通信システムの内部動作を推定するインテリジェント TCP アナライザ

大岸智彦† 井戸上彰† 加藤聰彦†

近年，インターネットの普及にともない，TCP/IP プロトコルに従った通信が広く行われているが，ネットワークの輻輳などによりスループット低下が生ずる場合がある．これらの原因を調査するため，筆者らは，対象とする通信システムの TCP モジュールの内部動作を推定することにより，通信の詳細を解析する「インテリジェント TCP アナライザ」を開発した．本稿では，本アナライザの要求条件，詳細設計，ならびに，本アナライザによる TCP の動作解析例について述べる．

Intelligent TCP Analyzer Estimating Behaviors of Communication Systems

Tomohiko Ogishi,† Akira Idoue† and Toshihiko Kato†

Recently, as Internet growing up, the communications according to TCP/IP protocols are widely used, however, there might be a chance the reduction of throughput occur due to the network congestion. In order to inspect the causes of the reduction, we developed "Intelligent TCP Analyzer" which analyzes the detail of communication by estimating the behaviors of TCP protocol module of the communication system which causes problem. In this paper, we describe the requirements, the detailed design and the implementation of Intelligent TCP Analyzer, and also examples of the analyses of TCP behaviors as a result of an actual usage.

1. はじめに

近年，コンピュータ通信において TCP/IP プロトコル[1)]が広く利用されてきているが，多くのユーザは通信手順の詳細を意識することなく，これらの通信機能を使用している．しかしながら，TCP/IP のプロトコルスタックのうち，WWW サーバアクセスや電子メールなど多くのアプリケーションで使用される TCP（Transmission Control Protocol）では，ネットワークの輻輳や伝送誤りによりパケット紛失が発生する場合や，双方の通信システムにおいてソケットバッファサイズなどのパラメータ値が異なる場合[2)]に，スループット低下などの問題が発生する場合がある．

このような TCP 通信で発生した問題を調査するためには，通常，市販のプロトコルアナライザ[3)]が用いられる．これらのアナライザは，ネットワーク上を流れるフレームを収集し，個々のフレームのフォーマットをプロトコルごとに解析する機能を有する．しかしながら，プロトコルアナライザを用いて TCP の通信手順を解析するには，次のような問題点がある．まず，プロトコルアナライザは，フォーマット解析のみを行うため，対応する入出力のパケットの解析，たとえば，データパケットとその応答確認パケット（以降 ACK パケットと呼ぶ）の対応づけ，再送のデータパケット（以降再送パケットと呼ぶ）とオリジナルのパケットの対応づけ，データの順序を示すシーケンス番号がウインドウ内であるかどうかの検査など，詳細な解析作業の大部分を手作業で行わなければならない．TCP では，シーケンス番号が，パケット単位でなくデータの各バイトに付与され，また再送パケットの含むデータが必ずしもオリジナルパケットと同一ではないなどの特徴があるため，上記の解析作業を手作業で行うには，専門家が時間をかける必要がある．さらに TCP では，輻輳制御手順で使用される輻輳ウィンドウのように，ネットワーク上に送出されない制御パラメータが存在する．これらのパラメータに基づく通信状況の解析には，個々のパケットの送受信タイミングから通信システム内部の TCP モジュールにおけるプロトコル手順を推定して，各パラメータの値がどのように変化した

† KDDI 研究所
KDDI R&D Laboratories

かを判断する必要がある．市販のプロトコルアナライザによる解析結果のみから上記のようなパラメータ値の変化を推定するためには，専門的な知識と煩雑な作業が必要になる．

したがって，TCP 通信における問題の原因調査を容易にするためには，上記のような専門家が手作業で行わなければならない詳細な解析を自動的に行うツールが有効となると考えられる．そこで筆者らは，ネットワーク上を流れるパケットを収集し，その情報をもとに，各パケットの送受信の時点における通信システム内部の TCP モジュールにおける状態や内部変数の値を推定し，その結果を用いて，データパケットと ACK パケットの対応づけや，輻輳制御手順の起動などの内部動作の解析を行い，さらにその結果を GUI で表示するインテリジェント TCP アナライザを開発した[7)~9)]．本アナライザを使用するオペレータ（以降単にオペレータと呼ぶ）は GUI 表示により TCP 通信の詳細な状況を理解することができる．本アナライザは，ネットワークを流れるパケットを収集すると，そのパケットを，出力側通信システムの送信，受け取った側の通信システムの受信として扱うことにより，通信しあう双方のシステムの TCP の内部動作を推定する．さらに，ネットワーク上でのパケット紛失や遅延を推定し，TCP の内部動作の推定にフィードバックさせる機能を持つ．

TCP 通信を詳細に解析するツールはすでにいくつか提案されており[4)]，その中でも tcptrace[5)]や TCPAnaly[6)]などが，本稿で提案するインテリジェント TCP アナライザと関連が深い．tcptrace は，tcpdump などの標準的なトラヒックトレースをもとに，特定の通信システム間の転送データ量や再送データ量などの統計情報の算出や，TCP 通信のスループットや RTT の時間変動の表示が可能である．しかし tcptrace は TCP モジュールの内部動作のうち，輻輳ウィンドウの時間的変化などの推定は行っておらず，したがってインテリジェント TCP アナライザに比較すると解析可能な範囲が制限される．一方 TCPAnaly は，TCP 実装の種別を評価することを目的としており，データパケットの受信にともなう ACK パケットの送出タイミングなど，実装依存の動作も含めた TCP モジュールの内部動作を詳細に推定することを特徴としている．これに対し，インテリジェント TCP アナライザは，TCP 通信における再送や輻輳制御手順などにより，スループット低下などが生じた場合の現象の明確化，その原因調査のサポートを行うことを目的としており，それに直接必要とはならない ACK の送出アルゴリズムなどの実装依存の内部動作までは推定しない．逆に，TCPAnaly と異なり，TCP コネクションごとの状態や内部変数の値の推定値や，スロースタートの開始などの処理内容の推定結果を，パケットの送受信時点ごとに記録し，その情報をもとにオペレータに便利な GUI を提供する機能を有する．さらに，ネットワークを監視する点と通信システムの間に伝送遅延がある場合に対しても，遅延を考慮して内部動作の推定を行う機能を有する．

