■3 群（コンピュータネットワーク）- 4 編（トランスポートサービス）

1 章 TCP（Transmission Control Protocol）の基礎

（執筆者：石田 賢治）[2013 年 12 月 受領]

■概要■

TCP の動作概要，及び，基本的な機能に関して概説する．

【本章の構成】

本章では， TCP の概要，TCP のプロトコル状態遷移，TCP データ転送手順（コネクションの確立／クローズ），TCP 再送制御，TCP 輻輳制御，TCP 輻輳制御の改良，TCP セグメントのフォーマットとポート番号について述べる．

■3 群 - 4 編 - 1 章

1-1 TCP とは

（執筆者：石田賢治）[2013 年 12 月 受領]

TCP（Transmission Control Protocol）は，トランスポート層を代表する通信プロトコルである．通常，コンピュータネットワークにより提供されるネットワークサービスは，コネクション型サービスとコネクションレス型サービスに分類される．TCP はコネクション型サービスを提供するプロトコルであり，その主な目的は，送信側と受信側に対応するエンドツーエンド間に対して信頼性の高いデータ転送サービスを提供することである．信頼性の高いデータ転送サービスを提供するために，TCP は，シーケンス番号，タイマ，エンドツーエンド間のチェックサム，確認応答，再送制御，輻輳制御などの仕組みをもつ．

TCP のもともとの仕様は，1981 年に発行された技術文書である RFC 793 （Request for Comments）[1] において記述されている．しかし， 記述の一部に不完全な部分もあり，その後に出された RFC [2] などで補足されている．また，この RFC 793 に至る背景や TCP を含む通信プロトコルの設計思想[3)〜5)]がまとめられている．TCP は，数多くのアイデアが導入された複雑な通信プロトコルであり，現在においてもその改良が続けられている．TCP で処理される情報の単位をセグメント（Segment）と呼ぶ．TCP に関しては，幾つかのテキスト[6)〜13)]が出版されている．

以降，1-1-2 項では TCP のプロトコル状態遷移について述べる．コネクション型サービスを提供する TCP においては，コネクションの管理が厳密に行われている．次に，1-1-3 項で TCP データ転送手順（コネクションの確立／クローズ）に関して説明する．1-1-4 項では，信頼性を提供するうえで必要不可欠な TCP 再送制御，及び，再送制御の効率化を目指した技術について述べる．1-1-5 項では，TCP のフロー制御と輻輳制御について説明する．1-1-6 項では，TCP のより改良された輻輳制御について述べる．最後に，1-1-7 項では，TCP セグメントのフォーマットとポート番号について述べる．

■3 群 - 4 編 - 1 章

1-2 TCP のプロトコル状態遷移

（執筆者：石田賢治）[2013 年 12 月 受領]

TCP の状態は，**図 1･1** のような有限個の状態からなる状態遷移図 [1)2)11)] により記述される．この有向グラフは，TCP の状態遷移の概要であり，TCP の動作全体の厳密な仕様ではない．例えば，エラーの条件やその際の対応は省略されている．図中の四角の囲みは TCP の各状態を表す．また，一つの状態から有向辺に付けられているラベルに基づき遷移する先は一つである．なお，ラベルは「イベント／アクション」を表す．アクションをともなわずに状態変移する場合には，図を見やすくするため「／アクション」の部分を省略している．

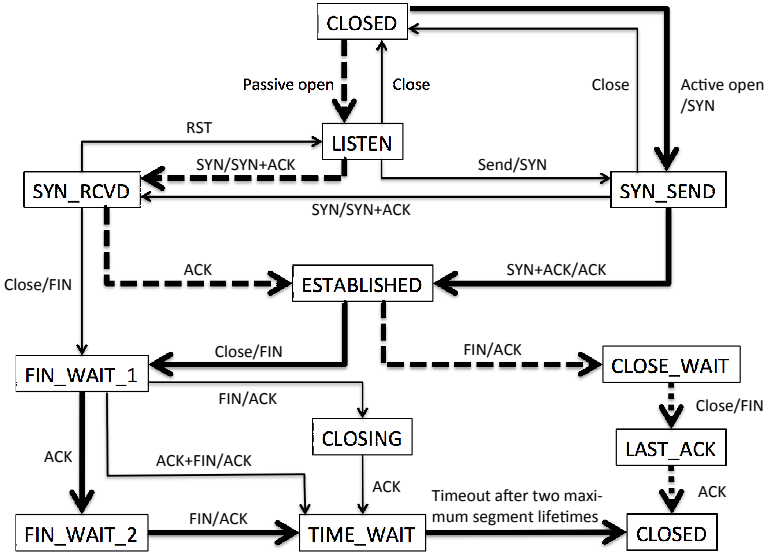

図 1･1　TCP **接続の状態遷移図**

イベントは大きく 2 種類に分けられる．一つは，アプリケーション層由来のユーザコールであり，OPEN，SEND，CLOSE などが相当する．他の一つは，タイムアウトや受信セグメントに含まれる制御情報である SYN，ACK，RST，FIN である．アクション部分は，基本的に記述されている制御情報の通信相手への送信を意味する．例えば，状態「SYN_SEND」から別の状態「ESTABLISHED」への有向枝のラベル「SYN+ACK/ACK」は，状態「SYN_SEND」が SYN と ACK を受信し，SYN に対応する ACK を送信した後に，状態「ESTABLISHED」に遷移することを意味する．

