実験的コンピュータネットワーク（PHENICS-I）について

An Interlaboratory Computer Network Using Deadlock-Free Local Controller —— PHENICS-I

吉田典可　菊野　亨　角田良明　若林真一
Noriyoshi YOSHIDA, Tohru KIKUNO, Yoshiaki KAKUDA and Shin'ichi WAKABAYASHI
広島大学工学部
Faculty of Engineering, Hiroshima University

1. まえがき

コンピュータネットワークについては，これまでいくつかの試みがなされてきている．これらは当初，資源の共用を目的として開発されていたが，近年，システムとしての経済性，信頼性，応答性，拡張性などにも多くの改善が行われてきている．コンピュータネットワークのうち，通信回線として公衆網を利用することなく独自の通信方式を採用し，工場，大学あるいは研究所などの敷地内に点在するコンピュータや端末をその通信方式で結合したものを，通常，ローカルコンピュータネットワークとよぶ[1,4]．筆者らは先にローカルネットワーク上で信頼性，及び応答性を高めるための技術について開発して[10,11]研究室内ネットワークの実験システムPLANET[16]を試作した．今回，これとは独立に，大学内コンピュータネットワークを目指した実験的コンピュータネットワーク（PHENICS-I）を，通信方式やインタフェース装置の設計・製作を含めて開発に着手したのでここに報告する．

PHENICS-Iでは次に示す3つの機能の提供を，安価に実現することを目的とする．

（1）ハードウェア資源及びソフトウェア資源の共用
各研究室が独立に保持しているコンピュータ，各種端末装置（例えばグラフィクディスプレイ，X-Yプロッタ，ワードプロセッサなど）や，プログラム（例えばコンパイラ，応用プログラムなど）をネットワークを介して利用する．

（2）データベース資源の共用
いわゆる分散形データベースシステムをネットワーク上に実現する．すなわち，各研究室が保管している固有のデータベース（主として文献情報に関するもの）を，必要ならばそのコピーを含めて，ネットワーク内で保管し，必要な情報を研究室間で相互利用する．

（3）電子郵便
研究室相互で（例えばセミナーに関する行事予定連絡についての）メッセージ交換をネットワーク上の同報通信で行う．

更に，上述の（1）を利用することにより，次の2つの機能も提供される．

（4）負荷の分担
ある研究室においてコンピュータの負荷が100％を超える場合，残りのジョブの処理をネットワーク内の他の（研究室が所有する）コンピュータに依頼することが可能である．

（5）信頼性の向上
ある研究室において故障のためコンピュータが使用不能な場合でも，ネットワーク内の他のコンピュータを利用して研究を継続することが可能である．

本ネットワークにはマイクロコンピュータ，ミニコンピュータ及び端末の計6台を接続予定である．なお，55年度中は時間の都合上，そのうち3台だけの接続を計画している．

ネットワークに接続されているコンピュータ

及び端末をホストとよぶ．PHENICS-I ではホスト間でパケット交換方式のデータ伝送を採用している．その理由は，通信制御装置が安価に実現できること，通信手順や速度の異なる端末との通信が容易に行えること，などである．

ハードウェアの面では，通信制御装置に時間管理の機能（タイマ）を組込むことにより，(1) ホスト間のシリアルデータ伝送が効率よく行える，(2) ホスト，通信制御装置，あるいは通信網における故障の検出が比較的容易に可能である，など経済性，信頼性の高いものとなっている．

ソフトウェアの面では，フロー制御とルーティングが同時に行えるプロトコルを開発した．これにより (1) ネットワーク上でのパケットの伝送遅延時間が短縮される，(2) ネットワークのスループットが向上する，などユーザにとって応答性の良いものとなっている．更に，ネットワーク管理を行うローカルコントローラはデッドロックフリーであることが示されており，ネットワーク上に接続されているホスト間のパケット交換が保証される．

以下，2. でPHENICS-I の概要としてハードウェア構成図，プロトコルの簡単な説明を与えたあと，3. では通信制御装置（CCU），4. ではCCUにおける制御プログラムについてその詳細を述べる．

2. PHENICS-I の概要

現在，開発中のコンピュータネットワーク PHENICS-I（Prototype of Hiroshima Univ. Electronic Circuits and Systems Lab. Network-I）は，大学（の各研究室）内に点在するコンピュータ，及びインテリジェント端末を有機的に結合することを目指す実験的ローカルコンピュータネットワークである．

2.1 物理構成（ハードウェア）

実験的ローカルコンピュータネットワーク PHENICS-I に接続予定のマイクロコンピュータ，ミニコンピュータ，及び端末は次の6台で，図1にその構成図を示す（図1の実線部分が既設及び今年度実施分，点線部分は次年度分を表す）．

① Nippon Data General : NOVA3/D
② Panafacom : L-16A
③ Logic Systems International : LOGIC-80
④ Sord : 機種未定
⑤ 研究室で開発中のインテリジェント端末
⑥ Super Brain : MCZ-80

図1でCCUは今回，マイクロプロセッサ（Z-80）を使用して開発した通信制御装置（Communication Control Unit）を表す．その詳細については3. で述べる†

