プロトコルの記述方法と検証方法について

河岡 司，友永 充宏，高橋 祥兼
（日本電信電話公社 横須賀電気通信研究所）

1．まえがき

近年、各種コンピュータ・ネットワーク・アーキテクチャが相次いで発表されているが、一般に、これらのアーキテクチャの中ではコンピュータネットワークの論理構造が定義され，それに基づくプロトコルが規定されている。論理構造はコンピュータネットワークの構成，すなわち、ノードの階層構成，エンティティおよびそれらの論理関係を定義するもので、プロトコルを規定する際の前提となっている。プロトコルは、異なるノードに存在する同一階層のエンティティ間の通信規約であり、エンティティ間の同期動作も含め，厳密な論理を規定している。

コンピュータネットワークは，それを構成する各種装置（ホスト計算機，前置処理装置，端末，通信回線網等）のハードウェアやソフトウェアがアーキテクチャで定められたプロトコルを正しくインプリメントすることにより実現される。この様に実現されたコンピュータネットワークが正しく動作するためには、以下の事項が満されていなければならない。

（A）アーキテクチャで規定されたプロトコルが一意に解釈可能であり，論理矛盾がないこと。

（B）コンピュータ・ネットワークの構成要素となる製品（ハードウェア，ソフトウェア）が、これらのプロトコルを正しくインプリメントしたものであること。

したがって，コンピュータ・ネットワークの実現に際しては、コンピュータ・ネットワーク・アーキテクチャの研究に関連し、これらの課題に対する以下の技術についての検討が必要となる（図1.）。

① 一意に解釈可能なプロトコル仕様を記述するための〝プロトコル記述法〟についての検討，②プロトコル仕様の論理的無矛盾性を確認するための〝プロトコル検証法〟についての検討，③プロトコル仕様にしたがった製品を容易に実現するための〝通信処理記述言語〟の検討，④製品がプロトコル仕様を正しくインプリメントしていることを確認するための〝製品検証法〟についての検討。

本稿では、上記の各技術について，既存技術の概要と特徴を述べ、更に、今後の課題に関して考察する。

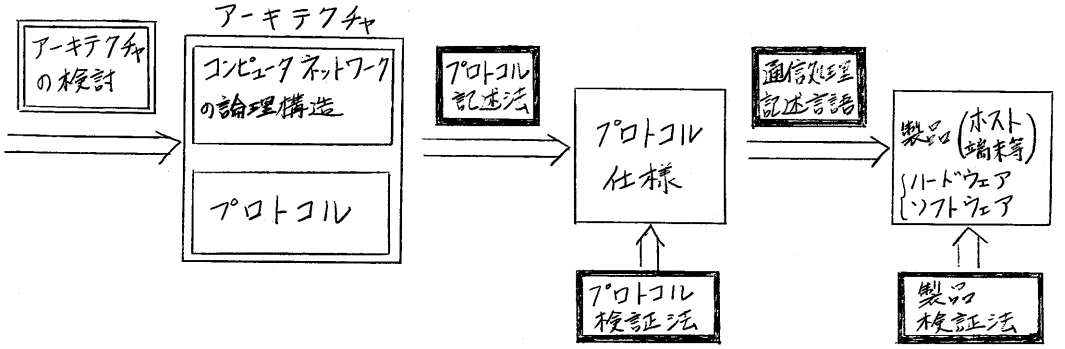

図1．コンピュータネットワークの実現に必要な技術

2. プロトコルの記述法

2.1 プロトコル記述法の必要性と具備条件

プロトコルにしたがって通信を実行する抽象的な装置を考え、これをプロトコルマシン（以下PMと略記する）と呼ぶことにする。PMは一種のオートマトン（一般には有限状態）とみなすことができる。したがって、プロトコルとして規定される内容は、PMのある状態に於て、あるイベントが発生した時に、PMがどの様な動作（処理手順）を実行し、どの状態に遷移するかということである。この様に、プロトコルを規定することは、1つの有限オートマトンを定めることに相当するが、PMが実際の装置上にインプリメント可能なためには、プロトコルが何らかの形で、一意に解釈できるドキュメントとして表現されていなければならない。更に、このドキュメントの記述法は、以下の様な条件を備えていることが必要と考えられる。