以下，TCP の各状態を簡単に説明する．

- LISTEN：通信相手からの接続待ちの状態．

- SYN_SEND：コネクション確立要求の送付後であり，対応するコネクション要求待ちの状態.
- SYN_RCVD：通信相手からのコネクション確立要求の受信，及び，こちらからのコネクション確立要求送付後であり，送付したコネクション確立要求に対する ACK 待ちの状態.
- ESTABLISHED：通信相手とコネクションが確立された状態.
- FIN_WAIT_1：通信相手からのコネクション終了要求あるいは既に送付したコネクション終了要求に対する ACK 待ちの状態.
- FIN_WAIT_2：通信相手からのコネクション終了要求待ちの状態.
- CLOSE_WAIT：ローカルユーザからのコネクション終了要求待ちの状態.
- CLOSING：通信相手からのコネクション終了要求に対する ACK 待ちの状態.
- LAST_ACK：通信相手へ送付したコネクション終了要求に対する ACK 待ちの状態.
- TIME_WAIT：コネクション終了要求に対する ACK 待ちの状態．時間的余裕をもって ACK を確認するため最大セグメント寿命の 2 倍の時間待ってから CLOSED に遷移.
- CLOSED：コネクションがない状態.

図中のアクティブオープンは，クライアントによる能動的なコネクション確立要求に対応する．一方，パッシブオープンは，サーバのように受動的なコネクション確立に対応する．アクティブ，パッシブの説明を明確にするため，ここでは通信相手をクライアントとサーバに区別する．

クライアント側の典型的な TCP のコネクション確立から終了（クローズ）へ至る状態遷移系列は，図 1･1 の太線となる．また，対応するサーバ側の典型的な TCP のコネクション確立から終了への状態遷移系列は，図 1･1 の破線となる．

■3 群 - 4 編 - 1 章

1-3 TCP データ転送手順（コネクションの確立／クローズ）

（執筆者：石田賢治）[2013 年 12 月 受領]

まず，コネクションの確立について述べる．TCP のコネクション確立手続きは，通常，クライアント側からのアクティブオープンにより始まる．クライアント側における TCP の状態遷移は，CLOSED の状態から，SYN_SEND を経て，ESTABLISHED に至る（**図 1･2** 参照）．また，対応するサーバ側の動作は，パッシブオープンと呼ばれる．その状態遷移は，CLOSED から，LISTEN，SYN_RCVD を経て ESTABLISHED となる．

TCP の三つのセグメントのやり取りにより，コネクションを安全に，かつ，確実に確立可能としている．その手続きは以下の通りであり，3 方向ハンドシェイク（Three-way Handshake）と呼ばれている．図 1･2 に，通常のコネクション確立におけるセグメントの送受信の様子を示す．

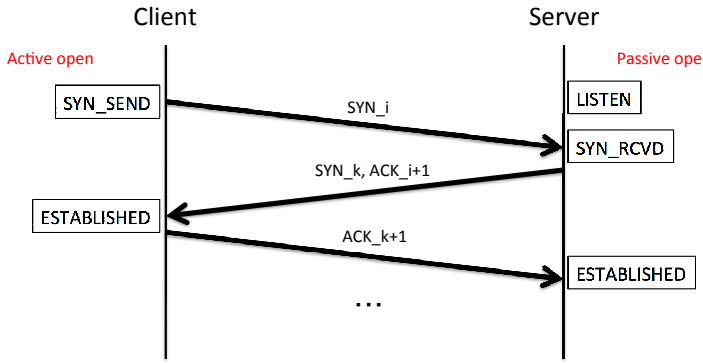

図 1･2　TCP コネクションの標準的な確立の流れ

1. クライアントは，接続したいサーバのポート番号とクライアントの初期シーケンス番号 i を含む SYN セグメントをサーバに送る．

2. サーバは，クライアントからの SYN セグメントに対する確認応答としてシーケンス番号 $i+1$ をもつ ACK と，クライアント側とは異なるサーバ側の初期シーケンス番号 k を含む SYN セグメントをクライアントに送る．

3. クライアントは，サーバ側からのシーケンス番号 k を含む SYN セグメントに対応する確認応答として，シーケンス番号 $k+1$ をもつ ACK を返信する．

3 方向ハンドシェイクは次の特徴をもつ同期方式である．クライアントとサーバが同一のシーケンス番号で手続きを開始することを要求しないので，両端がグローバルに同期する必要がない．また，最大セグメント生存時間により，ネットワーク内をさまよって遅れて到着するような，コネクション接続要求セグメントを排除 [14] できる．

次に，コネクションの終了について述べる．標準的なコネクションの確立では，合計三つのセグメントが両端でやり取りされる．一方，コネクション終了時には四つのセグメントが用いられる（**図 1･3** 参照）．これは，全 2 重である TCP のコネクションを二つの一方向コネクションとして扱い，それぞれを解放する手続きをとるためである．そのため，片方向だけのコネクションを閉じた状態であるハーフクローズの状態をとり得る．

