.0150.agr
20030101.0112.agr       20030101.0132.agr       20030101.0152.agr
20030101.0114.agr       20030101.0134.agr       20030101.0154.agr
20030101.0116.agr       20030101.0136.agr       20030101.0156.agr
20030101.0118.agr       20030101.0138.agr       20030101.0158.agr
```

図 3.23: agurify.pl による AGURI データの階層的蓄積

図 3.23に示したように，2003 年 1 月 1 日 1 時 10 分から 2 分に渡って収集されたデータは，“/aguri.log/2003/01/01/01/” に”20030101.0110.agr” というファイル名で蓄積される．

更に，このディレクトリの上位階層である “/aguri.log/2003/01/01/” には，20030101.01.agr というファイル名で，2003 年 1 月 1 日 1 時台全ての AGURI データから再帰的に作成された AGURI データが蓄積される．

3.3.2 トラフィックの視覚化

ネットワーク運用者は，順次蓄積された AGURI の出力結果を参照することによって，長期間にわたるトラフィック傾向を理解することができる．

しかし，ネットワーク運用者の直感的な理解を考慮した場合，このトラフィック傾向が視覚化されていることが望ましい．そこで，AGURI では gnuplot(28) というグラフ描画ソフトウェアへのサポートを行っている．このコマンドの実行例を以下の図 3.24に示す．

図 3.24に示したように，AGURI の”-P” オプションを使用することにより，入力された全ての AGURI データから，流量の多いノード数を抽出し，グラフ描画に必要な形式に変換する．

図 3.24中の，1 列目の項目がグラフにおける x 軸の値に適応され，2 行目以降の値がグラフの y 軸の値に適応される．

```
root#aguri -P aguri_file1 aguri_file2 ....
#[dst address]
#time total 192.168.0.1/24 10.0.0.1/24 192.168.0.50 10.0.0.100
2002/01/20:20:03:57 133465846 58917175 22270181 6079629 17416593
2002/01/20:20:05:57 200002634 131983828 19645774 7929372 14222195
2002/01/20:20:07:57 111259598 47331996 18638768 5461640 14004718
:
:
2002/01/20:20:55:57 161950992 65099353 14061061 8988122 29136263
2002/01/20:20:57:57 133135027 38934269 18064895 7453381 23622866
2002/01/20:20:59:57 132684021 41008924 19881333 6721708 21505533
```

図 3.24: AGURI による gnuplot サポート

また，図 3.24では destination address の結果が出力されているが “-x” オプションを用いることにより source address ,destination address, source port,destination port の結果も出力することができる．

更に，AGURI と同時に提供されている”makeplot.pl” というスクリプトを用いることにより，gnuplot の設定ファイルを動的に記述することができる．このスクリプトの出力結果を gnuplot に入力することによって，ネットワーク運用者は png 形式のグラフ画像を入手できる．

これらの AGURI の plot 機能と makeplot.pl スクリプトを web ブラウザ越しに利用することによって，蓄積されたデータの全ての時間の出力結果を参照することができる．この例を図 3.25，図 3.26に示す．

図 3.25は参照したい AGURI データの入力フォームである．ネットワーク運用者は参照したい時間をフォームに記入することによって，当該時間のトラフィック傾向を示したグラフ画像と AGURI データを参照することができる．このフォームデータは，以下に示す 3 つのフィールドを持っている．

- 年，月，日，時
- 時間粒度

  画像を作成する際の構成要素の時間粒度を設定する．分刻み，時間刻み，日刻み，月刻みの 4 つから選択可能．
- 出力対象

  source address,destination address,source port,destination port の 4 つから選択．

図 3.26は，フォームに入力された情報から作成されたグラフ画像と AGURI データである．画像処理によって特定のホストのアドレスなどは見えないように変換してあ

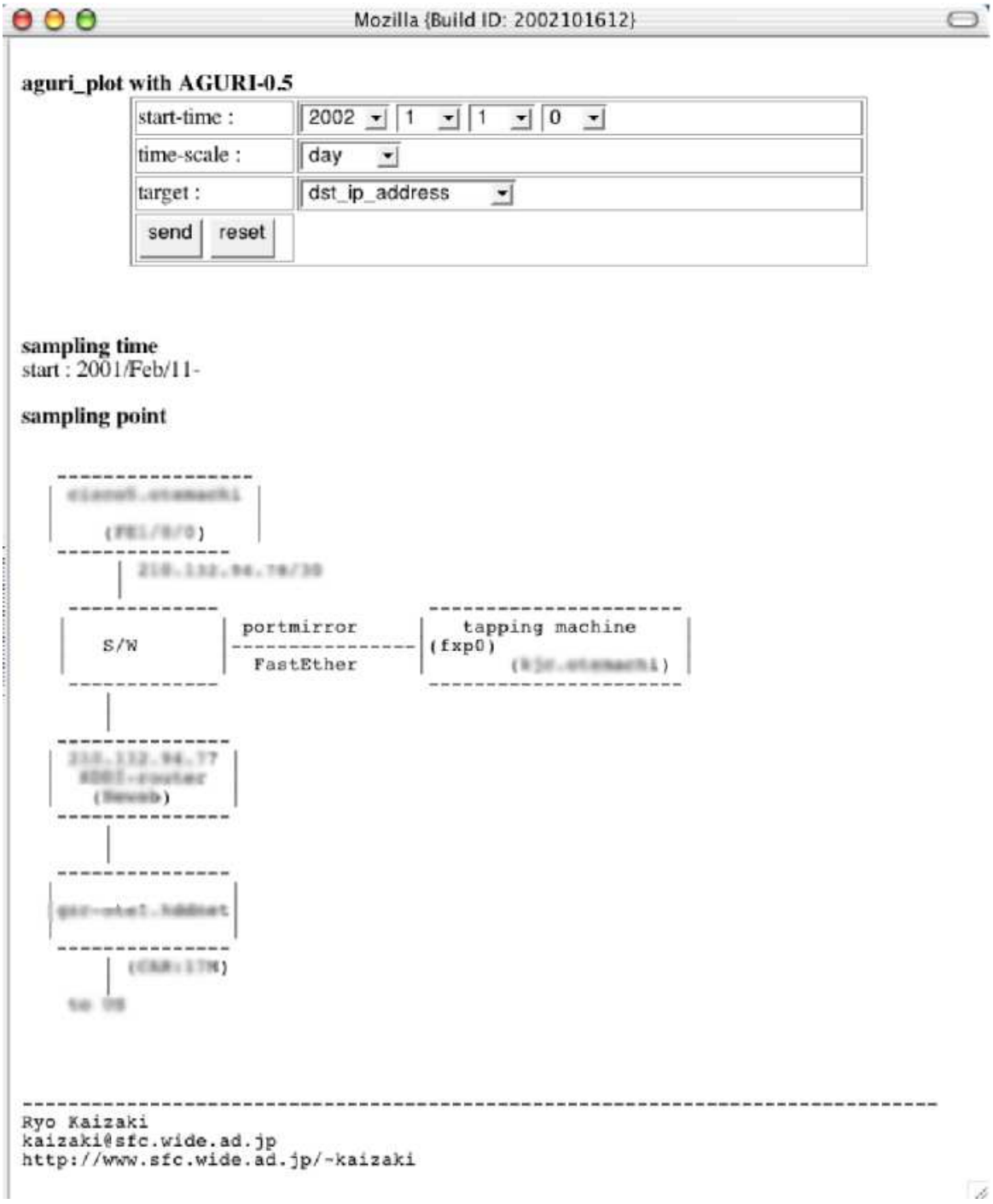

図 3.25: AGURI データ参照用フォーム

るが，図 3.26中には，流量の多い IP アドレスブロックやホストの情報が時系列で表示されている．

図 3.26に描画された青色の折れ線グラフ見られるように，flooding 攻撃を受けた場合は，グラフの形が急激に変化する．これによって，ネットワーク運用者は直感的に攻

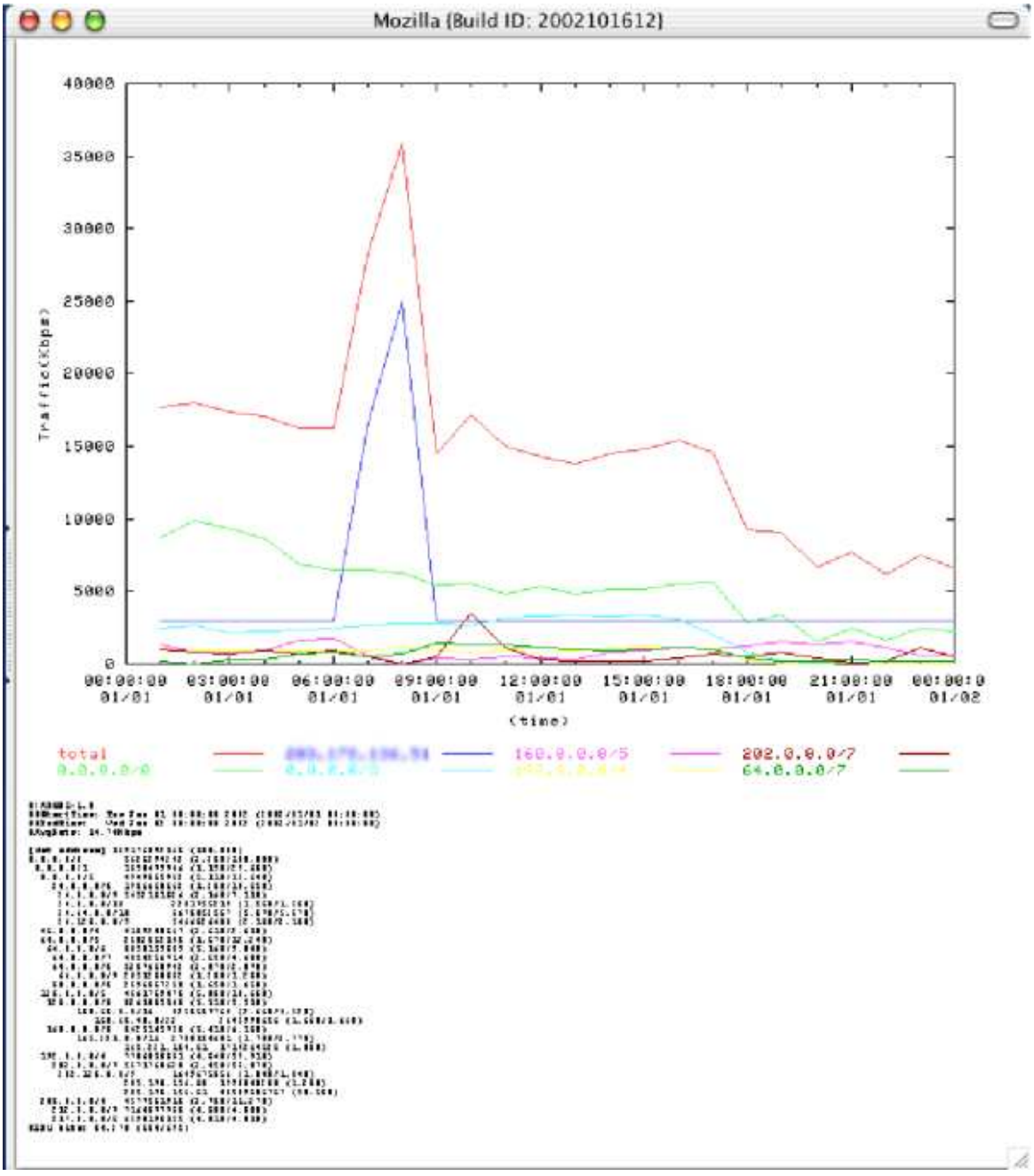

図 3.26: web ブラウザ越しの AGURI データ参照

撃を検知することができる．また，画像の下に記述された AGURI データを詳しく参照することによって，flooding 攻撃の特性を詳細に知ることができる．

3.3.3 まとめ：AGURI の機能概要

AGURI はトラフィックを集約することで，短期から長期に渡るトラフィックモニタリング可能とした．また，小さなフローを集約するため，広大なアドレス空間にも対応している．そしてアウトプットの際も同様のアルゴリズム，つまり短い時間を集約することで時間的な集約を行っている．一連のスクリプトは様々な時間尺度でトラフィックの集約を行い，更には視覚化を行うことができる．

AGURI の特徴は前もってルールセットを定義することなく，flooding 攻撃に代表されるトラフィックの急激な増加を検知することができることである．

AGURI は送信元アドレス，宛先アドレス，送信元ポート番号，宛先ポート番号の 4 つのプロファイルを生成する．IP アドレスはもともと階層的かつ集約できるように設計されているため，集約というアルゴリズムを簡単に用いることができる．IP アドレスに関しては IPv4 と IPv6 の両方に対応している．一方，プロトコル番号は階層的でないが，ポートの範囲を特定するのと同様の手法を用いることが可能である．具体的には，IP のバージョン，プロトコル番号，そして TCP と UDP のポート番号を結び付けることで，プロトコルプロファイルのための 32 ビットのキーを生成する．

