コンピュータ理工学部特別研究 II A・B

LAN内における定常トラフィック分析手法に関する研究

A Research on Regular Traffic Analysis in a Local Area Network

秋山特研

菅原 淳

平成 24 年 4 月 18 日

概要

近年，特定組織を標的とした標的型攻撃の被害が多発している．これらの攻撃は，LAN 内で感染を拡大するという特徴を有している．このような背景から，これまで行われてきたコアネットワークやインターネット接続部分の異常検知だけでなく，LAN 内において異常検知を行う必要がある．アノマリ型の異常検知では，正常時に一定の値を取る定常値の取得が必要である．既存研究には，トラフィックのパラメータをガウス分布により定義できた場合カルマンフィルタによる状態予測による異常検知が可能であることを示した研究がある．また，ハースト数と呼ばれる時系列の長期記憶性のバイアスとなる値を用いて定常トラフィック定義を行い異常検知に用いる研究もある．本研究では，これら手法に使用されているパラメータを本学の学内 LAN にて調査し，既存手法の学内 LAN における適用可能性について調査した．その結果，ガウス分布を用いた手法は必ずしも適用できないことを確認した．また，ハースト数の調査から学内 LAN のトラフィックはプロトコルによっては長期記憶性を有しており，ハースト数を用いた定常トラフィック定義が可能であることを確認した．さらに，トラフィックパラメータの度数分布図において想定される分布と異なった場合，その原因と考えられるパケットを抽出することで，詳細分析分析のコスト削減が可能であることを確認した．

Abstract

In recent years, the ”targeted attacks” are increasing. The attacks are created and customized to penetrate a specific target or organization. The infections of the attacks mainly spread in a Local Area Network(LAN). Since the spread of the infections in a LAN can not be detected by the anomaly detection at the core network and internet gateway. We need to detect suspicious behaviors in a LAN. Anomaly detection techniques require steady-state values of normal behavior. Previous works proposes various anomaly detection techniques. One of them uses Kalman filter which depends on Gaussian distribution traffic model, and another one uses Hurst exponent of normal behavior. In this research, we investigated the distribution and the Hurst exponent of the traffic parameter in our university LAN. As a result of the investigation, we confirmed the assumption of the distribution in the existing detection techniques is not always applicable to the LAN. On the other hand, Hurst exponent indicates a Long-range dependency. It shows the possibility to use the Hurst exponent for defining the steady-state. We also confirmed that the time of the detail packet anaysis can be reduced by extracting the packets with anomalous distribution.

目 次

1 はじめに

現在，コンピュータネットワークは重要な社会基盤となっている．ネットワーク管理者は未知の攻撃による異常なトラフィックを検出し，即時に対応することが求められている．これまで，ネットワークにおける異常検知では様々な手法が提案されてきた．近年，標的型攻撃やゼロデイ攻撃では外部接続部分での監視をすり抜け，LAN 内で感染被害を拡大する事例が多い．そのため，LAN 内での定常トラフィックを把握し，異常なトラフィックを検知して感染を食い止める手法が必要となる．そこで本研究では既存の異常検知手法を LAN 内のトラフィック分析に適用した場合の課題を調査する．本研究では，ネットワークにおいて日常的に観測されるパケット数，フロー数などのパラメータを用いて定常トラフィックの分析を行う．既存の異常検知手法の中には，パケット数やフロー数などのパラメータの度数分布が正常時に一定の分布になるという前提で異常を検知する手法 [1] や時系列の特徴を用い，ハースト数と呼ばれる値を用いて定常トラフィックを定義し異常検知に用いる研究 [2] が存在する．本研究では，これら手法を LAN に適用した場合，想定されている分布や時系列の性質が妥当かどうか，妥当でない場合何が原因であるかを明らかにする．トラフィックのパラメータの度数分布とパラメータの時系列に着目し，トラフィック分析を行う．度数分布を用いた分析ではトラフィックのパラメータの度数分布を混合ガウス分布で近似し，混合分布のパラメータにより想定分布との関連を調査する．さらに，想定分布との相違点を分析した上で，相違を生じさせるトラフィックを抽出可能なモニタリングシステムを実装し，相違の原因を調査する．時系列を用いた分析では R/S 解析を用いたハースト数推定による定常トラフィックの定義を目的に，R/S 解析を行う際のパラメータとハースト数の推定結果の関係や時系列での特徴とハースト数の関係について調査を行う．

以下 2 章では，ネットワーク異常検知に関わる既存技術について述べ，その特徴や問題点などを提示する．またそれらの問題点に関する既存研究についても述べる．3 章では，EM アルゴリズムや R/S 解析の概要をと，初期パラメータの設定値による結果の差異について述べる．4 章では，モニタリング実験の概要や実施環境，実験結果について述べる．5 章では 4 章で得られた結果に関して考察を行う．最後に 6 章で本研究のまとめと今後の課題を述べる．

2 ネットワーク異常検知に関わる既存技術

ネットワークやコンピュータに異常をもたらす原因は，ポートスキャン，SYN flood 攻撃，DDoS 攻撃，P2P アプリケーションなど多く存在する．また，コンピュータウイルスやワームなどはネットワークを介して他のコンピュータに感染を拡大する．これら異常をもたらす原因を検出するために様々な手法が提案されてきた．本章では，それらネットワーク異常検知に関わる既存技術や既存研究について述べる．

2.1 異常検知手法

ネットワークにおける異常検知手法は，大きく以下の二種類に分割することができる．

シグネチャ型

シグネチャ型では，あらかじめ異常とされるパケットや状態のデータベースを作成しておき，それに合致するパケットや状態が観測されると異常として検知する異常検知手法であり，シグネチャ型 IDS の Snort[3] や DPI(Deep Packet Inspection) 技術を用いた OpenDPI[4] などがある．シグネチャを用い

るため既知の異常が発生した場合確実に検出することができるが，未知の異常が発生した場合，異常を検知できない可能性がある．また，Snort は広く利用されているが，ルール記述のみでプロトコルの状態遷移を追うことができないという欠点が存在する．OpenDPI ではフローやプロトコルの状態遷移を追跡することが可能であるが，これも同様に未知の異常を検知することはできない．また，P2P アプリケーションのトラフィックを検出する場合，P2P アプリケーションは使用するポート番号が決まっていなかったり，プロトコルの仕様も公開されていない場合が多いことから，シグネチャを作成することが困難な場合があり，シグネチャ型による異常検知が P2P アプリケーションの検出に適しているとは言えない．

Snort では以下のように，プロトコルや IP アドレス，ポート番号などの情報に基づいたルール表記を用いてシグネチャが作成されている．現在無償で使用できるルールファイルには約 18,000 個のシグネチャが含まれている．

