

ネットワークアーキテクチャの互換性に関する考察

苗村憲司, 奥 光, 梶原俊男

(日本電信電話公社 横須賀電気通信研究所)

1. まえがき

計算機・通信回線網・端末を含むデータ通信システム（オンライン情報処理システム）の開発において、ネットワークアーキテクチャの思想と技術を活用することが重要な課題となってきた。ネットワークアーキテクチャは、ARPANET等のコンピュータネットワークにおける分散処理と計算機間通信の技術を基礎として生まれた基本方式である。即ち、その原形は、大規模なコンピュータネットワーク内の各種計算機等の機能分担とその間の通信規約（プロトコル）の技術にあったと言えよう。その後、マイクロプロセッサ等の技術の著しい進展によって端末等の小形の装置にまで処理機能を持たせることが経済的に可能となったため、この技術は広く一般のデータ通信システムにまで適用されるようになった。その結果、内外の主要メーカは独自のネットワークアーキテクチャを開発し自社製品に適用してきた。一方CCITTは公衆データ網（パケット交換網、回線交換網）の標準インタフェースの勧告という形でネットワークアーキテクチャの発展に寄与してきており、ISOにおいてもハイレベルデータリンク制御（HDLC）手順の国際標準策定作業をほぼ終了して漸くネットワークアーキテクチャの検討に着手した状態である。

ここでは、ネットワークアーキテクチャの多様化の原因を分析し、その標準化における問題点を考察する。一例として、CCITT勧告X.25によるパケット交換網インタフェースと専用ネットワーク用プロトコルの互換性の問題に触れる。次に、一般論としてプロトコル互換性に関する定式化の必要性、並びに非互換性解決手段の必要性について述べる。

2. ネットワークアーキテクチャの多様化の原因

ネットワークアーキテクチャにおける多様化の原因としては次の各項がある。

(1) ネットワーク構成の多様化

情報処理機能の分散形態について着目するだけでも、センタ計算機-端末間の機能分散（縦形分散）と複数センタ間の処理分散（横形分散）とがあり、更に両者の混合形態がある。縦形分散においては、例えばセンタ計算機上のプログラムと遠隔処理装置上のプログラムとの間、及び遠隔処理装置上のプログラムと端末オペレータとの間で通信が行われるが、この通信においては関連するリソースの管理・誤り制御等の方式に関して主従の関係を用いることが多い。これに対し、横形分散においては、各センタ計算機が各々のリソースを管理し誤り制御等についても責任を持つ対等の立場で通信を行う場合が多い。

通信制御機能の分散形態については、使用する通信回線網の種別に依存する点が多い。専用線（特定通信回線）等を用いる場合には、通信接続の制御・データ転送のフロー制御・ビット誤り制御等を含めすべての通信制御機能を計算機、端

末等において実現する必要がある。電話網等を用いる場合は通信接続の制御に関する分担に差異があるがデータ転送の制御に関しては専用線等の場合と同様である。これに対し、パケット交換網はデータ転送のフロー制御・ビット誤り制御等にも関与することにより網内の伝送回線等のリソースを有効利用して通信品質の向上と経済化を図るものであるから、必然的に計算機・端末等の通信処理機能を一部分担することとなる。

(2) 計算機・端末等の多様化

計算機機種のファミリー化により汎用計算機のアーキテクチャは統一されて行く方向にあると言えることもできる。しかし、ミニコンピュータは用途に合わせて多様化する動向にあり、更にマイクロコンピュータの出現により計算機と端末の境界に位置付けるべき各種の処理装置が出現してきた。これらの処理装置はまたセンタ計算機の前置処理装置として、あるいは端末寄りの遠隔処理装置等として用いられ、通信制御と情報処理の一部を分担することによりネットワークアーキテクチャに関与してくる。

また、マンマシンインタフェースとしての端末機器は、適用業務の種別やその利用形態等によって大幅に変化してきている。当初はキーボードプリンタ形の端末が主流であったが現在ではキャラクタディスプレイ装置が多く用いられるようになってきている。今後は漢字入出力・図形入出力等を目的とした新しい機器の開発が期待される一方、Point of Sales用やCredit Verification用の簡易端末の普及が予想されている。これに関連して電話機やファクシミリ端末のデータ通信端末としての応用にも注目する必要がある。これらの機器の制御方式と共に通信制御方式にも多様性を生じる可能性がある。

(3) 適用業務の多様化

計算処理を中心とした当初の適用業務のほかは、トランザクション処理、データベースアクセス等の業務分野が拡大しており、将来は文書処理等の新分野への適用も予想される。この結果、通信電文の形式・長さ、トランスミット密度等に多様性を生じ、またシステム設計上の主要な評価尺度であるスループット、応答時間、信頼性等に対する要求条件にも差異が現われている。

