第 10 部

ネットワークトラフィック統計情報の収集と解析

第 1 章

はじめに

WIDE MAWI WG は、広域分散環境におけるトラフィックデータの「収集」、「解析」、「保存」、「利用」等のために必要とされる技術に関する研究を行なうことを目的に活動を行なっている。

本年度は、トラフィックデータのビジュアライゼーションということで、SNMP で収集したデータをグラフ化し WWW で表示する為のツールの評価と実際に WIDE ホームページでの公開を行なった。また、国際線の解析として、TCP のコネクションに着目した解析を行なった。

以下の章ではそれぞれ以下の報告を行なう。

1. SNMP を利用したネットワーク・トラフィックの収集と解析
2. 国際線 TCP コネクションの統計的解析

第 2 章

SNMP を利用したネットワーク・トラフィックの収集と解析

2.1 はじめに

MAWI(旧 NetStat) ワーキンググループでは，WIDE バックボーンのトラフィック収集を実施している．ネットワーク・トラフィックのデータは NNStat というトラフィック収集のプログラムを利用して WIDE バックボーン上の各 NOC で，収集されている．しかし，回線の高速化，シスコやベイネットワークスなどの専用ルータの増加，ハブからイーサネットスイッチなどへの以降により，NNStat を利用したネットワーク・トラフィックの収集が困難な状況が生じている．

このような状況で，トラフィックを収集の一つの解決として SNMP(Simple Network Management Protocol) の利用が考えられる．SNMP のサーバはほとんど全ての専用ルータに実装されており，ワークステーションベースのルータにも容易に導入できる．管理者は SNMP クライアントを実行してリモートのルータ上で実装されている SNMP サーバと通信し，転送パケット数やバイト数のような単純な統計情報や，TCP/IP のデータ構造に対応する複雑な変数を含む概念的な変数の集合から，必要なデータの値を容易に取得できる．また，それらのデータを加工して WWW 上に視覚化して表現することも可能となる．

本章では，SNMP を利用したトラフィック・データ収集の足掛かりとして行ったツールの作成と，複数のルータの SNMP サーバからトラフィックデータを収集してグラフおよび HTML ファイルを生成する MRTG(Multi Router Traffic Grapher) というツール，および WIDE バックボーンにおける MRTG の利用について述べる．

2.2 トラフィック収集の現状

WIDE バックボーンのネットワーク・トラフィックの収集は，従来から NNStat を利用して行われている．NNStat は SunOS などで動作し，BPF(Berkely Packet Filter) または NIT(Network Interface Tap) を使用して設定ファイルに記述された設定により，パケットのソースおよびデスティネーションアドレスや転送バイト数，プロトコル番号別などの詳

細なトラフィック収集が可能である．

WIDE バックボーンでは従来はほとんどのルータが Sun SPARC Station であったため，ルータである SPARC Station で NNStat を起動しておけば収集することができた．専用ルータが増加しても，専用ルータと同一のセグメントに NNStat を稼働させた SPARC Station があれば，NOC セグメント側のインタフェースの MAC アドレスを指定することで，そのルータのトラフィックを収集することができる．

しかし，現状では回線の増加や高速化によるトラフィックの増加に対応して各 NOC 内のネットワークには通常のハブではなくイーサネットスイッチが使用されつつある．したがって NNStat が使用する BPF や NIT ではトラフィックを収集することが困難になっている．

2.3　SNMP を利用したネットワーク・トラフィック収集

NNStat は設定によりパケットそれぞれについて詳細な収集が行える長所があるが，前節に述べた理由により利用が困難になってきている．そこで SNMP(Simple Network Management Protocol) を利用したネットワーク・トラフィック収集について述べる．

2.3.1　SNMP と利用上の問題点

SNMP はサーバ・クライアントモデルで構成されており，管理者は SNMP クライアントをローカルなホストで実行し，リモートのルータ上で実行されている SNMP サーバと通信する．各 SNMP サーバは，転送パケット数やバイト数のような単純な統計情報や，TCP/IP のデータ構造に対応する複雑な変数を含む概念的な変数の集合を保持しており，MIB(Management Information Base) にサーバが保持する変数の集合と各変数の意味が定義されている．管理者は SNMP クライアントを実行して SNMP サーバと通信し必要な値を容易に取得できる．

SNMP サーバはほとんど全てのルータに実装されており，またワークステーション・ベースのルータにも容易に導入可能なフリーなパッケージがあり利用できる．しかし，SNMP クライアントは，フリーなパッケージでは必要最低限の機能のみを有するプログラムが多く，単体では実用にならない場合が多い．WIDE バックボーンでのネットワークトラフィック収集を考えた場合，複数存在するルータの SNMP サーバに対して必要なデータを容易に設定できること，サーバ毎にデータを保存できること，グラフや HTML を自動生成し，WWW での視覚的な表現できることが，ソフトウェアが求められる．

2.3.2 SNMP を利用した簡易収集ツール

我々は当初，CMU の SNMP パッケージ[1]とシェルスクリプト，perl スクリプト，そして gr というグラフを gif ファイルに出力するアプリケーションを利用してトラフィックデータ収集を行った．

動作の流れは以下の通りである．まず，SNMP クライアントの SNMPwalk を使用して，NSPIXP におけるイーサネットスイッチの各ネットワークインタフェースのフレーム数を 10 分ごとに収集し，ファイルに出力するシェルスクリプトを起動する．次に出力されたファイルを整形し，gr に入力させるためデータファイルを作成する perl スクリプトを起動する．最後に作成されたデータファイルを gr に入力させるとグラフの gif ファイルが出力される．

この実行結果を図 2.1に示す．

このプログラム群によって，1 日分のグラフを毎日出力することはできたが，データを長期に渡って収集することは考慮されていない．また，設定はシェルスクリプトに直接書き込む形であり，別のデータを収集するためには書き直す必要があること，複数のルータ，種類のデータを収集することが考慮されていないこと，などの問題点があった．

2.4 Multi Router Traffic Grapher(MRTG)

2.4.1 概要

SNMP を利用するネットワーク・トラフィック収集ツールとしては，MRTG[2](Multi Router Traffic Grapher) がある．MRTG はネットワークリンク上のトラフィックをモニターするツールである．MRTG はトラフィックの状態をビジュアルに提供する gif イメージを含んだ，HTML ファイルを自動で出力する．MRTG は，UNIX や WindowsNT 上で動作する．

2.4.2 MRTG の構成

MRTG は，ルータ上で動作する SNMP サーバからトラフィックなどの統計を読み出すための Perl[3]スクリプトと，データを記録し，グラフ化する高速な C のプログラムから構成されている．またグラフの gif ファイルを出力するために，gd[4]と呼ばれるライブラリを使用している．

---

[1]現在は UCDavis(University California at Davis) でメンテナンスされている．1997 年 5 月 26 日現在のバージョンは 3.2．`ftp://ftp.ece.ucdavis.edu/pub/snmp` 参照

[2]Swiss Federal Institute of Technology Zurich, Department of Electrical Engineering が作成配布．`http://www.ee.ethz.ch/~oetiker/webtools/mrtg/`を参照