Snort におけるルール表記の例

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-access";  uricontent:"/img"; nocase;
```

シグネチャ以外の Snort の重要な要素として，プリプロセッサがある．プリプロセッサは IP パケットが分割されている場合に正しく検査を行うために，分割されたパケットの再組み立て (reassemble) を行うプリプロセッサ (frag3) や，HTTP などのアプリケーション層のプロトコルの正規化などを行うプリプロセッサ (http-inspect) など様々なものが存在する．他に，ポートスキャンなどの異常な振る舞いを検知する機能を持ったプリプロセッサなども存在し，プリプロセッサを使用することでより詳細な分析による異常検知が可能であるが，プリプロセッサの開発にはプログラミングを要するためルール記述のように容易に作成することはできない．

アノマリ型

アノマリ型では，あらかじめ正常値による学習を行い学習したデータから逸脱したデータが観測されると異常として検知する手法と，異常値を含むデータによる学習で異常を検知する二種類の手法が存在する．シグネチャに依存していないため，未知の異常であっても検知できる可能性があることから，近年このアノマリ型の異常検知手法が注目されている．アノマリ型の異常検知手法では，何らかの方法で正常時のパラメータを学習し，そこから逸脱した際に異常として検出する手法が考えられるが，その際設定するしきい値が，検知率に変化をもたらす．最適なしきい値を設定することがアノマリ型の異常検知手法において重要となる．

2.2 ネットワークモニタリング手法

ネットワーク異常検知を行うためには，ネットワークモニタリング手法の選択が必要となる．必要なネットワークの情報やネットワークの規模に応じたモニタリング手法を選択しなければならない．以下に既存のネットワークモニタリング手法について特徴などを述べる．

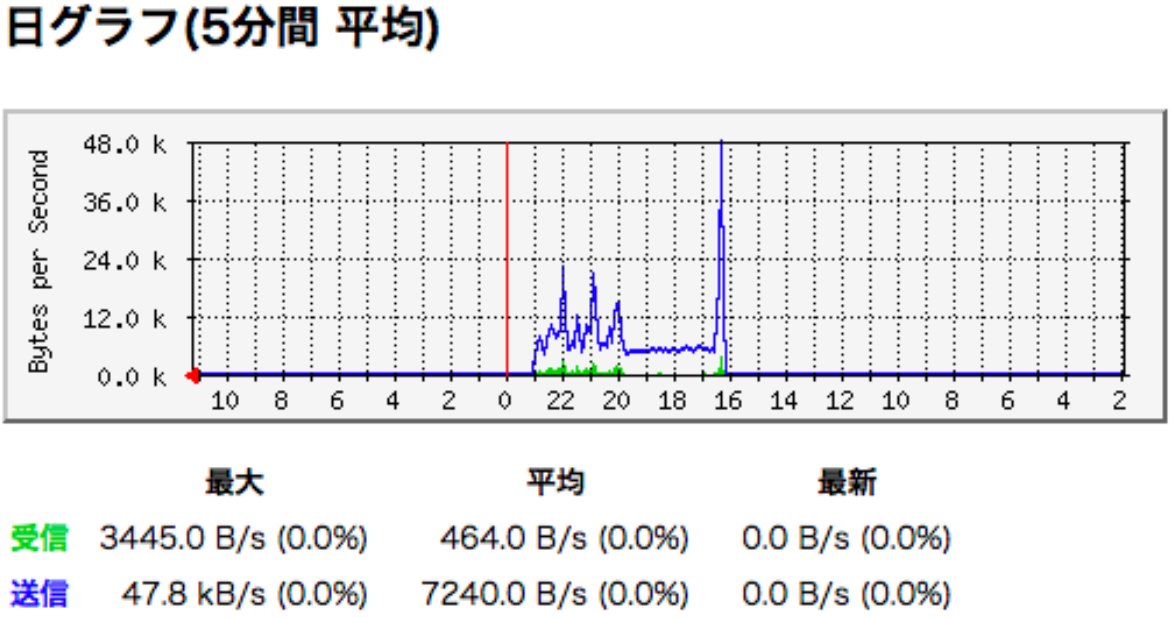
ミラーリング

ミラーリングとは，ネットワーク機器に設定することで指定されたポートを流れるパケットをコピーし，指定した別のポートへ転送する手法である．ネットワーク上のパケットをそのまま受け取り，分析することができるため，最も得られる情報量の多い手法である．しかし，ネットワーク上の全パケットを一つのポートで受け取るため，負荷が高くなりパケットを観測しきれない場合がある．これは，パケットを観測するマシンの性能やその回線などに依存する．ミラーリングにより観測したパケットを WireShark[5] などのパケットキャプチャツールで分析することで，様々な情報を分析することができる．その他 tcpdump[6] によりダンプされたファイルに対応した様々なツールが存在する．

SNMP

SNMP とは，ルータ等のネットワーク機器でトラフィック情報を計測しその情報を取得したり，ネットワーク機器の管理を行うことができるプロトコルである．管理する側をマネージャ，管理される側をエージェントという．エージェントは MIB(Management Information Base) と呼ばれる管理情報をマネージャの要求により読み書きすることができる．SNMP はミラーリングに比べて非常に負荷が少ないが，取得できる情報が限られており，フローを考慮した情報を得ることができない．SNMP を利用したツールとしては MRTG[7] が有名である．MRTG では図 1 のようにトラフィックデータを可視化することで，管理者が自ら異常を発見することができる．

図 1: MRTG によるトラフィックデータ可視化の例



NetFlow, sFlow

NetFlow とは Cisco Systems 社によって開発されたトラフィック情報取得システムであり，ミラーリングに比べルータの負荷を抑えながらフロー単位での分析が可能となっており，SNMP よりも多くの情報を得ることができる．sFlow ではトラフィックに対してサンプリングを行うことができるため，大規模なネットワークを監視する際に用いる必要がある．

2.3 既存研究

正常値による学習に基づくアノマリ型異常検知手法では，正常時の定常トラフィックのパラメータの分布が一定になることを利用している．[1][8]．文献 [1] ではカルマンフィルタを用いたアノマリ型異常検知手法を提案している．単位時間毎のフロー数の変動はガウス分布を形成すると仮定し，カルマンフィルタを用いて次状態推定を行うことで，急激なフロー数増加などを異常として検知している．本研究では LAN 内のトラフィックを対象としているが，既存の手法が，コアネットワークを対象としていた場合，LAN 内では適用できない可能性がある．また，文献 [8] では，ペイロードに含まれる ASCII 文字の分布をサービスごとに調査し，観測されたペイロードとのマハラノビス距離を計算することで，異常を検知している．

分布以外を用いる手法では，R/S 解析によって推定されるハースト数を用いた手法が存在する [2]．ハースト数をプロトコルごとに推定し，ネットワークの状態変化によりハースト数が変化することを用い異常検知に使用される．

3 定常トラフィック分析手法

本研究では，文献 [1] で述べられている，トラフィックのパラメータの分布はガウス分布を形成するという仮定のもと分析を行い，トラフィックのパラメータの度数分布を混合ガウス分布で近似した．そして，近似の結果得られた分布における個々のガウス分布の平均値，分散値，さらにガウス分布の混合比などのパラメータを用いて，分布の特徴を分析する．混合ガウス分布を求める方法としては，EM アルゴリズム [9] を用いた．また，文献 [2] で述べられているハースト数を用いた異常検知手法の適用可能性を調査するために，R/S 解析を用いたハースト数推定を行った．

3.1 ガウス分布・混合ガウス分布

ガウス分布とは，平均値を中央値に持ち平均値から離れていくに従いその確率が低くなるという特性を持った確率分布である．平均値を μ，分散値 σ^2 とした場合，その確率密度関数は以下の式 (1) で表すことができる．

$$f(x) = \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)}{2\sigma^2}\right) \tag{1}$$

本稿では平均値 μ，分散値 σ^2 であるガウス分布の確率密度関数を $\mathcal{N}(\mu, \sigma^2)$ と表す．一方，混合ガウス分布とは，複数のガウス分布を合成して生成した確率分布を指し，任意の確率分布を近似する際に使用される．ある混合ガウス分布において混合されたガウス分布の数を K とすると，混合ガウス分布の確率密度関数は式 (2) で表せる．ここで，γ_k は各ガウス分布の混合比であり $\sum_{k=1}^{K} \gamma_k = 1$ とする．

$$f(x) = \sum_{k=1}^{K} \gamma_k \mathcal{N}(x|\mu_k, \sigma_k^2) = \sum_{k=1}^{K} \gamma_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)}{2\sigma^2}\right) \tag{2}$$

3.2 EM アルゴリズム

EM アルゴリズムとは最尤法に基づき確率モデルのパラメータを推定する手法の一つである．EM アルゴリズムは E ステップと M ステップの 2 つで構成されており，後述の対数尤度が収束するまで各ステップを繰り返す．また，初期パラメータとしてはじめに，推定を行ないたい混合ガウス分布の混合数 K，K 個のガウス分布の平均値および分散値，さらに混合ガウス分布における混合比を設定する必要がある．はじめに行われる E ステップでは，観測された値に対して初回は初期値として入力されたパラメータ，2 回目以降は現在推定されているパラメータに基づいて観測した個々のデータが k 番目のガウス分布によって生成された確率を求める．M ステップでは E ステップで得られた確率分布から計算した尤度を最大化する各ガウス分布のパラメータを求め，次の E ステップのパラメータとして用いるか，推定結果として出力する．推定結果が収束したと判定するには尤度関数を用いる．ここである観測において N 個の観測値からなる観測値集合 $X = \{x_1, x_2, \cdots, x_i, \cdots, x_N\}(i = 1, \cdots, N)$ が得られたとする．観測値 x_i が現在推定されているパラメータの混合分布によって観測される確率は式 (2) より

$$f(x_i) = \sum_{k=1}^{K} \gamma_k \mathcal{N}(x_i|\mu_k, \sigma_k^2) \tag{3}$$

となる．実際に値 x_i は観測されたので確率は 1 であるが，x_i を観測するという事前確率（尤度）ができるだけ大きくなるようパラメータを調整すれば，推定する混合ガウス分布が観測値 x の確率密度分布に近づくことになる．尤度の値は一般的に小さな値となるため対数尤度が用いられる．すべての観測値が観測される尤度は

$$L(X|\mu, \sigma^2, \gamma) = \prod_{i=1}^{N} \sum_{k=1}^{K} \gamma_k \mathcal{N}_k(x_i, \mu_k, \sigma_k^2) \tag{4}$$

となり，両辺対数をとると

$$\log L(X|\mu, \sigma^2, \gamma) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \gamma_k \mathcal{N}_k(x_i|\mu_k, \sigma_k^2) \right) \tag{5}$$

となる．式 (5) の尤度関数で計算される対数尤度の各ステップごと差が一定値以下となった場合に収束と判定することができる．このように，尤度関数を最大化しパラメータを推定する手法が最尤法と呼ばれている．本研究では特記しない限り混合数は $K = 7$ に固定して調査を実施した．

EM アルゴリズムによる混合ガウス分布近似を行う場合の E ステップ，M ステップは以下のようになる．ここでは，K 混合ガウス分布で近似し，N 個の観測値集合を $X = \{x_1, x_2, \cdots, x_i, \cdots, x_N\}(i = 1, \cdots, N)$，平均値，分散値，混合比の初期値（M ステップ実行後は推定値）をそれぞれ $\mu_k, \sigma_k^2, \gamma_k\ (k = 1, ..., K)$ とする．

E ステップ

まず混合ガウス分布による各観測値の確率密度 $f(x_i)$ を算出する．

$$f(x_i) = \sum_{k=1}^{K} \gamma_k \mathcal{N}(x_i|\mu_k, \sigma_k^2) \tag{6}$$

次に，各ガウス分布の負担率を求めるために以下の式で各観測値についての重み $w_k(x_i)$ を得る．

$$w_k(x_i) = \frac{\gamma_k \mathcal{N}(x_i|\mu_k, \sigma_k^2)}{f(x_i)} \tag{7}$$

M ステップ

$w_k(x_i)$ は $\sum_{i=1}^{N}\sum_{k=1}^{K} w_k(x_i) = 1$ を満たし，$\log L(X|\mu, \sigma^2, \gamma)$ の値を変えることなく $w_k(x_i)$ を導入し，Jensen の不等式を適用すると以下のようになる．

$$\begin{aligned}
\log L(X|\mu, \sigma^2, \gamma) &= \sum_{i=1}^{N} \log \sum_{k=1}^{K} w_k(x_i) \frac{\gamma_k \mathcal{N}_k(x_i|\mu_k, \sigma_k^2)}{w_k(x_i)} \\
&\geq \sum_{i=1}^{N}\sum_{k=1}^{K} w_k(x_i) \log \frac{\gamma_k \mathcal{N}_k(x_i|\mu_k, \sigma_k^2)}{w_k(x_i)} \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} w_k(x_i)\{\log \gamma_k + \log \mathcal{N}_k(x_i|\mu_k, \sigma_k^2)\} - \sum_{i=1}^{N}\sum_{k=1}^{K} w_k(x_i)\{\log w_k(x_i)\} \quad (8)
\end{aligned}$$

ここで，式 (8) の混合ガウス分布のパラメータが含まれない部分を除いた式を

$$\mathcal{F}(X|\mu, \sigma^2, \gamma) = \sum_{i=1}^{N}\sum_{k=1}^{K} w_k(x_i)\{\log \gamma_k + \log \mathcal{N}_k(x_i|\mu_k, \sigma_k^2)\} \quad (9)$$

と置き $\mathcal{F}$ を大きくするために，$\mu_1, ..., \mu_K, \sigma_1^2, ..., \sigma_K^2, \gamma_1, ..., \gamma_K$ のそれぞれについて $\mathcal{F}$ の偏微分を行うことで，以下のような尤度を最大化するパラメータの更新式が得られる．偏微分によるこれら式の導出方法は文献 [10] の pp. 3-7 に掲載されている．

E ステップで求まった $w_k(x_i)$ を各 k ごとにすべての観測値について総和をとったものを S_k とする．

