「情報処理」第19巻第1号別刷　昭和53年1月発行

コンピュータ・ネットワークにおける NCP の設計に関する一考察

海老原　義　彦　　野　口　正　一

社団法人 情報処理学会

論 文

コンピュータ・ネットワークにおけるNCPの設計に関する一考察*

海老原 義彦** 野口 正一***

Abstract

This paper describes one of procedures of process connection which takes an important part of NCP (Network Control Program) in compter network. Main points are briefly; 1) Mutual definitions of formats and data types to communicate with different hosts in a Net are classified into net-level one, host-level one, subprocess-level one and so on. Process consists of subprocesses with hierarchical structure and communicates through the same leveled subprocesses. Therefore process connection is built up by establishing each subprocess connection hierarchically. 2) Process connection procedures are represented by state-transition diagrams of automation. 3) The connection control program consists of fundamental blocks suitable for its implementation. Through the implementation on ALOHA-NCP, some results are shown.

1. はじめに

最近のコンピュータ・ネットワークは種々のアプリケーション・プロトコル[1)~5)]を提供しており，NCP (Network Control Program) のもつ機能は複雑化かつ多様化する傾向にある．このような複雑なNCPのもつ論理を適切に表現することは，今後のNCP設計の上で重要な課題である．このため本論文では，NCPにプロセスの階層構造を導入し，NCPのもつ論理の明確化を計る一設計方針を考察した．特に本論文ではNCPの中核をなすプロセス結合制御について述べる．

一般論として複雑な論理の設計においては，論理を単純な単位に分け，それらを組み合せて一つの把握しやすい論理にまとめてゆくことが設計の一つの基本思想である．本論文では各種プロトコルの最少単位をプロセスと呼び，ネットワーク間の結合制御をプロセス単位で行う．このことによりすべての制御がプロセス単位になるので，ホストがどこまでオペレーティング・システムの仕事をするかを明確にすることができる．

次の問題はどの程度までプロセスに機能をもたせるかである．当然，その機能の範囲の決め方は各種プロトコルのもつ機能の構造に適合し，この動作をよく反映したものでなければならない．一般に，各プロトコルの実行において共通する制御の流れは，次のとおりである．まず，どのプロトコルのサービスを受けるか，そしてどのような形式で行われるかを定め，最後に要求の実行が行われるといった階層性がある．このことから必然的にプロトコルの実行には，各階層ごとに，それを支援するプロセスが存在し，それぞれが実行を行うと考えるのは，より一般的である．

本論文は以上の設計思想により，NCPの設計のための方法論を与え，その記述にオートマトン的手法を用いて，具体的に表現する方法を示したものである．特に，プロセス制御プログラムの基本的動作は，相手ホストからの入力メッセージによりプロセスの結合状態が逐次遷移してゆくので，オートマトン的記述は，プロセス結合プログラムの表現に，きわめて適合したものであると考えられる．

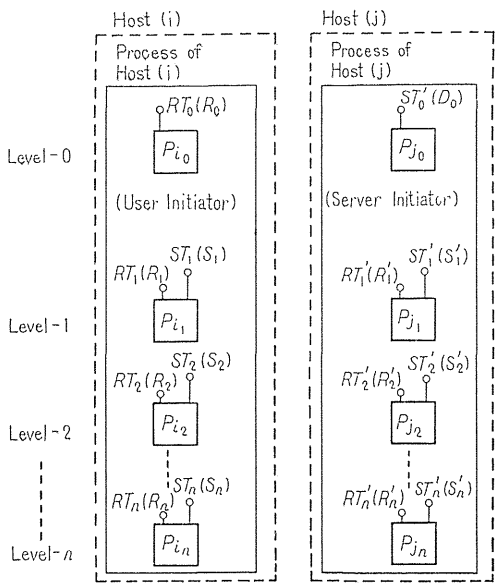
以下，本稿では，ARPAネットの結合手順に準じ

* On a Design of NCP for Computer Network by Yoshihiko EBIHARA (The Institute of Electronics and Information Science, Tsukuba University) and Shoichi NOGUCHI (The Research Institute of Electrical Communication, Tohoku University)
** 筑波大学電子情報工学系
*** 東北大学電気通信研究所

た ALOHA*-NCP[8),9)] の実例をかかげ，著者らの設計思想により，一般化したプロセス結合制御方式について述べる．

2. 結合手順における諸定義

プロセス：プロセスは **Fig. 1** に示すような階層構造をもったサブプロセス (P_{jn}) の集合からなる．各サブプロセスは受信および送信端子の一対をもち，それは実際の端子番号（ソケット番号）が与えられたとき，実行サブプロセスと呼ばれ，データの送受を実行しながら一連の処理を行うソフトウェア・モジュールである．また，サブプロセス間の交信は同層のもの同志しかできない．このような階層構造の意味は次のことから必要である．ネットワークには異なる機種からなるホストが存在し，各ホストでは，異なる符号体系または文字の定義を行っていることもある．そのため，相互の会話の基盤としてレベル-0 ではネットワーク全体で定めたビット・インディペンダンス符号を使い，レベル-1 の P_{i_1} と P_{j_1} のデータ送受はネットワーク全体，またはホスト間で定めた共通符号体系を設定して，これを使っている．また，P_{i_1} と P_{j_1} の会話をとおしてレベル-2 の P_{i_2} と P_{j_2} で実行される機能の詳細を定義する．このようにレベル-2 でのデータ情報の表現形体は，レベル-1 の会話で新たに定義することができ，レベル-1 の共通符号体系とは別な符号体系をとりうる．



