TCP制御フラグを使った輻輳検知に関する研究

S003068

樋上　智彦

計算機科学講座

田中研究室

２００４年2月２７日

目次

第１章　はじめに

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せずエンドノード間での輻輳回避をする必要がある。現在の技術では輻輳制御をＴＣＰ層で管理しているのが現状である。そして、輻輳を検知・制御している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。ソフトウェアによる輻輳の検知があまり行われない理由としてはソフトウェアでの輻輳の検知が困難であるという点がある。

ソフトウェアでの輻輳検知の困難さとしては

- 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（自分で推測してネットワークに送出することしかできない）
- さまざまな通信媒体の性質を抽象化しているために上位プロトコルから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

ということがあげられる。

本研究では TCP 制御フラグの変化をまとめ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、従来のやり方とは違ったユニークな方法が使えないかという目的で制御フラグを用いた実験を行った。

第２章　輻輳検知に関する技術

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。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してしまうことを意味する。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では、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　いう問題がおこる。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てしまう。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、その技術をもってして輻輳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　従来の輻輳検知に関する技術

ネットワークの混雑に対応するために、日夜輻輳に関する研究が行われている。ここではその一例を紹介したいと思う。（参考文献[1]）

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する。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。

・　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。

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

表価値）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

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）

で計算される。

２．３　輻輳技術に関係した先行研究

先行研究をまとめるにあたって、ＩＮＳＰＥＣ・ＳＣＩＲＵＳを使用した。これらの論文検索システムから本研究と似通った事例、先だって行われた研究をとりあげた。研究内容が近しいもの、共通項があるものなどを体系的に書き出すことによってこの分野一般にわたる先行研究をわかりやすくする

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	422	40	277
検索用語	Protocol	TCP/IP friendly	Congestion detection
件数	251	15	21

（注）これらの検索結果は 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。また、検索ワードとして ((((**検索用語**) WN CV) NOT ((**video**) WN All fields)) AND ((**streaming**) WN All fields)), 1990-2004

をもとに検索した。

２．３．１　Congestion control(輻輳制御)をもとにした研究事例

輻輳制御を元にした研究事例は一番多くの検索結果を得たが同時に他の検索用語と重複する部分があった。

表２　関連のある単語による検索結果その２

重複ワード	Variable Rate control	Adaptive control	Protocol
件数	36	69	34
重複ワード	TCP/IP friendly	Congestion detection	
件数	15	15	

興味深いのは輻輳制御とＴＣＰ親和性に関する研究が完全に一致していることである。以下に本研究と関連が深いと思われる研究をあげる

SRTP: TCP-friendly congestion control for multimedia streaming
Byunghun Song (Sch. of Electron. Eng., Kwangwoon Univ., South Korea;); Kwangsue Chung; Yongtae Shin **Source:** *Information Networking. Wireless Communications Technologies and Network Applications. International Conference, ICOIN 2002. Revised Papers, Part II (Lecture Notes in Computer Science Vol.2344)*, 2002, p 529-38

Streaming media congestion control using bandwidth estimation（ＳＭＣＣ）
Aboobaker, N. (Dept. of Comput. Sci., California Univ., Los Angeles, CA, USA); Chanady, D.; Gerla, M.; Sanadidi, M.Y. **Source:** *Management of Multimedia on the Internet. 5th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, MMNS 2002. Proceedings (Lecture Notes in Computer Science Vol.2496)*, 2002, p 89-100

◎Adaptive traffic-based techniques for live multimedia streaming
Muntean, G.-M. (Dept. of Comput. Sci., Univ. Coll. Dublin, Ireland); Murphy, L. **Source:** *Proceeding of the International Conference on Telecommunications*, 2002, pt. 1, p 1183-7 vol.1

◎A congestion control scheme for continuous media streaming applications
Balaouras, P. (Dept. of Inf. & Telecommun., Athens Univ., Greece); Stavrakakis, I. **Source:** *From QoS Provisioning to QoS Charging. Third COST 263 International*

Workshop on Quality of Future Internet Services; QofIS 2002 and Second International Workshop on Internet Charging and QoS Technologies, ICQT 2002. Proceedings (Lecture Notes in Computer Science Vol.2511), 2002, p 194-204

Detection of congestion signals from relative one-way delay

Tobe, Y. (Graduate Sch. of Media & Governance, Keio Univ., Japan;); Aida, H.; Tamura, Y.; Tokuda, H. **Source:** *Transactions of the Information Processing Society of Japan*, v 42, n 12, Dec. 2001, p 3222-30

２．３．２　輻輳検知に関しての先行研究

上記の図において検索された結果のうち、本研究の課題である輻輳検知についての先行研究を紹介する。本研究での独自の輻輳検知技術のための参考として活用した。

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

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

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

◎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

２．３．３　先行研究のまとめと本研究内容

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。また、TCP フラグレベルでの輻輳を検出する研究はなされていない。

本研究では先行研究でさかんに行われているような、ハードウェアレベルの研究ではなく、ソフトウェアによって輻輳を検知し、本来エンドノード自体では検出しにくい輻輳を検出しようという試みを行いたいと思う。

第３章　輻輳検知の基礎

３．１　プロトコルに関して

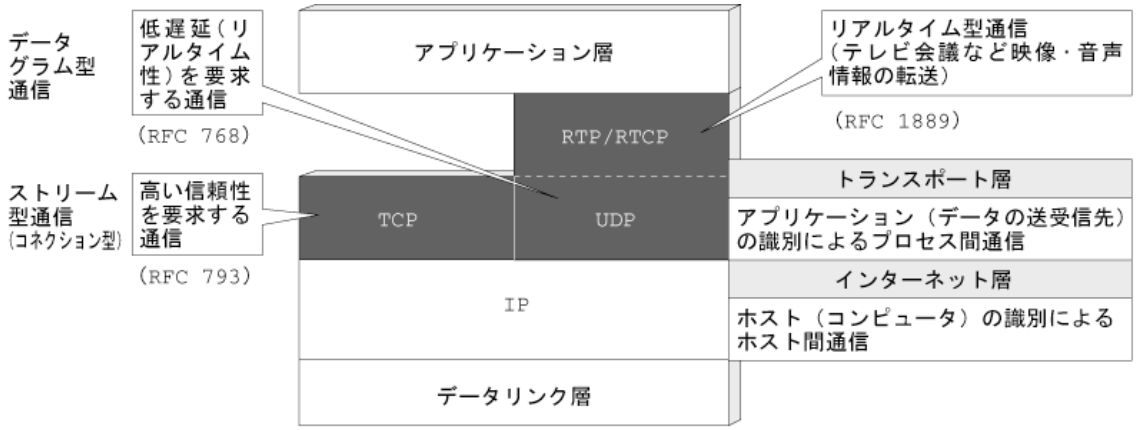


図1　トランスポート層プロトコル

現在のネットワークにおいて主に使われているプロトコルとしてはＴＣＰ／ＩＰがある。データリンク層に置けるデータ転送は、同一の通信路上のホスト間に限られているが、この制約を超えて複数のネットワークを経由してネットワーク間データ転送を可能にしているのがネットワーク層、トランスポート層（ＴＣＰ／ＩＰ）である。このうち本研究の内容であるトランスポート層のプロトコルについての説明をする。トランスポート層には大きく分けてＴＣＰとＵＤＰとの２つのプロトコルが存在している。以下文献[１]、[２]によりこれ等を紹介する。