本稿では，筆者らが提案するインテリジェント TCP アナライザについて，その設計方針および詳細設計について述べるとともに，再送タイムアウト時間の初期値が伝送遅延に比べて小さい場合にスループットが低下した TCP 通信などの，実際の実験環境における本アナライザを用いた解析結果を示す．本稿は以下のように構成される．2 章では，インテリジェント TCP アナライザの要求条件と設計方針について述べ，3 章では，その設計方針に基づいた詳細設計について述べる．4 章で，本アナライザの実装および実験環境での動作解析例をあげる．5 章で考察を述べ，6 章で結論を述べる．

2. 要求条件と設計方針

2.1 要 求 条 件

インテリジェント TCP アナライザの設計にあたって，以下のような要求条件を想定した．

前述のように本アナライザは，スループットが低いなどの問題が生じた TCP 通信に対して，オペレータがその原因を調査するのをサポートすることを目的としている．このためには，問題のある TCP 通信に参加した通信システムに対して，データパケットに対してどのように ACK パケットを返答したか，データパケットを再送したか，輻輳制御手順を起動し，輻輳ウィンドウに従ってどのようにデータパケットを送信したかなどを解析可能である必要がある．すなわち本アナライザは，ネットワークを流れるパケットの情報に基づき，通信システムの TCP モジュールの内部動作を推定する機能を有する必要がある．さらに，このような推定結果は，パケット収集後に，オペレータにより多様な GUI を利用して，さまざまな角度から使用されることが可能である必要がある．

TCP モジュールの内部動作の推定機能を実現するにあたっては，以下の条件を満足する必要がある．

- 内部動作の推定は，TCP 通信の問題の原因調査を行うために必要な動作のみを対象とする必要がある．

- TCP には Tahoe や Reno [10] といった動作の異なるバージョンが存在する．このため，内部動作を推定する際に，通信システムがどのバージョンを実装しているかを判定できる必要がある．
- 問題のある TCP 通信に参加する通信システムのいずれが原因になっているかを調査できるよう，ネットワークを流れるパケットの情報から，参加した複数の通信システムの内部動作を解析できる必要がある．
- 本アナライザが監視する点と通信システムの間のネットワークにおいて，遅延やパケットロスが発生した場合は，通信システムの内部動作の推定においてそれらを考慮する必要がある．

なお，本アナライザのパケット収集機能は，実用性を考慮して，100 メガビット/秒の Ethernet または 155 メガビット/秒の ATM ネットワーク程度の伝送速度までは対応可能である必要がある．

2.2 設計方針

前節に示した要求条件に基づいて，インテリジェント TCP アナライザの設計方針を以下に示す．

(1) 本アナライザは，**図 1** に示すようなネットワーク構成で用いられ，特定の通信システム間（たとえば (A) (B) 間または (A) (C) 間）の TCP 通信に着目し，その通信に参加する通信システムの TCP モジュールの内部動作を推定する．

(2) オペレータによる解析結果の調査はパケット収集後に行われること，155 メガビット／秒程度までのネットワークに対応することなどを考慮し，パケットの収集のみをオンライン（リアルタイム）で行い，収集した情報に基づく通信システム内の TCP モジュールの内部動作の推定およびオペレータへの推定結果の表示は，オフライン（非リアルタイム）に行うものとする．

(3) 本アナライザでは，TCP 通信の問題の原因調査に必要な動作の推定のみを行い，異なるバージョンの TCP に対応するという要求条件を満足するために，以下のような方針で内部動作の推定機能を実現することとする．

- 本アナライザは，1 つの通信システムのパケットの送信および受信に対して，その TCP モジュールの状態や内部変数の値がどのように変化したかをトレースする処理を行う．パケットの受信に対しては，現在推定されている状態で，そのパケットを受信した場合の状態遷移を模擬し，状態や内部変数の値を更新する．その状態遷移において，応答のパケットを送信する場合は，そのパケットが実際に送信されているかを確認する．一方，パケットの送信に対しては，現在推定されている状態において，自発的にそのパケットを送信する状態遷移が存在すれば，その遷移が発生したと解釈し，状態と内部変数を更新する．
- パケット受信に対するパケットの送信については，問題の原因推定に必要なものを規定することとする．具体的には，SYN パケットに対する SYN+ACK パケットの送信や，誤った順序番号を持つデータパケットに対する ACK パケットの送信など，TCP のプロトコル仕様上必ず送信されるパケットは，パケット受信に対する状態遷移の一部であるとするが，正しいシーケンス番号のデータパケットを受信した場合の ACK パケットの送信は，一連の状態遷移として扱わず，データパケットの受信と，ACK パケットの送信は別の状態遷移とする．
- 一方，輻輳制御手順については TCP のバージョンに依存するために，本アナライザでは，基本的に，Fast Retransmit のみをサポートする Tahoe か，Fast Retransmit と Fast Recovery の双方をサポートする Reno，あるいは，いずれもサポートせず Slow Start と Congestion Avoidance のみを行う仕様（本稿では Other と呼ぶ）を基本とすることとする．すなわち，内部動作を推定する通信システムは Tahoe，Reno，Other のいずれかに従っているとして，内部動作の推定を行う．それ以外のバージョンの通信システムについては，想定する動作とは異なる旨のエラーメッセージを表示し，詳細な解析はオペレータにゆだねることとする．
- 上述のように通信システムの状態遷移をパケットの送受信のたびにトレースすること，複数のバージョンの TCP に対応することを可能とすることを考慮して，本アナライザは TCP の状態遷移を状態遷移表の形で実装することとする．ただし，この状態遷移表は，通常のプロトコル実装に使用されているものとは異なり，本アナライザが検出

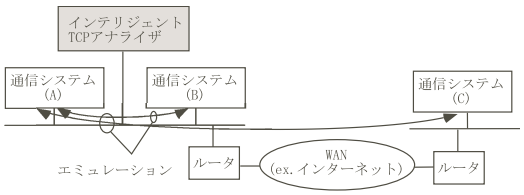


図 1 ネットワーク構成
Fig. 1 Network configuration.