For $n=1,2,3\cdots$
RT_n: Receive terminal at level-n, ST_n: Send terminal at level-n
R_n: Receive socket number assigned to RT at level-n
S_n: Send socket number assigned to ST at level-n
P_{in}: Subprocess of host(i) at level-n
P_{jn}: Subprocess of host(j) at level-n
Apostrophized RT_n, ST_n, R_n and S_n express those of host(j)
For $n=0$
RT_0: Receive terminal of host(i) at level-0
ST_0: Send terminal of host(j) at level-0
R_0: Receive socket number of host(i) at level-0
S_0: Send socket number of host(j) at level-0 $S_0=D_0$
P_{i0}: User initiator
P_{j0}: Server initiator

Fig. 1 Process related to subprosses

ネットワーク全体，またはホスト間で定めた共通符号体系を設定して，これを使っている．また，P_{i_1} と P_{j_1} の会話をとおしてレベル-2 の P_{i_2} と P_{j_2} で実行される機能の詳細を定義する．このようにレベル-2 でのデータ情報の表現形体は，レベル-1 の会話で新たに定義することができ，レベル-1 の共通符号体系とは別な符号体系をとりうる．

一般的には，レベル-n まで以上の操作を繰り返して，プロセスの機能を実行してゆくものとする．

次に，具体的にファイル転送プロトコルの実例をとりあげて，階層構造の正当性を示す．まず，レベル-0 ではネットトークの提供する種々のプロトコルからファイル転送処理をサービスするプロトコルの選択をしなければならない（本システムでは，これをイニシェータと呼ぶ）．次に，ファイルの転送をどのようなファイル構成，どの符号体系で，また，どのような機能で行うかを異なるホスト間で取り決める必要がある（これをファイル転送制御サブプロセスと呼ぶ）．レベル-2 ではレベル-1 の定義に従って実際のファイルの転送を実行する（これを，ファイル転送実行サブプロセスと呼ぶ）．このように各プロトコルは，いくつかの階層を経るのは一般的である，かつ，後述の実験と検討にも述べているように，プロセスの階層構造は経験的妥当性をも十分満足しているものである．

ソケット番号：ソケット番号は各サブプロセスの送受信端子を識別するためのものである．重複したソケット番号を与えないため，ネットワーク内のホスト番号をすべて異なるようにつけ，ホスト内のソケット番号を異なるようにすれば，サブプロセスの端子の識別はネットワークで唯一に定まる．ネットワーク中のソケット番号はホスト番号とソケット番号で定義される．以下では，ソケット番号はホスト内のソケット番号が主になるので，ホスト内のソケット番号を示す．

リンク：リンクはサブプロセス間を結合させる論理パスと定義する．その論理パスにもちいる線路を用途別または優先度によって分ける必要があるため，論理的線路にリンク番号**を与えて区別する．

---

* Additive Link On-line Hawaii Area の略

** 本論文では次のように与えている．

LINK 番号	使用目的
0	最優先度の制御メッセージ用
2～71	データ・メッセージ用
72～158	未使用
159～191	ネットワーク情報収集用
192～255	個人の実験用

サブプロセス間の結合：同層における1対1のサブプロセス間の結合である．たとえば，P_{i_1} と P_{j_1} のレベル-1 でのサブプロセス結合とは，P_{i_1} の受信端子にシステムから動的に与えられたソケット番号 (R_1) と P_{j_1} の送信端子に動的に与えられたソケット番号 (S_1') をそれぞれのホストに知らせ，R_1 と S_1' に論理パスを設定する．同様に P_{i_1} の送信端子のソケット番号 (S_1) と P_{j_1} の受信端子のソケット番号 (R_1') を交換し，論理パスを設定することと定義する．この手順は **3.** で詳しく述べる．

