「TSインプリメンタからみた INWG ENP to ENP プロトコル」

(財)日本情報処理開発協会　東　吉郎，伊藤哲史

1. はじめに

本論文は「ネットワーク結合のためのENP to ENPプロトコルの提案（IFIP WG 6.1，INWG General Note #96）」に「INWG 96に対する修正（同 Protocol Note #49）」の修正を加えたものを対象とし，本プロトコルをインプリメントする立場からみた場合の問題点について述べるものである。

本論文で問題点としてとらえているものは次の4つに分類できる。

① プロトコル上，矛盾あるいはあいまいな点。
② プロトコルとしては矛盾ではないが，インプリメントが不可能あるいは極めて難しい点。
③ インプリメントする上で注意を要する点。
④ インプリメントされたものを使用するユーザの立場からみて使いづらいと思われる点。

以下にこれらの観点から各問題点の指摘を行なうと同時に，可能な限り解決方法案を提示する。

なお，プロトコルの検討にあたっては，本プロトコルの下位に属するサブネットの機能を次のように想定した。

サブネットの想定機能

① パケット交換網である。
② パケットの許容最大テキスト長は238オクテット以上である。
③ パケットは発信側のホストが投入した順番で受信側のホストに渡される保障は無い。
④ サブネット内でのパケット紛失率は10^{-5}～10^{-8}であり，もし紛失しても，そのパケットの発信側，受信側のいずれにも紛失したことは通知されない。
⑤ バイナリデータが送れる（データの透過的伝送が可能である）。

上記のように，サブネットの機能としては最低限のものを想定した理由は，本プロトコルがあらゆるネットワークで利用できる標準的なENP to ENPプロトコルとして提案されているからである。

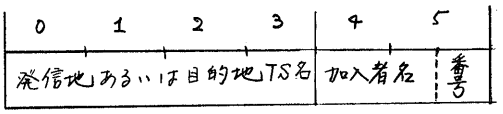
もしサブネットに高度な機能を要求するならば，本プロトコルを利用できないネットワークが生じることも考えられ，本プロトコルの主旨に反する結果となる。

2. プロトコルの概要

本プロトコルは，サブネットからみて一つの実体として認識できるリソースの集合である仮想ホスト（Virtual HOST）内のプロセス（その実体はJOB，ターミナル etc）間でのプロセス間通信を可能にするものである。各仮想ホストは1つのTS（Transport Station）を持ち，TSが前記プロセスとサブネットの間に介在し，本プロトコルを処理し，プロセス間通信機能を可能にする。

プロセス間を流れるメッセージをレターと呼び，レターの発信側と受信側を明らかにするためポートという名前空間が導入されている。ポートは図－1に示す

ようにTS名，加入者名，番号の6オクテットからなるポート・アドレスという名前がつけられ，ネットワーク内で一意に識別される。



図－1　ポート・アドレス

プロセス間通信は，発信ポートから受信ポートに向ってレターを送ることによって行なわれる。ポートというのは電話でいえば電話番号にあたるもので電話番号が大代表を持っているのと同じように，1つのポートで同時に複数個のプロセスと交信することもできるように設定されている。

プロセス間通信は，2つのモードで動作することができる。1つはレター・グラム・モードであり，もう1つは結合（liaison）モードである。

レター・グラム・モードは，郵便システムの手紙の配達にたとえることができるもので，データ交換網がデータ・グラムのサービスをおこなうことを想定し，これをプロセス間通信のレベルでサービスしようとするものである。

結合モードは，バーチャル・コールに相当するものであって，レターの伝送に先だって初期設定コマンドが交換される。初期設定では次のことが確認され，同意が得られれば双方向のレターの伝送が可能となる。

① 結合の両端のプロセスがアクティブであることの確認。
② オペレーションにはいるサービス種類の同意。
③ 付随したパラメータの初期化。

また，このモードでは種々の付加サービス（エラーコントロール，フローコントロール etc）が許され，エラーコントロールが採用されるとレターは発信側の投入した順番に受信側プロセスに渡される。さらに割り込み信号がこのモードでは送ることが可能である。

レターの最大長は25,000オクテットを越え，最後のものを除いて216オクテットのフラグメントという単位に分割され送られる。TSはプロセスから受け取ったレターをフラグメントに分割し（フラグメンテーション），パケットに乗せてサブネットへ送り出したり，サブネットより受け取ったパケットからフラグメントを取りだし，レターに組み立て（リアセンブリ）受信側プロセスに渡す作業を行なう。