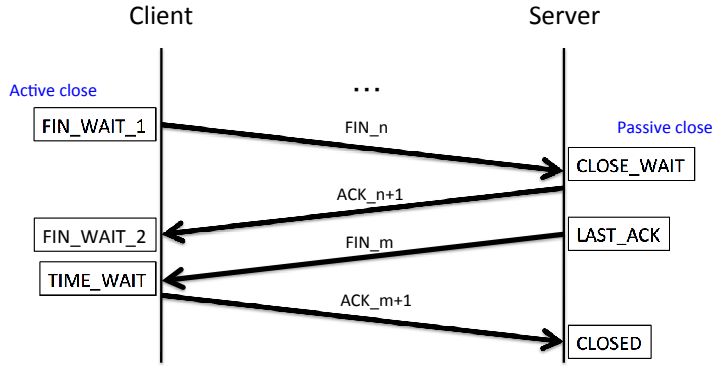

図 1･3　TCP コネクションの標準的な終了の流れ

TCP のコネクションの通常終了の手続きは，最初に FIN を送った端末から開始される．その端末は，アクティブクローズの動作を行ったことになる．FIN を受け取った対応する一方の端末は，パッシブクローズの手続きに入る．例えば，クライアント側からアクティブクローズを行った場合には，その TCP の状態遷移は，ESTABLISHED の状態から，FIN_WAIT_1，FIN_WAIT_2，TIME_WAIT を経て，CLOSED に至る（図 1･1 参照）．FIN_WAIT_2 で送付される ACK に対する確認応答はないため，TIME_WAIT にて最大セグメント生存時間の 2 倍の時間待ってから CLOSED に遷移する．また，対応するサーバ側の動作は，パッシブクローズとなる．その状態遷移は，ESTABLISHED から，CLOSE_WAIT，LAST_ACK を経て CLOSED となる．

ハーフクローズの状態をとり得るため，例えば，図 1･3 において，サーバ側は FIN_n を受け取り，その確認応答の ACK_n+1 を送信した後でも，サーバ側からクライアント側へデータ送信が可能である．その後，サーバ側における FIN_m の送信と ACK_m+1 の受信により双方向のコネクションは終了する．なお，コネクション終了手続きの途中にセグメントが失われた場合の通信プロトコル動作に関しての詳細な考察 [11] も行われている．

■3 群 - 4 編 - 1 章

1-4 TCP 再送制御

（執筆者：石田賢治）[2013 年 12 月 受領]

1-4-1 TCP 再送制御の基礎

TCP は，信頼性の高い通信サービスを提供するため，廃棄されたセグメントを再送する再送制御機能をもつ．まず，基本的なタイムアウトに基づく再送制御について述べる．次に，効率的な通信を目指した技術である早期再送（Fast Retransmit），SACK（Selective Acknowledgment）について簡単に説明する．

再送制御においては，シーケンス番号が用いられる．このシーケンス番号を送信端末及び受信端末間で参照し合うことでデータ転送の正確性を保証する．TCP ヘッダ内に格納されるシーケンス番号の初期値は，コネクション確立時に，送信端末及び受信端末のそれぞれで設定される．また，その単位はオクテット*である．また，コネクション確立時には，ウィンドウの初期値，最大セグメントサイズ（MSS：Maximum Segment Size）もやり取りされる．最大セグメントサイズ MSS は，TCP 及び IP のヘッダやオプションを除いたセグメント内のデータ部分の最大値を表す．

送信端末において，例えばシーケンス番号の初期値として 1000 オクテットが用いられる場合には，最初に送信されるセグメントのシーケンス番号は，1000 となる．つまり，送信端末におけるシーケンス番号は，送信するデータの先頭のオクテットを表す．送信するセグメントのデータ部分のサイズが，1460 オクテットであった場合には，2459 オクテットが当該セグメント内の最後のオクテットである．

受信端末は，受信確認として ACK（確認応答；Acknowledgment）を送信端末に返信する．ACK は，受信端末が次に受信したいデータの先頭オクテット数をシーケンス番号として含む．上記の例のように，シーケンス番号が 1000 であり，セグメントのデータ部分のサイズが，1460 オクテットであった場合には，対応する ACK に含まれるシーケンス番号の値は，2460 となる．このシーケンス番号を用いることにより，受信セグメントが順序通りに受信できていない場合でも，元の順序に並べ替える順序制御が可能となる．ACK は重要な制御情報であるが，それ自体はデータ送信そのものには貢献していない．そのため，ACK の送信効率を上げるために，逆方向のデータを転送するセグメントに ACK を相乗りさせるピギーバックという方式が利用されることがある．

送信端末から受信端末に，セグメントが連続して届かない場合もある．その対策として，TCP では 累積 ACK（Cumulative ACK）いう方式を導入している．累積 ACK とは，受信した最新のセグメントに対するシーケンス番号ではなく，連続して正しく受信された最後尾のセグメントに対するシーケンス番号を ACK 内に含めて，受信端末から送信端末に通知する方式である．

再送制御は，送信端末で行われる．TCP では，セグメントを送信した端末に対し，ある一定時間以内に対応する ACK が届かなかった場合には，そのセグメントが廃棄されたとみなして，送信端末が当該セグメントを再送する．この一定時間は，再送タイマにより管理される．再送タイマの値は，送信端末がセグメントを送信した時点から，それが受信端末に届き，受信

* 1 オクテットは 8 bit であり，1 byte が 8 bit の場合は 1 byte とみなしてよい．

端末からの ACK が送信端末に届くまでの時間であるラウンドトリップタイム RTT（Round Trip Time）に基づいて計算される．ネットワークの状況は時間とともに変化するので，再送タイマの値は RTT の平均値とその偏差によって次式のように計算される．

再送タイマ ← 平均 RTT + 4 × 偏差 RTT

ここで，

平均 $RTT_n \leftarrow (1-\alpha) \times$ 平均 $RTT_{(n-1)} + \alpha \times$ 測定 RTT_n

偏差 $RTT_n \leftarrow (1-\beta) \times$ 偏差 $RTT_{(n-1)} + \beta \times |$ 測定 $RTT_n -$ 平均 $RTT_n|$

であり，添字の n は時刻を表す．また，α, β の値としては，それぞれ，0.125，0.25 が推奨値とされている．このように再送タイマ値を設定することにより，RTT の変動が大きい場合には，再送タイマ値が長めに設定され無駄な再送が抑制される．また，上記の式より，RTT の値が安定している場合には，再送タイマ値は平均 RTT の値に近づくことが分かる．

