マルチメディア通信と分散処理 29－1
（１９８６．５．１５）

研究用ローカル・エリア・ネットワーク：ＬＩＰＳ－Ｎｅｔ

中井 秀司　稲井 寛　横平 徳美　西田 竹志　宮原 秀夫　高島 堅助

大阪大学基礎工学部情報工学科

当研究室で開発されたLAN (LIPS-Net : Laboratory Information Processing Service-Network)において，通信プロトコルは物理層，データリンク層，トランスポート層，およびセッション層の４階層から構成されている．データリンク層以下の層は NIF (Network Interface Facility)と呼ばれるネットワークインターフェース内で，それより上位の層はホストコンピュータ内で実現されている．本論文では，LIPS-Netのハードウェア構成について述べ，その上で実現された各階層の機能と実現方法，および各階層間の制御について説明を行う．また，LIPS-Netの通信機能を用いて作成された対話型メッセージ転送プログラムの機能および通信手順について述べている．

Shuji NAKAI　Hiroshi INAI Tokumi YOKOHIRA Takeshi NISHIDA　Hideo MIYAHARA　Kensuke TAKASHIMA

Department of Information and Computer Sciences Faculty of Engineering Sciences, Osaka University

"LIPS-Net:Information Processing Service Network in Laboratories"(in Japanese)
by Shuji NAKAI (Dept. of Information and Computer Sciences, Faculty of Engineering Science, Osaka Univ. Toyonaka, 560,JAPAN) et al.

Communication protocol for LIPS-Net (Laboratory Information Processing Service-Network) developed in our laboratory is composed of four layers, i.e. physical, data-link, transport and session. The lower layers (physical, data-link) are implemented on NIF (Network Interface Facility) which has been developed in our laboratory to connect between communication medium and host computer, and the upper layers (transport, session) is implemented on host computer.

In this paper we describe the hardware design of LIPS-Net, especially NIF, and describe how we implement this protocol. And, the application program developed with this protocol is also described.

研究用ローカル・エリア・ネットワーク：ＬＩＰＳ－Ｎｅｔ

LIPS-Net:Information Processing Service Network in Laboratories

中井 秀司 Shuji NAKAI　稲井 寛 Hiroshi INAI　横平 徳美 Tokumi YOKOHIRA　西田 竹志 Takeshi NISHIDA　宮原 秀夫 Hideo MIYAHARA　高島 堅助 Kensuke TAKASHIMA

大阪大学基礎工学部情報工学科

Department of Information and Computer Sciences,
Faculty of Engineering Sciences, Osaka University

１．まえがき

LAN(Local Area Network)を構築するためには，計算機とネットワークを接続するネットワークインターフェースと，計算機相互間の通信プロトコルの設計・開発が必要となる．しかし，市販されている一般のLANは，接続できる機種が限定されている場合が多く，また，LANを構築する場合，標準化されているプロトコル[1]は複雑過ぎる等の問題がある．

そこで，当研究室において，ホストコンピュータの機種によらず，研究室内での情報交換に必要である最小限の機能を提供するプロトコルを持ったLAN(LIPS-Net:Laboratory Information Processing Service-Network)の開発を行ってきた．LIPS-Netにおいては，通信に必要な機能が関数として提供されているので，ユーザは複雑な通信手順を知ること無しに，アプリケーションプログラムを作成することができる．

本論文では，次章で，LIPS-Netのハードウェア構成，特に，当研究室で開発されたネットワークインタフェースであるNIFについて述べる．第３章，第４章では，LIPS-Netのプロトコルにおける各階層の機能と構成，およびその階層間の制御について説明を行う．また，第５章では，アプリケーションプログラムの一つである対話型メッセージ転送プログラムの機能および通信手順について述べる．

２．LIPS-Netの概要

構成

LIPS-Netはバス型ネットワークであり，図1に示すようなハードウェア構成で実現している．

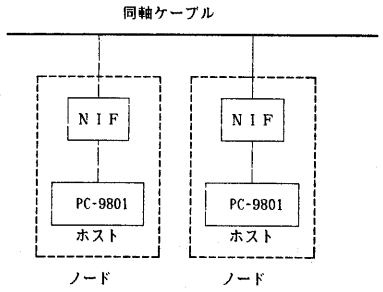

図１　LIPS-Netのハードウェア構成

各構成要素を以下に示す．

- 伝送媒体
  同軸ケーブル(75Ω)
- NIF(Network Interface Facility)
  当研究室で開発したネットワークインターフェース[2]
- ホスト
  PC-9801