３．１．１　TCP プロトコル

ＴＣＰは俗にコネクション型通信と呼ばれ、速度よりも送受信の信頼性を重視したプロトコルである。信頼性を確保するためにヘッダの中にコードビットを用意しており、ビットの有無によって送受信間での信頼性を保っている。また、輻輳制御機構をもっている。

TCP の特徴としては

① 基本的なデータ転送
② 信頼性
③ フロー制御
④ 多重化
があげられる。

1 3 2

<table>
<tr><td colspan="6">発信元ポート番号(16 ビット)</td><td colspan="2">宛先ポート番号(16 ビット)</td></tr>
<tr><td colspan="8">シーケンス番号(32 ビット)</td></tr>
<tr><td colspan="8">確認応答番号(32 ビット)</td></tr>
<tr><td rowspan="2">ヘッダ長
(4 ビット)</td><td rowspan="2">予約済み
(6 ビット)</td><td colspan="6">コードビット
(各 1 ビット)</td><td rowspan="2">ウィンドウサイズ
(16 ビット)</td></tr>
<tr><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td></tr>
<tr><td colspan="6">チェックサム(16 ビット)</td><td colspan="2">緊急ポインタ(16 ビット)</td></tr>
<tr><td colspan="8">(オプション)</td></tr>
<tr><td colspan="8">データ</td></tr>
</table>

図2　ＴＣＰヘッダ

３．１．２　UDP プロトコル

ＵＤＰはコネクションレス型のトランスポート規約であり、最小限のオーバーヘッドで単一のメッセージを送受信することができる。つまり、信頼性よりもスピードを重視したつくりになっている。ＵＤＰのデータグラムはＩＰデータグラムで送信されるために、ＩＰのエラーによって消失する可能性があるが、ＴＣＰのように信頼性の保証が行われていないために再送信などの処理はされない。ヘッダ部分もＴＣＰとは違い非常に簡素なものとなっている。

1　　３２

発信元ポート番号 (16 ビット)	宛先ポート番号 (16 ビット)
データ長 (16 ビット)	チェックサム (16 ビット)
データ	

図３　ＵＤＰヘッダ

３．１．３　TCP と UDP の比較

TCP と UDP にはそれぞれ長所と短所があり、それぞれの長所によってトランスポートプロトコルがさまざまな要求に応えられるようにしている。信頼性が求められる通信では TCP に長所があり、またリアルタイム性を見るならば UDP のほうが有効性がある。（下図参照）

また、TCP には UDP にはないネットワークの制御情報をもっており、ネットワーク上にはこの制御情報が行き交っている。この実際にネットワーク上を行き交っている制御情報をとらえることがネットワークの状態をしることにつながる可能性があると考えられる。

３．１．４　トランスポートプロトコルをつかった輻輳検知

以上のトランスポート層にある主要なプロトコルプロトコルの紹介により、輻輳の検出に有効と想定される方法としては、TCP に含まれているの制御情報をあつかったヘッダ内のビットパターンをパケットキャプチャリングすることによって読み出し、その情報をまとめることによって輻輳を検知できないかというものである。（詳細は第４章参照）

インターネット電話	TCP	UDP
メディア情報	△	◎
制御情報	◎	△

インターネット放送	TCP	UDP
オンデマンド放送	○	○
ライブ放送	×	◎
マルチキャスト	×	◎ (クラスD)
制御情報	◎	○ (カルーセル)

図４　ＴＣＰとＵＤＰの比較

３．２　パケットキャプチャライブラリ

本研究の核である TCP フラグを検出するために、「**WINPCAP**」というパケットキャプチャドライバをプログラムに組み込んだ。WINPCAP は WINDOWS 上でパケットキャプチャを行うのに必要なライブラリである。

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

３．３　ネットワーク負荷発生ツール

IP Load 特定の TCP あるいは UDP ポートを使い、ユーザーによって定義されたデータパケット（多数の IP フレーム）を個別にアドレスに送る。.このソフトウェアは、ネットワークをテスト、デバッグする手段として開発された。今回はこのツールを使いネットワークの負荷を発生させ、その輻輳の測定を行う。このソフトは以下で入手した。

入手先：　http://www.bttsoftware.co.uk/ipload.html（BTT　SoftWare）

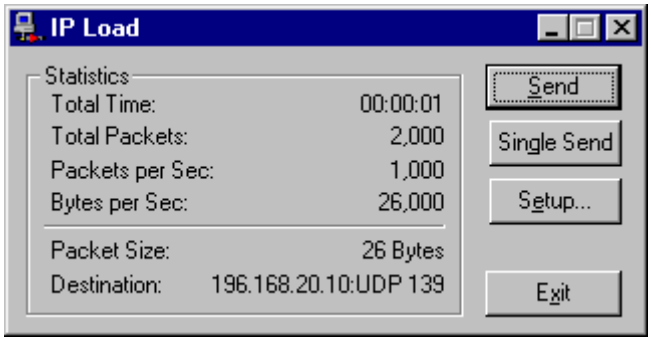


図５　IPLOAD

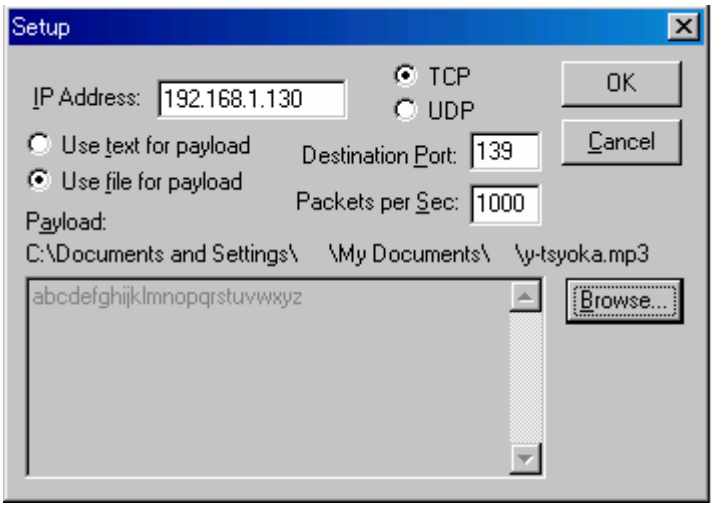


図６　IPLOAD 設定画面

設定項目を以下に示す

IP　Address	：送る先の IP アドレスを指定する
Destination Port	：Port 番号を指定する
Packets per Sec	：１秒間に送るパケット数を指定する
Browse	：送るデータファイルを指定する
TCP or UDP	：送る時のプロトコルを設定する
Use (text or file) for payload	：送出データが Text 形式か Binary 形式かを指定する。また、この項目でパケット長の調整を行う。

