Duplex Communicate Network の構築

Approach the Duplex Communicate Network

石原 秀和(*1)　　羽賀 隆洋(*2)
Hidekazu ISHIHARA　　Takahiro HAGA

Abstract : The "*Duplex Communicate Network*" (*DCN*) is new meaning of "Today's Network". The *DCN* have a few connection for other Application is able to connect some Network System of various protocol by the DirectX(Direct Play)(*3). On the security side, *DCN* have a original cipher coding. This is pointed, too. This coding is keyword cording of unsettled long.

1. はじめに

インターネットの高速普及により今までコンピュータに興味が無かった人々までもが自宅でウェブサーフィンをするようになった。これまでハイテク業界や学校、一部の企業からしか興味を持たれなかったネットワーク産業が目に見余る伸びを見せ、その付加価値産業も大きな発展を遂げている。それらの産業の多くは、ネットワークを利用したマーケティング情報管理や自社のネットワーク在庫管理からチャットシステムまで広く開発が進んでいる。そして、そのどれもが殆どその方向性が完成してしまっている。もちろん、それを掘り下げてさらなるユーザ・ビリティに富んだソフトウエアを開発する事も可能であるが、今回の研究では新たな試みとして、テレビなどでは実現不可能な双方向性を生かしたアプリケーション開発手段として用いることのできる仮想のネットワークシステム:Duplex Communicate Network(DCN)の構築を試みる事にする。

この研究ではMicrosoft社のDirectXを主に利用している。DirectXは同社が提唱する32bitWindows上で動作するCOM(Compact Object Module)が集まったAPIである。このAPI(Application Programmer Interfaces)を使用する意図は本来マシンが持つ性能を最大限に生かすところにあるが、ユーザが所有する様々な環境を一つのモジュールによって緩衝する為の意味合いもある。今回ネットワーク接続の為に使用したDirectPlayもユーザの所有するネットワークカードやモデム、COMポートのIRQなどを全く気にしなくとも殆ど全てのユーザ環境に対応する事ができ、これが意味するところは大きい。作成したソフトでは起動時に使用する回線及びプロトコルを選択できるようにしてある。起動時の選択により殆どのネットワークで使用できるであろう。また、起動時には他の設定もできるようにした。グラフィックカードやサウンドボードもその性能を最大限まで引き出せる設計にした。

2. 研究プロセス

研究に当たる前に開発プロセスを決定し、研究を円滑に行う事ができるようする。

今回開発するソフトウエアは研究内容が*Duplex Communicate Network*である事からも解るように、かなり抽象的になる事が予想されるが、かと言って単にネットワークを利用できるようになったソフトを作成しても面白い研究と言えない為、そのネットワークを利用した実用的なソフトウエア開発を研究の主体としたい。しかし開発にあたり実際に通信を行える段階に持っていってから応用ソフトの開発を行った方が無難である。従って、以下のプロセスにより開発を進める事にした。

[研究プロセス]
①暗号化技術の研究
②暗号化ソフトの開発
③DirectXシステムプログラムの設計／開発
④DCN通信プロトコルの設計
⑤DCNの構築
⑥汎用ソフトの試作

*1.愛知工業大学　情報通信工学科4年生(豊田市)　　*2.愛知工業大学　情報通信工学科(豊田市)
*3.DirectX及びMicrosoft WindowsはMicrosoft Corporationの米国及びその他における商標です。

上の手順で開発をすることにする。次章から行う研究内容の報告についてもほぼ上の順序で順次報告していく。コンソールはCUI(Character User Interface)にするかGUI(Graphical User Interface)にするか考えたが、暗号化ソフトはGUIにし、その他のソフトはDirect Drawを使用する為にGUIに近い特殊なユーザ・インターフェースにする事にした。また、研究内容に対し実際研究を行える期間が短い為、全ての研究内容を完結して報告するためにはかなり苦痛が伴う事になるが、寝る間を惜しんで開発に専念した事をご理解頂きたい。