本プロトコルでは図－2で示される10種類のコマンドによってTS間の会話が行なわれ，前記の機能が達成される。

TSの機能構成を図－3に示す。TSの機能は大きく2つに分かれる。※1はENDtoENDプロトコルインタープリータ部であり，※2はインタフェース部である。

(1) ENDtoENDプロトコルインタープリータ部

この部分はさらに次の4機能部分に分かれる。

① レター・グラム・モードのサービスをつかさどる部分。
② 結合モードサービスの基本機能をつかさどる部分。
③ 結合モードにおける種々のオプションサービス（エラー制御 etc）をつかさどる部分。

④フラグメントのリアセンブリ，フラグメンテーションをつかさどる部分。

(2) インタフェース部

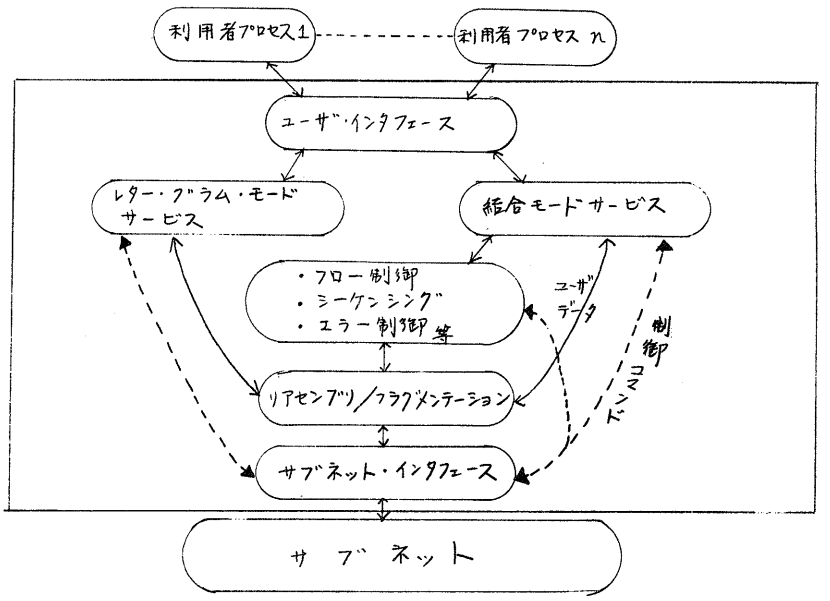
この部分はさらに次の2機能部分に分かれる。

①利用者プロセスとのインタフェースをつかさどり，使いやすいインタフェースでレター・グラム・モード，結合モードのレター交換機能を提供する部分（ユーザ・インタフェース）。

②サブネットとのインタフェースをつかさどり，フラグメントをパケット化してサブネットに送り出したり，サブネットより受け取ったパケットよりフラグメントを取り出す等の機能を果たす部分（サブネット・インタフェース）。

コマンド ＼ ビット	0	1	2	3	4	5	6	7	意味
LG-LT	R	0	0	1	0	0	0	0	レター・グラム・モードにおいて，レターのフラグメントを送るために使用される。
LG-ACK	0	0	0	1	0	0	0	1	レター・グラム・モードにおいて，レターのAckを送るために使用される。
LG-NACK	0	0	0	1	0	0	1	0	レター・グラム・モードにおいて，否定のAckを送るために使用される。
LI-LT	R	0	0	0	0	0	0	0	結合モードにおいて，レターのフラグメント及び逆方向のAckを送るために使用される。
LI-ACK	R	0	0	0	0	0	0	1	結合モードにおいて，Ackを送るために使用される。
LI-INIT	0	0	0	0	0	0	1	1	結合モードの初期設定に使用される。
LI-TERM	0	0	0	0	0	1	0	0	結合モードを終了するために使用される。
LI-PURG	R	0	0	0	0	1	0	1	結合モードにおいて，割り込みを送り，残っているレターの消去のために使用される。
LI-ERR	0	0	0	0	0	1	1	0	結合モードにおいて，エラー信号を送るために使用される。
LI-NACK	R	0	0	0	0	0	1	0	結合モードにおいて，否定のAckを送るために使用される。