プロセス結合：プロセス結合はレベル-0 から必要なレベル-n までのサブプロセス間の結合を行う．

プロセス結合は次の3つのフェーズからなる．

フェーズⅠ：イニシェーションのための操作．

フェーズⅡ：レベル-1 から n までの結合操作．

フェーズⅢ：レベル-n から1までの結合切断．

以上の諸定義をもとに，具体的なプロセス結合制御手順を述べる．

3. プロセス結合方式

プロセス間の結合方式を述べるにあたって，まず，どのようにして目的のプロトコルのサービスを受けるかについて説明する必要がある。ホスト (i) のユーザの相手ホスト指定コマンドで，相手ホスト (j) との物理的回線結合が行われ，次に，プロトコル指定コマンドによりホスト (i) 内の必要な自分自身のプロセスが定まる．そして，ホスト (i) は最初の制御メッセージ (D_0 指定) を相手ホストに送ることによりプロトコルを決めることができる．そのわけは，各ホストは他のホストからの処理要求に従った共通のソケット番号* (D_0) を有しているからである。ホスト (j) はホスト (i) からの最初の制御メッセージで自己内の目的プロセスを知ることができる．このように，共通ソケット番号を定めることにより，新たなアプリケーション・プロトコルの追加も容易となる．

以下の説明はホスト (i) を中心に述べ，結合の要求する側をユーザとし，要求を受ける側をサーバと定義する．この場合 Fig. 1 の P_{i_0} はユーザ・イニシェータ (UI) と，P_{j_0} はサーバ・イニシェータ (SI) と呼ぶことにする．

3.1 結合手順

今，ホスト (i) からホスト (j) のソケット番号 D_0 を指定しているものとする．さらに，ホスト (i) のシステムは代表ソケット番号 R_0 を，グルーピングされたソケット・プールから選び，CREATE (1) コマンドにより，R_0 をもった実行サブプロセス UI を生成しているものとする．また，各制御メッセージは3英文字記号で示し，記号の右肩の (i) はホスト (i) からホスト (j) へ，(j) はホスト (j) からホスト (i) へのメッセージの流れを表わす。各記号とそれに続く( ) のパラメータは，それぞれ **Table 1** と **Table 2** (次頁参照) の Note および Fig. 1 に説明してあるので参照のこと．

フェーズ I (レベル-0 の結合)

① ホスト(i)は相手ホスト(j)に D_0 を指定して，R_0 を含む制御メッセージ $\mathrm{RTS}^{(i)}(R_0, D_0, l_0)$ を送信する**.

② ホスト (j) は確認をとるため，制御メッセージ STR をホスト(i)に送る．そのフォーマットは次のものである．; $\mathrm{STR}^{(j)}(D_0, R_0, s_0)$

①と②項により D_0 と R_0 ソケット番号の認識と論理リンク l_0 が確立されたことになる．

③ ホスト(j)のシステムが P_{j_1} に割り当てる S_1', R_1' を選び，SI はリンク l_0 を介して受信ソケット $R'=S'-1$ を情報として UI に送る．ホスト (j) のシステムは，コマンド SEND $M(R')$ で送信を実行する．

④ 両イニシェータ間の交信を切断する．以下，これはプロセス結合のイニシェーションの終了を意味する．ホスト(i)の受信端子 R_0 からホスト(j)の送信端子 D_0 に切断制御メッセージを送る．; $\text{R-CLS}^{(i)}(R_0, D_0)$

⑤ 確認をとるためホスト (j) から (i) の R_0 に切断制御メッセージ $\text{R-CLS}^{(j)}(D_0, R_0)$ を送る．

以上 SI, UI では解放され，次の受け付けをまつ．

フェーズ II (レベル-1 の結合)

⑥ ホスト (i) はレベル-1 の P_{i_1} に R_1, S_1 のソケット番号を割り当て，実行サブプロセスを生成する．但し，$R_1=R_0+2, S_1=R_1+1$ とする．

⑦ ホスト(i)から $\mathrm{RTS}^{(i)}(R_1, S_1', l_2)$ をホスト(j)

---

* ホストの共通ソケット番号の種類の1例

ソケット番号 D_0	サーバ・イニシェータの機能
1	ホスト使用のイニシェーション
2	ファイル転送処理のイニシェーション
5	Remote Job Entry 機能のイニシェーション
65	Speech Data Base 機能のイニシェーション

** 以下，リンクに送信量を割り当てる制御フローは結合方式に直接関係ないので省略する．本システムでの ALL の送信は何回かできるものとする．すなわち，メッセージ数 a と総ビット数 b を ALL により相手に割り当て，a または b の一方の分のデータを受信した時点で，再び送信量割り当ての ALL を送信するものとする．