- 記述が容易なこと
- 記述されたものが解り易いこと
- 規定内容が厳密であること
- プロトコルの検証に有効なこと
- 例外処理を含めた網羅的規定が可能なこと
- インプリメントに対し有効であること

しかしながら、これらの条件の中には相反した性質のものもあって、全てを満足する様な記述法の開発は困難である。そこで通常は、プロトコルの記述に際し、その記述の目的に応じて特に重要な具備条件とプロトコルの特性とを考慮して最も適した記述法が使用される。

2.2 各種記述法

ここでは、既に適用され、あるいは提案されている各種のプロトコル記述法について、その概要を述べる。

（1）状態遷移図

最も広く用いられている方法であり、記法には種々の相違があるが、何れも本質的には、状態を表わす節点（ノード）と、状態間の遷移を表わす有向枝（アーク）から成る有向グラフとして構成される。アークには一般に、遷移要因とアクションとが対応している。

（2）状態遷移表

これは基本的には状態遷移図を表形式に描いたものであり、現状態と状態遷移要因の全ゆる組合せに対して、遷移先状態とアクションとを一覧表の形に記述する方法である。本方式は状態遷移図に比べてスペースファクタが良く、また規定の網羅性の点ですぐれているが、記述内容の解読の容易性という点では劣っている。

（3）状態遷移表と処理状態遷移表の組合せ（[8]）

一般に、状態遷移表のみでは処理手順（例えば、カウンタの更新などのパラメトリックな処理）の詳細な規定が困難である。本方式では、これらの処理を別表（処理状態遷移表）として記述しておき、状態遷移表の処理記述（状態と遷移要因との交点）において対応する処理のエントリ番号を指定してマッピングをとることにより、状態遷移に伴う処理動作の明確な記述をはかるものである。

（4）Petri-Net（[7]）

Petri-Netは、P節点（"○"で表わす）とT節点（"|"又は"—"で表わす）

の2種類の節点とそれらを相互に結合するアークとで構成される並列処理表現のグラフモデルである。P節点にはトークン(“・”で表わす)を複数個配置できる。ノつのT節点に関してその各々の入力P節点(そのT節点を終端とする様なアークの始端のP節点)にトークンが1個以上配置されているとき、そのT節点は発火可能であるという。発火可能なT節点が発火すると、その各々の入力P節点からトークンを1個づつ取り除き、各出力P節点(そのT節点を始端とする様なアークの終端のP節点)にトークンを1個づつ付け加える。一般に、各P節点におけるトークンの配置は系の状態に対応し、T節点の発火は事象の発生に対応する。図2.にPetri-Netによる簡単なプロトコルの記述例を示す。

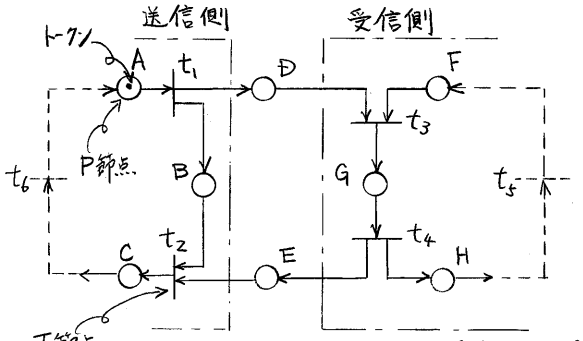

図2. Petri-Net(メッセージ送受信モデル)

表1. PDCを構成する節点

節点名称	記号	実システムへの対応
E節点	AND / OR	制御動作の実行完了 事象の発生
P		動作の実行中な実行待ち
Q		待ち行列(キュー)
S		制御及びデータの複数方向への同時移行
L		論理判断による制御又はデータの分岐

(5) PDC法[11]

これは、Petri-Netをベースとして通信の同期動作の表現と、実システムの制御動作との対応づけを可能とするための改良を加えた記述法である。PDC(Protocol Description Chart)は、表1.に掲げる様な各節点をアークによって結合した一種の有向グラフである。PDCではPetri-Netと同様に、E節点の発火によるトークン移動として通信系の動作を表現するが、更に実システムとの対応を容易にするためにE節点における処理手順の記述や、論理判断によるトークンの振り分けなどを可能としている。