$$S_k = \sum_{i=1}^{N} w_k(x_i) \quad (10)$$

S_k を用いて平均値，分散値，混合比を以下のように更新することができる．

$$\mu_k = \frac{\sum_{i=1}^{N} w_k(x_i) x_i}{S_k} \quad (11)$$

$$\sigma_k^2 = \frac{\sum_{i=1}^{N} w_k(x_i)(x_i - \mu_k)^2}{S_k} \quad (12)$$

$$\gamma_k = \frac{S_k}{N} \quad (13)$$

これら E ステップと M ステップを繰り返し行い，式 (5) に示した対数尤度の各ステップごとの差が一定値以下になった時に推定結果を出力する．

3.2.1 混合ガウス分布と単一ガウス分布の比較

ここで，ガウス分布を混合ガウス分布で近似する例を示す．まず，平均値 $\mu = 50$，分散値 $\sigma^2 = 16$(標準偏差 $\sigma = 4$) のガウス分布をとる 30,000 個のデータを生成し，そのデータに対して EM アルゴリズムによる

表 1: EM アルゴリズム初期値

γ	0.2000	0.2000	0.2000	0.1000	0.1000	0.1000	0.1000
μ	0.000	10.00	20.00	30.00	40.00	50.00	60.00
σ^2	500.0	500.0	500.0	500.0	500.0	500.0	500.0

表 2: EM アルゴリズムによる混合ガウス分布近似結果

γ	0.04562	0.1024	0.2017	0.1636	0.2189	0.2415	0.02591
μ	48.06	48.56	48.98	49.35	49.70	50.05	54.04
σ^2	8.719	14.57	16.71	16.05	15.52	15.10	8.777

7 混合ガウス分布近似を行った．初期値は表 1 に示した値を設定した．その結果，推定された混合ガウス分布の各パラメータは表 2 のようになり，図 2 のような推定結果が得られた．表 2 における γ は各ガウス分布の混合比を示している．図 2 より目視で確認する限りでは混合ガウス分布でうまく近似できているように見える．混合されるすべてのガウス分布の平均値，分散値が元の分布と一致すれば，近似した分布が単一のガウス分布に近いかどうかが容易に判定できるが，表 2 中の両端の分布の分散は必ずしも近い値となっていない．ここで，得られた 7 個のガウス分布の平均値 μ について γ による重みつき平均，重みつき分散から μ の重みつき標準偏差を計算すると，重み付き平均は 49.48 と元の分布の平均値に近く，その標準偏差も 0.9271 と十分に小さくなっている．同様に各ガウス分布の標準偏差 σ を計算すると，重み付き平均は 3.883，重み付き標準偏差は 0.2711 となる．よって，近似した分布とガウス分布の近さは，混合されるガウス分布の平均値，分散値について，重み付き標準偏差を求めればおおよその判定は可能であると考えられる．

3.2.2 EM アルゴリズムの初期値設定

EM アルゴリズムの推定結果は初期値に大きく依存する [11]．ここで，同様に生成したデータに対して，表 1 の初期値のうち平均値の 1 つを 1000 とし，入力される値の分布から大きく離れた値をとり，その分散値を 50.00 とした場合を考える．この初期値により EM アルゴリズムを適用すると，E ステップにおける条件付き確率の算出段階においてゼロ除算が発生してしまい，推定結果が得られない結果となった．このように，EM アルゴリズムでは初期値の平均値に対し入力するデータの分布に近い値を設定する必要がある．また，分散値を大きく設定することでゼロ除算の発生を抑制することも可能であるが，大きい分散値を設定した場合，図 3 のように複雑な分布を近似する際に正確な推定結果を得ることができない場合がある．

3.3 R/S 解析

近年，Ethernet トラフィックは自己相似性，長期記憶性をもつと報告されている [12]．自己相似とは，対象の分布などが時間などのスケールに対して不変であるようなデータの性質を言い，長期記憶性とは，相関

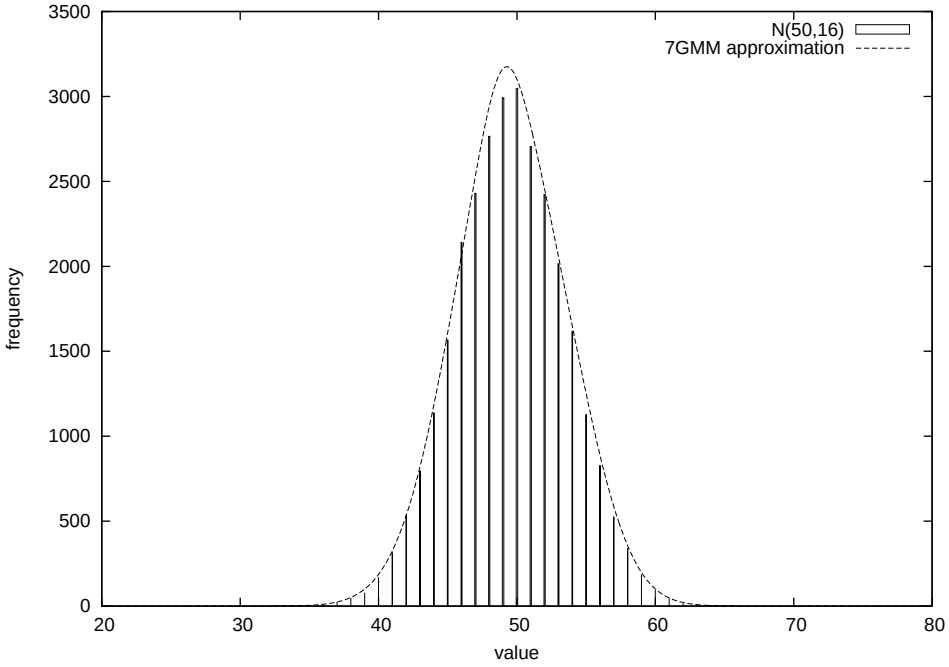


図 2: ガウス分布の混合ガウス分布近似

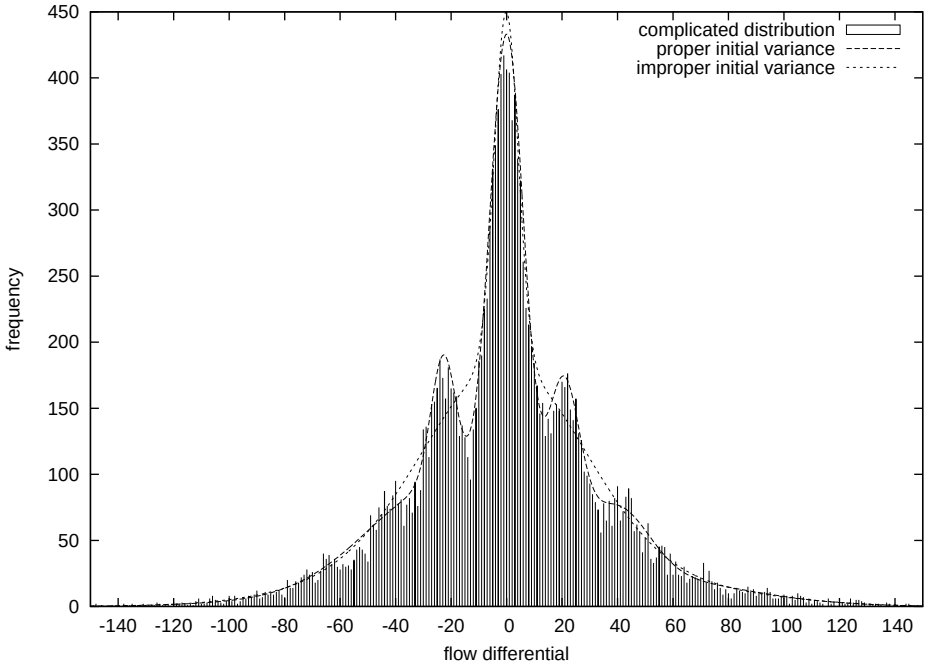


図 3: 初期分散値による推定結果の違い

表 3: ハースト数の値による時系列の性質

ハースト数 (H)	時系列の特徴
H = 0.5	将来に影響を及ぼさない (無記憶)
0≤H≤0.5	反持続的系列 上下の変動が激しい
0.5<H≤1	非整数ブラウン運動 長期記憶性あり

が長期にわたって持続するような時系列を言う．そこで，Ethernet の単位時間ごとのパケット数，フロー数などの自己相似性や長期記憶性の度合いとなるハースト数 (H) を R/S 解析によって推定することで，トラフィックの特徴として分析することができる．文献 [2] では，R/S 解析によるハースト数推定を行い，ネットワークの状態変化とハースト数の関連性について述べられている．ハースト数によるネットワークの定常状態を定義することが出来ればハースト数をネットワーク異常検知に利用できると考えられる．

R/S 解析とは観測値に内在する長期的な記憶効果を推定する機能を持っている．ハースト数とは，非整数ブラウン運動 (fractional Brownian Motion) のバイアスの尺度を示している．ハースト数は表 3 の 3 つのパターンを取り，それぞれ時系列の特徴を示す．

3.3.1 R/S 統計量の算出とハースト数の推定

R/S 解析の焦点は尺度変換調整レンジ (Rescaled adjusted range) の導出とハースト数の計算方法にある．ここで，尺度変換調整レンジを Q，ハースト数を H と置く．ある N 個の観測値からなる観測値系列 $X = \{x_1, x_2, \cdots, x_i, \cdots, x_N\}(i = 1, \cdots, N)$ を考える．ここで，$V(j) = \sum_{i=1}^{j} x_i$ とおくと X における平均値 $\overline{x}$ は

$$\overline{x} = \frac{1}{N} V(N) = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{14}$$

となり，分散値は

$$S = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\overline{x} - x_i)^2} \tag{15}$$

となる．

ここで，平均値 $\overline{x}$ からのずれの最大値と最小値の差である調整レンジ R を，

$$R = \max_{1 \le i \le N} (V(i) i\overline{x}) - \min_{1 \le i \le N} (V(i) i\overline{x}) \tag{16}$$

と定義する．様々な解析の結果，R と S の比が

$$Q = \frac{R}{S} \approx N^H \tag{17}$$

と表せることが知られている．

ここで，R/S 統計量と呼ばれる値を使用し実際にハースト数を推定する手順を示す．R/S 統計量を計算するために観測値系列 X において系列長 k，n 番目の観測値から始まる長さ k のサンプル系列 $X(n,k) = \{x_n, x_{n+1}, \cdots, x_{n+k-1}\}$ を考える．式 (16) と同様の調整レンジ R をサンプル系列ごとに求める．$V(n,j) = \sum_{i=n}^{n+j-1} x_i$ と置くと，$X(n,k)$ の平均は

$$\overline{x(n,k)} = V(n,k)/k \tag{18}$$

となるので，始点 n，系列長 k のサンプル系列の調整レンジ $R(n,k)$ および分散値 $S(n,k)$ は

$$R(n,k) = \max_{1\le j\le k}\left(V(n,j) - \frac{j}{k}(V(n,k))\right) - \min_{1\le j\le k}\left(V(n,j) - \frac{j}{k}(V(n,k))\right) \tag{19}$$

$$S(n,k) = \sqrt{\frac{1}{k}\sum_{j=n}^{n+k-1}(x_j - \overline{x(n,k)})^2} \tag{20}$$

これら 2 つの値を使用して尺度変換レンジ $Q(n,k)$ を以下のように定義する．

$$Q(n,k) = \frac{R(n,k)}{S(n,k)} \tag{21}$$

この値は R/S 統計量と呼ばれている．サンプル系列の始点は $1 \le n \le N-k+1$ の範囲で自由に選択できる．互いに重複しないサンプル系列は $m_c = \lfloor \frac{N}{k} \rfloor$ 個選べる．R/S 解析では $k_{\max}, k_{\min}$ を設定しサンプル系列の長さ k を $k_{\min}$ から $k_{\max}$ まで変化させ，各 k において m_c 個の R/S 統計量 $Q(1,k), Q(k,k), Q(2k,k), \cdots, Q(m_c,k)$ を算出する．ここで $Q(n,k)$ はべき則が成り立ち

$$Q(n,k) = ck^H \tag{22}$$

と表せる．c は定数である．両辺対数を取ると，

$$\log Q(n,k) = \log c + H \log k \tag{23}$$

となり，傾き H，切片 $\log(c)$ の 1 次方程式として表すことができる．最小二乗法により回帰直線を求めることでハースト数 H を推定することができる．

観測値系列 N の値が十分大きい場合十分な数のサンプル系列を得られるが，N が小さい場合，サンプル系列を重複して選択することでサンプル数を確保する．重複せずに解析する観測値の数を $\Delta n (1 \le \Delta n \le k)$ とすると $m_c = \lfloor \frac{N-k+\Delta n}{\Delta n} \rfloor > \lfloor \frac{N}{k} \rfloor$ となるため，より多くのサンプル系列を確保できる．ただし，観測系列の重複により本来の系列の特性と異なる特性が抽出されてしまう可能性がある．

3.3.2 設定パラメータと推定結果の関係

R/S 解析では $k_{\min} \le k \le k_{\max}$ において作成されるサンプルの数 m_c は一定である方が推定結果が安定するとされている．そこで，$k_{\max}$ の時に必要サンプル系列数 m_c を得られるように Δn を k によって変えず，取りうる最低値に固定してに使用する．$k_{\max}$ の時サンプル集合数を m_c 個得られるような Δn は以下のようにして求まる．

$$\Delta n = \frac{N - k_{\max}}{m_c - 1} \tag{24}$$

次に，サンプル系列の長さは本来 $k \to \infty$ の極限での挙動が長期記憶性を決定するため，$k_{\max}$ も大きくする必要があるが，計算時間を考慮し 100〜500 程度に設定する．必要サンプル系列数 m_c は標本の統計を決定するため，十分なデータ数がある場合可能な限り m_c を大きくすることが望ましい．こちらも計算時間を考慮して 100〜1000 程度に設定する．このように $k_{\max}, m_c$ は双方大きく設定することが望ましいが，計算時間とデータ数の関係上制約されることとなる．観測値の数 $N = 10000, k_{\max} = 500, m_c = 1000$ のパラメータで R/S 解析を行った場合，CPU の 1 コアのみを使用すると 12 分 39 秒かかることがわかった．準リアルタイムな異常検知手法を考えるにあたってこの計算時間は早いとは言えず今後さらに高速な計算手法を用いる必要がある．また，先に述べたように多くのサンプル系列を確保すると，重複する割合 $(k_{\max} - \Delta n)/k_{\max}$ が増加するため，この値がなるべく小さくなるような $k_{\max}, m_c$ を取ることが望ましい．Δn が $k_{\max}$ を取るときに，重複する割合はゼロとなる．

ここで，ガウス分布の乱数を 10000 個生成した観測値集合に対して $k_{\max}, m_c$ を変えた場合のハースト数の推定結果の変化を図 4，5 に，$k_{\max} = 500$ の時の R/S 統計量を図 6 に示す．図 6 における直線は最小二乗法による回帰直線である．ガウス分布は無記憶過程であるため，理論上のハースト数は 0.5 である． $k_{\max}$

図 4: $k_{\max}$ を変えた場合の変化

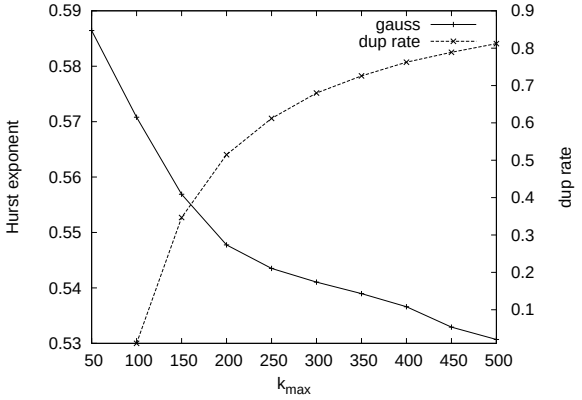
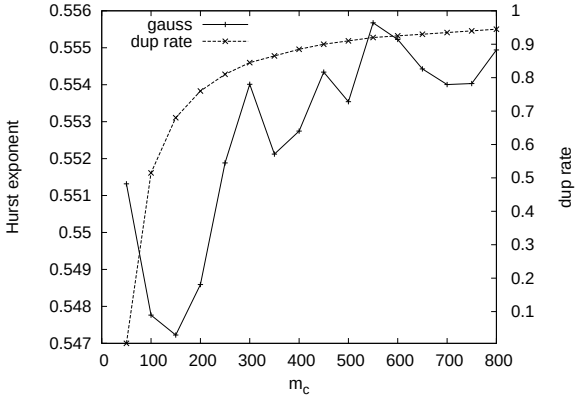


図 5: m_c を変えた場合の変化



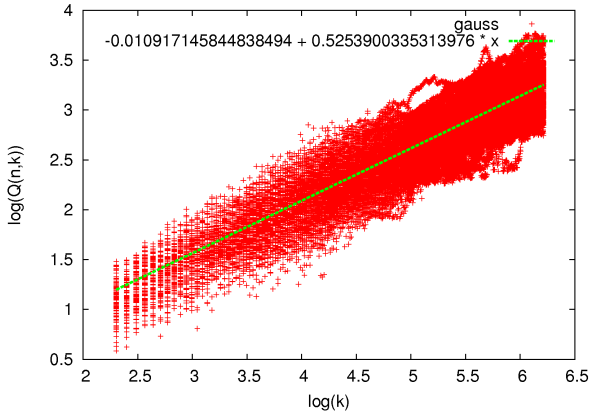
を変化させることにより，推定結果が徐々に 0.5 に近づいていることがわかる．ガウス分布は無記憶であるため，$k_{\max}$ の増加による Δn の減少が記憶に影響をもたらさない．つまり，Δn が減少することにより単にデータ数増えたと取ることができる．そのため，$k_{\max}$ を増加させることでより精度が高くなり 0.5 に近づいたと考えられる．

なお，ガウス分布乱数データは R[13] により以下のコマンドで生成することができる．

R によるガウス分布乱数生成