Table 1 Input symbols

CREATE (1): Creation of User Initiator.
CREATE (2): Creation of subprocess.
CONNECTION (1): Request for UI initiation from system.
CONNECTION (2): Connection request of subprocess from system.
CLOSE (1): Termination of UI initiation.
CLOSE (2): Termination of SI initiation.
CLOSE (3): Disconnection of subprocess connection.
SEND: Transmission request for data message from system.
RECEIVE: Receive request of data message from system.
RELEASE: Resource release request, used in subprocess connection, from system.
LISTEN: SI's waiting for initiation request from UI.
READY: End of interrupt management.
INSG: Acceptance of interrupt management.
Δ: Status inquiry of subprocesses.
[NA]: Last message transmission error } [(1)] for control one.
[AC]: Successful last message transmission } [(2)] for data one.
*: Time-out for elapse of a certain time in state.
**(n): Resource release of level-n.
E(S): Event indicating Send Counter (SC) being not zero.
$\overline{E(S)}$: Event indicating SC being zero.
E(R): Event indicating Receive Counter (RC) being not zero.
$\overline{E(R)}$: Event indicating RC being zero.
*(D): Complete processing of remained data messages.
AM(j): Any messages from the host (j).
M(j), M(R_1'): Data message from the host (j).
N(jI): Interrupt data message from the host (j).
INT: Interrupt request from the system.
RTS$^{(j)}$: Host(j)'s receiver-to-sender control message for a connection, RTS$^{(j)}$ (R_n', S_n, l_{2n-1}), where R_n', S_n, l_{2n-1} are host(j)'s receiver and host(i)'s sender sockets, and link at the level-n, relatively.
STR$^{(j)}$: Host(j)' sender-to-receiver control message for a connection, STR$^{(j)}$ (S_n', R_n, s_{2n}), where S_n', R_n', s_{2n} are host(j)'s send and host(i)'s receive sockets, and a logical byte size, relatively.
ALL$^{(j)}$: Host(j)'s space allocation control message, ALL$^{(j)}$(l_{2n-1}, a, b) where a and b are message and bit count assigned to l_{2n-1}.
R-CLS$^{(j)}$: Host(j)'s close control message to host(i)'s receive terminal of the level-n subprocess, R-CLS$^{(j)}$(S_n', R_n).
S-CLS$^{(j)}$: Host(j)'s close control message to host(i)'s send terminal of the level-n.
INS$^{(j)}$: Host(j)'s interrupt control message, INS$^{(j)}$(l_{2n-1}).

NOTE
[ ]: Subnet control message.
_____: Underline of symbol means event from the system of host(i). Nonexistence of parameters and control message is INS$^{(j)}$(l_{-1}), RTS$^{(j)}$(R_0', S_0, l_{-1}) and S-CLS$^{(j)}$(R_0', S_0).

Table 2 Output symbols

RTS$^{(i)}$: Host(i)'s receiver-to-sender control message for a connection, RTS$^{(i)}$(R_n, S_n', l_{2n}).
STR$^{(i)}$, ALL$^{(i)}$, S-CLS$^{(i)}$, R-CLS$^{(i)}$: Explanations of the symbols are almost same as those of input symbols in table 1, replacing (i) by (j) and (j) by (i), except parameters, STR$^{(i)}$ (S_n, R_n', s_{2n-1}), ALL$^{(i)}$(l_{2n}, a, b), S-CLS$^{(i)}$(S_n, R_n'), R-CLS$^{(i)}$(R_n, S_n').
NO*: Subprocess troubles at a process connection request.
M(iI): Host(i)'sinterropt data massage.
DEQ: Buffer release of transmitted last massage from queue.
ENQ: Acception of received massage on queue.
D(RC): Decrement received massage and bit space from receive count.
D(SC): Decrement transmitted message and bit space from send count.
S(RC): Set RC a mount of message and bit space assigned by ALL$^{(i)}$.
S(SC): Set SC a mount of message and bit space assigned by ALL$^{(j)}$.
NEG: Neglect all messages.
INS$^{(i)}$: Host(i)'s interrupt cotrol message, INS$^{(i)}$(l_{2n}).
OK: Completion of creating UI or subprocess.

NOTE
Nonexistence of parameters and control messages is STR$^{(i)}$(S_0, R_0', s_{-1}) and S-CLS$^{(i)}$(S_0, R_0').
parameters l_m and s_m means:
l_m: $m=2n$, a link number assigned by host(i).
$m=2n-1$, a link number assigned by host(j)
s_m: $m=2n$, a logical byte size of data transferred on link l_m.
$m=2n-1$, a logical bytes ize of data transferred' on link l_m.
where, m is interger number and n is interger number indicating level of connection.

に送信する．

⑧ ホスト (j) はレベル-1 の P_{j_1} に S_1', R_1' のソケット番号をもつ実行サブプロセスを生成する．

⑨ ホスト (j) から (i) に $\mathrm{STR}^{(j)}(S_1', R_1, s_2)$ を送る．

⑦と⑨項により，R_1 と S_1' ソケット番号の認識と論理リンク l_2 が確立されたことになる．

⑩ ホスト (j) から (i) に $\mathrm{RTS}^{(j)}(R_1', S_1, l_1)$ を送信する．

⑪ ホスト (i) から (j) に $\mathrm{STR}^{(i)}(S_1, R_1', s_1)$ を送信する．

⑩と⑪項により，S_1 と R_1' ソケット番号の認識とリンク l_1 の確立がなされた．⑦, ⑨, ⑩と⑪項の手続きにより，サブプロセス P_{i_1} と P_{j_1} との結合がなされる．

