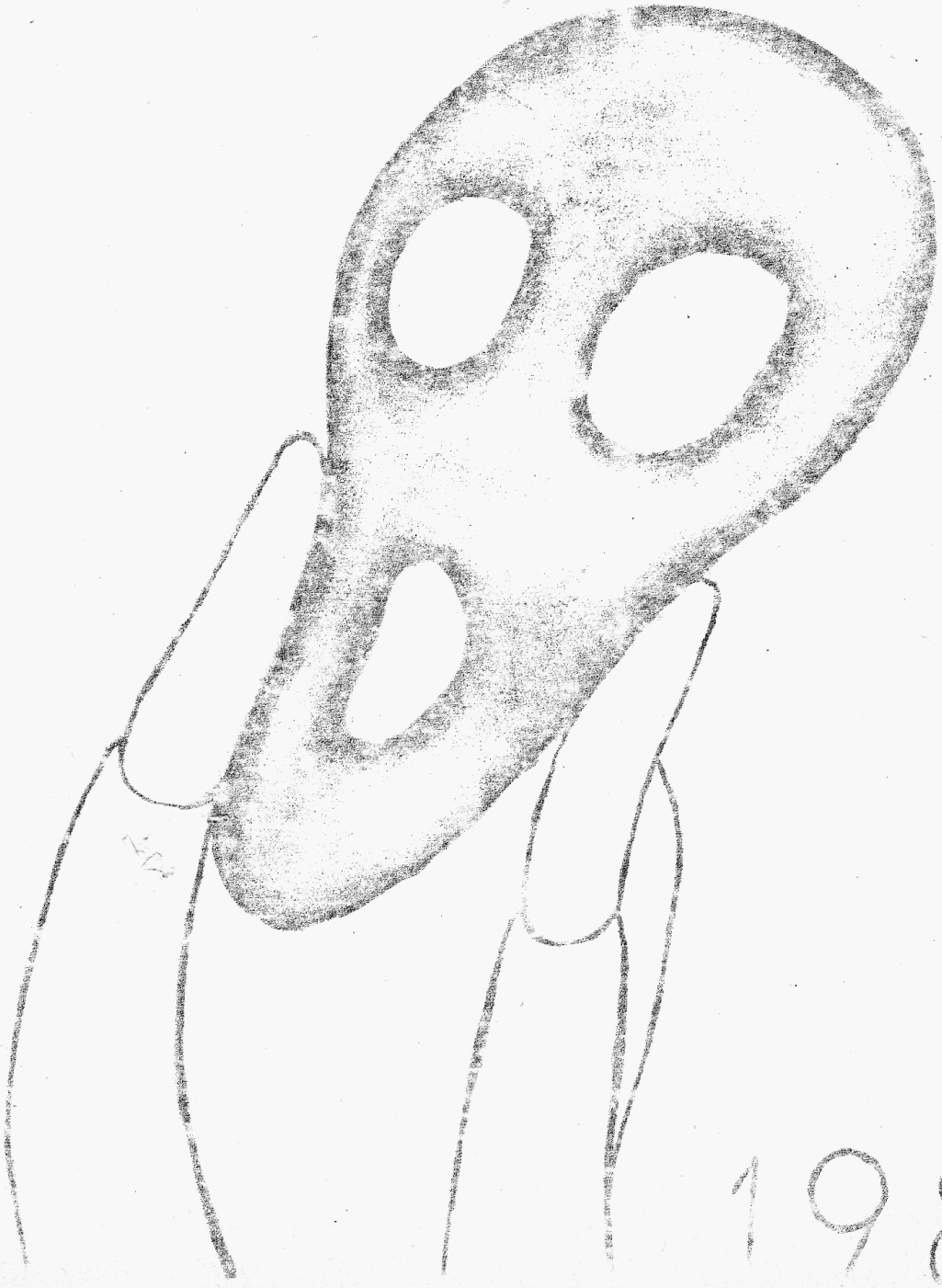
MEMA



1987

MMA 調布祭 パンフレット 目次

＊＊＊＊　　部長さんのお言葉　　＊＊＊＊

最近３２ｂｉｔマイコンを使ったパソコンも本格的に市場に出回るようになって、ＭＭＡも、その活動の意味を再確認せざるを得ないようです。創立当時の「欲しいものがないから、自分で作った」という状況は１０年間の間に大きく変わりました。今時の３２ｂｉｔの構造の複雑さや、メモリアクセスタイムの短さは、今までのような単層基盤にラッピング配線というやり方を困難にしています。かといって、手作りマイコンの為にたった一枚の四層基盤を作ることも金額的に釣り合いません。