```
write.table(rnorm(10000, mean=0, sd=1),file="gauss.dat",
row.names = FALSE, col.names=FALSE)
```

図 6: ガウス分布の R/S 統計量 ($k_{\max} = 500$)



4 トラフィックモニタリング実験

本章では，トラフィックモニタリング実験を行うために必要な環境，実験手順，実験結果を示す．

4.1 モニタリング環境

本研究ではトラフィック情報の取得にミラーリングを採用し，2 つのモニタリング環境を用意した．1 つ目のモニタリング環境は研究室に設置されたスイッチ (Apresia 13100-48X) であり，このスイッチに対して以下のような設定を行うことで，ミラーリングの設定を行った．以降このモニタリング環境を研究室内 LAN と呼ぶ．2 つ目は，10 号館，10201,10202 教室に設置された，コンピュータ理工学部のネットワークを流れるパケットをミラーリングした環境である．以降この環境を学部 LAN と呼ぶ．研究室内 LAN と学部 LAN の大きな違いは，研究室内 LAN では TCP パケットは数台の PC から発せられるパケットしか観測することができない環境であることに対し，学部 LAN では講義時間帯に百数十人が接続を行うため，研究室内 LAN と比べ規模の大きいモニタリング環境となる．このような特徴のため，研究室内 LAN では主に UDP パケットを対象とした実験のみ行った．

Apresia のミラーリング設定

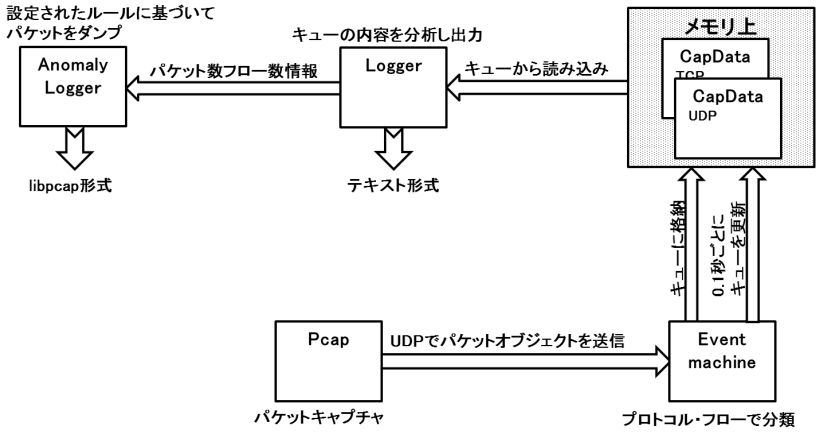
```
mirroring rx from port 1/49
mirroring rx to port 1/2
mirroring tx from port 1/49
mirroring tx to port 1/2
```

4.2 監視・分析プログラム

監視プログラムは Ruby を用いて開発した．ミラーリングによって転送されたパケットのキャプチャには Ruby/Pcap と呼ばれる libpcap[6] の Ruby インタフェースライブラリを使用した．またパケット数やフロー

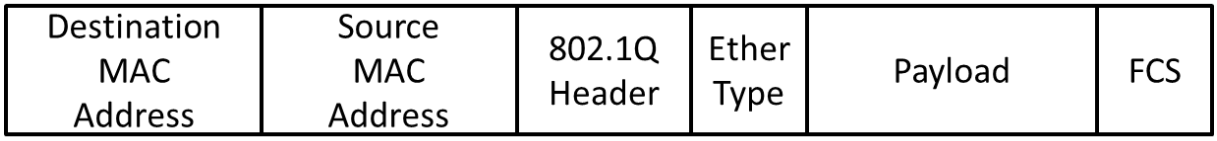
数の計測には Eventmachine ライブラリを使用した．このプログラムには定常トラフィックのパラメータとなる，単位時間ごとのパケット数，フロー数を出力する機能を実装した．本研究では，単位時間を 5 秒間とした．図 7 に監視プログラムの処理フローを示す．

図 7: 監視プログラムの処理フロー



ここで，研究室内 LAN のモニタリング環境において Apresia の仕様により送信フレームのミラーリングを行った場合，図 8 のように必ず IEEE802.1Q VLAN タグが付加され転送されることがわかった．これが付加された場合，フレーム本来の EtherType がある位置に TPID(Tag Protocol Identifier) として 0x8100 という値が入るため，Ruby/Pcap では Ethernet フレームとして認識されないことがわかった．

図 8: 802.1Q ヘッダが付加された Ethernet フレーム



そこで，この問題を解決するために Ruby/Pcap ライブラリのソースコードを以下のように書き換え，802.1Q ヘッダが付加されたパケットが観測された場合，802.1Q ヘッダを以降のデータで上書きする処理を追加しこの問題を解決した．

Ruby/Pcap packet.c IEEE802.1Q ヘッダの上書きを行うコード

