

複数計算機システムのTSSユーザ空間上のプロセスを 結ぶネットワーク・ユーティリティ

NETWORK UTILITY FOR INTERCONNECTION OF PROCESSES ON
TSS-USER SPACES OF ONE OR MORE COMPUTER SYSTEMS

中田 衛志[†] 田畑 孝一^{††} 大野 豊[†]
EIJI NAKATA KOICHI TABATA YUTAKA OHNO
[†]京都大学工学部 ^{††}京都大学情報処理教育センター

FACULTY OF ENGINEERING KYOTO UNIV.
EDUCATIONAL CENTER FOR INFORMATION PROCESSING KYOTO UNIV.

[1] はじめに

近年コンピュータ・ネットワークの技術は実用技術として、我々にとって非常に身近なものになってきた。すなわち、

- (1) コモン・キャリアによる新データ網のサービス開始。
- (2) 計算機メーカーによるネットワーク・アーキテクチャの提供。

の2つの事実が、コンピュータ・ネットワークの構成を飛躍的に容易にし、したがってその適用領域はますます広がってくるものと予想される。

しかし、現在提供されているネットワーク・アーキテクチャ、あるいはそれを利用するための通信アクセス方式は、いわばプロフェッショナル・ユーザによる専用のオンライン・システム開発のためのものであり、自分の研究の手段として計算機を利用するような半々人(カジュアル・ユーザ)が上記アクセス方式を用いてネットワークを構成するには不便な点が多い。

本稿では、我々が現在開発を進めているCNUU(Casual Network Use Utility)について報告する。¹⁾²⁾これは、計算機のカジュアル・ユーザが複数の計算機システムのTSSユーザ空間上のプログラムを結合して、彼個人の論理的なネットワークを比較的容易に開発することを可能にするためのシステムである。

[2] 基本的要求

TSSの利用者が、独力では、TSSユーザ空間上のプログラムを結合して協同処理を行わせようとすると、次のような問題点に直面するだろう。

- (P-1) 複数の端末を確保し、それらを同時に操作することが必要なこと。

通常、1つのTSSサービスを受けるためには1台の端末が必要である。したがって、複数のTSSユーザ空間上にまたがるシステムを開発する場合、TSS空間の数だけの端末を確保し、それらを同時に操作することが必要になる。

- (P-2) コンピュータ・ネットワークに関するかなりの知識が要求されること。

ネットワークを通じて他のホストの応用プロセスと通信するには、かなりの知識が要求される上、通信アクセス方式はホスト毎に異なっている場合が多い。

(P-3) プログラミング言語の選択に制限があること。

ネットワークの機能を使用するプログラム、あるいはサブルーチンは、多くの場合アセンブリ言語で記述しなければならない。これは(P-2)の問題とも相まって、カジュアル・ユーザにはかなりの負担を強いることになるだろう。

(P-4) テストが困難であること

複数の計算機にまたがるシステムのテストには面倒な点が多く、1つの計算機内でのプログラム毎のテストでは誤りが発見できないことがある。例えば、複数の計算機上のプロセス間で協同処理を行う場合、当然それらの間でデータ転送を伴う。この時、1つのプロセスがデータの転送手順を誤ると、デッドロック状態に陥ってしまうことがある。このような誤りの原因をつきとめるのは、かなり難しいであろう。

上記のような問題点の分析から、我々はCNUUに対する基本的な要求は次のようなものであると考えた。

(R-1) 単一ユーザ端末による操作。

複数のTSSユーザ空間にまたがるシステムを開発する場合、TSS空間の数だけの端末を確保し、それらを同時に操作することは非常に煩わしく、またそれが不可能なことが多い。したがって、一台の端末で複数のTSSサービスを同時に受けられる機能が要求される。

(R-2) 端末からの応用プロセスの管理。

端末からのコマンドにより、各TSS空間上の任意の応用プロセスを起動・停止できなければならない。また、任意の応用プロセスと端末とでメッセージの受け渡しが可能でなければならない。さらに、利用者のプログラムの開発、テストを容易にするために各応用プロセスの状態を端末から監視できることが望ましい。