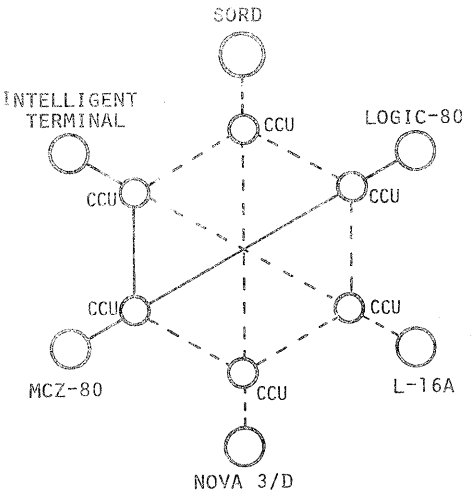


図1　ネットワーク構成図

2.2 論理構成（プロトコル）

ネットワークの基本構成要素はHOSTノード，CCUノード，リンク，プロセスである．コンピュータ相互間の通信に関心があるので，以降では特に断らない限りノードはHOSTノードとCCUノードの対を意味し，リンクは通信回線を，プロセスはアプリケーションプログラムをそれぞれ表すものとする．

PHENICS-I におけるネットワークアーキ

† 次年度計画においては，今回よりも更に機能を向上させたCCUを製作し，接続する予定であり，それについては別の機会に報告する予定である．

テクチャの概要を図2に示す.

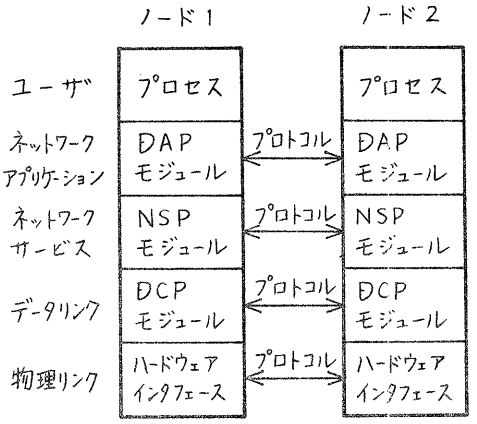


図2　論理構造（プロトコル）

① 物理リンク層：ハードウェアインタフェースに関するプロトコルが規定される.
② データリンク層：隣接するCCUノード間のデータ伝送制御を行うため，データ通信メッセージプロトコル（DCP）が規定される.
③ ネットワークサービス層：プロセス間の通信を管理する層で，論理的な通信路の設定・解放，データフロー制御，ルーティングなどに関するネットワークサービスプロトコル（NSP）が規定される.
④ ネットワークアプリケーション層：他のノードの入出力装置やファイルに対する制御を行う層で，ファイル伝送，ファイルアクセスなどに関するデータアクセスプロトコル（DAP）が規定される.
⑤ ユーザ層：プロセスが存在する層である.

このアーキテクチャはDECのDNA（digital Network Architecture）[6][05]と同様，互いに独立に管理されているコンピュータ間を結ぶ通信に重点を置いたものとなっている．従って，1. でも述べたように，各ノードが独自に所有する種々の資源を互いに共用しあい，コンピュータネットワーク全体の処理能力の向上を図ることを目指すものである.

［注1］　ここで提案するネットワークアーキテクチャとDNAの違いは，DNAではNSPモジュールが更に2つに分かれていてデータフロー制御とルーティングがそれぞれ異なる層に属することである.

2.3 基本ソフトウェア

ネットワークを効率良く運用するためのソフトウェアは図3に示す5つのブロックから構成される.

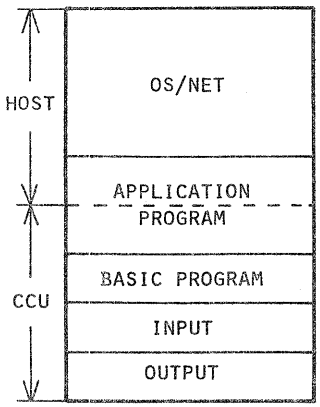


図3　ソフトウェアの構成

① OS/NET：ユーザあるいは周辺機器とPHENICS-Iとのインタフェース部に相当するもので，以下に示すⓐ-ⓗのコマンドを提供する．なお，コマンドの形式は，現在LOGIC-80，MCZ-80Aで稼動中のオペレーティングシステムCP/Mのコマンドと同じにする．これにより見かけ上CP/Mの管理のもとに，直接PHENICS-Iへのアクセスが可能である.
ⓐMTR：通信文を送る.
ⓑRER：他のノードに対しファイルの伝送，あるいは資源（TTY，FDDなど）の利用の承認を求める.
ⓒACR：RERに対する承認を与える.
ⓓFTR：ファイルを伝送する（ファイル名，ソースノード，デスティネーションノードを指定する）.
ⓔCNET：現在のネットワークに関する情報（稼動中のHOST名など）を得る.
ⓕBATCH：他のノードにプログラム（アセンブラなど）の実行を依頼する.
ⓖPROTECT：他のノードに対して，ノー