1-4-2 早期再送（Fast Retransmit）

セグメントの廃棄は，基本的に再送タイマのタイムアウトで検出される．しかしながら，タイムアウトのみによる検出だと，セグメント廃棄が生じるごとにタイムアウト期間だけ送信が停止するため，TCP のスループットが大きく低下することがある．このような状況を回避するため，送信元が重複 ACK を受信することにより，そのセグメント廃棄を検出して当該セグメントを再送するという，早期再送（Fast Retransmit）[15] という技術が提案されている．早期再送において，送信端末は 3 個以上の重複 ACK が返ってきた場合，セグメントが破棄されたと判断し，タイムアウト期間を待たずに当該セグメントを再送する．早期再送は，早期回復（Fast Recovery）とともに利用される技術であり 1-6-1 項でより詳しく説明する．

重複 ACK は，例えば，以下のような場合に発生する．TCP では 累積 ACK を用いているため，受信端末が 10 番目のセグメントまでのすべてのセグメントを正しく受信している場合，受信端末は 11 番目のセグメントの送信を促す ACK を送信端末に返す．しかしながら，受信端末が期待している 11 番目のセグメントではなく 12 番目のセグメントが到着した場合，受信端末は 11 番目のセグメントの送信を促す ACK を送信端末に再度返す．早期再送の技術によって，より早期にセグメント廃棄が検出可能になり，スループットの低下が防止できる．

1-4-3 SACK（Selective Acknowledgment）

選択的確認応答（SACK：Selective Acknowledgment）[16][17] は，累積 ACK の課題を補う技術である．TCP では 累積 ACK を用いるため，１番目のセグメント 1 から 5 番目のセグメント 5 までの一連した通信において，廃棄されたセグメントが唯一セグメント 2 だけであった場合においても，受信端末はセグメント 2 の送信を促す ACK を送信端末に返す．つまり，廃棄されたセグメントの後続セグメントが正しく受信されているにもかかわらず，廃棄されたセグメント以降の全セグメントの再送を要求する．一方，SACK では，シーケンス番号が不連続であることにより受信端末がセグメント廃棄を検出した場合，TCP ヘッダのオプションフィールド *を用いて再送すべきセグメントを送信端末に対し明示的に通知可能である．このようにして，再送効率の向上を目指している．

* オプションフィールドのオプション種別に SACK を示す 4 を書き込む．

■3 群 - 4 編 - 1 章

1-5 TCP 輻輳制御

（執筆者：石田賢治）[2013 年 12 月 受領]

TCP における輻輳制御（Congestion Control）は，ネットワークの輻輳を検知，解消，回避する機能をもつ[18]．輻輳（Congestion）とは，ネットワークの混雑を意味する．輻輳が発生すると，セグメントが中継ノード（ルータ）で長時間待たされたり，中継ノードのバッファ溢れにより廃棄されることがある．輻輳制御とは，送信端末からネットワークやネットワーク内の中継ノードなどに対して，必要以上にデータを流し込まないようにするなどの，送信端末とネットワークが関係する制御である．その要素技術として，輻輳が生じたらネットワークへのパケット流入を制御する呼受付制御，混んでいるリンクを迂回するルーチング，及び，フロー制御（Flow Control）などがある．フロー制御とは，送信端末と受信端末間の制御である．フロー制御では，受信端末のバッファ溢れが起きないように，送信端末が送信レートを調整する．

TCP は，ウィンドウフロー制御によって，フロー制御と輻輳制御を行っている．ここでは，TCP の代表的なバージョンである TCP Reno に導入されている技術に関して主に述べる．まず，TCP のウィンドウフロー制御の基礎となるスライディングウィンドウによるウィンドウフロー制御を説明する．次に，TCP のフロー制御と輻輳制御の概要について説明する．

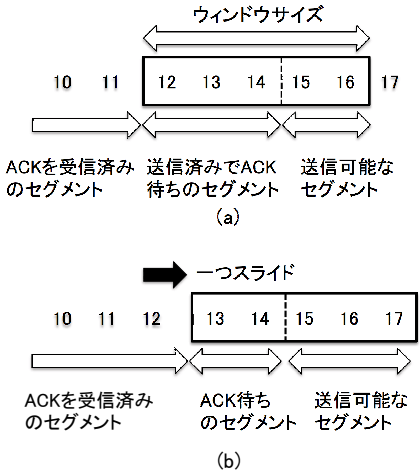

図 1･4　スライディングウィンドウの動作

1-5-1 ウィンドウフロー制御

ウィンドウフロー制御は，フロー制御の代表的な方式の一つである．通信の効率を上げるために，送信端末と受信端末間において，送信端末が ACK の到着を待たずに連続して送信可能なセグメント数を決定する．その最大数がウィンドウサイズである．

図 1･4 は，送信端末におけるウィンドウの動作の概要である．ここで，この図のウィンドウサイズは 5 セグメントである．また，図 1･4(a) において，セグメント 11 までは対応する

ACK を受信済みである．ウィンドウに入っている五つのセグメントのうち，セグメント 12 から 14 までは，既に送信済みで ACK 待ちである．セグメント 15，16 は，送信可能状態である．図 1･4(b) は，送信端末がセグメント 12 に対する ACK を受信してウィンドウが右に一つスライドした状態を表す．スライドした結果，セグメント 17 が新たに送信可能状態となる．ACK が届くごとにウィンドウがスライドするので，スライディングウィンドウフロー制御方式と呼ばれる．