するパケットの受信および送信に対して，TCP モジュールがどのように遷移するかを規定する．

(4) 通信システムでのイベントの順序を正しく推定するために，アナライザとその通信システム間の物理的な距離による伝播遅延や，パケットの長さに依存する転送遅延などを考慮し，パケットの送信または受信のイベントが発生する時刻を推定する．

(5) パケットの取りこぼしや推定したイベント順序の誤りなど，アナライザが推定したイベントシーケンス（イベントの列）が通信システムが実際に処理したものと異なる場合が考えられる[11]．本アナライザでは，このような状況の発生を検出するとともに，正しいイベントシーケンスを推定する．

(6) 状態および内部変数の出力結果をもとに，データとその応答確認の対応づけなどを示したシーケンス図や，内部変数の時間的変化を示したグラフなどを GUI を用いて表示する．

3. 詳細設計

3.1 ソフトウェア構成

本アナライザは，**図 2** に示すように，パケットの収集を行うオンライン部と，TCP の内部動作の推定および動作推定の結果の表示を行うオフライン部から構成する．

オンライン部は，ネットワーク上を流れるパケットを収集するパケット収集部と，IP および TCP のプロトコルに従い，パケットフォーマットを解析し，それらをオンラインログとして蓄積するパケット解析部から構成する．

オフライン部は，オンラインログをもとに，着目する通信システムがパケットを送受信した時刻および順序（以下，各通信システムにおけるパケットの送信および受信を，それぞれ送信イベントおよび受信イベントと呼ぶ）を推定するイベントシーケンス推定部と，イベントシーケンスに基づき，個々の通信システムにおける TCP の内部動作の推定を行う TCP 動作推定部，さらには，内部動作の推定の結果を GUI によりグラフィカルに表示する解析結果表示部から構成する．

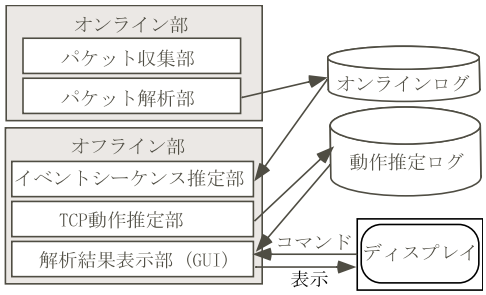


図 2 ソフトウェア構成
Fig. 2 Software structure.

3.2 オンライン処理

オンライン部では，ネットワーク上を流れるパケットの情報から，オフラインでの内部動作の推定に必要な情報のみを切り出し，オンラインログに蓄積する．まず，個々のパケットを収集した際に，パケット収集時刻を付加する．次に，個々のパケットは，本アナライザが設置されたネットワーク上でのフレームフォーマット（たとえば Ethernet 上なら Ethernet フレームなど）として収集されるため，その中から発着の IP アドレスと TCP ヘッダ部分の情報のみを切り出す．このとき，対象としない通信システムが送受信したパケットや，TCP 以外のパケットは，解析の対象外であるため破棄する．TCP のユーザデータは破棄し，ユーザデータ長のみを蓄積する．さらに，IP ヘッダや TCP のチェックサムの計算を行い，チェックサムの成否を蓄積する．

また，IP でのフラグメンテーションが施され，1 つの TCP のパケットが 2 つ以上のフレームに分割されて転送される場合がある．これに対処するため，フラグメンテーションを含む IP を検出した場合は，すべてのフラグメントを検出した後に，リアセンブリングを行い，TCP のパケットを再構築する．ただし，一定時間内にすべてのフラグメントが揃わなかった場合は破棄する．なお破棄するまでの時間としては，典型的には 5 秒程度で十分と考えている．

まとめると，オンラインログは，パケットごとの情報として，以下の情報から構成する．

- TCP パケットの先頭フレームの収集時刻と，TCP パケットの最終フレームの収集時刻
- 発 IP アドレスと着 IP アドレス
- TCP チェックサムを除く TCP ヘッダのパラメータ
- TCP ユーザデータ長
- IP ヘッダチェックサムおよび TCP チェックサムの正否を示すフラグ

3.3 イベントシーケンスの推定

イベントシーケンス推定部では，オンラインログをもとに，対象とする通信システムで発生した送信または受信イベントとその発生時刻を推定し，通信システムごとに，イベントを発生時刻順に並べたイベントシーケンスを作成する．個々のイベントは，オンラインログの要素のうち，フレームの収集時刻（TCP パケットの先頭フレームの収集時刻と，TCP パケットの最終フレームの収集時刻）を，その通信システムに

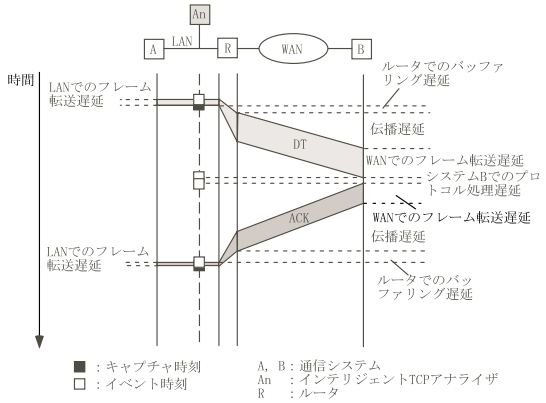


図 3 イベントシーケンスの推定
Fig. 3 Estimation of event sequence.

おけるイベント発生時刻に置き換えた情報となる．

アナライザ，通信システム間は，複数のルータやさまざまな物理速度の回線を経由する場合が考えられ，イベント発生時刻の計算には，物理的な距離に基づく伝播遅延，そのパケットを回線上に送出するのに要するフレーム転送遅延などを考慮する必要がある．**図 3** に示すように，アナライザはパケットを受信し終えた時点で，タイムスタンプを刻む．したがって，通信システム A，B におけるイベント発生時刻は，送信/受信のいずれのイベントであるかに従って，図 3 により求められる．したがって，通信システム B でのトータルの遅延は，近似的に，パケット長に比例する一次式で計算できる．

なお，IP フラグメンテーションが行われない場合は，オンラインログにおいて，TCP パケットの先頭フレームの収集時刻と，TCP パケットの最終フレームの収集時刻は同一となる．このため，イベント発生時刻はそのフレーム収集時刻をもとに計算する．また，フラグメンテーションが行われた場合は，送信イベントの発生時刻は先頭フレームの収集時刻をもとに，受信イベントの発生時刻は最終フレームの先頭時刻をもとに，それぞれ計算する．