ドの資源の使用を禁止する．

ⓗ RELEASE ： PROTECT の解除．

上記の8つのコマンドのうち，ⓐ ⓑ ⓓ については PLANET[16] 上で開発したものを用いる．

② APPLICATION PROGRAM ： 2.2 で述べたプロトコル，主として NSP，DAP を記述したプログラムである．HOST 側では，OS/NET のコマンドの実行，あるいは CCU からのメッセージで指定される，ファイルへのアクセス，パケット生成などを行う．一方，CCU 側にはローカルコントローラ（4. で述べる）が実装され，各種の基本サブルーチンを用いて，主としてパケット伝送処理を行う．

③ BASIC PROGRAM ： 2.2 で述べたプロトコル，主として DCP を記述したプログラムである．CCU 内の入力キュー IQ（128 Bytes），出力キュー OQ（512 Bytes）の処理，及びクロックカウンタの処理などを行う．主なものを以下に示す．

ⓐ SIZEIQ(I,L)，SIZEOQ(I,L)： I 番目の IQ（OQ）の現在の大きさを L に入れる．

ⓑ FRONTIQ(I,J,V)，FRONTOQ(I,J,V)： I 番目の IQ（OQ）のフロントより J 番目のデータ（1 Byte）を V に入れる（IQ の内容は変化しない）．

ⓒ ADDOQ(I,V)： I 番目の OQ に V（アドレス）より 72 Bytes のデータを入れる．

ⓓ DELIQ(I,J)，DELOQ(I,J)： I 番目の IQ（OQ）のフロントより J Bytes のデータを消去する．

ⓔ CLKCLR(I)： I 番目のクロックカウンタをクリアする．

ⓕ CLOCK(I,J)： I 番目のクロックの値を J に入れる（クロックは 20 msec おきにカウントアップし，値は 0～255 とする）．

④ INPUT ： CCU 内の UART（3.1 で述べる）が受信した 1 Byte のデータを IQ に移すプログラムである．

⑤ OUTPUT ： OQ に格納されているパケットやコントロールメッセージを CCU 内の UART に移すプログラムである．

3. 通信制御装置（CCU）

3.1 CCU の概要

CCU のハードウェア構成を図4に，主な仕様を表1に示す．CCU には，汎用非同期送受信器（Universal Asynchronous Receiver Transmitter，以下 UART と略す）を3台実装する．3台にした理由は，ネットワークの形態に柔軟性をもたせるための必要最少限の台数であることによる．将来は，CCU の機能を拡張するために UART を4台にすることも検討中である．

CCU は，論理的に，次の2つのブロックに分けられる．

i） CPU ブロック

ii） I/F ブロック

CPU ブロックはマイクロプロセッサ，メモリ，及びタイマの3つから構成されている．マイクロプロセッサは，メモリ上に 4. で述べる制御プログラムを実装し，それを実行する．タイマはシリアルインタフェース（後述）におけるデータの効率よい送信，及び CCU 内の時間管理に用いる．次に，I/F ブロックは，① HOST† - CCU 間のデータ伝送を行うためのパラレルインタフェースと ② CCU - CCU 間のデータ伝送を行うためのシリアルインタフェースの

表1 CCU の主な仕様

CPU	Z-80（クロック 2.5 MHz）
ROM	3 KBytes（2708×3）
RAM	4 KBytes（2114×8）
パラレル I/O	1 Byte(8 bits) ハンドシェイク法 メモリマップド I/O
シリアル I/O	9600 bps 全二重通信 RS-232-C レベル メモリマップド I/O

† 2.2 の HOST ノードと同じ意味で，CCU に接続されるコンピュータ，端末などを表す．

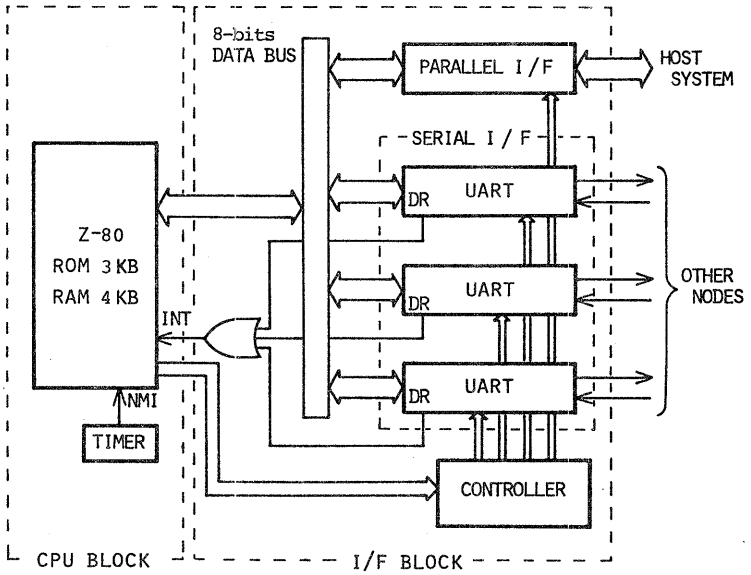


図4　CCUの構成図

2つから構成される．

3.2 CPUブロック

CPUブロックでは，メモリ上に実装された制御プログラムを実行する．上で述べたように，CPUブロックをマイクロプロセッサ（メモリを含む）とタイマの2つに分けて考えることができる．以下では，順次，それらのハードウェア構成の詳細について述べる．