(6) プロトコルグラフ

Gouda[4]はPMの動作を図3.に掲げる4種の基本動作によって流れ図的に記述する方法(プロトコルグラフ)を提案している。この方法はメッセージの送受信シーケンスの表現に着目した記述法であり、複数個(3以上)のPMが通信する場合にも適用性があるが、PMの状態遷移がわかりにくくなるという欠点をもつ。

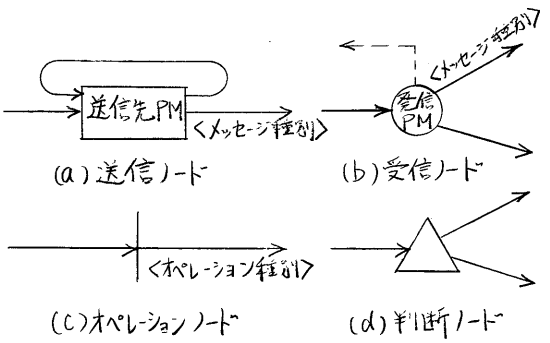

図3. プロトコルグラフの構成要素

(7) 高級言語による方法[17]

PL/IやPASCALなどの汎用プログラム言語に近い言語を用いてプロトコルの記述を試みるものである。この方法の特徴は、処理手順の明確な記述が可能なこと、インプリメントへの移行が容易なこと、及びプロトコル仕様のマシン管理への適用性が高いことなどがあげられる。一方、プロトコルの視覚的把握が困難であり、また、処理順序の一意的記述により過剰規定に陥り易くなる。

（8）その他の方法

以上の他に、有限オートマトンに対応する形式言語を用いたもの［5］や、階層構成をなすプロトコルの各レイヤを3入力2出力オートマトンとして形式的に定義する方法［6］などが提案されている。

2.3 記述法のまとめ

前節で述べた各種のプロトコル記述法は、大別して通信する2つのPMを一体の系として記述する方法と、個々のPMを個別に記述する方法とに分類できる。各々の一般的特徴とその適用領域を表2.に示す。一方、別の観点からの分類として、通信系の状態遷移に着目した方法と、実システムの処理手順に即した方法、及びその中間的な方法とに分けることができる。これらの特徴とその適用領域を表3.に示す。表2.,表3.は記述法の原理的特徴からみた分類，評価であり、各カテゴリに属する具体的記述法が共通に持つ本質的な利害得失である。これらの各カテゴリは何れかが他よりすぐれたものであるというわけではなく、一般に、プロトコルの特性と記述の目的とによってその適用性を異にするものである。今後も各種のプロトコルの設計/インプリメントとともに、種々の記述法が考案されるものと考えられるが、望ましい方向としては、表2.,表3.に掲げた各カテゴリ対応に標準的な記述法を制定することであろう。

表2. プロトコル記述法の分類（1）

記述法	特徴評価	適用領域	例
PM対の一体記述	・同期動作の把握が容易(○) ・非同期的例外処理の規定が困難(×)	プロトコルの中核機能のみを正確に記述する必要がある場合など	・Petri-Net ・PDC法
各PMの個別記述	・例外処理の規定が容易(○) ・同期動作がわかりにくい(×)	例外処理を含めて網羅的な規定を必要とする場合	本稿で述べた上記以外の各記述法

表3. プロトコル記述法の分類（2）

記述法	特徴評価	適用領域	例
状態遷移の記述	・同期動作の把握が容易(○) ・実システムの処理手順との対応が困難(×)	同期動作を主体とするプロトコル（コネクション設定/解放プロトコルなど）	・状態遷移図(表)
処理手順の記述	・実システム動作との対応が容易(○) ・インプリメントに有効(○) ・同期動作がわかりにくい(×)	処理動作を主体とするプロトコル（データ転送プロトコルなど）	・高級言語 ・プロトコルグラフ
状態遷移と処理手順を併せた記述	・上記2者の中間的特徴をもつ	同期動作と処理手順とが混在するプロトコル	・状態遷移表と処理状態遷移表の組合せ ・Petri-Net ・PDC法