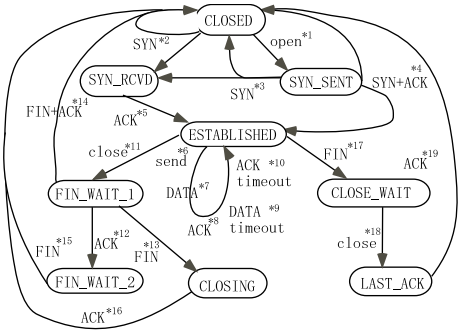
3.4 TCP の状態遷移表

設計方針に示したように，本アナライザは TCP の状態遷移を状態遷移表の形で実現する．この状態遷移表に使用される状態および内部変数を**表 1** に示す．

また，状態遷移表の一部を，**図 4** に示す．図中の遷移*1 では状態 CLOSED において SYN パケットを送信する遷移を示す．この遷移において，iss，snd_una などの内部変数を SYN パケットの中のパラメータの値に設定し，状態を SYN_SENT に移すことを示している．また，遷移*2 は，逆に，状態 CLOSED において SYN

表 1 状態および内部変数
Table 1 State and internal variables.

状態（state）			TCPの状態、値として、CLOSED, SYN_SENT, SYN_RCVD（コネクション確立時）、ESTABLISHED（データ転送時）、FIN_WAIT_1, FIN_WAIT_2, CLOSING, CLOSE_WAIT, LAST_ACK（コネクション解放時）を取りうる。
内部変数	送信制御用	snd_nxt	次に送信する送信シーケンス番号
		snd_una	送信済みの未確認の最小の送信シーケンス番号
		snd_max	これまでに送信した最大の送信シーケンス番号
		snd_wnd	相手側より通知されたウインドウサイズ
		iss	初期送信シーケンス番号
		mss	最大セグメントサイズ
		送信バッファ	未確認の送信データを蓄積するバッファ
	受信制御用	rcv_nxt	次に受信する受信シーケンス番号
		rcv_una	受信済みで未確認の最小の受信シーケンス番号
		rcv_adv	受信ウインドウの上限値
		rcv_wnd	受信ウインドウサイズ
		irs	初期受信シーケンス番号
		受信バッファ	未確認の受信データを蓄積するバッファ
	輻輳制御用	status	現在行われている輻輳制御手順（どれも行われていない場合も含む）
		cwnd	congestion window
		cwnd'	congestion window（Fast Retransmit後のTahoeの内部動作を推定するために設けたもの）
		ssthresh	slow start threshold
		dupacks	duplicate ACKを受信した数
	TCP_spec		推定中のTCPの仕様を表すフラグ．Tahoe, Reno, および，Fast Retransmitを行わないOtherの仕様に対応するフラグから構成される．



遷移*1: *open* in CLOSED
output: *SYN*; next state: SYN_SENT;
variable update:
iss=sequence number(SEQ) in *SYN*; snd_una=iss;
snd_nxt=iss+1; snd_wnd=window size(WND) in *SYN*;
mss=maximum segment size (MSS) in *SYN*;

遷移*2: *SYN* in CLOSED
1) output: *SYN+ACK* with acknowledgment number (ACKNUM)= SEQ in *SYN*+1;
next state: SYN_RCVD;
variable update:
iss=SEQ in *SYN*; irs=SEQ in *SYN+ACK*; snd_una=iss; snd_nxt=iss+1;
snd_wnd=WND in *SYN*; rcv_nxt=irs+1; rcv_wnd=WND in *SYN+ACK*;
mss=min(MSS in *SYN*, MSS in *SYN+ACK*); or
2) output: *RST+ACK* with SEQ=0 and ACKNUM=SEQ in *SYN*
+TCP data length (LEN) in *SYN*;
next state: CLOSED;

遷移*4: *SYN+ACK* in SYN_SENT
1) output: *ACK* with SEQ=snd_nxt and ACKNUM=SEQ in *SYN+ACK*+1;
next state: ESTABLISHED;
variable update:
irs=SEQ in *SYN+ACK*; snd_una=ACKNUM in *SYN+ACK*;
snd_wnd=WND in *SYN+ACK*; rcv_nxt=irs+1;
rcv_wnd=WND in *ACK*;
mss=min(current mss, MSS in *SYN+ACK*); or
2) output: *RST* with SEQ=ACKNUM in *SYN+ACK*;
next state: CLOSED;

（注）遷移の詳細については一部抜粋

図 4 TCP の状態遷移
Fig. 4 State transition.

パケットを受信した遷移を示す．この遷移においてはSYN+ACK パケットを送信して状態 SYN_RCVD に遷移する場合と，RST+ACK パケットを送信して，CLOSED に戻る場合の 2 つの可能性がある．またこの場合は応答として送信される SYN+ACK パケットまたは RST+ACK パケットのパラメータに条件が付与され，また各遷移に規定される内部変数の変更が行われる．

3.5 TCP の内部動作の推定

(1) **基本方針**

TCP 動作推定部では，イベントシーケンスをもとに，コネクションごとに，状態および内部変数を推定する．本アナライザでは，ネットワークの監視により通信システムの TCP モジュールの内部動作を推定するため，アプリケーションからの要求（たとえば，図 4 での open 要求）の存在や，ネットワークへの送受信をともなわない状態遷移を検出することはできない．したがって，実際にネットワークへ送出されたパケットをもとに，通信システム内部で発生したイベントを推定し，状態遷移を行う．

(2) **基本アルゴリズム**

TCP の状態遷移は，TCP パケットの送出，受信，TCP パケットの受信とその応答の送出を契機に行う．したがって，TCP の内部状態を推定するためには，次のように送受信イベントを処理する．

(a) 送信イベントは単独で処理する．

(b) 受信イベントを処理する場合は，以降のイベントシーケンスより送信イベントを抽出し，受信イベント単独，あるいは，受信/送信のイベントの組での状態遷移が可能か否かを検査する．