A．マイクロプロセッサ

マイクロプロセッサ及びメモリも含めてその周辺回路には，回路の簡単化と信頼性の向上のためにZ-80 CPUの1ボードマイコンキット（CRC-80）[5]を使用した．しかし，CRC-80をそのままで使用するとプロトコルの実装に不都合が生じるので多少の改造を加えた．以下で主な改造点について説明する．

CCUではCCU間のデータの送受信のために，2つのレベルの異なる割込みを使用している．先ず，優先度の高い割込み（Non Maskable Interrupt，以下NMIと略す）をデータの送信に使用する．NMIはタイマによってある一定の周期で起こる．次にNMIよりも優先度の低い割込み（Interrupt，以下INTと略す）を使用することによって，データの受信を行う．INTはUARTがデータを受信するたびに起こる．他の改造点としては，メモリのアドレスデコーダの変更，制御線のボードからの引き出しなどがある．

B．タイマ

他のCCUへデータを送信する場合，シリアルデータ伝送におけるハードウェア（UART）の制約により，ソフトウェアでの送信のタイミングの管理は，プログラムを複雑にすると共に効率の低下を招く．そのため，送信を割込み（NMI）を利用して一定周期で行うのが良いと思われる．その周期を与えるためにタイマ（Timer）とよばれる発振器を利用する．タイマで発生される割込み周期はデータの伝送速度で決定される．すなわち，UARTでは1Byteのデータを伝送するために，スタートビット（1bit），ストップビット（2bits），パリティビット（1bit）の計4bitsの情報を付加するため，全体で12 bitsのデータの伝送が必要である．従って，今，伝送速度を9600 bpsとすると，

$$9600\,[\text{bits/sec}] \div 12\,[\text{bits}] = 800\,[1/\text{sec}]$$

故に，求める周期は，

$1 \div 800\,[1/\mathrm{sec}] = 1.25\,[\mathrm{msec}]$

となり，データ1Byte伝送するためには，制御のための時間も含めて約1.3 msecの時間が必要となる．タイマには，ハードウェアの簡略化のために，一般のタイマ用ICを使用した．

更に，通信リンクの故障の検出をこのタイマを用いて（タイムアウトとして）行っている．

3.3 I/Fブロック

I/FブロックはHOST-CCU間，及びCCU-CCU間のデータ伝送のためのインタフェース機能をもつ．先にも述べたように，I/Fブロックは，①パラレルインタフェースと ②シリアルインタフェースの2つに分けて考えることができる．以下では，順次，それらのハードウェア構成の詳細について述べる．

A．パラレルインタフェース

これはHOST-CCU間のデータ伝送機能をもつ．データ伝送の制御手順としてハンドシェイク法を用いている．図5にHOST-AからCCU-Bへの一方向のデータ伝送の概略図を示す．

図5では，一方向にしかデータ伝送できないので，図5の回路を双方向に組合わせることによってパラレルインタフェースを実現する．なお，パラレルインタフェースは，メモリマップドI/O方式を採用している．

又，将来はGP-IBなどの標準インタフェースを採用することも検討中である．

B．シリアルインタフェース

これはCCU-CCU間のデータ伝送機能をもつ．UARTを3台実装し，3台のCCU間でデータ伝送が可能である．データ伝送は全二重通信で行う．データの送信についてはタイマの周期による割込み（NMI）によって行う．一方，データの受信についてはUARTがデータを受信した時に立つデータレディフラグ（DR）を使用し，マイクロプロセッサに割込み（INT）をかけることによって行っている．シリアルインタフェースの主な仕様を表2に示す．なお，シリアルインタフェースも，メモリマップドI/O方式を採用している．

将来は現在の通信速度9600 bpsを更に高速化することについて検討中である．

表2　シリアルインタフェースの仕様

UART数	3
通信速度	9600 bps
通信方式	全二重通信
送信方式	タイマによる一定周期送信(NMIによる)
受信方式	DRによる割込み(INT)処理
その他	RS-232-Cレベル プログラムリセット方式 メモリマップドI/O

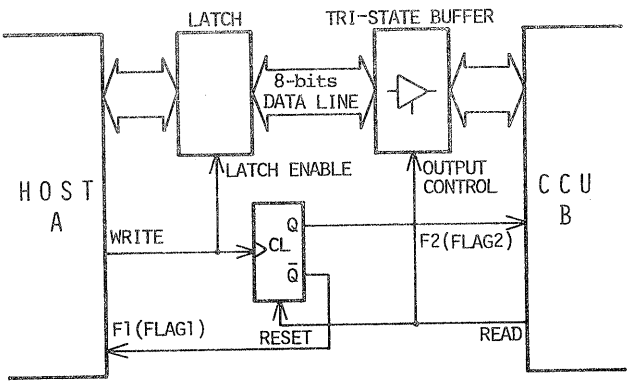


図5　ハンドシェイク法概略図

4. CCUにおける制御プログラム

2.3 の基本ソフトウェアで示した APPLICATION PROGRAM 内のCCU側と明示した部分と BASIC PROGRAM を総称して，CCUにおける制御プログラムとよぶ．このプログラムはCCUのCPUブロックに実装され，実行される．