さて、導入はこれで終了して次章からは実際に行った研究内容をできる限り簡潔にまとめて報告していきたい。

3. 暗号化方式

DCNに実装するかどうかはともかくとして、ネットワークと銘打つものを作り上げる時、システムの保全性から見ても暗号化技術は切っても切れない物である。そこで、まずはDCNが実用化に至ったときに実装されるべきであろう暗号化技術を研究した。しかし、暗号化のための専門書を一切読まなかった。本来は専門書を読んで過去の遺産を継承して新技術を見つけ出していくのであろうが、以前から考えていて実現可能であろうと推測していた単純な上に解読が難解な暗号化方式がある為にそれを使いたいと思う。

暗号化の歴史は古く、悲しいかな暗号技術は主に戦争が起こるごとに技術が向上している。日本軍も暗号を使用していた事は有名な「トラトラトラ」で誰もが周知の上だろう。この暗号はあらかじめ伝達がいっている暗号の意味により日本軍にのみ意味が分かるというお粗末な方式であり、暗号化の基本でもある。

暗号のやり取りを行う対象は3種類に分けられる。まずは送り主、暗号文を送る人である。次に受け主、暗号を受け取る人である。そして最後が傍受主、暗号を傍受した人である。暗号化は要するに傍受した人が解読不可能にする事であるが、3人が全員、同じ情報量を共有した場合絶対に傍受主が解読可能になってしまう。安全な暗号化とは送り主と受け主に傍受主が知らない特殊な情報及び取り決めを共有することにより実現できる。基本的であるが、まずはこれが意味する事を正確に理解しなくてはならないと言えるだろう。

暗号を考えるときに、まずは暗号の送り主と受け主が共有する特殊な情報及び取り決めで何を採用するかを考えなくてはならない。さて、ここで注意しなくてはならない事がある。暗号に対する取り決めとは要するに暗号化方式のことであるわけだが、解読を難解にする為に暗号化の方式を必要以上に複雑にしてはならないということである。一般にネットワークで暗号化を行うとき、バッチ処理ではなくリアルタイムで行わなくてはならないからである。複雑にしてはならないというのは語弊になるので訂正しておくと、ネットワークでネイティブに使用する暗号化方式及び復号化方式は複雑でもかまわないが、リアルタイム送受信が可能な程度の高速性を持ったアルゴリズムを提供できる方式でなくてはならないということである。この事をしっかり頭に入れておけば、開発ソフトは実用にも耐えられるだろう。

では、次から実際に暗号化方式を紹介していく。今回採用した暗号化方式は基本的にはとても簡単な方法である。が、解読が不可能に近く、暗号／復号化時間も短いところに特徴がある。また、バイナリ転送を前提として暗号／復号化を考えたので全ての処理は論理演算によって暗号化を行う。

演算はほとんど排他的論理輪和(Exclusive-OR:XOR)をを用いる。まずはXORの利用可能な特性を考える。XORには下の式①のような基本的な特性を持っている。

・排他的論理和の基本性質

$$A \hat{} B \hat{} B = A \quad \cdots\cdots ①$$

(^はXOR符号とする)

式1から考えられるにこの論理演算には暗号化の基本理念を伴った特性を持っている事がわかるだろう。Bという情報を送り主と受け主の共有する情報にすれば、暗号が成立する。と、ここまでは誰もが考えられることだろう。暗号技術として打ち出すにはこれらを利用しつつBの生成方法及び規則的暗号化の解除である。規則的暗号の解除は、Bの集まりであるB群という要素の一部が書き換えられた場合に違うBに形成し直す事で可能となる。B群の規則性が無くなれば、傍受主にしてみれば、Aに対しての規則性も無くなることも解るだろう。では、Aの規則性が無くなればどうなるかと言えば、A^BよりAを解析する傍受主にとってはAを個々に解読するより方法が無くなるのである。しかも、一般にデータは複数あって初めて意味をなすために解読成功であるかどうかの確認を取ることが難しくなるのである。まず考えてもみてほしい、排他的論理和は他の論理演算と違

違い、桁あふれも起こさなければ元の数値がどんな値にも関わらず、例としてバイト単位の演算の場合においても00H～FFHの全ての演算結果をもたらす事から暗号化に適した演算方法であると言える。

