TCP 制御フラグによる
輻輳検知方法のプロトコル別検証

s013051
関本　啓介
計算機科学講座
田中研究室

2004 年 2 月 18 日

目次

第１章　序論

インターネット上の輻輳制御をおこなうのに、ネットワーク自身には自立的な輻輳回避技術が存在しないためエンドノード間での輻輳回避をする必要がある。しかし、

- エンドノードはネットワークの全体の状態がわからない
- 上位プロトコルからでは通信媒体の性質が見えにくい
- ネットワークの許容量が不明
- ネットワークの混雑度が不明

などの理由で輻輳の検知・制御は大部分がハードウェアによって行われており、ソフトウェアでの検知というのはあまり行われてはいない。

表１　関連のある単語による検索結果

検索用語	Congestion control	Variable Rate control	Adaptive control
件数	9013	40	23625
検索用語	Protocol	TCP/IP friendly	Congestion detection
件数	0	0	35

（注）これらの検索結果は INSPEC を使用したものである[1]。また video streaming に関する研究を除外しており、音声ストリーミングおよびネットワークの制御に関する検索結果となっている。**表１**は検索ワードとして
((**検索用語**)WN CV) NOT (((video) WN All) AND ((streaming) WN All)), 1990-2005
をもとに検索した結果である。

しかし、昨年当研究室で TCP 制御フラグの変化を見ることでネットワーク負荷時の輻輳状態の検知に応用すると言った、**表１**の Congestion detection のサブ集合などには記載のない従来のやり方とは違った方法が考えられた[1]。それは従来のように送受信できそうなパケット量を予想する方法と比べると、輻輳の程度を知ることができるため送受信可能な量ではなく輻輳が起きにくいパケットの送受信量を知ることができるというものであった。結果を見るとそのまま実用できる精度ではなかったが、確かに通信量との相関が見られたため、本研究では実用度を検証するためにさらにプロトコル別に実験を進める。

[1] 2005 年 02 月 10 日 現在

第２章　輻輳検知について

２．１　輻輳検知の必要性

輻輳とは、方々からいろいろな物が 1 か所に集まることや、混み合うことを指す言葉である[2]。通信業界では、大勢の人が同時に回線を利用するなどでトラフィックが増え、ネットワークが混雑して渋滞することを意味する。チケット予約の際などに電話がかかりにくくなる現象は、まさにこの輻輳が起きている。ネットワークにおいては、ネットワーク間のトラフィックの増大によって、情報の流れに遅延や損失がおこったり、新たにネットワークに参加できないという問題が起きる。さらに、情報の流れの遅延により情報の再送要求が大量に出され、それによってさらにネットワークの状態が悪くなるという悪循環が起きる。このネットワークの混雑を回避するためにはいち早い輻輳の検知が必要であり、それによって通信量を制御するという事柄につながる。そのために現在はさまざまな輻輳の検知手法が考えられている。

２．２　従来の方法

ここで、ネットワークの混雑に対応するために現在行われている、輻輳に関する手法の一例を紹介する[2][3][4] [5]。

[ネットワークの状態推測機構]

ネットワークの情報は直接手に入らないのでエンド間の通信から得られる情報で推察する。単純なアルゴリズムとしてはパケットの受信数が落ちないなら転送量をあげそうでないと下げるというものがある。

[スロースタート]

通信を開始するときに控えめに転送を初め、そして徐々に転送速度を上げていく。長所としてはネットワークの突発的なトラフィック流入を避けることができるが、その反面短所として、転送速度が収束するのに時間がかかる。

この時、輻輳が起きる直前のウィンドウサイズをA、輻輳が起きたときのウィンドウサイズをBとすると、最適なウィンドウサイズXは

$$A < X < B$$

と推測される。ここではウィンドウサイズをいったんおとして、再度増加させながらXを探る。

---

[2] 参考：http://www.interq.or.jp/blue/rhf333/OVER_L.htm

[ＲＴＴの相関関係を利用]

ＲＴＴ（Round Trip Time）からわかることは、輻輳の度合い。パケットロスの指標である。ＲＴＴから得られる指標を平滑化することによってＲＴＴを評価し（ＲＴＴ評価値）輻輳を検出する。ＲＴＴはネットワークのトラフィック量に従って時間とともに変化するので常に適切な値に補正しなければならない。ここではＡＣＫの送信から受信までにかかる時間を測定する式を示す。

ＲＴＴ評価式（ＲＦＣ７９３準拠）

Ｒ　＝　αＲ＋（１－α）Ｍ　……（１）

Ｒ：ＲＴＴ評価値

Ｍ：新しい測定値

α：平滑化係数（推奨値は０．９）

[回線利用率からの推測]

ネットワーク間における、ネットワークトラフィックから利用率を測定し輻輳状態の推測をおこなう。回線利用率は以下の式で計算される。

回線利用率　＝　１時間に発生する総データ量（Bit）/　回線容量（B）×　時間　……（２）

これらの先行研究では、ソフトウェアレベルでの研究ではなくてハードウェアレベルでの研究を元にしている。本研究での検知法は、本来エンドノード自体では検出しにくい輻輳をソフトウェアによって検出するものであり、その実用化を試行していきたいと思う。

２．３　樋上による検知法の原理

（文献：TCP 制御フラグを使った輻輳検知に関する研究）

本研究で検証するアルゴリズムについての説明をする。
まず、TCP ヘッダは**図１**のような構造をしている。このヘッダの中には、フロー制御のためのフラグが６種類含まれており、その中の１つである ACK（Acknowledge）フラグと応答番号をモニターすることで輻輳を検知する。

図 1

URG	緊急データを含む	ACK	応答確認番号が有効
PSH	データを上位層に渡すよう要求	RST	コネクションの一方的切断
SYN	コネクションの開始	FIN	コネクションの終了

TCP はパケットの送受信に対して制御を行うプロトコルである。その制御情報の変化から、ネットワーク上の輻輳情報を取りだせないかというものである。

具体的な例をあげると**図２**のようになる。受信側は届いていないデータがある場合、送信側から送られてくるデータにかかわらず、再送してもらえるまで損失したパケット番号を ACK フラグ付きで送り続けるのである。

つまり損失が頻繁に発生したり、また回線が混雑して再送が滞ればそれだけ多くの再送要求が出されるのである。
以下、再送要求のために発信される、同じ番号を送り続ける ACK フラグ付きパケットのことを便宜上「**重複 ACK**」と呼ぶことにする。

樋上による手法は、この重複 ACK と通信量の間にある相関関係を見ることで、本来ならネットワーク全体の様子が解りづらいエンドノード単体でも輻輳を検知できるようにするものである。

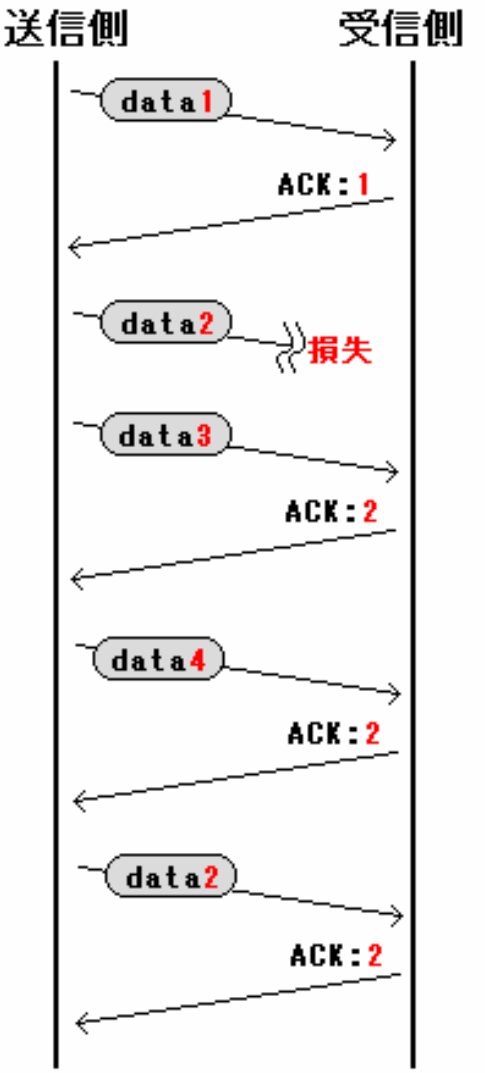


図 2

第３章　詳細な検証実験の必要性

３．１　先行研究での不足点

本研究の元となった論文では、ネットワークの負荷に IP Load（© BTT SoftWare）を用いている。これはネットワークのテストツールであるが、設定したパケットを設定した間隔で送り続けるだけであり、サーバーの応答は考慮されていない。また通信開始直後から常に同じ内容の通信を続けるというのは現実にはあり得ない状況である。

また実験環境において、クライアント－サーバー間にモニターが設置されていないため、実際の通信とキャプチャーされたデータとの差違を確認，検証することができない。つまり重複 ACK そのものが損失する割合や、それによる予測値と実測値の誤差などが解らず、推測の域に留まっている。

データの送信先ポートが、その時に開いているポートを適当に選んでいるため実際にはどのような通信時を想定しているのか、どのような場合に応用できるのかが不明確である。またポートの違いによる受信側レスポンスの検証が行われていないため、実験データ間の結果に影響がでている可能性がある。

