6U－6

プロトコルエミュレーション機能を用いた TCP/IPトラヒックの動作解析

大岸　智彦　　井戸上　彰　　加藤　聰彦　　鈴木　健二
国際電信電話株式会社　研究所

1．はじめに

インターネットの普及に伴い、TCP/IP に従う通信が広く行われている。この内、高信頼性なデータ転送を保証するTCPは、受信確認、フロー制御、タイムアウト再送などの基本機能[1]に加えて、輻輳制御などのアルゴリズム[2]が新たに定められている、新たなアルゴリズムは必ずしも実装されているとは限らない、バッファサイズなどのパラメータ値により動作が異なる場合があるなどの理由により複雑な振舞いを示す。このために、フォーマット解析機能のみを有する市販のプロトコルアナライザだけでは、TCP/IP 通信の十分な解析を行うことができない。そこで筆者らは、TCP/IP プロトコルの状態や内部変数を推定する機能(プロトコルエミュレーション機能)を有するインターネット対応リンクモニタ[3][4]を開発した。本稿では、輻輳制御を中心としたTCP/IP トラヒックの、本モニタによる動作解析の結果について述べる。

2．インターネット対応リンクモニタの概要

インターネット対応リンクモニタは、TCP プロトコルの動作を追随するプロトコルエミュレーション機能を持つ。本モニタは、以下の手順でプロトコルエミュレーションを行う。

(1) 図1に示すように、モニタは、一対の通信システムを対象として、双方の通信システムの TCP のエミュレーションを行う。対象とする通信システム間のフレームを取得し、一方の通信システムに対しては送信イベント、他方の通信システムに対しては受信イベントとして扱う。

(2) エミュレーション部では、TCP コネクション毎に状態や内部変数を管理し、TCP のプロトコル仕様をもとに、モニタの振舞いとして独自に定義したエミュレーション仕様[4]に従って、イベントに対して状態や内部変数を更新する。

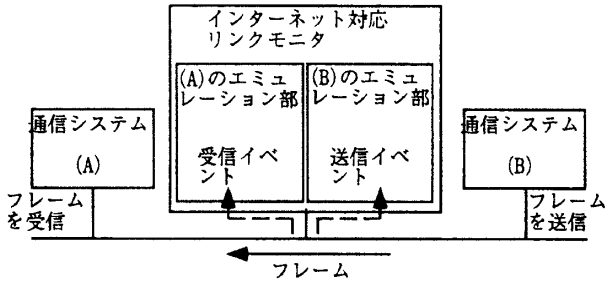

図1　インターネット対応リンクモニタの構成

(3) エミュレーション仕様はTCPの基本機能とslow start and congestion avoidance や fast retransmit and fast recovery の輻輳制御もサポートする。

(4) イベントは発生時刻順に処理し、その都度 TCP の状態や内部変数を推定する。送信イベントを処理する場合は、そのイベントの送信が可能かを判断し、可能な場合は状態及び内部変数を更新する。受信イベントを処理する場合は、応答と考えられる送信イベントを選択し、受信／送信の組で遷移可能かを判断し、可能なら状態及び内部変数を更新する。