図－2 Transport コマンドの種類



図－3
TS 機能構成

3. プロトコルの問題点

以下にプロトコルの問題点を逐次項目をあげながら述べる。

(1) レター長

レター長は最大25,000オクテット以上となっているが，次の問題点が考えられる。

① 25Kオクテットを越えるレターの取り扱いは，事実上バッファサイズ等の関係でTSのインプリメントが困難になると考えられる。

② タイム・シェアリング・システムのみをサポートするTSでは，25Kものレターを取り扱う必要はまずない。通常数百オクテット以下のものがほとんどである。

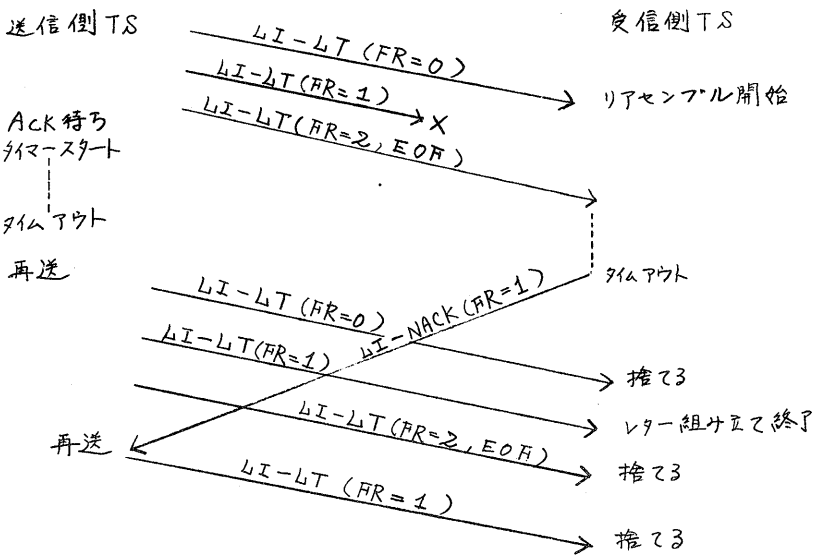
したがって，プロトコルでは各TSがサポートすべきレター長の範囲（ミニマムおよびマキシマム）のみを定め，インプリメント上の最大レター長はその範囲内で各TSインプリメンタにまかせるのが妥当であろう。

(2) LG－NACK，LE－NACK

プロトコルでは両コマンドともコードはあるが，そのコマンドの規定はされておらず付記として次のように記されている。

「LE－NACKは再送の効率を良くするためのものであり，受信側が特定のフラグメントに対し，否定のAckを返し，送信側が特定のフラグメントを再送するのを許すような方法で定義されるべきである。」

しかしLG－NACK，LE－NACKの使用が常に効果があるかは疑わしい。その理由を図－4の例で説明する。図は結合モードで送信TSが3つのフラグメントからなるレターを送り，第2番目のフラグメントが消えた場合を示している。このとき，受信側TSはリアセンブリタイムアウトによりフラグメント1が消えたことを知り，LE－NACKで送信側TSに通知している。他方，送信側TSもAck待ちタイマーをかけているので，それがたまたま受信側からのLE－NACKを受けることにより送信側がフラグメント1を再び再送するということが起こっている。



図－4　LE－NACKでの再送例

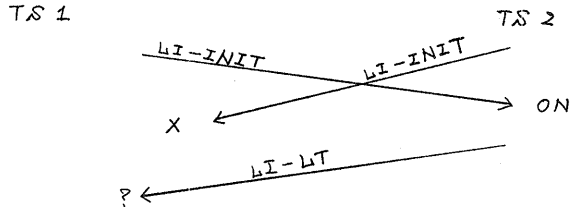
この図で示した例の場合はLI－NACKは全く役立っておらず，逆に悪影響を与えている。このような事が起こる可能性は十分あると考えられる。

(3) LI－INITのcredit

初期設定を行なう為のLI－INITにはフロー制御が指定されたときcreditを便乗させることができる為，次の問題点が考えられる。

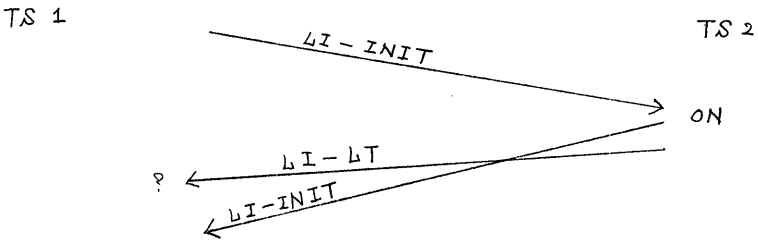
問題点

①図－5で示すように，TS2よりのLI－INITが消えた場合，TS1がONでないときにレターが到着する。



図－5　TS2よりのLI－INIT消滅の場合

②サブネットでは送信側TSが投入した順序で，受信側TSにパケットが渡される保証は無いため，図－6で示すように，TS2よりのLI－INITとLI－LTの順序が入れ替ったときに，①と同様TS1がONでないときにレターが到着する。