(3) **イベントシーケンスの推定誤りの対処**

本アナライザは，通信システム間のある 1 点でパケットの収集を行っているため，アナライザが推定したイベントシーケンスと，通信システムの TCP が実際に処理したイベントの順序が異なる場合が考えられる．具体的には，**図 5** に示すような場合が想定される．(1) の場合では，そのイベントを送信/受信する通信システム A，B とも誤った内部動作の推定を行ってしまうのに対し，(2) の場合では，通信システム B，(3)，(4) の場合では，通信システム A の内部動作の推定が実際と異なる結果となる．

このように誤った推定が行われる場合，次のイベント処理は，誤った状態/内部変数をもとに行われる．たとえば，(1) の場合では，アナライザは自分自身が取りこぼしたイベントを処理できないため，そのイベント処理前の状態/内部変数に基づき，次のイベントを処理する．

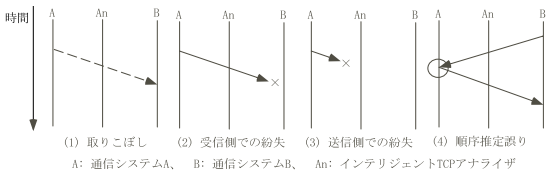


図 5 イベントシーケンスの推定誤り例
Fig. 5 Example of estimation error of event sequence.

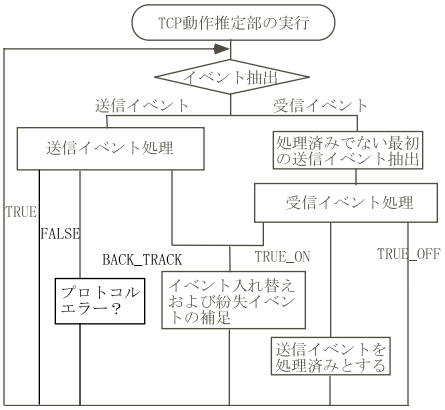


図 6 動作推定のアルゴリズム
Fig. 6 Algorithm for estimating behavior.

本アナライザでは，イベントシーケンスの推定誤りが発生した場合でも，正しく TCP の内部動作が追随できるように，推定失敗となる場合に対して，取りこぼし，送信側紛失，受信側紛失，ならびに，順序推定誤りの可能性を考慮し，次のように処理する．

(a) 取りこぼしと，送信側紛失の場合，次のイベントを抽出し，そのイベントに含まれるパラメータより，紛失したイベントを処理した後の状態/内部変数を推定する．

(b) 受信側紛失の場合，現在紛失イベントを処理している場合は，状態/内部変数の更新を行わずにイベント処理を終了し，また，紛失イベントの処理がすでに終了している場合は，一部の内部変数を，紛失イベントを処理する前の値に更新する．

(c) 順序推定誤りの場合，そのイベントを処理する前の状態/内部変数に戻し，イベントシーケンスを正しい順序に入れ換え，入れ換えた最初のイベントより，内部動作の推定を開始する．

(4) **内部動作の推定のためのアルゴリズム**

TCP モジュールの内部動作を推定するためのアルゴリズムを**図 6** に示す．初めに，イベントシーケンスログより逐次イベントを抽出する．送信イベントであ

るか受信イベントであるかにより，以下のように処理が分かれる．

送信イベントの場合は，初めに，そのイベントが送信可能か否かを判断するため，遷移の探索を行う．探索成功（TRUE）ならば，状態/内部変数の更新をともなう遷移を実行する．探索失敗なら，イベントシーケンス推定誤りの可能性を検出し，イベント紛失やイベント順序の推定誤りの可能性があれば（BACK_TRACK），前者の場合は，紛失したイベントを捕捉するための状態/内部変数の更新を行い，その後に，その送信イベントに対する遷移を行い，後者の場合は，イベントの入れ換え処理を行い，入れ換え前のイベントから，内部動作の推定を行う．これらの処理を行っても遷移の可能性がなければ，内部動作の推定に失敗したと判断し（FALSE），状態を不明とし，すべての内部変数を初期値に戻す．

受信イベントの場合は，初めに，処理済みでない最初の送信イベントを抽出し，それを受信イベントに対する応答と仮定し，受信イベント処理を行う．受信イベント処理では，初めに，遷移探索を行う．探索成功の場合，状態/内部変数の更新をともなう遷移を実行する．このとき，その遷移が，送信イベントを含めた遷移であれば（TRUE_ON），その送信イベントを処理済みとし，その旨をイベントシーケンスに記録する．送信イベントを含めない遷移の場合は（TRUE_OFF），受信イベントを単独で処理する．探索失敗なら，イベントシーケンス推定誤りの可能性を検出し（BACK_TRACK），イベント紛失の可能性があれば，紛失イベントを捕捉した後，受信イベントに対する遷移を実行する．そのイベントが受信側に届いていない可能性があれば，遷移を行わずに次のイベントの処理を行う．イベント順序の推定誤りの可能性があれば，イベントの入れ換え処理を行う．これらの処理を行っても遷移の可能性がなければ，内部動作の推定に失敗したと判断する．

3.6 解析結果の表示

TCP の内部動作の推定結果を分かりやすく表示するために，本アナライザは，以下のような解析結果の表示機能を持つ．

(1) イベント発生時刻，イベント種別，パケットに含まれるパラメータ値，状態/内部変数を，通信システムごとに，列挙して表示する機能．また，特定のイベントを検索する機能

(2) 2つの通信システム間での通信のイベントシーケンス図を表示する機能

(3) 対象とする通信システムに関し，シーケンス番号，輻輳ウィンドウ（congestion window: cwnd）などの内部変数の時間的変化を表示する機能

(4) Karn のアルゴリズム[1]に従い，再送パケットを除いたデータパケットと対応する ACK パケット間の RTT（Round Trip Time）の計測値を表示する機能

4. インテリジェント TCP アナライザの実装および動作解析例

筆者らは，上記の設計に基づき，インテリジェント TCP アナライザを実装した．実装は，Soraris 2.x, Windows NT 4.0 上で行った．本アナライザのソフトウェアのうち，パケット収集部の実装方法は，オペレーティングシステムに依存する．パケット収集のためのデバイスドライバインタフェースとして，Soraris 2.x では，UNIX カーネル上で動作する DLPI（Data Link Provider Interface）モジュール[12]，ならびに，各パケットのタイムスタンプを刻む BUF（Buffer）モジュールを用い，Windows NT 4.0 では，NDIS（Network Driver Interface Specification）[13]上の TDI（Transport Driver Interface）ドライバとして実装した．また，内部動作の推定機能などは C 言語により，GUI は JAVA により開発した．