[3]Perl はバージョン 5 以降を使用

[4]`http://www.boutell.com/gd/`を参照

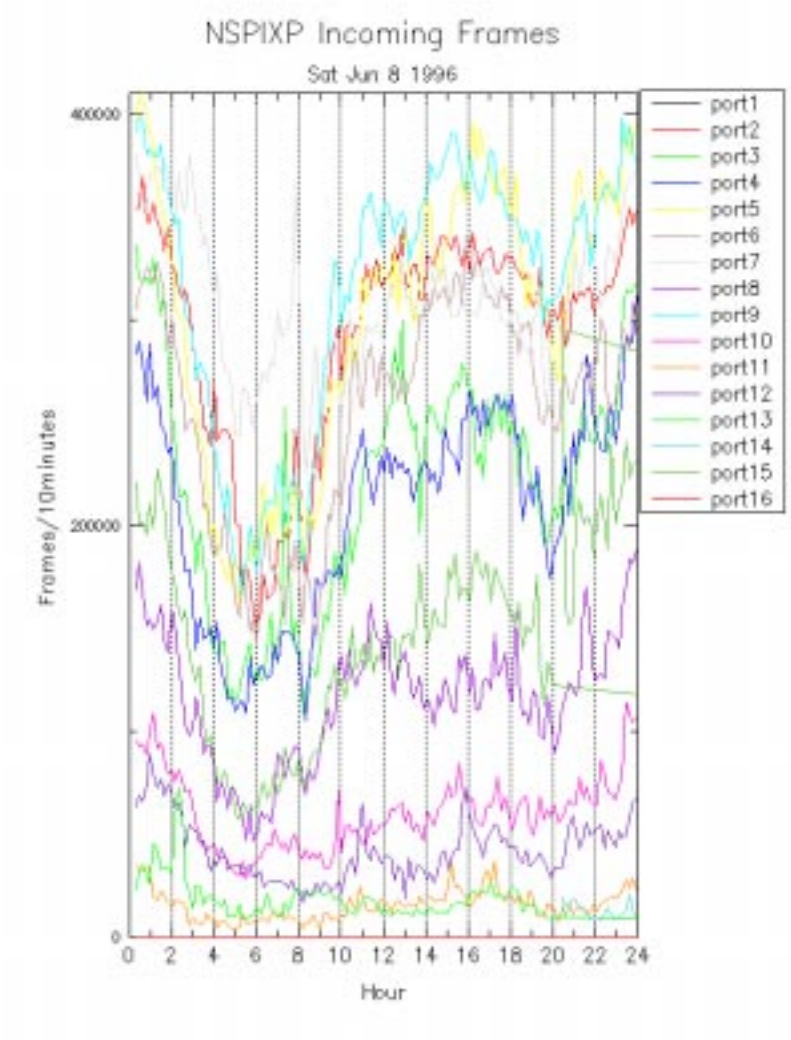

図 2.1: SNMP 簡易収集ツールによる実行結果

2.4.3 MRTG の設定

MRTG は，プログラム本体を 5 分毎に実行するように crontab で設定する．

設定ファイルの記述は，cfgmaker というツールを使用すると容易に行える．community name とルータのホスト名を指定するだけで，そのルータに存在するネットワークインタフェースの設定を出力するので，編集して必要なものだけ残す．図 2.2に cfgmaker の出力例を示す．

```
Target[hoge.wide.ad.jp.4]: 4:foo@hoge.wide.ad.jp
MaxBytes[hoge.wide.ad.jp.4]: 192000
Title[hoge.wide.ad.jp.4]: hoge (hoge.wide.ad.jp): Serial1
PageTop[hoge.wide.ad.jp.4]: <H1>Traffic Analysis for Serial1
<BR></H1>
<TABLE>
  <TR><TD>System:</TD><TD>hoge in WIDE NOC, JAPAN</TD></TR>
  <TR><TD>Maintainer:</TD><TD>Osamu Nakamura <osamu@wide.ad.jp></TD></TR>
  <TR><TD>Interface:</TD><TD>Serial1 (4)</TD></TR>
  <TR><TD>IP:</TD><TD>hoge.wide.ad.jp (AAA.BBB.CCC.DDD)</TD></TR>
  <TR><TD>Max Speed:</TD>
      <TD>192.0 kBytes/s (propPointToPointSerial)</TD></TR>
</TABLE>
```

図 2.2: cfgmaker の出力例

MRTG はデフォルトでは，ifInOctets と ifOutOctets をルータの SMNP サーバに対して読みに行く．MIB に載っているオブジェクトなら設定すれば全てグラフ化される．

2.4.4 収集データの保存

ルータの SNMP サーバから取り出したトラフィックなどのデータは，新しいものは精度を高く，古いものは精度を低くして保存している．1 日のデータは 5 分平均，1 週間のデータは 30 分平均，1 か月のデータは 2 時間平均，1 年のデータは 1 日平均で保存されるが，保存期間が過ぎると破棄される．このようにデータが圧縮されて保存されるので，データを保存されるファイルはある大きさ以上になることはない．

2.4.5　WIDE バックボーンにおける MRTG の利用

WIDE バックボーンでの MRTG の利用を図 2.3および図 2.4に示す.
MRTG で出力される HTML ファイルには,

```
<META HTTP-EQUIV="Refresh" CONTENT=300 >
```

というタグが記述され，Netscape Navigator などの WWW ブラウザを使用すれば，5 分毎に同じファイルを読み直すので，ほぼリアルタイムにトラフィックを確認できる.

2.4.6　考察

MRTG は，SNMP のトラフィックモニターおよび収集に関して非常によく考えられたツールである．従来 SNMP を利用したトラフィックのモニタリングツールは必要最低限のものしかなく，あっても command line で使用するもの，機能が限定されたもの，グラフィカルな表現が貧弱でわかりにくいものが多かった．この MRTG ではトラフィックデータの収集に関しては，SNMP というカウンターベースのデータの特徴を踏まえて，古いデータから順に平均化することでデータを圧縮し，データを必要以上に保存することを避けている．SNMP からトラフィックを収集あるいはモニターする場合，新しいデータは短い時間間隔で収集し，古いデータに関しては平均などによってデータ数を削減するが過去の傾向はわかるように保存して収集することが重要である.

実際にトラフィック収集そして解析を行う場合，MRTG のデフォルト設定である転送バイト数だけを収集するだけでは回線の状況を把握することは難しい．MIB にある様々なデータの種類のうちどのようなデータが，トラフィックの解析のみならず今後の運用，回線容量，速度の変更を決定するのに有用であるかという議論はなされておらず，今後の課題である．例えば，パケットロス (`ifInDiscards`，`ifOutDiscards`) や MRTG での ping roundtrip の収集が重要であると考えている．我々は今後もこの MRTG を評価しながら利用し，SNMP で利用できるデータを収集分析しつつ，必要なデータの種類を比較検討し，SNMP でのトラフィック収集および解析を行う予定である.

2.5　まとめ

以上，ネットワークトラフィック収集の現状とその問題点について述べ，SNMP を利用したトラフィック収集の特徴とその手法について述べた．また，我々が作成した簡易ツールと MRTG というパッケージについて述べ，それぞれを評価した.

SNMP を利用することにより，これまで NNStat で収集できなかった Ether Switch を利用している NOC セグメントや，高速なバックボーンの回線に対しても，容易にトラフィックを収集できることが明らかになった．また，MRTG の利用により，SNMP で収集された

データを管理者のみならずエンドユーザも WWW を通じてほぼリアルタイムにネットワークトラフィックをモニターできるようになった.

今後は，SNMP で取得できる様々なデータを収集しながら，ネットワークの状態をより明確に把握するために，有用なデータを比較検討し，SNMP でのネットワークトラフィック収集および解析の環境整備を目指す.

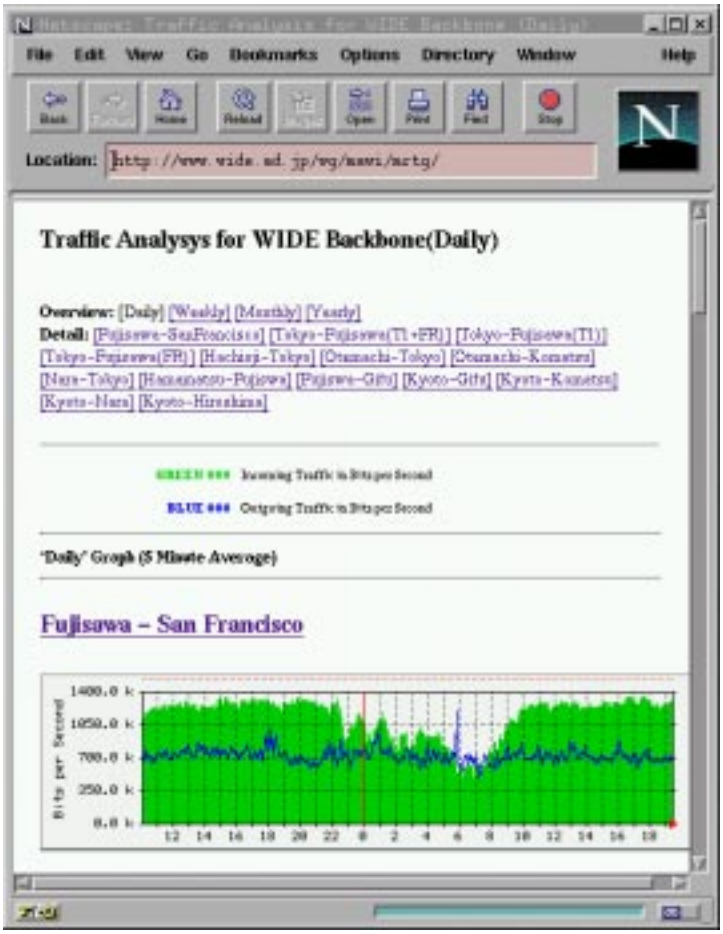

図 2.3: WIDE バックボーンでの MRTG(1)