比較対象としている RTT 測定の手順が明記されていない。

３．２　追加実験内容

上記の不足点を補うために、本研究では以下の条件で実験を行う。

・通常よく使用されると思われる HTTP、FTP、POP、SMTP の４つのプロトコルについて検証を行う。これによりポートの固定、実環境との比較をする。
・それぞれ、クライアント自身で測定するものと第三者による観測付きの２パターンを試みる。重複 ACK 自体の損失が結果に与える影響を見る。
・ping による RTT の測定を、重複 ACK の実験と同時うことで同環境での比較をする。

3．3　実験環境

実験に使用した機材は、以下の通りである。

PC を表2に、接続機器を表3に示す。

表 2

	Server	観測 PC	PC　1	PC　2	PC　3
OS	Windows2000 Advanced Server	Windows2000 SP4	WindowsXP	WindowsXP SP2	WindowsXP
CPU	Celeron 1.7GHz	Celeron 2.4GHz	Pentium4 2.4GHz	Pentium4 3GHz	Celeron 2.4GHz
Memory	512MB	256MB	1GB	512MB	512MB
LAN	100Mbps	100Mbps	100Mbps	100Mbps	100Mbps
IP address	192.168.1.200	192.168.1.150	192.168.1.132	192.168.1.148	192.168.1.110
Browser	IE 6	IE 6	IE 6	IE 6	IE 6
Mailer	―――――	edmax　2.85.5F	Beckey!　2.05.04	ALmail　1.13	Outlook　6
FTP client	Tiny FTPd[3] 0.52d	FFFTP　1.92	FFFTP　1.92	FFFTP　1.92	FFFTP　1.92

表 3

	スイッチ A	スイッチ B	スイッチ C	ルータ A
型番	Catalyst3524-XL	FS716XJ	ES3008A	RT140e
メーカー	Cisco	アライドテレシス	ACTON	YAMAHA
種類	L3－Switching HUB	Switching HUB	Switching HUB	Router
通信速度	10/100 Mbps	100 Mbps	100 Mbps	100Mbps

これらの機器を用いてネットワークを構成するが、検証するプロトコルにより構成が異なるため接続図は次章で随時示すことにする。またパケットをモニターするためのソフトウェアは、Windows で使用可能な汎用キャプチャドライバである WinPcap を用いて自作した。4.4.2 では問題の原因究明のため例外的に Ethereal を使用した。

[3] TinyFTPd は FTP サーバ用のソフトウェアである

第４章　実験結果

４．１　HTTP の実験

４．１．１　HTTP の実験構成

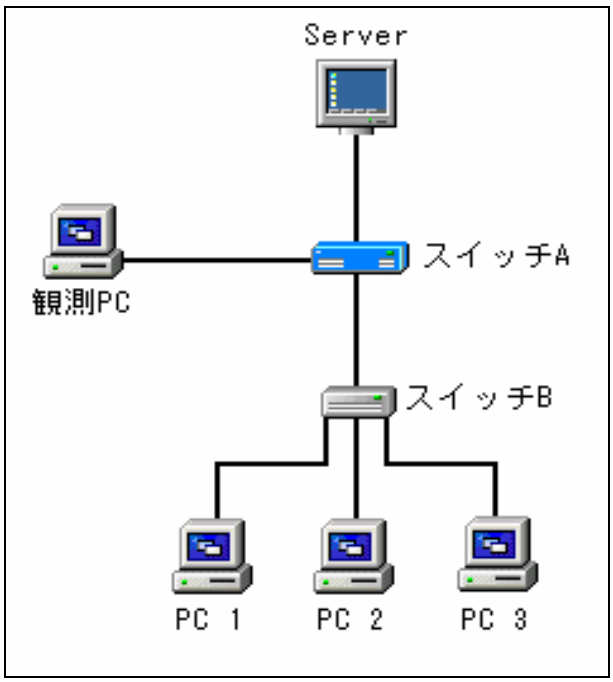


図 ３

HTTP プロトコルにおける実験は図３に示すネットワーク構成を用いた。ノード間の接続は全て 100Mbps である。

まず Server（Apache, version 2.0.52 win）には、文字のみでファイルサイズ 100MB の HTML ファイル（eins.html）を置いておく。次に、観測 PC から Server へ向けて実験終了まで連続した ping を打ち続ける。最後に、PC 1～PC 3 が順次 Server へとアクセスして用意しておいた eins.html を読み込む。その通信状況を、スイッチ A を通して観測 PC でモニターするという形を取った。

（ネットワーク中にある各ノードの性能は表２，表３参照）

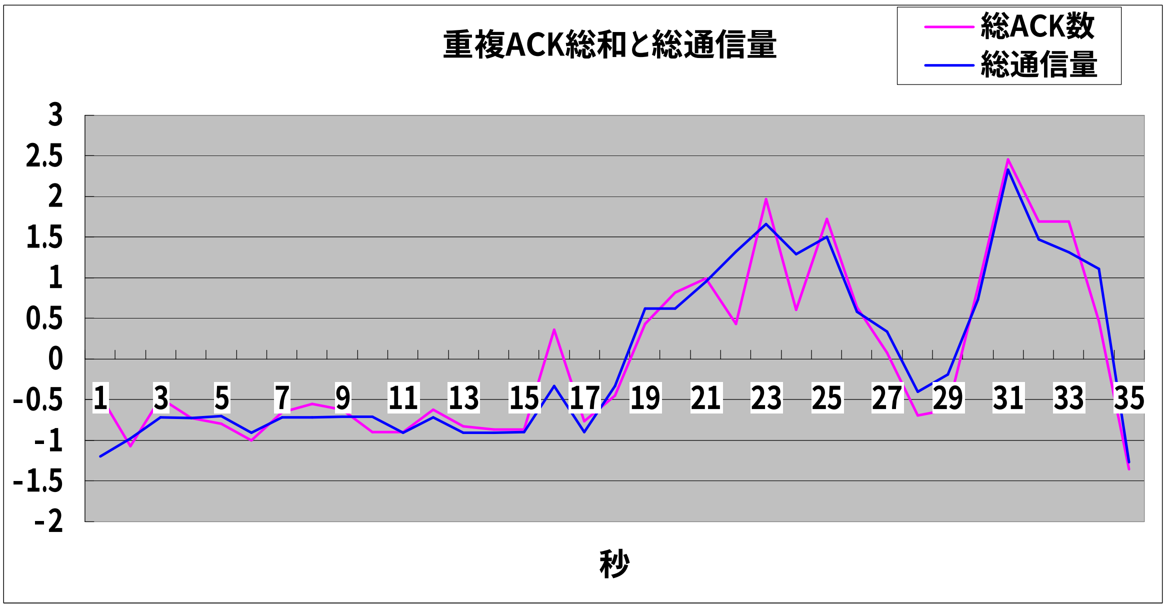
１，観測 PC より Server へ ping の連続送信を開始する
２，１より１５秒後、PC 1 が Server 上の eins.html を読み込み始める
３，２より１０秒後、PC 2 が Server 上の eins.html を読み込み始める
４，３より１０秒後、PC 3 が Server 上の eins.html を読み込み始める
５，４より１０病後、実験終了

[eins.html の概要]

```
<html>
<head><title>実験用 html</title></head>
<body>
<!- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
                          中略
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-->
表示されたら終了。<BR> </body></html>
```