(R-3) プロセス間協同処理の支援。

異なるTSSユーザ空間上のプロセスが協同処理を行う場合、プロセス同士は簡単な手続きで同期・データの送受信が行えることが望ましい。

(R-4) 利用者の負担を最小限に

利用者はプログラムをすべて高級言語で記述可能でなければならない。また利用する各ホストの差異を意識せずにプログラミングできることが望ましい。したがって、CNUUで用意する機能はすべて高級言語から利用できるような考慮されていなければならない。また、マシン・ディペンデントな部分CNUUでできるだけ吸収してやれることが望ましい。

[3] システム概要

CNUUの機能を実現するためにTSSユーザ空間上に用意するプログラムを、マネージャと呼ぶ。マネージャは利用者が使用する各TSS空間毎に1つずつ存在するが、ローカル・ホストのそれをマスタ・マネージャ、すべてのリモート・ホストに用意するマネージャをスレーブ・マネージャと呼ぶ。前節で述べた要求は次のようにして実現していく。

3.1 単一端末による操作

1台の端末で複数のTSSサービスを同時に受けられるようにするために、ローカル・ホスト

のTSSユーザ空間内にリモートTSSアクセス・プロセッサなるモジュールを用意する。

利用者はまず、ローカル・ホストのTSS空間内にマスタ・マネージャを起動する。そして特定のリモート・ホストのTSSサービスを受けることを要求する旨のコマンドをマスタ・マネージャに与えると、リモートTSSアクセス・プロセッサが起動され、そのホストのTSSサービスが利用可能となる。

3.2 スレーヴ・マネージャの起動とマネージャ間の結合

3.1の操作でアクセス可能になったTSSユーザ空間上にスレーヴ・マネージャを起動し、マスタ・マネージャとスレーヴ・マネージャを端末からのコマンドにより結合する。マネージャ間の結合はマスタ・マネージャを中心として星状に行われる。

3.3 応用プロセスの起動・停止・メッセージの授受

端末から、応用プロセスの起動・停止、あるいは応用プロセスとのメッセージの授受を要求すると、指定した応用プロセスがローカル・ホストにある場合にはマスタ・マネージャ自身がそれらの作業を行い、リモート・ホストにある場合にはスレーヴ・マネージャがマスタ・マネージャからの指令に基づいて必要な処理を行う。

3.4 プロセス間協同処理の支援

異なるTSSユーザ空間上の応用プロセスはCNUUで用意しているサブルーチンを呼び出すことにより、任意の他の応用プロセスと同期・データの送受信を行うことができる。この場合、応用プロセス間の通信はマネージャ間の通信で代行してやることになる。

3.5 システム開発の支援

マスタ・マネージャはリモートホストに生成されるものも含めて全応用プロセスの状態を監視している(後述)。そして、利用者は端末からコマンドを入力してマスタ・マネージャの保持している応用プロセスの状態に関する情報を調べることで、応用プロセスの処理の進行状況や、応用プロセスが利用者の意図した通りに動作しているかどうかを確認することができる。

また協同処理を行っている応用プロセスのグループがデッドロックの状態になったことを、マスタ・マネージャが検出すると、その旨が端末に出力される。

Fig-1は協同処理を行っている3つのプロセスがデッドロックの状態になった例である。プロセスAはプロセスBとの同期を要求して待ち状態に入り、プロセスBはプロセスCからデータを受信しようとして待ち状態に入り、プロセスCはプロセスAとの同期を要求して待ち状態に入っている。

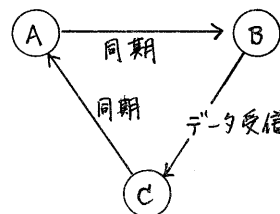


Fig-1 デッドロックの状態例

CNUUでは、応用プロセスは同時には1つの応用プロセスとしか協同処理(同期・データの送受信)を行えないため、協同処理を要求する関係性をアークとし、応用プロセスをノードとした有向グラフを考えると、1つのノードからは高々1本のアークしか出て行かない。ある応用プロセスが他の応用プロセスと何らかの協同処理を行うためにマネージャに対してサービス要求を出すと、マスタ・マネージャはこの有向グラフにアークを1本加え、アークを加えたことによりループができるかどうかを調べる。ループの存在が検出されると、デッドロックの発生を関係プロセスに通知するとともに端末にその旨を出力する。

