基礎から学ぶWindowsネットワーク

第14回　信頼性のある通信を実現するTCPプロトコル（その1）

（3/3 ページ）

2003年12月25日 00時00分 公開

[デジタルアドバンテージ，著]

前のページへ 1 2 3

前ページでは、TCPにおけるウィンドウ制御の基本的な概念について解説した。そこでは何パケットかまとめてTCPデータを送信し、まとめてACKを返すことにより、効率よく（帯域幅ほぼいっぱいまで）回線を利用することができる例を示した。

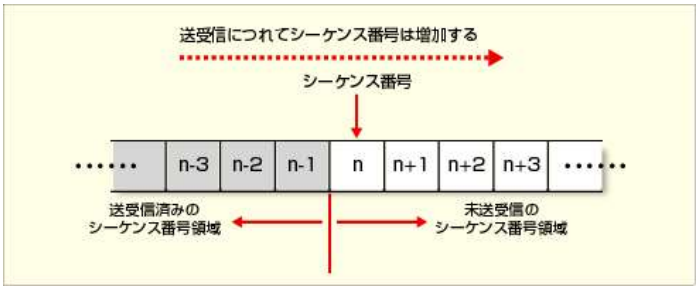
だが実際のTCPにおけるウィドウ制御では、ウィンドウ・サイズはパケット数ではなく（何パケットまとめて送信できるかではなく）、バイト数で数えることになっている。パケットのサイズは実際の物理的なネットワーク媒体によって異なるが、IP層によってそれらは仮想化されているので、パケットの数で扱うことはできないし、その必要もない。以下では、物理的なパケットの構造から離れて、バイト・データに基づいたウィンドウ制御の詳細について解説する。

シーケンス番号

TCPではストリーム型の信頼性のある通信を行うが、ストリームとは具体的には「バイト・データの連なり」を意味する（1byteは8bitであるとする）。2つのアプリケーション間でTCPのコネクション（接続）を確立すると、それらの間にはストリーム型の通信路が形成される。そしてある一方のアプリケーションからデータ（バイト・データの列）を通信路に書き込むと、書き込んだ順番の通りに、バイト・データ列を相手側で読み出すことができる。通信路は双方向になっており、どちらから書き込んでも、同じように相手側に届けられる。以下では片方向のみを取り上げて解説するが、実際にはまったく同じメカニズムが反対方向にも用意されている。

バイト単位のストリーム型通信を実現するため、TCPでは通信路内を流れるバイト・データに対して、それぞれのバイト位置を決める「シーケンス番号」を定義している。そしてシーケンス番号に基づいてデータを整列したり、ウィンドウ制御を行ったりする。

シーケンス番号とは、具体的には32bitの符号なしの整数値である。TCPのコネクションを開設すると、通信路にはそれぞれランダムなシーケンス番号が初期値として与えられる（双方向通信なので、2つ存在する）。通信路に対して割り当てられたシーケンス番号は、次のような図で表すことができる。ここで、1つの四角は1byteのデータを表し、それぞれにシーケンス番号と呼ばれる32bitの数値が割り当てられている。通信路内のバイト・データごとに1つずつシーケンス番号が割り当てられている。シーケンス番号はデータ1byteごとに1ずつ増えていき、32bitの最大値（0xFFFFFFFF）に達すると、次は0に戻り（ラップアラウンド（循環）する）、また1ずつ増えていく。32bitの数値なので、4Gbytesごとに同じシーケンス番号が割り当てられることになる。



シーケンス番号の定義

TCPでは、通信路内を流れるバイト・データに対して、「シーケンス番号」と呼ばれる整数値を割り当てている。初期値はランダム値で、データ1byteごとに1ずつ増加する。そしてシーケンス番号に基づいてデータの整列や、ウィンドウ制御を行う。シーケンス番号は32bitの整数値なので、4Gbytesごとにラップアラウンド（循環）する。現在のシーケンス番号がnであるとすると、n-1以下はすでに送受信が完了した領域、n以降はこれから送受信が行われる領域となる。