図－6　TS2よりのLI－LTとLI－INITが発信順序とは逆に到着した場合

上記①②が起ったとき，②の場合はLI－LT受信後，しばらく待てばONになるが，TS1にとってはLI－LTを受信した段階では①なのか②なのか判らない。

上記問題の解決法としては次の2案が考えられる。

① LI－INITにcreditを便乗させず，creditはLI－Ackでやりとりする。

② 結合がONでないときにレターを受け取ったならば直ちにエラーとするのではなく，一定時間LI－INITが到着するのを待つ。

しかし，上記問題点はフロー制御を指定された結合モードのみで発生するのではなく，結合モードなら必ず発生の可能性がある。したがって解決法としては②がよりベターと考える。

(4) 結合モード初期設定時のLI－ACK

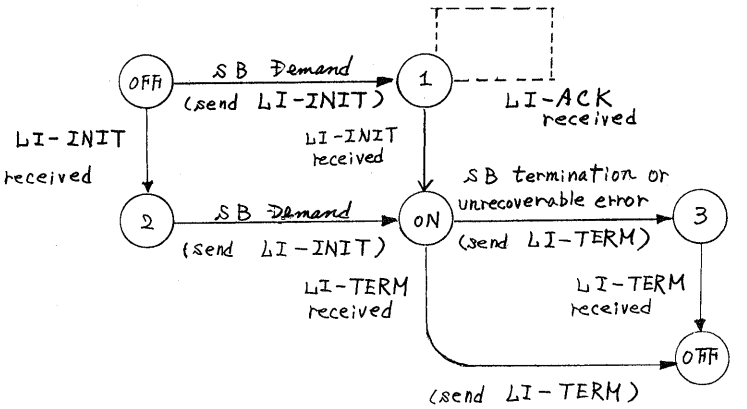
結合モードの初期化と終了の図（図－7）において，④のところでLI－AC

Kを受けつけ可能にしておかねばならない　（点線で示した部分を加える）。

なぜならばLI－ACKがLI－ENITを途中で追い越す場合があるからである。また，このようなLI－ACKを受信したTSは次のように処理するのがよいと考える。

① LI－ACKのフロー制御のためのcreditは有効とする。

② LI－ACKの確認応答勧誘のRビットがONになっていても無視する。



図－7
結合モードの初期化と終了の図
（一部抜粋）

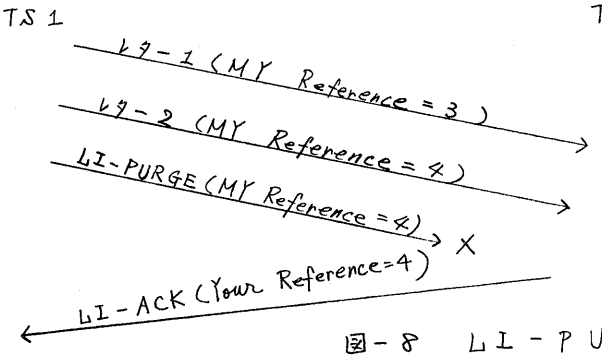
(5) LI－PURGEのMY Reference

結合モードにおいて割り込み信号を送るためのLI－PURGEには，それが確実に届くことを保証するため，結合のシーケンス番号（レターに付加される）をMY Referenceに持つことになっている。また，INWG Protocol Note #49によると，この番号はLI－PURGEを送る以前に送られた最後のレターのリファレンスを示すことになっている。しかし，これではLI－PURGEを確実に届けるためには何の役にもたたない。例えば図－8において，LI－PURGEが途中で無くなっても，送信側および受信側のいずれのTSにもそのことはわからない。したがってLI－PURGEのMY Referenceは次のようにつけるべきであると考える。