【4】 情報の流れについて

4. 1 端末からやりとりされる情報

(1) システム・コマンドとシステム・レスポンス
マネージャに対して、応用プロセスの生成・停止、マネージャの保持している各種の情報の表示等の要求を行うコマンドがシステム・コマンドであり(Fig-2)、システム・コマンドへの応答がシステム・レスポンスである。なお、応用プロセスの消滅やデッドロックの発生をマネージャが検出した時に端末に出力されるメッセージもシステム・レスポンスに含める。

システム・コマンドのうち、ローカル・ホストでの応用プロセスの起動・停止、あるいは情報の表示等の要求はマスタ・マネージャが処理し、リモート・ホストでの応用プロセスの起動・停止等は該当ホストのスレーヴ・マネージャがマスタ・マネージャからの指令により処理する。

\$ACT	応用プロセスの起動
\$TERMINATE	応用プロセスの停止
\$WTP	待ち状態にある応用プロセスの表示
\$STS	指定した応用プロセスの状態表示
\$DLNK	通信を行っている応用プロセスの対
\$CON	マネージャ間の結合 表示
\$DISCON	マネージャ間の結合解除
\$CANP	プロセス・コマンドのキャンセル
\$HOST	ホスト番号とホスト名の表示
\$END	CNUIHに関する全ての処理の終了
#OPN	リモートTSSの要求
#host-name	中断していたTSSコマンドの入力再開

Fig-2 システム・コマンド一覧

(2) プロセス・コマンドとプロセス・リプライ

端末から応用プロセスに渡されるメッセージがプロセス・コマンドで、単一の応用プロセスを指定するものと、複数ホスト上の応用プロセスを指定して、それらに同時に与えるもの——複合コマンド——とがある。複合コマンドには各応用プロセスに別々のコマンドを与える場合と、全ての応用プロセスに同一のコマンド名を与える場合がある。プロセス・リプライは応用プロセスから端末へ渡されるメッセージである。対象となる応用プロセスがローカル・ホストに存在している場合、マスタ・マネージャが直接メッセージの集配を行い、リモート・ホストに存在する場合は該当ホストのスレーヴ・マネージャがマスタ・マネージャの指令に基づいて行う。

マネージャはプロセス・コマンド、リプライの集配を司るのみで、その内容は関知しない。利用者はどのような情報でもプロセス・コマンド、リプライにのせてよいわけだが、我々はこの機能を次のように利用されることを念頭に設計している。即ち、プロセス・コマンドは、端末から応用プロセスに対して、それが持つ機能の1つの実行を指示したり、協同処理をしようとしている応用プロセスのグループに対して、処理を始めるきっかけを与えたりする目的で使用する。プロセス・リプライは、与えられたプロセス・コマンドで指定された機能の実行に関する報告を行うために用いる。

(3) リモートTSSアクセス・プロセッサ・コマンドとレスポンス

リモート・ホストのTSSの機能を制御するいわゆるテルネット・コマンドと、リモートTSSアクセス・プロセッサ自身の制御を行うためのコマンドと、それに対する応答である。

(4) リモート・ホストTSSコマンド、レスポンス

(5) プログラムI/Oデータ

応用プログラム自身が通常の入出力手続きで端末と通信するデータである。

4. 2 応用プロセスとマネージャで交換される情報

(1) 同期リクエスト・同期コントロール

応用プロセスが、他の応用プロセスと同期をとるためにマネージャに出す要求が同期リクエストであり、それに対するマネージャからの応答が同期コントロールである。Fig-3は同期リク

エスト、同期コントロールの交換の様子を示している。 応用プロセス A, B は初期の独立に処理を行っている(①)。 応用プロセス A は CNU において用意しているサブルーチン (SYNCH) を呼び出すことで応用プロセス B と同期をとることをマネージャに要求し、マネージャからの応答を待つ (SYNCH の中で) 待ち状態に入る(②)。 次に応用プロセス B が同様の手続きで応用プロセス A との同期を要求して待ち状態に入る(③)。 マネージャはこの時点で両プロセスの同期がとれたことを認識し、両者の同期コードを交換してやる。 マネージャからの同期コントロール、即ち相手プロセスの同期コードを受けると両プロセスの待ち状態はとける(④)。