３．４　モニタリングソフト概要

ここでは本研究で作成した制御フラグのモニタリングソフトについて述べる。本プログラムは前述（３．２）の WINPCAP ライブラリを用いて作成した。今回作成したモニタリングソフトの機能は大きく分けて以下の機能である。

・パケットキャプチャ機能
・プロトコル解析機能
・重複 ACK 判別機能

パケットキャプチャリング機能は、ネットワーク中に流れてくるパケットをコンピュータに取り込む機能のことである。この機能によってコンピュータは本来 OS で処理される単位であるパケットを見ることができるようになる。

次にプロトコル解析機能であるが、一般にパケットを取り込んだパケットがなんのプロトコルを使っているかを判別する機能である。この機能によってほしいプロトコルを使ったパケットをキャプチャしたり、いらないプロトコルを使ったパケットをフィルタリングをすることが可能である。

最後に重複 ACK 判別機能については後述（４．２．２および５）されているが、ここで少し触れておくことにする。一般に重複 ACK はパケットが失われた場合などで受信コンピュータで発行されるフラグのことである。このフラグを数え上げることはそのままパケット損失率につながる。それは輻輳とも関係のあることなので、この機能が本プログラムで一番重要視される。また、この機能をつかい、通常の ACK とこの重複 ACK の割合をとることで輻輳の検知に役立てている。

第４章　TCP 制御フラグによる輻輳検知

今回の研究ではトランスポート層に位置するプロトコルのうち、TCP（トランスミッションコントロールプロトコル）を使用し、そのプロトコルに含まれる制御フラグの情報をもとにネットワークの輻輳の検知を行う。（参考文献[1][2]）

４．１　制御フラグによる輻輳検知の原理

TCP には TCP ヘッダ内部に６ビットのフィールドが用意されている。そのフィールドには、ネットワークの接続の確立や切断、応答を示すための情報を含んでおりそれらのフラグの情報を元にしてネットワーク接続の信頼性を確立している。制御フラグはネットワーク中にさまざまな形で存在しているので、TCP トラフィック中に TCP 制御フラグがどのような割合で含まれており、また輻輳状態になったときにどのように変化をしていくのかをキャプチャリングする。

輻輳検知手法はいくつか存在するが、この制御フラグを使った輻輳検知手法のメリットとしては、従来に検知手法にあるような計算式は使わずに、単純に単位時間あたりの制御フラグの個数を数え上げることにある。そして、制御フラグの出現比率などを測定することによって、輻輳を導き出す。検知する制御フラグは以下のとおりである。

表３　TCP 制御フラグ

FIN	送り手のデータ送信要求が終了したことを伝える。
SYN	コネクションを開始するために同期を取る。
RST	コネクションの一方的な切断に用いる。
PUSH	バッファに取り込まれたデータを上位層に送る。
ACK	応答確認番号を有効化される。
URG	緊急/非常とされるときに用いられる。

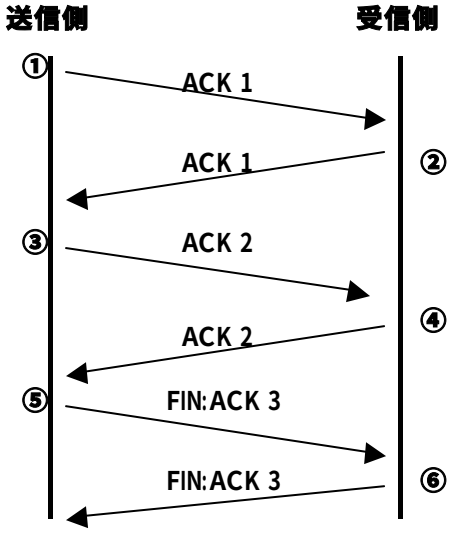
これらのうちで、ACK フラグとそれに付属してくる応答確認番号を利用して、実験を行った。

４．２　ACK を用いた検知原理

４．２．１　ACK（Acknowledgement）

ACK とは肯定確認応答のことで送信側から送信されたパケットがちゃんと送られているかということを確認するために送られるフラグである。このフラグの特徴としては送信側から送られる一番最初のコネクション開始要求以外のすべてのパケットに付属するということである。また、送られてきた ACK が正しいのかどうかを判断する基

準としてはTCPヘッダに付属している応答確認番号を利用する(図２TCPヘッダ参照)。このメカニズムは次のように説明される（図７)。また、応答確認番号は次に受信すべきシーケンス番号を表している。



①まず、送信側から ACK１が送られる（１は応答確認番号）
②受信側が ACK を受け取ると送信側に受け取った旨として同じ番号を送り返す
③送信側はつづけて番号を更新し ACK２を送り出す。
④受信側も受け取った番号を返す
⑤パケットの最後になると終了の合図である FIN と同時に ACK３を送る
⑥同じように FIN と ACK３を返す。

図７　送受信間のやりとり

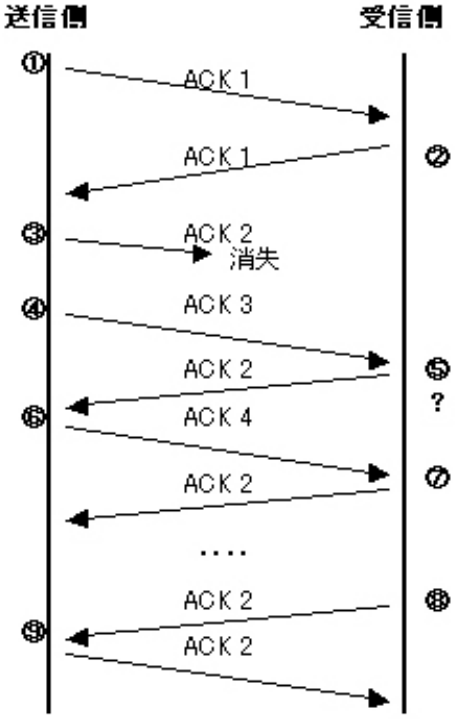
４．２．２　重複 ACK

ACK はすべてのパケットに付属しており、確認応答番号を確認しながら送受信を繰り返している。しかし、確認応答番号が予想していたものと違った場合においては送受信の動作が変わってくる。このときの確認応答番号が違っていた例としてはパケットの損失があげられる。