AGURI は libpcap を用いることでネットワークトラフィックをモニタし，HUP シグナルを受け取った際にサマリを生成する．また，動いている AGURI プログラムに対して，cron から HUP シグナルを送ることによって，定期的なサマリを取得することができる．

第4章　実トラフィックの収集

本章では，実際にネットワークにおいて収集したデータに関して述べる．まず，収集ポイントや収集マシンなどの収集環境について述べ，その後，収集したデータについて述べる．

4.1　実データの収集に関して

本研究の目的である flooding 攻撃特徴を検出には，実際のインターネットにおけるトラフィックを収集が必要である．

第 3章で述べたように，AGURI がトラフィックを収集する手法は以下の 3 つである．

- ネットワークインターフェースから直接収集する
- tcpdump データからデータを収集する
- AGURI データから収集する

上記の 3 つの収集方法は全て，収集ポイントに収集マシンを設置する方法で実現される．

しかし，トラフィック情報にはユーザのプライバシに関する情報やネットワークの運用ポリシに関する情報が大量に含まれているため，ネットワーク上におけるトラフィック収集を許可する ISP は存在しない．

そこで本研究では，著者自身も運用に携わっている WIDE インターネットにおいてデータを収集する．尚，収集の際には，以下の 3 つの点を踏まえている．

- パケットのヘッダ部分の情報のみを利用すること
- 収集したデータは研究，運用目的に用いる
- 研究目的に用いる場合でも，データを外部に公開する際はホストやユーザの情報が特定されないように隠蔽もしくは変換をおこなう

このため，第 5章で述べるデータについても，ホストやユーザが特定されないように IP アドレスやホスト名を変換している．

4.1.1 WIDE インターネット

WIDE インターネットは WIDE プロジェクトの運用するネットワークである．WIDE インターネットは大規模分散環境の構築のための実験環境としての位置づけをもっており，産官学の組織によって共同運用されている．

WIDE インターネットは札幌，仙台，東京，八王子，藤沢，浜松，北陸，岐阜，奈良，大阪，京都，広島，福岡，サンフランシコの 14 個所の NOC(Network Operation Center) を中心拠点に約 140 の組織を接続している，また，2 本の国際専用回線を用いて国際的なインターネットとの相互接続も実現している．

WIDE インターネットは 203.178.136/21 の IP アドレス空間を保持し，AS2500 番という AS 番号を割り当てられている．また，”wide.ad.jp ”というドメイン空間の運用を行っている．

4.1.2 データ収集ポイント

本研究では，WIDE インターネットの 3 個所でデータ収集を行う．この 3 個所はいずれも国際専用回線のデータである．このデータ収集ポイントを以下の図 4.1に示す．

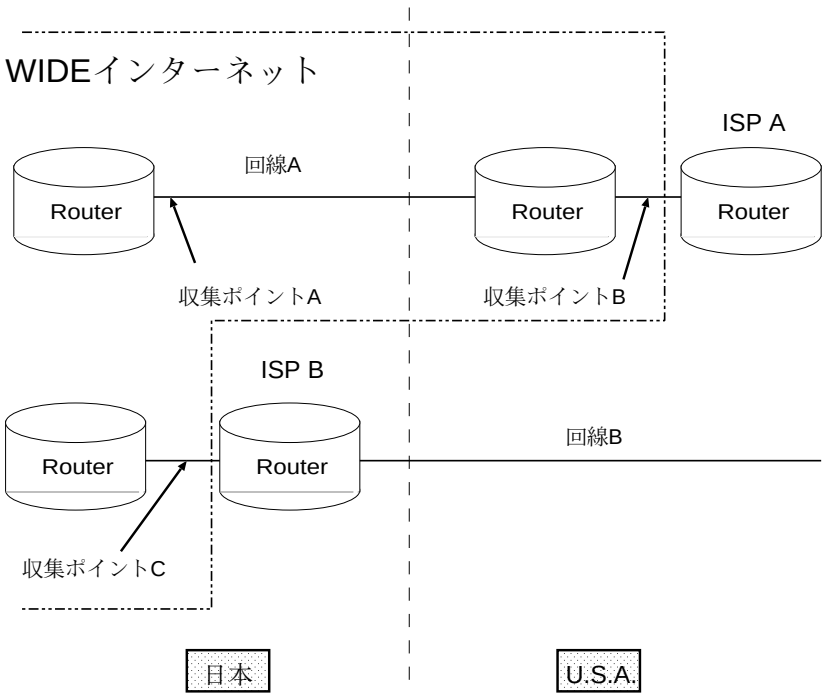

図 4.1: データ収集ポイント

図 4.1に示したように，本研究で選択した 3 個所のデータ収集ポイントは，全て国際線のトラフィックデータを収集する可能なポイントである．国際線を選択した 2 つの理由は，1)，国際線が WIDE インターネットの出入り口であること 2) 国際線の回線帯域と国内バックボーンの回線帯域の格差が存在することである．

第一に，WIDE インターネットは外部組織との接続を，この 2 本の国際回線と NSPIXP2(29) などの国内 IX(30) による相互接続によって実現している．flooding 攻撃の検知を考えた場合，攻撃は一般的に外部から来るため組織の出入り口でデータを収集するのが適当である．

第二に，WIDE インターネットの主要な国内バックボーンは Gigabit Ethernet で構成されている，しかし，2 本の国際専用線環境はそれより劣っている．この 2 本の国際線の環境を以下に示す．

- 回線 A

  回線 A は，国際回線自体は OC-3(31) で接続されているため，約 155Mbps の回線環境を持っている．また，回線 A の先に接続している ISP A との回線は 60Mbps で接続されている．

- 回線 B

  回線 B は，回線自体は ISP B の資源である．WIDE インターネットはその ISP B の持っている回線の 17Mbps を使用するサービスを購入している．

上記の回線 A, 回線 B の両者とも Gigabit Ehternet で構成される国内バックボーンと比較した場合，回線帯域が狭い．flooding 攻撃を WIDE インターネット内の計算機が受けた場合，この回線帯域が狭い場所のネットワーク資源が枯渇する可能性が非常に高い．そのため，本研究では国際線のデータを収集する．

本研究では，収集ポイント A で収集したデータを” データ A”，収集ポイント B で収集したデータを” データ B”，収集ポイント C で収集したデータを” データ C” としする．

4.1.3 データ収集マシンの計算機環境

データ A とデータ C は同一計算機で収集している．データ A，データ C を収集しているマシンをマシン A, データ B を収集しているマシンをマシン B とする．表 4.1にマシン A，マシン B の計算機環境を示す．

表 4.1: 収集マシンの計算機環境

	マシン A	マシン B
OS	FreeBSD 4.7-STABLE	FreeBSD 4.7-STABLE
CPU	Pentium III 1.2GHz	Pentium III 1.2GHz
Memory	256MB	128MB
Hard Disk	50GB	80GB
Network Interface	Fast Ehternet(intel eepro100)× 2 Gigabit Ethernet(Netgear GA620)	FastEthernet(intel eepro 100)× 2

本研究では， 表 4.1に示した 2 台の計算機を用いてトラフィックデータを収集する．また，実際のトラフィック収集においては，スイッチのポートミラー機能 (32) を利用する．この機能を利用することによって，当該ルータに負荷をかける事なく，国際線に接続されたルータを通過する全てのパケットと同様のパケットを収集することが可能である．

4.2 収集データ

本節では収集したデータの基礎的な情報について述べる．

4.2.1 期間

トラフィック収集開始年月日を以下に示す．

- データ A: 2002 年 1 月 16 日
- データ B: 2002 年 10 月 14 日
- データ C: 2001 年 2 月 11 日

これらの収集ポイントでは現在も引き続きデータ収集を行っている．また，以下の表 4.2，表 4.3，表 4.4に収集された AGURI データの量を示す．