B群から違うB群（B'群とする）を生成する訳だが、これもXORを使って簡単に行うことができる。

・コードブックの再生成

$b_0 = b_0 \hat{} b_{n-1}$ …… ②

$b_i = b_i \hat{} b_{i-1}$ …… ③ （iが0以外の時）

（B群はn個のb_iの集まりであるとする）

上の式②及び③をn-1回演算することにより、全てのb_iに対してその他全てのb_jが影響を与えることになる。このとき重要になるのが、nの値が任意で良いという事である。そしてnの値が任意で良いということは結果的に解読難度を変化することができる暗号化方式になることが解る。さらにはこの生成方式を使用する時には、nの値が増加することにより解読難度が偏らず、B'群全体で上昇することがわかる。今回はさらに暗号化に用いる最初のコード位置もB群全体の演算結果から求めることにより、不規則性を強めた。

さて、次は自分が暗号の解読者として考えてみる。その時、まずは暗号のコード群の長さ、つまりB'群の長さにについて調べるだろう。何らかのアルゴリズムによりその規則性を見つけ出しその周期ごとの平均から推測することになると思うが、これの解読難度を上昇させるにはどうすればいいのだろうか。

・データの再生成

$a_i = a_i \hat{} a_{i-1}$ …… ④ （iが0以外の時）

上の式④をA群に対し行えば、基定値となる値が無くなるために解読は困難となる。あとはB群の事であるコードブック作成を困難にするためにそのコードブック自体も変異させるべきであるが、コード群長さが不定のため、長いコードの時にトランスレートを低下させる危険性があるのでやめておいた。

この様に暗号化自体は非常に簡単な手段となっているが、どれくらいのクオリティを持っているのか暗号化のみを行うためのソフトウエアを試作してみた。次章ではこれを紹介する。

4. オリジナル暗号化の有効性

暗号化実験のみの目的で*Protector.exe*を開発した。このソフトでは暗号化による出力を複数用意した。それの出力形態はバイナリのデータからネットで転送できる文字のみで構成させるデータに変換するためにあり、用意した3種類はASCII(7bit)/1ByteJIS(8bit)/All of JIS(8,16bit)である。これらは変換後の出力長を最小にする為の工夫が凝らされた未公開の形式にしてある。

では、実際に暗号化を行ってみたいと思う。

表1：暗号化処理データ ～その1～

データ（A群）	今期の決算は黒字傾向にある
キーワード（B群）	本日の暗号化キーワード
暗号化されたデータ	30イ6:89ヌ>808Nyモ6308l0:=?
f(A'B'^A')	5?tセムセS6::9¥ツ&

上の表1で暗号化に使用した内容を掲示した。まず、データは受け主に送りたいデータのことである。バイナリデータでもいいのだが、上では「今期の決算は黒字傾向にある」という文字列にした。キーワードは暗号化の為のコードブックで、これもバイナリでもかまわないが「本日の暗号化キーワード」という文字列にした。このデータ長によって暗号解読の難度が上昇することになる。今回の場合は22[Byte]使用している事になり、22[Byte]の場合発生しうる暗号の種類は256の22乗通りとなり、解析には非常に苦しむ事になる。今回作成したソフトでは、キーワード長の制限が64000[Byte]であるために最高で256の64000乗通りの難度を持たせることができる。さらに、ここからが重要なのだが、先ほどから言っているコードブックの長さは各暗号に対して変化するので、解読者はそれに対しても解読しなければならない。

さて、これまで解読の難度ばかり解説してきたが、この暗号化方式は高速に行えるところに本当のメリットがある。ある程度の長さのデータにおいても時間を全く必要としない。さらに、上の表1で暗号化されたデータは1ByteJISにより出力されているが、バイナリ転送が可能な場合には暗号化データの長さはこれより小さくなる。