ソフトも同様です。一昔前のミニコンクラスに匹敵するＯＳ等が、新しい３２ｂｉｔパソコンの上でも走るようになりました。我々のシステム上でもこれらのＯＳの一つが走りますが、そこまでの苦労はこれからのサポートの手間を除いても購入金額より割高になるでしょう。

では、我々はどうすればいいのでしょうか。我々もパソコンを買ってきた方がよいのでしょうか。その方がいいという人もいるでしょうが、我々はやはり、今までの方針を貫くと思います。

我々は丁度手製のカヌーで川を下っているかのようです。川幅が広くなればなるほど、他のカヌーとの差も開いて行きます。手製のオールが他の人より小さい為なのでしょうか、それともひょっとしたら支流に迷い込んだかもしれません。他のカヌーに遅れてしまったことを、決して残念でないとは言いませんが、自分達で作ったカヌーで川下りすることの面白さが、どうしても忘れられないのです。いつかは皆に追い付くこともあるかもしれません。

システムを手作りしていくことの良さ、それを我々は大切にして行きたいと思います。

＜ＮＥＴＷＯＲＫ＞

§１．プロジェクトの目指すもの

現在MMAでは、コンピュータ間に高速の通信手段を提供するネットワークプロジェクトが進行している。このプロジェクトの目指すのは、２～３Ｍｂｐｓの通進路によってコンピュータを接続し、リモートなホスト間での自由なファイルアクセスの手段や、リモートなプロセス間での通信を提供することである。
倶楽部内には数多くのＯＳ９／６８ｋのシステムがあることから、初めはそれらへのインプリメントを行い、徐々に他のホストへのインプリメントをおこなっていく予定である。
そもそも、ネットワークプロジェクトの発足の契機となったのは、異なるＯＳやＣＰＵ間での自由な通信の必要性であった。これを実現するためには、ネットワークのインプリメントは特定のＯＳやＣＰＵに偏ったものであってはならない。我々のアプローチは、次のようになった。

１．　ＯＳＩ７層のモデルでいう、物理レイヤからトランスポートレイヤまでをサポートするノードプロセッサを作製し、各ホストはその機能を利用してネットワークに参加する。
２．　ノードプロセッサとホストとの通信はシステムバス（マルチバス、ＶＭＥバス等）上の共有メモリと、割り込みを通じて実現する。
３．　ひとつのノードプロセッサに複数のホストを接続できるようにし、コストの削減をおこなえるようにする。

上のような方針に従って、現在２枚の、マルチバス上で動作するノードプロセッサが製作されている。この詳細については別の文章で触れることにする。

§2. ノードプロセッサの　ハードウェア

1）特徴

2.5Mｂｉｔ／ｓｅｃの　ネットワーク上のデータ転送速度

改良型トークンパッシング プロトコルを採用

１ネットワーク中に　最大２５５ノードをサポート

ＳＭＣ社ＣＯＭ９０２６を使用

アークネット　コンパチブル

インテル社Ｍｕｌｔｉ－Ｂｕｓインターフェースをサポート

殆どのＭｕｌｔｉ－Ｂｕｓマスターモジュールが使用できる

ノードプロセッサは　モトローラ社ＭＣ６８０００ＭＰＵを採用

2）ラインプロトコル

ラインプロトコルはそれぞれのバイトがスタートインターバルとストップインターバルで区切られているので、独立している。それぞれのデータバイトは一定の時間のインターバルを持ち１ｂｙｔｅの転送に４．４マイクロセコンドかかる。

送信がアラートバーストで始まり、８ｂｉｔのデータキャラクターが２ユニットのマークと１ユニットのインターバルをおいて送られる。以下に示すように、５種類の送信形式がある。

１．送信開始

アラートバーストに続いて３つのキャラクターが送られる。ＥＯＴと２つのＤＩＤキャラクターである。このメッセージは１つのノードから他のノードへトークンを送るために用いられる。

（注）ＤＩＤ　送信先識別子

２．空バッファ調査

アラートバーストに続いて３つのキャラクターが送られる。３つのキャラクターはＥＮＱと２つのＤＩＤである。このメッセージは他のノードにデータパケットを受け取る事が出来るかどうかを尋ねるために用いられる。