表 4.2: データ A の AGURI データ量

年	月	データ量 (単位:KB)
2002 年		1622122
	1 月	72940
	2 月	133725
	3 月	132124
	4 月	144031
	5 月	146769
	6 月	139612
	7 月	125349
	8 月	147437
	9 月	145673
	10 月	147818
	11 月	142263
	12 月	144097

上記の 3 つの表に示したように，AGURI を用いた場合，長期にわたるトラフィック収集でも蓄積されるデータ量を抑制することが可能であると言える．

表 4.3: データ B の AGURI データ量

年	月	データ量 (単位:KB)
2002 年		455238
	10 月	157212
	11 月	148166
	12 月	149860

表 4.4: データ C の AGURI データ量

年	月	データ量 (単位:KB)
2001 年		1397188
	2 月	80935
	3 月	136817
	4 月	135051
	5 月	134250
	6 月	129944
	7 月	133586
	8 月	132316
	9 月	125443
	10 月	129998
	11 月	123588
	12 月	135188
2002 年		1584280
	1 月	137266
	2 月	122898
	3 月	133218
	4 月	135188
	5 月	133606
	6 月	129310
	7 月	132232
	8 月	138121
	9 月	131717
	10 月	130205
	11 月	127546
	12 月	132892

4.2.2 収集データの種別

本研究で収集した AGURI データは以下の 4 つである.

- source IP address
- destination IP address
- source port number
- destination port number

ただし，source port number と destination port number のデータには，IP のバージョンとプロトコルの情報も同時に収集されている．この例を以下の図 4.2に示す．

```
[ip:proto:srcport] 57935749616221 (100.00%)
0/0:0:0 409305060882 (0.71%/100.00%)
4:0/3:0 2742521188175 (4.73%/99.29%)
                4:1:0   3162265669467 (5.46%)
4:6:0/0 1199135819954 (2.07%/85.74%)
     4:6:0/5    1576615645630 (2.72%/57.18%)
                4:6:20  2250368259498 (3.88%)
                4:6:80  24282139890554 (41.91%)
      4:6:1024/6        2203295665900 (3.80%/8.66%)
       4:6:1024/7       1130048721229 (1.95%/4.86%)
                4:6:1214        1684972755864 (2.91%)
     4:6:2048/5 2067480098695 (3.57%/4.93%)
      4:6:3072/6        788716463034 (1.36%/1.36%)
    4:6:4096/4  2646309460859 (4.57%/11.76%)
     4:6:6144/5 932164817043 (1.61%/7.19%)
                4:6:6346        976651859576 (1.69%)
                4:6:6699        2256191547961 (3.89%)
 4:6:32768/1    1579319777739 (7.13%/7.13%)
 4:17:0/1       821804386840 (1.42%/3.36%)
                4:17:53 1124097819260 (1.94%)
```

図 4.2: AGURI データ（ポート番号）の出力例

図 4.2中の，”4:6:1214 1684972755864 (2.91%)” という行は，IPv4 で プロトコル番号 6(= TCP) で 1214 番をソースポートとするデータが，1684972755864byte 流れ，そのデータ量は総トラフィック量の 2.91%にあたることを意味している．

4.2.3 収集データの長期的特徴

本小節では，3 個所で収集した AGURI データの長期的傾向を示す．これを以下の表 4.5に示すように，図 4.3から図 4.18に示す．

尚，データ A，データ B は 2002 年分のデータを，データ C は 2001 年分，2002 年分の 2 年分のデータを用いる．

表 4.5: 長期 AGURI データ一覧表

	source ip address	destinaton ip address	source port	destination port
データ A(2002 年)	図 4.3	図 4.4	図 4.5	図 4.6
データ B(2002 年)	図 4.7	図 4.8	図 4.9	図 4.10
データ C(2001 年)	図 4.11	図 4.12	図 4.13	図 4.14
データ C(2002 年)	図 4.15	図 4.16	図 4.17	図 4.18

長期データを収集し，可視化することによって，トラフィック量の増加傾向や流行しているアプリケーションなどを見いだすことができる．

本研究で図 4.3から図 4.18に示された図からは，以下のことが分かる．

- hostA 宛のトラフィック量が定常的に多い

  図 4.4に hostA のアドレスが抽出されている．このことからデータ A 中のには hostA 宛のデータが定常的に大量に流れていることがわかる．

- トラフィック流量の変化が急激である

  グラフ中の total 値の増減が急激であることから，総トラフィックの流量の変化が急激であることがわかる．また，逆説的に捕らえれば，回線は定常的には余裕があるといこうことが分かる．これは，定常的に回線がトラフィックで埋まっている場合，トータルのトラフィック流量は時間によってそれほど変化しないからである．

- web トラフィックが多い

  全ての source port のグラフに”4:6:80” が抽出されている．これは，IPv4 の tcp の 80 番ポートを送信元としたパケットが多いことを意味している．つまり，HTTP トラフィックである．

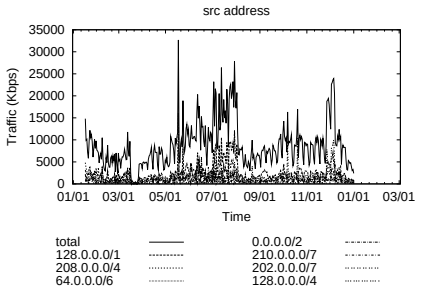

図 4.3: データ A soure ip address

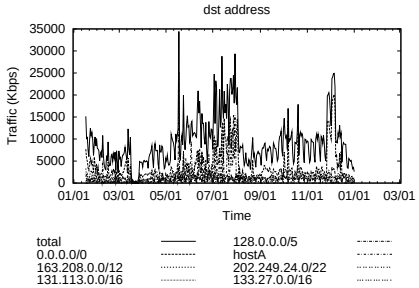