送信側から送られたパケットが受信側につく前に損失してしまうと、受信側は受け取った旨を表す ACK を送信側に送ることができない。送信側では受信側からの ACK を待たなくても送信ができるため（オーバーヘッドの軽減のため）次のパケットを送ってしまう。このときに確認応答番号は次の番号になってしまう。このためにこのパケットを受け取ったときに受信側では、受け取っていない ACK 確認応答番号を送信側に返す。

以後、送信側が失われたパケットを再送出してくるまでに受信側は失われたであろうパケットの ACK 確認応答番号を送りつづける。このときに確認応答番号が同じものを送るために「重複 ACK」と呼ばれる。

重複ACKが発行されるとそれだけパケットの損失が出ているということなので失われたパケットの指標となりうると考えられる。またそのパケット損失率が高ければ高いほどネットワークが混雑していると考えられ、それがエンドノードが輻輳を計るひとつの要因になるのではないかと考えられる。



①まず、送信側から ACK 1 が送られる（1 は応答確認番号）
②受信側が ACK を受け取ると送信側に受け取った旨として同じ番号を送り返す
③送信側はつづけて番号を更新し ACK 2 を送り出すがパケットが消失。
④続けて送信側が ACK 3 を送信
⑤このとき受信側では ACK 2 が届いていないので ACK 2 を再送信するように再送要求を出す。
⑥受信側が ACK 番号を間違っている可能性もあるので様子を見るために次の番号を送る。
⑦・⑧届いていない番号を送りつづける。
⑨何度送っても同じ答えが返ってくるので ACK 2 を再送する。

図８　重複 ACK のメカニズム

４．３　モニタ動作

ここでは作成したフラグを検出するモニタの動作を説明する。モニタのフラグ検知からその解析までの手順は以下のとおりである。

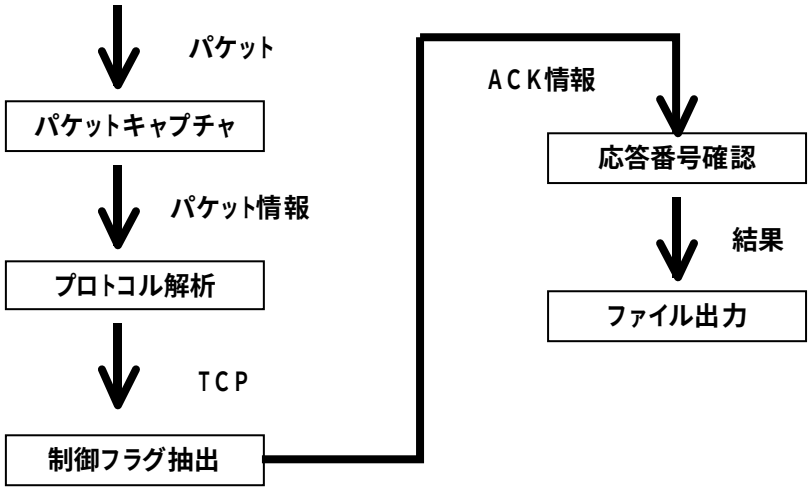


図９　モニタの動作

① パケットキャプチャ

一番最初に行うことは流れているパケットを検出するパケットキャプチャである。今回作成したモニタでは流れてくるパケットをプロトコルがなんであれ、無差別に検出するようにしている。

② プロトコル解析

パケットをキャプチャリングすると次にそのパケットのプロトコル解析を行う。この機能を使ってプロトコルを判別し、プロトコルのフィルタリングを行う。今回では必要としているプロトコルはＴＣＰ/ＩＰであり、それ以外のプロトコルを使ったパケットはフィルタリングされることになる。

③ 制御フラグ抽出

プロトコルによって分けられたパケットは、その中に含まれる 6 つの制御フラグの検出を行うことになる。

④ 応答番号確認

制御フラグに分けられた後は、パケットにＡＣＫが付加されていたときにその応答番号の確認を行う。ここで、送信側から送られてきた番号と受信側から送った番号に食い違いがあったときはそのＡＣＫは重複ＡＣＫとみなされカウントされる。

⑤ 結果出力

モニタが設定した時間によって、フラグの状態とＡＣＫの割合を出力する。ここで出力されたものを結果として、比較・考察をおこなう。

４．４　実験環境

実験は研究室内の LAN を使って行った。実験に用いた LAN 構成は以下のとおりである。なお、実験は外部を用いずに内部のみで行った。

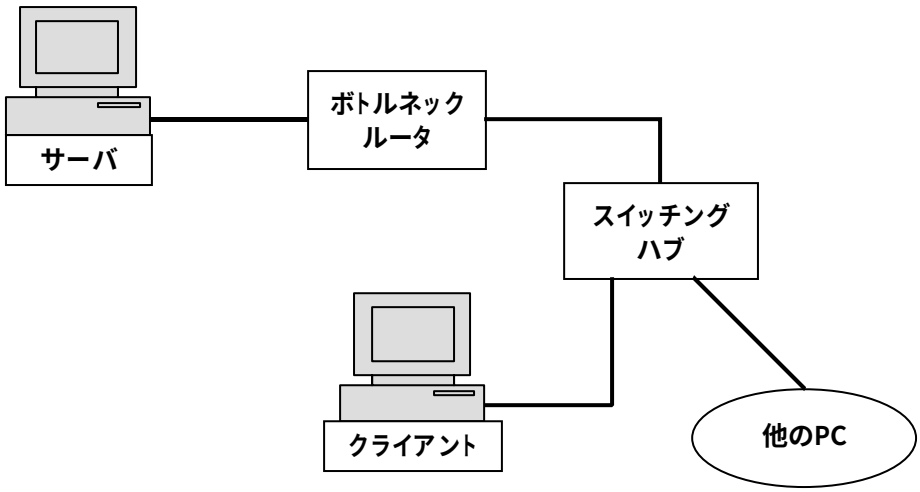


図１０　研究室 LAN 環境

また、実験に使用した PC のマシンスペックは以下のとおりである。

表４　使用環境スペック

環境	サーバー	クライアント	ルータ
CPU	PentiumⅢ 800MHz	PentiumⅢ 600MHz	i486 互換 100MHz
MEMORY	２５６MB	２５６MB	8MB
HD	１０GB	３０GB	ROM 2MB
LAN	１００(10)Base/T	１００(10)Base/T	１００（10）Base/T
IP アドレス	10.210.8.16	192.168.1.130	10.210.8.12