３．データパケット

アラートバーストに続いて以下のキャラクターが送られる。

- ＳＯＨ　（START OF HEADER）
- ＳＩＤ　（SOURCE IDENTIFICATION）
- ＤＩＤ　２個
- パケット長
- ＣＲＣ

4．送信許可応答

アラートバーストに続いて１つのキャラクターが送られる。

１つのキャラクターはＡＣＫである。（２）の答えとして空バッファがあるときに送られる。

5．送信不許可応答

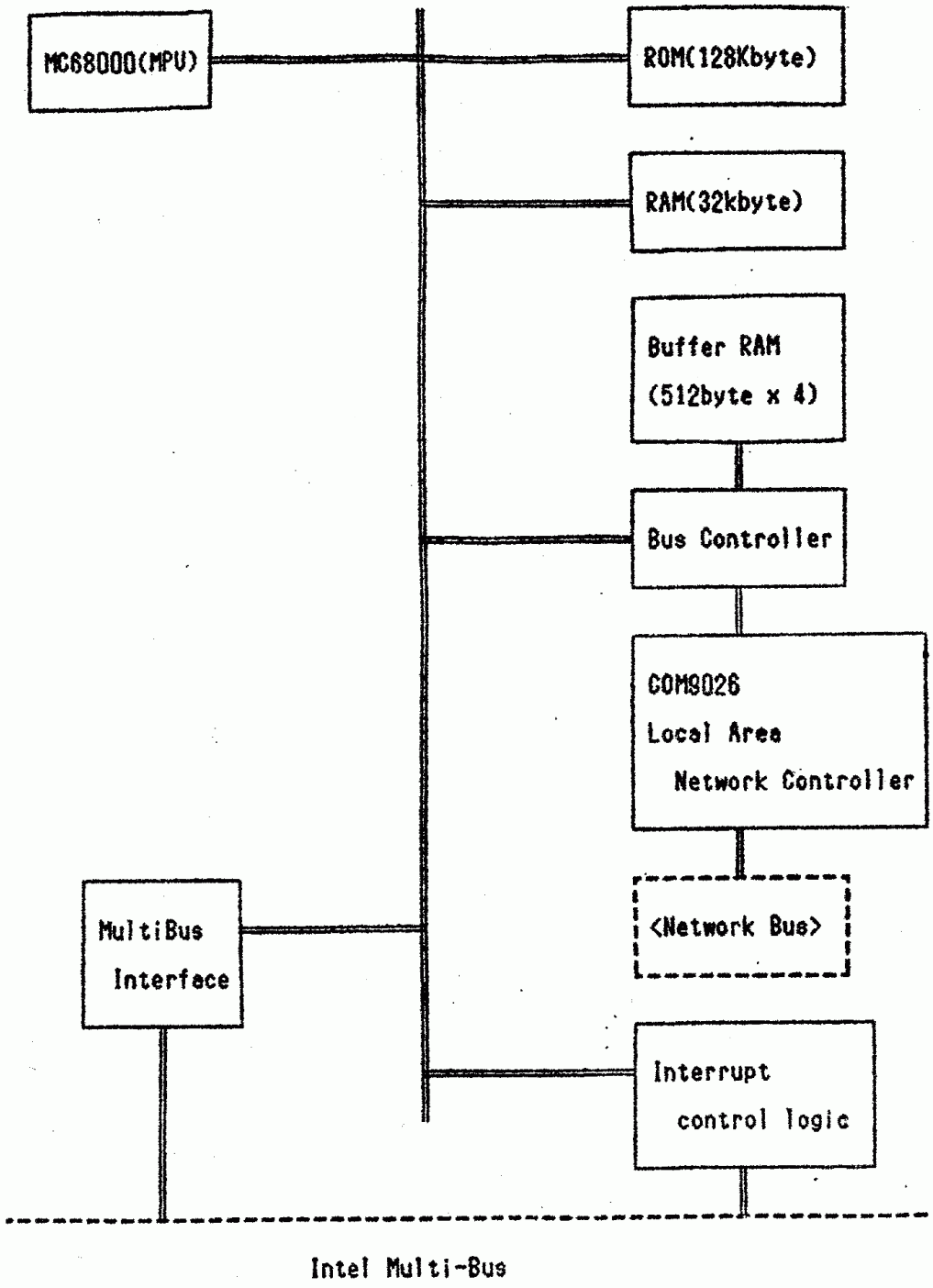
アラートバーストに続いて１つのキャラクターが送られる。

１つのキャラクターとはＮＡＫである。（２）の答えとして空バッファが無いときに送られる。

3）ネットワークプロトコル

ネットワーク上のプロトコルは”改良型トークンパッシングプロトコルに基いている。改良型トークンパッシングとはトークンをうけとったノードが応答を返すためである。ネットワークの構成と管理がＣＯＭ９０２６のマイクロプログラムシーケンサーによって行われる。プロセッサーやインテリジェントペリフェラルは送りたいデータパケットとディスティネイションアイディーを空バッファに送るだけで良い。するとＣＯＭ９０２６はつぎのトークンを受け取ったときに空バッファ調査を行い受信ノードが送信許可応答を返してきたときに送信される。パケットの受信が成功したならば受信ノードは（４）の応答をして送信が終了する。