1-5-2 TCP のフロー制御と輻輳制御の概要

TCP のフロー制御や輻輳制御においても，スライディングウィンドウフロー制御方式が用いられる．TCP におけるこのフロー制御を単にウィンドウフロー制御と呼ぶこともある．ただし，TCP ではウィンドウサイズが動的に変更される．

$$\text{送信端末ウィンドウ} \leftarrow \min(\text{輻輳ウィンドウ}, \text{フロー制御ウィンドウ}) \qquad (1\cdot1)$$

TCP のウィンドウフロー制御では，送信端末のウィンドウのサイズを式 (1･1) のように設定する．ここで，式 (1･1) 右辺の輻輳ウィンドウのサイズは，輻輳制御の観点から調整される．一方，フロー制御ウィンドウのサイズは，フロー制御の観点から決められる．つまり，送信端末のウィンドウのサイズは，フロー制御，及び，輻輳制御の双方の観点から決定されるため，TCP のウィンドウフロー制御は，フロー制御のみならず輻輳制御をも考慮したものとなっている．

まず，フロー制御について述べる．フロー制御ウィンドウのサイズは，受信端末から申告される受信端末における利用可能バッファ量に基づくものである．送信端末から送られるセグメント量は，このフロー制御ウィンドウのサイズ以下になるため，受信端末のバッファが溢れることはない．

受信端末のバッファが一杯になった場合には，フロー制御ウィンドウのサイズが 0 の情報をもつ ACK が返送される．この場合，送信端末は，新たなセグメントの送信を停止する．以降の通信の中断を避けるため，フロー制御ウィンドウのサイズ 0 の ACK を受信した送信端末は，一定時間間隔で，プローブセグメントと呼ばれるセグメントを受信端末に送り続ける．プローブセグメントを受け取った受信端末は，未だバッファが一杯の場合には，このプローブセグメントを単に破棄する．一方，受信端末のバッファに空きができた場合には，受信端末はその空き容量を ACK に含めて送信端末に返す．その結果，送信端末からのセグメント送信が再開される．

次に，輻輳制御について述べる．TCP の輻輳制御は，AIMD（Additive Increase Multiplicative Decrease）[18)19)] という考え方に基づいている．AIMD においては，輻輳が検知されていない状態では，セグメントの送信レートを比較的緩やかに線形的に増加させる．一旦，輻輳が検知されると送信レートを乗算的に急激に減少させる．

輻輳検知は，基本的に再送タイマのタイムアウトにより行われる．ネットワーク内にセグメントが溢れ輻輳状態になると，途中のルータでバッファ溢れが生じる．その結果，セグメント廃棄が起きる．再送タイマのタイムアウトは，このセグメント廃棄を間接的に観測していることに相当する．TCP は，輻輳発生を検知した時点で，式 (1･2) のように輻輳ウィンドウのサイズを半分にする．

輻輳ウィンドウ ← 輻輳ウィンドウ/2 (1･2)

また，再送タイマの時間以内に ACK が到着し続けているような輻輳が検出されない場合には，送信レートを以下の式 (1･3) のように増加させる．例えば，現在の輻輳ウィンドウのサイズと MSS がともに 1460 オクテットの場合，一つの ACK を受信すると，右辺の値は，1460 + 1460 × 1460/1460 で 2920 オクテットとなる．その結果，輻輳ウィンドウのサイズは MSS 2 個分となり，次の送信においてセグメントを連続して 2 個送信可能となる．引き続き，二つのセグメントが連続してほぼ同時に送出され，それに対する ACK もほとんどの場合，連続して返ってくる．この一連の時間は，ほぼ RTT に等しい．そのため，輻輳ウィンドウのサイズが MSS 2 個分の場合，RTT 経過後に輻輳ウィンドウのサイズは MSS 3 個分に増加することとなる．つまり，1 RTT ごとに 1 セグメント（MSS）分だけ輻輳ウィンドウが増加することになる（**図 1･5** 参照）．その結果，式 (1･3) のように送信レートも線形的に増加する．

輻輳ウィンドウ ← 現在の輻輳ウィンドウ + MSS×MSS/現在の輻輳ウィンドウ (1･3)

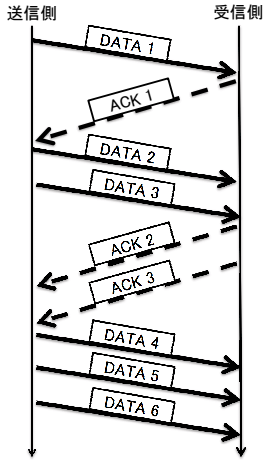

図 1･5　ウィンドウサイズの線形的増加によるセグメント送信

このような AIMD の概念のみに基づく制御を行うと，輻輳ウィンドウサイズの変動が大きいため，セグメントの送信が安定しにくい．そこで，TCP では，AIMD の概念に加えて，(1) スロースタート（Slow Start），(2) 輻輳回避（Congestion Avoidance），(3) スロースタート閾値 ssthresh（Slow Start Threshold）の概念を導入して，できるだけ安定してセグメント送信を可能とする工夫がされている．次に，この三つの概念を説明する．

(1) スロースタート（Slow Start）

TCP の通信開始時や輻輳検出後に適用される制御である．まず，輻輳ウィンドウのサイズは，1 MSS に設定される．また，コネクション確立時にはスロースタート閾値 ssthresh とし