図 4.4: データ A destination ip address

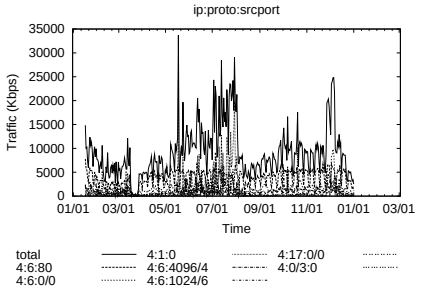

図 4.5: データ A source port

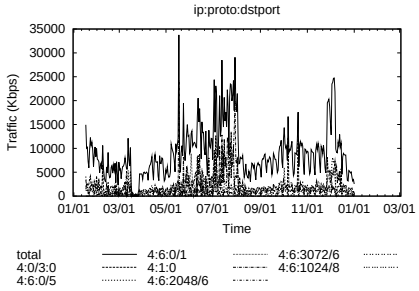

図 4.6: データ A destination port

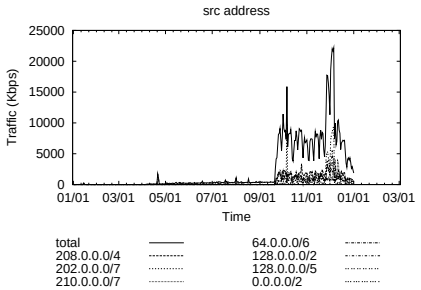

図 4.7: データ B source ip address

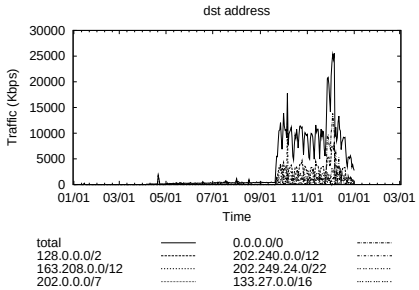

図 4.8: データ B destination ip address

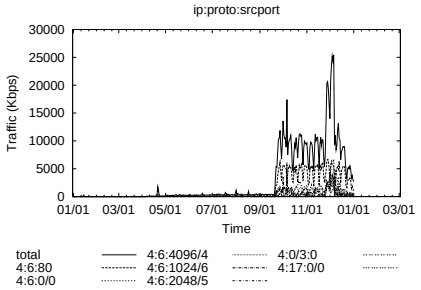

図 4.9: データ B source port

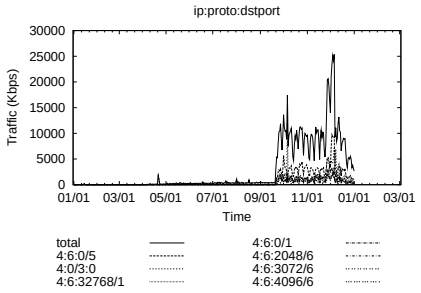

図 4.10: データ B destination port

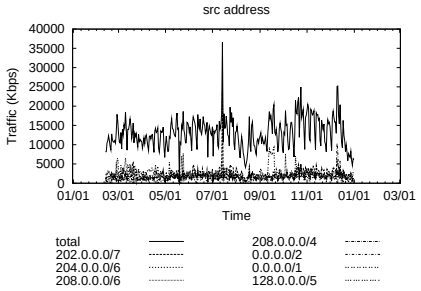

図 4.11: データ C (2001 年) source ip address

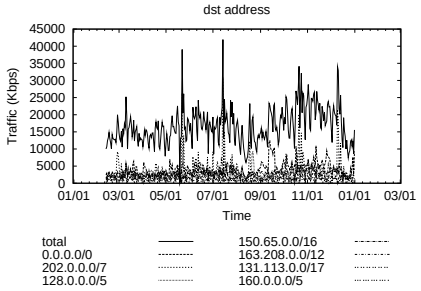

図 4.12: データ C (2001 年) destination ip address

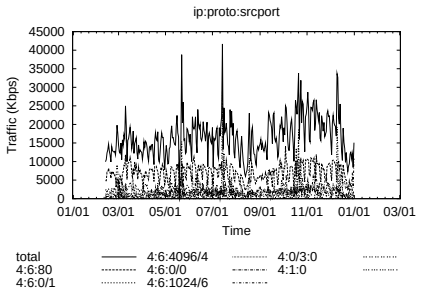

図 4.13: データ C (2001 年) source ip address

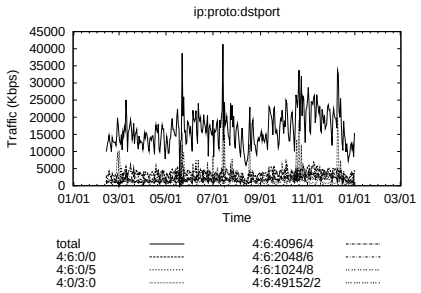

図 4.14: データ C (2001 年) destination ip address

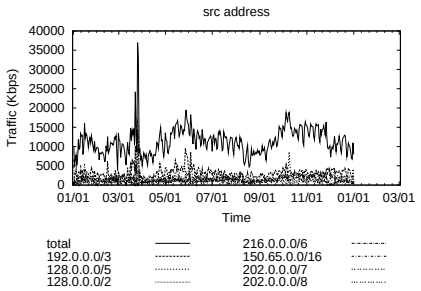

図 4.15: データ C (2002 年) source ip address

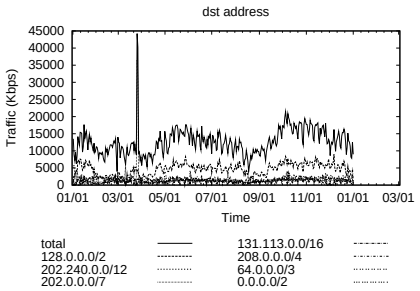

図 4.16: データ C (2002 年) destination ip addres

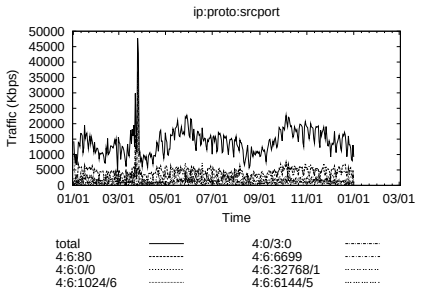

図 4.17: データ C (2002 年) source port

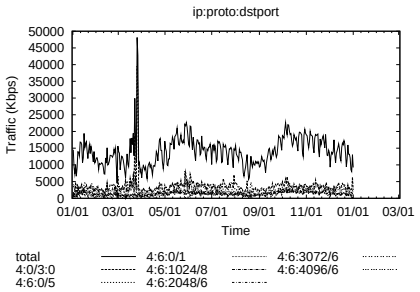

図 4.18: データ C (2002 年) destination port

第5章　flooding攻撃の傾向分析

本章では収集したデータから，flooding 攻撃の情報を抽出し，攻撃の特徴を分析する．また，その分析結果からネットワーク運用や flooding 攻撃対策ソフトウェアを支援する提言を行う．

5.1　収集したデータにみられるflooding攻撃

本節では，まず本研究における flooding 攻撃の定義を行う．そのうえで，定義に合致するデータを第 4章で述べたデータ A，データ B，データ C の 3 つの収集データから抽出し，攻撃の発生頻度を述べる．

5.1.1　flooding攻撃の定義

flooding 攻撃を抽出し傾向を把握するためには，攻撃パケットをフロー単位に集約する必要がある．なぜなら，logic 攻撃の場合と異なり，flooding 攻撃はパケット一つ一つを観察した場合には通常のトラフィックと差別化することができないからである．そのため，パケットの集合であるフロー単位でトラフィックを観察することにより，初めて flooding 攻撃を検知することができる．

一般的にフローは，5-tipuls と呼ばれる以下に示す 5 つの要素が同一のパケットの集合と定義される場合が多い．

1. source ip address
2. destination ip address
3. source port number
4. destination port number
5. protocol

しかし，flooding 攻撃を観測する上で 5-tipuls を用いたフローの定義を適応することは適切ではない．なぜなら，攻撃者によるネットワーク資源を占有は，port scan に代表される異なる port 番号のパケットによる攻撃や，複数のソースアドレスを詐称した攻撃によって引き起こすことができるからである．このような攻撃手法を用いた場

合，5-tipuls のフロー定義では，大量のフローが検出されるだけで，攻撃自体の特徴を抽出することはできない．

そこで，本研究では，flooding 攻撃の定義をトラフィック傾向に現れる症状を基に柔軟に行う．具体的には以下に示す 4 つの条件を一つでもみたしたものを flooding 攻撃フローと定義する．

1. AGURI データ (source ip address) 内で，総トラフィックの 20%以上を占有するフロー

   AGURI では，IP アドレス空間からの流量とホストからの流量の両者を抽出することができる．そのため，抽出されたホストの IP アドレスが，同じく IP アドレス空間に含まれる場合がある．本研究では，この場合最長適合を用い，ホストからの流量を採用する．例えば，”192.168.0.5” というホストの IP アドレスと，”192.168.0/24” という IP アドレス空間を抽出した場合，本研究では最長適合する”192.168.0.5” をフローとして採択する．

   また，抽出された IP アドレス空間が”10.0.0.0/4” のように，あまりに広大である場合，その IP アドレス空間から分散してトラフィックが送信されたことを意味し，flooding 攻撃の特徴として捕らえるには現実的ではない．そのため，本研究では”/24” より狭い IP アドレス空間のみをフローの対象とする．

2. AGURI データ (destination ip address) 内で，総トラフィックの 20%以上を占有するフロー

   source ip address の AGURI データと同様の定義を行う．

3. AGURI データ (source port) 内で，総トラフィックの 20%以上を占有するフロー

   AGURI では，IP のバージョン，使用されるプロトコル，ポート番号を抽出することができる．しかし，IP のバージョンが同様であるだけ，使用されるプロトコルが同様であるだけでフローを定義すると，ほぼ全てのパケットがわずかなフローに集約される．そのため，本研究では，ポート番号まで同一のパケットの集合を同一フローであると定義する．

4. AGURI データ (destination port) 内で，総トラフィックの 20%以上を占有するフロー

   source port の AGURI データと同様の定義を行う．

上記の 4 つ条件はいずれも，総トラフィックの 20%以上のフローを検出することである．総トラフィックの 20%以上の流量のあるフローを検出するためには，全てのフローの中から，流量の多いフローを上位 5 つ抽出することで可能になる．