次にもう一つの例を挙げてみることにする。次の図2に於いては図1の暗号化されたデータと見比べて頂けばその違いが解ると思う。

表2：暗号化処理データ ～その2～

データ（A群）	今期の決算は黒字傾向にある
キーワード（B群）	本日の暗号化コードブック
暗号化されたデータ f(A^B'^A')	pワ69^11|510:9:91ミ29ロ001? S>19:935?]-Zフ1>;>

表2を見てもらえば解ると思うが、先ほどと違うのはキーワードが「本日の暗号化コードブック」になっているところである。前のキーワードとの共通情報が半分近くあるというのに暗号化されたデータを客観視したところで全く法則が見つからない。これは、前章で書いた規則性の解除が有効に働いている証拠である。

では、最後に解読者に対する挑戦として、次に暗号化済みのデータを挙げておく。是非、以下の暗号文を解読してみてほしい。だが、解読し始めると誰もがシンプルが上の難解さに気が付くだろう。ヒントを提供すると、データは日本語の文章で多重暗号化(*1)が施されているということである。

表3：暗号解読者向けデータ

51(gH-ZナX1>ス93>0#3>R3:10iハヤ01=15?51ト,51<:y08レ
<9メム>?u=>8?30>:?9ア9:90!0:80%サリヒ73489:;?3>1?ナC
¥19ケ4?58<8ヤ60+?0s=8uセ??)>0ノ11>93?8060R]41レヤO<
3レ?0>9ナ"¥ヒ099:N;090メ:0ミ7>9:^;0W=0k:>ノ68タ>:19ス
8>?:オ1:>>ス79ク3:21TイD09N4:3>へ5:3?98^9889Uシ9371
83a3909タ::/A>11?0;=?=143FVへJ0?ZA=9yカ!28cナtnルヨ
3001ケ?3.6>,Z?021<9イマ>84>6:ネ2:G11ア?7X2:::61=?
>3ヤへ{88Qルヒ1?51_I19Y:0<8;8T61;>?>モj<0::931178ム
ト63R?>$]>1/D.-T10~?0K;>5883O19ユX=1D~;:#6>]>:シ
>>;1^2>2?0>>?8?クo1858:9uレ}セ :3<:1321J6963?>1?
e??9>19ノ:?584:;8715:?:03;?s28?>AFム,8:L,60B=>:
1dタ8:2:91m30Dテ59T

上の表3のコードを正確に解読できれば、11行の日本語文章を見ることができる。

5. システムプログラム

この研究は他のアプリケーションにDCNを提供するのが目的であるために、作成されるシステムが結果的に研究成果となる。そうなるとシステムプログラムの設計も再利用可能な形で他人に提供できるものにしなくてはならない。なお、システムの設計はWindowsプログラミングに準拠した形のものになる。そして環境の初期化には作成したシステム初期化関数をプログラム中の任意の位置で呼び出せば良い。初期化のための情報は起動時に表示されるウインドウで細かく設定する事ができる。下図は起動時のウインドウである。

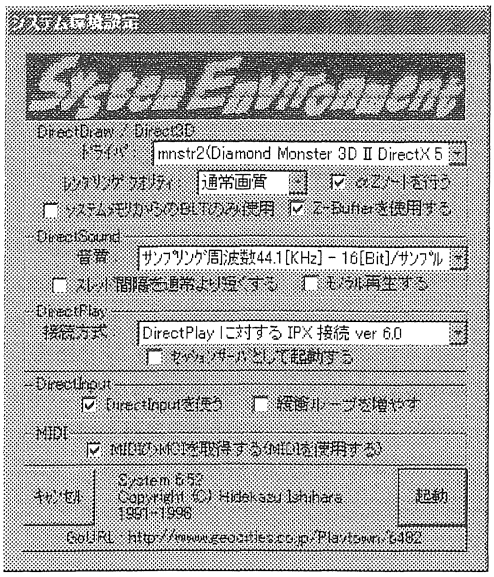

図1：環境設定ウインドウ

上の図1のウインドウでシステムの環境を設定すると、システムはそれらの情報から各インスタンス(*2)を生成し、管理するので利用者は初期化するだけで良い。

6. DCNアプリケーション上のプロトコル制約

システムプログラムが使用するプロトコルはDirectPlayの使用できる全てのものになるのだが、一般のアプリケーションからDCNに対するプロトコル制約を考えなくてはならない。通常はシステムを完全にアプリケーションモジュールの一部から独立した形態でDDEメッセージ等を使用してモジュール間でのギャップを埋めるのだが、このシステムはデベロッパーに対して完全に解放することを基に考えているので、システムプロトコルはデベロッパーが自由にターゲットソフトの特性を考えて最適化したプロトコルを設計する事ができる。