フェーズ II（レベル-2 の結合のための諸定義*）

⑫ リンク l_2 を介して，ホスト (j) のシステムが選んだソケット番号 R_2' をホスト (i) に送る．SEND $M(R_2')$ コマンドによる．

⑬ リンク l_1 を介して，ホスト (i) のシステムが選んだソケット番号 R_2 をホスト (j) に送る．SEND $M(R_2)$ コマンドによる．

フェーズ II（レベル-2 の結合）

フェーズ I（レベル-1 の結合）と同様な手順で，レベル-2 のサブプロセス結合を実行する．すなわち，P_{i_2} に S_2, R_2 のソケット番号を，また P_{j_2} に S_2', R_2' のソケット番号をそれぞれ割り当て，実行サブプロセスを生成する。その後，相互のソケット番号と論理リンク l_3, l_4 を確立することである．但し $R_2=R_0+4$, $S_2=R_0+5$, $S_2'=R_2'+1$ かつ $R_2'=R_1'+2$ である．

以下，ファイル転送プロトコル等においては，実際の情報が l_3, l_4 を介して行われる．

フェーズ III（結合切断）

⑭ ホスト (i) の受信ソケット番号 R_2 からホスト (j) の送信ソケット番号 S_2' に $\mathrm{R\text{-}CLS}^{(i)}(R_2, S_2')$ を送信する．

⑮ ホスト (j) の送信ソケット番号 S_2' からホスト (i) の受信ソケット番号 R_2 に，確認のための $\mathrm{R\text{-}CLS}^{(j)}(S'_2, R_2)$ を送信する．

⑭, ⑮ 項により，S_2', R_2 のソケット番号とリンク l_4 が解放される．

⑯ ホスト (i) の S_2 からホスト (j) の R_2' に $\mathrm{S\text{-}CLS}^{(i)}(S_2, R_2')$ の送信を行う．

⑰ ホスト (j) の R_2' からホスト (j) の S_2 に確認のための $\mathrm{S\text{-}CLS}^{(j)}(R_2', S_2)$ を送信する．

⑯, ⑰項により，ソケット番号 S_2, R_2' とリンク l_3 が解放される．また，⑭, ⑮, ⑯と⑰により，レベル-2 の P_{i_2} と P_{j_2} の結合切断が完了したことになる．

レベル-1 の P_{i_1} と P_{j_1} の結合切断も，レベル-2 の切断手続きと同様な処理を行う．すなわち，切断制御メッセージによる相互のソケットの確認を行って，R_1, S_1' と S_1, R_1' のそれぞれのソケットおよびリンク l_2 と l_1 が解放される．以上で，プロセスの結合切断が終了する．このように，レベル-n までも，同様なプロセス結合制御手順で実行される．

4. プロセス制御手順のブロック化

結合手順を 6 つの基本的なブロックに分け，各ブロックは状態遷移図で表現している．図の中で，○と▭は状態を表わし，各状態の説明は **Table 3**（次頁参照）に示す．矢印は遷移先を表わし，複数の遷移先がある状態では，複数入力記号が識別できるものとする．状態から状態への遷移時の処理に関する完全なドキュメンテーションは NCP 設計にあたって記述しているが，ページ数の制限のため，詳細を省き，本稿では入出力事象の関係を中心に述べる．なお，グローバルな処理動作は Table 1 と Table 2 の入出力記号において記述している．また，図の△と□は他のブロックにまたがる場合であり，その中の記号または番号は行き先きを示す．

次に遷移図における異常入力事象の表現について述べる．本システムでは，あらゆる異常入力信号は，プロセス結合制御プログラムにあがってくるまでに，前処理過程でチェックされる．前処理とは，①IMP (Interface Message Processor) 間制御プログラムによるもの，②IMP・ホスト間の制御プログラムによるものと③ホスト間制御プログラムによるものがある**．プロセス結合制御プログラムは，これらの前処理の内容は一切関知しない．関係するものは，前処理の結果として生じる事象のみである．それらのうち異常入力事象について主なものを，次に述べる．