(5) エミュレーションの結果は、図2に示すような

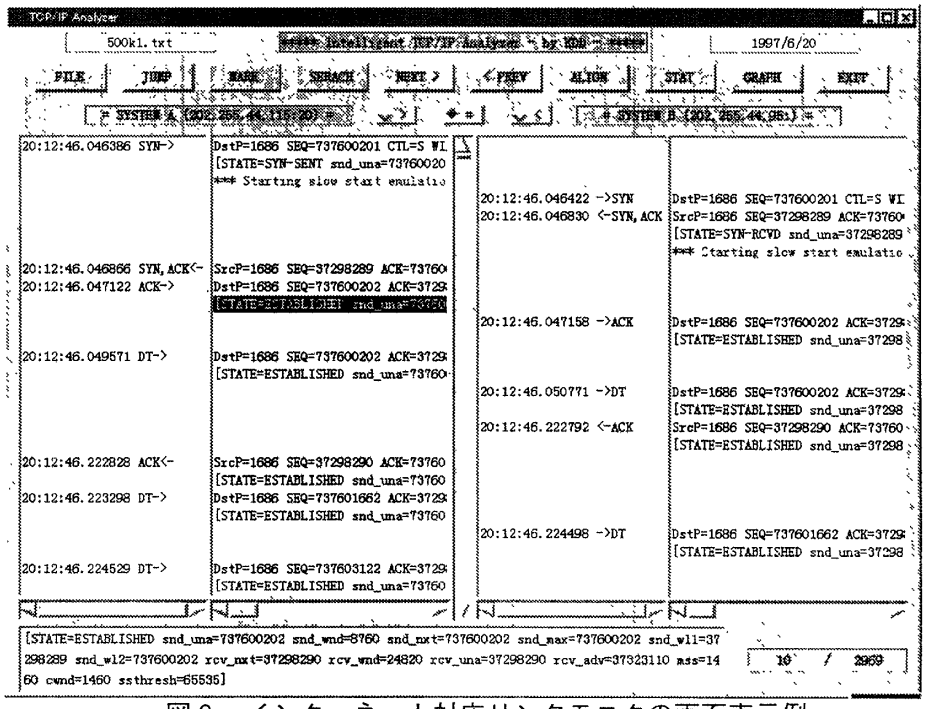

図2　インターネット対応リンクモニタの画面表示例

Analysis of TCP/IP Traffic Using Protocol Emulation Function
Tomohiko OGISHI, Akira IDOUE, Toshihiko KATO and Kenji SUZUKI
KDD R&D Laboratories

GUI を用いて表示する。双方の通信システムのエミュレーション結果は左右に分けて表示する。また、各イベントに対し、イベント発生時刻、SYN 送信などのイベント種別、パラメータ、状態／内部変数、slow start の開始や再送の発生など通信状況を示すコメントを表示する。

3．TCP トラヒックの解析

通信システムとして、SUN ワークステーション(Solaris2.5.1)を用いて輻輳制御の動作解析を行った例を、以下に示す。

(1) slow start and congestion avoidance の解析

TCP では、コネクション確立直後またはタイムアウト再送が発生した場合に、cwnd (congestion window)と呼ばれる送信側が調節するウインドウを１セグメント(最大セグメントサイズ：MSS)に設定する。以後、ACK を受信する毎に cwnd を１セグメントずつ増加させる。このアルゴリズムを slow start と呼ぶ。また cwnd が ssthresh (slow start threshold)で示された閾値を越えると、ACK を受信する毎に cwnd を線形的に増加させる。これを congestion avoidance と呼ぶ。

本実験では、コネクション確立前からモニタを開始することにより slow start の動作を観測した。図 3(a)に送信側の通信システムのエミュレーションの結果を示す。コネクション確立時(SYN,ACK 送信時)に slow start が発生するものと推定し、cwnd 及び ssthresh の値を、slow start 発生時の値に設定した(①)。このとき、cwnd の値より、送信可能なデータが１セグメント(1460 バイト)に制限されており、推定通りに１つだけ DT の送信が行われている(②)。ACK を受信したとき、cwnd が２セグメントに増加するものと推定する(③)。このとき、２セグメントのデータが送信可能となり、２つの DT の送信が行われている。さらに ACK を受信したとき、cwnd が３セグメントに増加するものと推定する(④)。

(2) fast retransmit and fast recovery の解析

TCP では、３つの Duplicate ACK を受信したとき、セグメントがネットワーク上で紛失したものと判断し、ssthresh を cwnd/2 に、cwnd を ssthresh + (受信した duplicate ACK の数)*MSS に設定し、要求されたセグメントを再送する。このアルゴリズムを fast retransmit and fast recovery と呼ぶ。