さらに，本アナライザを用いて，実際の TCP 通信の解析を行った．**図 7** に示すように，Solaris 2.5.1 をオペレーティングシステムとして使用する通信システム間で，往復伝播遅延 1 秒の 1.5 メガビット/秒の回線上を経由して，ftp によるデータ転送をする実験を行った．通信システム B における ftp クライアントの応答によると，この通信は，約 20 キロビット/秒程度のスループットしか得られなかった．

図 8 には，本アナライザにより，双方の通信システムにおける各パケットのパラメータ，イベント処理時の状態/内部変数，再送の発生などのコメントなどを含む内部動作の推定結果を示す．左半分，右半分がそれぞれ通信システム A，B の推定結果を示している．**図 9** に，解析結果をもとに作成したイベントシーケンス図を示す．この図では，もとのデータパケット（以降 DT と呼ぶ）と再送の DT の関連や，DT と対応する ACK パケット（以降単に ACK と呼ぶ）の関連を矢印

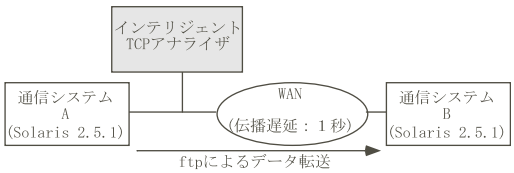


図 7 実験ネットワーク構成
Fig. 7 Experimental network configuration.

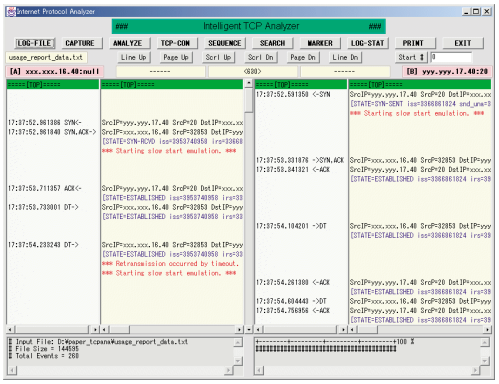


図 8 内部動作推定の結果
Fig. 8 Result of behavior estimation.

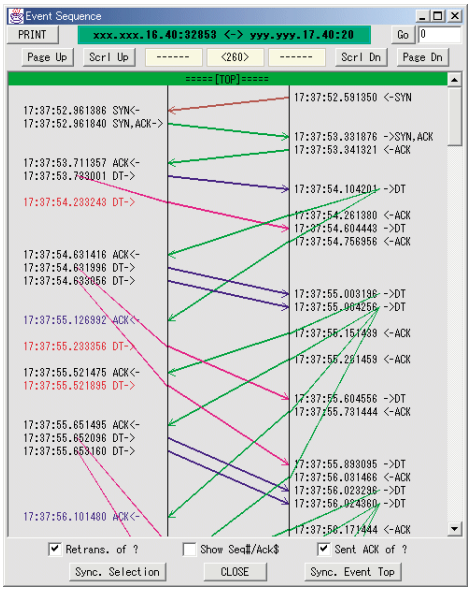


図 9 イベントシーケンス図
Fig. 9 Event sequence diagram.

で示している．これにより，通信システム A が最初に送信した DT が再送されていること，通信システム B が最初に受信した DT に対して，2 つの ACK が関連づけられている，つまり，2 つめの ACK は duplicate ACK であること，などが分かる．**図 10** では，解析結果をもとに作成した，シーケンス番号と cwnd の時間的変化を示す．これによると，たび重なる再送が発生しているため，シーケンス番号の変化は，グラフ上において，のこぎり型の線を描いている．cwnd も，再送時には減少するため，同様の折れ線となっている．また，10 秒でシーケンス番号が約 25 キロバイトになっていることから，20 キロビット/秒程度のスループットであることが分かる．

本アナライザによる解析の結果を用いて，スループット低下の原因は以下のように考えられる．

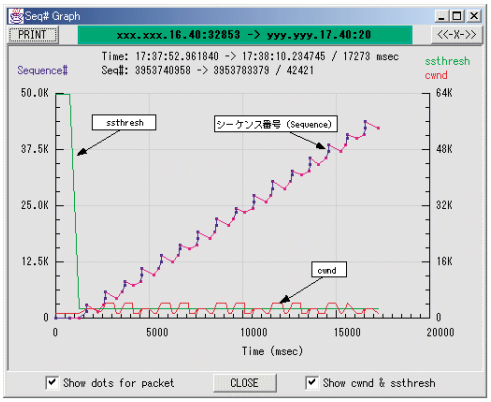


図 10 パラメータ値の時間的変化
Fig. 10 Time variation of parameter values.

(1) 図 9 より，通信システム A において，コネクション確立直後に送出されたデータパケットは，0.5 秒後にタイムアウト再送されていることが分かる．すなわち，通信システム A では再送タイムアウト時間の初期値が 0.5 秒程度に設定されていると考えられる．

(2) このとき図 10 から分かるように，cwnd は 1 パケット（=1,460 バイト）となる．次に，最初の DT に対する ACK が遅れて届いたため，同様に cwnd が 2 パケット（=2,920 バイト）となる．したがって，図 9 に示されているように，通信システム A は 2 つの DT パケットを送信し，ACK 待ちの状態になる．ここで再び，DT が再送されている．

(3) 以降は，その動作が繰り返され，図 10 に示すように，cwnd が 1〜2 パケットの間でしか変動せず，1.5 メガビット/秒の物理回線であるにもかかわらず，20 キロビット/秒程度のスループットしか得られなかったと考えられる．

そもそも，通信システム A における再送タイムアウト時間の初期値が 0.5 秒程度であることに問題があると考えられる[17]．本アナライザを用いることにより，スループット低下の原因が，通信システム A における再送タイムアウト時間によるものであることが判明した．本アナライザは通信システム内で発生した現象しか示さないが，プロトコルの振舞いによる性能低下などの原因の解析においては，それが有益であると考えられる．