ⓐ IMP または相手ホストのシステム・ダウン

---

* 個々のアプリケーション・プロトコルで定めたその他の情報交換がなされるが，制御手順の観点から相互のソケット交換のみを示す．

** 各制御プログラムの前処理の内容

① IMP 間のパケット同期，誤り検出，再送処理や再送要求処理等

② IMP・ホスト間のメッセージ長チェック，メッセージ・パケット間の分解・合成やパリティ・エラー処理等

③ ホスト間のメッセージ・フォーマットのチェック，リンクやソケット番号の論理的正当性のチェック処理等

Table 3 State symbols

INITIAL-1: Initial state for connection, where distinction between CONNECT(1) and LISTEN is assumed to be done.
INITIAL-2: Initialization end state.
LISTEN: Wait state for acception of connection request.
ACCEPT A_1: Wait for request of subprocess test inquiry from system.
A_2: Wait for reply [AC(1)] of transmission STR(i).
OPEN
O_1: State able to receive data message.
O_2: Waiting for data message.
O_3: Check if there is enough space to continue receiving data.
O_4: Wait for CLOSE(1) command.
O_5: State able to send data message.
O_6: Wait for successful transmission $M(R_1)$.
O_7: Check if there is enough space to continue transmitting data
O_8: Wait for CLOSE(2) command.
CLOSE
C_1: Wait for successful transmission of close control messages.
C_2: Wait for replies to close control messages from host (j).
C_3: Wait for resource-release request.
C_4: Wait for end of received data message processing.
C_5: Wait for termination of resource release.
RELEASE
R_1: Wait for termination of resource release.
CONNECT
T_1: Wait for request of creating executing-subprocess.
T_2: Wait for successful transmission.
NORMAL OPEN
N_1: State able to receive and send data messages.
N_2: Wait for data message $M(j)$.
N_3: Wait for data message $M(i)$.
N_4: Check if there is enough space to continue receiving.
N_5: Check if there is enough space to continue sending.
N_6: Wait for successful transmission ALL(i).
N_7: Wait for ALL(j).
FORCED CLOSE
F_1: Wait for request of resource release.
F_2: Wait for termination of resource release.
INTERRUPT
I_1: Wait for successful transmission INS(i).
I_2: Wait for RECEIVE command.
I_3: Wait for $M(jI)$.
I_4: Check if there is enough space to continue receiving.
I_5: Wait for successful transmission ALL(i).
I_6: Check if there is enough space to continue sending.
I_7: Wait for successful transmission ALL(i).
I_8: Wait for starting signal of interrupt management INSG.
I_9: Wait for completion of interrupt management.
I_{10}: Wait for request SEND command.
I_{11}: Wait for successful transmission $M(iI)$.
I_{12}: Check if there's enough space to continue sending.
I_{13}: Wait for space-allocation control message ALL(j).

ⓑ メッセージに論理エラーがあった.
ⓒ 送信メッセージが相手に届かない.
ⓓ 受信すべきメッセージが受信できない.
ⓔ サブプロセスの異常処理等.

これらの異常状態は，すべてタイム・アウト事象としてプロセス制御プログラムは処理する．但し，異常処理プログラムは，タイム・アウトの種類を区別できるものとする．

U-INITIATE ブロック：このブロックはホスト(i)の UI 側のフェーズⅠにあたる (**Fig. 2**(次頁参照)．次に，このブロックの各状態を述べる．INITIAL-1；プロセス結合の初期状態であり，システムからの CREATE コマンドにより，ホスト (i) に実行サブプロセス P_{i_1} を生成する．T_1；システムからの CONNECT コマンドにより R_0 と D_0 のパラメータ (**3.1** 参照) をもつ RTS(i) と ALL(i) をホスト(j)に送信して状態 T_2 に移る．T_2；RTS(i) と ALL(i) が，正しくホスト(j)に伝わる ([AC(1)]) と受信カウンタに ALL(i) で指定したメッセージとビット数を記憶する．そうでない場合は ([NA(1)]) はタイム・アウト (∗) 内で，[AC(1)] が返るまで再送を繰り返す．T_3；ホスト(j)から RTS(i) の応答である STR(j) を受信して状態 O_1 に移る．応答待ちのタイム・アウトが生ずると強制切断処理に入る．O_1；システムからの RECEIVE コマンドにより，ホスト(j)からの P_{j_1} の受信ソケット待ちである状態 O_2 に移る．O_2；ホスト(j)から受信ソケット R_1' をタイム・アウト内に受信すると，これをホスト(i)の UI に登録する．のちに，システムはこの R_1' を，レベル-1 のサブプロセス結合に使用する．O_3；R_1' のみの受信であるから，受信カウンタの値にかかわらずに O_4 状態に移る．システムの CLOSE (1) コマンドにより R-CLS(i) を送信する．C_1；R-CLS(i) の [AC(1)] が返ると状態 C_2 に移り，そうでない場合は R-CLS(i) を [AC(1)] が返るまで，タイム・アウト内で再送を繰り返す．C_2；タイム・アウト内に R-CLS(i) に対する R-CLS(j) を受信すると状態 C_5 に移る．C_5；システムからのリソース解放 (RELEASE) コマンドにより，状態 R_1 に移る．R_1；UI が使用したリソースの解放の終了により INITIAL-2 に移る．

以下のブロックの説明は同様の動作であるので，詳細を省き，概略を述べる．

S-INITIATE ブロック：このブロックはホスト(i)の SI 側のフェーズⅠにあたり，相手ホストからのプロセス結合要求を受けつける．かつ，結合のためのイニシェーションを行う (Fig. 2 参照)．

NORMAL OPEN-n ブロック：レベル-n の P_{i_n} と P_{j_n} の半２重のデータ・メッセージ送受信と割り込み処理を行う．ホスト (i) からのデータ・メッセージ $M(i)$ とホスト (j) からのデータ・メッセージ $M(j)$ のフォーマットや構成符号等は，NORMAL OPEN-