```
                    //* 略 *//
    pkt->hdr.pkthdr     = *pkthdr;
    pkt->data = (u_char *)pkt + sizeof(*pkt) + pad;
    pkt->udata = Qnil;
    memcpy(pkt->data, data, pkthdr->caplen);

    nl_len = pkthdr->caplen - nl_off;

// 802.1Q header overwrite
    struct ether_header *ethhdr;
    ethhdr = (struct ether_header *)pkt->data;
    if(ntohs(ethhdr->ether_type) == ETHERTYPE_VLAN){
      memcpy(pkt->data + 12, pkt->data + 16, pkthdr->caplen - 16);
      pkt->hdr.pkthdr.caplen = pkthdr->caplen - 4;
      nl_len = pkthdr->caplen - nl_off - 4;
      nl_type = ETHERTYPE_IP;
    }
// 802.1Q header overweite

    if (nl_off >= 0 && nl_len > 0)
        pkt->hdr.layer3_off = nl_off;
                    //* 略 *//
```

また，R/S 解析を行うプログラムも同様に Ruby を用いて開発した．最小二乗法の適用は Ruby/GSL ライブラリを使用したが，Ruby/GSL をインストールする際，GSL の最新バージョンである 1.15 をインストールした環境では Ruby/GSL のコンパイルが正しく終了しない問題があった．GSL1.14 を使用することでこの問題を解決することができた．

4.3 ナイル川最小水位データによる R/S 解析プログラムの検証

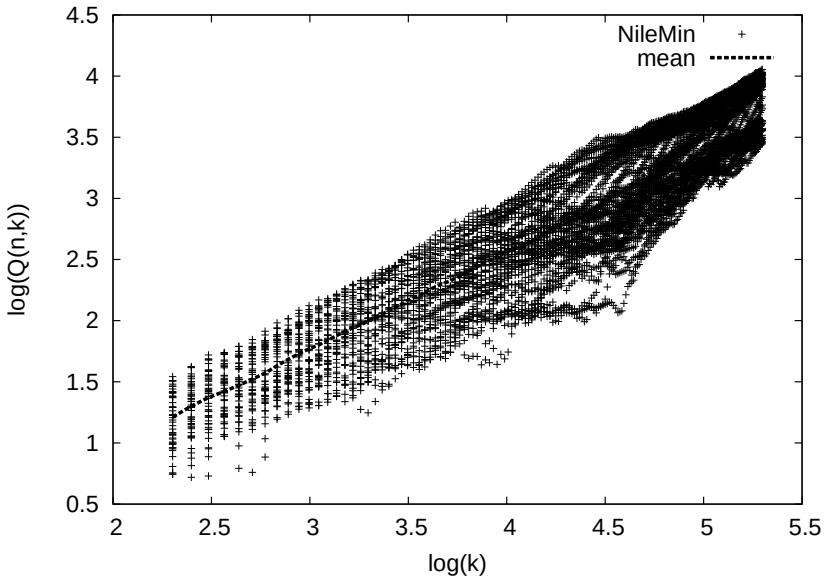
イギリスの水理学者であるハーストはエジプトのカイロで観測されたナイル川の最小水位データにより長期記憶を発見した．R/S 解析によって分析するデータとしてナイル川の最小水位データは最も有名である．作成した R/S 解析プログラムの正当性を確認するために，このデータを用いて検証を行った．パラメータは $1 \leq k \leq 200, m_c = 50$ とした．ナイル川の最小水位データはカイロで 622 年から 1284 年に観測された一年間におけるナイル川の最小水位データ 663 個から成る．このデータは R の FGN パッケージにより取得することができる．R で FGN パッケージをインストールしロードした後，以下のコマンドによりファイルに出力することが出来る．

ナイル川最小水位の出力コマンド