3. プロトコルの検証法

3.1 検証の必要性

プロトコルとしての規定内容が複雑になると、それがたとえ前述の様な形式的記述法によって厳密に記述されていたとしても、それにもとづく通信が所期の目的を達成できるかどうかは必ずしも明らかではなくなる。即ち、例えば、次の様なプロトコルエラーが存在する可能性がある。

・同期動作に関する規定ミスのためシステムがデッドロック状態に陥る

・ある一定の動作を無限に繰返して通信が進行しない．
・通信終了時点に於て送信側／受信側の送信確認／受信確認メッセージ番号に不一致を生じている
・伝送中に生じた誤り（メッセージの紛失など）の回復ができない

したがって、プロトコルの設計段階でこれらの論理ミスがないことの確認，即ちプロトコルの検証が必要となる。また、この様な検証が厳密に行われるためにはそれに先立って、プロトコル仕様自体が正確に記述されていることが必要であり検証はそれをベースとして行われる。このことから、プロトコルの検証法は記述法と密接な対応関係を持っている。

2.2 各種検証法

ここでは，既に公表されている主なプロトコル検証法についてその概要と特徴を述べる。

（1）状態遷移図を主体とした方法

Zafiropulo[14]は、状態遷移図で規定されたプロトコルについて、その初期状態から出発して再び初期状態に戻る迄のイベント系列を、通信する2つのPMに対して組合せたもの（dialogue）を、一定のアルゴリズムの下に，それが(a)実行可能であるか，(b)起り得ない系列であるか，或いは(c)プロトコルエラーの可能性があるかに分類して検証する方法を提案している。この検証手順は自動化されているが、この方法が適用できるためには、状態遷移図には初期状態を含まないループが存在しないことなど、かなり制約条件が多い。 Danthine[3]は，先ず、通信の実行に於て考えられる全ゆるデッドロック状態を見出し、実際に通信系がその様な状態に到達するか否かを調べるために、通信する2つのPMの状態対を、デッドロック状態から初期状態へ向けて逆向きにたどることを提案している。この検証手順は自動化されているというが、[3]にはその具体的アルゴリズムは述べられていない。

（2）合成状態図を用いた方法

Sunshine[10]は通信する2つのPMの状態対と，PM間を転送中のメッセージの状態（種別，方向，順序など）を同時に考慮して描いた状態遷移図（合成状態図）を用いて通信系が到達可能な状態（デッドロック，ループ或いは目的の状態など）を確認する方法を提案している。この方法は基本的には考えられる合成状態間の遷移を網羅的に描くことを必要とするが，これはスペースファクタや記述工数の点でも困難であり、[10]では検証上本質的な状態遷移のみを描くという方法をとっている。West[13]は同様の方法を3つ以上のPMが相互に通信を行う場合にも適用している。

（3）Petri-Netのトークンマシン（TM）を用いる方法

Petri-Netにおいてある時点での全てのP節点に対するトークンの配置（マーキング）はT節点の発火とともに変化する。この変化の状況をグラフ化したものをトークンマシン（TM）と呼ぶが、Petri-Netで記述されたプロトコルの場合TMは一種の合成状態図とみなすことができ、それを解析することにより（2）と同様の検証が可能である。Merlin[7]は通常のPetri-Netに発火条件の制限を付加したTime-Petri-Netの概念を用いてトークンの紛失（転送中データの紛失）に対するプロトコルの誤り回復可能性の問題を取扱っている。

（4）処理ロジックを解析する方法

Stenning[9]はプログラム言語PASCALにより記述されたデータ転送プロトコ

ルについて、通信実行時における状態変数（送受信シーケンス番号など）間に成立つ論理関係式の検証を行っている。各論理関係式の検証の方法には、並列処理プログラムの正当性証明の手段と同様、一般に、プロトコルの実行に伴うイベント系列に関する帰納法が適用される。但し、この方法で検証される内容はプロトコルにもとづく通信が"実行される"ことを前提とした上での正当性（部分正当性）の証明であり、実行されるかどうかの検証は別途行う必要がある。

（5）状態遷移と処理ロジックの解析を併用した方法

Bochmann〔2〕は、1つのプロトコルを状態遷移図による記述部分と言語による手順記述部分とに分けて記述し、（1）と（4）を併用する様な方法によってプロトコルが実行され、かつ実行結果が正しいこと（全正当性）を証明している。また、〔12〕では、PDC法により記述されたプロトコルについて、同様な検証を行っている。