3. ネットワークアーキテクチャの標準化のために解決すべき課題

前節のように多様化の原因を認識した上でネットワークアーキテクチャの標準化を狙う場合に解決すべき問題点が多いが、特に重要と考えられる二つの課題について述べる。

(1) ネットワークの論理モデル

実際のネットワークを構成する装置やプログラムの多様性には依存しない論理モデルを作成することが必要である。この論理モデルは、基本的には装置に対応するノードと、ノード間を結びリンクとで構成する(図1)。但し各装置で分担可能な機能範囲は必ずしも標準ネットワークアーキテクチャのノードに割当てた機能範囲に対応しない場合があるので、各装置を1個のノードに対応させることは必須条件ではない。例えば、センタのホスト計算機と前置処理装置を合わせて1個のノードに対応させたり、既存の端末についてはこれをネットワークアーキテクチャとしての標準仕様に变换(仮想化)処理する装置のノードに含めて写像させたりすることが考えられる。また、通信回線網の扱いとしては、専用線等はリン

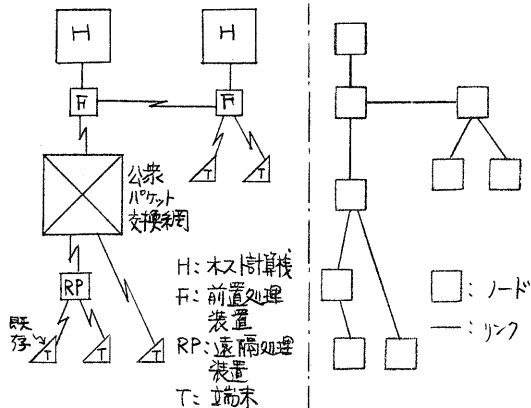


図1. データ通信ネットワークとその論理モデル(例)

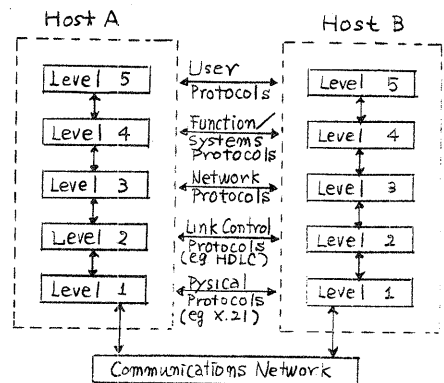


図2. 通信機能階層例 (ISO資料より)

クに対応させるのが自然であるが、パケット交換網のように通信制御機能を分担する網の扱いには次の3通りの案が考えられる。

- (a) 専用線等と同じくリンクに対応させる。
- (b) 処理装置等と同じくノードに対応させる。
- (c) パケット交換機をノードに対応させ、交換機間の伝送回線をリンクに対応させる(この結果、パケット交換網は全ネットワークのサブネットワークとなる)。

これらの案の中で、(a)案は種々の通信回線網を画一的にリンクとして取扱えるという点で簡明であるが、パケット交換網に取込んだ通信処理機能を効果的に利用できないことに難点がある。(b)案では図1の例のようにパケット交換網を他の装置等と同格のノードとして取扱うので、その機能を活用するのに適しているが、パケット交換網インタフェース(X.25等)を標準ネットワークアーキテクチャのプロトコルとして位置付けることが前提条件となる。(c)案はARPANETのようにパケット交換網をコンピュータネットワークの一環として開発する場合には適しているが、公衆パケット網のように多数のネットワークシステムで共用されるものに対しては適用できない。

(2) 機能階層構成

論理モデル化したネットワークの各ノードには原則として同一の階層構成の通信機能を割当てることとし、各ノード対間には標準プロトコルを適用させるのがよい。こうすることにより、ネットワーク構成の変更、新装置の組み込み等に起因する通信管理方式の修正量を最小化することができる。

通信機能の階層構成に関して確定した方式はないが、最近ISO/TC97/SC6内で審議されている次の構成は標準化の基礎として有用であると思われる(図2)。

- (a) レベル1 (Physical Protocols): 物理回線の制御(例: 回線交換網用X.21)
 - (b) レベル2 (Link Control Protocols): 隣接ノード間の論理リンクの制御(例: HDLC)
 - (c) レベル3 (Network Protocols): 複数ノード、リンクを通るEnd-Endのデータ転送制御
 - (d) レベル4 (Function/Systems Protocols): データの形式、アクセス方式等の制御(例: 파일 전송)
 - (e) レベル5 (User Protocols): ユーザの業務プログラム内で独自に行う通信の機能
- これらの中でレベル1はほぼ確定している。一方レベル2は情報処理内容に依

存してユーザが個別に定める自由度を残しておくことが必要であろう。従って、標準ネットワークアーキテクチャとしてはレベル2~4を主対象として定めることが考えられる。

(3) 多様化への対処

以上により画一的な論理ネットワークモデルとその上での通信機能の階層構成を定め、プロトコルを規定することとなるが、現実のネットワークに対してこれを適用するに当っては各装置等の多様性を考慮しなければならない。