て 64 K オクテットが設定される．この制御では，スロースタートアルゴリズム[18)20)] が利用される．

通信効率を上げるため，送信端末は ACK を受け取るごとに式 (1･4) のように輻輳ウィンドウのサイズが閾値 ssthresh に至るまで，あるいは，タイムアウトにより輻輳が検出されるまで，輻輳ウィンドウを急激に拡大する．このとき，1 RTT ごとに輻輳ウィンドウサイズは 2 倍となる（**図 1･6** 参照）．

$$\text{輻輳ウィンドウ} \leftarrow \text{現在の輻輳ウィンドウ} + \text{MSS} \tag{1･4}$$

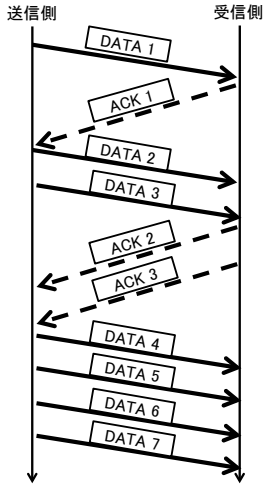

図 1･6　スロースタートによるセグメント送信

（2）輻輳回避（Congestion Avoidance）

スロースタートにおいては，急激に輻輳ウィンドウを拡大するため，セグメントの送信レートも急激に増加する．そして，輻輳ウィンドウのサイズがスロースタート閾値 ssthresh に到達した時点で，送信端末は輻輳発生の予防を目的とした輻輳回避の制御に入る．この制御では，輻輳回避アルゴリズム[18)20)] が利用される．具体的には，上記に示した式 (1･3) に従い輻輳ウィンドウのサイズをゆっくりと線形的に増加させる（図 1･5 参照）．

（3）スロースタート閾値 ssthresh

スロースタート閾値 ssthresh は，スロースタートフェーズと輻輳回避フェーズを区別するために用いられる．輻輳ウィンドウのサイズの拡大の結果，タイムアウトなしで閾値 ssthresh に達した場合には，送信端末は輻輳回避の制御に入る．一方，セグメント廃棄を意味するタイムアウトにより輻輳が検知された場合には，以下の式 (1･5) に基づいて，スロースタート閾値 ssthresh と輻輳ウィンドウのサイズが再設定[18)] される．この設定値は，AIMD において輻輳発生時に輻輳ウィンドウのサイズを半分に設定することに対応している．また，送信端末はスロースタートの制御に入る．

スロースタート閾値 ssthresh ← 現在の輻輳ウィンドウ/2

輻輳ウィンドウ ← MSS (1･5)

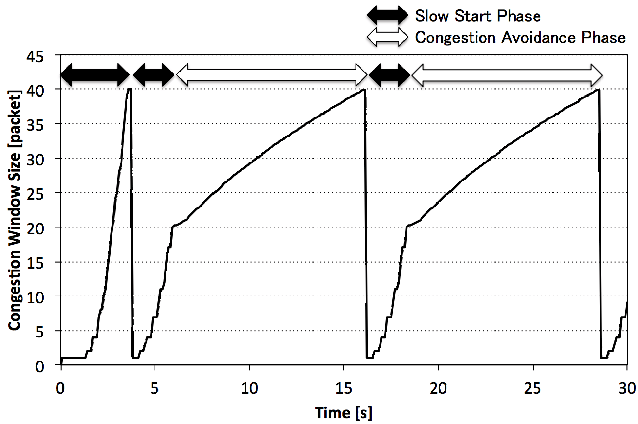

図 1･7　輻輳ウィンドウの変化例

図 1･7 に輻輳ウィンドウの変化例を示す．図 1･7 において，5 s 以降のスロースタート閾値 ssthresh は 20 packet となっている．この図は，TCP Reno より以前の TCP のバージョンである TCP Tahoe の輻輳ウィンドウの変化に対応している．TCP Reno は，ここで説明した基本的な機能に加えて，次節で述べる二つの機能が導入されており，より効率的なデータ転送を可能としている．上記の (1), (2), (3) は，以下のように整理される．

○輻輳が検出されない（ACK のタイムアウトなし）場合：
・輻輳ウィンドウ ＜ ssthresh の場合：スロースタートアルゴリズムによるセグメント伝送
・輻輳ウィンドウ ≥ ssthresh の場合：輻輳回避アルゴリズムによるセグメント伝送
○輻輳が検出された（ACK のタイムアウトあり）場合：
スロースタート閾値 ssthresh と輻輳ウィンドウのサイズの再設定の後，スロースタートアルゴリズムによるセグメント伝送

■3 群 - 4 編 - 1 章

1-6 TCP 輻輳制御の改良

（執筆者：石田賢治）[2013 年 12 月 受領]

TCP の輻輳制御の基礎は前節で述べた通りである．ここでは，データ転送の高速化を可能にしている代表的な二つの機能である早期再送（Fast Retransmit）と早期回復（Fast Recovery）[15] について述べる．これらの機能は，TCP Reno に実装されている．

1-6-1 早期再送（Fast Retransmit）

TCP では，セグメントの廃棄にともないタイムアウトが発生し失われた当該セグメント以降を送信端末が再送する．また，前節で説明したように，輻輳に対して TCP は，スロースタート値 ssthresh を現在の輻輳ウィンドウの半分に設定し，輻輳ウィンドウの大きさを MSS 1 個分にまで小さくしてしまう．その結果，一時的に極端に送信レートが低下する．

輻輳は，基本的にセグメントの廃棄にともなうタイムアウトにより検出されるが，それ以前に重複 ACK という形でも観測される．送信端末に同じセグメントを要求する ACK が来た場合，受信端末がそのセグメントの受信を必要としていることを示している．この重複 ACK という輻輳の兆候をとらえて，輻輳に対応する技術が早期再送である．