eins.html へのアドレスは http://192.168.1.200/eins.html である。

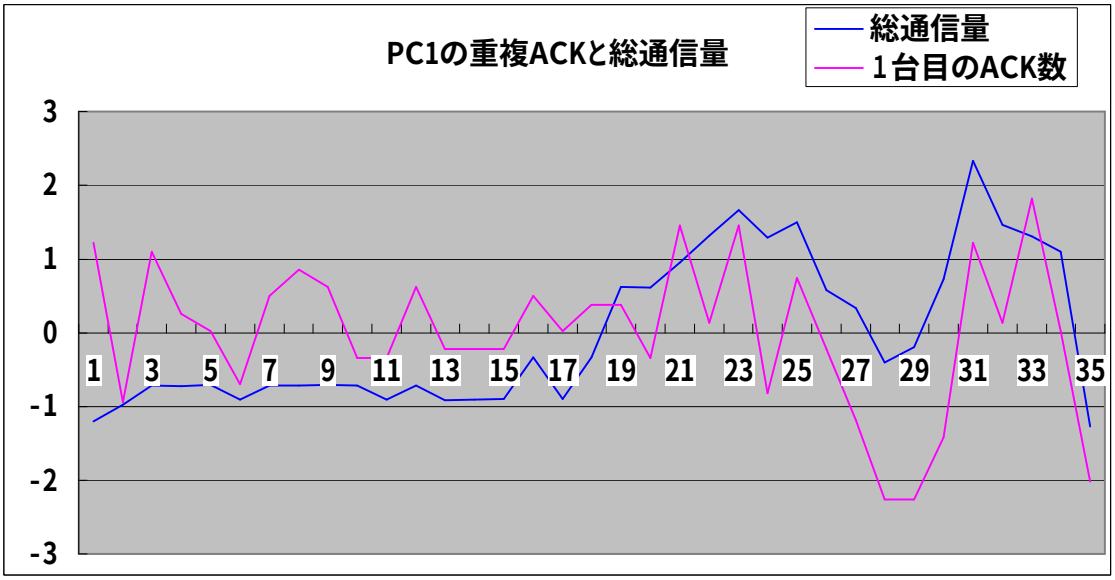
４．１．２　HTTP の実験結果



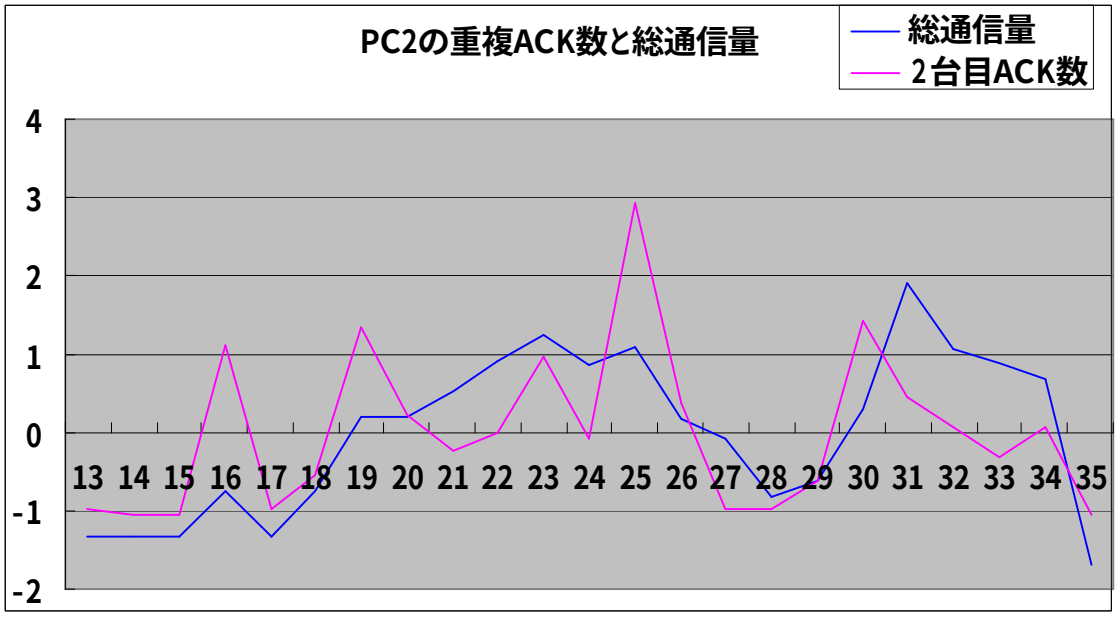
相関係数：0.946915　　　　図４

モニターが測定した総通信量と、同じくモニターされた全ての重複 ACK の総数のグラフ[4]を図４に示す。このように通信量と発生する重複 ACK は非常に強い相関があることが解る。これは全ての PC から発生した ACK の総和であるが、もし各々の PC で同じように強い相関が得られれば、RTT のように指標として実用できることになる。

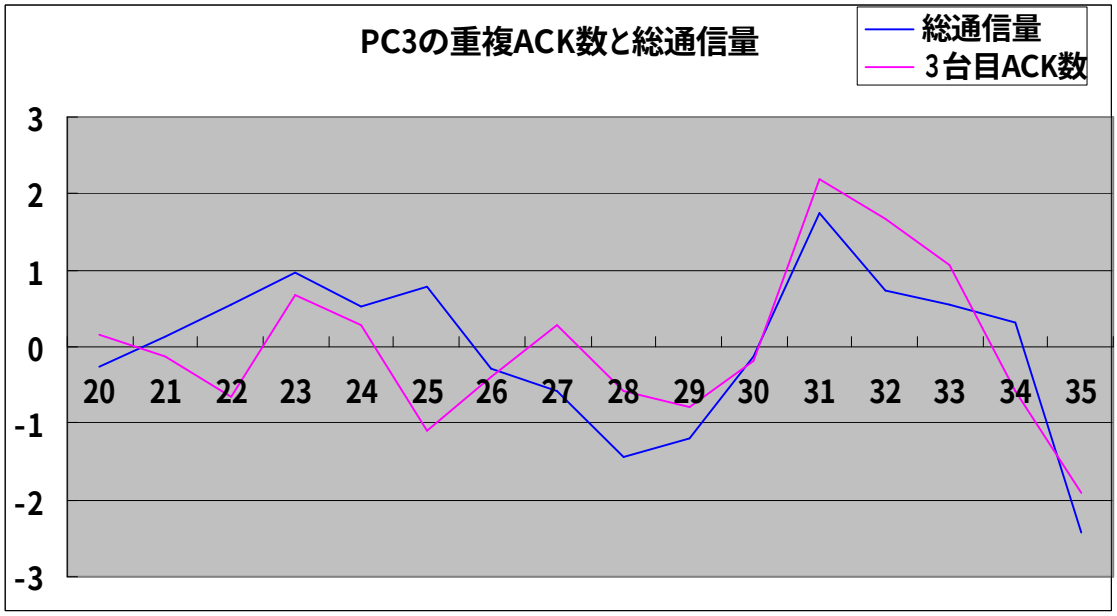
[4] それぞれのデータを 平均０,標準偏差１ へと正規化した上でグラフ化してある



相関係数：0.284893723　　**図5**



相関係数：0.597245117　　**図6**

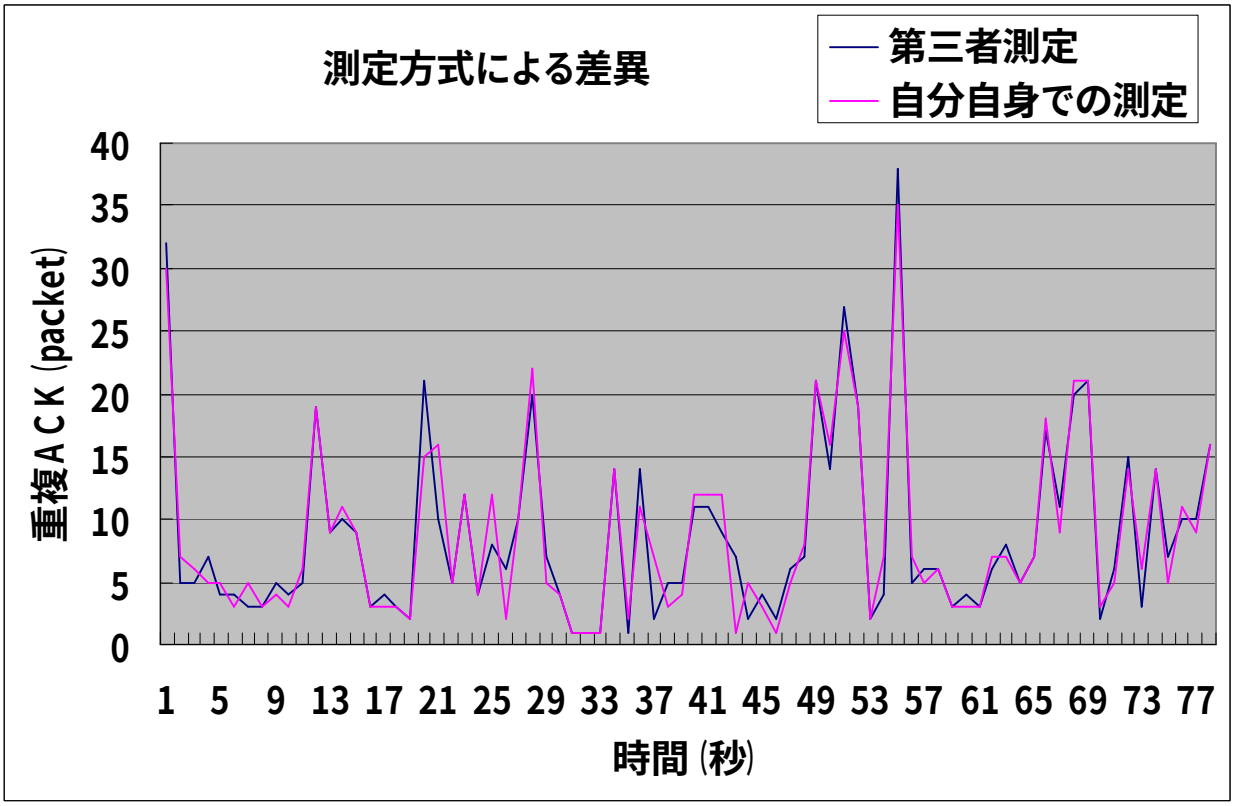


相関係数：0.706455459　　図７

図５は最初に接続を開始する PC1 から発生した重複 ACK と総通信量のグラフである。
図６は最初に接続を開始する PC2 から発生した重複 ACK と総通信量のグラフである。
図７は最初に接続を開始する PC3 から発生した重複 ACK と総通信量のグラフである。