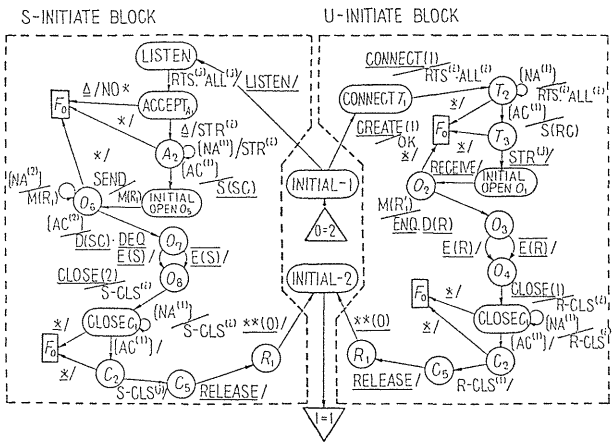


Fig. 2 Initiation block.

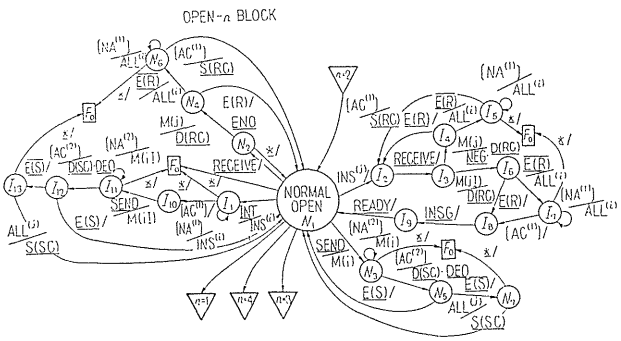


Fig. 3 Open-n block.

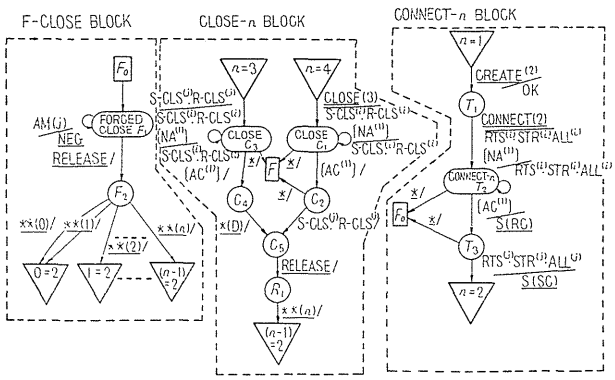


Fig. 4 F-Close, Close-n and Connect-n blocks.

$(n-1)$ の会話で定義されたものに従う(但し $n\geq 1$). 割り込み処理に関しては，プロセス結合制御と直接な関係はないと思われるので，説明を省く（**Fig. 3** 参照）.

CLOSE-n ブロック：レベレ-n $(n\geq 1)$ の実行サブプロセス P_{i_n} と P_{j_n} 間の結合切断を行う．以下，切断要求のタイミングに関して述べる．

システムからの CLOSE (3) コマンドか，または相手ホスト (j) からの切断制御メッセージのうち，先に発生した事象が優先的に処理され，状態 (N_1) からそれぞれ状態 C_1 または C_3 に移る（**Fig. 4** 中央図）．レベル-n のサブプロセスのリソース解放終了後，制御はレベル-$(n-1)$のオープン状態 (N_1) に戻る．但し，レベル-1 の時は INITIAL-1 状態に戻るものとする (Fig. 4 中央図).

CONNECT-n ブロック：結合制御手順のフェーズⅡのうち，レベル-n $(n\geq 1)$ のサブプロセス P_{i_n} と P_{j_n} 間のサブプロセス結合を実行する．Fig. 4 の右図がこのブロックにあたる．図中，制御メッセージのパラメータは省略してあるが，サブプロセス結合に必要な R_n, R_n' はレベル-$(n-1)$ の実行サブプロセス $P_{i_{n-1}}$ と $P_{j_{n-1}}$ 間でえられたものである．但し，S_n, S_n' は $S_n=R_n+1, S_n'=R_n'+1$ の関係にあるものとする．また R_1, R_1' はそれぞれ SI, UI のイニシェーションにより知るものとする．

FORCED CLOSE ブロック：各サブプロセス結合に異常が生じた場合，そのサブプロセス間結合の強制切断処理が行われる．レベル-n におけるリソース解放の終了後，事象 ＊＊(n) により，状態 F_2 からレベル-$(n-1)$ のオープン状態 (N_1) に移る．但し，レベル-1 およびイニシェーション中の強制切断処理は，それぞれ ＊＊(1) および ＊＊(0) 事象により，INITIAL-1 状態に戻るものとする (Fig. 4 の左図).