4.1 CCUのモデル

制御プログラムの説明の都合上，ここでCCU（図4参照）に対するモデルとして，図6に示すものを導入する．なお，入力キューIQ(I)及び出力キューOQ(I)（I＝0，1，2，3）の大きさはそれぞれ 73 Bytes，72×6 Bytes とする．

4.2 パケット交換

PHENICS-I ではパケット交換方式[13][14]でデータ伝送を行う．そのパケットの形式，及びパケットヘッダの形式を図7，図8にそれぞれ示す．各パケットの大きさは 72 Bytesで，そのうち 64 Bytes がデータフィールド，8 Bytes がコントロールフィールドとなっている．

4.3 ルーティング方式[8][10]

基本的には迂回中継方式[7]を採用する．ネットワーク上で各ノードを頂点とする極大木（Spanning tree）をあらかじめ複数個定めておいて，所定の順序でそれらのうちの1つに沿っ

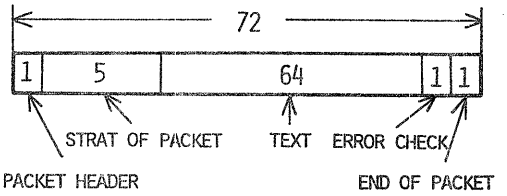


図7　パケットの形式

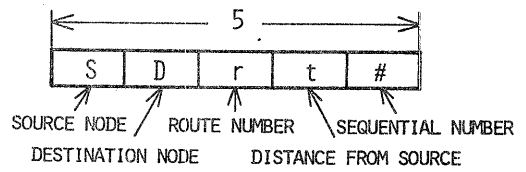
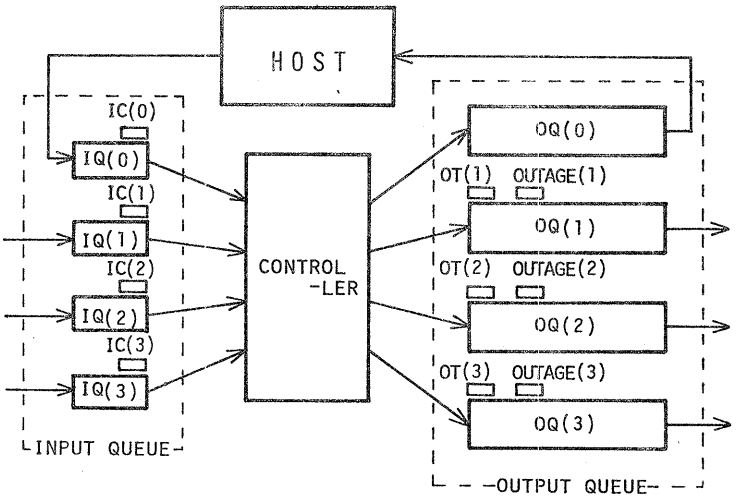


図8　パケットヘッダの形式



IQ(I) (I=0,1,2,3) : Input Queue ; 73 Bytes
OQ(I) (I=0,1,2,3) : Output Queue ; 72×6 Bytes
IC(I) (I=0,1,2,3) : Input queue Counter
OT(I) (I=1,2,3) : Output queue Timer
OUTAGE(I) (I=1,2,3) : OUTAGE flag

図6　CCUのモデル

て（頂点に向かう）パケット伝送を行う．パケットの発信局をソース，受信局をデスティネーションとよぶ．この方式は次の利点をもつ．

① ある極大木に沿ってパケット伝送が（例えば回線故障によって）不可能となった場合でも他の極大木を迂回路として利用することにより信頼性を高くできる．

② 極大木で経路を維持しているので，経路上に閉路は存在しない．

③ 通信リンクの容量など ネットワーク資源を活用するため複数の極大木上の経路でトラヒックを分割できる．

④ ネットワークのトポロジーやトラヒックパターンの重大な変化が起きても各デスティネーションを起動して，そのデスティネーションまでの伝送遅延時間が最小の極大木やその迂回路のための極大木を再構成できる．[12]

⑤ 固定式ルーティングの拡張であるこの方式では，原則的にメッセージの順序は変わらないので，バッファ予約による順序制御は不必要である．

図9にルーティングテーブルの形式を示す．テーブルの大きさは1KBytes，ノード数（CCUの数）は高々64個，経路数（上述の極大木の数）は高々8個を予定している．