なお実験は 4.1.1 で示したとおり行ったが、IE がコネクションを確立してアクセスを始めるまで多少時間が前後する。そのため PC1 と PC2、PC2 と PC3 の実際のデータ転送間隔は正確に１０秒というわけではない。本実験は通信量の増加に伴う重複 ACK の変化を見ることが目的であるため、実験結果には支障がないと判断しそのまま続行した。

次に、スイッチを通して全体を監視した場合と自分自身で測定した場合の差を図８に示す。
これはスイッチＡからＰＣ１～３間でのパケットの損失が、実験結果に影響を与えるほど大きいかどうかを確認するものである。



相関係数：0.963375549　　**図８**

多少の差はあるものの、0.96 という非常に高い相関を示しており、モニターを通して第三者が観測した値と自分自身で測定した値との間に大きな差はなく、総通信量を予想する指標として自分自身で測定した値を用いても問題がないと言える。実際に双方のデータを総通信量との比較に使用した場合でも、相関係数の解釈に全く影響を与えない程度の差しかなかった。

４．２　FTP の実験

４．２．１　FTP の実験構成

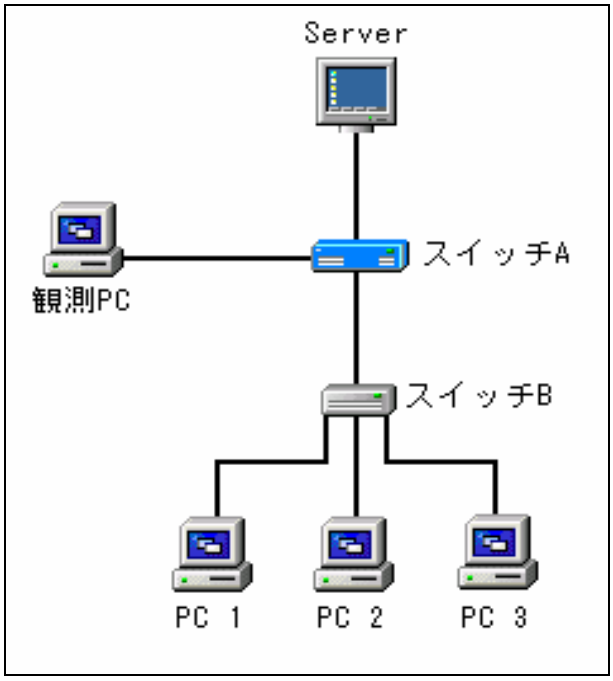


図 ９

FTP プロトコルにおける実験は**図９**に示すネットワーク構成を用いた。

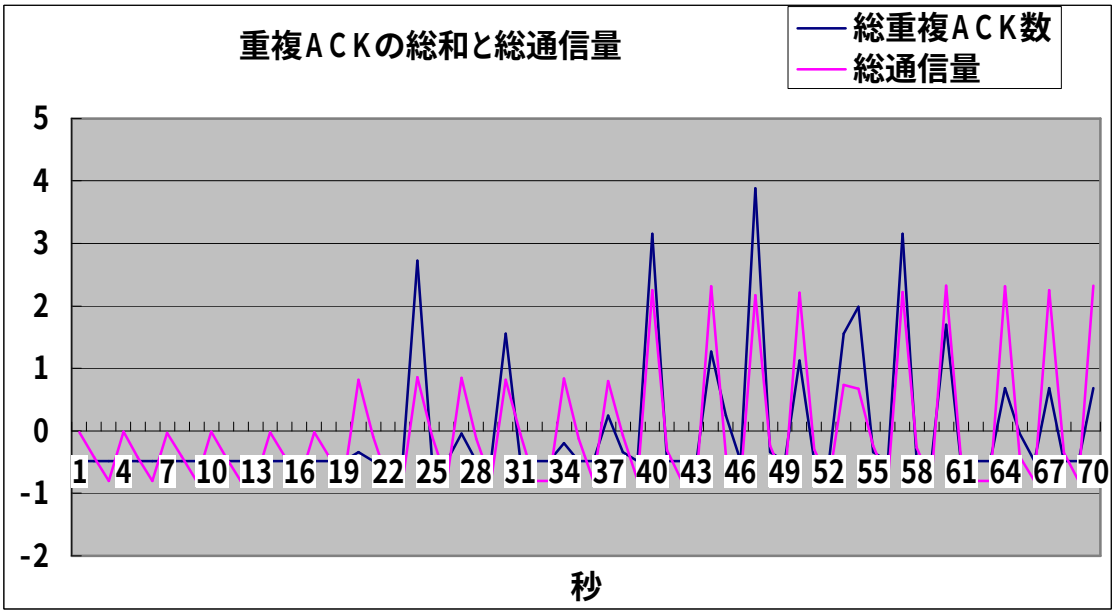
まず Server（TinyFTPd，version 0.52d） には、ファイルサイズ約 300MB の avi ファイル（se1.avi）を置いておく。次に、観測 PC から Server へ向けて実験終了まで連続した ping を打ち続ける。最後に、PC 1～PC 3 が順次 Server へとアクセス[5]して用意しておいた se1.avi[6]を読み込む。その通信状況を、スイッチ A を通して観測 PC でモニターするという形を取った。

（ネットワーク中にある各ノードの性能は**表２，表３**参照）

１，観測 PC より Server へ ping の連続送信を開始する
２，１より２０秒後、PC 1 が Server 上の se1.avi を読み込み始める
３，２より２０秒後、PC 2 が Server 上の se1.avi を読み込み始める
４，３より２０秒後、PC 3 が Server 上の se1.avi を読み込み始める
５，４より３０秒後、実験終了

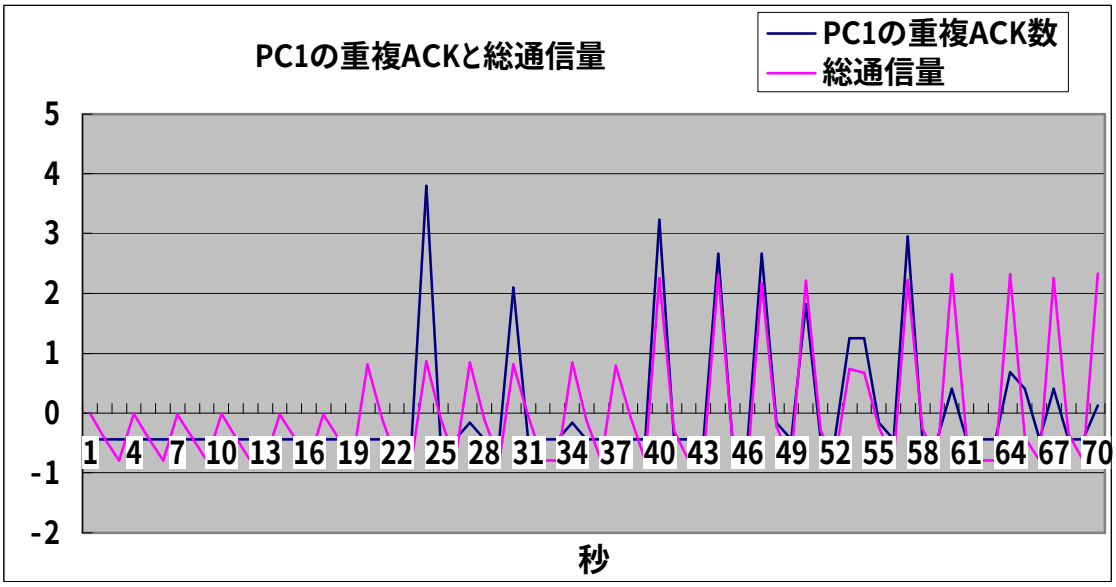
[5] FTP Server のアドレスは 192.168.1.200、Anonymous 接続を用いた。
[6] 306,018,927 Byte