(2) 端末専有要求・端末専有許可・端末解放要求

応用プロセスの間で端末の競合をさせないよう、プロセス間で同期をとる必要がある場合がある。 このような場合の便宜をはかるために、マネージャは端末の要求待ち行列の管理機能を有している。 応用プロセスが端末専有要求を出した際、マネージャは端末の使用状況を調べ、使用可能なら端末専有権を要求元の応用プロセスに与える。 また、既に端末が他の応用プロセスによって専有されているときは、現在専有しているプロセスが端末解放要求を出すか消滅するまで要求元プロセスを待ち状態におく。

(3) 通信路確立要求・閉鎖要求と応答

ある応用プロセスが他の応用プロセスと通信するためにマネージャに通信路確立要求を出すと、そのプロセスは待ち状態になる。 相手プロセスが先に要求を出した応用プロセスを指定して通信路確立要求を出し、互いに通信し合っている応用プロセスの対が確定すると、マネージャは両プロセスに対し通信路が確立した旨通知し、両プロセスの待ち状態はとける。 なお、応用プロセスは一時には1つの応用プロセスとしか通信路を持つことができない。

(4) データの送信・受信要求と完了通知

(3) で確立した仮想的な通信路を通じて、応用プロセス同士は通信を行うことができる。 送信側のプロセスは送信データのバッファのアドレスを指定して、マネージャにデータ送信要求を出し待ち状態に入る。 受信側プロセスがデータ受信用のバッファ・アドレスを指定してデータ受信要求を出すと、マネージャは送信側プロセスのバッファから受信側プロセスの