4）構成図


MC68000(MPU)
ROM(128Kbyte)
RAM(32kbyte)
Buffer RAM
(512byte x 4)
Bus Controller
COM9026
Local Area
Network Controller
<Network Bus>
MultiBus
Interface
Interrupt
control logic
Intel Multi-Bus

§3．ノードプロセッサの提供するソフトウェアやモデル、サービスの種類について

ノードプロセッサ上にはＯＳＩ参照モデルにおける物理レイヤからトランスポートレイヤまでが実装される。

ソフトウェアでは、データリンクレイヤおよびトランスポートレイヤが実現される。これらによって以下の機能が提供される。

1．　データグラムによる通信（同報を含む）

2．　バーチャルサーキットによる通信

１はホストに対しパケット単位でのデータ伝送を提供するもので、誤り制御等は行われない。

２はホストに対し、バイトストリームによるデータ伝送を提供し、同時にフロー制御、誤り制御、シーケンシング等も行う。

ノードプロセッサの機能の最も重要なものの１つがこのバーチャルサーキットの提供であり、これは、データグラムの上に構築される。したがって、以下は、主としてバーチャルサーキットに関する説明になる。

バーチャルサーキットとは次の機能を用いて実現される、仮想伝送路である。

1. connection control
2. Packet assembly, disassembly
3. sequencing
4. data flow control
5. Error control

これらの機能に関して順に説明する。

1. Connection Control

ホスト内で一意に識別されるパケットのソースおよびディステイネーションをソケットと呼ぶ。２つのソケットの一時的な論理的接続をコネクションと呼び、バーチャルサーキットにおけるデータの伝送はこのコネクションを通じて行われる。逆にいえば、バーチャルサーキットの論理的なデータ伝走路がコネクションであり、その両端がソケットである。バーチャルサーキットにおける通信はまずコネクションの確立に始まり、以後、このコネクションを通じ以下に述べる機能を用いて伝送されるデータの正当性を保証することにより行われる。さらに、広域のネットワークにおいてはルーティング（複数の伝送路が存在する場合に、どの経路を選択するか）の問題が生ずるが、本ネットワークにおいては、ルーティングに対する拡張性を確保し、他のネットワークと接続する場合は、ゲートウェイ（他のネットワーク

このインターフェースを通過する際にルーティングを行うものとする。

2. Packet Assembly Disassembly

PAD と略されるこの機能は、ホストから渡されるデータストリームを、実際の伝送路が受け付けられるサイズに区切り、伝送先アドレスシーケンス番号をふくむヘッダ情報を付け加えパケットストリームとしてホストへ受け渡すものである。

3. Sequencing

バイトストリームをパケッティングして、転送すると、パケットの順番が狂うことがある。これを防ぐのがsequencingで、パケッティングをする際に、各パケットにシーケンス番号をふり、受信側でそれをチェックすることにより受信パケットの順序の正当化を保証する事を目的とするものである。

4. Data Flow Control

これは効率よくデータを伝送するためにデータの流れを制御するもので、相手のアクノリッジを待たずに送出できるパケット数（これをウィンドウサイズという）をコネクションの開通時に決定し、以後、一括アクノリッジによるパケット交換を行うことにより伝送効率の向上がはかられる。固定ウィンドウサイズ、可変ウィンドウサイズの2つの方式があるが、このネットワークでは、固定ウィンドウサイズを用いた特殊なスライディングウィンドウアルゴリズムを用いる予定である。