４．５　実験方法

サーバ・クライアント間の環境において、そのネットワーク上に負荷を与えてその様子をパケットキャプチャリングし制御フラグの変動・相関関係を調べることとする。

実験方法としてはサーバーからクライアントにデータを送り、その中でネットワークに負荷を定期的に与えるようにする。負荷を与えたときにクライアント側で TCP 制御フラグの測定をし、それをファイルに書き出すようにしている。ネットワークに負荷を与えるのには「IPLOAD」（３．３　ネットワーク負荷ツール参照）を使用した。IPLOAD の設定は以下のように設定した。

① CBR（固定長）１．５KB のデータを送りつける
② １秒間に送るパケット個数を輻輳が起こりうる値まで次第に増加させていく
③ TCP を用い、またポートは１３９またはそのときの空きポートを使用する

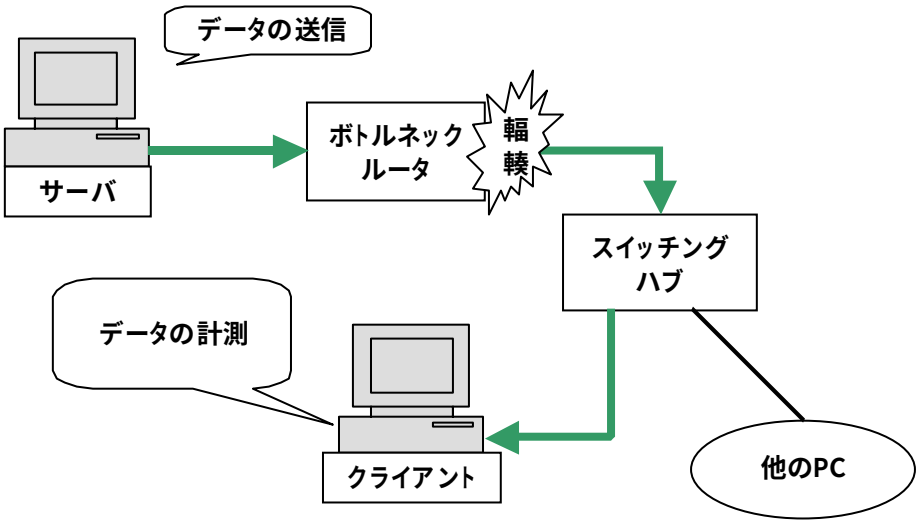


図１１　データの流れ

第５章　実験方法、結果および考察

実験としては平常時と負荷時の２つの状態時によるネットワーク中の制御フラグの総量の変化を見るために行った。平常時の実験ではネットワークに負荷をかけていない状態で、逆に負荷時の実験では特定の時間帯に負荷をかけるようにしてその変化の比較を行った。実験方法の詳細は第４章４項を参照。

５．１　実行結果

５．１．１　平常時の計測結果

最初に負荷をかけていない状態での TCP 制御フラグのキャプチャリングを行った。このときは負荷が起こらないようなレベル（既知の回線容量以下の転送量）で行った。また、このときにＡＣＫと重複ＡＣＫとの割合の変化も同様にチェックを行った。その様子が以下の図である。まず、図１２．１では制御フラグ全体の値の変化である。図１２．１を見てみるとわかるように平常時のときであってもＡＣＫの状態の変化が一番よく現れていることがわかる。

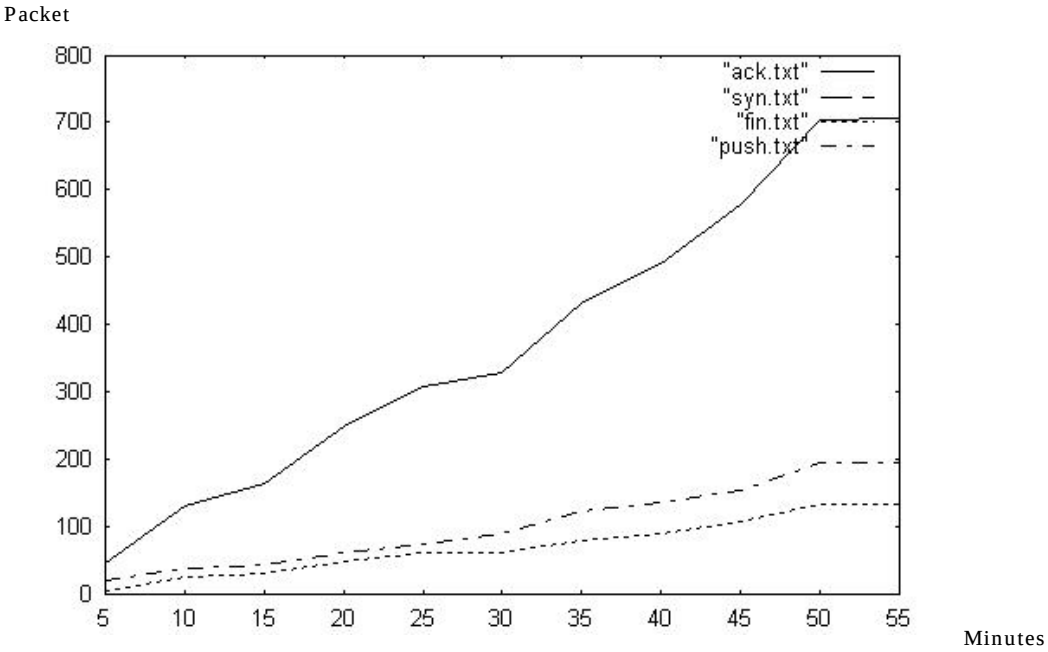
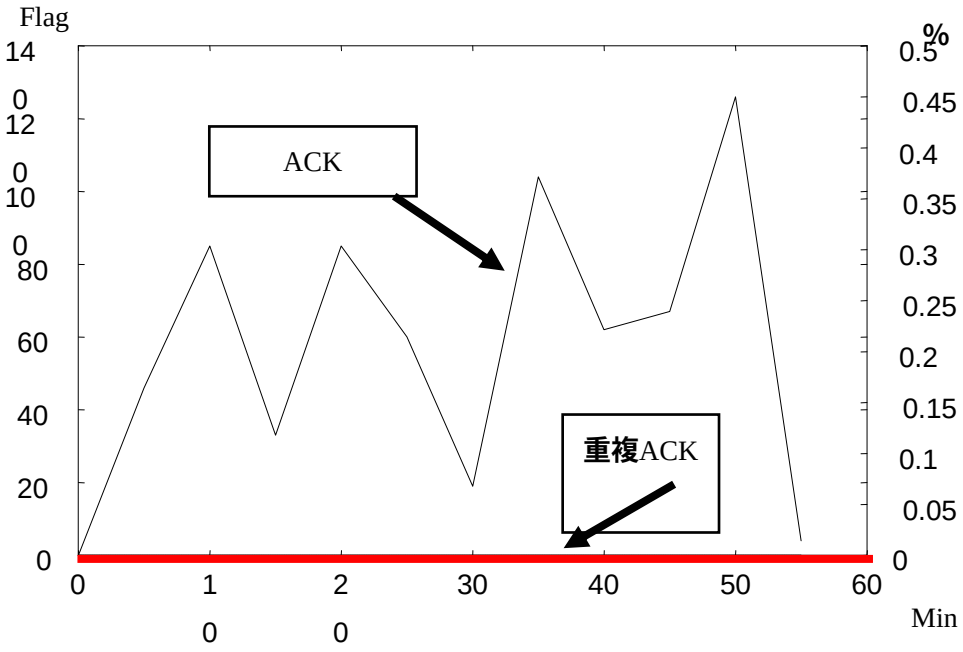