４．２．２　FTP の実験結果

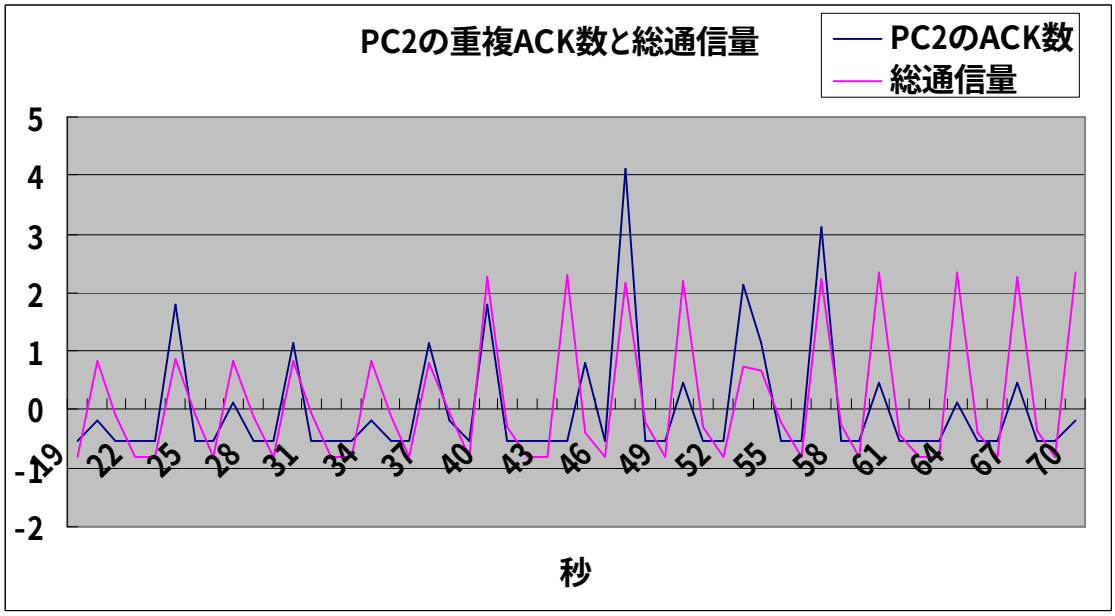


相関係数：0.782339051　　**図１０**

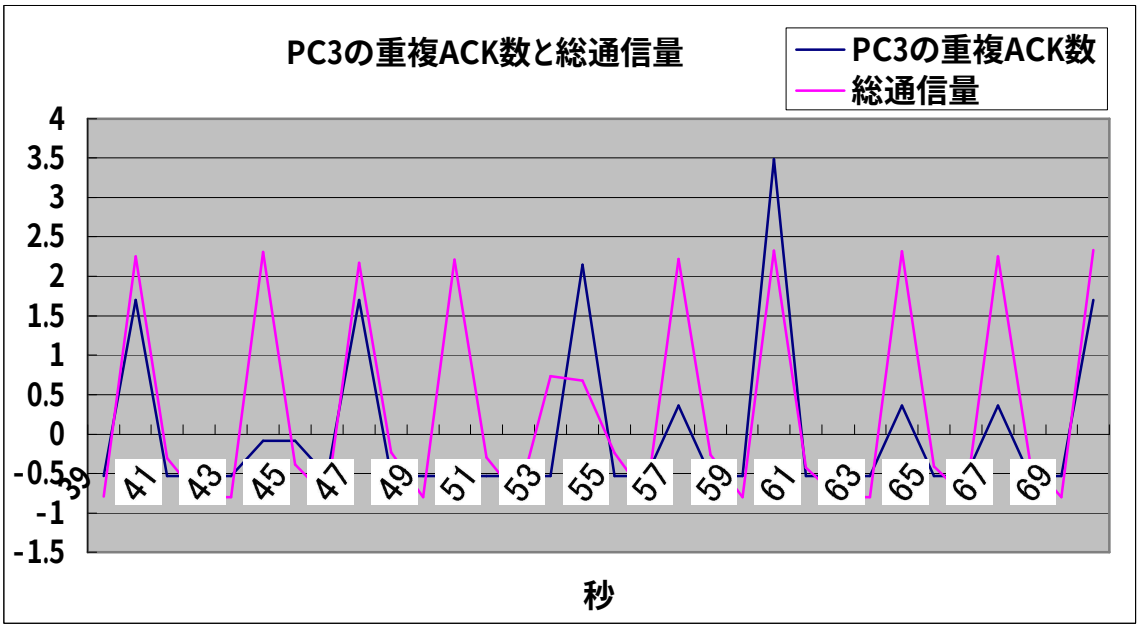
総通信量と発生した重複ＡＣＫの総数のグラフを**図１０**に示す。こちらも HTTP の場合と同様に強い相関が見られる。一定時間ごとに通信量が０になっているが、これはサーバー機の性能か FTPdaemon ソフト（© TinyFTPd）の性能によるものと思われる。



相関係数：0.724582982　　**図１１**



相関係数：0.636454327　　**図１２**



相関係数：0.68335364　　**図１３**

図１１は最初に接続を開始する PC1 **から発生した重複** ACK **と総通信量のグラフである。**
図１２は最初に接続を開始する PC2 **から発生した重複** ACK **と総通信量のグラフである。**
図１３は最初に接続を開始する PC3 **から発生した重複** ACK **と総通信量のグラフである。**

４．３　POP の実験

４．３．１　POP の実験構成

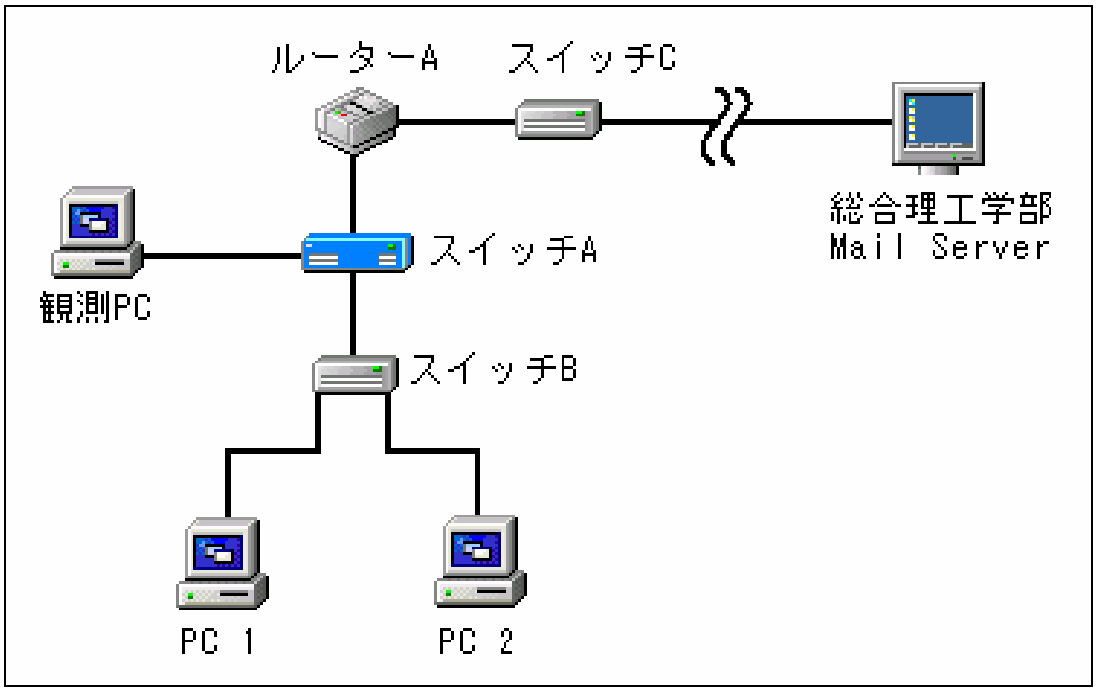


図１４

POP プロトコルにおける実験は図１４に示すネットワーク構成を用いた。

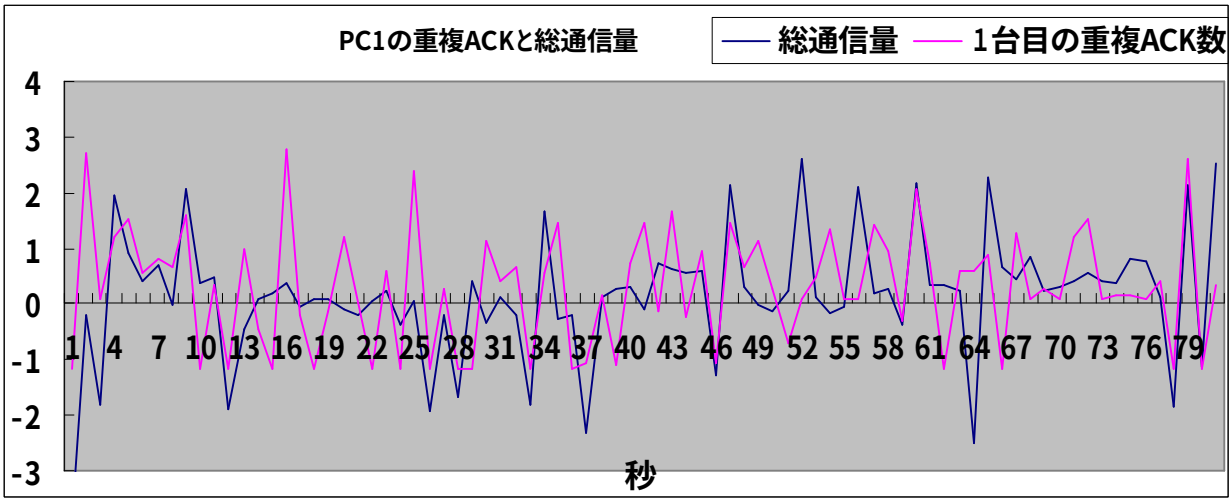
今回の Server には、強度の関係で総合理工学部のものを使用した[7]。ファイルサイズ約 50MB の mpeg ファイル（ba.mpeg）を置いておく。次に、観測 PC から Server へ向けて実験終了まで連続した ping を打ち続ける。最後に、PC 1～PC 2 が順次 Server へとアクセスして用意しておいた ba.mpeg を読み込む。その通信状況を、スイッチ A を通して観測 PC でモニターするという形を取った。