*1.多重に暗号化をかけてあるが、暗号解読に用いるキーワードは同一である。

*2.ほとんどのAPIがCOMの考え方で提供されている。

では、ターゲットソフトに最適化したプロトコルとは、どう設計すればいいのだろうか？*DCN*を使用する場合、*DCN*に接続する端末はセッションサーバとセッションクライアントの2種類に分かれる。*DCN*デベロッパーは同一のプログラム上で別々の設計をしなくてはならない。そして*DCN*でネットワークを使用するほとんどの場合、小規模なネットワークで通常使用されるようなクライアントーサーバ式の同期取りを使用するのが適していると考えられる。*DCN*で作成されるアプリケーションは、双方向性を持ちつつ同時に動作する別々のプロセス群であり、それら個々のプロセスは他のプロセスの更新情報をリアルタイムで欲している。シェアリングを用いる時間的な割り振りを各プロセスに対し実際に任せると高度な理論が必要になる。また、その理論通りプログラミングしても接続人数が少ない場合はサーバが一手に行った方が速い。しかしサーバが全てを担当する場合、作成するターゲットソフトに対してのメッセージングで問題点が発生する。商用のバーチャルモールとリアルタイム株式情報処理ソフトとチャットでは、必要とされるプログラマのメッセージング技術は全く違うのである。

*DCN*におけるクライアントーサーバのメッセージングを各アプリケーション独自で特化して、全てまかせる事はデベロッパーの技術を生かすためであると上で述べた通りであるが、ここではチャットに対して最適化したメッセージングを実際に考えてみることにする。

クライアントーサーバ型はクライアントがまずサーバに対し要求をだすところから始まる。これに対しサーバが許可を与えクライアントの通信及び処理が行われる。*DCN*のチャットの場合、キーボードの押下情報に対しリアルタイムでレスポンスを行うことも比較的楽に可能になる。つまり、1行単位のメッセージングではなく、個人が意見を発信しようとしたその時点から相手に意思の疎通が可能になるということである。もちろん現行のネットワークでもプログラミング可能であるが、*DCNを使用*すれば手軽に行うことができる。また、そのチャットソフトを作成する時に問題となるのが各人の発言をいかに表示するかであるが、これは全く問題ない。まず発言がが最初に発生したときに発言表示用の位置を取得する。その後はその位置に随時追加及び削除の文字列操作を行う。もし、スクロールして発言領域が領域外に出てしまった場合、次に状態変化が起こった時点で最新の発言情報として表示すればよい。この方法により日本語が独自に持ち合わせる先読みの意味理解をする事もできるようになる。まさに通常会話するが如くネット上でおしゃべりが可能になるのである。これが*DCN*を有効に利用したチャットである。

