4F－5

インテリジェントTCPアナライザを用いたTCPプログラムの動作分析

大岸　智彦　　井戸上　彰　　加藤　聰彦　　鈴木　健二
KDD研究所

1．はじめに

近年、インターネットの普及により TCP/IP に従う通信が広く行われている。TCP/IP プロトコルスタックのうち、高信頼なデータ転送機能を持つ TCP[1] においては、輻輳制御[2]等に関する仕様が近年も変更されており、また高速化を図るためベンダー固有の実装を行っている場合も多い。このため TCP プログラムは、オペレーティングシステム(OS)の種別やバージョン毎に、通信手順やパラメータ値が微妙に異なっている。このため場合によっては、TCP 通信のスループットが低下するなどの問題が生ずる。

これまでに筆者らは、RFC で規定されている標準的な TCP の仕様をもとに、通信システム内の TCP の状態や内部変数を推定するインテリジェント TCP アナライザ[3]を開発している。そこで筆者らは、本アナライザを用いて、Solaris や Windows 等の各種 OS において実装されている TCP プログラムの動作を分析した。本稿では、その結果について述べる。

2．TCPの輻輳制御と応答確認の手順

以下に、標準的な TCP の輻輳制御と応答確認の手順について述べる。TCP では、相手の通信システムから通知される awnd (advertised window) と、送信側の内部で調節する cwnd (congestion window) のうち、小さい方の値分、応答確認されていないデータを転送することができる。図1 (1) に示すように、コネクション確立時に、cwnd は 1*mss (maximum segment size) に設定され、ACK を受信する毎に 1*mss ずつ増加させる。この手順を slow start という。cwnd が ssthresh (slow start threshold) と呼ばれる閾値を超えると、cwnd を (3) の式により線形的に増加させ、congestion avoidance を行う。データが相手の通信システムに届かなかった場合には、タイムアウト再送か、3つ以上の duplicate ACK (応答確認やウインドウ更新を行わない ACK) を受信した時点で要求されたデータを再送する fast retransmit のどちらかにより再送が行われる。タイムアウト再送時間の初期値は、システム毎に異なるが、コネクション毎に RTT の2倍に近づくように再計算される。タイムアウト再送では、cwnd, ssthresh は (4) に従い、slow start を行う。一方、fast retransmit では、cwnd, ssthresh の更新は (5) に従う。さらに、再送後新たなデータを確認する ACK を受信すると、cwnd を ssthresh の値に設定し、congestion avoidance を行う。これを fast recovery と呼ぶ。

```
(1) コネクション確立時
  cwnd : 1*mss、ssthresh : 通常、cwndの最大値
(2) slow start
  応答確認のACK受信毎に、cwndをmssずつ増加
(3) congestion avoidance
  応答確認のACK受信毎に、(a) cwndをmss*mss/cwnd
(4) タイムアウト再送時
  cwnd=1*mss, ssthresh=max(2*mss, min(cwnd,awnd)/2)
(5) fast retransmit
  再送発生時：ssthresh=max(2*mss, min(cwnd,awnd)/2)
              cwnd=ssthresh + dupacks*mss
    (注) dupacks : 受信したduplicate ACKの数
  duplicate ACK受信毎に、cwndをmssずつ増加
(6) fast recovery
  fast retransmit中に、応答確認のACK受信時に、
  cwnd=ssthreshとし、congestion avoidanceを行う
```

図1　TCPの輻輳制御手順

TCP では、二つ以上のデータを受信するかタイマが満了するまで応答確認を行う ACK を返さない。このとき、タイマの満了を待たずに送出された ACK を immediate ACK、タイマ満了後に送出された ACK を delayed ACK、タイマを delayed ACK タイマと呼ぶ。

3．TCPプログラムの実装の検査

(1) 試験方法

図2に各々の TCP プログラムの実装を解析するためのネットワーク構成を示す。インテリジェント TCP アナライザは、通信システム (A)(B) 間の TCP の通信をモニタし、(A) が送受信した TCP セグメントに着目して、TCP の動作を解析する。意図的に再送を発生させるため、(A)(B) 間には ATM スイッチ及び回線シミュレータを接続し、伝送エラーを発生させる。Solaris2.5.1、Solaris2.6、Windows 95、Windows 98、Windows NT4.0 が動作する通信システ

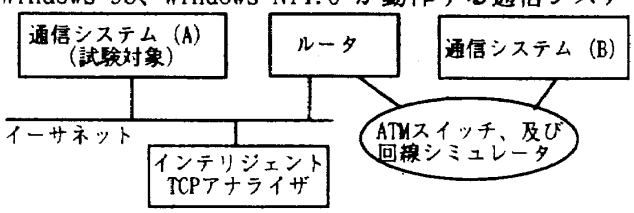

図2　ネットワーク構成

Behavior Analysis of TCP Programs using Intelligent TCP Analyzer
Tomohiko Ogishi, Akira Idoue, Toshihiko Kato and Kenji Suzuki　KDD R&D Laboratories

ムを試験対象とした。

(2) 結果と考察

本アナライザを用いて、TCP の輻輳制御機能を解析した結果の例（(A) の解析結果のみ抜粋）を図3に示す。①の DT がネットワーク上で紛失したため、②において①が未受信であることを示す ACK を受信している。これは、以降に送信した各 DT に対して行われるため、複数の duplicate ACK が受信される(③)。3つ目の duplicate ACK を受信したとき、fast retransmit による再送が発生する可能性があることを表示している(④)。再送を検出したとき、④で3つの duplicate ACK を受信していること、①からの時間差がタイムアウト再送に比べて短いことなどにより、fast retransmit による再送であると判断する(⑤)。このとき、図1(5) により、cwnd と ssthresh を計算する。さらに、新しい ACK 番号を持つ ACK を受信したとき、図1(6) に従い、congestion avoidance に移行すると判断する(⑥)。