（ネットワーク中にある各ノードの性能は表２，表３参照）

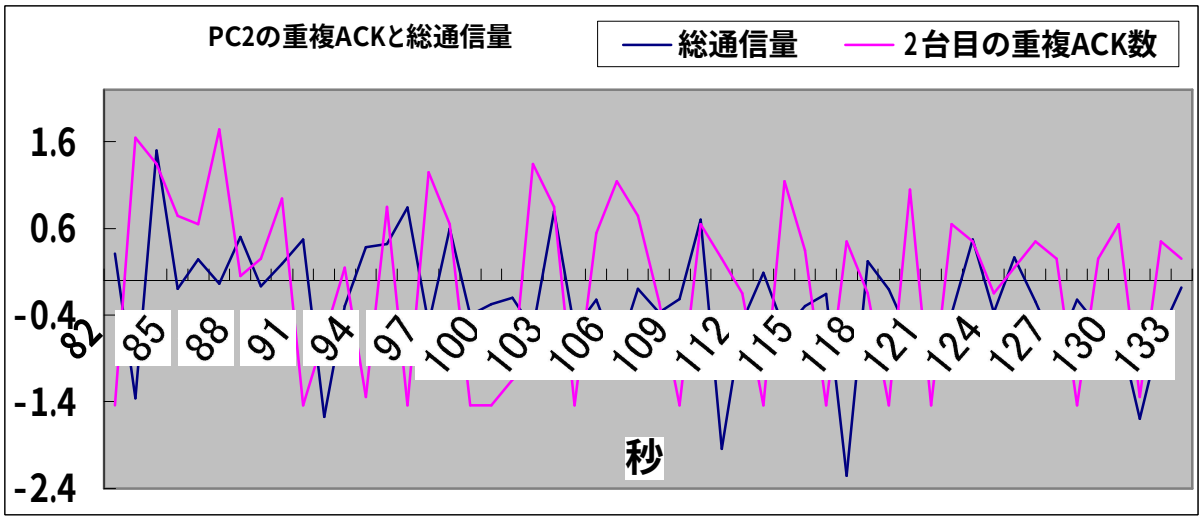
１，観測 PC より Server へ ping の連続送信を開始する
２，１より２０秒後、PC 1 が Server 上の ba.mpeg を読み込み始める
３，２より２０秒後、PC 2 が Server 上の ba.mpeg を読み込み始める
４，３より３０秒後、実験終了

[7] POP サーバの詳細は不明である

４．３．２　POP の実験結果



相関係数：0.437782359　　**図１５**



相関係数：-0.01989638　　**図１６**

図１５は最初に接続を開始する PC1 から発生した重複 ACK と総通信量のグラフである。
図１６は最初に接続を開始する PC2 から発生した重複 ACK と総通信量のグラフである。

なおサーバーの負荷を考え、実験は HTTP,FTP と違い２台で行った。

４．４　SMTP の実験

４．４．１　SMTP の実験構成

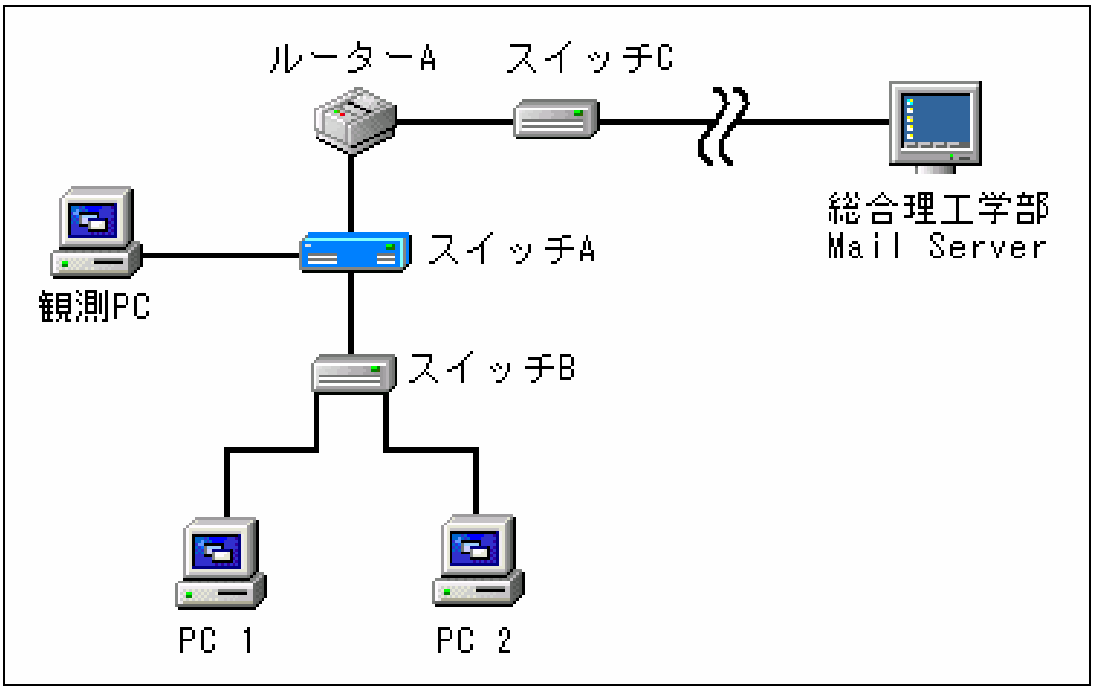


図１７

SMTP プロトコルにおける実験は図１５に示すネットワーク構成を用いた。

今回の Server には、強度の関係で総合理工学部のものを使用した[8]。ファイルサイズ約 50MB の mpeg ファイル（ba.mpeg）を置いておく。次に、観測 PC から Server へ向けて実験終了まで連続した ping を打ち続ける。最後に、PC 1～PC 2 が順次 Server へとアクセスして用意しておいた ba.mpeg を読み込む。その通信状況を、スイッチ A を通して観測 PC でモニターするという形を取った。

（ネットワーク中にある各ノードの性能は表２，表３参照）

１，観測 PC より Server へ ping の連続送信を開始する
２，１より２０秒後、PC 1 が Server 上の ba.mpeg を読み込み始める
３，２より２０秒後、PC 2 が Server 上の ba.mpeg を読み込み始める
４，３より３０秒後、実験終了

[8] SMTP サーバの詳細は不明である

４．４．２　SMTP の実験結果

SMTP で実験を行った結果、重複 ACK がほぼ全く発生していなかった。実験中のスイッチングハブは、負荷時には collision ランプが点灯し続けている状態であり、輻輳が発生していないとは思えない。そこで Ethereal を使い通信中のデータを調べた結果が図１６である。

本来なら ACK 値や他の制御フラグなどが表示されるフィールドに Message Body としか表示されていない。データ部分のサイズよりメール本文であることが解るが、ACK フラグが常に０であることやその他のフラグも使われている様子がない事からこれは標準的な TCP 通信ではないと思われた。そこで SMTP が定義されている RFC821 文書を調べたところ、SMTP は TCP の他にも NCP、NITS、X.25 の転送サービスが使用される可能性があることが解った。今回は X.25 が使用されたと思われるが、コネクションの確立、切断（３ハンドシェイク）にしか通常の TCP が使用されない。