```
write.table(NileMin,file="ファイル名",row.names = FALSE, col.names=FALSE)
```

図 9 にナイル川最小水位の R/S 統計量を示す．結果，文献 [14] に掲載されている同様のパラメータによる解析結果と特徴が一致することから，作成したプログラムの正当性が確認された．

図 9: ナイル川最小水位の R/S 統計量



4.4　使用したマシン

表 4 に研究室内 LAN の監視に使用したマシン，表 5 に学部 LAN の監視に使用したマシンについてまとめる．

表 4: 監視環境 (研究室内 LAN)

マシン	Apple iMac (Mid 2010)
CPU	Intel Core2Duo E7600 @ 3.06GHz
メモリ	8GB
ネットワーク	Apple USB Ethernet アダプタ MC704ZM/A
OS	Mac OS X v10.6 Snow Leopard
開発言語	Ruby1.9.2
ライブラリ	Ruby/Pcap 0.7.3
スイッチ	Apresia 13100-48X

表 5: 監視環境 (学部 LAN)

マシン	HP ProLiant SL390s G7 2U×4
CPU	Intel Xeon X5650 @ 2.67GHz × 2
メモリ	24GB
ネットワーク	HP NC543i 2 ポート 4x QDR IB/10Gb
OS	CentOS release 5.5(final)
開発言語	Ruby1.9.2
ライブラリ	Ruby/Pcap 0.7.7
スイッチ	Apresia 13000-X24-PSR

4.5 実験結果

学内 LAN においてモニタリング実験を行った結果について述べる．

4.5.1 研究室内 LAN における UDP トラフィックの分析

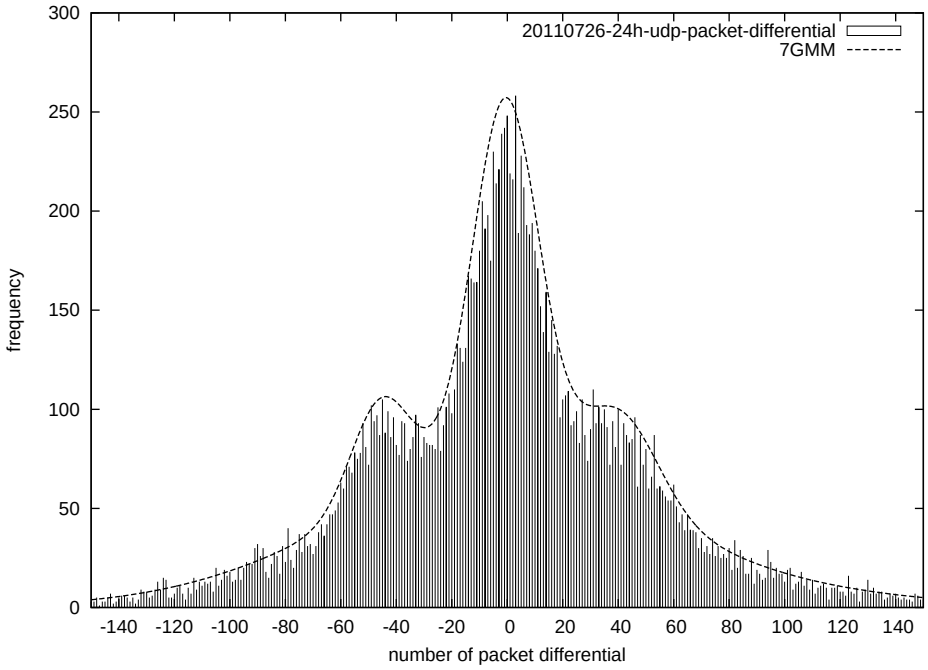
研究室内 LAN のモニタリング環境において観測された UDP トラフィックのパケット数変動，フロー数変動の度数分布図の分析を行った．UDP パケットのパケット数変動の度数分布図と EM アルゴリズムによる混合ガウス分布近似結果を図 10 に示し，EM アルゴリズムによるパラメータ推定結果を表 6 に示す．この図から，平均値ゼロを中心とした分布が存在することがわかるが，EM アルゴリズムの結果では，±40 付近に平均値を持つ異なるガウス分布が存在するという推定になっている．ここで，3.2.1 節で述べた判定方法で調べても，μ の重みつき標準偏差は 23.46，σ の重みつき標準偏差は 26.73 となり，単峰分布ではなく，複峰分布になっていることが推定できる．このパケット数変動の分布について，何らかのプロトコルにより頻繁に起こる ±40 のパケット数変動である可能性が考えられ，この分布について定常トラフィックと捉えるべきかを調べるために，引き続き調査を行った．

表 6: EM アルゴリズムによるパラメータ推定結果（UDP パケット数変動）

γ	0.07491	0.08722	0.1564	0.03862	0.2981	0.2596	0.08519
μ	-45.20	40.40	23.99	29.63	-0.5470	-21.81	9.290
σ^2	118.1	198.8	3671	1677	147.3	2672	9119

次に，文献 [1] で述べられていた，フロー数変動はガウス分布となるという仮定を検証するために，フロー数変動について度数分布図と EM アルゴリズムによる分析を行った．フロー数変動の度数分布図と EM アルゴリズムによる混合ガウス分布近似結果を図 11 に示し，EM アルゴリズムによるパラメータ推定結果を表 7 に示す．図 11 のフロー数変動度数分布図を見ると，ゼロを平均値とした分布が見られるが，およそ ±20 を平均値とした分布も見られ，フロー数変動では，±20 の部分を中心に比較的高い頻度でフロー数変動が起こっていることを示している．同様に 3.2.1 節の判定を行ったが，μ の重みつき標準偏差は 21.45，σ

図 10: UDP パケット数変動の度数分布図



の重みつき標準偏差は 18.08 となり，やはり複峰分布と推定される結果となった．数日間の監視によって得られたデータの多くが同様の特徴を持っていたことから，LAN 内における何らかのプロトコルにより ±20 のフロー数変動が定常的に発生していると考えられる．また，パケット数変動がガウス分布とならないのもこのプロトコルが原因ではないかと考え，以後このフロー数変動について調査した．

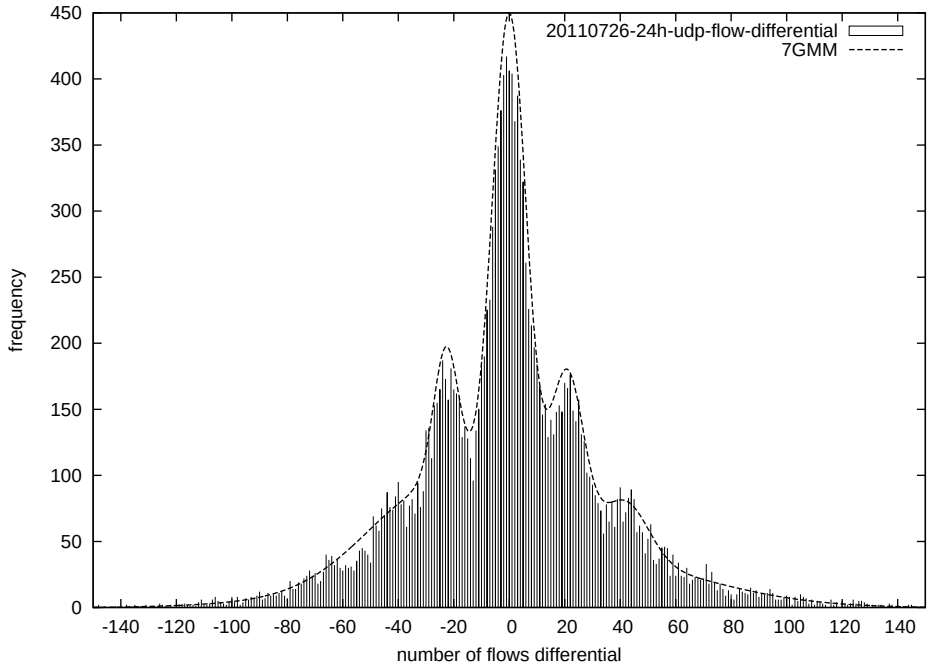
表 7: EM アルゴリズムによるパラメータ推定結果（UDP フロー数変動）

γ	0.3067	0.1036	0.06192	0.05209	0.2543	0.07399	0.1474
μ	0.09188	20.87	-22.65	41.68	-25.28	41.77	2.600
σ^2	37.59	36.07	20.57	72.81	653.1	1030	3311

4.5.2 パケット抽出によるフロー変動調査

表 7 の，EM アルゴリズムの推定結果より，特徴とされる分布の中心が 20.87,-22.65 であると推定された．この結果に基づき，監視プログラムにおいて，フロー数変動が 21±5 の範囲内で観測された際の単位時間内の UDP パケットを libpcap 形式でダンプし，20 前後のフロー数変動が起こった際にどのようなパケットが多く観測されるのか調査を行った．libpcap 形式とは libpcap を用いたパケットダンプファイルであり，tcpdump や WireShark[5] などのパケット分析ツールで扱うことができる．調査では，2011 年 10 月 21 日 13 時から 18 時の 5 時間観測を行い，その結果出力された libpcap 形式のダンプファイルを WireShark を用い

図 11: UDP フロー数変動の度数分布図



て分析した．比較対象として，フロー数変動が 21±5 以外の時間帯に観測されたパケットの libpcap 形式のダンプファイルも用意した．以下，21±5 フロー変動時のパケット群を抽出パケット，比較対象としたそれ以外のパケット群を非抽出パケットと呼ぶ．WireShark はパケットのヘッダ情報やペイロードに含まれる情報からプロトコルを分析することができる．ここで，キャプチャ結果に含まれるプロトコルの割合を算出する，WireShark の ProtocolHierarchy 機能を使用し，抽出パケットと非抽出パケットでは含まれるプロトコルの割合に差が出るか比較した．抽出パケットと，非抽出パケットのプロトコル割合の上位のみを表 8 に示す．これらを比較すると，抽出パケットのプロトコルは Data が 83.74%であるのに対し，非抽出パケットの Data は 73.80%と約 10%差が出ていることがわかる．それ以外のプロトコルについては，抽出パケットが非抽出パケットに比べ，割合が低くなっていることがわかる．WireShark では未知プロトコルと判断された場合，プロトコル名は Data として出力されるが，21±5 のフロー変動の発生率が高い原因として WireShark において Data と認識されるパケットが原因であると考えられる．

次に，どのようなパケットが原因で，21±5 フロー数変動が起こったのかを調査する．ここでまず，WireShark の Endpoints 機能を用い，どのような IP アドレスとポート番号によるパケットが最も多く含まれているのかを調査した．Endpoints 機能では，IP アドレスとポート番号の組の一覧を表示し，そのパケット数，送受信バイト数などを見ることができる．抽出パケットと，非抽出パケットの Endpoints 機能による分析の結果をパケット数順にソートしたものの上位を表 9，表 10 に示す．なお表中の 133.101.56.64 は DNS サーバのアドレスである．抽出パケット，非抽出パケットを比較すると，双方最上位に 133.101.63.255:9 のパケットが来ている．次に，抽出パケットでは二番目に 255.255.255.255:9 が来ており，非抽出パケットと異なる順になっていることがわかる．抽出パケット数，非抽出パケット数それぞれのパケット数の割合を表 9，表 10 のパーセンテージに示した．双方を比較すると，133.101.63.255:9，255.255.255.255:9 の 2 つのみが抽出時

表 8: ProtocolHierarchy(5 時間)

プロトコル名	抽出 (%)	非抽出 (%)
Data	83.74	73.80
DNS	7.23	10.17
STUN	2.94	5.39
Canon BJNP	1.86	3.48
BootstrapProtocol	1.20	2.07

表 9: Endpoints - 抽出パケット (5 時間)

IP アドレス:ポート	パケット数	(%)
133.101.63.255:9	13698	56.21
255.255.255.255:9	3127	12.83
133.101.56.64:53	1684	6.911
244.0.0.1:626	1155	4.740
133.101.63.255:22936	936	3.841

に含まれる割合が増加し，他は減少する結果となっている．つまり，今回適用したフロー数変動が 21±5 時のパケットというルールによって UDP ポート 9 番のブロードキャストパケットをわずかに高い割合で抽出できていることが分かった．

4.5.3 学部 LAN におけるトラフィックの分析

学部 LAN で観測された TCP のフロー数変動について，度数分布図と EM アルゴリズムによる分析を行った．ここでは，24 時間観測した分布ではなく，2011 年 12 月 19 日の講義時間中 (13:15-14:45) のみの分布である．講義時間中のみとした理由は，講義時間中以外は研究室内 LAN 同様，ほとんどパケットが観測さ

表 10: Endpoints - 非抽出パケット (5 時間)

IP アドレス:ポート	パケット数	(%)
133.101.63.255:9	51969	43.42
133.101.56.64:53	11867	9.914
224.0.0.1:626	10113	8.449
255.255.255.255:9	7135	5.961
133.101.63.255:22936	6966	5.820

れないため，ネットワークが利用されている状態での分布を得たいと考えたためである．また，EM アルゴリズムによる推定では少ないデータからより詳細な特徴を近似するために 9 混合ガウス分布推定を行った．図 12 に度数分布図と EM アルゴリズムによる混合ガウス分布近似結果，表 11 に EM アルゴリズムによるパラメータ推定結果を示す．

図 12: TCP フロー数変動の度数分布図

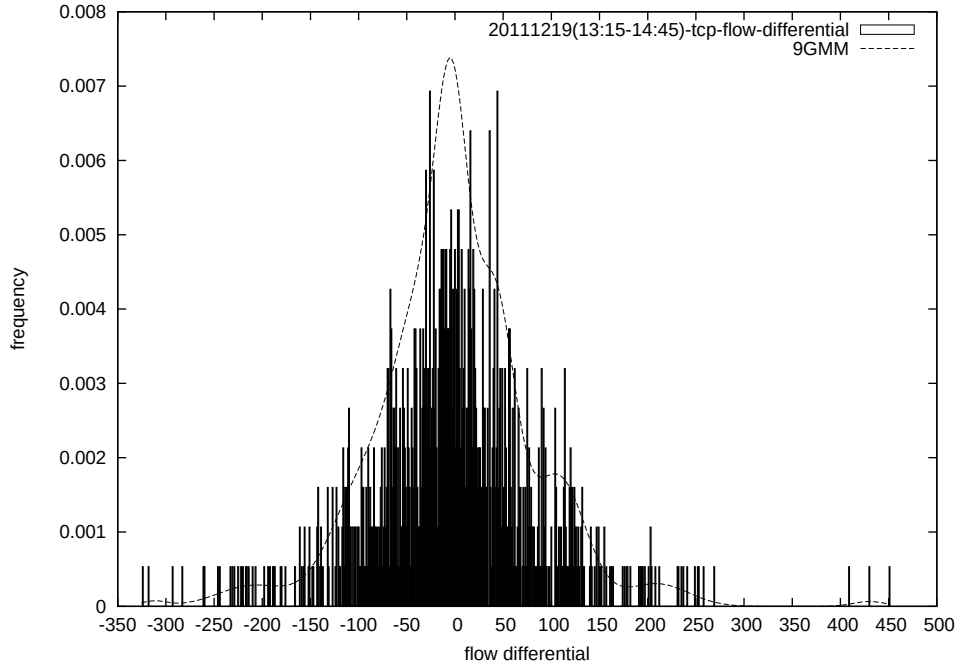


表 11: EM アルゴリズムによるパラメータ推定結果（TCP フロー数変動）

γ	0.1893	0.1225	0.02485	0.002780	0.1326	0.02595	0.002467	0.2118	0.2876
μ	-2.358	106.9	209.5	430.0	-95.52	-208.9	-312.3	41.25	-32.25
σ^2	284.2	791.2	1066	0.2939	1147	1362	202.2	454.9	905.4

同様に，3.2.1 節で述べた判定方法で調べると，μ の重みつき標準偏差は 79.04，σ の重みつき標準偏差は 6.350 となり，単峰分布ではなく，複峰分布になっていることが推定できる．ここでは，UDP 同様ゼロを中心とした分布が得られたが，研究室内 LAN の UDP に比べ裾の長い分布となっている．EM アルゴリズムで分布の特徴を平均値として得られているが，観測時間が短いため今後更に観測時間を広めた分布の調査を行う必要がある．

4.5.4 講義中の学部 LAN のトラフィック特性

次に，同様に 2011 年 12 月 19 日の講義時間中 (13:15-14:45) に学部 LAN で観測されるトラフィックを時系列で見た場合どのような特徴が見られるのかを調査した．ここではプロトコルごとに 5 秒ごとのパケット数を調査し時系列変化を確認する．なお，プロトコルごとのパケット数の出力機能は監視プログラムにおいて実装できていないため，tcpdump によりパケットを出力した後，各プロトコルごとにファイルを分けた後，別途 tcpstat[15] を使用し 5 秒ごとのパケット数を算出した．tcpstat において任意の形式を出力するには以下のようなコマンドを実行する．

tcpstat で任意の形式でデータを出力