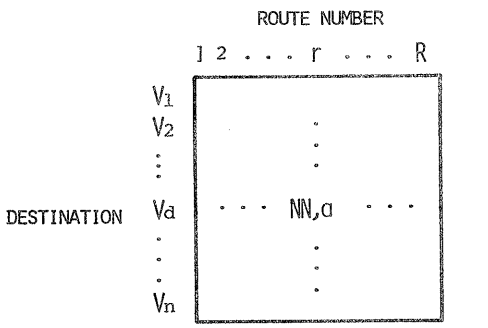


図9　ルーティングテーブル

4.4 フロー制御

フロー制御はネットワークの入口においてネットワークに入ろうとするトラヒックを制御するものである．[2][3][13] 本方式では新たにパケットが入ろうとするネットワークの入口に位置するCCU内の既存のパケットに関する情報のみを調べることによって，トラヒックの停滞を予想しそれを回避することができる．提案するローカルコントローラDCに従って行うトラヒック制御は，ネットワークに入ろうとするパケットだけでなく，既にネットワーク上で伝送中のパケットにおいても実行される．DCは複数の極大木上の経路に基づいて定義されているので，本方式のトラヒック制御はルーティングとフロー制御を同時に実行しているとみなすことができる．このコントローラは制御が簡単であるので，ネットワークに入ることが認められたメッセージに対してはかなり良いパフォーマンスを与えることが予想される．

4.5 ローカルコントローラ（DC）

ネットワーク内の各CCUにおいて各パケットごとに（指定される経路の下での）ソースからの距離，及びデスティネーションまでの距離を併用することによって柔軟性に富んだデッドロックフリーなローカルコントローラDCが構成できる．[10] ここでローカルとは各コントローラはネットワーク全体の状態を調べるのではなく，ローカルな情報だけを用いてパケットを受理すべきか否かを決定できるという意味である．以下では，ローカルコントローラDCに基づいた，各CCUにおけるパケットの処理について説明する（図7～図9参照）．なお，この部分を高級言語で記述したプログラムを付録に示す．

パケットに関するプロトコル

(1) 入力キューIQ(I)内のパケットに対し，パケットヘッダのDを見て，処理を行っているノードがそのパケットのデスティネーションかどうかを確かめる．デスティネーションDとルートナンバrによりルーティングテーブルを参照し，デスティネーションまでの距離α，及びそのパケットの進むべき次のノードNNを得る．

(2) (1)で求めたデスティネーションまでの距離α，パケットヘッダに書き込まれているソ

ースからの距離t，及び出力キュー内の空きバッファの数mを用いて定義されるDCの条件式†を満たすかどうかの判定を行う．

A．条件式を満たす場合（この場合コントローラはパケットを受理するという）

(3) 受理されたパケットに対するエラーチェックを行う．エラーが検出されれば，Ret メッセージを発生し，そのパケットを消し，(4)～(6)の処理を行わない．エラーがなければ，パケットヘッダ内の（ソースからの距離）tをインクリメントする．

(4) (1)で求めたノードNNに接続されている出力キューOQ(J)に入力キューIQ(I)からそのパケットを伝送する．

(5) OQ(J)にパケットが入り終ったらIQ(I)上のパケットを消す．

(6) Ack メッセージをIQ(I)に接続されているノードに返す．

B．条件式が満たされない場合（この場合コントローラはパケットを受理しないという）

(3)′パケットをその入力キューIQ(I)に残し他の入力キューIQ(I′)についてメッセージを取ってくる．

(4)′同一のパケットに対して(3)′の操作が何回行われたかをカウンタIC(I)でカウントする．

(5)′IC(I)の値がある値ℓに達してもなおパケットが受理されない場合，ℓ+1以降の操作ではすべての経路に対して1回ずつ条件式を判定し，受理しうる経路があれば直ちにそのパケットをその経路上のOQ(J)に伝送する．

(6)′受理しうる経路がなければ，やはりパケットを入力キューIQ(I)に残し，他の入力キューIQ(I″)についてメッセージを取ってくる．

(7)′IC(I)の値が(ℓ+ℓ′)に達してもなおパケットが受理されないとき，そのパケットはそのノードにおいて拒否されたものとし，Rej メッセージをIQ(I)に接続されているノードに返し，そのパケットを入力キューIQ(I)から消す（ℓ及びℓ′の値は別途シミュレーションにより決定する）．

最後に，ローカルコントローラDCに基づくリンクの故障からの回復について説明する．

(1) 故障したリンクに接続されている出力キューOQ(J)の先頭にあるパケットのヘッダ及びルーティングテーブルを見てαとtを得る．

(2) 他のすべての経路の中からDCの条件式を満たすものを探す．

(3) (2)で条件式を満たすルートr′が見つかればパケットはr′で送るものとし，ルートナンバrをr′に変更し，ソースからの距離をα−t＝α′−t′を満たすt′に変更する．

(4) パケットをr′に対応する出力キューOQ(J′)に移し換える．

(5) 条件式を満たす経路が見つからない場合，パケットを伝送できないとみなして，そのパケットを消す．

5．むすび

本研究室で開発中の実験的ローカルコンピュータネットワークPHENICS-I についての紹介を行った．ここでPHENICS-I のネットワークとしての特長をまとめると，①パケット交換方式のデータ伝送を採用している，②多経路選択方式でパケット伝送を行う，③任意の形状のネットワーク構成が可能である，④ホストコンピュータの機種は任意でよい，などである．1981年4月より，3台のコンピュータを用いたネットワークは実用段階に入り，引続き，残り3台のコンピュータの接続を行って，PHENICS-I としての完成を目指す予定である．現在，CCUに対するシミュレータを開発中であり，これによるシミュレーションの結果に基づいて各種の定数を定めると共に，次期のネットワークPHENICS-II の設計・製作へ進める計画である．