次に，NIFについて簡単に述べる．

NIFは，図2のように，T(Tranceiver)部,TC(Tranceiver Controller)部,CM(Common Memory)部，インターフェース

部で構成される．

・T部
　プログラムで実現された種々のアクセス方式を実行する．
・TC部
　T部の制御及びホストとの通信を行う．
・CM部
　送受信されるデータが一時的に格納されるバッファメモリ．
・インターフェース部
　NIFとホストとの間でデータをやりとりする時に用いるレジスタ群．

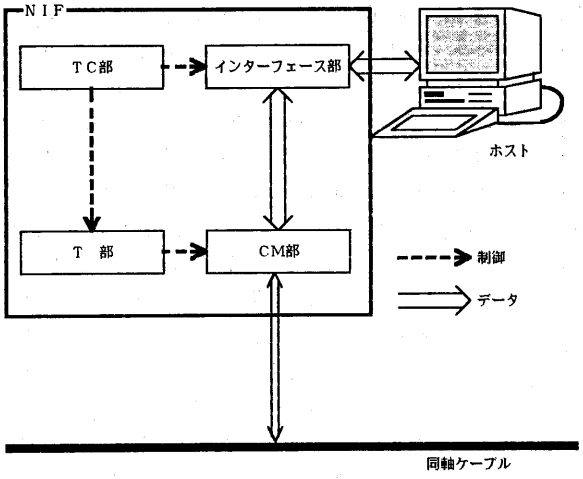

図2．NIFの構成

ホストからのデータはTC部の制御のもとに，インターフェース部を介して，CM部に格納される．格納されたデータはT部の制御のもとに同軸ケーブル上に送出される．

同軸ケーブル上のデータは，逆の手順でホストに転送される．

3．LIPS-Netのプロトコル階層

LIPS-Netのプロトコルは図3に示すように階層的に構成され，データリンク層以下をNIFで，それより上位の層をホストで実現している．また，隣接階層間のインターフェースのために，図3に示したファンクション，リターン，コマンド，レスポンス，レクエスト，及びリプライが用いられる．

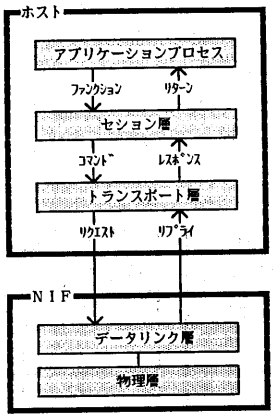

図3．LIPS-Netのプロトコル階層

3．1　NIFで実現される層

3．1．1　物理層

物理層は物理チャネルを提供する層であり，ビット転送の制御，データのエンコード／デコード，キャリア検知，衝突検出等を行う．

3．1．2　データリンク層

データリンク層はノード間にリンクを設定し，データをその意味を変える事なく送受信するための機能を持つ．物理チャネルへのアクセス方式として，CSMA/CD（Carrier Sense Multiple Accese with Collision Detection）を用いて，誤り検出，再送による回復等を行う．

LIPS-Netでは，このCSMA/CD方式をNIF中のT部で，プログラムにより実現している．

3．2　ホスト内で実現される層

アプリケーションプロセス，セショ

ン層，トランスポート層の論理リンクに対する関係を図4に示す．論理リンクとは，通信を行うノード間に設定される仮想的なリンクのことである．

トランスポート層では論理リンクの終端であるポートの状態を管理することによりマルチリンクを実現する．また，各論理リンクは，そのノード内で一意的な識別子であるポートIDが割り当てられており，ポートIDとアプリケーションプロセスとの対応付けはセション層が行う．