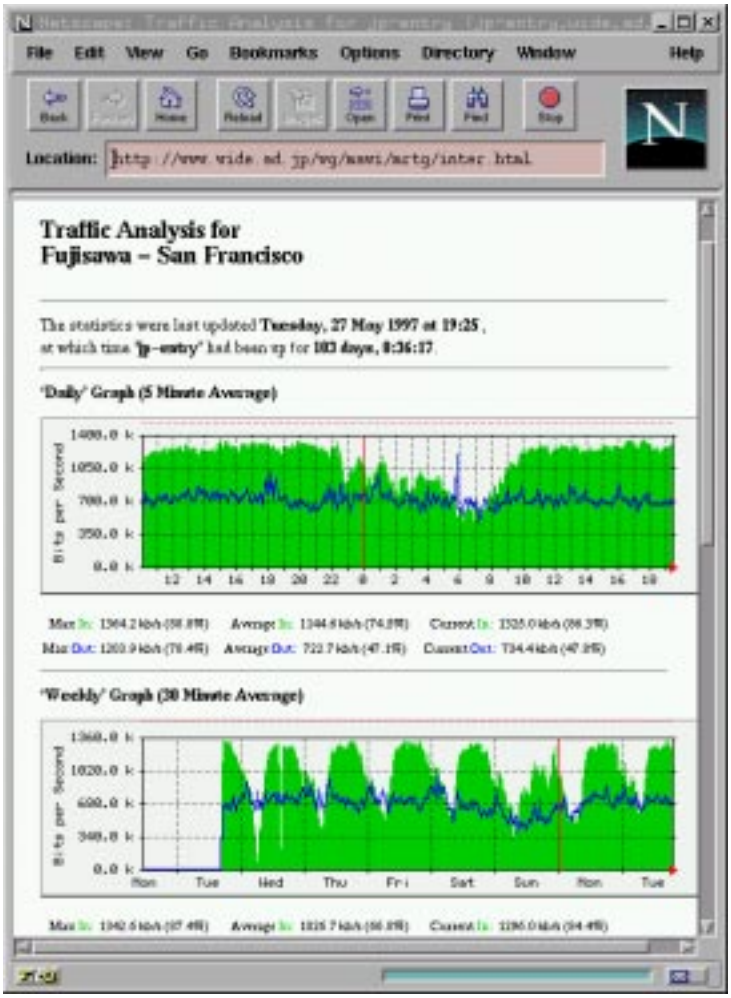

図 2.4: WIDE バックボーンでの MRTG(2)

第 3 章

国際線 TCP コネクションの統計的解析

3.1 データの収集および解析について

当 WG では、以下の条件でトラフィックデータを収集し、その統計的解析を行なった。

場所: WIDE 藤沢 NOC 〜 WIDE カリフォルニア NOC

日時: 97 年 5 月 27 日 19 時 40 分 〜 97 年 5 月 28 日 19 時 39 分 (24 時間)

データ: 通過する全パケットのヘッダ部分 40 バイトを収集

収集を行なった地点のネットワーク構成は、図 3.1 の通りである。

以下では、この収集データをもとに TCP コネクションの統計的解析を行ない、顕著な特性の見られた HTTP コネクションについて理論的分布を作成して、実際の分布との違いに関する検証/考察を行なう。

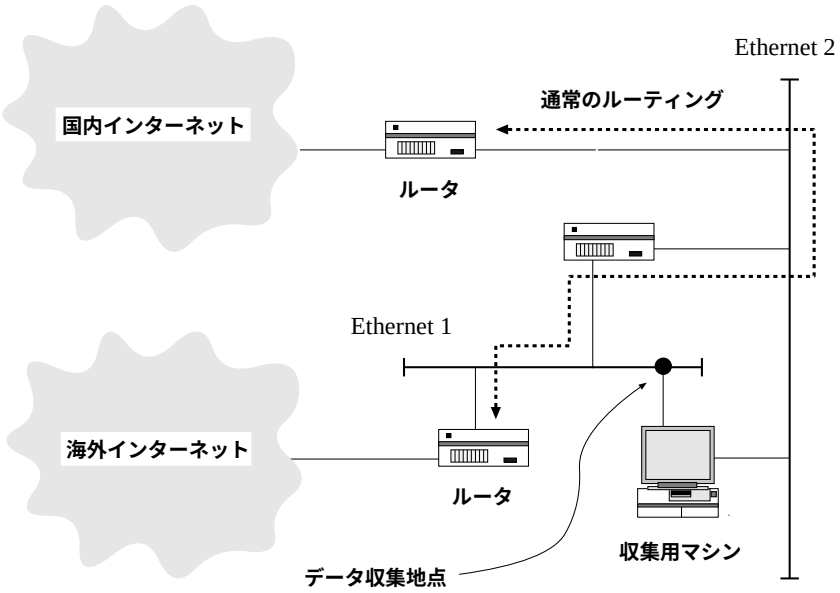

図 3.1: ネットワーク構成

3.2 各サービスごとの TCP コネクションに関する解析

今回収集した全データ総量は、以下の通りであった。

全パケット数		1.88×10^7 (パケット)[1]
TCP	パケット数	1.50×10^7 (パケット)[2]
	コネクション数	6.67×10^5 (コネクション)[3]
	データサイズ	3.36 (ギガバイト)[4]
UDP	パケット数	3.11×10^6 (パケット)[5]
	データサイズ	262.6 (メガバイト)[6]

また、TCP コネクションの時間帯別発生数は、図 3.2 に示すようなグラフになった。このグラフは、毎分のコネクション発生数を 3 分ごとに平均したものである。個人ユーザによるダイアルアップ IP を中心とした商用プロバイダの回線とは異なり、16:00〜21:00 頃が通信のピークとなっている。

[1] 18,772,296 パケット
[2] 14,992,922 パケット
[3] 666,516 コネクション
[4] 3,608,407,127 バイト (各ヘッダを含まず)
[5] 3,105,454 パケット
[6] 275,368,782 バイト (各ヘッダを含まず)