一つの考え方としては、上述の階層構成の各階層毎に複数のプロトコルを定めておき、システム設計又は通信開始に当ってその中のどのプロトコルを使用するかをパラメータとして選択できるようにする方法がある。このようにして選択したパラメータの組合せをプロフィールと呼び、標準的なプロフィールは直接指定できるようにする方法が CCITT SG VII において端末インタフェースの一環として検討されている。

また、プロトコルの規定対象候補となる項目毎に一定の符号を割当て、プロトコルヘッダ内でこれを列記する HIC (Heading Item Code) 方式が ISO/TC97/SC6 において検討されている。この方式では、ネットワーク内を転送される各データブロック毎に上述のプロフィールに相当する HIC の系列を標記することとなり、オーバーヘッドの増加の恐れがある。

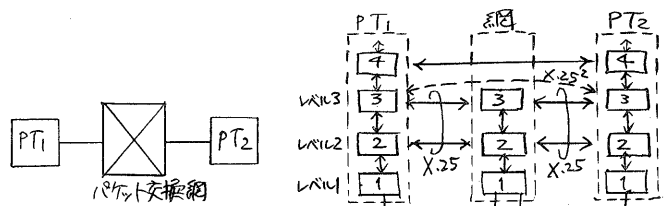
現実的なアプローチとしては、ネットワークアーキテクチャに対して必然的に要求される多様化への要求をいくつかの段階に分けて対処することが適切であろう。例えば、ネットワークシステムの設計時に選択すべき事項、装置対応のノード種別を表わす概念、通信開始時の選択、通信中の選択等を別の段階として対処することが考えられる。

4. レベル3プロトコルの互換性について

前節で述べた機能階層の中でレベル2のプロトコルとしては HDLC 手順が標準的であり、パケット交換網用の X.25 においてもそのサブセットを利用している。従って、次に標準化の対象として検討される主対象はレベル3のプロトコルである。

X.25 ではレベル3としてパケットモードのプロトコルを規定している。これを用いる計算機等のパケットモード端末 (PT) は網を通して相手の PT との間でバーチャルサーキット (VC) を設定してデータ転送を行うことができる。但し、X.25 は基本的には PT と網との間の規約であり、両端の PT 相互間には網を通して X.25 を cascade 接続した形のプロトコル (これを仮に $X.25^2$ と呼ぶ) が適用されると見ることができる。

従って、標準ネットワークアーキテクチャのレベル3プロトコルとして X.25 又は $X.25^2$ を位置付けることができれば、ネットワークシステムにおけるパケット網の活用



(i) ネットワーク構成

(ii) 論理モデルでの扱い

図3. パケットモードの通信の論理モデルでの扱い

が可能となる。

X.25を専用ネットワークのレベル3プロトコルとして利用する場合は、ルーティング、論理チャネル番号の変換、発呼と着呼の衝突、フロー制御方式の整合等の問題点を解決する必要があるが、X.25の方式を根本的に変えずに機能追加等を行うことによって対処可能と考えられる。

5. プロトコル互換性に関する一般的手法の必要性

ネットワークアーキテクチャ又はプロトコルの標準化を行う狙いは、各社の開発する製品を相互に接続して目的に合ったシステムを作成できるようにすることにあると考えられる。しかし、前述したようにその画一的な統一という意味での標準化は困難と考えられるので、何らかの手法によって異なるプロトコル相互間の互換性に関する処置を取ることが必要となる。

アナロジーをプログラム言語に取れば、JISにおける水準の考え方によってプログラム言語（又はコンパイラ）間の上位互換性が保証されるように、プロトコル相互間にも上位互換性を定義することができれば好都合である。

しかし、プロトコルにおける上位互換性は、プログラム言語の場合と異なり、そのまま製品の上位互換性を意味するとは限らない。言うまでもなくプログラム言語においてはソースプログラムとコンパイラがユーザ/サーバの関係にあり、前者の水準が後者の水準に比較して低い（即ちソースプログラムがコンパイラの提供する機能を使い切っていない）ことは実害とならない。しかし、プロトコルにおいては、対話する左右の製品が互いに相手のユーザであると同時にサーバでもあるという形態が普通であるため、一方のプロトコル水準が他方よりも低いことは一方向の通信のみを可能とする結果となる恐れがあるからである。

従って、今後の標準化動向としては、非互換のプロトコルを有する製品相互間で互いに共通のプロトコル水準（サブセット）を見出す目的であらかじめ行うべき協議のための通信規約を定めることも重要となる。この規約は、プロトコルを選定するためのプロトコルという意味で「メタプロトコル」とでも呼びべきものである。その方式としては、CCITT SG VIIにおけるパラメータやプロフィールの概念、ISOにおけるHICの手法等を応用する可能性がある。

6. あとがき

ネットワークアーキテクチャは、システム内の計算機、通信回線網、端末等を論理的モデルに写像し、その上で通信を行うためのプロトコル等を定めるものである。しかし、その仕様は現実の装置等の多様性を反映する必要があり、その互換性に関する研究の必要性は大きい。今後、ISO等において標準化検討が進められるのに並行して、互換性に関する理論研究が望まれる。