今回のCCUは主としてHOST にマイクロコンピュータを仮定して設計しているため，CCUの効率はそれほど重大でない．更に，PHENICS-I のプロトタイプとしての開発を

† $DC=\{(m_i, S_{d,r}, D_{d,r}) \mid$ for some r $(1\leqq r\leqq R)$, $D_{d,r}-S_{d,r}<2m_i-k$ and for any r $(1\leqq r\leqq R)$, $0\leqq S_{d,r}+D_{d,r}\leqq k$ and $1\leqq m_i\leqq b\}$（但し，$D_{d,r}=\alpha$, $S_{d,r}=t$, $m_i=m$とする），この条件式の詳しい説明は文献(10)を参照されたい．

容易にするためプロトコルのかなりの部分（図2におけるNSPの一部とDAP）をHOSTに依存している．次期のネットワークPHENICS-Ⅱにおいては，CCUを2プロセッサ化し，DAP以外のプロトコルはすべてCCU上でファームウェア化することを検討している．これによりパケット交換に関する処理はすべてCCU側で行われるためHOSTの負担が軽減される．その結果，ネットワークが汎用性を増し，パフォーマンスも向上すると思われる．一方，ソフトウェアの面では，DNPの機能を拡張し，その上に本格的な分散形データベースを開発することも検討している．なお，PHENICS-ⅡはPHENICS-Iとハードウェア及びソフトウェアの面で互換性を有する様に設計する．

最後に，PHENICS-Iの開発に直接当たってくれている本学学生，荒田一紀，石川裕次，村重　彰，安澤雅行の諸君に感謝する．

文献

(1) 阿江，Vuong，尾崎，片山，伊藤："μCCPによるインハウスネットワークINHOUSE-1" 信学技報，EC79-75，pp. 51-60 (Feb. 1980).

(2) Ahuja,V.:"Routing and flow control in systems network architecture", IBM Syst. J. Vol.18, No.2, pp.298-314 (1979).

(3) Chu,W.W. and Shen,M.Y-C.:"A hierarchical routing and flow control policy (HRFC) for packet switched networks", IEEE Trans. Computer, Vol. C-29, No.11 (Nov. 1980).

(4) Clark,D.D., Pogran,K.T. and Read, D.P.:"An introduction to local area networks", Proc. IEEE, Vol.66, No.11, pp.1497-1517 (Nov. 1978).

(5) "CRC-80 マイコンキットユーザマニュアル"，(株) コンピュータリサーチ (1979).

(6) Contant,G.E. and Wecker,S.:"DNA: an architecture for heterogeneous computer networks", Proc. 3rd ICCC (1976).

(7) 電子通信ハンドブック，電子通信学会編，オーム社，pp. 799-807 (昭和54年).

(8) Jueneman,R.R. and Kerr,G.S.: "Explicit path routing in communications network", ICCC 76, Toronto, Canada, pp.340-342 (1976).

(9) 倉田，村重，安澤："ローカルコントローラの仕様書(2)－高級言語による記述－" ECS Lab., Hiroshima Univ. Tech. Rep. No.80-14 (Dec. 1980).

(10) Kikuno,T., Yoshida,N. and Kakuda,Y.: "Local controllers for packet switching networks", 10-th International Symposium on Fault-Tolerant Computing (FTCS-10), Kyoto, Japan, pp.325-327 (Oct. 1980).

(11) Kikuno,T., Yoshida,N. and Kakuda,Y.: "Theoretical considerations in explicit path routing for computer networks", FTCS-11, to be submitted.

(12) 菊野，吉田，倉田，村重，安澤："ネットワーク上の極大木の計算"，電気四学会中国支部連合大会 72304 p.125 (Nov. 1980).

(13) Kleinrock,L.:"Queueing Systems Vol.II: Computer Applications", John Wiley & Sons (1976).

(14) McQuillan,J.M., Ritcher,I. and Rosen,E.C.:"The new routing algorithm for the ARPANET", IEEE Trans. Commun. Vol.COM-28, pp.711-719 (May 1980).

(15) 苗村，河岡："ネットワークアーキテクチャ"，電子通信学会誌，Vol.62, No.11, pp. 1337-1342 (1979).

(16) 辻川，大寺："小規模コンピュータネットワーク：PLANETに関する研究"，広島大学工学部卒業論文 (1980).