```
tcpstat -r traffic.dmp -o "%A\n" 5 > arp.data
tcpstat -r traffic.dmp -o "%C\n" 5 > icmp.data
tcpstat -r traffic.dmp -o "%T\n" 5 > tcp.data
tcpstat -r traffic.dmp -o "%U\n" 5 > udp.data
```

結果を図 13〜19 に示す．また，一部プロトコルのパケット数変動度数分布図を図 20〜22 に示す．

これら図から TCP, NFS において，講義開始直後に高いパケット数が観測されていることが分かる．NFS は TCP の上位層プロトコルであり，TCP と NFS の時系列変化がほぼ一致していることから TCP パケットの多くが NFS プロトコルであると考えられる．講義開始直後に NFS パケットが最も多くなる原因として，NetBoot の利用が挙げられる．NetBoot とは，Mac OS X Server の提供する機能で，サーバ側に置かれたディスクイメージをネットワーク経由で読み込み，Mac OS X を起動し使用できるようにする仕組みである．今回観測を行った時間帯はプログラミング演習が行われており，講義開始時に一斉に NetBook により起動することで NFS のパケットが多く観測されるものと考えられる．起動後は，アプリケーションの起動やファイルの読み書き程度にとどまりパケット数がほぼ一定値となる．図 23 に示した TCP フロー数の時系列変化では，開始直後はやや高いフロー数となりながらも全体を通して変動が小さい変化となっている．このようにパケット数の時間変化と比較してフロー数の変化が大きく見られない理由として TCP パケットの多くを占める NFS がほぼ同一のフローしか用いないためと思われる．また，UDP, HTTP においてもパケット数は起動直後大きな値を示しているが NetBoot との関連性については不明である．その他では，ARP が特徴的な時間変化をしており，上下の変動が大きくその頻度も高い時系列となっている．このような時間変化をする理由は現在のところ不明であるが，一定周期の変動が見られるため，ARP キャッシュなどが関連している可能性がある．

また，度数分布図ではゼロを中心とした分布になっている．この特徴は掲載していない他のプロトコルでも確認されている．各プロトコルにおいてパケット数ゼロである状態は ICMP を除きほぼ観測されていないことから，ある一定値が続くことが多い時系列であると考えられる．図 22 の ARP に関してはゼロを中心としながらも，左右の分布も他より高い値を示しており変動が多い時系列であると言える．

4.5.5 R/S 解析によるトラフィックの分析

学部 LAN において観測された 2011 年 12 月 19 日の講義時間中 (13:15-14:45) のトラフィックパラメータで R/S 解析を用いたハースト数推定を行った．各プロトコルごとにその結果の値を示す．このように，ARP

図 13: 時系列変化 (TCP)

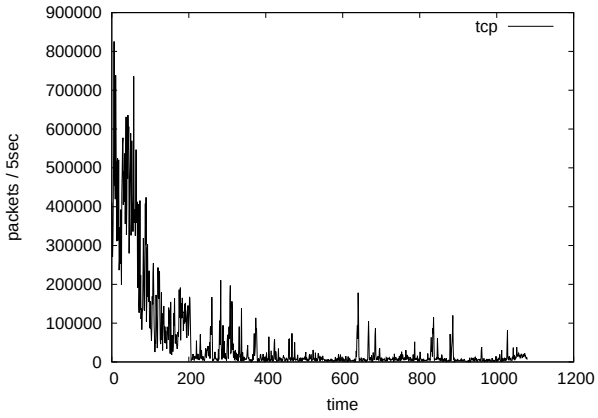


図 14: 時系列変化 (UDP)

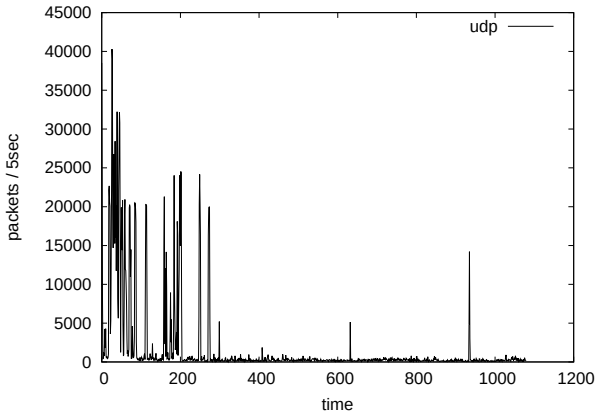


図 15: 時系列変化 (HTTP)

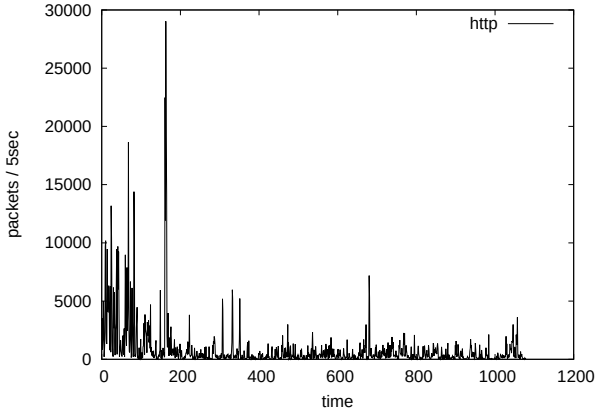


図 16: 時系列変化 (DNS)

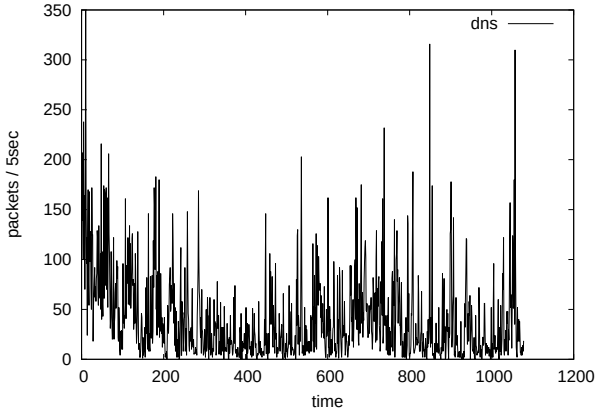


図 17: 時系列変化 (NFS)

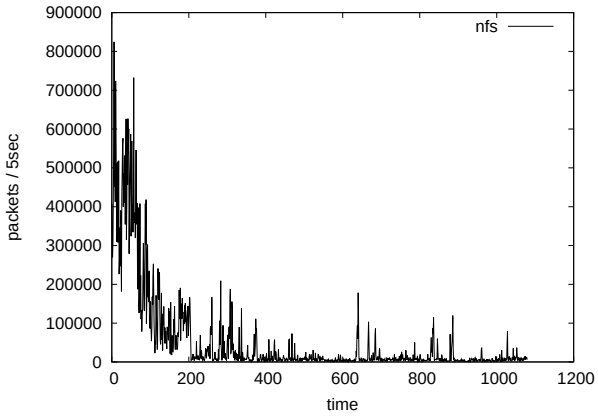


図 18: 時系列変化 (ARP)

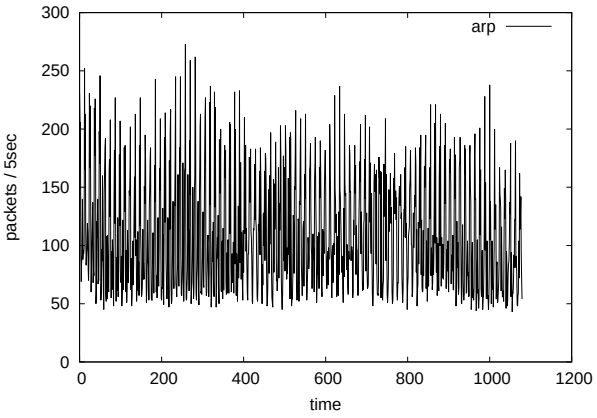


図 19: 時系列変化 (ICMP)

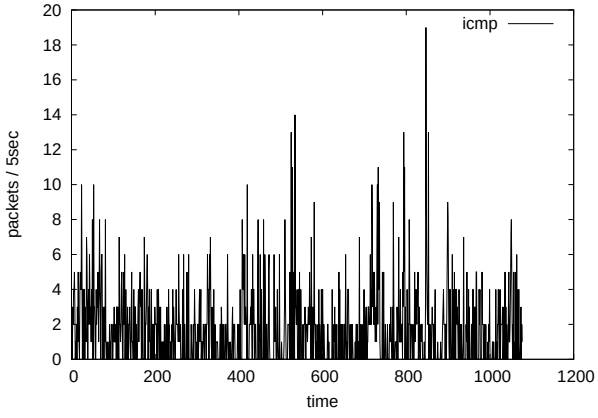


図 20: パケット数変動度数分布図 (TCP)

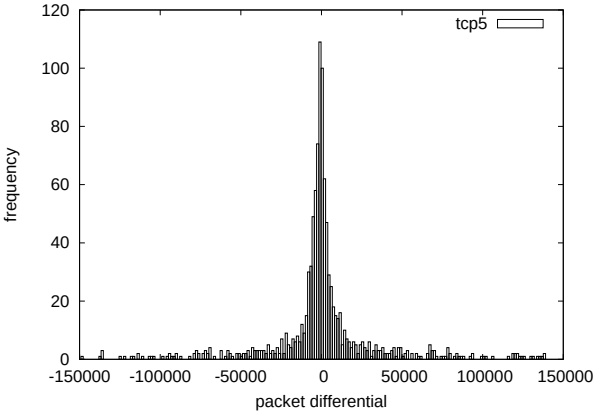


図 21: パケット数変動度数分布図 (UDP)

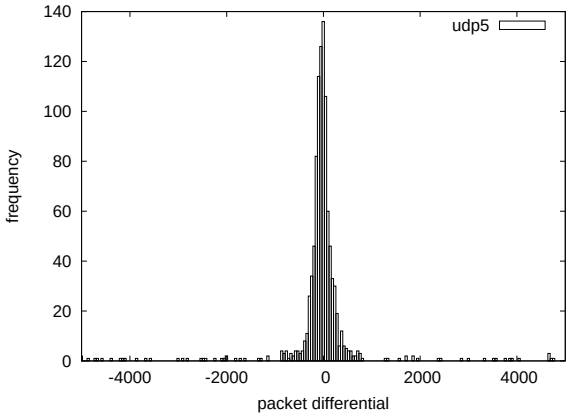


図 22: パケット数変動度数分布図 (ARP)

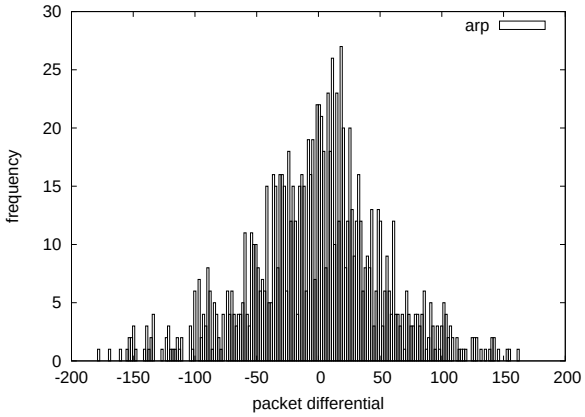
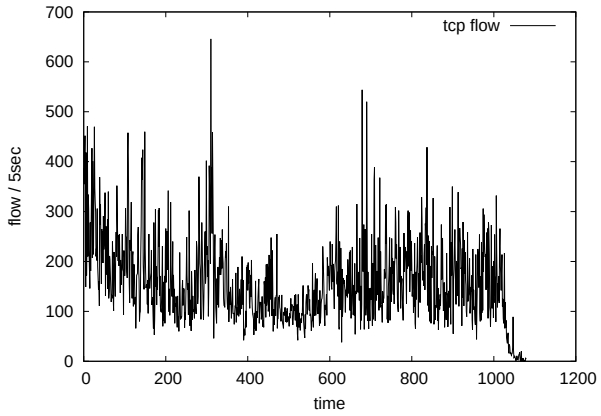


図 23: 時系列変化 (TCP フロー数)



以外のプロトコルはすべて $H > 0.5$ となり自己相似性，長期記憶性があると推定された．ARP は，図 18 において上下の変動の激しい時間変化が確認されており，表 3 より理論上正しい推定結果と言える．

4.5.6 サンプリング周期を変えた際のハースト数の調査

ここで，先程と同様の期間の TCP, UDP ダンプファイルを用いて 0.01 秒ごとのパケット数〜10 秒ごとのパケット数とサンプリング周期を変えた観測値を用意しサンプリング周期の変化によりハースト数の変化が見られるかどうか調査を行った．観測時間は 90 分であるため，サンプリング周期 0.01 とした場合 54 万個の観測値がられる．パラメータは $k_{\max} = 100, m_c = 100$ に固定したところ，図 24 のような結果がとなった．両プロトコルとも，サンプリング周期 0.6〜0.7 秒までは推定結果は急激に減少している．その後 UDP はすぐに安定した推定結果が続き，サンプリング周期が 7 秒以降でばらつきが大きくなる．TCP はサンプリング周期 0.7 秒以降やや上昇傾向となり，4 秒以降に安定した推定結果が得られている．ここで，0.01 秒から 0.7 秒の間の急激な変化と 3.3.2 節で示した重複の割合の関連性について考える．$k_{\max}, m_c$ より R/S 解

表 12: プロトコルごとのハースト数推定結果

プロトコル	ハースト数
TCP	0.811
UDP	0.715
HTTP	0.694
DNS	0.722
NFS	0.814
ARP	0.189
ICMP	0.730

析によって分析に使用される最大データ数は，10000 個となる．サンプリング周期が 0.5 秒より短い場合，10000 個以上のデータが得られることから重複する割合はサンプリング周期 0.5 以下で 0 となる．重複する割合がゼロの場合，Δn は最大値を取るため理想であるが，サンプリング周期が 0.01 秒の場合 54 万個のデータに対して初めの 10000 個しか分析に使用されない点に注意しなければならない．この場合，初めの 100 秒間の長期記憶性の推定結果となったため高いハースト数が推定されたと考えられる．より多くのデータによる推定は精度の高い結果をもたらすが，データ全体を用いて分析を行うような $k_{\max}, m_k$ を設定するかまたは，そのようなデータ数を用意することが重要である．つまり，少なくとも $k_{\max}, m_c$ または N を以下の条件を満たすよう設定する必要がある．

$$N \leq (k_{\max} m_c) \tag{25}$$

UDP でサンプリング周期が 7 秒以降，結果にばらつきが見られる理由について，重複の割合の変化が小さいにもかかわらずばらつきに変化が見られることから，データ数の減少が主な原因になっていると考えられる．しかし，TCP では最後まで結果が比較的安定していることから，ハースト数の高さも結果のばらつきに関係している可能性があり今後さらに調査する必要がある．この結果，適切なパラメータを設定することで，サンプリング周期を変えた場合でもハースト数の推定結果がほぼ一定値を示すことが確認された．

また，図 25 にサンプリング周期が 1 秒から 10 秒のデータに対して最大重複率が 0.9 となるよう k_{max} を調整しハースト数推定を行った結果を示す．図 25 でもサンプリング周期の変化にかかわらずハースト数はほぼ一定の値を示していることが分かる．

図 24: サンプリング周期によるハースト数変化

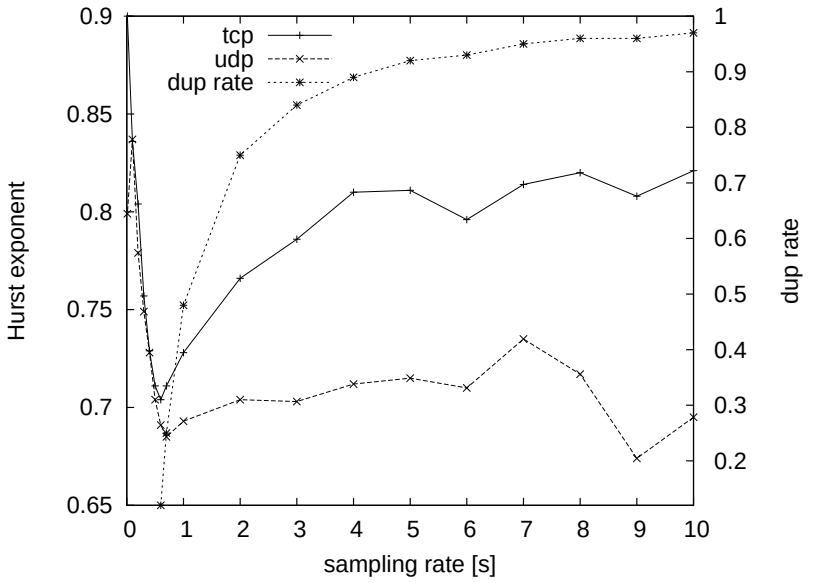
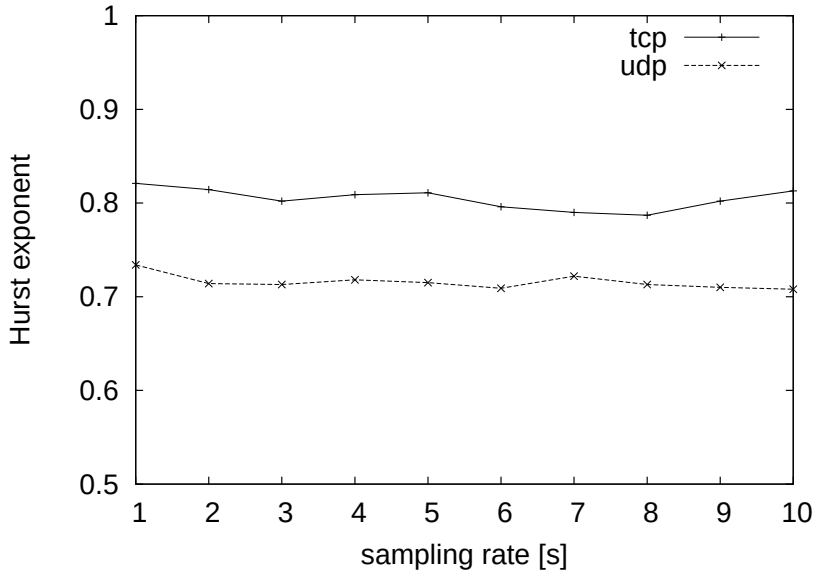


図 25: サンプリング周期によるハースト数変化 (重複率 0.9)



5 考察

以下では実験結果と，実装したモニタリングシステムについて考察を行う．

5.1 LAN 内定常トラフィックに関する考察

今回の調査によって研究室内 LAN のトラフィックにはフロー数を約 ±20 変動させるプロトコルの存在が予想され，ルールに基づいたパケット抽出の結果 UDP9 番ポートのブロードキャストパケットの抽出率が高いことが分かった．このブロードキャストパケットは WOL(WakeOnLAN) によるものである．WOL はネットワーク経由でコンピュータの電源操作を行うプロトコルで，マジックパケットと呼ばれるパケットによってコンピュータの電源操作を行なっている．マジックパケットは普通 UDP ブロードキャストパケットでありペイロードは FF:FF:FF:FF:FF:FF に続けて操作を行いたい機器の MAC アドレスを 16 回繰り返したものである．今回モニタリングを行った研究室内 LAN には，他研究室所有の複数の NAS(Network Attached Storage) が設置されており，その NAS に対してあるクライアントから 30 秒から 60 秒の範囲で定期的にマジックパケットが発せられていることが分かった．また宛先 IP アドレスの 133.101.63.255 と 255.255.255.255 の違いについては，クライアントの OS による違いであることがわかった．

WireShark において WOL という既知のプロトコルが判別出来ず，Data として分類されていた原因として，Ruby/Pcap ライブラリの問題であることが分かった．Ruby/Pcap ではパケットを libpcap 形式に出力する機能を持っているが，出力の際にパケットのペイロード部分が削られてしまう場合があることが分かった．WireShark を用いてパケットをキャプチャした場合，正しく WOL として認識されたため，ペイロード部分が削られた結果 WireShark においてプロトコル分析に失敗し，プロトコル名が Data として認識されたと考えられる．この Ruby/Pcap の問題については今後修正する必要がある．本研究では LAN 内でのパケット数やフロー数の変動がガウス分布に基づいているという仮定をしてきたが，WOL のようなプロトコルを利用する機器が LAN 内に複数存在すると，ガウス分布に従わないトラフィックが生成されることが分かった．ガウス分布を仮定すると分析処理が容易になるが，LAN 内では必ずしも適用できないと考えられる．複峰分布のトラフィックから異常を検知する手法についてはまだ確立できていないが，今後定常トラフィックの調査を進める中で検討する必要がある．

また，学部 LAN における実験結果については詳細な分析ができていないため，今後さらなるデータ収集と分析が必要となる．