このシーケンス番号は、TCPが通信路内のデータに対して順序付けやウィンドウ制御を行うためだけに使用するものであり、上位のアプリケーションがこの値を使用することはないし、意識する必要もない。そのため、4Gbytesごとにラップアラウンドしていても上位アプリケーションには影響はない。

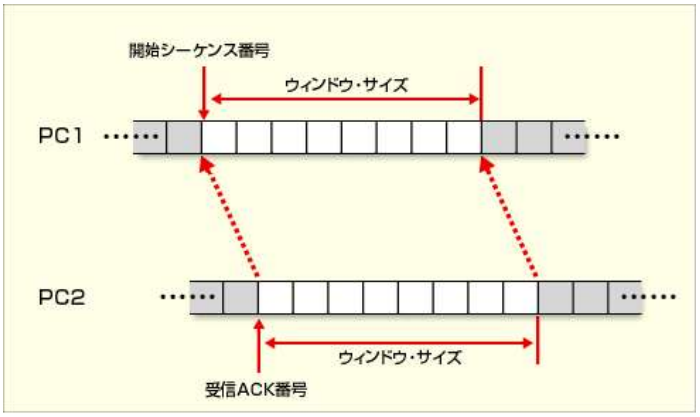
シーケンス番号は0や1から始まるわけではなく、コネクションごとにランダムに決められる。これは、（あるマシンが保持している多数のコネクションの間で）シーケンス番号がなるべく重複しないようにするためである。通信やアプリケーションが異常終了したりした場合、（アプリケーションが）新たにTCPコネクションを作り直すが、常に同じ番号（0や1など）から始まるとすると、以前と同じシーケンス番号を持つパケットが再生成されてしまう可能性が高くなる。すると、まだネットワーク上に存在している（かもしれない）直前のコネクションに対する送受信パケットが双方のコンピュータに届き、新たなコネクションの一部とみなされ、TCPのプロトコル・スタックの動作が不正になる可能性がある。このような異常事態を避けるため、以前のコネクションとはまったく異なるパケットになるように、ランダムな初期値を利用する（ことが推奨されている。実装によってはこうなっていない可能性もある）。

シーケンス番号とウィンドウ・サイズ

TCPプロトコルにおける送受信では、シーケンス番号とウィンドウ・サイズが重要な意味を持つ。「シーケンス番号」は、これから送受信しようとしているデータの位置、「ウィンドウ・サイズ」は送信可能なデータの最大サイズである。ウィンドウ・サイズが大きくなると、一度に送信できるデータの最大サイズが増え、逆に小さくなると送信できるサイズも少なくなる。この値は通信につれて動的に変更されるが、初期値はTCP/IPの実装やアプリケーションの指定などによって変更される。

一般的にはウィンドウ・サイズの初期値は数Kbytesから数十Kbytes程度である。この値は16bitの符号なし整数値で表現されるので、最大なら64Kbytesまで拡大することも可能であるが、現在ではTCP/IPの規格が拡張され、さらに大きなサイズにすることも可能となっている。ただしウィンドウ・サイズを大きくすると、その分プロトコル・スタックで使用するメモリ領域も増えることになるので、大きければよいというものでもない。またこの値は、次回説明するMSS（Maximum Segment Size）とも関係するので、どのような値でもよいというわけではない（詳細は次回解説予定）。

次の図をみていただきたい。これは、通信中のあるTCPセッションの状態を示したものである。上側は送信側のコンピュータ、下側は受信側のコンピュータをそれぞれ表している（実際にはTCPは双方向通信だが、ここでは上から下への送信に関する部分のみを取り出している）。