5. Error Control

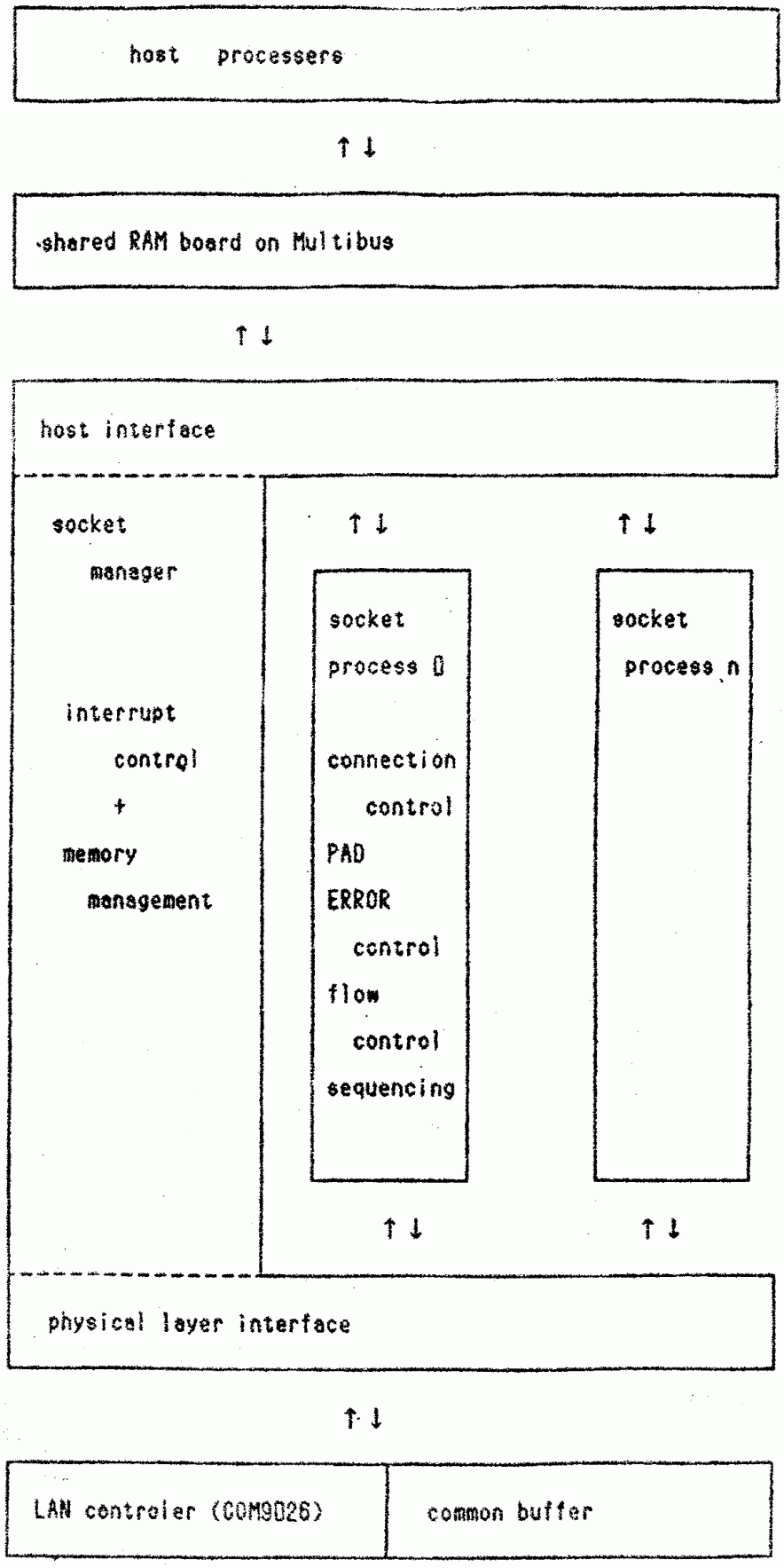
これは、データの伝送の途中に発生するエラーを検出しソースノードに対しエラーを報告し、パケットの再送を要求するなどして、誤りのないデータ伝送を保証するものである。

次に本ネットワークにおけるバーチャルサーキットの実現方法について、その概要　を述べる。

ノードプロセッサ上のソフトウェアは、OS9／68000上に構築される。これはOS9／68000のカーネルの機能を利用することで、新たにマルチタスクカーネルを用意する必要がなくなること、開発、デバッグの効率の向上等を考えるとOS9／68000の環境上で作業を行うことが有用である等の理由による。

ノードプロセッサ上でのソフトウエアは次頁の図のような構成になる。

ノードプロセッサ　ソフトウェア構成図



↑↓はその間でデータのやり取りないしは、プロセス間の通信が行われることを示す。また、点線部は本来独立しているべきであるが、効率の関係から、1つのモジュールとなっていることを示している。

この図からわかるように、ノードプロセッサ上のソフトウエアは次の4つに分かれる。

1. Host Interface
2. Phisical Layer Interface
3. Socket Manager
4. Socket Driver

それぞれの機能は、以下の通りである。

1. Host Interface
   マルチバス上の共有RAMを通じて、ホストとの間の通信を行う部分で、特定領域へのコマンドのセットおよびインタラプトの発行により通信を行う。
2. Physical Layer Interface
   ノードプロセッサ上のハードウェアを直接操作する部分
3. Socket Manager
   socket driver processの生成、共有RAM上のメモリの管理等を行う。またinterrupt handler routineもここに含まれている。
4. Socket driver
   socket managerによりforkされるプロセスでソッケト1つに対して1つのプロセスが対応する。PAD、Error Control, Multibus上の共有RAMを直接アクセスする場合がある。