図１２．１　全体のフラグの累積値の遷移

図１２．１の結果より次にＡＣＫの変化について調査を行った。そのときの結果が図１２．２である。左軸はＡＣＫのフラグ個数、右軸はＡＣＫにおける重複ＡＣＫの割合を示している。

図１２．２　ＡＣＫと重複ＡＣＫ



このときの変化を見てみると、ＡＣＫは検出されているが重複ＡＣＫは検出されることはなかった。重複ＡＣＫはパケット損失がおきたときに現れるので、それがそのままパケット損失率に近くなる。また、パケット損失率はそのまま輻輳の判断材料につながるので、これらの結果からは負荷をかけていないレベルの状態の時では「パケットの損失が起きていない」＝「輻輳が起こっていない」と判断できる。

５．１．２　負荷時の計測結果

通常のネットワークの状態時で測定を行っているときに 30 分､50 分付近で負荷をかけることにした。このときも平常時の時と同じく５分ごとにフラグの量を求めその変化を記した。

結果を図１３．１で見てみると、負荷をかけた時間帯である 30 分と 50 分付近にはＡＣＫが影響を受けていることがわかる、他のフラグはＡＣＫに比べて変化の割合が少ないもしくはまったくないといった状態であることがわかる。

Packet

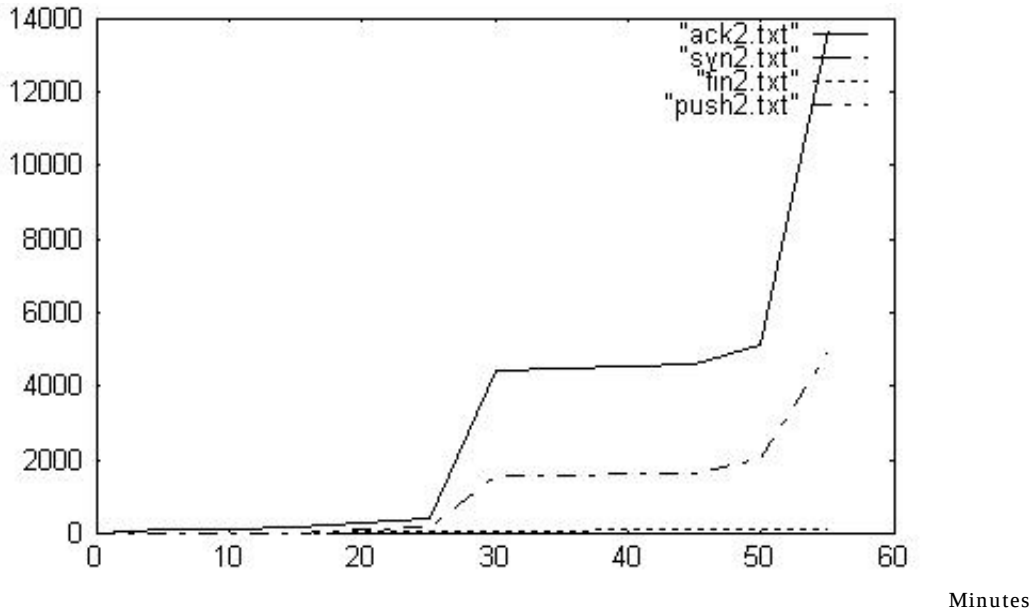


図１３．１　負荷をかけた時の状態

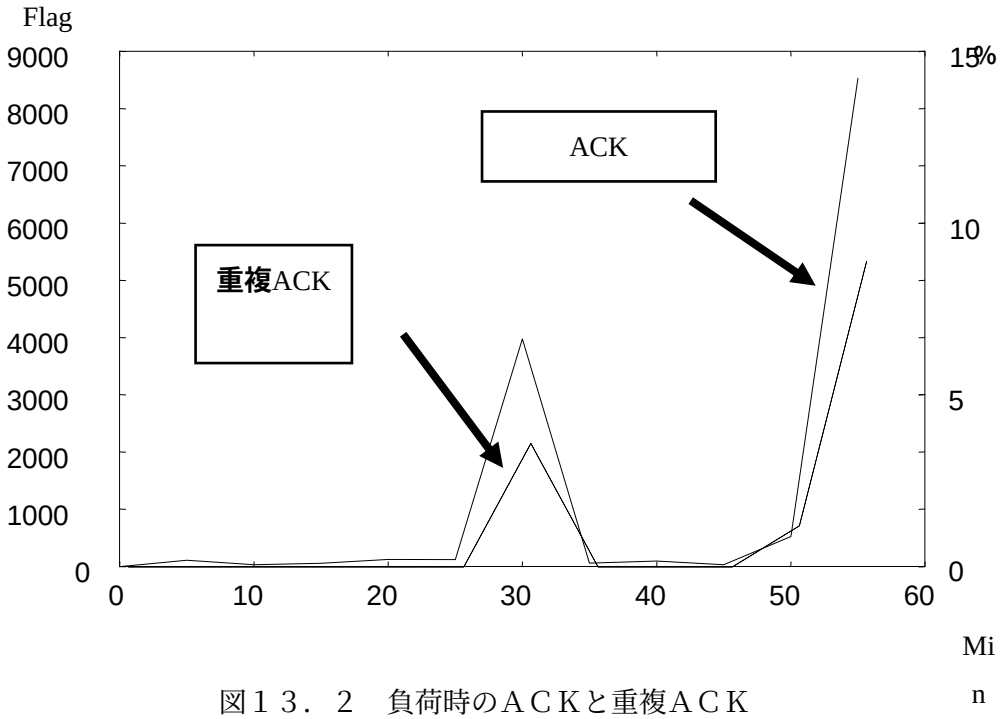


図１３．２　負荷時のＡＣＫと重複ＡＣＫ

全体のフラグの推移から次にＡＣＫと重複ＡＣＫの関係をグラフにした（図１３．２.)。平常時に計ったグラフとは違い、負荷時のＡＣＫと重複ＡＣＫの関係性が現れているグラフになっている。負荷をかけた時間帯にはＡＣＫが多数検出され、それにともなって重複ＡＣＫも検出されている。このグラフから見ると負荷時に輻輳が起こっていることを考えると、重複ＡＣＫが少なからず検出されるということである。