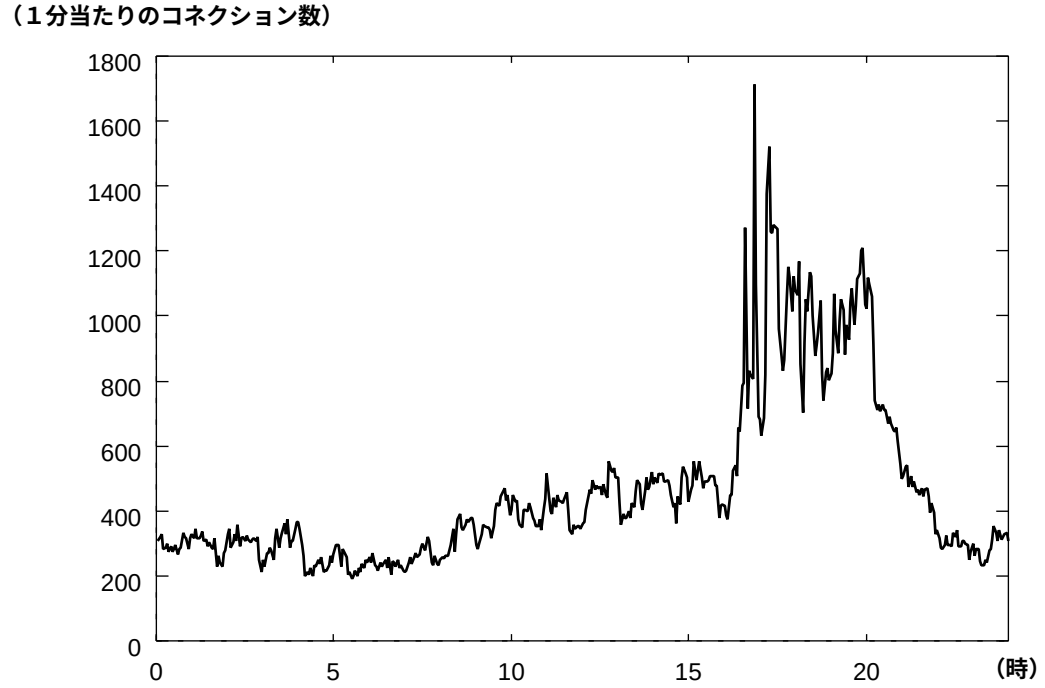

図 3.2: 時間帯別コネクション発生数

次に、収集したデータを各サービスごとに分類し、TCP の全通信のうち、それぞれのサービスがどの程度の通信を行なっているかを解析する。

まず最初に、それぞれのサービスでどの程度のコネクションが発生しているかを解析した結果は、以下の通りであった。

TCP コネクション	666,516（コネクション）	100 %
HTTP	531,940	79.8 %
FTP	5,731	0.9 %
TELNET	1,258	0.2 %
SMTP, NNTP	38,174	5.7 %
その他	89,413	13.4 %

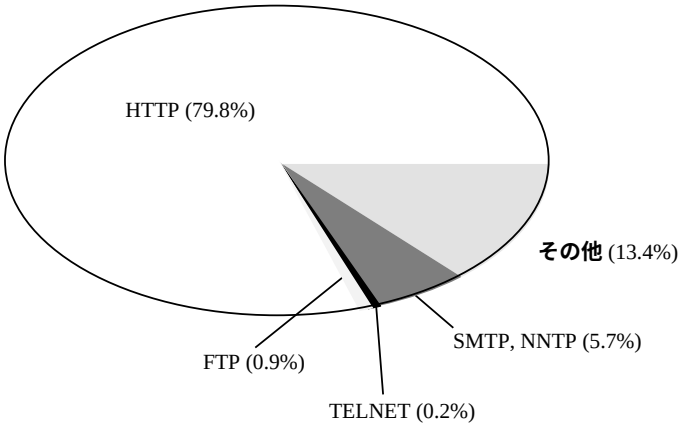

図 3.3: 各サービスごとのコネクション発生数の比率

FTP に関しては、ポート番号 20 番 (データポート) および 21 番 (コントロールポート) を用いて行なわれた通信のみを FTP の欄に計上し、パッシブ転送により行なわれた通信は、その他の欄に計上してある (以下の解析でも全て同様)。以上の結果より、発生しているTCP コネクションのうち、約 8 割のコネクションが HTTP コネクションであることがわかる。

次に、各サービスでやり取りされたパケット数を解析した結果は、以下の通りであった。

全パケット	18,772,296 (パケット)	
TCP パケット	14,992,922	100 % (全パケット比 79.9%)
HTTP	10,234,327	68.3 %
FTP	749,948	5.0 %
TELNET	103,628	0.7 %
SMTP, NNTP	1,066,152	7.1 %
その他	2,838,867	18.9 %

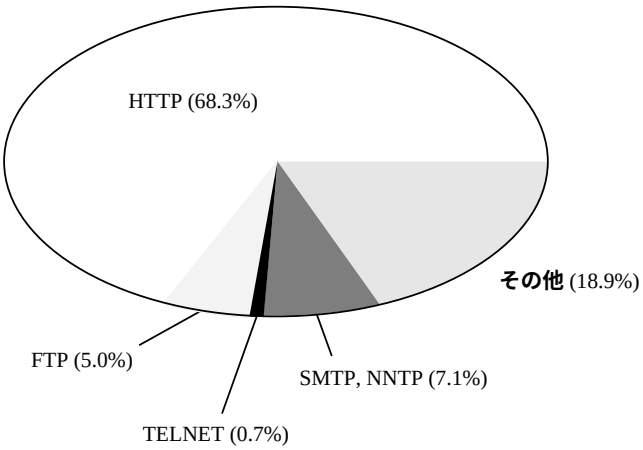

図 3.4: 各サービスごとのパケット数の比率

また、各サービスでやり取りされたデータ量は、以下の通りであった。

TCP データ	3,608,407,127（バイト）	100 %
HTTP	2,462,237,894	68.2 %
FTP	183,608,383	5.1 %
TELNET	4,565,357	0.1 %
SMTP, NNTP	189,550,019	5.3 %
その他	768,445,474	21.3 %

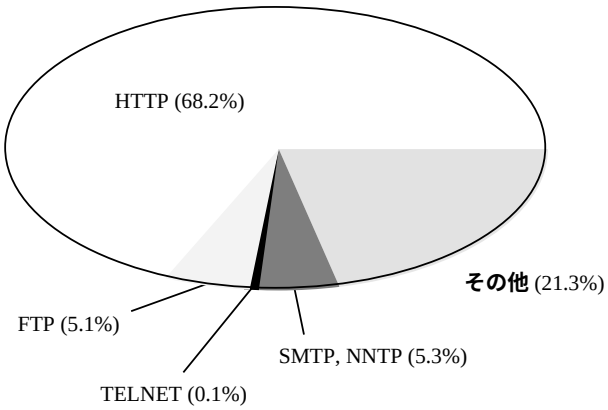

図 3.5: 各サービスごとの送信データサイズの比率

最後に、同時に開設される有効コネクション数の分布を解析した結果を図 3.6 に示す。この解析では、120 秒を越えたコネクションは、何らかの原因によりコネクションが途切れたものとして計上してある。

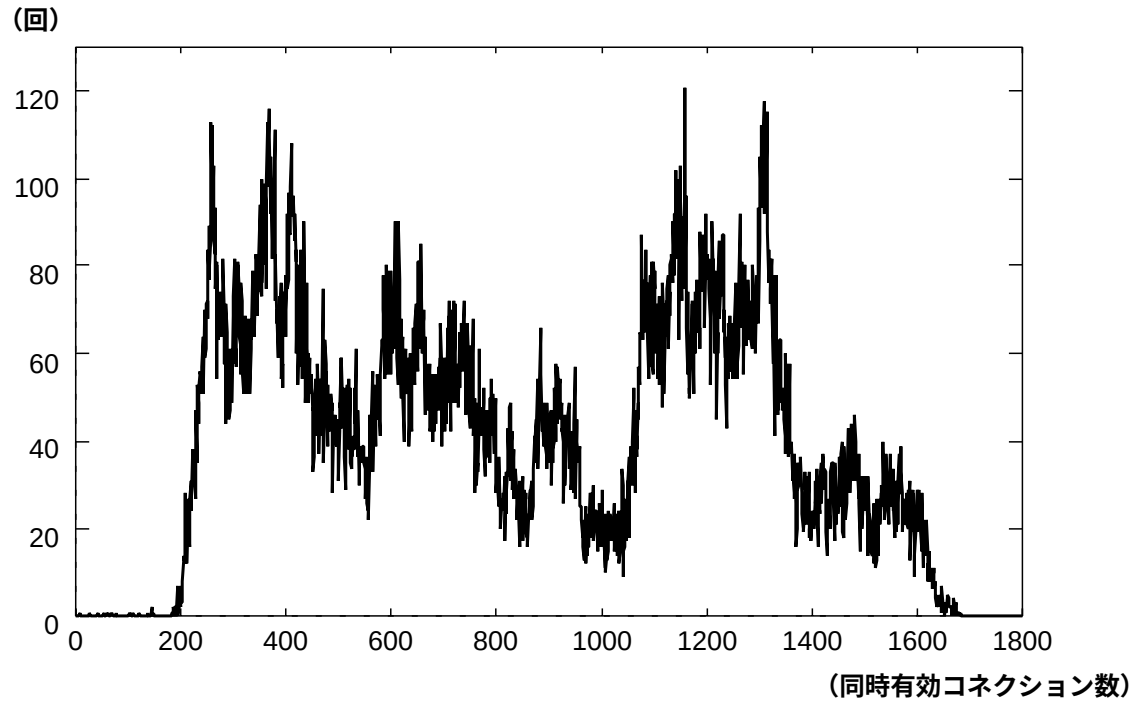

図 3.6: 同時コネクション数の分布

以上より、今回測定した TCP コネクションにおいては、全コネクション数/送受信データ量の実に 7 割〜8 割が HTTP によるものであることがわかった。また、HTTP による通信では、発生したコネクション数と比較して、やり取りされるデータ量が少ないことも指摘できる。これは、HTTP の利用形態から判断して、FTP による通信と比べて通信先を次々と切替えながら (つまり、コネクションを次々と張り替えながら) 通信を行なっていることが原因だと予想される。