下図に*DCNアプリケーション*を作成する上での位置付けを示す。線で結ばれているものの間にはプロトコルが存在し、データの流れを表している。

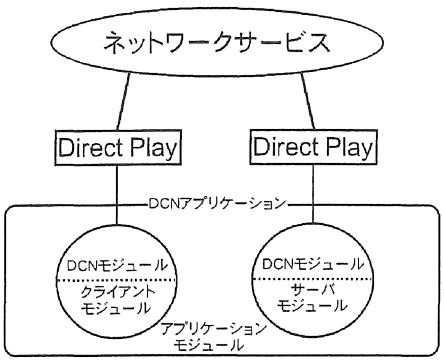

図2：DCNアプリケーションの形態

7. DCNアプリケーションの作成方法

*DCNアプリケーション*はDirectXを使用している為にWindowsプログラミングに準拠して作成することを前に述べたが、ここでは実際にどの様に作成するかを述べる。しかし、現在のWindowsプログラミング法には大きく分けて2種類存在している。どちらを採用するかはデベロッパーが決めることであるが、*DCNシステム*はライブラリかDLL(*1)として提供されるためにどちらでも対応できる様になっている。今回はWIN32アプリケーションとMFC(*2)を利用した2種類の作成法からWIN32アプリケーションとしてMFCを使用しないソースを作成する方法を述べる。ちなみに、前の章で紹介した*Protector.exe*はMFCを利用して作られている。一般にMFCを利用して作られるアプリケーションは高速性を伴わなくても良いオフィス商品に向いている。アルゴリズムを洗練して高速性を追及するなら使用しない方が適しているだろう。プログラムのルーピングに際しても40～50[ms]サイクル位まで対応することができる。マシンクロックの都合

*1.Dinamic Link Liblaryの略で、ライブラリ形態でポインタによりライブラリ関数を呼び出す。

*2.Microsoft Fundation Classの略で、Microsoftが提供するWindowsプログラミングの枠組み。

によりこれ以下の高精度の時刻取得は必要性もない事からも行っていないが、1クール10[ms]位までなら可能である。

さて、*DCNアプリケーション*ではまず初めにユーザから使用するシステム環境を指定してもらい、その情報をシステムがインスタンス生成情報として取得しなければならないが、その作業には*DCNライブラリ*が提供する関数をプログラム開始直後に呼び出せば良い。

リスト1：システムの初期化

```
if ( !initSystem() ) return FALSE;
```

上のリスト1の一文を加えると、initSystem中で2つ前の章で紹介した図1の環境設定ウインドウを生成して各情報を取得する。取得した情報はシステムに記憶され、後からインスタンスを生成するときに*DCNシステム*が利用する事になる。

システム環境の取得が終了したなら次はウインドウの生成に移る。方法は通常と変わらない方法で良い。ウインドウの各種情報をレジストして生成すれば、生成後直ぐに生成されたウインドウ・プロシージャにWM_CREATEのメッセージが送られる。この中で使用するインスタンスを生成すると良い。

リスト2：各インスタンスの初期化

```
case WM_CREATE:
        existMMX();
        if ( !initDirectPlay() ) {
                ErrorFlg = TRUE;
                GetCursorPos( &pCursorPos );
                DestroyWindow( hwnd );
                break;
        }
        initDirectSound( hwnd );
        initMidi();
        if ( !initDirect3D( hwnd ) ) {
                ErrorFlg = TRUE;
                GetCursorPos( &pCursorPos );
                DestroyWindow( hwnd );
                break;
        }
        initDirectInput();
        initCursor();
        startApplication( hwnd );
        break;
```

通常はインスタンスの生成にかなり面倒な作業を伴うのだが、上のリスト2によりその全てを代行してくれる。初期化の順序についてはリスト2の通りでなくても構わないが、中には初期化順がシビアにかみ合っているところもある為、なるべく上の順で初期化する事をお勧めする。必要のないインスタンスの場合、呼び出さなければ生成される事はないので必要なもののみ初期化する事もできる。リスト2は全てを初期化する例である。なお、初期化関数のうちのexistMMX();はMMX命令(*1)の存在を調べているだけなので、呼び出さなくても構わない。その時は強制的にMMXを使用しない。

さて、初期化については上の通りで良いが、逆にアプリケーションを終了する時にもリソースの開放の作業が残っている。

リスト3：リソースの開放

```
case WM_DESTROY:
        cleanDirect3D();
        cleanDirectSound( hwnd );
        cleanDirectPlay();
        cleanMidi();
        cleanDirectInput();
        cleanCursor();
        exitApplication( hwnd );
        break;
```

リスト3によりリソースの開放が行われる。そして、リソースから生成されたオブジェクト等に関してもシステム自体が管理しているものに対しては全て元の状態に復帰する為の作業を行っている。デベロッパーは全く気にする事なくシステムを利用することができる。また、未生成のリソースに関して開放を試みた時に発生する致命的なエラーをを回避する為のルーチンも内蔵する事によりフリーズする事なく、そのまま通過していく設計であるので安心して取得及び開放ができる。