５．１．３　負荷時のフラグの状態

前項で出た結果では負荷をかけたときに重複ＡＣＫが検出されおり、その時間帯になにかイベントが起こっているということがわかった。これをもとに次に負荷のかかっている状態の時にＡＣＫフラグと重複ＡＣＫの状態がどのように変わっていくかを調べた。負荷時には制御フラグの状態を１分間ごとに記録し、測定による誤差を少なくするために測定を 5 回行い、その結果を近似線で表した。

以下にその結果を示す。

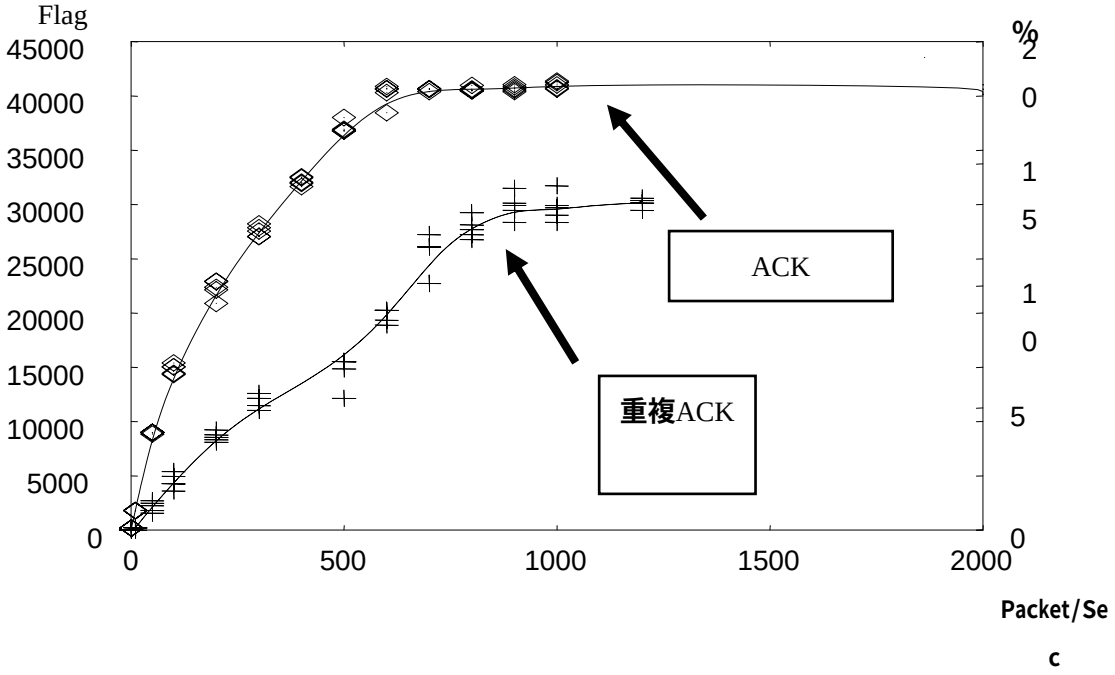


図１４．１　ACK フラグと重複ＡＣＫの推移

図１４．１では左軸にＡＣＫの状態を、右軸に重複ＡＣＫの割合を表している。このときの結果では、1 秒間に流すパケット数が１０００を超えたあたりで、接続の調子が悪くなり始め、接続が落ちることが多くなったために正確なフラグの検出ができなくなった。これはボトルネックとなっているルータがパケットを処理しきれなくなったために

起こった現象だと考えられる。

また、図ではＡＣＫと重複ＡＣＫの関係性をあらわしたものであるが、この結果としては 1 秒間に流すパケットの個数（ＡＣＫ）とともに重複ＡＣＫの割合も増えていっている。これはルータが単位時間に処理するパケットが増えるにしたがって、ルータで落ちるパケット（ルータが処理しきれないパケット）も増えているということをあらわしている。