これに関連して、図 3.6 より、最大で 2000 本近くのコネクションが同時に開設されていることもわかった[7]。

[7] 今回時間切れ (120 秒以上) として計上したコネクションのうち、何割かはまだ有効なコネクションだと考えると、同時開設コネクション数はさらに増大する。

次の節では、このような HTTP コネクションに関して、もう少し詳しく解析を行ない、解析より得られた HTTP コネクションの振舞いを元に、ネットワーク全体に与えている負荷などについて考察を行なう。

3.3　HTTP コネクションに関する解析

この節では、HTTP コネクションにおける送信データサイズの分布をもとに、コネクション時間の理論的分布を求め、実際の分布と比較してみる。

HTTP コネクションでやり取りされたデータサイズの分布は、図 3.7 のようであった。

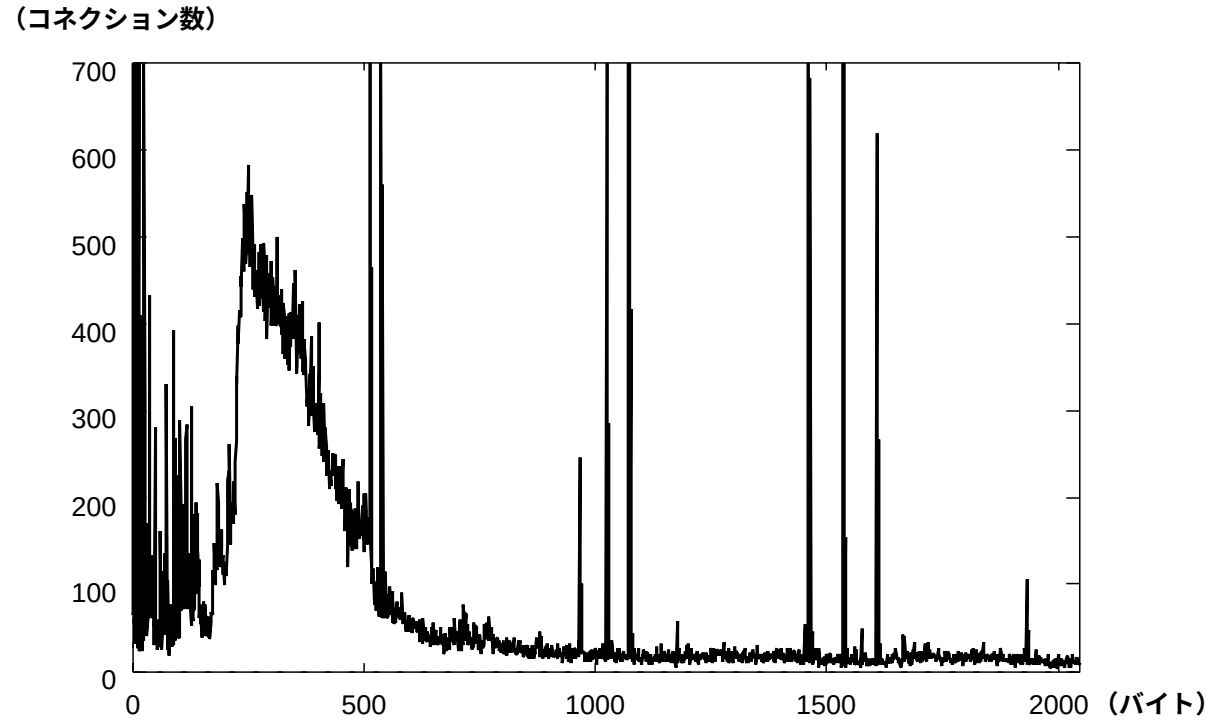

図 3.7: HTTP コネクションの送信データサイズの分布

上記の分布グラフのうち、上方にはみ出している部分は、特にコネクション数の多かったデータサイズである。中でも、全くデータをやり取りしていないコネクション (送信データサイズが 0 バイトのもの) は桁違いに多く、全 HTTP コネクションの半数近く (約 16 万コネクション強) であった。この 0 バイト-コネクションについては、後の節で再び取り上げる。

前記の分布をもとに、コネクション時間の理論的分布を求める。モデルを簡単にするために、ある短い時間間隔 Q ごとの状態を考える。まず、コネクションが、時間 Q の間に終了する確率をσとして、1 回のコネクション時間が次のような幾何分布に従うとする。

$$g_i = \sigma^{i-1}\,(1-\sigma) \quad (0 < \sigma < 1,\ i = 1, 2, ...)$$

また、パケットの処理に必要な時間は、パケットの伝送に必要な時間の 100 倍と仮定し[8]、さらにネットワークの帯域を 38.4kbps と仮定して、各コネクションでは等間隔でパケットが送信されることとする。

以上の仮定のもと、次のような $W_k(j)$、P_jを考えると、

$W_k(j)$: システム内に j 本のコネクションが存在している状態で、新たに接続時間が k だけ必要なコネクションが発生した場合、そのコネクションが実際に費す時間の期待値

P_j : j 本のコネクションが発生する確率

接続時間が k だけ必要なコネクションが、実際に費すことになるコネクション時間の理論値は、以下のように表わすことができる。

$$W_k = \sum_{j=0}^{\infty} P_j\ W_k(j) \quad \cdots\cdots\ (1)$$

以下では、この W_kを求めて、図 3.7 のグラフから得られたデータを適用することで、コネクション時間の理論的な分布を求める。

まず、$U_i(j)$ を、

$U_i(j)$: コネクション発生時にコネクションが j 個存在していた状態で、そのコネクションが、$i-1$ パケット目の送信から i パケット目の送信までに要する時間を表わす確率変数

と考えると、

$$W_k(j) = \sum_{i=1}^{k} E\,[U_i(j)] \quad \cdots\cdots\ (2)$$

となる。ここで、$U_i(j) = x\ \ (i \geq 2)$ とすると、$\frac{x}{Q} - 1$ は、$i-1$ 個目のパケットを出してから i 個目のパケットを出すまでに、他のコネクションから出されたパケットの数を表わしている[9]。この後、$i+1$ 個目のパケットを出すまでに、平均して$\sigma(\frac{x}{Q} - 1)$ 個のコネクションが終了し、平均してλx 個のコネクションが新たに発生している。よって、$i+1$ 個目のパケットが送信されるのに要する時間は、

[8]つまり、通信が 1 秒かかった場合、実際の伝送に掛かった時間は 10 ミリ秒で、残りの 990 ミリ秒はパケットの組立てや解体などのソフトウェア処理で消費していると仮定する。

[9]あるいは、i 個目のパケットを出す時点で存在しているコネクションの数。

$$\mathrm{E}\,[\mathrm{U}_{i+1}(\mathrm{j})] = \mathrm{Q}\,\{\sigma(\frac{x}{Q} - 1) + \lambda \mathrm{x} + 1\}$$
$$= (\lambda \mathrm{Q} + \sigma)\,\mathrm{E}\,[\mathrm{U}_i(\mathrm{j})] + \mathrm{Q}\,(1 - \sigma)$$
($\mathrm{U}_i(\mathrm{j}) = \mathrm{x}$ より $\mathrm{E}\,[\mathrm{U}_i(\mathrm{j})] = \mathrm{x}$ であることに注意)

となる。また、$\mathrm{E}\,[\mathrm{U}_2(\mathrm{j})]$, $\mathrm{E}\,[\mathrm{U}_2(\mathrm{j})]$ に関しては、以下のようになる。

$$\mathrm{E}\,[\mathrm{U}_2(\mathrm{j})] = \lambda \mathrm{Q} \cdot \mathrm{E}\,[\mathrm{U}_1(\mathrm{j})] + \mathrm{Q}\,(\sigma \mathrm{j} + 1) \cdots\cdots (3)$$