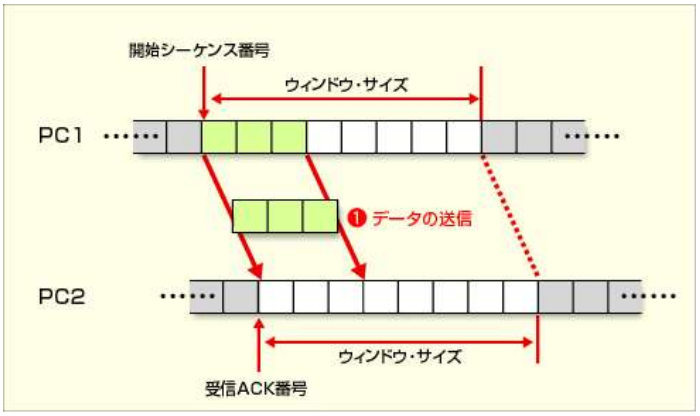
シーケンス番号とウィンドウ・サイズ
双方のシーケンス番号は同期し（同じ値を保持している）、同じウィンドウ・サイズ情報を共有している。シーケンス番号情報は送信側が、ウィンドウ・サイズ情報は受信側がそれぞれ相手に提供する。

「開始シーケンス番号」も「受信ACK番号」もどちらも同じ「シーケンス番号」のことであり、それぞれ送信側と受信側で呼び方を変えているだけである。シーケンス番号は、送信側（この例では上のPC1側）が生成し管理している。受信側のACK番号は、送信側から渡された値を元に決められている。

「ウィンドウ・サイズ」は、受信側（この例では下のPC2側）で一度に受信可能なデータのサイズを表している（この例では8bytes）。送信側におけるウィンドウ・サイズは、受信側から指示された値である。

送信手順―1．データ・パケットの送信

データを送信する場合は、まず送信側でウィンドウ・サイズに入るだけのデータ・パケットを用意し、TCPヘッダを付けてIP層に送信を依頼する。上位アプリケーションは現在のウィンドウ・サイズを知っているわけではないので、送信しようとするデータが必ずしもウィンドウ・サイズ以下になっているという保証はない。そのため、TCP層では、必要ならばウィンドウ・サイズに以下になるようにデータを切り分け、先頭から順次送信を行う（正確には、MSSサイズ以下に切り分けて送信する。詳細は次回解説）。もちろんウィンドウ・サイズ以下ならば、一度に送信することが可能である。以下の例では、ウィンドウ・サイズは8bytesだが、データのサイズは3bytesなので、1パケットで送信することができるだろう（1byteずつ3回に分けて送信してもよいが、そのような操作には意味がない）。



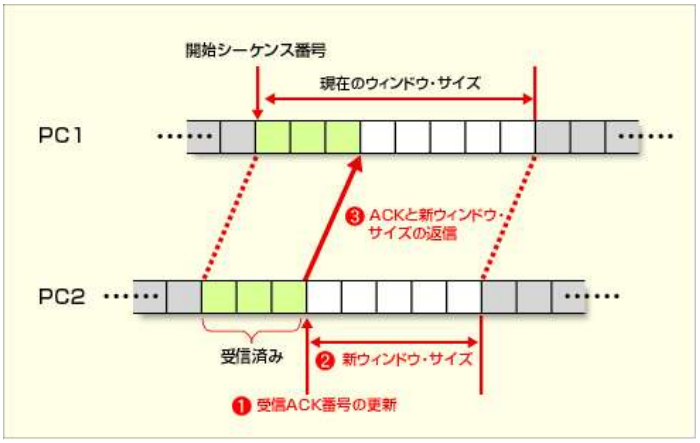
TCPパケットの送信処理
TCPパケットにユーザー・データ（この例では3bytes）と開始シーケンス番号を付け、IPパケットに載せて送信する。送信する場合は、最小1byte（0byteでも可）からウィンドウ・サイズいっぱいまでデータをまとめて送ることができる。

この状態では、TCPパケットはまだ相手に届く前であり、送信側も受信側もシーケンス番号やウィンドウ・サイズに変化はない。送信側では、再送に備えて、送信したパケ

ットのコピーをまだ保持している（送信したパケットのデータを破棄してはいけない）。

送信手順―2．データ・パケットの受信とACKの返信

パケットが受信側に届くと、受信側ではデータを取り出し、上位のアプリケーションへそれを渡すと同時に、ACKの返信を行う。ただし実際のプロトコル・スタックではすぐにACKを返さずに、ある程度受信データがまとまってから（パケットの受信が途切れてから）返信するのが普通である。そうすれば、無駄なACKを送信しなくても済むからだ。



