

開始日	2007年05月26日
最終更新日	2009年06月03日

はじめに

RFC3782について自分でまとめた情報を記しています。このRFCは[WindowsVistaのTCPで採用](#)されています。

オリジナル

「The NewReno Modification to TCP's Fast Recovery Algorithm」

<http://www.ietf.org/rfc/rfc3782.txt>

公開日：2004年4月

注意

- もともと個人利用を目的としてまとめているために、けっこう意識している部分があります。「意味分らないよ」とか「おかしいんじゃない?」とかいうのがあれば、オリジナルを参照するか、コメントで質問してください(がんばって調べます)。

用語

訳文に出てくる各語に対応する原文と意味を以下に記します。

略語	正式名称	意味
SMSS	SENDER MAXIMUM SEGMENT SIZE	送信側が送ることのできる最大セグメントサイズ
cwnd	CONGESTION WINDOW	輻輳ウィンドウ。送信側が相手のACK無しで一度に送ることができるサイズの一要因
ssthresh	Slow Start THRESHold	スロースタートから輻輳回避アルゴリズムに移るときのしきい値
FlightSize	FLIGHT SIZE	ネットワーク中に流れているデータのバイト数
recover	--	NewReno実装における肝となる変数。初期値は初回に送信したセグメント数(単位：バイト)で、再送処理時に更新される

更新履歴

- 2007/5/26 作成開始

概要

このドキュメントは、TCPのRFC2582に記された高速再転送 & 高速リカバリアルゴリズムに記されている「実装依存(Variant)」という用語について、それを明確にする目的で書かれた。RenoとNewRenoの違いを明らかにするのも目的に含まれる。
なお、このドキュメントはSACKが無効であるときの通信を前提としている。

このドキュメントでRFC2582を改訂する (= RFC2582はObsolete扱い)。したがって、「NewReno」といえば本RFCを指す。

まとめ

このRFCの構成

本RFCの、各章の概要は以下の通り。

1-2章	概要と用語の説明
3章	NewRenoにおけるFast RetransmitとFast Recoveryの基本動作 (RFC2581の振り返り)
4-6章	幾つかのNewRenoの実装の紹介と、それら実装の背景、ACKベースのヒューリスティックとTimestampベースのヒューリスティックの紹介
7-8章	送信側、受信側における推奨動作
9章	NS2におけるシミュレーションファイルの紹介
10章	RenoとNewRenoの比較、NewRenoのメリット・デメリットの紹介
11章	RFC2582からの変更点
12-15章	まとめ ~ 参考文献

現状

NewRenoが公開されてから、RFC2582でいうところの実装依存部が数多くの派生実装を生み出す結果になっている。

高速再転送 & 高速リカバリアルゴリズムの問題点

SACKがないときに、1 ウィンドウサイズ内の複数パケットがロストした場合に同アルゴリズムが有効に動作しない点。

1 個のパケットがロストした時は、高速再転送 & 高速リカバリアルゴリズムによって素早く元の通常通信に戻る。しかし、複数のパケットがロストした場合、その先頭パケットのみACKが届き、高速再転送アルゴリズムが始まる前には送信済みで、しかし途中でロストしたデータについてはACKされない。

これを「Partial acknowledgement (中途半端なACK) 」と呼んでいる。

SACKが動作すれば必要なパケットのみの再送によって速やかに復旧するため問題はないが、SACKはノードの両端がSACKに対応していないといけないため、SACKに頼るだけでは非効率な通信になってしまう可能性がある。

ちなみにタイムスタンプオプションもPartial acknowledgementには有効。なぜなら、高速再転送のときに飛んでいた重複ACKが、高速再転送の前か後かが分かるため。

NewRenoを一言で言うと？

Renoよりも改善されたFast Recoveryアルゴリズムを持っているTCPシステム。

NewRenoの動作

NewRenoの動作シーケンスを以下に記す。[RFC2581](#)でFast RetransmitとFast Recoveryについて触れているが、NewRenoにおいてのそれらは少々異なる動作をする。

- 3つの重複ACK：3つ目の重複ACKを受信し、なおかつ送信側がFast Recoveryモードになっていない場合、累積のACKフィールドが変数"recover"の値よりも大きいかどうかを確認する。大きければ1.1へ、そうでなければ1.2へ移行する。
 - Fast Retransmitモードの開始：ssthreshを「FlightSize/2」または「2*SMSS」のどちらか大きい方に設定し、さらにこれまでに送信された最大シーケンス番号を変数"recover"にセットする。2へ移る。
 - Fast Retransmitモードの回避：Fast Retransmit/Fast Recoveryモードには遷移せず、ssthreshも更新しないで、本シーケンスを抜ける（重複ACKが届いても何もしない）。
- Fast Retransmitによる再送：ロスしたTCPセグメントを再送し、cwndに「ssthresh+3*SMSS」をセットする。3SMSSを足すのは重複ACKを考慮してすぐに復帰するため。
- Fast Recoveryへの遷移：本期間中に重複ACKを受信したら、その都度cwndをSMSSずつ増加させる。これはネットワーク中に残っているセグメントを考慮してすぐにcwndを拡大させるため。
- Fast Recoveryの続き：cwndまたはレシーバの受信バッファが許す限り、セグメントを送信する。
- 新しいデータに対するACKが届いた場合、そのACKは「2で再送されたデータへのACK」または「それより後の再送データへのACK」のどちらかと考えられる。後述。
- 再送のタイムアウト：再送に対するタイムアウトが発生した場合、recoverには送信した最大のシーケンス番号を記録しておき、可能ならばFast Recovery状態を抜ける

5番のときの処理について、もう少し掘り下げよう。

「2で再送されたデータへのACK」

つまり、送信側から転送したデータがすべて確認応答された、ということの意味している。"recover"で記録した値も含めて、そこでcwndを

- $\text{MIN}(\text{ssthresh}, \text{FlightSize} + \text{SMSS})$
- ssthresh

のどちらかに設定し、ウィンドウサイズを収縮(deflate)させる。この時点で、これ以上ウィンドウサイズを拡張するのは危険だと判断するため。

「それ(2で再送されたデータ)より後の再送データへのACK」

RFC2582との違い

幾つかある。

複数回の高速再転送アルゴリズムが連続して発生してしまうこと

高速再転送のロジックにはACK-basedヒューリスティックと、TIME-basedヒューリスティックが存在する

参考サイト・RFC

- TCP Congestion Control : <http://www.ietf.org/rfc/rfc2581.txt>
 - 1999年4月
- NewReno Modification to TCP's Fast Recovery : <http://www.ietf.org/rfc/rfc2582.txt>
 - 1999年4月