また，AGURI を用いた flooding 攻撃の検知はトラフィック傾向の偏差を算出することによっても可能である (33)．

しかし，この手法を用いた場合，攻撃の検知のみ可能であり，どのフローが攻撃を引き起こしているかを知ることはできない．また，長時間に渡る flooding 攻撃フローの検知には向いていない．そのため，本研究ではこの手法を用いない．

また，フローの開始からフローの終了は AGURI によるトラフィックデータの収集を基に行い，AGURI による検出が始まった瞬間から検出ができなくなった時点までを 1 つのフローとして扱う．これを，以下の図 5.1を用いて説明する．

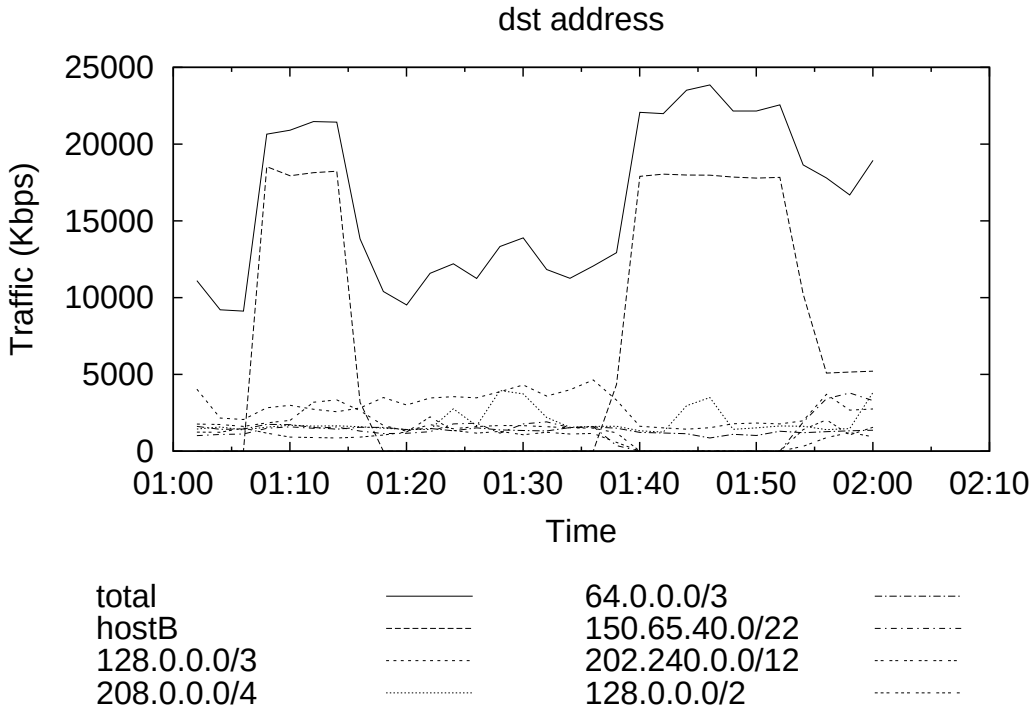

図 5.1: 2 回の flooding 攻撃例

上記の図 5.1では，1 時 6 分頃から hostB 宛の flooding 攻撃が始まり，1 時 18 分頃に収束した．その後，改めて 1 時 36 分頃から同様の flooding 攻撃が始まり，2 時にいたっても収束していない．

この場合，本研究の定義では，この flooding 攻撃は 1 時 18 分に一度終了し，1 時 36 分から新しい flooding 攻撃が来たと判断し，2 回の flooding 攻撃があったものとする．

5.1.2 flooding 攻撃の発生頻度

上記の定義に基づいて，データ A，データ B，データ C のそれぞれのデータから flooding 攻撃の発生回数を集計した．データ A についての攻撃発生回数を以下の表 5.1に，データ B についての攻撃発生回数を以下の表 5.2に，データ C についての攻撃発生回数を以下の表 5.3に示す．

上記の表 5.1，表 5.2，表 5.3に示されたように，2002 年の 8 月を境に flooding 攻撃の

表 5.1: データ A における flooding 攻撃発生件数

年	月	回数
2002 年		306
	1 月	45
	2 月	39
	3 月	62
	4 月	32
	5 月	34
	6 月	36
	7 月	43
	8 月	2
	9 月	2
	10 月	1
	11 月	2
	12 月	8

表 5.2: データ B における flooding 攻撃発生件数

年	月	回数
2002 年		11
	10 月	1
	11 月	2
	12 月	8

発生件数が激減している．また，2002 年の 12 月にかけて再び flooding 攻撃の発生件数が増加しはじめている．

これは，WIDE インターネットのネットワーク運用における flooding 攻撃対策の影響である．WIDE インターネットでは頻発する flooding 攻撃対策として，2002 年 8 月 12 日と 8 月 21 日に，それぞれ接続している ISP に特定の IP アドレスを送信先とするパケットのに対するフィルタ記述を依頼した．そのため，9 月以降は flooding 攻撃がほとんど検出されなかった．その後，2002 年 12 月に入ってから，フィルタ依頼をしていない別の IP アドレスを送信先とするパケットを用いた攻撃が発生したため，再び flooding 攻撃が検出されるようになった．

5.2 flooding 攻撃の特徴

本節では，抽出された flooding 攻撃のデータを以下に示す 3 つの視点から分析を行う．

- 攻撃に使用される手法の推移

表 5.3: データ C における flooding 攻撃発生件数

年	月	回数
2001 年		184
	2 月	15
	3 月	27
	4 月	21
	5 月	19
	6 月	8
	7 月	27
	8 月	6
	9 月	13
	10 月	18
	11 月	14
	12 月	16
2002 年		149
	1 月	18
	2 月	12
	3 月	25
	4 月	9
	5 月	12
	6 月	23
	7 月	29
	8 月	12
	9 月	3
	10 月	0
	11 月	1
	12 月	5

- 複数データの同時間における関連性
- 攻撃の時間的推移

その上で，発見された顕著な 4 つの特徴を述べる．

5.2.1 攻撃先 IP アドレス

収集した source ip address の AGURI データから，flooding 攻撃を受けた攻撃ホスト，並びに，IP アドレス空間を抽出することができる．この攻撃先 IP アドレスの flooding 攻撃データを解析した結果，非常に顕著な特徴が現れた．その傾向とは，攻撃先 IP アド

レスが一部のアドレスに限定されていることである．しかも，このアドレスは大きく分て 1)IRC(Internet Relay Chat)(34) サーバの IP アドレス，2) ルータインターフェースの 2 つに分類することができる．この攻撃対象別 flooding 攻撃発生件数を表 5.4に示す．

表 5.4: 攻撃対象別 flooding 攻撃発生件数

攻撃対象	発生件数
irc server	548
irc server 1	299
irc server 2	103
irc server 3	146
router's interface	97
router 1	32
router 2	24
router 3	30
router 4	11
その他	5

表 5.4から flooding 攻撃は明らかに，irc サーバとルータのインターフェースを攻撃先にする場合が多いと言える．

以下，irc サーバ，ルータインターフェースに関して詳細な特徴を述べる．

- IRC サーバ

  WIDE インターネットが外部組織に対して広くサービスを提供している IRC サーバは全部で 3 台である．flooding 攻撃は，その 3 台全てを攻撃対象にしている．これは，WIDE プロジェクトの保持する広大な IP アドレス空間から考慮した場合も，偶然 IRC サーバに攻撃が集中したとは考えにくい．

  また，WIDE プロジェクトの提供する IRC サーバは 2002 年に IP アドレスの変更を行ったが，変更後も IRC サーバへの flooding 攻撃数は変化しなかった．

  一般に，flooding 攻撃はネットワークの提供する主要サービスである，メールサーバや，ウェブサーバ，ネームサーバに集中すると言われているが，本研究の解析ではそれらのサーバに対する flooding 攻撃は検出されなかった．

- ルータインターフェース

  WIDE インターネット内には少なく見積もっても 100 台以上のルータが存在し，300 以上のルータインターフェースが存在する．しかし，収集した flooding 攻撃が対象としている 4 つのルータインタフェースは全て，バックボーンを担う主要なルータのインタフェースである．

更に，攻撃先別の分析では稀な例であるが，複数の IP アドレスを攻撃先とする flooding 攻撃を検出することができた．この攻撃を以下の図 5.2に示す．

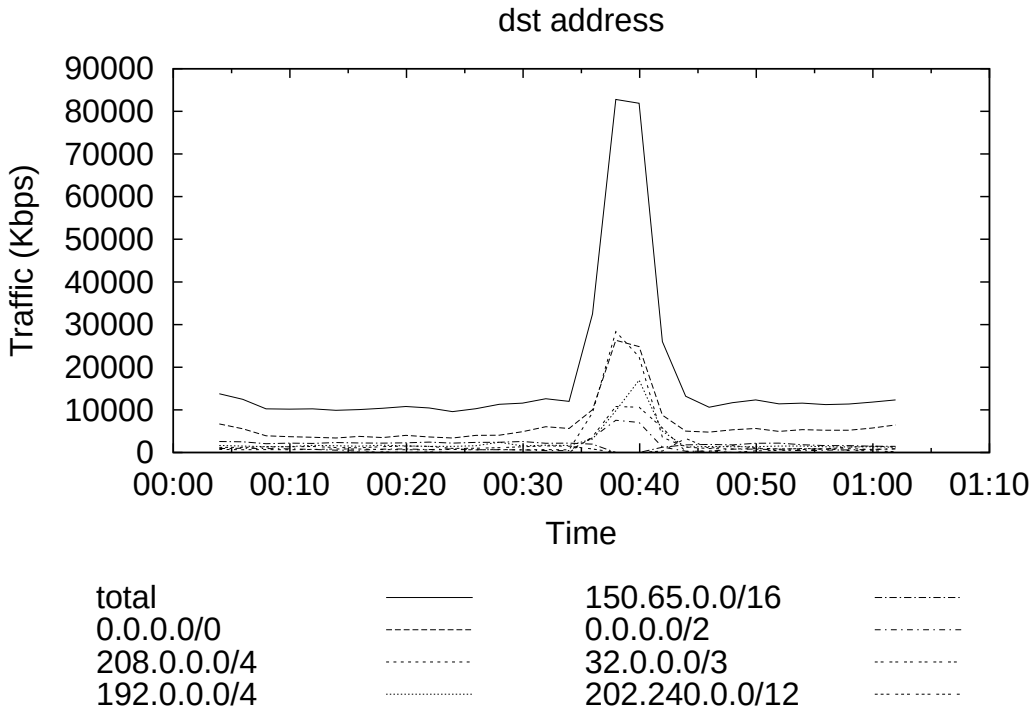

図 5.2: 分散した攻撃先 IP アドレスへの flooding 攻撃

こ図 5.2はデータ C において，2002 年 4 月 30 日に観測された攻撃を示している．この攻撃は 0 時 35 分頃から始まり，10 分程度で収束している．この攻撃の特徴は，攻撃先 IP アドレスが広範囲に分散していることである．