受信処理とACKの返信
TCPパケットを受信した側では、データを受信バッファにコピーしてアプリケーションに通知すると同時に、受信したことを示すACKパケットを返信する。ACKパケットには、データを受信したことを示す更新済み受信ACK番号と、新しいウィンドウ・サイズがセットされている。
（1）受信したデータのサイズに応じて、受信ACK番号を更新する。この場合は3bytes受信しているので、受信ACK番号を3つ進める。
（2）受信したデータの分だけ受信バッファの空きサイズが減るので、その分ウィンドウ・サイズを減少させる。
（3）新しいACK番号とウィンドウ・サイズをACKパケットに載せて送信する。

TCPのデータ・パケットを受信すると、まず受信したデータを取り出し、それをTCP/IPのプロトコル・スタック内部の受信バッファへコピーすると同時に、上位のアプリケーションへと引き渡す。通常受信側では、ウィンドウ・サイズに相当する分だけの受信バッファを持っており、そこへ受信したTCPパケットからデータを取り出して格納する。

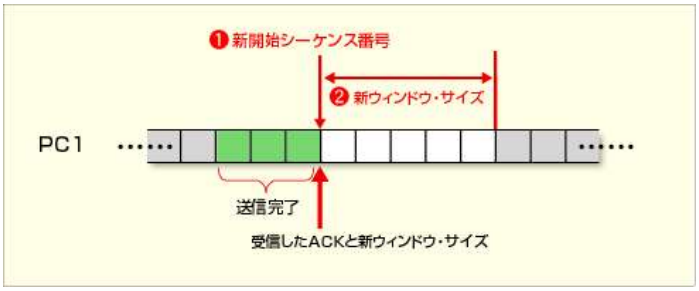
データを取り出したら、そのbytes数分だけ、受信ACK番号を進める。この例では、**（1）**のように、3bytes分だけ受信ACK番号を増加させる。次に受信ウィンドウ・サイズを、受信したデータの分だけ小さくする（**（2）**）。

次に、新しい受信ACK番号と受信ウィンドウ・サイズをTCPパケットにセットして、送信側へと返送する（**（3）**）。

以上で受信側の操作は完了である。受信側では、データを受信したことを確認し、自身の管理している情報（受信ACK番号とウィンドウ・サイズ）を更新し、ACKパケットの返送までを行っている。送信側では、送信がまだ完了したことになっていないので（ACKを受け取っていないから）、両者の管理しているシーケンス番号やウィンドウ・サイズの情報には、ずれが生じている。

送信手順―3．ACKパケットの受信

受信側が送ったACKパケットを受け取ることにより、送信側では、先に行った送信操作が完了したことを始めて認識することになる。ACK受信後の送信側の状態は次のようになる。



ACKパケットの受信

送信側では、ACKパケットを受信して初めて送信処理がすべて完了したことになる。ACKパケットを受信できなかった場合は、以前のパケットを再送信する。受信したACKパケットには新しいシーケンス番号とウィンドウ・サイズが入っているので、それに基づいて情報を更新する。

（1）新しいシーケンス番号は、受信側のパケットにセットされていたもの。これより左側のシーケンス番号領域はすでに送信が完了していることが保証されているので、再送に備えてとっておいたデータはもう解放してもよい。

（2）受信側では、データを受けた分だけバッファの空き領域が少なくなり、その分ウィンドウ・サイズが縮小している。この例では、今度は5bytesまではいっぺんに送信してもよいことになる。

ACKパケットには受信側が確認した新しいシーケンス番号が含まれており、送信側では、それを見て自身の開始シーケンス番号を更新する。と同時に、さきほど再送に備えて取っておいた送信データのコピーを破棄する。なぜなら送信が正常に行われ、もう再送する必要がなくなったからだ。