3.3 検証法のまとめ

前節に述べた各種検証法とその特徴は、手法の原理的観点からそのベースとなった記述法に対応して表4.の様に3つに分類してまとめることができる。しかしながら、これ迄に知られている手法は何れも未だ十分に実用的とは言い難い様である。即ち、合成状態図を用いる方法では、プロトコルの複雑化によって、合成状態数がたとえ計算機によっても処理できない程に増大してしまうことが起り、また、処理ロジックを解析する方法では、プログラムの正当性証明と同様、フォーマルな証明法としてはあまり有効なものはない現状である。今後検証法が有効なソフトウェア工学として発展するためには、次の様な課題の研究が必要であろう。

① 検証手順の自動化手法
② 非同期的に発生する例外処理規定に関する検証法（種々のタイミングで発生するエラーに対する回復可能性の検証法）
③ 階層構成をなすプロトコルの検証法
④ 100％の検証能力はないが実用上有意な程度の能力をもつ実用的手法
⑤ 検証が容易なプロトコル記述法の開発
⑥ プロトコル検証に係わる種々の決定可能性問題の解法

表4. プロトコル検証法の分類

検証手法	対応する記述法	検証内容	特徴評価
状態遷移をもとにした方法	系の状態遷移に着目した記述法（状態遷移図/表など）	・デッドロックの可能性 ・所期の（又は不当な）ループの存在性 ・特定の状態への到達可能性	・通信系全体としての動作可能性のチェックが容易（○） ・検証自動化が容易（○） ・処理ロジックの正しさの検証には不向き（×）
論理関係式の証明を行う方法	処理ロジックの記述（高級言語など）	・プロトコル実行中に成立つ状態変数間の関係式の検証	・パラメトリックな処理手順の検証に適す（○） ・検証自動化は困難（×） ・系の動作可能性の検証には不適（×）
上記2者を併用した方法	状態遷移と処理手順の両方を記述する方法（PDC法など）	・上記2者を併せた内容	・上記2者を併せた利害得失をもつが、プロトコルの規定内容にも依存する

4. 通信処理記述言語

ネットワークアーキテクチャにもとづきコンピュータネットワークを構築するためには、それを構成する各ノード上にプロトコルマシンを実現することが必要で

ある。特に、PMがソフトウェア(プログラム)によって実現される場合には、その処理ロジックは1つのプログラム言語を用いて記述することが必要である。

一般に、アーキテクチャにもとづくプロトコル処理は次の様な特徴をもつ。

・通信するPM相互の同期制御
・プロトコルで用いられるヘッダフォーマットの定義
・同一ノード内の複数のPM相互のインタフェース

したがって、プロトコル処理プログラムの記述のために専用の言語(通信処理記述言語)を検討することは、ネットワークの実現する上で必要な技術の1つになると考えられる。この様な言語は、プロトコル仕様とのマッピングの容易性の点から、一般にプロトコル記述法と関連した言語仕様になると考えられるが、プロトコルの規定とは異り、各ノードのインプリメント依存条件等も考慮した幅広い処理を記述できる必要がある。この方面の検討は未だあまり例を見ない様であり今後の課題であろう。

5. 製品検証技術

ここでは、ネットワークアーキテクチャのプロトコル仕様にもとづいてインプリメントされる各種装置およびそれらの相互接続によって構築されるネットワークシステムの検証技術について述べる。

5.1 製品検証の意義

プロトコル仕様に基づいてインプリメントされたノードおよびネットワークシステムが実際にプロトコル仕様に正しく準拠していることを確認することを製品検証と言う。以下、製品検証のことを単に検証と略すこともある。この検証の定義は特に以下のことを意味していることを注意しておく。

① 検証はノードを単位として行うものであり、ノードを構成するハードウェア製品またはソフトウェア製品を単位として行うものではない。
② 検証対象の異常原因の究明は行わない。したがって、異常原因の究明を主目的とするデバグとは異る。
③ ノードまたはネットワークシステムの設計段階での仕様の確認は含まない。
④ プロトコル仕様では規定されないインプリメンテーション固有の事項については確認しない。また性能評価は検証の範囲外である。