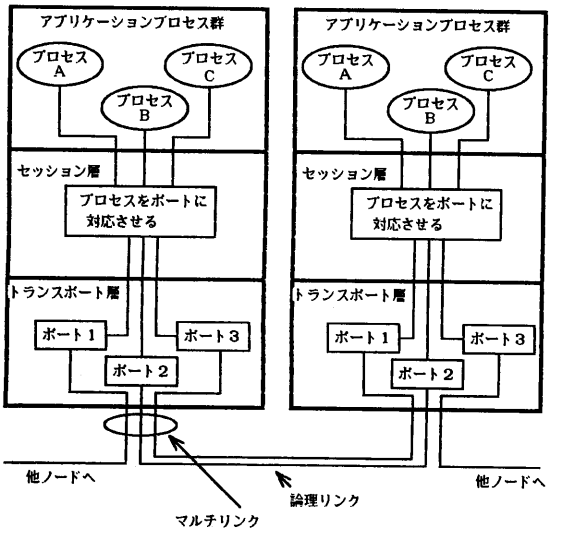

図4． 論理リンクとアプリケーションプロセスの関係

3．2．1 トランスポート層

トランスポート層の機能は以下の通りである．

- 論理リンクの設定と解放
- データの分割
- フロー制御
- 誤り回復

次に，これらの機能について述べる．

論理リンクの設定と解放

各ノード内の任意のアプリケーションプロセス間で，論理リンク設定要求や解放要求によって，論理リンクが動的に設定，解放される．各プロセスは，同時に複数の論理リンクを張ることが可能である．また，論理リンクの解放は，そのリンクに関与するいずれのノードからも行うことができる．

論理リンクの設定は，図5に示すような形態で行われる．まずノードAではセション層からの論理リンクの接続要求に対して，ポートテーブルより未使用のポートID（図ではA3）を捜し，それを付加してコネクト要求メッセージ（以下，相互のトランスポート層間で，通信の制御のために用いられる情報の単位をメッセージと呼ぶ）として相手ノード（ノードB）に送出する．ノードBのトランスポート層はその要求をセション層に伝え，セション層から受諾の返事があればポートテーブルにノードBでの未使用ポートIDと送信されてきた相手ポートID（図ではB5とA3）とをポートテーブルに登録する．そして，B5を付加してコネクト受諾メッセージとしてノードAに送信する．ノードAのトランスポート層では，これを受け取るとやはり同じようにポートテーブルにA3とB5の項目に追加してセション層にノードBからの接続受諾の返事を伝える．

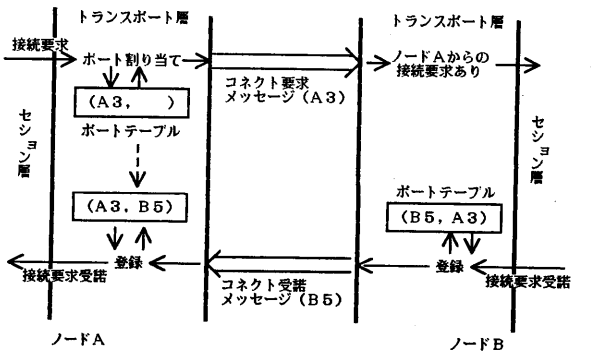

図5．論理リンクの設定

データの分割

NIF中のバッファは，伝送効率等を考慮して，ある一定の大きさ（以下，ブロックと呼ぶ）に分割されて，トランスポート層に管理される．もし，セション層からのデータ長がブロック長より大きい時には，ブロック長に等しい大きさで分割される．この分割されたデータをデータブロックと呼ぶ．データブロックはシーケンシャルな番号を付けて管理される．

フロー制御

バッファのオーバーフローを防ぐため，データブロック単位のフロー制御を行う．

データ送信側からの送信要求に応じて，データ受信側は，データ送信側にフロー制御メッセージを送る．フロー制御メッセージを用いてデータ送信側に通知される情報は，バッファ内に確保されたブロック数である．

データ送信側は，データ受信側から通知してきたブロック数までしかデータブロックを送出しない．

誤り回復

データ受信側・送信側は，データブロックの誤り回復をそれぞれ次のように行う．

・データ受信側

受信側はデータブロックを受信するとそのデータブロック番号をチェックする．この番号が期待するブロック番号と一致したときは，データブロックをセション層で指定された領域に転送する．以前に正常受信したものと同じデータブロックを受信した場合は，そのデータブロックを無視する．データブロック番号が期待された番号より大きい場合は，そのデータブロックを無視し，最後に正常受信したデータブロックの番号を伴ったNACKメッセージを送出する．また，あるデータブロックを受信してから一定期間を経過しても相手側より，次のデータブロックが送られて来ない場合にもNACKメッセージを送出する．