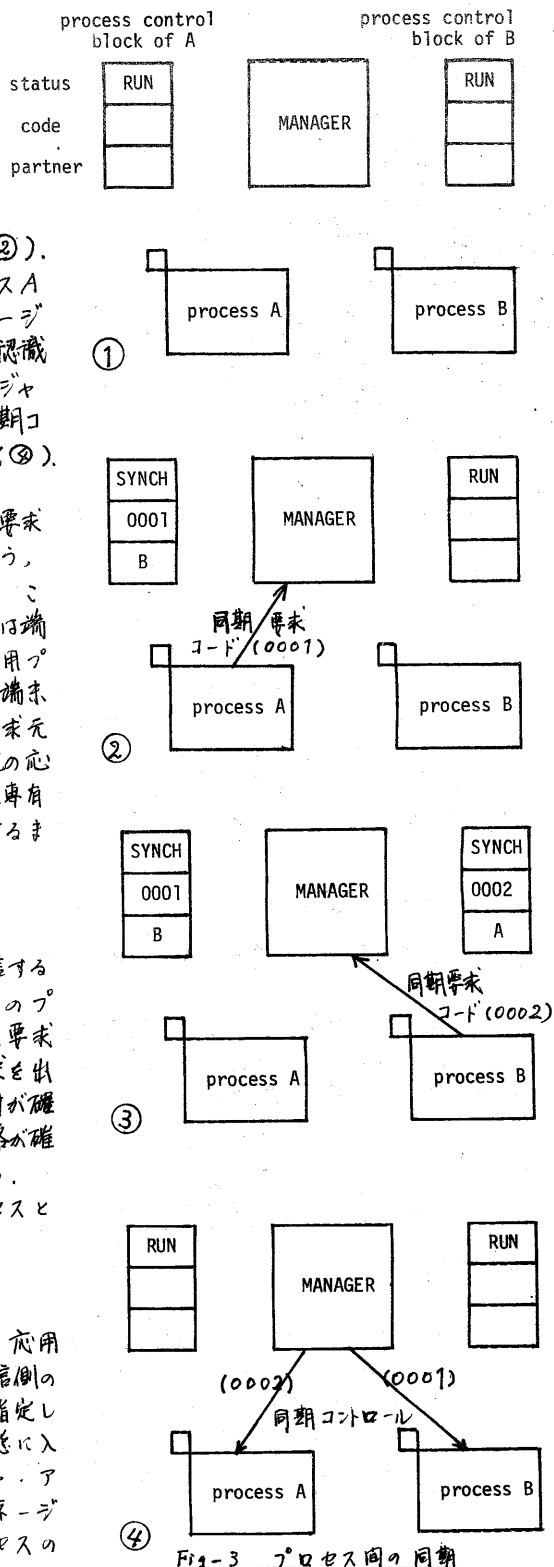


Fig-3 プロセス間の同期

バッファにデータを転送し、転送終了通知を両プロセスに出す。終了通知を受けると両プロセスの待ち状態はとける。

(5) プロセス・コマンドとプロセス・リプライ。

既に述べたように、端末と応用プロセスが通信したい場合、プロセス・コマンド・リプライの機能を使用できる。なお、端末から入力されたプロセス・コマンドの場合、対象となるすべての応用プロセスからプロセス・コマンド受信要求が来そうまで、先に要求を出した応用プロセスは待ち状態におかれる。

4.3 TSSユーザ空間の間に張られる3対のリンクについて

ローカル・ホストのTSSユーザ空間と1つのリモート・ホストのTSSユーザ空間の間には次の3対のリンクが張られる。

(1-1) リモート・ホストのTSSサービスを受けるために使用するリンクで、4.1の(3)~(5)のデータがこの上を流れる。このリンク上を流れるデータは全て文字型(NVTコード)である。

(1-2) 4.2の(2)で述べたデータ、即ち応用プロセス間でやりとりされるデータをのせるためのリンクで、データ・タイプは任意である。

(1-3) マネージャ間でやりとりされる情報をのせるためのリンクである。このリンクを通じてスレーヴ・マネージャからマスタ・マネージャへ送られる情報には、リモート・ホストに生成された応用プロセスのマネージャに対する種々のサービス要求、スレーヴ・マネージャの検出した応用プロセスの終了状態等があり、マスタ・マネージャからスレーヴ・マネージャへ送られる情報には、応用プロセスの生成・停止指令、応用プロセスの出したサービス要求に対する応答等がある。この上を流れるデータをすべて文字型である。

[5] システム構成

本節ではCNUUを構成する主要なサブモジュールについて述べる(Fig-4参照)。

5.1 マスタ・マネージャ(MASTMGR)

MASTMGRはCNUUの中核となるモジュールで、ローカル・ホストのTSSユーザ空間に1つ存在し、以下の役割を果たす。すなわち、

- (i) ローカル・ホストにCNUUで用意するすべてのプロセスを生成し、
- (ii) 端末から入力されるシステム・コマンド、プロセス・コマンドを処理し、
- (iii) 応用プロセスの状態の監視・応用プロセスからの要求に対するサービスを行う。

なお(ii)(iii)を行う時に必要に応じてSLAVMGRと4.3で述べた(1-3)のリンクを用いて通信を行う。

5.2 スレーヴ・マネージャ(SLAVMGR)

このモジュールは各リモート・ホストTSSユーザ空間毎に1つずつ用意されるもので、

- (i) そのTSS空間内にCNUUで用意するすべてのプロセスを生成し、
- (ii) MASTMGRからの指令に基づき応用プロセスを生成停止し、
- (iii) 応用プロセスからの要求をMASTMGRに伝え、それに対するMASTMGRからの応答を応用プロセスに伝える。

5.3 リモートTSSアクセス・プロセッサ (REMTACP)

リモート・ホストのTSSサービスを受けられるようにするために、ローカル・ホストのTSS空間内に用意されるモジュールで、1つのリモート・ホストに対応して1つのプロセスが生成される。REMTACPは4.3の(1-1)のリンクを管理しており、端末とリモート・ホストのTSS (のテルネット・サーバ) との間のメッセージの仲介を行うほか、REMTACP コマンドの解釈・実行を行う。これは、いわゆるテルネット・ユーザと同じ働きをするが、1台の端末で同時に複数のTSSサービスを受けられるようにするために、テルネット・ユーザとは若干構造が異なり、また同じ理由からリエントラントな手続きの形をしている。

5.4 応用プロセス監視モジュール (APMONTR)

MASTMGR (またはSLAVMGR) からの指令により応用プロセスを生成し、その状態を監視する。応用プロセスが終了したことを検出すると、その旨をMASTMGR (またはSLAVMGR) に通知する。

5.5 応用プロセス間通信代行モジュール (IPCSBST)