この攻撃手法で flooding 攻撃を行った場合，ホスト単位でのトラフィック収集ではこの攻撃を検知できない．AGURI の IP アドレス集約機能によってのみ収集できる攻撃手法である．

5.2.2 DDoS 攻撃の検出

本研究では同時間の異なるデータを比較することにより，DDoS(Distributed Denial of Service)(35) 攻撃を検出した．検出した DDoS 攻撃を，以下の図 5.3，図 5.4に示す．

図 5.3，図 5.4はそれぞれ Host1 に対する flooding 攻撃を表している．図 5.3，図 5.4両者を比較すると，同じ時間に flooding 攻撃が終了し，その後，同じ時間に再開していることがわかる．

一般的に，ネットワーク上の一地点でトラフィック収集を行った場合，flooding 攻撃を検知することは可能であるが，その攻撃が分散した複数台のマシンからの攻撃であるかどうかを判断することはできない．

これは，ホストによりソース IP アドレスの詐称が可能であるためである．つまり，収集した flooding 攻撃のソース IP アドレスが分散していたとしても，それは攻撃元 IP

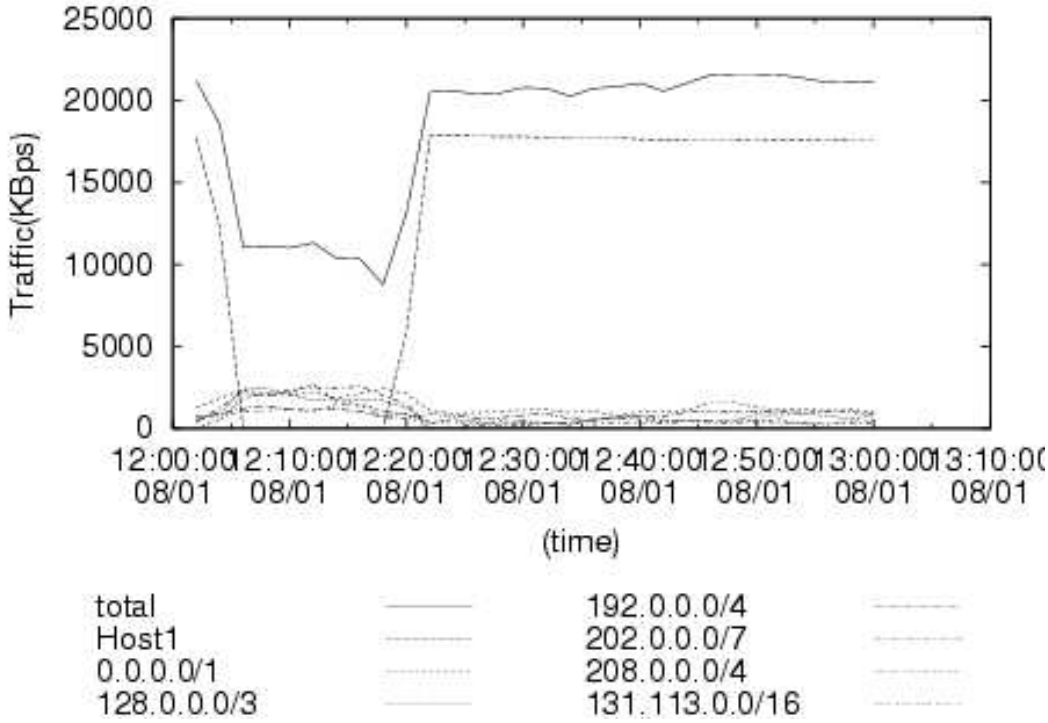

図 5.3: データ A における DDoS 攻撃

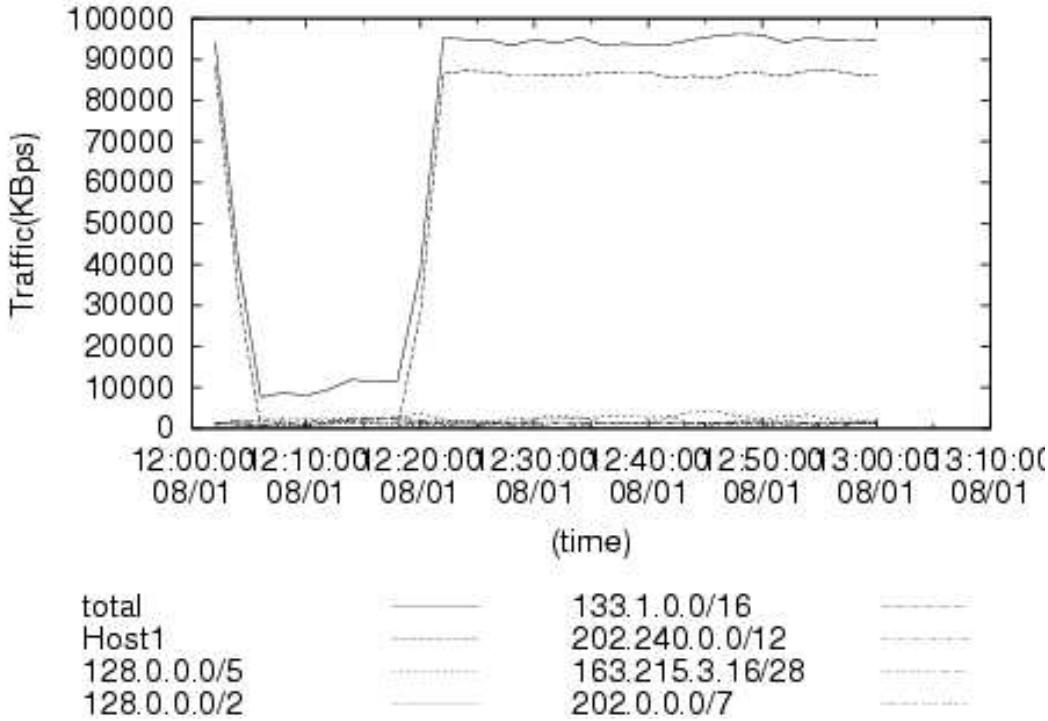

図 5.4: データ C における DDoS 攻撃

アドレスを隠蔽化するための常套手段であり，本当に複数台のマシンからの攻撃であると判断できないためである．

しかし，本研究では，複数地点でトラフィック収集を行うことにより，DDoS 攻撃を検出することができた．

データ A，データ C のそれぞれのデータは，それぞれ異なる国際線から収集されたものであり，接続している ISP も異なる．そのため，1 台のマシンから送信された同じホスト宛パケットが，同時間に複数の国際線を通過することは無い．つまり，異なるネットワークに所属する 2 台以上のマシンから flooding 攻撃が行われたと言える．

5.2.3 攻撃手法

収集データの解析の結果，flooding 攻撃に用いられる手法は，以下にしめす 3 つのプロトコルを使用する手法が検出された．これら 3 つの手法について個別に特徴を述べる．

1. TCP を用いた攻撃

   TCP を用いた攻撃の場合，同一のホストの複数のポートに対して大量のデータを送信する攻撃手法が多く検出された．

   また，1 つのポート宛にパケットを送信する flooding 攻撃は検知することができなかった．

2. UDP を用いた攻撃

   UDP を用いた攻撃の場合，TCP と同様，同一のホストの複数のポートに対して大量のデータを送信する攻撃手法が多く検出された．

3. ICMP を用いた攻撃

   ICMP を用いた攻撃は，ICMP echo reply を用いた攻撃手法が多く検出された．これは，一般的に反射攻撃と言われる，攻撃者が攻撃対象の IP アドレスを詐称し，複数のマシンに ICMP echo request を送信するタイプの攻撃手法である可能性が高い．

上記の説明のように，TCP や UDP を用いた攻撃の場合は，複数のポート番号を使用している．この例を図 5.5に示す．

図 5.5は，複数のポート番号に分散した flooding 攻撃を示している．この攻撃は 2 番ポート，5 番ポート，8 番ポートという具合いに，3 の倍数から 1 を引いたポート番号に攻撃データが分散している．

一般的に TCP を用いた攻撃としては，以下に示す 3 つの攻撃が有名であるが，AGURI を用いて，この攻撃のソース IP アドレスを調べることにより，攻撃手法を限定することができる．

```
%!AGURI-1.0
%%StartTime: Sun Oct 14 14:00:00 2001 (2001/10/14 14:00:00)
%%EndTime:   Sun Oct 14 15:00:00 2001 (2001/10/14 15:00:00)
%AvgRate: 24.30Mbps
[ip:proto:dstport] 10933438650 (100.00%)
0/0:0:0 50394643 (0.46%/100.00%)
4:6:0/0 123970078 (1.13%/96.16%)
   4:6:0/3      136730580 (1.25%/95.03%)
          4:6:0/10      110321675 (1.01%/51.22%)
            4:6:0/12    180612063 (1.65%/11.77%)
                4:6:2   220337940 (2.02%)
                4:6:5   220259760 (2.01%)
                4:6:8   224630700 (2.05%)
                4:6:11  220901820 (2.02%)
                           :
                           :
                4:6:104 229349040 (2.10%)
                4:6:107 220964460 (2.02%)
                4:6:110 221768098 (2.03%)
                4:6:119 213498789 (1.95%)
```

図 5.5: TCP を用いた攻撃のポート番号の分散

- TCP syn flooding 攻撃

  大量の TCP の SYN パケットを攻撃先ホストに送信することにより，当該ホスト内に TCP establish のプロセスを発生させる．これにより，当該ホストの計算機資源を占有しサービス不能状態に陥れる．

- ポートスキャン

  複数のポート番号に対して TCP データを送信し，返答を待つことにより，攻撃先ホストの空いているポートを推測する攻撃である．

- アプリケーションを対象とした flooding 攻撃

  ポートスキャンの後に行われることの多い攻撃．攻撃先ホストの空いているポート番号から，当該ホスト上が提供しているサービスを推測し，そのサービスを提供しているソフトウェアのバッファオーバーフローを狙う攻撃である．

このように flooding 攻撃のデータから，TCP を用いた攻撃であると分かった場合でも，複数の攻撃手法の可能性が考えられる．しかし，複数のポートにデータを送信している点から，アプリケーションを対象とした攻撃である可能性を低くすることがで

る．更に，同じデータのソース IP アドレス情報を見ることによって，ポートスキャンを狙ったものか，SYN flooding 攻撃か判断することができる．

送信した TCP データの返答を期待するポートスキャンの場合，ソース IP アドレスを詐称する可能性が低いからである．