5. 考　　察

5.1 関連ツールとの比較

市販のプロトコルアナライザは多数のプロトコルのフォーマット解析を行うが，対応するパケット間の関

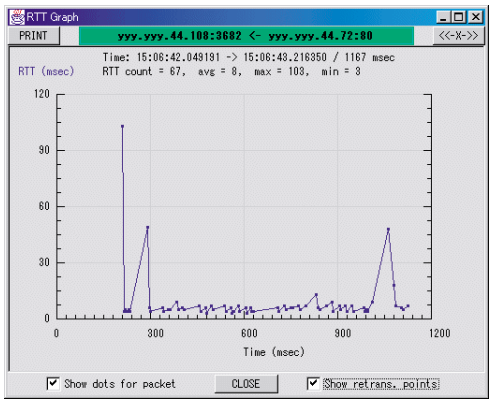


図 11 RTT の時間的変化
Fig. 11 Time variation of RTT.

連づけやプロトコル手順の推定など，大部分の解析を，専門的な知識を持ったユーザが手作業で行わなければならない．これに対し，本アナライザは，TCP に特化し，実際に通信システムが行った TCP の手順を自動的に解析する機能を持つ．

tcptrace は，tcpdump などの標準的なトラヒックトレースをもとに，シーケンス番号や応答確認番号など，パケットに含まれるパラメータ値を比較することにより，DT とそれを確認する ACK の対応づけなどを解析することができる．しかしながら，輻輳制御手順などの TCP の内部動作の推定は行っていないため，図 10 における cwnd の時間的変化を表示することはできない．このため，本ツールでは，スループットの低下が TCP の輻輳制御手順と関連していることを推定することはできない．

TCPAnaly は，1 章で述べたように，TCP の実装を評価することを目的としている．このため，ACK の受信と，ACK 受信による送信ウインドウの上限の増加により発生する DT 送信，あるいは，DT の受信と，DT 受信により発生する ACK 送信との対応づけを厳密に解析することを必要としている．したがって，因果関係を持つ DT と ACK の因果関係の種別ごとの統計情報を出力する機能を有する．しかしながら，本アナライザが提供するように，個々のパケットを解析した後の状態や内部変数を出力する機能は有していない．たとえば，本アナライザでは，4 章で示したように，図 8〜図 10 などの図を用いて，通信システムが実際に行った動作をグラフィカルに表示することができるが，TCPAnaly ではこれを行っていない．本アナライザは，**図 11** に示すように，RTT の変動などを表示することも可能とする．

```
遷移*6: send in ESTABLISHED
1) condition: status==FR
output: DATA with SEQ>snd_nxt in DATA;
next state: ESTABLISHED;
variable update:
  if (snd_nxt-snd_una>cwnd') TCP_spec &= Reno;
  /* Estimating TCP_spec is not Tahoe, but Reno */

遷移*8: ACK in ESTABLISHED
1) condition: dupacks==2 and ACKNUM in ACK==snd_una
and WND in ACK==snd_wnd
output: DATA with SEQ==snd_una in DATA;
next state: ESTABLISHED;
variable update:
  TCP_spec &=(Tahoe | Reno);
  /* Estimating TCP_spec is Tahoe or Reno */
  ssthresh=max(2*mss, 1/2*min(cwnd, snd_wnd));
  cwnd=ssthresh+3*mss; cwnd'=mss; dupacks=0; status=FR;
  /* Estimating fast retransmit algorithm */
output: none
next state: ESTABLISHED;
variable update:
  TCP_spec &= Other;
  /* Estimating TCP_spec is Other */
2) condition: dupacks<2 and ACKNUM in ACK==snd_una
and WND in ACK==snd_wnd
next state: ESTABLISHED;
variable update:
  if (status!=FR) dupacks=dupacks+1;
  else if (TCP_spec || Reno) cwnd=cwnd+mss;
3) condition: snd_una<ACKNUM in ACK <= snd_nxt
next state: ESTABLISHED;
variable update:
  snd_una=ACKNUM in ACK; snd_wnd=WND in ACK; dupacks=0;
  if (status==FR)
    if (TCP_spec || Reno) cwnd=ssthresh; status=CA;
    /* Meaning the condition is true if Reno flag is set on TCP_spec */
    /* This is the start of congestion avoidance following fast recovery */
    else cwnd=cwnd'; status=SS;
  if (status==SS)
    if (cwnd<=ssthresh) cwnd=cwnd+mss;
    else cwnd=cwnd+mss*mss/cwnd; status=CA;
  if (status==CA)
    cwnd=cwnd+mss*mss/cwnd;
    if (cwnd>65535) status=NONE; cwnd=65535;
  /* These two are slow start and congestion avoidance */
```

(注) statusにおけるSS,CA,FRは，それぞれSlow Start,Congestion Avoidance, Fast Recoveryのアルゴリズムを実行中であることを示し，NONEはいずれのアルゴリズムも実行されていないことを示す．

図 12 輻輳制御手順に関する TCP の状態遷移
Fig. 12 TCP state transition on congestion control.

5.2 異なる仕様への対応

TCP は，OS ごとに固有に実装されているため，すべての OS の TCP の動作を正しく推定するには，その OS の実装に依存した状態遷移を行う必要がある．本アナライザでは，Tahoe および Reno の仕様と，Fast Retransmit も行わない仕様（Other）を対象として内部動作の推定を行っている．**図 12** に示すように，Tahoe または Reno とそれ以外の仕様を判別するため，3 つの duplicate ACK の受信を検出した場合には，これに対応する Fast Retransmit による DT が存在するかどうかの検査を行っている．DT が存在する場合には，Tahoe または Reno の仕様，DT が存在しない場合には，Other の仕様に従っているものと判断し，以降の内部動作の推定を行う．また，Tahoe と Reno の仕様を判別するため，Fast Recovery 中に，それぞれの仕様に対応した cwnd の値を保持している．Tahoe に対応した cwnd を超える DT の送信が検出された場合に，Reno に従っているものと判断する．