以上の構成の基に、ノードプロセッサはホストに対し次に示すようなサービスを提供する。

1. socket no_reg
   ソケットナンバーの割り当ての要求
2. clear_socket_no
   ソケットナンバーの返還
3. connection_open
   2つのソケット間におけるコネクションの確立要求
4. connection_close
   確立されているコネクションの解放

5. send

データの送出

6. recieve

データの受信

7. abort

コネクションの中断

8. send_to

データグラムによる、パケットの送出

9. recieve_from

データグラムによるパケットの受信

§４．ユーザに提供されるネットワークライブラリについて

１）はじめに

ユーザプログラムに対してネットワークの機能をどの様な形で提供するかという点について、我々は２つの方針があると考えた。ひとつはＵＮＩＸ　４．２ＢＳＤの、クライアント←→サーバーモデルであり、いまひとつはＯＳ９のネットワーク（マイクロウエアが提供している）のデバイスオリエンテッドなモデルである。それぞれがどのようなものかを、簡単に説明しよう。

１．ＵＮＩＸ　ＢＳＤ　４．２　のアプローチ

ユーザに提供されるのは、ＯＳＩのモデルと良く似通っている。すなわち、ＯＳＩモデルのデータグラム、バーチャルサーキット（用語解説参照）を使うためのＣのライブラリを用意している。ユーザは、これらを利用して、リモートファイルトランスファー、リモートプリンタサーバ、リモートログインサーバ等のアプリケーションを作成する。とは言っても、直接にこれらのアプリケーションを作成するユーザは限られたシステムプログラマであり、一般のユーザは、これらのアプリケーションプログラムを利用するに留まるのが実状であろう。

２．ＯＳ９のアプローチ

ＯＳ９上では、ネットワークは１つの入出力デバイスとして扱うことができ、リモートなシステム上のパスを指定するには、次のようなパスリストを使う。

／ネットワーク名／ホストの名前／パスリスト

例えば、次のように使うことができる。

```
copy  /net1/Ares/h0/foo  /Ether/Tess/d0/user/kei/bar
```

これは、ｎｅｔ１というネットワーク上の、ホストネームＡｒｅｓの／ｈ０／ｆｏｏというファイルを、ネットワークＥｔｈｅｒ上のホストネームＴｅｓｓの／ｄ０／ｕｓｅｒ／ｋｅｉ／ｂａｒにコピーする。

また、

```
list  /d0/user/kei/bar  >/net1/Ares/p
```

は、／ｄ０／ｕｓｅｒ／ｋｅｉ／ｂａｒなるファイルを、ｎｅｔ１上のホストＡｒｅｓに接続されているプリンタ（デバイス名ｐ）に出力（リダイレクト）する。これによりリモートなプリンタへのリスティングを行うことができる。

このように、ＯＳ９はファイルやデバイスを、リモートなものかどうかにかかわらずトランスペアレントに利用できる。これが我々がＯＳ９のネットワークをデバイス指向と呼ぶ理由である。

それぞれのアプローチの概要は以上の通りだが、結論を先に言えば、我々はＢＳＤ４．２のアプローチをとった。ＯＳ９のアプローチには汎用性において問題が認められたためである。次のような例を考えてみて欲しい。

リモートな、複数ホストからのリストアウト要求にも応えることのできるプリンタサーバを設計したいとする。この場合、ＢＳＤのアプローチに従えば、バーチュアルサーキットをつかってこともなげに記述する事ができるが、ＯＳ９のアプローチではどうすればよいというのだろうか？恐らくプリンタサーバ用の仮想的なデバイスを作成し、それらへリモート・リダイレクトすることによって実現するしかないだろう。こうすると、新たなサーバ（ログインサーバ、等々）を作る度に、新たなデバイスドライバを設計しなければならない。しかしながらＯＳ９のデバイスドライバは、その詳細な情報が提供されておらず、まったく新しいデバイスドライバの作製は困難を極める。

相互の得失を見きわめた上で、我々はＢＳＤのアプローチをとることに決定した。

2）ネットワーク機能ライブラリの概要

前節で述べたように我々はＢＳＤ４．２と同等のアプローチをとることにしたが、実際にネットワークの機能を利用する際に、どのようなライブラリ関数をどのように使って処理を進めるのかを、具体例を示しながら説明したい。ここでは、プリンタサーバとそのクライアント（リモートホストのサーバと通信し、リストアウトを要求するプロセス）を例にとって説明する。