このように，収集したデータの解析により，攻撃に使用するプロトコル毎それぞれの攻撃手法が固定化していることがわかった．また，同一の攻撃対象に対して，TCP と UDP の両者を用いるような，複数の手法を組み合わせた攻撃は検出されなかった．

5.2.4 時間的推移

攻撃を時系列に並べ解析することによって，flooding 攻撃は 2 つの大きな特徴を持っていることが分かった．この 2 つの傾向を以下に示す．

- 同一の IP アドレスに対して攻撃が繰り返される
- 同種の IP アドレスに対して攻撃対象が移行する

この 2 つの傾向を両方持っているトラフィック情報を例として，以下の図 5.6に示す．

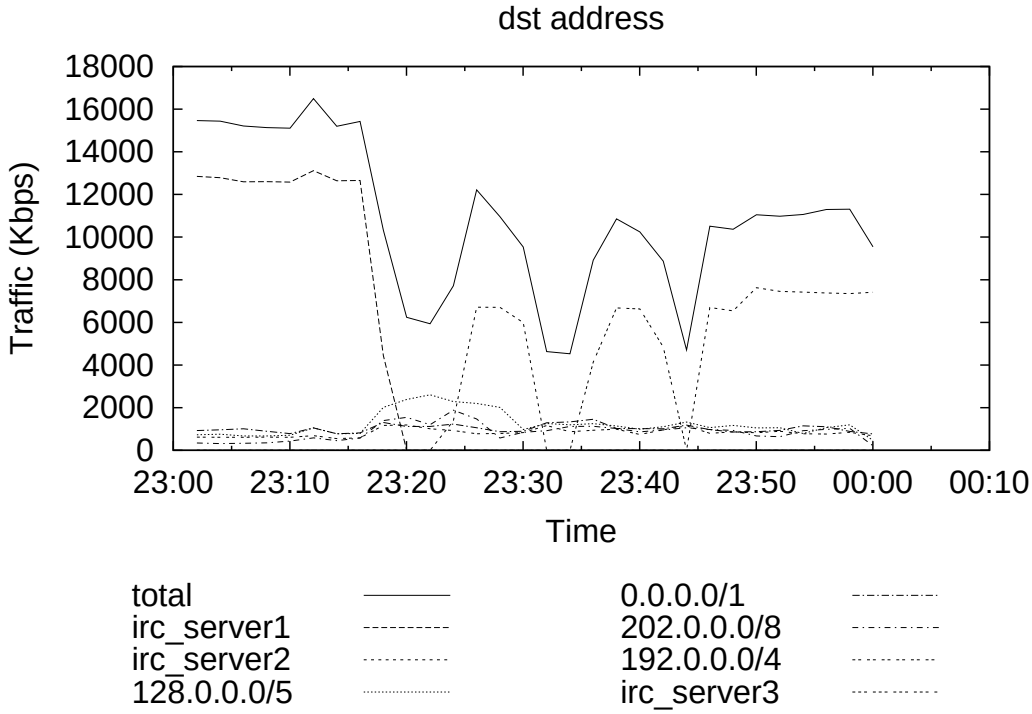

図 5.6: flooding 攻撃対象の時間経過による推移

図 5.6は，4 回の flooding 攻撃を表している．

1 回目の攻撃は，irc server 1 を攻撃先とした攻撃であり，23 時以前から継続しており 23 時 20 分頃収束する．

2 回目の攻撃は，irc server 2 を攻撃先とした攻撃であり，23 時 22 分頃から始まり 23 時 32 分頃収束する．

3 回目の攻撃は，irc server 3 を攻撃先とした攻撃であり，23 時 34 分頃から始まり 23 時 44 分頃収束する．

4 回目の攻撃は，再び irc server 2 を攻撃先とした攻撃であり，23 時 34 分頃から始まり 24 時移行も継続している．

この例に示したように，flooding 攻撃は，ある一定間隔をあけた後，再び同一ホストを攻撃する．また，flooding 攻撃は，その攻撃対象を同種のサービスを提供している別のホストに移行させる．

この 2 つの特徴は，前者については同例が 49 件検出されている．後者の特徴に関しては 17 件検出されている．この 2 つの特徴は，irc server だけでなく，ルータインターフェースを攻撃対象とした場合にも検出された．

更に，攻撃が繰り返されたり，同種のホストへ対象が移行する場合は必ず，攻撃に使用されるプロトコルやポートは引き続き同様のものが用いられていた．

5.3 分析結果と提言

本研究による flooding 攻撃解析の結果，以下に示すの 3 つの特徴が明らかになった．

特徴 1. 攻撃先 IP アドレスは限定される

特徴 2. 攻撃に使われる手法を組み合わせた攻撃は行われない，また，攻撃途中での手法変更も行われない

特徴 3. 攻撃はある間隔をおいて繰り返えされる，または，同種の対象に移行する

本研究では，これらの 3 つの特徴から，flooding 攻撃対策には提供するサービスを指標としたホストのグループ化が有効であると提案する．

” 特徴 3” から，ひとたび flooding 攻撃を受けた場合，同種のサービスを提供しているホストへの攻撃が移行する可能性を予測することができる．また，flooding 攻撃が収束した場合も，攻撃が再開する可能性を予測することができる．つまり，flooding 攻撃を引き金として，グループ化された同種のサービスホスト全てに対してフィルタの記述を行うことができる．また，flooding 攻撃の収束によって，フィルタを全て解除するのではなく，ある一定期間の攻撃に備えることができる．

それに加え，” 特徴 2” から，次の攻撃の攻撃手法は，現在の flooding 攻撃に使用されている手法と同様であると予測することができるため，より精度の高いフィルタをあらかじめ記述することができる．

更に，運用しているネットワーク内に存在する全てのサービスをあらかじめ把握しグループ化しておくことはできないが，” 特徴 1” により，攻撃を受けたホストの提供しているサービスのみをグループ化することで，それほど管理コストを必要とせずに flooding 攻撃対策を行うことができる．

第6章　まとめ

インターネットが日常生活のための情報基盤となった現在，flooding 攻撃への対策を行うことは非常に重要である．なぜならば，flooding 攻撃を受けてネットワーク資源が占有されてしまった場合，そのネットワークを経由する全ての通信が障害を受けてしまい，非常に広範囲の被害を被ってしまうからである．

しかし，現状では flooding 攻撃の情報を詳細に取ることはできない，そのため，flooding 攻撃の実体は不明点が多い．このため，現状では flooding 攻撃に対する効果的な対策が存在しない．これらの現状を第 2章で述べた．

その上で，本論文では，第 3章において，flooding 攻撃の情報を詳細に収集する機構である AGURI の設計，実装に関して述べ，第 4章，第 5章において，収集した flooding 攻撃の情報の分析, flooding 攻撃に対する効果的な対策の提言を行った．

これらのデータ解析の結果，flooding 攻撃は 1) 攻撃先のホスト，2) 用いられる手段，3) 時間的推移において特徴を持っていることがわかった．これらの結果によって，ネットワーク運用者は警戒するポイントを大きく絞り込め，実際にある攻撃を受け場合に次の攻撃を予測することができる．また，DoS 攻撃対策 software の作成に関しても，ルールセットのグループ化といった，現状の flooding 攻撃の現状に即した設計を可能にした．

本研究の成果により，社会基盤となるインターネットを flooding 攻撃に対して強固にすることができる．通信経路上において flooding 攻撃によるネットワーク資源占有を効果的に防止し，もし攻撃による被害が出たとしても，迅速な対応と，予測による次の攻撃の対策によって，ユーザに対しより快適なネットワーク環境を提供することができる．

謝辞

本研究を進めるにあたり，御指導を頂きました，慶應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，同学部の中村修博士，SonyCSL の長健二朗博士に深い感謝の意を表します．

重近範行氏，杉浦一徳氏，小川晃通氏をはじめとする慶應義塾大学政策メディア研究科博士課程の方々，並びに同大学環境情報学部の徳田・村井・中村・楠本・南研究室の諸氏には，本論文執筆の上で様々な議論をして頂き，また耐えざる励ましとご援助を頂きました．ここに，深い感謝の意を表します．

最後に，2 年間に渡って, 辛抱強く私を信じて見守って下さった母に感謝いたします．

以上を持って謝辞と致します．

平成 15 年 1 月 14 日　　研究室にて

参考文献

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

[2] G.Kessler,S.Shepard "A Primer On Internet and TCP/IP Tools" chapter 2.3, RFC1739, December 1994

[3] G.Kessler,S.Shepard "A Primer On Internet and TCP/IP Tools" chapter 2.2, RFC1739, December 1994