以上の差分方程式を解くと、$\mathrm{E}\,[\mathrm{U}_i(\mathrm{j})]$ は、次のようになる。

$$\mathrm{E}\,[\mathrm{U}_i(\mathrm{j})] = (\lambda \mathrm{Q} + \sigma)^{i-2}\,\mathrm{E}\,[\mathrm{U}_2(\mathrm{j})] + \mathrm{Q}\,(1 - \sigma)\,\frac{1 - (\lambda Q + \sigma)^{i-2}}{1 - (\lambda Q + \sigma)} \cdots\cdots (4)$$

(3), (4) を (2) に代入して、等比級数の和の公式を用いると、

$$\mathrm{W}_k(\mathrm{j}) = \mathrm{E}\,[\mathrm{U}_1(\mathrm{j})] + \mathrm{Q}\,\frac{k-1}{1-\rho} + \mathrm{Q}\,\{\lambda \mathrm{E}\,[\mathrm{U}_1(\mathrm{j})] + \sigma \mathrm{j} - \frac{\rho}{1-\rho}\}\frac{1-\alpha^{k-1}}{1-\alpha} \cdots\cdots (5)$$
ただし、$\alpha = \lambda \mathrm{Q} + \sigma, \quad \rho = \dfrac{\lambda Q}{1-\sigma}$

となる。(1), (2) より、

$$\mathrm{W}_1 = \sum_{j=0}^{\infty} \mathrm{P}_j\,\mathrm{W}_1(\mathrm{j}) = \sum_{j=0}^{\infty} \mathrm{P}_j \sum_{i=1}^{1} \mathrm{E}\,[\mathrm{U}_i(\mathrm{j})] = \sum_{j=0}^{\infty} \mathrm{P}_j\,\mathrm{E}\,[\mathrm{U}_1(\mathrm{j})] \cdots\cdots (6)$$

に注意して、(5), (6) を (1) に代入すると、結局 W_kは

$$\mathrm{W}_k(\mathrm{j}) = \mathrm{W}_1 + \mathrm{Q}\,\frac{k-1}{1-\rho} + \mathrm{Q}\,\{\lambda \mathrm{W}_1 + \sigma \bar{n} - \frac{\rho}{1-\rho}\}\frac{1-\alpha^{k-1}}{1-\alpha} \quad (\mathrm{k} \geq 2)$$

ただし、$\bar{n} = \sum_{j=0}^{\infty} \mathrm{j}\mathrm{P}_j$　　(P_jの平均)

となる。

ここで、前述の P_jの定義より、$\bar{n}$ (P_jの平均) の意味を考えてみる。コネクションの発生間隔がポアソン分布、コネクションの接続時間が指数分布の時の、ある瞬間のコネクション数を求めるということは、M / M / 1 型待ち行列における列の長さを求めることに等しい。したがって、$\bar{n}$は、M / M / 1 型待ち行列における列の長さの平均に等しく、

$$\bar{n} = \rho + \frac{\rho^2(1+\sigma)}{2(1-\rho)}$$

となる。また、W_1は、

$$W_1 = \frac{1-\rho}{2} Q + \bar{n}Q$$

となる。

以上より、送信されたデータサイズから求めたコネクション時間の理論的分布と、実際のコネクション時間の分布とを図 3.8 に表わす。単位時間間隔 Q を 10 ミリ秒として、また、時間間隔 Q (つまり 10 ミリ秒間) にコネクションが終了する確率σおよびコネクションが発生する確率λを、収集データを元に$\sigma = 6.8 \times 10^{-5}$, $\lambda = 0.3$ とした。

この比較により、理論分布と比べて実際の分布の方が、裾野の長い分布であることがわかる。理論分布は、送信されるデータサイズの分布より求めたものであり、したがって、実際のコネクションの長さは、データサイズから想定される長さよりも長いものが多いと指摘できる。

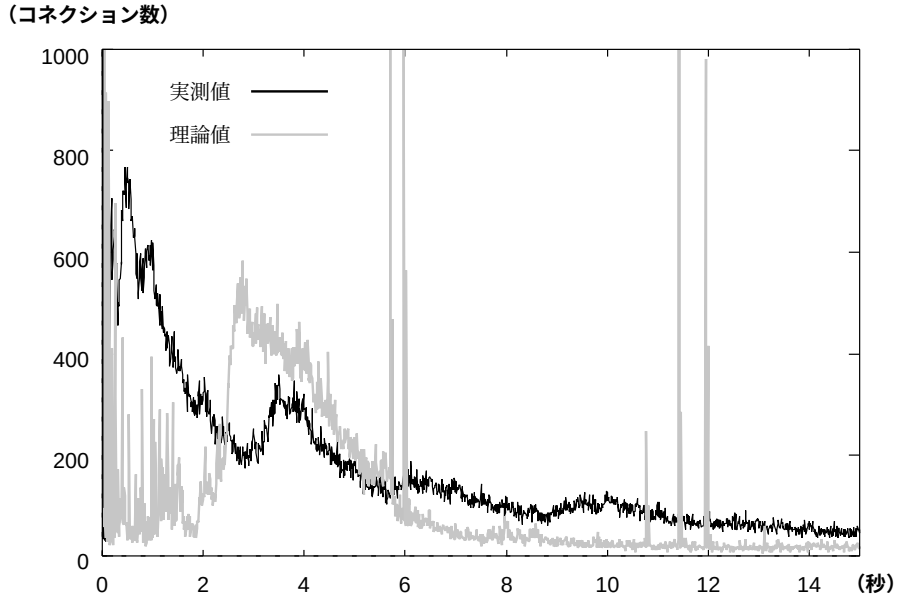

図 3.8: HTTP コネクション時間の理論値と実測値との比較

インターネットにおける通信は Best-Effort で行なわれており、そのコネクション時間は、単純にデータサイズから想定される時間と比べて、ランダムであると言える。次では、それぞれの通信がどの程度のランダム性を持っているかを検証してみる。最初は、前述のコネクションモデルを元に、データサイズをランダム要素とした理論値と実際のコネクショ

ン時間との分布を比較してみる。

まず、最もランダム性が高い場合として、指数分布との比較を行なってみる。解析の対象とした 347,022 コネクションのデータサイズの平均値が 251.4 バイトであることから、その分布を以下のような$\lambda = 0.0040$ の指数分布に近似した後に求めた理論値との比較を、図 3.9 に表わす。

$$f(x) = n\lambda \cdot e^{-\lambda x}$$

ただし n = 347,022, $\lambda = 0.0016$

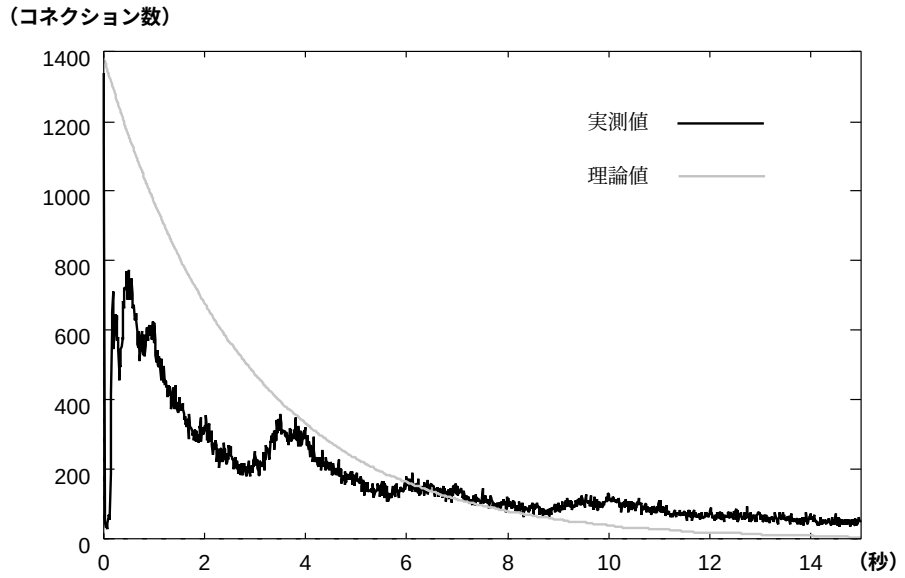

図 3.9: ファイルサイズの分布を指数分布に近似した場合の比較

次に、指数分布と比較してランダム性の低い場合として、データサイズの分布を以下のような次数 2 のアーラン分布に近似して求めた理論値との比較を、図 3.10 に表わす。