セグメントが失われることなく，到着順序が入れ替わって受信端末に到着した場合にも重複 ACK が発生する可能性がある．そこで，早期再送では，送信端末が同じ重複 ACK を 3 回受信した場合にセグメント廃棄と認識し，タイムアウトを待たずに直ちに当該セグメントを再送する．

1-6-2 早期回復（Fast Recovery）

早期回復は，通常，早期再送と組み合わせて利用される．この早期回復の手続きは以下の通りである．

(1) 三つの重複 ACK を受信した場合，この ACK に対応するセグメントを再送する（早期再送）．そして，スロースタート閾値 ssthresh と輻輳ウィンドウを次のように設定する．ここで，3 MSS を加えているのは，三つの重複 ACK 分に相当するセグメントが既に受信端末に到着済みでネットワーク内に存在せず，その分だけネットワーク内に空きがあると想定しているためである．

スロースタート閾値 ssthresh ← 現在の輻輳ウィンドウ/2
輻輳ウィンドウ ← ssthresh + 3MSS

(2) 更に重複 ACK と同じ重複 ACK を受信した場合，輻輳ウィンドウを

輻輳ウィンドウ ← 現在の輻輳ウィンドウ + MSS

と設定する．

(3) 重複 ACK ではなく，新しいセグメントを要求する ACK を受信した場合，

輻輳ウィンドウ ← ssthresh

と設定する．このスロースタート閾値 ssthresh の値は，(1) で設定した値である．以降のセグメント送信は，輻輳ウィンドウがスロースタート閾値 ssthresh より大きな状態

のため，輻輳回避の制御が適用される．

図 1･8 に，早期再送と早期回復の機能をもつ，TCP Reno の輻輳ウィンドウの変化例を示す．図 1･7 と比較して，輻輳ウィンドウサイズが大きな状態で維持されており，通信の効率が改善されていることが分かる．

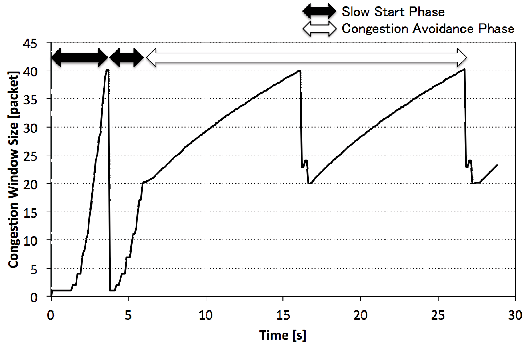

図 1･8　早期再送と早期回復の機能をもつ TCP Reno **の輻輳ウィンドウの変化例**

■3 群 - 4 編 - 1 章

1-7 TCP セグメントのフォーマットとポート番号

（執筆者：石田賢治）[2013 年 12 月 受領]

1-7-1 TCP セグメントのフォーマット

図 1･9 に，TCP セグメントのフォーマット[1)21)]を示す．32 bit で折り返して表示している．

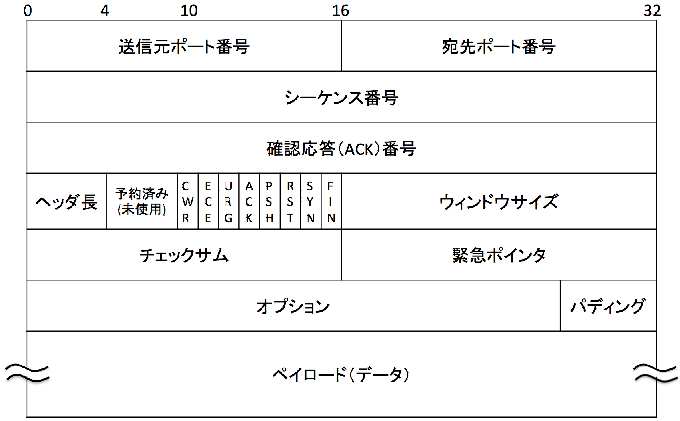

図 1･9　TCP セグメントのフォーマット

フォーマット内の各項目について述べる．

送信端末ポート：16 bit　送信端末のポート番号．

受信端末ポート：16 bit　受信端末のポート番号．

シーケンス番号：32 bit　送信データのオクテット単位の番号．

ACK 番号：32 bit　セグメントの受信を確認するシーケンス番号．次に受信端末が受信したいデータの先頭オクテット数をシーケンス番号として含む．累積 ACK の概念が導入されている．

ヘッダ長：4 bit　32 bit を単位とした TCP ヘッダ長を表す整数値．オプションを含む TCP ヘッダは，32 bit 長の倍数である．

予約済み：4 bit　予約されており未使用．

コントロールビット（CWR, ECE, ･･･, FIN）：8 bit

CWR：1 bit　IP プロトコルによる明示的な輻輳通知機能である ECN（Explicit Congestion Notification）とともに利用される．