解決案

LI－PURGEもレターの1種と見なす。すなわちLI－PURGEのMY Referenceは自分自身に付けられたシーケンス番号とする。

例えば図－7のLI－PURGEではMY Reference＝5となる。



図－8　LI－PURGE紛失例

(6) LI-PURGEの機能

LI-PURGEは受信側プロセスに「割り込み信号」を伝えること以外に，LI-PURGEまでの未処理レターを消すことになっている。そこで問題となるのは未処理の範囲がどこまでであるかということである。範囲の区分としては次のものが考えられる。

① TSがまだAckを返していないもの。
② ①及び，まだ受信側プロセスに渡さずTS内に保持しているもの。
③ ②及び，TSの手を離れているが，まだ受信側プロセスが処理していないもの。

上記3区分のうち③はTSレベルでは判定が困難である。したがって未処理の範囲としては①あるいは②が考えられるが，次に述べる理由により①がベターだと考える。

理由

①のように定義することにより，逆に送信側はLI-PURGEを出す前にAckを受け取ったレターは消えることがないという保証を得られる。

一方，①のように定義すると，受信側プロセスにとっては都合の悪いことが生じる可能性がある。図-9において，受信プロセスがLI-PURGEを受けた段階で，それ以前のレター（レター（MR=1），----レター（MR=n））は不要となり，次に必要とするレターはレター（MR=n+2）であると仮定する。このとき，受信プロセスはレター受信コマンドをn+1回出し，そのうちn回は読みすてることとなり，nが大きくなると大変な無駄となる。また，レター内容を見るだけで必要なレターとそれ以外の識別ができないことも十分あり得ると考えられる。かかる場合にはTSよりLI-PURGEの次のレターを識別するための情報を通知してもらう必要がある。

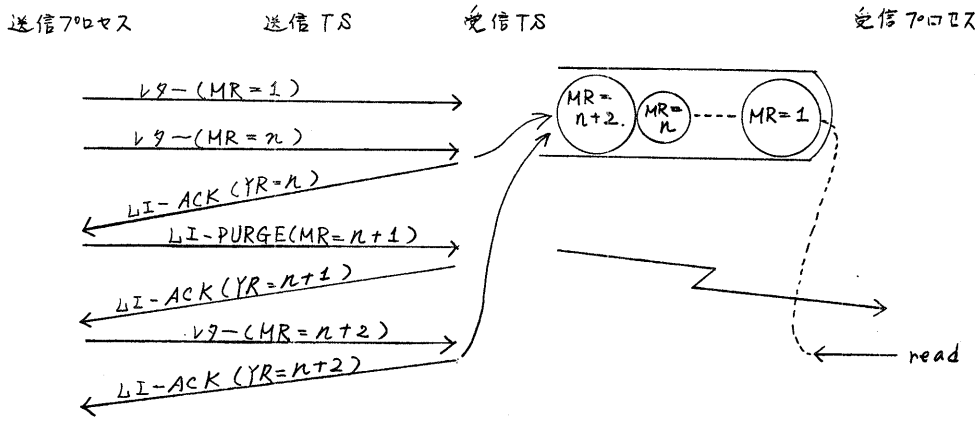


図-9 LI-RURGEと受信プロセス

上記，受信側プロセスにとっての問題点は次のようにすることで解決できる。

解決案

① ユーザ・インタフェース・コマンドとして，TSが保持しているレターのうち指定番号までのものを消す機能を持つものを設ける。
② 受信プロセスに割り込み信号を通知するとき，LI-PURGEのMY Reference も通知する。

(10) タイマー

プロトコルに現われているタイマーとしては次のものがある。

① 結合モードの初期化，終了時タイマー。

② 受信側TSのリアセンブリー放棄タイマー（NACKのトリガーでもある）。

③ 送信側のAck待ちタイマー。

しかしながら次の2つのタイマーが更に必要と考えられる。

④ フロー制御オプション時に，レター送信のためのcreditが紛失した場合のタイマー。

このタイマーはcredit送信のためのムエーACKが紛失したときに生じるデッドロックを回避するため必須のものである。このタイマーの実現法としては次の方法が考えられる。