本実験では、ネットワーク上 10^{-6} 程度のビットエラーを発生させることによりセグメントを紛失させ、fast retransmit を発生させた。図 3(b)に送信側の通信システムのエミュレーションの結果を示す。⑤の DT がネットワーク上で紛失したため、DT を送信する度、Duplicate ACK を受信している。３つ目の Duplicate ACK を受信したとき、後に fast retransmit が行われるものと判断し、その旨を示すコメントを表示している(⑥)。⑦は、既に送信済みのシーケンス番号の持つ DT であり、データの再送であると判断している。⑥で fast retransmit が期待されていること、往復遅延時間の２倍に比べて極めて短い約 16ms 後に再送を行っていることから、fast retransmit が行われたものと判断し、ssthresh を cwnd/2 に、cwnd を ssthresh + 3*MSS に設定している。再送された DT を確認する ACK を受信したとき、congestion avoidance を開始するものと判断し、cwnd を ssthresh + MSS に設定している(⑧)。

```
①14:14:37.873958 SYN<-          SEQ=2793552896 WIN=24820 MSS=1460
 14:14:37.874505 SYN,ACK->       SEQ=3780367690 ACK=2793552897 WIN=8760 MSS=1460
                                 [STATE=SYN-RCVD cwnd=1460 ssthresh=65535]
                                   ** Starting slow start emulation **
  14:14:37.874924 <-ACK          SEQ=2795552897 ACK=3780367691 WIN=24820
                                 [STATE=ESTABLISHED cwnd=1460 ssthresh=65535]
②14:14:37.883326 DT->            SEQ=3780367691 ACK=2793552897 WIN=8760 LEN=1460
                                 [STATE=ESTABLISHED cwnd=1460 ssthresh=65535]
③14:14:37.884966 <-ACK           SEQ=2793552897 ACK=3780369151 WIN=24820
                                 [STATE=ESTABLISHED cwnd=2920 ssthresh=65535]
  14:14:37.885296 DT->           SEQ=3780369151 ACK=2793552897 WIN=8760 LEN=1460
                                 [STATE=ESTABLISHED cwnd=2920 ssthresh=65535]
  14:14:37.886531 DT->           SEQ=3780370611 ACK=2793552897 WIN=8760 LEN=1460
                                 [STATE=ESTABLISHED cwnd=2920 ssthresh=65535]
④14:14:37.888151 ACK<-           SEQ=2793552897 ACK=3780372071 WIN=24820
                                 [STATE=ESTABLISHED cwnd=4380 ssthresh=65535]
```

(a) slow start and congestion avoidance

```
⑤22:00:29.977229 DT->            SEQ=2827957322 ACK=527366987 WIN=24820 LEN=1460
                                 [cwnd=10220 ssthresh=65535]
        :
  22:00:29.990541 DT->           SEQ=2827963162 ACK=527366987 WIN=24820 LEN=1460
                                 [cwnd=10220 ssthresh=65535]
⑥22:00:29.991922 <-ACK           SEQ=527366987 ACK=2827957322  WIN=27740
                                 [cwnd=10220 ssthresh=65535]
                                 ** Duplicate ACK received. **
                                 ** Fast retransmit expected. (third dupacks) **
  22:00:29.991977 DT->           SEQ=2827965514 ACK=527366987 WIN=24820 LEN=1460
                                 [cwnd=10220 ssthresh=65535]
  22:00:29.993264 <-ACK          SEQ=527366987 ACK=2827957322 WIN=27740
                                 [cwnd=10220 ssthresh=65535]
                                 ** Duplicate ACK received. **
⑦22:00:29.993314 DT->            SEQ=2827957322 ACK=527366987 WIN=24820 LEN=1460
                                 [cwnd=10950 ssthresh=5110]
                                 ** Fast retransmit and fast recovery. **
  22:00:29.994578 <-ACK          SEQ=527366987 ACK=2827957322 WIN=27740
                                 [cwnd=12410 ssthresh=5110]
                                 ** Duplicate ACK received. **
⑧22:00:29.994578 <-ACK           SEQ=527366987 ACK=2827961702 WIN=27740
                                 [cwnd=6570 ssthresh=5110]
                                 ** Starting congestion avoidance emulation. **
```

(b) fast retransmit and fast recovery

図３　TCPトラヒックの解析例

4．おわりに

本稿では、TCP のプロトコルエミュレーション機能を有するインターネット対応リンクモニタを用いて、TCP/IP トラヒックの動作解析を行った結果について述べ、本モニタにより、輻輳制御などの TCP の複雑な振舞いを、容易に解析することができることを示した。最後に日頃ご指導頂く KDD 村上取締役に感謝する。

参考文献

[1] DARPA Internet Program Protocol Specification, "Transmission Control Protocol," RFC793, Sep. 1981
[2] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms," RFC2001, Jan. 1997
[3] T. Kato et. al, "Design of Protocol Monitor Emulating Behaviors of TCP/IP Protocols," to appear in Proc. of 10th International Workshop on Testing of Communicating Systems, Sep. 1997
[4] 大岸他、"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の設計," 、情処 DiCoMo ワークショップ、pp.143-148、Jul. 1997