5.2 製品検証の必要性

コンピュータネットワークの効用は、異機種コンピュータネットワークの実現により一層増大する。異機種コンピュータネットワークは異る製造会社の各種製品を相互接続することによって構築されるため、以下に示す検証が必要となる。

① 製品間接続の安全性の検証
アーキテクチャ製品は同一のプロトコル仕様に基づくものであるが、そのインプリメント形態には製造会社ごとに多様なバリエーションが考えられる。したがって、製造会社ごとに個別に開発されたアーキテクチャ製品が、ノードとして正常に動作することを確認し、特にネットワーク内の他のノードに悪影響を与えないことを確認する必要がある。

② ネットワークの安全性の検証
製造会社の異る各種ノードを相互接続して構築されたネットワークシステムが正常に動作することを確認し、特にネットワークシステムが当該ネットワークで提供されるアプリケーションサービスシステムに悪影響を与えないこ

とを確認する必要がある。

5.3 検証システムの実現方式

アーキテクチャ製品（ノード）及びネットワークシステムを統一的な基準に基づいて検証するための製造会社共通の手段を検証システムと呼ぶ。近年、データリンクレベルのプロトコルに関しては、検証センタ及びポータブルシミュレータ等が内外で開発されてきており、また、DDXパケット交換網では、網への加入資格審査という立場からX.25プロトコル試験について検討されている[1]。現在のところ、特定のアーキテクチャに基づく製品及びネットワークシステムに対してプロトコル仕様を確認するための検証システムの実現例はなり様であるがその実現方式としては以下の各方式が考えられる。

（1）ノード検証方式

ノードを検証対象とし、主にプロトコル処理機能を検証する方式であり、次の2方式が考えられる。

① 方式A：検証システムを検証対象とは別の装置として実現する方式（図4.）。

② 方式B：検証システムを検証対象の装置内で走行するプログラムとして実現する方式（図5.）。本方式ではプログラマブルなノードのみが検証対象となる。

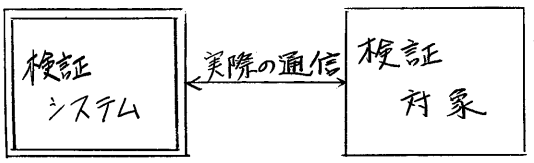

図4. ノード検証方式A

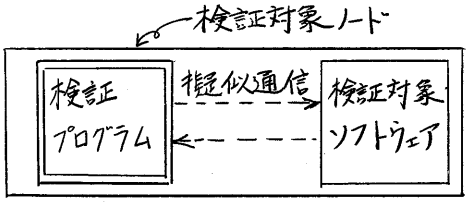

図5. ノード検証方式B

（2）システム検証方式

ネットワークシステムを検証対象とし、主にネットワーク管理機能を検証する方式であり、以下の2方式が考えられる。

① 方式C：ネットワーク内の任意の必要箇所にセンサ（情報投入／収集装置）を設置し、これと検証センタを回線で接続しオンラインで検証を行う方式。

② 方式D：ネットワーク内の必要なノードで情報の投入／収集を行い、収集情報をオフラインで検証センタに運び検証する方式。

ノード検証方式による検証システムとシステム検証方式による検証システムでは機能的に共通な部分が多いと考えられる。従って、両システムを一つのシステムとして実現することが望ましい。図6.に最もハイレベルな検証システムの実現イメージ（方式Aと方式Cの融合イメージ）を示す。