さらにウィンドウ・サイズを、ACKパケットで通知されたものに更新する。この場合は、残りのウィンドウ・サイズは5bytesとなっている（**（2）**）。

ここまでの操作で、一連の送信処理は完了である。この後は、上位アプリケーションに対して送信完了を通知し、次の送信データを待つ。だがもしACKパケットが一定時間経っても戻ってこなかったら、パケットを何度か再送し、ACKを待つ。規定回数再送してもACKを受信できなければ、送信エラーとしてコネクションをクローズしたり、上位アプリケーションへエラーを伝えたりする。

ウィンドウ制御によるフロー制御

以上の手順で、データが正常に送信され、それにつれてシーケンス番号も増加しているのが分かったであろう。だがウィンドウ・サイズは当初よりも小さくなったままである。もし次にさらに5bytesデータを送信したら、どうなるであろうか。

結果は、シーケンス番号が5つ増え、ウィンドウ・サイズは0になる。ウィンドウ・サイズが0ということは、送信側から見ると「もうデータを送信してはいけない」という意味であり、受信側からみると「送信を停止して欲しい」ということを意味する。これは、いわゆる「フロー制御（データの流れを許可したり、禁止したりすること）」として機能していることになる。TCP/IPではこのように、現在のウィンドウ・サイズを受信側から毎回通知することにより、ウィンドウ制御とフロー制御を同時に実現している。

送信手順―4．ウィンドウ・サイズの拡大

データ・パケットの受信によってウィンドウ・サイズは縮小されるが、逆に受信バッファに空きができた場合には、ウィンドウ・サイズの拡大を行う必要がある。これは受信側の仕事である。

受信バッファへコピーされたデータは、上位アプリケーションへ通知され、アプリケーションへと渡される。だがこのデータの引渡しは、TCPのデータ・パケットの受信時に毎回すぐに行われるわけではない。通常のアプリケーションは、TCPコネクションからのデータを常に待ち続けている（ポーリングし続けている）わけではないし、データ・ドリブン（データが到着したら、それをきっかけにして処理を開始するような構造）になっているわけでもないからだ。

そのため通常のTCP/IPのプロトコル・スタックでは、受信したデータを自身自身の持つ受信バッファにコピーし、すぐに送信側に対してACK応答を送信する。上位アプリケーションは、TCP/IPパケットの受信タイミングとは関係なく、自分自身の準備ができた時点で（非同期で）受信バッファからデータを読み出せばよい。

TCP/IPのプロトコル・スタックでは、受信バッファからアプリケーションにデータが引き渡され、空きが増えた場合にウィンドウ・サイズを増加させる。新しいウィンドウ・サイズの通知は、次回ACKを送信する場合に行ってもよいし、直ちに送信してもよい。このあたりもやはり実装依存となっている。

以上のような仕組みになっているため、アプリケーションのデータ処理速度がネットワークの受信速度よりも十分速ければ、ウィンドウ・サイズはずっと大きいままで、帯域幅をフルに活用することができる。逆にアプリケーションの処理速度が遅ければ、ウィンドウ・サイズが自動的に縮小して、データの受信を抑制する。

次の回へ »

インデックス 「連載 基礎から学ぶWindowsネットワーク ―― Windowsネットワーク管理者への道 ―― 」

前のページへ 1 | 2 | 3

基礎から学ぶWindowsネットワーク 連載一覧

全 23 回

「ネットワークが分からない」状態からでも丸ごとサポート **New!**

「ほとんど誰も見ていない」社内ポータル、どう変えるべき？ **New!**

データは「守りながら活用する時代」に

自分が作ったアプリがスマホで動くさまを見ると、学生の目が輝くんです **New!**

@IT Special へ

この記事に関連する製品／サービスを比較（キーマンズネット）

- L4負荷分散とL7負荷分散どちらを重視？『ADC／ロードバランサ』製品一覧
- 構築したいネットワーク要件で大きく変わる『ルーター』の選び方
- まずネットワークの性質を十分に見極めよう！『ネットワーク管理』製品比較
- 信頼性や可用性に対する取り組みは？『ネットワークスイッチ』製品比較
- 既存のネットワーク構成とマッチする？『WAN高速化』製品の選び方