[4] Oetiker,Tobias, "MRTG - The Multi Router Traffic Grapher", USENIX System Administration Conference, December 1998

[5] S.McCanne,V.Jacobson, "The BSD Packet Filter: A New Architecture for User-Level Packet Capture", '93 USENIX Conference, January 1993

[6] J.Postel "Transmission Control Protocol", RFC 793, September 1981

[7] Floyd, S., and Jacobson, V. "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol.1 No.4 pp.397-413, August 1993

[8] John Nagle "On packet switches with infinite storage" IEEE Tromp. Commun. Vol.COM-35 No.4, April 1987

[9] Floyd, S. and Jacobson, V. "Link-sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking Vol. 3 No. 4 pp.365-386, August 1995

[10] C. Perkins, "IP Mobility Support." Internet Engineering Task Force, Request for Comments 2002, October 1996.

[11] A.C.Snoeren,C.Partridge,L.A.Sanchez and C.E.Jones "Hash-Based IP Traceback", ACM SIGCOMM 2001, 2001

[12] S.Savage,D.Wetherall,A.Karlin and T.Anderson "Practical Network Support for IP Traceback", ACM SIGCOMM 2000,pp.295-306, 2000

[13] S.M.Bellovin "ICMP Traceback Message", Internet Draft, May 2000

[14] "横河電気株式会社", http://www.yokogawa.co.jp

[15] Internet Engineering Task Force, www.ietf.org

[16] J. Hawkinson and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", RFC 1930, March 1996

[17] S.McCanne,V.Jacobson, "The BSD Packet Filter: A New Architecture for User-Level Packet Capture", '93 USENIX Conference, January 1993

[18] J.Case,M.Fedor,M.Schoffstall,J.Davin "A Simple Network Management Protocol (SNMP)", RFC1157, May 1990

[19] K.McCloghrie "Management Information Base for Network Management of TCP/IP-based internets : MIB-II", RFC1213 March 1991

[20] T.Oetiker, "Round Robin Database Tool", http://www.caida.org/Tools/RRDtool/

[21] J.Allen, "Cricket Monitoring System", http://www.munitions.com/ jra/cricket/

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

[23] "Cooperative Association for Internet Data Analysis", http://www.caida.org

[24] "netflow documentation", http://www.cisco.com/univercd/td/doc/product/rtrmgmt/nfc/nfc_3_0/nfc_ug/nfcover.htm

[25] "Cisco IOS Software Configuration", http://www.cisco.com/univercd/cc/td/doc/product/software/index.htm

[26] MORRISON, D. R. "PATRICIA—practical algorithm to retrieve information coded in alphanumeric", ACM 15,514 - 534, 1968

[27] Kenjiro Cho,Ryo Kaizaki,Akira Kato, "AGURI: An Aggregation-Based Traffic Profiler", QofIS2001, September 2001

[28] Thomas Williams and Colin Kelley, "Gnuplot", http://www.gnuplot.info

[29] NSPIXP "NSPIXP(Network Service Provider Internet eXchange Point) WWW page", http://nspixp.sfc.wide.ad.jp

[30] Network Startup Resource Center "EDI Meets the Interne", RFC 2901, 2000

[31] Gary Nicholl, "SONET Technology Primer", http://grouper.ieee.org/groups/802/3/

[32] Intel Corporation " Express 100BASE-TX Switching Hub ", http://www.intel.com

[33] Ryo Kaizaki, Kenjiro Cho,Osamu Nakamura, "Detection of Denial of Service attacks using AGURI", ICT2002, June 2002

[34] J.Oikarinen and D.Reed "Internet Relay Chat Protocol", RFC 1459, 1993

[35] P. Ferguson and D. Senie "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2827, 2000