よって本研究の対象外であり、除外せざるを得なかった。

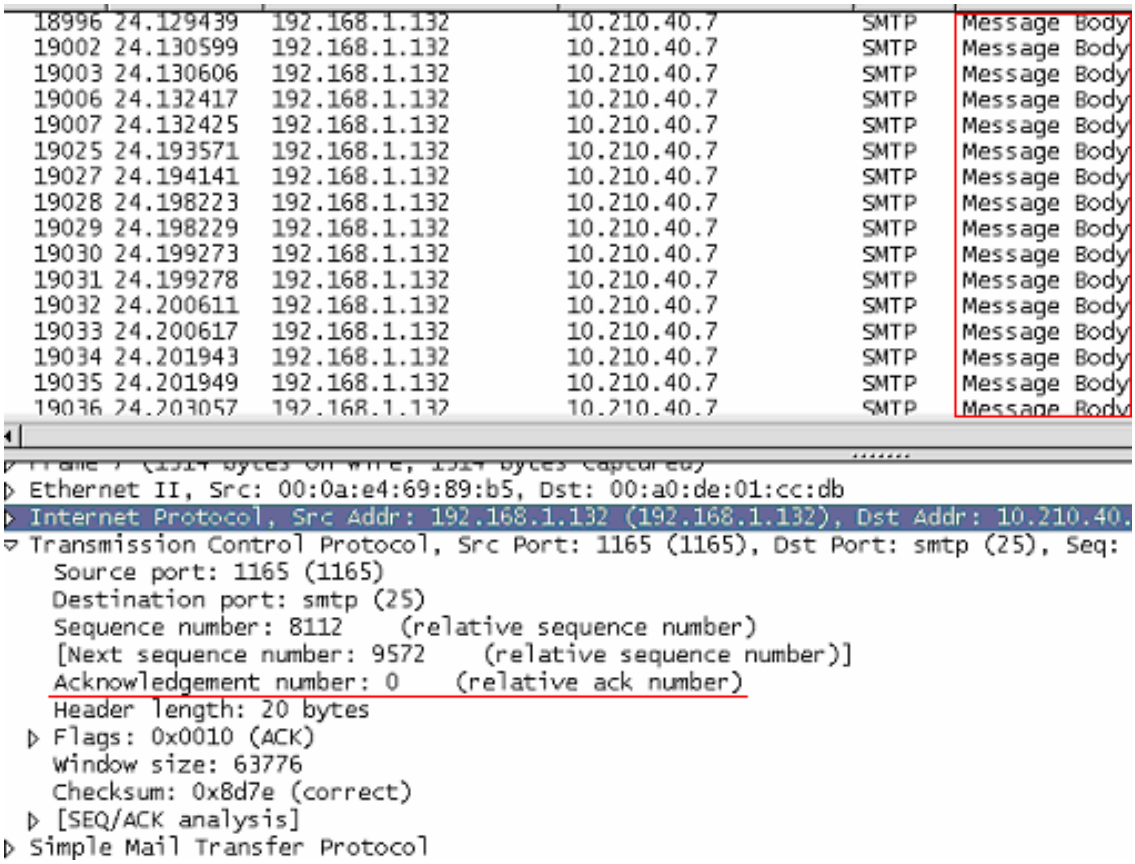


図１８

第５章　実用性の検証、考察

５．１　ＲＴＴとの比較

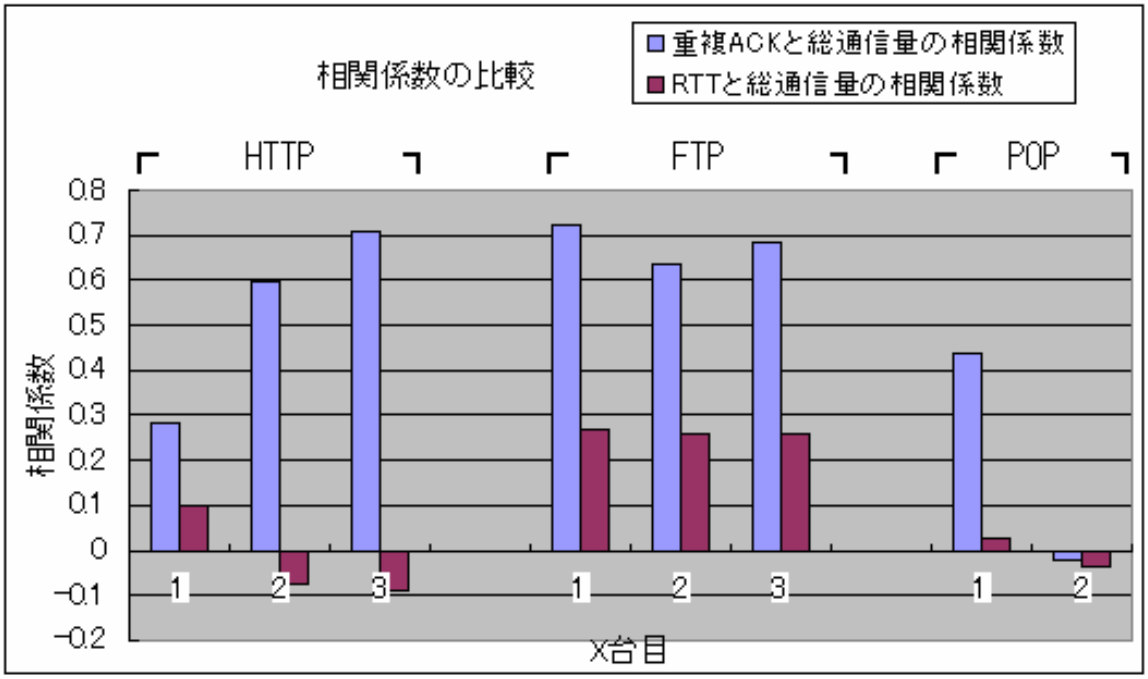


図１９

図１９はそれぞれのプロトコルについて、総通信量に対する重複 ACK と、ping 応答結果を用いて秒単位で測定した RTT の相関係数を示した図である。すべての場合において、重複 ACK は RTT よりも通信量との相関が高い[9]。ping は確実な実測値であるため実通信速度の指標となっているが、RTT 値が大きくてもそれは単に帯域が細いためなのか、回線の混雑によるものなのかは解らない[10]。それに対し、重複 ACK は現在の通信量に比例して発生数が多くなる性質を持つため実効帯域幅を予想することができる。つまり実通信が 100KB/s であってもそのうち 30KB/s がデータの再送であれば実効帯域幅は 70KB/s であると言える。

RTT と比較した場合、次の利点がある

１，回線の混雑度をエンドノードで推測できる

２，サーバーに負荷をかけない（ping 攻撃などと見なされることはない）

３，接続先のサーバーに合わせてリアルタイムで結果が得られる

４，ポートの開閉、ルーターによるフィルターなど環境の影響を受けにくい

---

[9] POP の２台目は負の相関なので、値が小さい方が良い事になる。

[10] ping パケットはよほどの混雑でないとパケットロストしづらい事が実験中に判明。

５．２　重複 ACK の精度向上

通信量との相関が認められた重複 ACK であるが、**図１９**の HTTP に現れるようにあまり長時間の測定をすると（急に通信量が増加し、そのままの状態が続くと）精度が落ちていることが見て取れる。しかし ping のようにその１瞬だけを計るという方法は取れない。そこで、もっとも良い精度が得られる一定時間を考察することにした。

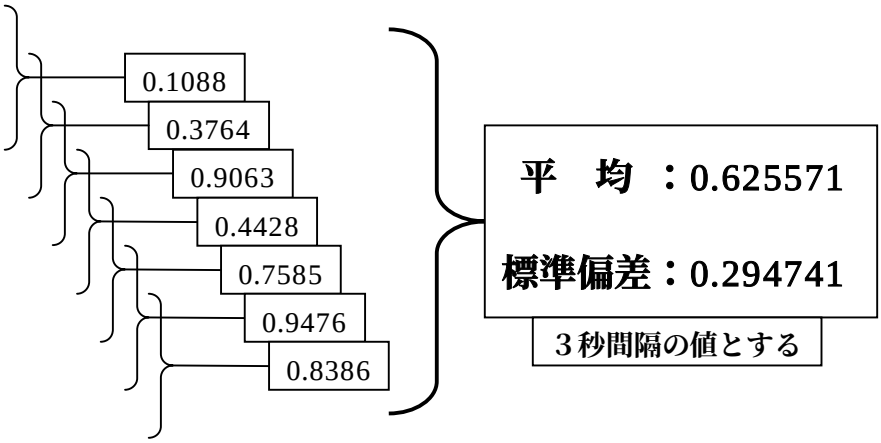
それぞれのプロトコル別データに対して、

[３秒間隔の場合]

１，　１～３秒目の総通信量と重複 ACK の相関係数
２，　２～４秒目の総通信量と重複 ACK の相関係数
３，　３～５秒目の総通信量と重複 ACK の相関係数
：
：

のように相関係数を出していく。３秒間隔で算出した全ての相関係数の平均と標準偏差を**３秒間隔の値**とする。なお総通信量とは、その１秒間におけるネットワーク中の全通信量であり、それまでの累積通信量ではない。
（例外として、観測 PC とスイッチ A 間の通信量は含めない）

	総通信量	重複 ACK
１秒目	3.280405	7.923642
２秒目	0.197735	2.804111
３秒目	4.070253	1.283217
４秒目	6.063623	4.583998
５秒目	2.957024	1.557153
６秒目	3.043833	4.914399
７秒目	5.601141	6.428925
８秒目	0.921222	0.582794
９秒目	9.436021	5.730151



同様に３秒間隔から２０秒間隔まで計算し、プロットしたものが**図２０，図２１，図２２**である。３秒から始めたのは、２秒では値の分散値が０となり相関係数が出ない場合が多かったためである。終了は、HTTP において明らかな変化が見られたことと当初の目的[11]からみて十分であると判断し、２０秒まで算出した。