これでシステムの起動と終了が可能になったが、そ

*1.intel社の拡張命令セットの事。SIMD(Single Instruction, Multiple Data)技法を採用し、単一命令で複数データに処理する事が可能だが、使用するMMXレジスタは浮動小数レジスタの流用である。新たにKNI命令が追加される予定だ。

の他にもウインドウ・プロシージャ中に必要なシステム関数の処理が存在する。キーボードの押下に伴い呼び出さなければならない処理は以下の通りである。

リスト4：キーボード押下情報をシステムへ通知

```
case WM_KEYDOWN:
        if ( wParam == ESC_KEY ) {
            DestroyWindow( hwnd );
        } else pushKey( wParam );
        break;
```

上のリスト4のような処理文を挿入すればシステムに押下情報が送られる。ちなみに、アプリケーション終了キーの判定はここで行うのが適しているので、送る前に判定をしておけばいいだろう。後はウインドウ切り替えが行われた時に行わなければならない処理がある。

リスト5：ウインドウ切り替えに対する処理

```
case WM_ACTIVATE:
        if ( bActiveApp ) {
            if ( !(bActiveApp = wParam) )
                cleanCursor();
        } else {
            if ( bActiveApp = wParam ) {
                if ( !ErrorFlg ) {
                  initCursor();
                  nChangeActivePerCount = 1;
                }
            }
        }
        break;
```

リスト5までがウインドウ・プロシージャ中で呼び出されるべきシステム処理である。これらの関数は*DCNシステムのライブラリ*関数である為、*DCNライブラリ*を使用する事によりコンパイルできる。当の*DCNライブラリ*はソースそのものとDLL形式での配布を予定している。

さて、ここまでが*DCNシステム*を利用する上で必要なウインドウ・プロシージャの処理であるが、では*DCNアプリケーション*のメイン処理を何処で行えば良いかという話になる。もちろんメッセージ・ポンプを利用して頂いても結構なのだが、ここはリアルタイム性を強める為にWindowsからアプリケーションに割り与えられる時間でアプリケーションが待機している時間を全て使用する方が適していて、リスト6はそのメッセージ・ポンプである。ここから呼び出すMainProcess(hwnd);がメイン処理部となる。リスト7はMainProcess();内部でシステムが必要とする処理である。デベロッパーが行うのはこの中でLastUpdatePassTime[ms]だけの処理を行うプログラムを開発すればよい。その設計はシステムに依存しないので、全く自由で構わない。

リスト6：メッセージ・ポンプの空き時間を利用

```
while ( TRUE ) {
        if ( PeekMessage( &msg, NULL, 0, 0,
                          PM_NOREMOVE ) ) {
          if ( !GetMessage( &msg, NULL, 0, 0 ) )
                return  msg.wParam;
          TranslateMessage( &msg );
          DispatchMessage( &msg );
        } else if ( bActiveApp )
          MainProcess( hwnd );
}
```

リスト7：メイン処理ルーチン部

```
void MainProcess( HWND hwnd )
{
        passTime();
        if ( nChangeActivePerCount ) {
          if ( LastUpdatePassTime >
              dwTmpLastUpdatePassTime ) return;
          if ( nChangeActivePerCount == 1 )
                restoreSurface();
          if ( nChangeActivePerCount++ >
              nPassTimeBuffSize )
                nChangeActivePerCount = 0;
        }
        getKeyStatus();

        /* << 経過時間分の描画処理を挿入 >> */
        /* LastUpdatePassTime分の処理を行う */

        dwTmpLastUpdatePassTime
          = LastUpdatePassTime;
        flipPrimarySurface();
        cleanKeyBuff();
}
```

前述リスト7の注釈文がある位置にアプリケーションのメイン処理部を記述する。なお、*DCN*の機能を使用してシステムのネットワーク機能を使う場合、まずは作成するアプリケーションに適したプロトコル及び設計を行わなければならない。チャットを作成する場合に必要なネットワーク・メッセージはサーバへのハローメッセージとそれに対するアック、その逆のメッセージ、発言の要求と発言表示の要求、その他であろう。