今回抽出された WOL の例のように，必ずしも完全に把握していない端末やプロトコルが同一 LAN 内に存在するのは大学ネットワークの 1 つの特徴でもある．大学では研究室ごとに異なる環境を構築しているケースが多く，一般的な企業で観測を行う場合に比べてトラフィックの把握が困難になる可能性が高い．しかし，大学の端末がボットネット等で不正利用されることが多く，今後は大学でも LAN 内の分析が必須となる可能性がある．本研究で検討したパラメータ分布の近似による定常トラフィックの分析手法は必ずしも低コストで分析できる手法ではないが，今後コストを低減することで，低コストなトラフィック分析手法に拡張できる可能性がある．観測範囲を拡大する際に問題となるのが，必ずしも全学でモニタリングに関する合意が得られない点である．合意が得られにくい理由の一つとして，個人情報の問題が挙げられる．今回のモニタリングではブロードキャストパケットに対して WireShark でペイロードを分析するという処理を含んでいる．学内において不特定多数の使用する環境ではペイロードに個人情報が含まれる可能性があり，

ペイロード部分をカットするといった対応が必要になる．例えばポート番号のみを分析するなど，個人情報に配慮した調査手法の検討が必要になる．

5.2 R/S 解析に関する考察

R/S 解析を使用する上で重要であるのは，可能な限り大きなデータ数を用意すること，そのデータをすべて使用し，さらに重複の割合が小さくなるような $k_{\max}, m_c$ を設定することにある．

また，実験で行った UDP トラフィック分析においてサンプリング周期を変えた場合にハースト数が変化しないことは，今後の課題である異常検知手法への適用時に，準リアルタイムでの異常検知の可能性を示している．さらにこのことは，同様に長期記憶性を示している他のプロトコル以外にも適用可能であると考えられる．また，ハースト数による異常検知手法を考えた場合，ネットワークにどのような変化が起こった場合にハースト数が変化するのかを調査する必要があり，異なるデータによる調査が必要である．

5.3 LAN 内のトラフィック分析コストに関する考察

今回の調査で用いたようなパラメータ分布によるトラフィック分析を定常トラフィックの把握に利用するためには，分析コストの低減が重要となる．そこで，以下では今回の分析にかかったコストについて考察する．

5.3.1 抽出による処理時間の削減

研究室内 LAN での実験結果のように，定常トラフィックのパラメータがガウス分布とそれ以外の分布のように分類できる場合，ガウス分布のパラメータについては既存の手法で異常検知し，それ以外のパラメータのみパケット単位で分析するといった方法が考えられる．表 13 では，研究室内 LAN において全 UDP パケットを出力した場合と，21±5 のフロー数変動という条件でパケットを抽出した場合のパケット数，フロー数の比較である．パケット数，フロー数などをほぼ半分に削減することができており，上述のようなハイブリッドな分析手法についても検討する価値があると考えられる．

表 13: 抽出時非抽出時比較 (5 時間分)

	全 UDP	抽出時
パケット数	315,032	132,929
フロー数	88,677	45,434

5.3.2 DB 化によるコスト削減

本研究では WireShark によって分析を行なってきたが，WireShark には大量のパケットがダンプされたファイルを読み込む際に時間がかかってしまう問題がある．時間がかかる原因の一つに，プロトコルデコード処理による読み込み時間の長時間化が挙げられる．プロトコルデコード機能を切ることで，読み込み時間

は大幅に削減できるが WireShark においてトラフィックを分析する場合，プロトコルデコード機能は必要不可欠であり，この機能を使用せずに WireShark における分析を行うのは困難であると考えられる．そこで，研究室内 LAN のモニタリング環境においてパケットのヘッダ情報のみを DB 化することで記録，分析コストの低減を図った．DB には MySQL を使用しインタフェースは PHP を用いて開発を行った．

まず，監視プログラムに MySQL への出力機能を付け加えた．DB 化する情報はパケットの送信元，送信先 IP アドレスと同ポート番号のみで，ペイロードをカットすることでさらなるコスト低減を図る．PHP で開発したインタフェースにより WireShark における Endpoints 機能，フロー数分析などのフロー分析機能，パケットの IP アドレスやポート番号，プロトコル名による検索機能を実装した．これにより先に述べた WireShark による分析を Web ブラウザ上で行うことが可能となった．次に，libpcap 形式で出力した場合と DB に出力した場合のコストを比較した．表 14 は libpcap 形式ダンプファイルと DB の分析に必要なコストの比較である．このように，ファイルオープンにかかる時間やファイルサイズを大きく削減することができた．また，その他の分析機能の所要時間においても DB の優位性が確認された．Web ブラウザ上で閲覧できる分析インターフェースを図 26，図 27 に示す．

表 14: libpcap 形式と DB の比較（24 時間分）

	libpcap	**DB**
ファイルオープン	1 分 5 秒	5 秒未満
ファイルサイズ	69.1MB	15.1MB

図 26: PHP によるトラフィックデータ閲覧ツール (各パケット)

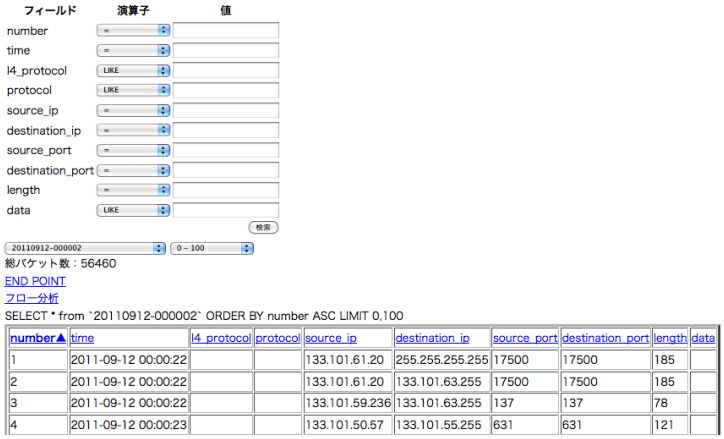
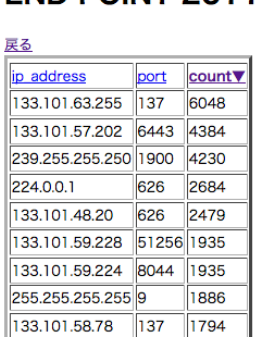


図 27: PHP によるトラフィックデータ閲覧ツール (EndPoint)



6 まとめと今後の課題

本研究では，LAN 内のパケット数の変動はガウス分布になるという仮定のもと定常トラフィックのパラメータの度数分布の分析を実施した．EM アルゴリズムによる混合ガウス分布近似を行い，LAN 内において観測されるパケット数やフロー数といったパラメータが必ずしもガウス分布にならないことを確認した．また，R/S 解析によるハースト数の推定の結果短時間の観測による分析ではあるが，学内の LAN トラフィックにおいて長期記憶性を確認することができた．分析コストを調査し，パケットの抽出やヘッダ情報の DB 化など，さらに改良を加えることで本研究で調査に用いた手法をさらに低コストで利用できる可能性がある．LAN 内の定常トラフィックの分析に加えて，今後低コストな分析手法の開発も実施する予定である．また，学内におけるモニタリングに関しては，技術面以外の課題も存在し，今後個人情報を考慮した手法の検討が必要である．

LAN 内の異常検知において，トラフィックパラメータの分布について単純なガウス分布を想定できない可能性があることが確認できた．今後，ペイロード長などのパラメータの度数分布に関しても調査を行う必要がある．また，学部 LAN において処理能力を超えるトラフィック量となった場合，sFlow のようなサンプリング手法についても検討する必要がある．さらに，トラフィック分布が特異になる原因について調査する際に WOL の例のようにプロトコルの詳細分析が必要になるが，DB に全パケットを格納して事後分析する機能についても実装する予定である．ハースト数によるトラフィック分析において，さらに長期的な観測を行う必要がある．

また，ハースト数を用いたトラフィック分析では，異常検知に利用可能と考えられる長期記憶性，自己相似性を持ったトラフィックを観測可能であることがわかった．R/S 解析による分析を行う際に適切なパラメータの設定が必要であり，精度の高い結果を得るためには一定の条件を満たす必要があることがわかった．異常検知への適用可能性は今後さらに長期的な観測を行い，ハースト数の定常性を確認する必要がある．

謝辞

本研究を進めるにあたり，研究や論文執筆において熱心なご指導や，実験・研究環境の構築に尽力頂いた秋山豊和先生に心より感謝いたします．また，学内ネットワークモニタリング環境の構築と実験にご理解，ご協力をいただいた大本先生，荻野先生，安田先生をはじめとする本学コンピュータ理工学部の教員の方々に心より感謝いたします．そして，合同発表会において貴重なご意見とご指摘を頂きました林原特研の皆様，日々の報告会等において活発なご議論を頂いた秋山特研の皆様に心から感謝いたします．最後に，日々の生活を支えていただいた家族に感謝いたします．

本研究の一部は，SCOPE 若手 ICT 研究者育成型研究開発の助成を受け実施したものである．

参考文献

[1] 原田薫明, 川原亮一, 森達哉, 上山憲昭, 廣川裕, 山本公洋, “異常トラヒック発生検出および終了判定手法”, 電子情報通信学会技術研究報告. IN, 情報ネットワーク 106(420), pp. 115-120, (2006).

[2] 上田浩 他, “確率過程による LAN トラヒックのモデル化における一考察”, 情報処理学会論文誌. 数理モデル化と応用 48(SIG_2(TOM_16)), pp. 167-174, (2007)

[3] “Snort” http://www.snort.org/

[4] “OpenDPI” http://www.opendpi.org/

[5] “WireShark” http://www.wireshark.org/

[6] “libpcap/tcpdump” http://www.tcpdump.org/

[7] “MRTG” http://oss.oetiker.ch/mrtg/

[8] K.Wang and S.Stolfo, “Anomalous payload-based network intrusion detection”, Recent Advance in Intrusion Detection (RAID), (2004).

[9] Dempster, A.P., Laird, N.M., Rubin, D.B., “Maximum Likelihood from Incomplete Data via the EM Algorithm”, Journal of the Royal Statistical Society, Series B (Methodological) 39 (1), pp. 1-38, (1977).

[10] J.Bilmes, “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models”, International Computer Science Institute, Technical Report, (1998).

[11] Karlis D, Xekalaki E “Choosing Initial Values for the EM Algorithm for Finite Mixtures.”, Computational Statistics & Data Analysis, 41, pp.577-590, (2003).

[12] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, “On the self-similar nature of Ethernet traffic (extended version)”, IEEE/ACM Trans. on Networking, Vol.2, pp. 1-15, (1994).

[13] “The R Project for Statistical Computing” http://www.r-project.org/

[14] 松葉育雄, “長期記憶過程の統計—自己相似な時系列の理論と方”, 共立出版, (2007).

[15] “tcpstat” http://www.frenchfries.net/paul/tcpstat/