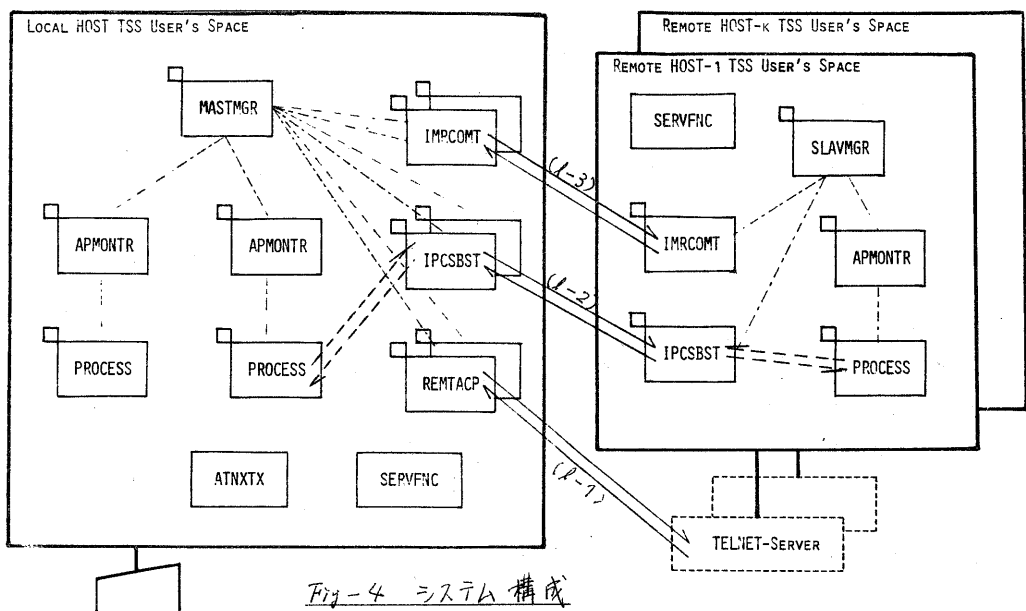
4.3の(1-2)のリンクを管理しており、MASTMGRまたはSLAVMGRからの指令により応用プロセス間の通信を代行してやる。ローカル・ホストには結合されているリモートホストの数だけ生成され、リモート・ホストには1つだけ生成される。

5.6 マネージャ間通信管理モジュール (IMRCOMT)

MASTMGRとSLAVMGRの間の通信を管理するもので、4.3の(1-3)のリンクを管理している。IPCSBSTと同じくローカル・ホストには結合されているリモート・ホストの数だけ生成され、リモート・ホストには1つ生成される。

5.7 CNUサービス・ファンクション (SERVFNC)

応用プロセスがマネージャにサービスを要求し、その結果を得るまでの一連の手続きは非同期



応用プロセスはマネージャに対して何らかのサービスを要求するとき、これらの中の適当なサブルーチンと呼ぶ。これらのサブルーチンのうち、 $NVT \leftrightarrow EBC$ 変換コード変換ルーチン以外は、マネージャと同期通信を行ってマネージャからサービスを受けるようになっている。いずれにせよ、そのサブルーチンから応用プロセスに制御が戻ってくる。

EBCDNTV (EBCDIC → NVT コード変換)
 NVTEBCD (NVT → EBCDIC " ")
 RCVCMD (プロセス・コマンド受信要求)
 REPLY (プロセス・リプライ送信要求)
 SYNCHRO (他の応用プロセスとの同期)
 OPNCHNL (" " " " " " " " 仮想通信路確立)
 CLSCHNL (" " " " " " " " " " 閉鎖)
 SNDDAT (データ送信)
 RCVDAT (データ受信)
 GETTRML (端末再帰要求)
 RELTRML (端末解放要求)

Fig-5 CNUU サービス・ファンクション

利用者が端末から入力するコマンドの大部分は非同期に入力可能でなければならぬ。また、端末からコマンドを受取り処理するプロセス(MASTMGR, REMTACP)は端末からのコマンド