まず、クライアントはリモートプリントサービスを行っているサーバプロセスのネットワークアドレス（以降ＮＡＤ）を知らなければならない。このために、gethostbyname()という関数が用意されている。この関数は各ホスト中にローカルに存在するサービスネームテーブルを検索して、そのサービスを提供しているサーバプロセスのＮＡＤを返す。このサービスネームテーブルはサーバホストのダウンや、起動（電源ＯＮ等）に応じて、ダイナミックにアップデートされる。

次に、得られたＮＡＤへ対してcon_req()を使ってコネクションの確立を要求する。

サーバプロセスは、accept()を呼び出すことによって、どこかからのコネクション要求を待っているが、要求が到着するとその要求を行ったプロセスとの間にコネクションを確立しそのコネクションに対してサービスするプロセスを起動する。以降、そのプロセスがクライアントから送られてくるリストアウトファイルをスプーラー用のディレクトリにコピーする。

クライアントは正常にコネクションが確立されたら、そのコネクションを通じてリストアウトの方法（高密度印字やレタークオリティ、希望するプリンタの種類等々）、ファイル名、ファイルの本体等をsend()を使って送り、最後にコネクションをcon_close()を使って終結させる。

プリントサービスの概要は上のようになるが、上に現れたライブラリ関数以外にも多くの関数が存在する。例えば例中にも示された、サービスネームテーブルのアップデートには、一斉同報機能を利用するのが妥当であるが、このためにはバーチュアルサーキットではなく、データグラムをサポートする関数が、ライブラリ中に存在する。

§５．ネットワーク用アプリケーションについて

ネットワークによって結合されたシステムに対して、ユーザはどのような機能を要求するだろうか。我々がネットワークを構築する契機となったのは、リモートなファイル転送や、プリンタサーバの必要性であった。これらの機能を提供するためのネットワークユーティリティは早急に準備されなければならない。

また、ＢＳＤ４．２のアプローチをとることによって失われた、デバイスへのトランスペアレンシーを取り戻すための、ネットワーク用ｓｈｅｌｌ（ｎｓｈ）を作製する事も必要であろう。ここでは、我々のネットワークの将来像を示す意味でも、このｎｓｈが、どのようなものになるのかを考えてみたい。

１．　コマンドラインによって与えられたパスリストを解析し、それがリモートなホスト上へのパスであった場合には、然るべきホストのサーバと通信を行って、コマンドを処理する。例えば、

```
list   /net/Ares/h0/foo   >/net/Tess/d0/bar
```

というコマンドラインが投入された場合、ホストネームＡｒｅｓ上のファイルサーバに対して／ｈ０／ｆｏｏなるファイルの転送を要求し、それをｌｉｓｔコマンドへ渡してやる。また、リモートホストへのリダイレクトが指定されているので、ｎｓｈは、ｌｉｓｔをｆｏｒｋする前に、ｓｔｄｏｕｔをテンポラリファイルにリダイレクトしておき、ｌｉｓｔプロセスの終了後、テンポラリファイルを、Ｔｅｓｓ上のファイルサーバと通信を行いながら／ｄ０／ｂａｒへとコピーする。

２．　ネットワーク全体でのｃｐｕに負荷分散を行う。ネットワーク上に複数のｃｐｕ（ホスト）が存在するのだから、そのうちのロードアベレージの低いホスト上でコマンドの実行を行う事が考えられる。とは言っても、ｉ／ｏバウンドのジョブを他ホストで実行しても意味はないため、そのコマンドの性格を、ｃｐｕ／ｉ／ｏレシオによって把握する必要ががある。このための、コマンドごとのデータベースが必要となるだろう。または、リモート実行については、ユーザ（コマンドラインの投入者）が指定を行うという方針も考えられる。例えば次の様になる。

```
.cc foo.c -f=/net/Ares/h0/cmds
```

プログラム名の始めにピリオドを前置した場合に限り、リモートなホスト上（最も軽負荷なホスト）で実行されるというものである。また、各ホストの負荷状態を知るための、ダイナミックに更新されうるデータベースを用意する必要がある。この更新のためには、データグラムプロトコルの一斉同報機能を利用するのが適切であろう。