・データ送信側

データ送信側は，NACKメッセージが送られてこないかぎり，送り出したデータブロックが正しく受信側に届いたものと解釈する．NACKメッセージを受信したときは，NACKメッセージに付けられているブロック番号の次のデータブロックから再送する．

図6にデータブロック転送の一例を示す．

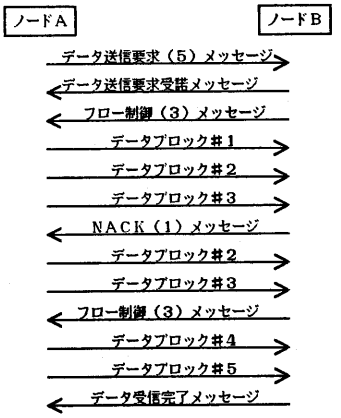

図6．　データブロックの転送

ノードAのトランスポート層は，セション層からノードBへの送信要求を受けると，そのデータを幾つかのデータブロックに分割する．ここでは，送信するデータを5つのデータブロックに分割したとする．データの分割が終わると，ノードBへ送信データブロック数を伴ったデータ送信要求メッセージを送信する．

ノードBのトランスポート層は，ノードAから5データブロック分のデータ送信要求を受けると，セション層にそのことをレスポンスを用いて伝える．セション層が上記要求を受諾すると，トランスポートでは，受信バッファを確保すると共に，ノードAへデータ送信受諾メッセージを送信する．図では，3ブロック分のバッファを確保したので，それを通知するフロー制御(3)メッセージを送出する．ノードAは，そのメッセージを受信すると，最初の3データブロクをノードBへ送信する．

いま、データブロック#2が送信途中で壊れてノードBに届かなかったとする．ノードBでは、データブロック#1の次のデータブロック#3を受信すると、最後に正常に受信したデータブロック番号を伴ったNACK(1)メッセージを送出する．それに伴いノードAは、データブロック#2から再送する．

ノードBでは，データブロック#2,#3を受信すると，バッファを使い切ったので，新たにバッファ確保する．いま，バッファが2ブロック分確保できたとすると，それを通知するフロー制御(2)メッセージを出す．それに伴い，ノードAはデータブロック#4から2ブロック分のデータブロックを送出する．

ノードBでは，データブロック#4,#5を受信すると，データ受信の完了をセション知らせる共に，ノードAへデータ受信完了メッセージを送出して，次の処理に移る．

ノードAでは，データ受信完了メッセージを受信すると，セション層にデータ送信の完了を知らせ，次の処理に移る．

ソフトウェア構成

今述べたような機能は図7に示すソフトウェア構成で実現される．次に，各構成要素を説明する．

◎リプライキュー
データリンク層からのリプライが入れられるFIFO(First-In First-Out)待ち行列．

◎レスポンスキュー
セション層へのレスポンスを入れるFIFO待ち行列．

◎ポート管理モジュール
各ポートに関する情報が格納されているポートテーブルの管理を行う．

◎送信データブロック管理モジュール
・セション層から送信要求のあったデータをデータブロックに分割し，ブロック番号を付加する．
・送信データブロックテーブルに基づいて，次に送出すべきデータブロックを決定する．

◎受信データブロック管理モジュール
・受信データブロックのブロック番号の管理を行う．
・受信データブロックテーブルを基にして次に送られてくるデータブロックを主記憶のどこに格納するのかということを決定する．

◎受信バッファ管理モジュール
受信バッファテーブルを用いて，NIFの受信バッファをブロック単位で管理する．

◎データ送出モジュール
データブロック，各種メッセージに必要なヘッダを付け，NIFに送出する．

◎コマンド処理ルーチン群
コマンドごとに用意されたコマンド処理ルーチンの集まりである．セシ

ョン層から関数呼び出しで起動される．

◎リプライ解析モジュール
データリンク層から送られてきたリプライをリプライキューから受け取り，そのリプライの内容を解析し，そのときのポートの状態に応じた処理を行う．