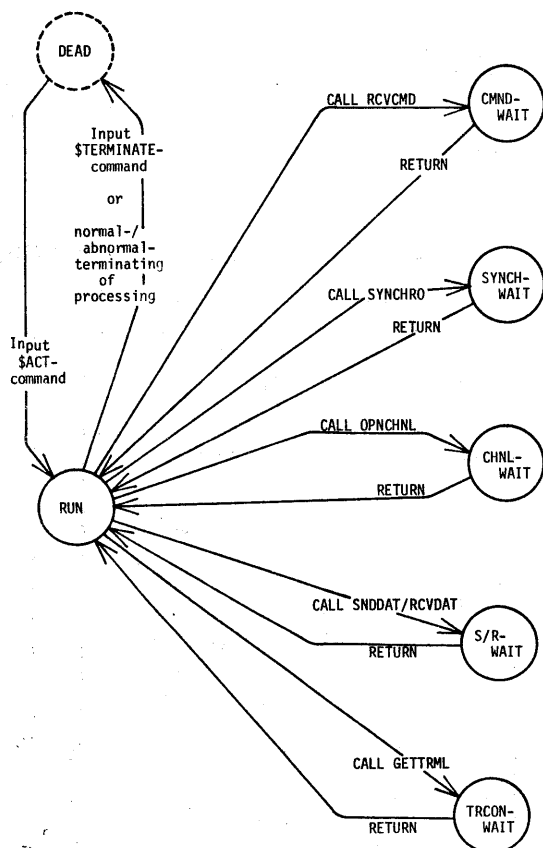


Fig-6 応用プロセスの状態遷移

ドのみならず、他の非同期に発生する事象——
 例えば応用プロセスからのサービス要求——に
 も対処できなければならない。したがって、
 前述の各プロセスは割込みを受ける形で要求を
 受付けるようになっていゝる。利用者が端末か
 らコマンドを非同期に入力する際は、MASTER
 (またはREMITACP) に対して割込
 む(アテンション・キーを押す)ことでその意
 志を伝える。

ATNX TXはこの処理のために用意されているトラップ・ルーチンで、アテンション割込みが発生すると現在のモードによってMASTER MODE, または通常のREMTACDに利用者のコマンド入力意志を伝える。

「6」 応用プロセスの状態遷移

マネージャは応用プロセスからの要求に飛
びマサービスを行ったり、応用プロセス間での
協同処理の過程でデッドロックが生じた時にそ
れを検出したりする目的で、全応用プロセスの
状態を監視している。Fig-6はマネージャ
が通知する応用プロセスの状態遷移図である。
応用プロセスの状態遷移は、応用プロセスがマ
ネージャに何らかのサービスを要求した時に起
きる場合が殆どである。

利用者はコマンドによってマシナリウェアの保

持っている応用プロセスの状態に関する情報を調べることにより、自分の構築したシステムの処理の進行状況や、各応用プロセスが自分の意図した通りに動作しているかどうかを検査することができる。

[7] 例題

CNUUを用いて構成する協同処理システムの例をFig-7に示す。これはリモート・ホストからローカル・ホストへあるファイルを転送するシステムで、リモート・ホスト側の応用プログラムがSENDER、ローカル・ホスト側の応用プログラムがRECVRである。なお、リモートホストのNCPアドレスは(1234)₁₆、ローカル・ホストのそれは(5678)₁₆とする。また利用者が端末から入力する部分をFig-7では下線で示している。