ECE（ECN-Echo）：1 bit　ECN-Echo を表す．エンドツーエンドの通信において，

通信相手からのこちらへの経路が輻輳していたことを通信相手に伝える．ECN とともに利用される．

URG：1 bit　1 の場合，緊急ポインタフィールドが意味をもつ．0 の場合，緊急ポインタフィールドは意味をもたない．

ACK：1 bit　1 の場合，ACK 番号フィールドが意味をもつ．0 の場合，ACK 番号フィールドは意味をもたない．

PSH：1 bit　1 の場合，ペイロード内のデータを即座に上位層（アプリケーション層）に渡す．

RST：1 bit　1 の場合，現在の確立されているコネクションをリセットする．

SYN：1 bit　1 の場合，コネクション確立のための SYN であることを表す．

FIN：1 bit　1 の場合，コネクション終了のための FIN であることを表す．

ウインドウサイズ：16 bit　受信端末の空きバッファ容量を表す．

チェックサム：16 bit　TCP セグメント内に誤りがないかを確認するために使われる．このチェックサムの計算は，送信端末や受信端末の IP アドレスも含む形で行われる．そのため，送受信端末の IP アドレスも含む疑似ヘッダ [1)] *を当該セグメントの前に付けたものが計算対象となる．16 bit の倍数となるように 0 がデータの最後尾以降に連続して埋め込まれる．送信端末では，まずチェックサムフィールドが 0 に初期化される．次に，16 bit 単位で 1 の補数和をとり，得られた結果に対し 1 の補数をとったものが，チェックサムフィールドに入れられて送信される．受信端末では，当該の疑似ヘッダを受信セグメントの前に付けた形で 16 bit ごとに区切り，チェックサムフィールドを含むすべてのデータに対し，同様な（1 の補数和をとり，その 1 の補数をとる）計算が行われる．計算結果の 16 bit すべてが 1 であれば誤りなく受信できたとする．

緊急ポインタ：16 bit　緊急を要するデータの格納先を示すポインタ．

オプション　各種オプションが記述される 8 bit の倍数の長さをもつフィールド．SACK など TCP の制御を高度化するために利用される．

1-7-2 TCP のポート番号

IP アドレスは，受信端末の識別子として機能する．つまり，ルーチングによって送信端末からのパケットは，IP アドレスに基づいて受信端末まで届けられる．しかしながら，IP アドレスだけでは，受信端末のアプリケーション層で動作している複数プロセス内のどのプロセスに受信データを渡せばよいか分からない．そこで，適切なプロセスの識別子としてポート番号が利用される．

アプリケーション層で動作する，標準的な多くのプロセスに対して，well-known ポート番

* 送信 IP アドレス，受信 IP アドレス，TCP パケット長，プロトコル番号，パディング（疑似ヘッダ長を 16 bit の倍数にするための詰め物の 0）から成る．

号[22]が定義されている．現在，ポート番号は，Internet Assigned Numbers Authority（IANA）によって管理されている．例えば，web で用いられる HTTP には，80 番が割り当てられている．クライアントから web サーバにアクセスする場合には，このポート番号である 80 番を受信端末のポート番号として指定することにより，所期の web サービスを提供しているプロセスと通信可能となる．一方，送信端末であるクライアントのポート番号は，well-known ポート番号に縛られることなくクライアントで適切に決められる．

■参考文献

1) J. Postel (ed.), “Transmission Control Protocol,” RFC 793, 1981.
2) R. Braden (ed.), “Requirements for Internet hosts - communication layers,” RFC 1122, 1989.
3) V. Cerf and R. Kahn, “A protocol for packet network intercommunication,” IEEE Trans on Commun., vol.Com-22, no.5, pp.637-648, 1974.
4) B. Leiner, R. Cole, J. Postel, and D. Mills, “The DARPA internet protocol suite” IEEE Communications Magazine, vol.23, no.3, pp.29-34, 1985.
5) D. D. Clark, “The design philosophy of the DARPA Internet protocols,” Proc. SIGCOMM ’88, Computer Communication Review, vol.18, no.4, pp.106-114, 1988.
6) 宮原秀夫, 尾家祐二, “コンピュータネットワーク,” 共立出版, 1999.
7) 村田正幸, “マルチメディア情報ネットワーク –コンピュータネットワークの構成学–,” 共立出版, 1999.
8) 村山公保, 西田佳史, 尾家祐二, “トランスポートプロトコル,” 岩波講座　インターネット　第 3 巻, 岩波書店, 2001.
9) D. E. Comer (村井　純, 楠本博之訳), “TCP/IP によるネットワーク構築 Vol. I　原理・プロトコル・アーキテクチャ第 4 版,” 共立出版, 2002.
10) 小林　浩, 江崎　浩, “インターネット総論,” 共立出版, 2002.
11) A. S. Tanenbaum (水野忠則, 相田　仁, 東野輝夫, 太田　賢, 西垣正勝訳), “コンピュータネットワーク第 4 版,” 日経 BP 社, 2003.
12) 竹下隆史, 村上公保, 荒井　透, 苅田幸雄, “マスタリング TCP/IP 入門編 第 4 版,” オーム社, 2007.
13) 池田博昌, 山本　幹, “情報ネットワーク工学,” オーム社, 2009.
14) R. S. Tomlinson, “Selecting sequence numbers,” Proc. the 1975 ACM SIGCOMM/GIGOPS Workshop on Interprocess Communications, pp.11-23, 1975.
15) V. Jacobson, “Modified TCP congestion avoidance algorithm,” end2end-interest mailing list, April 30, 1990.
16) M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP selective acknowledgment options,” RFC 2018, 1996.
17) S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, “An extension to the selective acknowledgement (SACK) option for TCP,” RFC 2883, 2000.
18) V. Jacobson, “Congestion avoidance and control,” Proc. ACM SIGCOMM’88, pp.314-329, 1988.
19) R. Jain, K. Ramakrishnan, and D.-M., Chiu, “Congestion avoidance in computer networks with a connectionless network layer,” Tech. Rep. DEC-TR-506, Digital Equipment Corporation, 1987.
20) M. Allman, V. Paxson, and W. Richard Stevens, “TCP Congestion Control,” RFC 2581, 1999.
21) K. Ramakrishnan, S. Floyd, and D. Black, “The addition of explicit congestion notification (ECN) to IP,” RFC 3168, 2001.
22) J. Reynolds and J. Postel, “Assigned numbers,” RFC 1700, 1994.