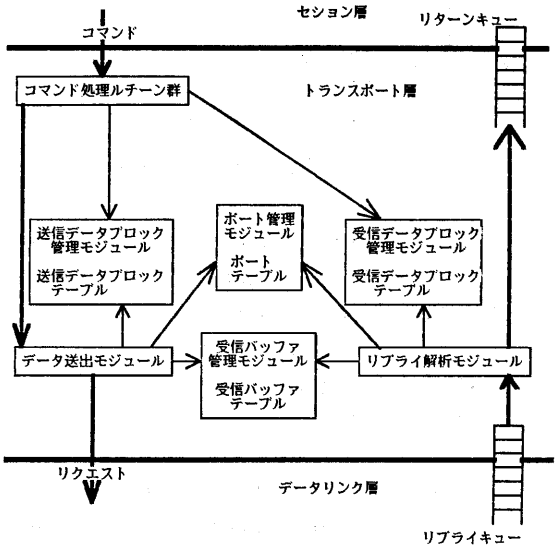

図7．トランスポート層のソフトウェア構成

３．２．２　セション層

セション層は，アプリケーションプロセスとポートを結び付ける層である．また，他ノードのアプリケーションプロセスから通信要求を受信した場合，そのプロセスに対応する受信側のアプリケーションプロセスを起動する．この層は，ユーザに対して，ファンクションと呼ばれるサービスを関数として提供しており，ユーザはこれを用いてアプリケーションプログラムを作成することができる．

ソフトウェア構成

今述べたような機能を実現するソフトウェア構成を図8に示す．次に，各構成要素を説明する．

◎レスポンスキュー
トランスポート層からのレスポンスが入れられるFIFO待ち行列．セション層のソフトウェア構成で述べたキューと同一のキュー．

◎リターンキュー
アプリケーションプロセスへのリターンを入れるFIFO待ち行列．

◎ファンクション群
アプリケーションプロセスから直接呼び出されるファンクションの集まりである．各ファンクションはアプリケーションプログラムを作るために必要な機能を持っており，それぞれのファンクションはコマンド呼び出しモジュールを通してトランスポート層中のコマンドを呼び出す．

◎コマンド呼び出しモジュール
このモジュールは，セションコネクションが張られている間，各アプリケーションプロセスに設定されているプロセス番号と，トランスポート層中のポート番号を対応づけて記憶している．ファンクションの引数として渡されるプロセス番号から，対応するポート番号を求め，そのポート番号を引数としてトランスポート層のコマンドを呼び出す．

◎レスポンス解析モジュール
トランスポート層からのレスポンスを受け取り，そのレスポンスをアプリケーションプロセスにリターンキューを通して伝える．

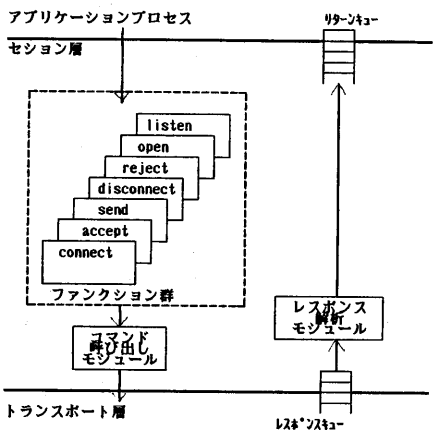

図８．　セション層のソフトウェア構成

４．階層間の制御

各階層は図9に示したように，それぞれ独立に作られている．データリンク層とトランスポート層間，つまりNIFとホスト間のインターフェースは割り込みを使い，I/Oポートを通して行われる．

ホスト内の各階層間のインターフェースは下位層のサービスを利用するときは関数呼び出しで，上位層に応答を返すときは待ち行列で実現される．

このように独立に作られたトランスポート層，セション層，アプリケーションプロセスを制御するために，階層間応答管理モジュールがある．このモジュールは，同時には一つのプロセスしか走らない環境下に各プロトコル階層の動きを制御するためのものである．各階層間の応答は一旦それぞれの待ち行列に入れられ，処理を待つ．このモジュールは，応答待ち行列を監視し，各待ち行列に応答が入るとそれを受け取るべき階層を起動するという機能を持っている．