$$g(x) = n\,\frac{(\lambda k)^k x^{k-1}}{(k-1)\,!}\,e^{-\lambda k x}$$

ただし n = 347,022, $\lambda = 0.0016$, k = 2

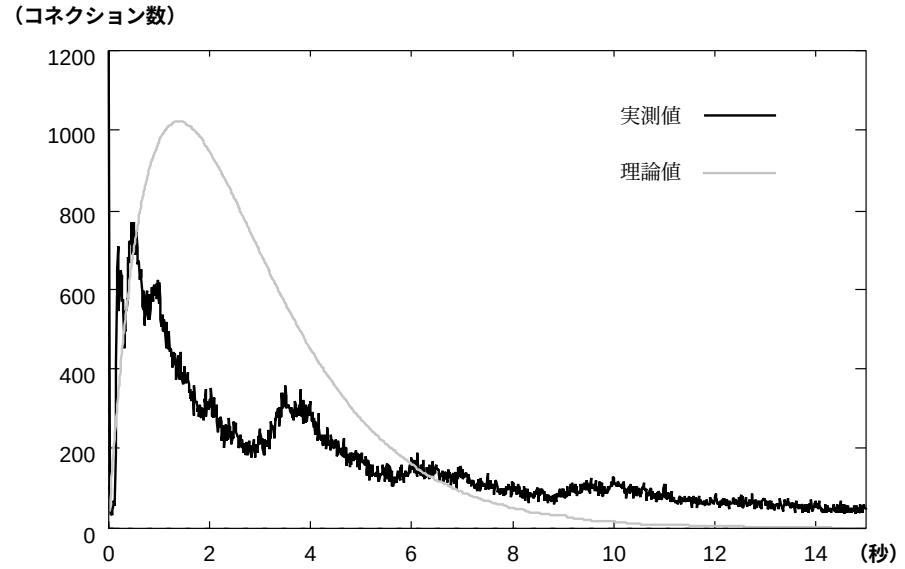

図 3.10: ファイルサイズの分布を次数 2 のアーラン分布に近似した場合の比較

これらより、HTTP のコネクション時間の分布はランダム性が高く、データサイズなどから予め想定することがほとんど困難であることが指摘できる。

コネクション時間とデータサイズとの間に、どのような関係が存在するかを調べるために、各コネクションごとにやり取りされるデータサイズとコネクション時間の相関を調べてみる。やり取りするデータサイズを変化させた場合の相関係数を計算し、その変化をグラフにして図 3.11 に表わす。

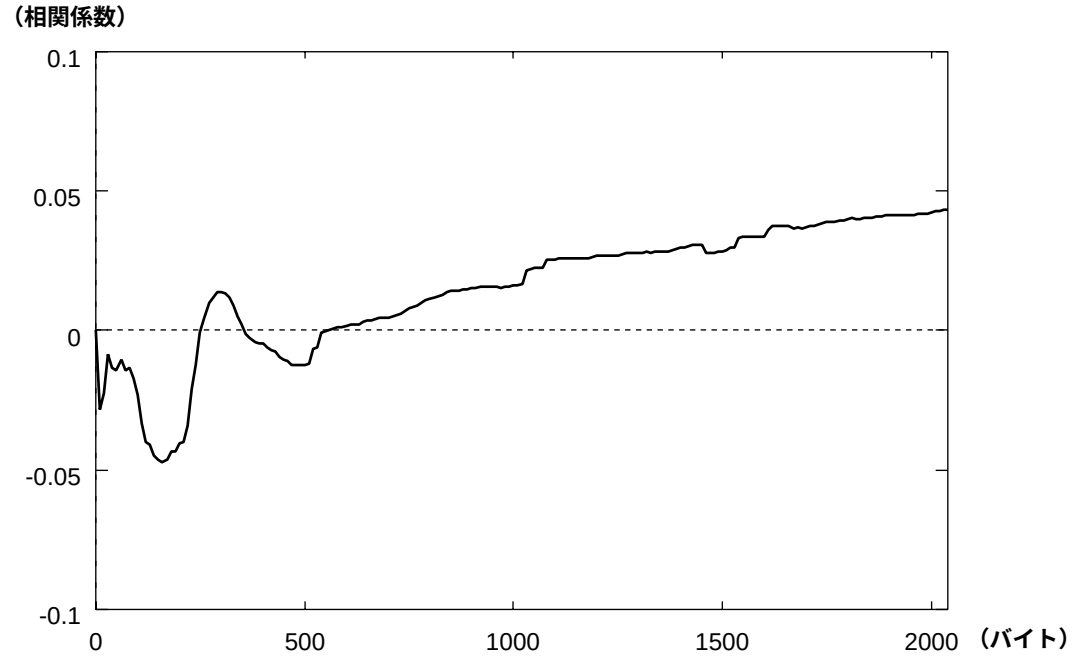

図 3.11: データサイズとコネクション時間との相関

図 3.11 より、次のような事実を指摘することができる。

- 1 回のコネクションで送信するデータが特に約 500 バイトより小さい場合は、送信するデータが少ないからといってコネクション時間が短いとは限らない。
- しかしながら、相関係数は全て−0.05〜0.05 の間に収まっており、送信するデータサイズとコネクション時間の長さとの間には、基本的にほとんど相関が見られないと言える。

次の節において、その原因を考察する。

3.4　0 バイト-コネクションに関する解析

前節で述べたように、HTTP コネクションでは、その約半数が何のデータもやり取りしていない 0 バイト-コネクションになっており、ネットワークに対する負荷を増大させている。本節では、この 0 バイト-コネクションについて解析し、その原因を考察する。

今回の解析により、HTTP コネクションでは、0 バイト-コネクションの他にも次のような特徴が存在していることがわかった。

- 多くのコネクションは、RST により終了されている。
  FIN を用いて Graceful-Close を行なっているコネクションと、RST によりいきなり終了されているコネクションとの比率は、以下の通りであった。

終了を確認できたコネクション	77,149（コネクション）	100 %
FIN で終了したコネクション	17,558	22.8 %
RST で終了したコネクション	59,591	77.2 %

  また、FIN を用いて終了したコネクションと、RST により終了したコネクションで、それぞれやり取りされたデータサイズの分布を図 3.12 に示す。図 3.12 では、FIN により終了したコネクション数と、RST により終了したコネクション数とが同数になるように補正してある。これより、RST により終了されるコネクションはその大