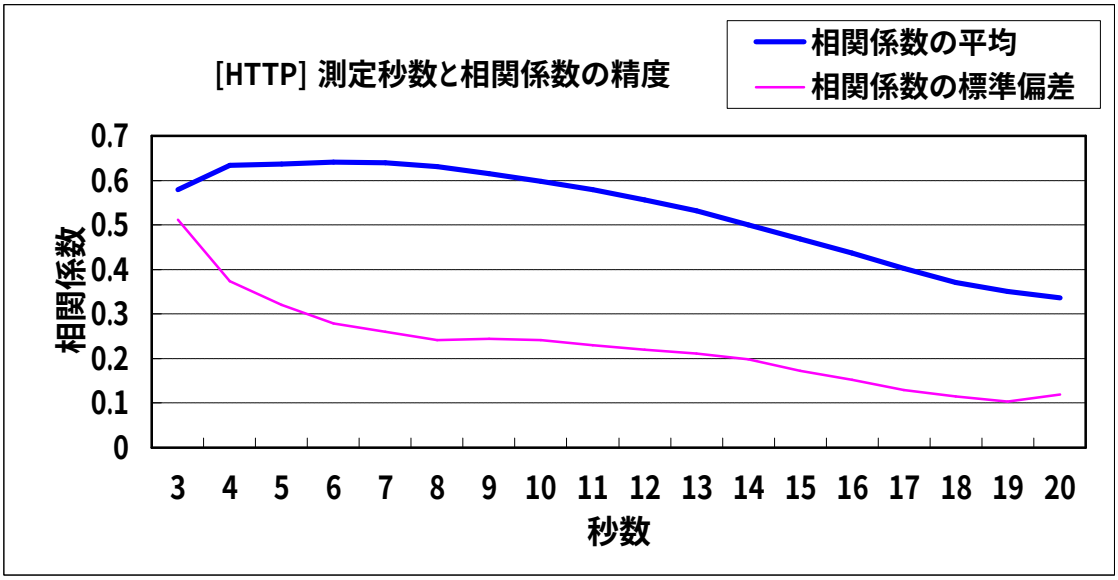


図２０

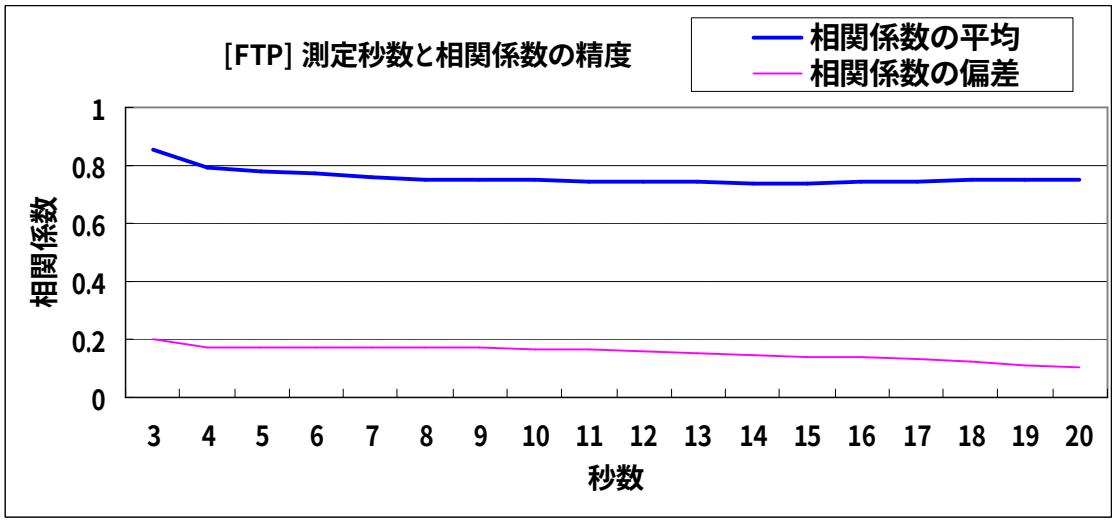


図２１

[11] **あまり長時間の測定をすると精度が落ちている（P.25）**

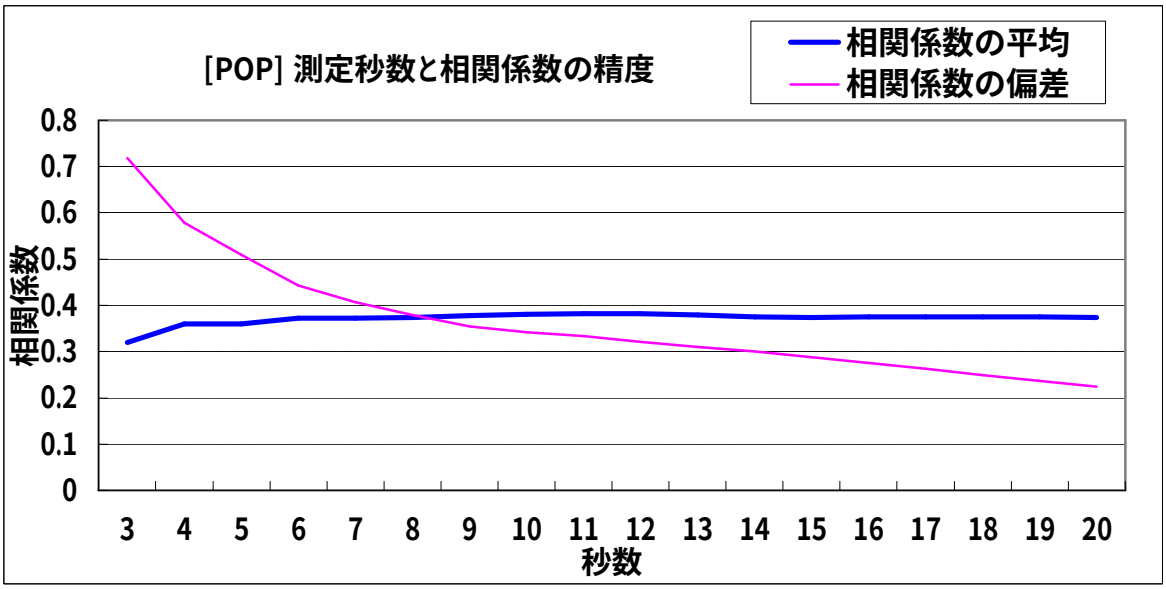


図２２

判断の基準として、当然相関係数の平均は高い方がよい。しかし、相関係数の上下が余りにも激しい場合（＝値が分散している場合）というのは測定結果の信頼度がその時々によって違うということであり、好ましい状態ではない。よって相関係数は高く、標準偏差は低い方がよいと言える。

HTTP, FTP, POP のグラフを上記の基準で考察した結果、測定開始から**８秒間**（前後）の測定を行ったときが最も精度が良いと判断した。

５．３　まとめと展望

本研究では HTTP, FTP, POP, SMTP の４種類のプロトコルについて検証実験を行い、そのうち HTTP, FTP, POP の３種類について有効であるという結果を得ることが出来た。現実にはさらに多種多様なプロトコルが存在するか、測定時に自分自身がこの３種類のうちいずれかのプロトコルを使用していれば測定できることになる。

検証の実験環境が大学内の LAN という限定された場所であったため、グローバルネットワークというさらに多くのノードが存在する場合でも同様の結果が得られるかは解らない。また Windows95 などが動いている旧式の PC では、パケットフィルタリングを行う時 CPU へ高い負荷がかかり正確な測定ができない可能性がある。しかし既存のシステムのみで目的地までにある輻輳の度合いを検出するという方法は、その高い精度もさることながら低コストであり、また他のソフトウェアとも柔軟に連動できるため実装が容易であるという利点がある。

今後も高速化され続けるだろうネットワークにおいて、傾向を事前に検知し輻輳を減らせる事は大量のデータを有効活用するための重要な要素となるだろう。

５．４　謝辞

本研究を進めるにあたり、最後まで熱心なご指導、ご助言をして頂きました田中章司郎教授には心より御礼申し上げます。また、本研究のきっかけとなりましたすばらしい研究とアイデアを提供して下さいました樋上さんには感謝の言葉もありません。同研究室の皆様にも、研究内容のみに留まらない数々の御協力と御助言を頂きました。ここに厚く御礼申し上げます。

なお、本論文，本研究で作成したプログラム及びデータ，ならびに関連する発表資料等の知的財産権を、本研究の指導教官である田中章司郎教授に譲渡致します。

５．５　参考文献

[1]

TCP 制御フラグを使った輻輳検知に関する研究　　樋上 智彦　2004

[2]

Congestion detection in ATM networks

Jian-Min Li (Dept. of Appl. Math., Adelaide Univ., SA, Australia;); Widjaja, I.; Neuts, M.F. **Source:** *Performance Evaluation*, v 34, n 3, 19 Nov. 1998, p 147-68

[3]

TCP Vegas: New techniques for congestion detection and avoidance

Brakmo, L.S. (Dept. of Comput. Sci., Arizona Univ., Tucson, AZ, USA); O'Malley, S.; Peterson, L.L. **Source:** *Computer Communication Review*, v 24, n 4, Oct. 1994, p 24-35

[4]

End-to-end wireless TCP with non-congestion packet loss detection and handling

Jae-Joon Lee (Dept. of Electr. Eng., Univ. of Southern California, Los Angeles, CA, USA); Fang Liu; Kuo, C.-C.J. **Source:** *Proceedings of the SPIE - The International Society for Optical Engineering*, v 5100, 2003, p 104-13

[5]

◎Congestion detection based on rate and buffer occupancy measurements

Cheng, T.H. (Sch. of Electr. & Electron. Eng., Nanyang Technol. Inst., Singapore); Cheah, K.L.; Oh, S.H. **Source:** *International Journal of Communication Systems*, v 8, n 6, Nov.-Dec. 1995, p 373-5

[6]

ルーティング&スイッチングハンドブック　Gene/著　2003　秀和システム/出版

[7]

C言語による TCP/IP ネットワークプログラミング　小俣光之/著　2001

[8]

Cisco 社、Catalyst 設定資料各種

[9]

IETF Home Page

http://www.ietf.org/

http://www.ietf.org/rfc.html

５．６　使用した公開プログラム

Tiny FTP daemon

http://hp.vector.co.jp/authors/VA002682/tftpd_frame.htm

WinPcap，Pcaplib

http://netgroup-serv.polito.it/winpcap/

AL-mail32

http://www.almail.com/

FFFTP

http://www2.biglobe.ne.jp/~sota/

Ethereal

http://www.ethereal.com/