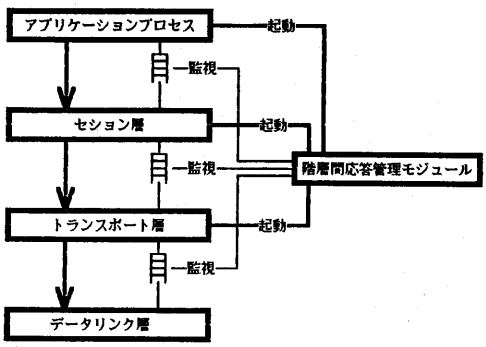

図9．　階層間の制御

５．アプリケーションプログラム

今述べたセション層のファンクションを用いて，下記に示すアプリケーションプログラムが開発されている．

・ファイル転送プログラム
・電子メール転送プログラム
・対話型メッセージ転送プログラム

以下に，これらのアプリケーションプログラムの内，対話型メッセージ転送プログラムの通信手順について説明する．

対話型メッセージ転送プログラムの通信手順

対話型メッセージ転送プログラムは，相手局と対話型で通信を行うプログラムであり，起動側プログラムと応答側プログラムから成る．ここで起動側プログラムとは，最初にコネクションを要求するプログラムであり，応答側プログラムとは，その要求に対して応答するプログラムである．

対話型メッセージ転送プログラムの通信手順を図10に示す．この手順を簡単に説明する．

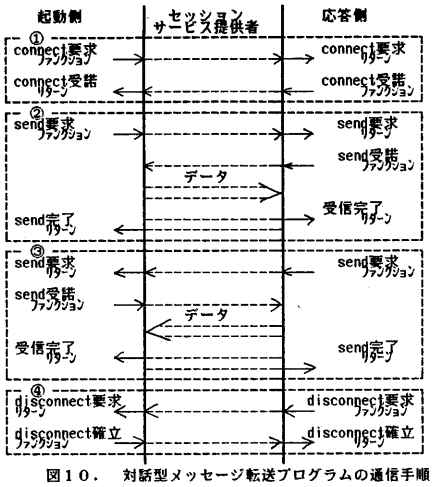

図１０．　対話型メッセージ転送プログラムの通信手順

①コネクションの設定

起動側からコネクト要求ファンクションを送出する．その要求に対して応答側からコネクト受諾ファンクションが返されるとコネクションが設定される．

②データ送信

起動側から送信要求ファンクションを送出する．その要求に対して応答側から送信受諾ファンクションが返されると，起動側から応答側へデータが送出される．データの送信が終了すると，起動側・応答側に対して，それぞれ送信完了リターン，受信完了リターンが返される．

③データ受信

②とは逆に起動側が応答側からのテキストデータを受信する．

どちらの局も通信を続ける場合は②③を繰り返す．

④コネクションの解放

通信をやめる場合，どちらかの局からコネクション解放のためにディスコネクト要求ファンクションを送出する．相手側でそれを受け取ってディスコネクト確立ファンクションを返すとコネクションの解放が行われる．

６．あとがき

研究室内で開発したLIPS-Netにおける通信プロトコル階層の機能，および実現方法を述べた．LIPS-Netのプロトコルは，通信に必要な種々の機能を関数という形で提供している．従って，ユーザは，複雑な通信機能を知ることなく，それらの関数を呼び出すだけで容易にアプリケーションプログラムを作成することができる．

これらの関数を用いて，アプリケーションプログラムとして，ファイル転送プログラム，電子メール転送プログラム，対話型メッセージ転送プログラム等を作成した．

今後の課題として，様々な要求に応じたアプリケーションプログラムの充実が望まれる．

参考文献

[1]鈴木，東田
”ネットワークアーキテクチャ(開放型システム間相互接続)の標準化動向”，
情報処理，Vol. 26, No. 4, pp. 290-298 (1985)

[2]松本，三木，西田，宮原，高島
”スターカップラを用いた構内網におけるネットワークインターフェースの設計”，
信学会技報，CS82-19(1982-5)