㋑. レター送信側でcreditが無くなったときにセットし，タイムアウトになるとcredit送信勧誘のムエーACKを出すためのもの。

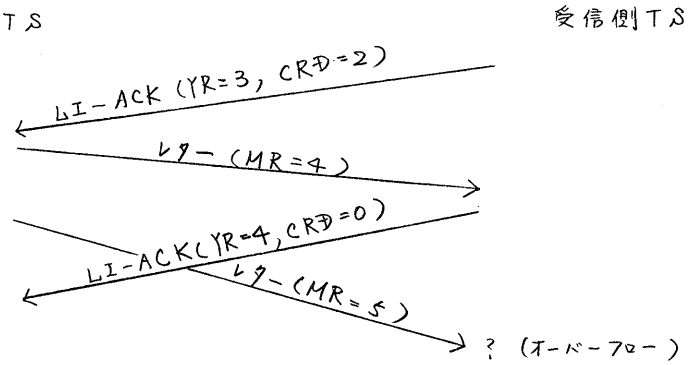
㋺. レター受信側にcreditを送っているにもかかわらずレターが送られてこない時に，credit送信のコマンドが紛失したと考え再送を行なうきっかけとして用いるもの。

⑤ エラー制御オプション時，受信側がレターを受け取ってからAckを返すまでに許されるタイム間隔を計るタイマー。

このタイマーは必須ではないが，効率を上げる為にはあった方がよいと思う。なぜならば，Ackはレターの送信動作をさまたげない限り，まとめて行なった方が効率的であるからである。

(11) credit

結合モードにおいては受信側TSから送信側TSに対してYour Reference＋CRD－NBという形でフローコントロールの為のcreditを与えているが，より小さいcreditを与え直すことにより，一たん与えたcreditを取り返すことができる。その場合，送信側よりのレターとcreditのすれ違いにより，あたかも送信側TSが間違ったようにみえることがある。（図－10参照）。したがってcreditを取り返すようなTSをインプリメントする場合，かかる場合に送信側エラーと見なさないようインプリメントしなければならないと考える。



図－10　creditの取り返し

(12) ユーザ・インタフェース

プロトコルではTSユーザ・インタフェース・コマンドとして次のものがあげてある。

① OPEN-PT：ポートをアクティブにする。
② RECV-LG：レター・グラム・モードで遠隔地ポートからレターを受け取る。
③ SEND-LG：レター・グラム・モードでレターを送る。
④ CLSE-PT：ポートをアクティブでなくする。
⑤ OPEN-LI：結合を初期設定する。
⑥ RECV-LI：結合モードでレターを受け取る。
⑦ SEND-LI：結合モードでレターを送る。
⑧ CLSE-LI：結合を終了処理する。

しかし，ユーザに使いやすい機能を提供するには上記以外に次のコマンドを設ける必要があると考える。

⑨ POST-LI：結合モードで他系へ割り込み信号を送る。
⑩ ENTR-LI：結合モードで他系からの割り込み信号を受け取る。
⑪ WAIT：SEND-LI，RECV-LG等のコマンド終了を待つ。
⑫ GET-PN：TSが保持しているポートを動的に割り当ててもらう。このコマンドはOPEN-PTの機能も含む。
⑬ FRE-PN：GET-PNで割り当てられたポートをTSに返却する。このコマンドはCLSE-PTの機能も含む。
⑭ LSTN-LI：現在アクティブなポートに対する他系よりの結合モード開始要求を待つ。
⑮ PURG-LI：他系より割り込み信号を受け取ったときに，その信号以前にTSに到着し，受信側プロセスにまだ通知していない結合モードのレターを全て消す。

上記コマンドのうち，⑫～⑭は高位プロトコルをインプリメントする場合に有効と考えられる。

4. おわりに

以上，INWGのEND to ENDプロトコルに対し，TSをインプリメントするという立場から気のついた問題点を述べてきたが，見逃がしている点あるいは誤解をしている点もあるかと考えている。また，最近はサブネット・レベルのプロトコルとしてX.25や日本電信電話公社のHDMなどが発表されているので，これらを用いてTSをインプリメントするという立場から再度，本プロトコルを見なおすのは決して無意味ではないであろう。

参考文献

1. Cerf. Mckenzie, Scantebury, Zimmermann : Proposal for an Internetwork End to End Protocol, IFIP WG 6.1, INWG General Note #96, 1975
2. GIEN : AMENDMENTS TO INWG 96 - POINTS FOR FURTHER STUDY, IFIP WG 6.1, INWG PROTOCOL NOTE #49, 1976
3. 電子計算機利用に関する技術研究会，周辺問題分科会，リソースシェアリングシステム研究班：リソースシェアリングシステム（プロトコルの論理設計），P22～P50，1977