半が 500 バイト以下であり、特に 1 キロバイトを越えるコネクションはあまり存在しないことがわかる。

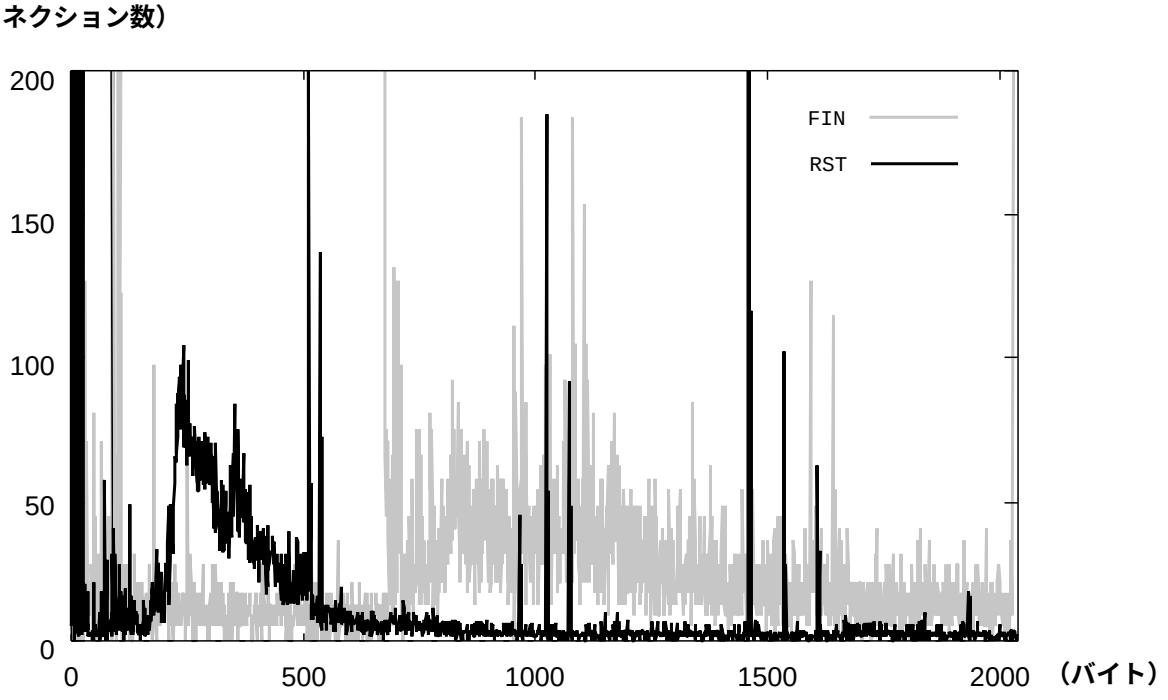

図 3.12: FIN により終了した接続と RST により終了した接続のデータサイズ分布

- やり取りされる HTTP パケットにおいては、データサイズが 0 である ACK の占める割合が非常に多い。
  全 HTTP パケットに対する 0 サイズ ACK の占める割合は、以下の通りであった。

全 HTTP パケット	10,234,327 (パケット)	100 %
0 サイズ ACK	4,757,013	46.5 %

上記より、0 バイト-コネクションでは、TCP の制御用パケット (特に ACK) のみがやり取りされていると考えられるが、TCP コネクションで 0 サイズ ACK の送信が増える場合としては、以下のような場合を挙げることができる。

1. パケットの到着間隔が伸びてしまう場合。
   例えば、あるパケットが到着した後、次のパケットが到着する前に Delayed ACK のタイムアウトが来てしまうような状態が続くと、各パケットごとにそれぞれ ACK

を送ることになってしまう。

2. データパケットのやり取りに比べて、制御用パケットのやり取りが多くなる場合。例えば、1 コネクション当たりでやり取りするデータパケット数が、少なくなれば少なくなる程、ACK パケットの占める割合は多くなる。

上記 1. については、確かに ACK を増加させる要因として挙げることは可能であるが、データをやり取りした際の ACK が影響を及ぼしている場合も想定されるので、これが直接 0 バイト-コネクションを増大させる原因だと考えることはできない。

次に、上記 2. を検証するために、1 回の HTTP コネクションでやり取りされるパケット数の分布を求めてみる。

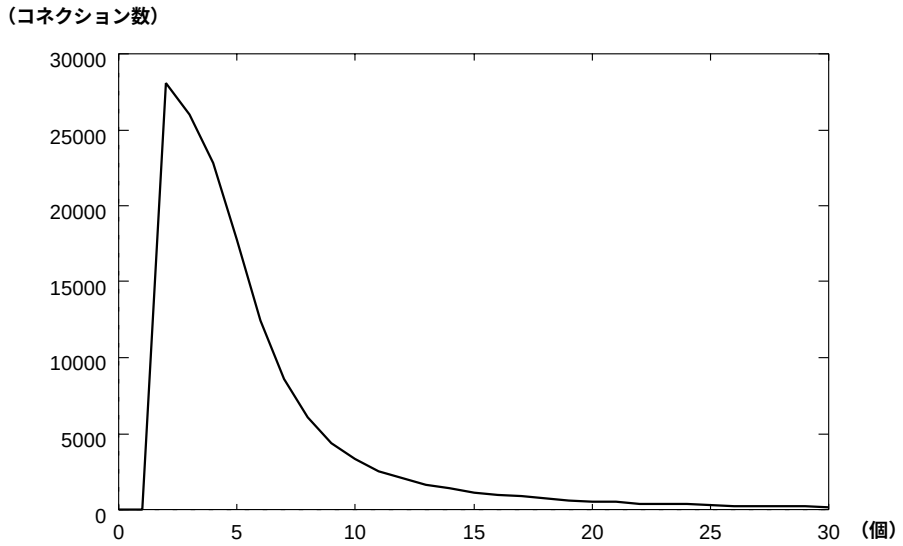

図 3.13: 1 コネクション当たりのパケット数

図 3.13 より、HTTP コネクションでは、1 コネクション当たりのパケット数は (ACK パケットや SYN、FIN、RST パケットなどの制御用パケットも含めて) その大半が 10 個以下であることがわかる。これより、各コネクションでは実際にデータをやり取りするためのパケットに比べて、コネクションの制御のためのパケットの比率が相対的に大きくなり、通信効率を悪化させていると言える。

また、パケット数が 5 個以下のコネクションが多数を占めているが、通常の TCP の接続/終了手順を考えると、パケットを 3 個程度しかやり取りしないコネクションとしては、コネクションの開設時に SYN を送信した後、ACK のやり取りで通信が途切れてしまい、RST で通信を終了させている場合を挙げることができる。HTTP の利用形態を考えてみる

と、これは、ブラウザ上で新しい Web サーバへの接続を試みたもののうまく接続できず、再接続を重ねる割合が高いという状況を表わしているものと思われる。

3.5 まとめ

TCP コネクションのうち、顕著な性質が見られた HTTP コネクションに関して、その送信データサイズとコネクション時間との関係を中心にして、解析を行なった。

今回、解析の対象とした HTTP コネクションは、その利用形態より、やり取りするデータサイズに比べて多くのコネクションを開設して通信を行なっている。HTTP コネクションでは、次々とコネクションを張り替えながら通信を行なっているために、他のサービスに比べてネットワークに対する負荷を増大させている。特に 0 サイズ-コネクションの多さは圧倒的であり、全 HTTP コネクションの半数近くを占めている。これにより、今回の解析では、最大で 2000 本程度のコネクションが同時に開設されていることがわかった。これら、ネットワークに対する負荷により、送信するデータサイズとコネクション長との間にはほとんど相関が見られず、コネクション時間を前もって予測することは全く不可能であることがわかった。

0 サイズ-コネクションが発生する状況としては、コネクションの開設時に SYN を送信した後の ACK のやり取りで通信が途切れてしまい、RST で通信を終了させている場合、つまりブラウザ上で新しい Web サーバへの接続を試みたもののうまく接続できず、再接続を重ねているような場合を挙げることができる。

効率的な通信 (例えばスイッチングなど) を行なうためには、今後 0 サイズ- コネクションの数を減らす必要があるが、今回の解析を通して、次のような提案が有効だと思われる。つまり、 0 サイズ-コネクションの最大の原因は、ブラウザを利用したインターネット-サーフィンによる頻繁なコネクションの張り替えであると考えられ、Web サーバを渡り歩いてもコネクションを張り替える必要がないプロキシサーバの導入などは、最も効果的な対策だと言える。

第 4 章

まとめと今後の課題

本報告では 1996 年度に収集されたトラフィックデータの解析結果に基づき WIDE インターネット上のトラフィックに付いての考察を行なった。とくに、SNMP を用いて収集可能なデータの可視かのツールの評価や、WIDE の国際線上を流れる TCP データに注目し、国際線がどのように利用されているのかの考察を行なった。

今後は新しい統計収集方法をバックボーン全体に広げていく。またバックボーン回線が高速になるにしたがい全てのトラフィックを収集して解析すると言う方法が、ルータに与える負荷の問題などから現実的ではなくなりつつある。そろそろ NSFNET バックボーンで行なわれていたような、サンプリングによる統計処理を行なう手法を確立していく必要がある。

だが、それを行なう前に今回行なったような全てのパケットヘッダの収集・解析を行ない、インターネットトラフィックの統計的なモデリングを行なうことを考えないと、サンプリングをどのように行なえば、サンプルされたデータに対する解析が、トラフィック全体の性質を代表しているのかどうかを検証することができない。

この全てのパケットヘッダを収集することは、ネットワークリンクの高速化とネットワークトポロジの複雑さにより、かなり困難な作業ということができ、実際のネットワーク機器とは別立てでトラフィック収集解析用のシステムを考えて行く必要があるだろう。

今後はこのような高速ネットワーク環境で、データロス無くパケットヘッダの収集解析を行なう為のシステムの構築を考えて行きたい。