付録　ローカルコントローラの高級言語による記述

以下では，4.5 で述べたローカルコントローラDCの主要な部分についてのみPASCAL

風言語で記述する．現在，これらのプログラムをアセンブリ言語に変換する作業は既に終了している．なお，ローカルコントローラDCの高級言語による完全な記述は文献(9)を参照されたい．

```
program Protocol
  var I,       {current processing queue number}
      CMB(I), {Control Message Buffer of OQ(I)}
      NN,      {Next Node}
      B:array[1..72]; {working area for packet}

  procedure CHANGE(I);
           {CHANGE route: OQ(I)→OQ(I')}
    label L1;
    const R, {total number of Route}
          k; {network constant}
      var D, {Destination node}
          r, {route number}
          t, {distance from source}
         NN, {Next Node}
          a; {distance to destination}
  begin
   FRONTOQ(I,3,D);
   FRONTOQ(I,4,r);
   FRONTOQ(I,5,t);
   TABLE(D,r,NN,a);
   for r':=1 to R do {r' is new route.}
   begin
    TABLE(D,r',NN',a'); {' shows new object.}
    if NN'≠I then {exclude original OQ}
    begin
     SIZEOQ(NN',L);
     t':=t+a'-a
     if (a'+t'≤k) and (L≤360) then
     begin
      FETCHOQ(I,B(J));
      B(4):=r'; {set new route}
      B(5):=t'; {set hop count}
      ADDOQ(NN',B(1));
      SIZEOQ(NN',L);
      if L=72 then OUTPAC(NN');
      goto L1
     end
    end
   end;
   BUSY(I)
L1: end;

  procedure FAILURE(I);
  begin
   OUTO(I);
   SIZEOQ(I,L);
   if L≥72 then CHANGE(I)
  end;

procedure Packet;
  label L2,L3;
  const U1, {Upper-bound of judge time}
        U2, {Upper-bound of judge time}
         N, {Node number itself}
       Rej, {Reject message}
         k, {network constant}
         R; {total number of route}
    var m, {number of empty buffers}
        D, {Destination}
        r, {route number}
        t, {distance from source}
        a; {distance to destination}
begin
 SIZEIQ(I,L);
 if L≥72 then
 begin
  if I=0 then {packet from HOST}
  begin
   r:=1; {initialize packet route}
   t:=0; {initialize hop count}
   FRONTIQ(I,3,D);
   TABLE(D,1,NN,a)
  end
  else {all nodes except for HOST}
  begin
   FRONTIQ(I,3,D);
   FRONTIQ(I,4,r);
   FRONTIQ(I,5,t);
   TABLE(D,r,NN,a)
  end
 end;
 SIZEOQ(NN,L);
 if (a-t < 2*m-k) and (t+a≤k) and (L≤360)
       {DC decision along primary route}
 then  {Packet is accepted.}
 begin
   FETCHIQ(I,B(J));
   if I=0 then
   begin
    B(2):=N; {set source Node number}
    B(4):=1; {initialize route number}
    B(5):=0 {initialize hop count}
   end
   else B(5):=t+1; {increment hop count}
   TRANSMIT(NN,B(J));
   goto L2
 end
else {Packet is not accepted.}
begin
 if IC(I)≤U1 then IC(I):=IC(I)+1
 else
 begin
  for r':=1 to R do
  begin
   TABLE(D,r',NN',a');
   if NN≠NN' then
   begin
    SIZEOQ(NN',L);
    t':=t-a+a';
    if (a'-t' < 2*m-k) and (L≤360) and
       (a'+t'≤k)
    {DC decision along alternate route}
    then {Packet is accepted.}
    begin
     FETCHIQ(I,B(J));
     B(4):=r';
     B(5):=t'+1;
     TRANSMIT(NN',B(J));
     goto L2
    end
   end
  end;
```

```
      if IC(I) < U1+U2 then
      begin
       IC(I):=IC(I)+1;
       goto L3
      end
      else
      begin
       CMB(I):=Rej;
       OUTCMB(I);
       DELIQ(I,72)
      end;
  L2: IC(I):=0
     end
L3: end
   end;
```

```
    {*** Main Procedure ***}
     begin
      Initialize;
      while H=0 do
      begin
      {*** Input Queue Process ***}
       HOSTIQ;
       for I:=1 to 3 do
       begin
        FRONTIQ(I,1,V);
        case V of
        SOP:  Packet;
        Ack:  Acknowledge;
        Rej:  Reject;
        Ret:  Retransmit;
        Rec:  Recovery;
       Zero:  Recoverydetect;
     Others:  Others
        end
       end;
       {*** Output Queue Process ***}
       HOSTOQ;
       for I:=1 to 3 do
       begin
        if OUTAGE(I)=1 then FAILURE(I)
        else
        begin
         OUTCMB(I);
         LOCL(I)
        end
       end;
       LOOKUP(H) {check if Halt or not}
      end
     end.
```

The details of the following procedures are omitted.

```
procedure SetTable
procedure Initialize
procedure TABLE(D,r,NN,a)
procedure FETCHIQ(I,B(J))
procedure FETCHOQ(I,B(J))
procedure TRANSMIT(NN,B(J))
procedure OUTPAC(NN)
procedure OUTCMB(I)
procedure BUSY(I)
procedure HOSTIQ
procedure HOSTOQ
procedure Acknowledge
procedure Reject
procedure Retransmit
procedure Recovery
procedure Recoverydetect
procedure Others
procedure OUTO(I)
procedure LOCL(I)
procedure LOOKUP(H)
```