5. 階層構造によるプロセス結合

前章ではブロック単位にプロセス結合制御手順の状態遷移図を説明したが，それらのブロックを使って，階層構造をもつ全体的なプロセス結合の流れを表わしているのが，**Fig. 5**（次頁参照）である。図では，入出力記号の関係は省略してあるが，ブロック単位の状態遷移図で述べたものと同じである．また，タイム・アウト事象 (＊) も省略してあるが，その事象が発生したときは Fig. 5 の㊀の強制切断処理に移るものとする．

6. 実験と検討

本論文のオートマトン的手法をもちいたプロセス結

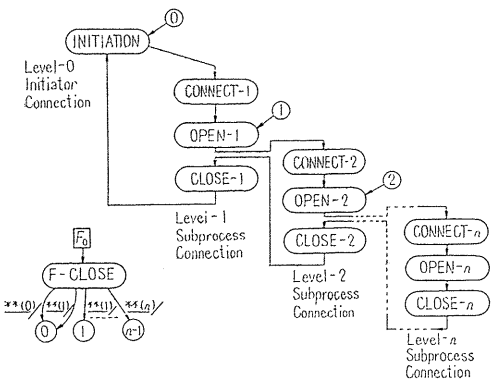


Fig. 5 Hierarchical structure of process connection.

合方式をもとに作成したのが，ALOHA-NCP 内の ARPA-プロトコルのプロセス結合制御プログラムである．このプログラムは，機種 HP 2115 (16 k 語，1 語 16 ビット) にアセンブラ言語を使って作成され，NCP のうち，そのプログラム自体の大きさは，共通サブルーチンを含めて 4 k 語程度である．

次に，実験システムをとおして，明らかになった点を述べる．

① プロセス結合制御は，イニシェーション・ブロックと各層のサブプロセス結合を生成する基本的ブロック (CONNECT-n, OPEN-n, CLOSE-n) の組み合わせからなっているので，結合論理の全体的な把握が容易かつ明解である．

② 階層構造をもつ結合手順は層単位の拡張や変更が比較的容易である．それ故に，新たなアプリケーション・プロトコルの結合制御に適応性がある．

③ レベル-n のサブプロセスはレベル-$(n-1)$ のサブプロセスの管理下にあるので，上位のサブプロセス間の会話をとおして，レベル-n のデバッグ中のサブプロセスに関するステータスまたはエラー情報のやりとりが可能である．それ故に，新たなプロセスまたはサブプロセスの開発やデバッグに適している．

④ プロセス結合制御手順の状態図を追うことにより，デバッグ中の細かなレベルのプログラム誤りを検出することができる．

⑤ ネットワークの仮想端末化を目的とした TELNET[6] プロトコル (本システムでは 2 ステップの階層) をのぞいて，ファイル転送プロトコル (3 ステップの階層をとる) のような他のプロトコルに共通していえる階層の深さは 3 ステップ程度であるので，プロセスを階層構造にするためのオーバヘッドは問題ないと思われるが，量的な検討は今後の問題としてのこされる．

7. 結 論

本稿では，NCP の中心をなすプロセス結合制御について述べている．特に，異なるホストのプロセス間結合で送受信するデータ情報のタイプの定義が，ネットワーク全体，ホスト間，サブプロセス間にわたって共通なものかにより，階層分けしたプロセス結合手法を示す．また，その制御手順はオートマトン的手法による状態遷移図で表現しており，プロセス結合制御プログラムは基本的なブロックから成り立っている．なお，**6.** に列挙した①から⑤の事柄は，プロセス結合制御プログラムの実際の作成から確証したものである．

以上，本稿で採用したプロセス間結合方式は，コンピュータ・ネットワークにおける NCP 設計のための一つの基本的方針であると思われる．

この研究は日米科学共同研究の一環として行った実験システムである．

参 考 文 献

1) C. S. Carr, S. D. Crocker and V. G. Cerf: HOST-HOST Communication Protocol in the ARPA Network, Proc. SJCC, Vol. 36, pp. 589～597 (1970)
2) The File Transfer Protocol, ARPA Network Information Center, No. 7813 (1971)
3) Graphics Protocol, ARPA Network Information Center, No. 15358 (1973)
4) Lincoln Speech Data Facility (LSDF), ARPA Network Information Center, No. 10917 (1972)
5) Remote Job Entry Protocol, ARPA Network Information Center, No. 12112 (1972)
6) Telnet Protocols, ARPA Network Information Center, No. 9348 (1972)
7) S. D. Crocker, J. F. Heafner, R. M. Metcalfe and J. B. Postel: Function-Oriented Protocols for the ARPA Computer Network, Proc. SJCC, Vol. 40, pp. 271～279 (1972)
8) 大泉，海老原，野口：汎太平洋教育研究用ネットワーク，情報処理，Vol. 16, No. 9, pp. 650～653 (1975)
9) Y. Ebihara and M. Willson: MENEHUNE ARPANET Driver, ALOHA System Internal Document, CCG/G-56 (1974)

(昭和 51 年 6 月 30 日受付)
(昭和 52 年 3 月 17 日再受付)