次に回線利用率（式番号[2]）と重複ＡＣＫの関係を示した（図１４．２）。この結果としても回線使用率の変化に従って、重複ＡＣＫの割合も増加していることがわかる。

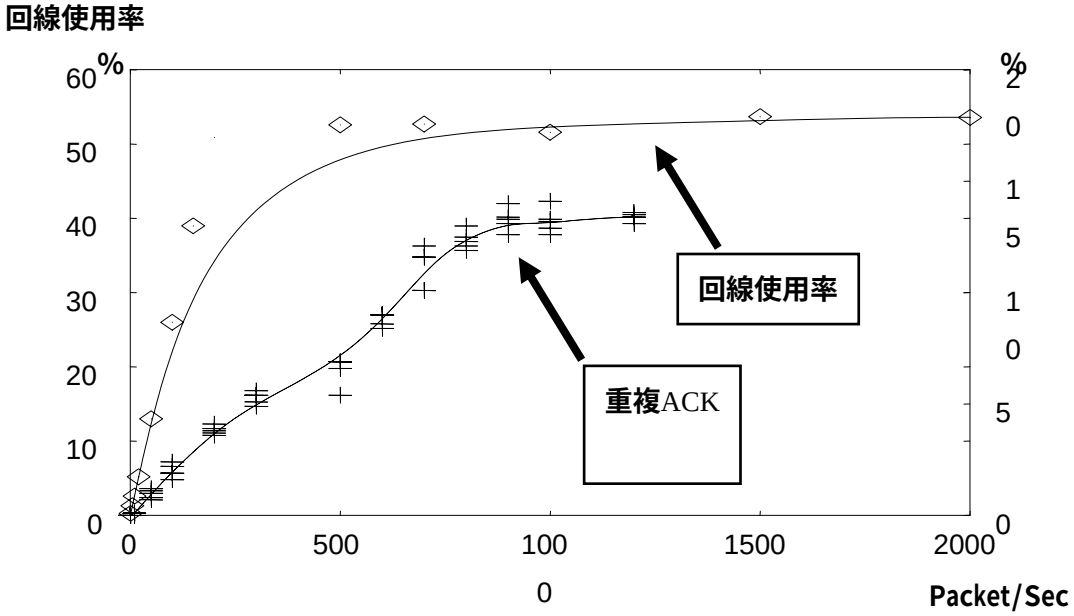


図１４．２　回線使用率と重複ＡＣＫ

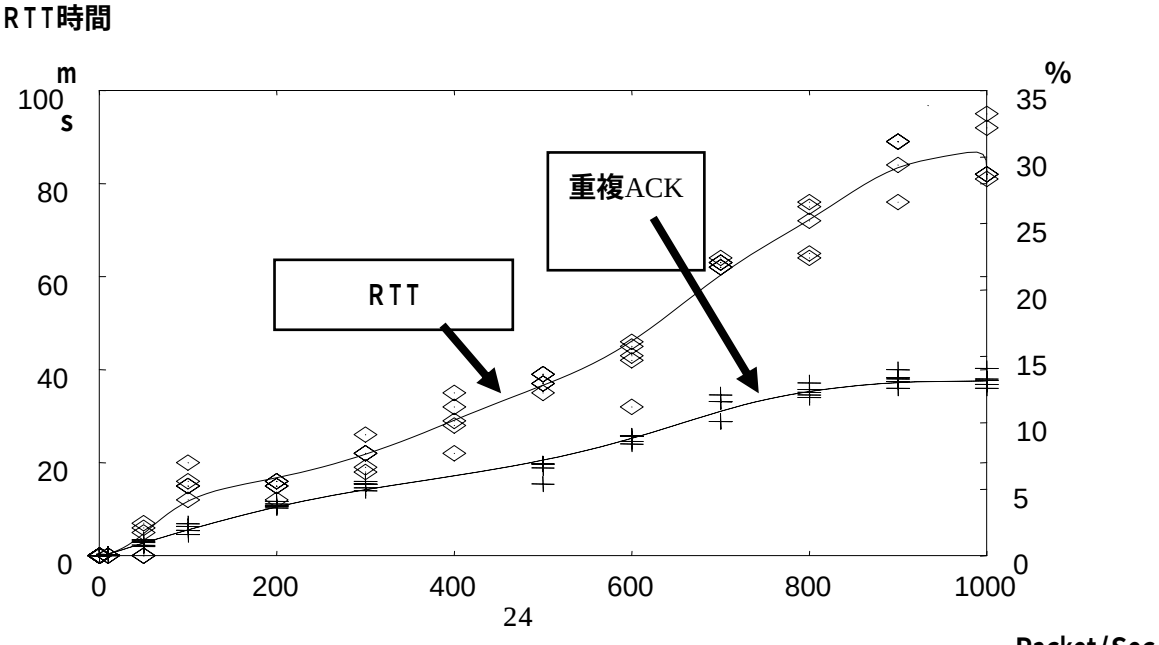


図１４．３　ＲＴＴと重複ＡＣＫ

最後に、ＲＴＴと重複ＡＣＫとの比較を行った。その結果が図１４．３である。ＲＴＴは前述（第 2 章 2 項）のように輻輳の度合いを測るもののひとつとして用いられている。この結果としては、ＲＴＴの度合いが大きくなるにつれて、すなわち輻輳がおき始めるに従って重複ＡＣＫの出現する率も増加していっているということがわかった。

５．２　考察

実験結果を見ると、輻輳が起こり始めるイベントが起こると（パケットの増加、回線使用率の増加、ＲＴＴの増加)、それに伴って重複ＡＣＫの割合も増加している。重複ＡＣＫが増加する理由として考えられることは受信側に届くはずのパケットの損失である。重複ＡＣＫは送信側から送られたパケットと受信側で受け取ったパケットとの不一致により受信側が発行するもので、送信側に再送要求として出される。そのときにこの再送要求として出したパケットも損失してしまうと、送信側では再送要求が到着せずに間違って新たなパケットを送ってしまう。そのために受信側では重複ＡＣＫを発行しつづけなければならない。このために全体のフラグにおける重複ＡＣＫの割合が増えると考えられる。

また、実験結果より重複ＡＣＫフラグの比率は回線の負荷率、すなわち輻輳の状態によって変化するという関係性が明らかになった。重複ＡＣＫは受信側で発行されるために、重複ＡＣＫによって輻輳がある程度わかるということは受信側で輻輳の傾向がつかめるということになる。今回の結果ではエンドノードでは輻輳の状態がわかりにくいということを解決するひとつの策になりうると考えられる。

今回の結果で注目したいのは、ＲＴＴとの関係である。回線利用率などは実験した規模が小さかったので推測はできたが、大規模になるとその関係は未知数になる。ＲＴＴはそれ自体がひとつの輻輳の指標であるので重複ＡＣＫとの関係に適しているといえる。ＲＴＴと重複ＡＣＫとの瞬間の増加率を求めることによってある程度は求まると考えられる。

５．３　まとめと展望

本研究では研究室内の小規模の LAN で行ったこともあり、ＬＡＮで使われている回線使用率、回線容量などは予測することができたが、実際に大規模なネットワークではそうすることがむずかしい。そのために、さらに広いネットワーク上で実際に使えるかを実験する必要があるだろう。また、現在使用しているパケットキャプチャソフトではキャプチャリングできる度合いが計測する PC の CPU に左右されるために過剰に TCP 制御フラグが送られるとフラグのキャプチャリングがしっかりと行われない恐れがある。そのようなことがおきないようなソフトウェアの改良の必要性もある。

また今後の展望として、エンドノードでのある程度の輻輳が検知できることになることによって、たとえばストリーミングデータの接続速度などを自由にエンドノードが選ぶことができるようになり、回線が混雑してきたときなどに即座にビットレートを変更できるようなシステムに応用できることが期待される。また、輻輳が起こっているノードがどこなのかを検知することによって、経由期間の経路選択にも使えるのではないだろうか？

最後にネットワークの発展によって、回線容量も増え、ルータなどの処理能力も上がっていきインフラが整えられていくだろうが、それに比例してトラフィックの量は増加の一途をたどるだろう。そのときにどの区間でどの程度の輻輳が起こっているのか検知し、それを制御する必要性がある。もし、ノードごとに検知・制御できるようになればより快適にネットワークを使うことができるのではないだろうか。

謝辞

本研究において最後まで熱心な御指導、叱咤激励をしていただ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、本研究室所属の大学院生である高木さん、長田さん、喜代吉さん、また同じ学部生の和田さん、下田君、本村君のさまざまな助言、協力に感謝します。特に高木さん・喜代吉さんには実験方法についてなど非常にためになる助言をいただきありがとうございます。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。

参考文献

[1]詳解 TCP/IP　Vol.1　プロトコル　W･リチャード･スティーブンス　著､橘　康雄　訳､
　井上　尚司　監訳
[2]コンピュータネットワーク　池田　克夫　編著
[3]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

引用元・説明

WINSock

http://www.katto.comm.waseda.ac.jp/~katto/Class/GazoTokuron/　（早稲田大学）

IPLoad

入手先：　http://www.bttsoftware.co.uk/ipload.html（BTT　SoftWare）

WINPCAP

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