これらのメッセージングにより実際に*DCNシステム*を利用したチャットを試作した。Direct Drawを使用している為にWindowsのウインドウ環境が使用できないのでオリジナルのウインドウ環境を作り、これをインターフェースとした。普通のチャットと違う点は前の章で述べた通りであるが、これを実現させるには通常のサーバークライアント方式とは違うメッセージングが必要となった。

7. ネットワークの利用価値と発展性

ネットワークは急速に発展している。今ある付加価値商品が明日にも需要があるかどうかは誰も保証ができないだろう。これから未来にかけて石油社会から電気社会に切り替わる大きな波があるだろう。その中で情報産業は他産業の影響を受けつつ更なる高速化、信頼性を見せる事になる。開発者が今まで空想のみで実現できなかった様な事ができるようになるだろう。例えば遠隔医療技術などがその走りであろう。離れた孤島において、緊急を有する患者が執刀を待っている時本島の優秀な医師が実際の手術と変わらない要領でそれが可能になるのである。ネットワークTVもさらに増える事だろう。そのネットワークTVでは番組で扱った商品やCMの商品をコンピュータ上で仮想的に試用できる様になるだろう。衣類の場合であれば、コンピュータにあらかじめ自分の身体情報を与えておくか、企業がその時提供するソフトに入力すれば外観及び着心地も伝えてくれる様になるであろう。

この様にネットワークは日常で行っている事はもちろん、日常では行えないような事も仮想的に可能となる。が、ネットワーク社会がもたらす問題として文化面から環境面まで至る様々な問題が有ることを忘れてはならく、利用者の理解を進めていかなければならない。

都市化の進む社会では精神の孤独が進んでいるが、ネットワーク社会でも同じ問題が生じる事になる。そして、環境破壊が深刻な問題として叫ばれる昨今、電脳社会が抱える難問も多く絶対に目を背けてはならない分野である。良い点も多くある。例えば、商品の梱包に於いてである。ネットワークで商品を販売する限り見た目に付く派手で奇抜なデザインでなくても良くなる。また、手軽にできる事が多くなり、近くの店に行く為にも車を使用していた人々が無駄に出歩かなくなる。商品の移動手段が最小限のエネルギーで良くなるのである。しかし、ネットワーク自体が普及したり文化として取り入れられるには多くの資源を必要としてしまう。これはあらゆる産業に対し依存する事だが、技術自体の開発も大切であるが、これからは限られた資源を利用しつつ新しい資源を見つけ、いかにクリーンな発展を遂げるかが人間の地球上に於ける存続にとって一番重要ではないかと思う。実際の話、歯止めのかからない消費に対して自分ができる事など微量かもしれないが、地球上で人間が地球を痛く汚している事実を直視しなければならない時に来ているのだと思う。

この研究報告の最後にあえてこのような事を記述するのは、今、自分がとても大きな恐怖に駆られているからである。大地には木々の根の変わりにケーブルが敷かるようになり、そのケーブルに流れる情報の中には犯罪に纏わるものから自殺願望の悩み等々、種の悲鳴にも似た情報も交錯している。現実にもネットワーク独特の匿名性を利用したネットワーク犯罪も増加している。もちろん犯罪が存在すれば、それを取り締まるものが構成されるのも自然で、最近になりようやく動き始めた様だが完全な予防や駆除を行う事は不可能近いのである。草の根ネットなどのBBSでなくとも、FTPで誰の目に付かれずファイルをアップできる現状では普及しているインターネットでも駆除はできても完全な予防は不可能である。管理者自体が犯罪に手を染めている場合すらも貴ではないのが現状である。

電話や手紙、そしてTVに変わる大きなメディアになりつつあるこの秀逸且つ問題児である文化を、私達はどうやって受け入れていくべきなのだろうか。今やネットはその色、深さ共に想像を超えて広大である。

（受理　平成11年３月20日）