すでに，輻輳制御手順を改良した新しい仕様[18),19)]や，SACK（Selective Acknowledgment）[20)]などの新しい機能を含む仕様が固まっており，これらの仕様を選択的に実装した OS も出現してきている．これらの OS の動作を解析するには，状態遷移を新たに追加する必要がある．

本アナライザは，図 4 に示した状態遷移を C 言語によりコーディングすることにより実装している．したがって，新しい TCP の仕様に対応するには，状態遷移表の修正を行い，修正部分のソースコードを書き直す必要がある．したがって，修正に必要なソフトウェアの規模は，新しい仕様を TCP に組み込む場合の規模に比例するものと考えられる．

6. おわりに

本稿では，回線上を流れるパケットをもとに，互いに通信する通信システムの TCP の内部動作を推定するインテリジェント TCP アナライザの設計および実装と，本アナライザによる TCP の動作解析の結果を示した．本アナライザは，通信システム間で実際に送受信されたパケットをもとに，標準的な TCP の仕様に従って，通信システムの TCP の内部動作を推定する．また，解析結果をもとに，DT と ACK の関連づけを含めたシーケンス図や，パラメータ値の時間的変化などを GUI を用いて表示する機能を持つ．さらに，遅延が大きい WAN を介した環境において，本アナライザの解析例を示した．この解析結果より，TCP の再送タイムアウト時間の初期値が小さいことが，スループット低下の原因となっていることが判明し，本アナライザの通信システムの内部動作を推定する機能の有効性を示すことができた．

謝辞 最後に，日頃ご指導いただく KDDI 研究所・浅見所長に感謝する．

参考文献

1) Stevens, W.R.: *TCP/IP Illustrated, Volume 1: The Protocols*, Addison Wesley (1994).
2) Moldeklev, L. and Gunningberg, P.: How a Large ATM MTU Causes Deadlock in TCP Data Transfer, *IEEE Trans. Networking*, Vol.3, No.4 (1995).
3) Tekelec: *Chameleon User's Manual* (1992).
4) Parker, S. and Schmechel, C.: Some Testing Tools for TCP Implementors, RFC2398 (1998).
5) Ostermann, S.: tcptrace Homepage. http://www.tcptrace.org/
6) Paxson, V.: Automated Packet Trace Analysis of TCP Implementations, *Proc. SIGCOMM'97* (1997).
7) 大岸智彦，井戸上彰，加藤聰彦，鈴木健二：TCP の振舞いをエミュレートするインターネット対応リンクモニタ，情報処理学会 Dicomo ワークショップ (1997)
8) Kato, T., Ogishi, T., Idoue, A. and Suzuki, K.: Design of Protocol Monitor Emulating Behaviors of TCP/IP Protocols, *Proc. IWTCS'97*, pp.416–431 (1997).
9) Kato, T., Ogishi, T., Idoue, A. and Suzuki, K.: Intelligent Protocol Analyzer with TCP Behavior Emulation for Interoperability Testing of TCP/IP Protocols, *Proc. FORTE/PSTV'97*, pp.449–464 (1997).
10) Fall, K. and Floyd, S.: Simulation-based Comparisons of Tahoe, Reno, and SACK TCP, *ACM Computer Communication Review*, Vol.26, No.3, pp.5–21 (1996).
11) Ogishi, T., Idoue, A., Kato, T. and Suzuki, K.: Intelligent Protocol Analyzer for WWW Server Accesses with Exception Handling Function, *Proc. IWTCS'98*, pp.49–64 (1998).
12) Sun Microsystems, Inc.: STREAMS Programming Guide, *Solaris 2.6 Software Developer Collection Vol.1* (1997).
13) Microsoft, Inc.: Windows 95 DDK (includes documentation updates), *Microsoft Developer's Network Development Platform* (1998).
14) DARPA: Transmission Control Protocol, RFC793 (1981).
15) Allman, M., Paxson, V. and Stevens, W.: TCP Congestion Control, RFC2581 (1999).
16) 大岸智彦，井戸上彰，加藤聰彦，鈴木健二：インターネット対応リンクモニタを用いた TCP プログラムの動作分析，第 57 回情報処理学会全国大会，Vol.3, pp.370–371 (1998).
17) Paxson. V., Allman, M., Dawson, S., Fenner, W., Griner, J., Heavens, I., Lahey, K., Semke, J. and Volz, B.: Known TCP Implementation Problems, RFC2525 (1999).
18) Floyd, S. and Henderson, T.: The NewReno Modification to TCP's Fast Recovery Algorithm, RFC2582 (1999).
19) Allman, M. and Floyd, S.: Increasing TCP's Initial Window, RFC2414 (1998).
20) Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A.: TCP Selective Acknowledgment Options, RFC2018 (1996).

(平成 13 年 5 月 7 日受付)
(平成 13 年 10 月 16 日採録)

大岸　智彦（正会員）

1992年東京大学工学部電気工学科卒業．同年国際電信電話（株）（現KDDI（株））入社．現在,（株）KDDI研究所ネットワーク管理グループ研究主査．この間，通信システムの試験，通信プロトコルのモニタリング，ネットワークの品質管理に関する研究に従事．1998年情報処理学会大会奨励賞受賞，電子情報通信学会会員．

井戸上　彰（正会員）

1984年神戸大学工学部電子工学科卒業．1986年同大学院修士課程修了．同年国際電信電話（株）（現KDDI（株））入社．現在,（株）KDDI研究所モバイルIPネットワークグループ主任研究員．この間，OSIやインターネット等の通信プロトコルに関連して，ハードウェア／ソフトウェアによるプロトコル実装方式，通信ボード用OS，プロトコル試験，モバイルコンピューティング等の研究に従事．1992年情報処理学会大会奨励賞，1998年情報処理学会大会優秀賞受賞．電子情報通信学会会員．

加藤　聰彦（正会員）

1978年東京大学工学部電気工学科卒業．1980年同大学院修士課程修了．1983年同大学院博士課程修了．同年国際電信電話（株）（現KDDI（株））入社．1987年，1988年，Carnegie Melon大学客員研究員．現在,（株）KDDI研究所執行役員．形式記述，分散システム，ATM，高速インターネット，モバイルインターネットの研究に従事．1993年より，電気通信大学大学院情報システム科客員助教授．1990年元岡賞受賞．