まず、端末からTSSセッションを開き(①)、ローカル・ホストに用意する応用プログラム、即ちRECVRに送られてくるデータを収めるべきファイル名を与え、MASTMGRRと結合する(②)。そしてMASTMGRを起動し(③)、アテンションを入力してコマンドの入力意志をMASTMGRに伝える(④、なおここではアテンションを④で示す)。次にリモート・ホストのTSSサービスを要求する。ここでABCDは利用者が今から使用しようとしているリモート・ホストにつけた識別子である(⑤)。これでこのホストのTSSサービスが受けられるようにする。なのでTSSのセッションを開き(⑥)、送るべきファイル名をSENDERに与えてSLAVMGRと結合し(⑦)、SLAVMGRを起動する(⑧)。SLAVMGRは起動されるとMASTMGRの存在するホスト、即ちローカル・ホストのNCPアドレスを聞いてくるので(⑨)、これを入力する(⑩)。一応、このホストへのTSSコマンドの入力は一段落したので、システム・コマンドやプロセス・コマンドが入力可能なモードに切替えるコマンド#OVERを入力する(⑪)。次にMASTMGRとSLAVMGRを結合し(⑫)、応用プロセスをローカル・ホストとリモート・ホストに生成する(⑬、⑭)。(⑮)以下5行は今起動した応用プロセスSENDER、RECVRにプロセス・コマンドを与えている部分である。そして⑯、⑰は応用プロセスからメッセージを介して送られてきたプロセス・リプライである。最後の2行は応用プロセスSENDER、RECVR

```
LOGON      - - - - - ①
/* set file name & link RECVR to MASTMGR */ ②
CALL MASTMGR ③
@ ④
CNUU>PLEASE INPUT COMMAND
#OPN 1234 ABCD ⑤
LOGON      - - - - - ⑥
/* set file name & link SENDER to SLAVMGR */ ⑦
CALL SLAVMGR ⑧
SLAVMGR>KEY IN ADDRESS OF MASTMGR ⑨
5678 ⑩
@
ABCD>PLEASE INPUT REMTACP-COMMAND
@
#OVER ⑪
@
CNUU>PLEASE INPUT COMMAND
$CON ⑫
@
CNUU>PLEASE INPUT COMMAND
$ACT RECVR ⑬
CNUU>RECVR ACTIVATED IN LOCAL HOST
@
CNUU>PLEASE INPUT COMMAND
$ACT SENDER ABCD ⑭
CNUU>SENDER ACTIVATED IN ABCD
@
*<SENDER RECVR> ⑮
CNUU>PLEASE INPUT PROCESS COMMAND FOR SENDER
SEND
CNUU>PLEASE INPUT PROCESS COMMAND FOR RECVR
RECEIVE
REPLY SENDER > EOF ⑯
REPLY RECVR > END DATA ⑰
CNUU>SENDER HAS TERMINATED ⑱
CNUU>RECVR HAS TERMINATED ⑲
```

Fig-7 CNUU使用例

が終了したことをマネージャが検出し、端末に出力してきたものである。

〔8〕 おわりに

計算機のカジュアル・ユーザが、複数の計算機システムのTSSユーザ空間上のプログラムを結合して、彼個人の論理的なネットワークを比較的容易に構成できるよう支援してやるユーティリティ — CNUU — について述べてきた。

一般に、複数の計算機にまたがるシステムを開発するときには多大の労力が必要で、気軽にその開発を行うことはできない。しかも、特に実験的なシステムを開発する際には、できあがるシステムの効率や信頼性よりも、システム開発の容易さ、あるいは変更・拡張の容易さに重点がおかれるであろう。CNUUを用いれば、TSSを使い慣れている利用者であれば、気軽にネットワークを構成でき、このような要求に十分耐えることができると考えている。

本システムは現在N-1ネットワーク上に京都大学大型計算機センタのM200を用いて開発中であるが、他のネットワーク・アーキテクチャにおいても次の2つの手段が提供されれば、その上に同様の機能が実現できる。

- (1) ローカル・ホストのTSSユーザ空間から他のホストのTSSサービスを受ける手段。
- (2) 異なるホストのTSSユーザ空間上のプロセスの間に論理通信路を設定する手段。

最後に、我々の研究室では現在、分散型データベース管理システムの開発が計画されているが、その開発にあたりCNUUを用いることが予定されている。

〔謝辞〕

日頃お世話になっている京都大学大型計算機センタの桶谷猪久氏を始めとする技官の方々、今井恒雄氏を始めとする富士通SEの方々に深く感謝します。

〔参考文献〕

- 1) 山本彰, 中田, 田畑, 大野. "複数計算機システムのTSSユーザ空間上のプロセスを結合管理システム" 昭和54年度情報処理学会第20回全国大学講演論文集 pp. 457-458 (1979)
- 2) 田畑. "コンピュータ・ネットワークの新技術動向" 情報処理学会講習会「コンピュータネットワーク技術とその応用」(1979)
- 3) Yamamoto. "Management System for Interconnection of Processes on TSS-User Spaces of One or More Computer Systems" Master Thesis of Kyoto Univ. (1978)