＊＊＊＊　　グラフィックボード　"まつだいら"　　＊＊＊＊

グラフィックボード「まつだいら」は、どこまで部品を削り、どこまで簡単な回路で実用となるものが出来るのかを追及して設計されています。

苦労の結果、どのくらいの部品数にまとまったのかをわかりやすく言うと、ＩＣが３０数個になりました。ＰＡＬは２個使ってあり、もっとＰＡＬを使えば更に部品数を減らすことが可能なのですが、部室のあちこちに散らばっているＴＴＬ等を使ってより安価に仕上げようとした結果、前期した程度の部品数にとどまってしまいました。ちなみに、基盤やＩＣソケット等も含めて、必要な部品のほとんどが、あまり物を利用していて、このプロジェクトのためにあらためて買ったのは、ラッピングワイヤー（配線材）等数点のみです。中心となるＣＲＴＣ（ＣＲＴコントローラ）はＨＤ４６５０５Ｓで解像度は１０２４×５１２ｄｏｔです。（実際に表示されるのは７２０×５１２。）表示色は今のところ単色ですが、ＲＡＭの増設等の改良により、表示色を増やすことが出来ます。

システムの形状ですが、クラブのコンピュータからアクセス出来るようにマルチバス規格のボードに作られています。（部品の数を削ったおかげで、ボードの約半分強に全回路が収まっています。カードの残り半分はＲＡＭの増設等の増設回路のためにとってあります。）出力はＰＣ－ＫＤ６５１というモニタを使っています。あのＰＣ－１００用のモニタといえば、お分かりの方も多いでしょう。

****　ターミナル　****

仕様）CPU：MC68008（8MHz）

RAM：256k　StaticRAM×4＝128kbyte

ROM：256k　EP-ROM　×2＝　64kbyte

周辺LSI

タイマー　：MC68230（PI／T）

シリアルインターフェース：MC68681（DUART）

このターミナルはホストの負担を軽減すべく設計されたものです。今MMAで使われているOSはOS9／68000で、外部記憶がなくても動いてくれるので、これをターミナルにのせて、インテリジェント端末にする予定です。上記の仕様はすべてその布石です。CPUの68008は68000MPUのDATABUS8bitのバージョンでありMC68230のタイマーはOS9／68000を走らせるのにぜひとも必要なものです。64kbyteのROMはOS9／68000を焼くため、また128kbyteもRAMがあればかなりの処理をホストにやらすことなくターミナルに行わせる事が可能です。

その他の特長として、モニターへの出力は、アトリビュートで反転、点滅、アッパーバー、アンダーバー、文字フォント2種類の切り換え、8階調の輝度などがあります。

**** Ｓｅｒａｐｉｓ ****

『１８ボイス＝ミュージックシンセサイザ』

１・ＦＭ方式の考え方

音源部はキャリア（発振部）とモジュレータ（変調部）より構成されます。キャリア、モジュレータ共に１個の場合も複数の場合もあり、キャリアに対しさらに上位のキャリアが付くこともあります。

一番簡単なキャリア、モジュレータ共に１個の場合を例にとって説明します。キャリアは、本来、基本周波数を発振しますが、モジュレータの出力を受取れるようになっていて、受取った出力値によって発振周波数を基本周波数よりずらすことができます。もしモジュレータの出力も一種の波であればキャリアの発振周波数は基本周波数を中心として上下します。ここで、モジュレータの発振周波数が、キャリアの基本周波数にごく近いかあるいはその倍数であったとすれば、キャリアの発振する１つの波の各部分部分で、周波数が異なってきます（こうなってはもう厳密な意味での周波数とは呼べませんが）。つまり、キャリアの発振する波の形は元となるサインカーブから大きく外れ、その独特な波形が人間の耳に音色として聞こえるようになります。我々のミュージックシンセサイザではこの方式により発声しています。

２・Ｓｅｒａｐｉｓの構成

ＳｅｒａｐｉｓはヤマハのＦＭ音源ＬＳＩ・ＹＭ２２０３（ＯＰＮ）を６個持っており計１８音を同時発声させることができます。

マルチバスを用いてコンピュータとの通信が可能であり、またＭＩＤＩインターフェイスを使って、ＭＩＤＩ規格のシーケンサやシンセサイザによる制御も可能です。

演奏用ソフトはＭＩＤＩ風コマンドで、１つのＬＳＩを１つのチャンネル（つまりシンセサイザ）にみたて、Ｓｅｒａｐｉｓ全体を６台のシンセサイザの集合として取扱っています。

MMA

編集人：Ｂｅｎｊａｍｉｎ　（１Ｉ）
　　ＤＹＥ　　（１Ｉ）
　　ｏｉｌ　　（１Ｄ）
発行所：ＭＭＡ（Microcomputer Making Association）
　　〒１８２　調布市　調布ヶ丘　１－５－１
　　電気通信大学　サークル会館２Ｆ
発行日：１９８７年１１月２１日