本アナライザを用いて、TCP プログラムの実装を比較した結果を表1に示す。以下に考察を述べる。

・Windows 98/NT4.0 では、cwnd の初期値が 2*mss と推定された。これは、スループット向上のため、コネクション確立時に、相手側に immediate ACK を行わせるためと考えられる。

・start と congestion avoidance の実装の有無は、図1に従って cwnd を推定しながら、ウインドウ長を超えるデータを送信しているか否かを検査することにより行った。調査対象の全てのシステムがこれらの手順を実装しているという結果が得られた。ただし、Solaris 2.5.1/2.6 では、congestion avoidance のウインドウの増加幅を cwnd*cwnd/mss +mss/8 としていることが判明した。

・fast retransmit の実装の有無は、3つ以上の duplicate ACK を受信したとき、通信システムの応答時間内に再送が行われるか否かで判断した。Windows 95 は、fast retransmit を実装していないという結果が得られた。

・Solaris 2.5.1/2.6 では、fast retransmit 後に、ウインドウ長の推定値を超えるデータを送信した。これにより、これらは fast recovery を実装していないと判断した。

・Solaris 2.5.1（初期リリース、パッチなし）では、初期タイムアウト時間は 500ms と推定された。

・delayed ACK タイマのタイムアウト時間は、相手側のシステムから単一の DT を受信したときから ACK を返すまでの時間で測定した。対象とした通信システムはいずれも、複数回の測定において、DT 受信から delayed ACK を行うまでの時間がほぼ一定であった。従って、これらの TCP は DT 受信の際に delayed ACK タイマを起動していると考えられる。

```
11:09:29.555148 DT-> SEQ=(582201) LEN=1460
   [snd_una=(550893) awnd=37960 snd_nxt=(583661)
    cwnd=65535 ssthresh=65535]                    ——①
11:09:29.558859 DT->
      :
11:09:29.560816 ACK<- ACK=(582201) WIN=37960
   [snd_una=(582201) awnd=37960 snd_nxt=(588041)
    cwnd=65535 ssthresh=65535]                    ——②
11:09:29.560885 ACK<- ACK=(582201) WIN=37960
   [snd_una=(582201) awnd=37960 snd_nxt=(588041)
    cwnd=65535 ssthresh=65535]
    *** Duplicate ACK received. ***               ——③
      :
11:09:29.587643 ACK<- ACK=(582201) WIN=37960
   [snd_una=(582201) awnd=37960 snd_nxt=(588041)
    cwnd=65535 ssthresh=65535]
    *** Duplicate ACK received. ***               ——④
    *** Fast retransmit expected (third duplicate ACK). ***
      :
11:09:29.589152 DT-> SEQ=(582201) LEN=1460
   [snd_una=(582201) awnd=37960 snd_nxt=(619349)
    cwnd=55480 ssthresh=18980]
    *** Retransmission occurred by fast retransmit *** ——⑤
11:09:29.590853 ACK<- ACK=(619349) WIN=37960
   [snd_una=(619349) awnd=37960 snd_nxt=(619349)
    cwnd=19092 ssthresh=18980]
    *** Starting congestion avoidance emulation. ** ——⑥
```

SEQ:シーケンス番号、ACK:ACK番号、WIN:ウインドウ長、LEN:データ長
snd_una:未確認の最も古いデータのSEQ、snd_nxt:次に送るべきデータのSEQ

図3　インテリジェントTCPアナライザによる解析例

表1　TCP プログラムの実装の比較

	Solaris 2.5.1*	Solaris 2.6	Windows 95	Windows 98	Windows NT4.0
初期 cwnd	1*mss	1*mss	1*mss	2*mss	2*mss
slow start の実装	○	○	○	○	○
congestion avoidance の実装	△	△	○	○	○
fast retransmit の実装	○	○	×	○	○
fast recovery の実装	×	×	×	○	○
初期タイムアウト時間	500ms	3s	3s	3s	3s
delayed ACK タイマのタイムアウト時間	50ms	100ms	130ms	130ms	130ms

(注1) Solaris 2.5.1 は、初期リリース、パッチなしのシステムで試験を行った。以降のリリースやパッチありのシステムは、Solaris 2.6 と同様の結果を示した。
(注2) congestion avoidance の実装で、△は、ウインドウ増加分を cwnd*cwnd/mss+mss/8 にしているシステムを示す。

また、本試験において、Solaris では以下のような実装が行われていることが判明した。

・Solaris 2.5.1/2.6 では、単一のデータ受信に対しても、immediate ACK を行う場合がある。

・Solaris 2.6 では、SYN+ACK では小さいウインドウ長を通知し、コネクションが確立されてから、ウインドウ長を拡大する場合がある。

・Solaris 2.6 では、数 100ms の遅延がある環境で 512K バイトのソケットバッファサイズを使用したとき、fast retransmit による再送が頻繁に発生し、その場合はソケットバッファサイズが 64K バイト以下のときと比べ、スループットが約 1/5 に低下する。

4．まとめ

本稿では、インテリジェント TCP アナライザを用いて TCP の動作解析を行い、代表的な OS の種別やバージョン毎に、TCP プログラムの動作を分析した。

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]大岸、井戸上、加藤、鈴木、「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、Sep. 1997.