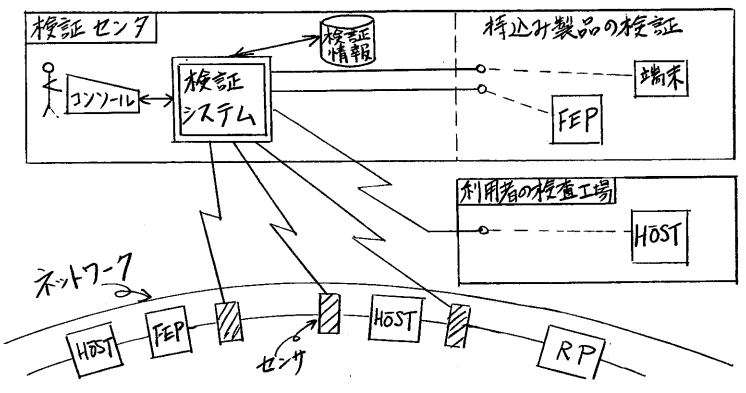

図6. 検証システムの実現例

5.4 検証方式の概要

以下、ノード検証方式及びシステム検証方式の概要を述べる。

(1) ノード検証方式

ここでは、プロトコル仕様において、プロトコルが複数のレベルに分割され、各レベル毎に、当該レベルへの入力の正否をチェックする機能と、正しい入力を受けた場合の状態遷移処理とが規定されているものとする。

①検証項目：各レベル毎にチェック機能と状態遷移処理の2つのカテゴリに分け、各々について規定された機能単位を検証項目とする。

②確認手法：チェック機能に関しては、異常データの投入に対応するセンスコードを含む否定応答によって確認し、状態遷移処理に関しては、状態と遷移要因毎に処理内容及び状態遷移先を確認する。

③環境設定：検証項目毎に、検証システムまたは検証対象サイドのオペレータ等をトリガとして検証対象を検証すべき状態に設定する。

(2) システム検証方式

システム検証の対象となるネットワークシステムの各構成要素は既にノード検証を終了しているものとする。

①検証項目：規模、コスト等の点で技術的にノード検証が困難な項目及びノード対間のインプリメンテーション情報の不一致に基づくプロトコル違反が起り得る項目(ペーシング値，ブロッキング／デブロッキング等)を確認する。

②確認手法，③環境設定：基本的にはノード検証方式と同様である。

5.5 検証方式の具体化に伴う諸問題

(1) 検証の完全性

検証においては、検証項目が網羅的でしかも各検証項目毎にきめ細かい検証がなされることが重要である。この検証項目の網羅性および検証方式の精密性(これを検証の完全性と呼ぶ)に関する要求条件を定量的に把握するとともに、検証対象内で定義されているプロトコルの中で、どの程度が検証されるかということ(検証率)を明らかにすることが必要である。この場合、厳密な意味での有効な検証率を得るためには、検証項目数が無限となるが、ノードメネットワーク稼動時のプロトコル使用頻度を考慮することにより所期の検証率を推定できる。

(2) 検証の効率性

所期の検証率を満す検証項目群を最も効率良く実行する手順(検証手順)を求めることが必要である。検証効率を最大にするためには、検証に必要なデータ量、マニュアル操作量及び検証時間(検証対象及び検証システムを検証のために占有する時間)を最小化する手順を求めることが有効である。

検証の完全性，効率性に関しこれ迄に以下の検討結果が得られている。

① 単一レベルの検証の場合

検証対象が単一のプロトコルレベルで構成され、当該レベルのプロトコルが単一の状態遷移表として規定されているとする。この場合、状態遷移表の遷移要因の順序列対応にその発生確率を考慮することにより、与えられた検証率を満す有限な検証項目群を求め、更に、検証時間最小化の観点からの最大効率を得る検証手順を求めることができる。これらの手順は、グラフ理論及び線形計画法の理論を援用してアルゴリズムとして得られている。

② 複数レベルの検証の場合

検証対象が複数レベルのプロトコルで構成され、各レベルの状態遷移処理が単一の状態遷移表として規定されている場合、全体の状態遷移処理を表現する複合状態遷移を考え、①で得られたアルゴリズムを適用する。

6. あとがき

本稿では、主要なプロトコル記述法と検証法について、その概要を紹介し、それらの適用に当って考慮すべき点を明らかにするとともに、プロトコルのインプリメントに係わる問題として、記述言語と製品検証技術について考察した。プロトコルの記述法の中には、実用的プロトコルに適用されているものが多いのに対し、検証法は未だ実用上の十分な応用をみていない様である。この分野の研究は未だ日が浅く、今後の発展が望まれるところである。また、通信処理記述言語も未だ実用化の例をみないが、インプリメントに即したプロトコルの記述法とも関連し、今後の課題であろう。一方、最近の国内外における標準アーキテクチャ制定の動きからも、製品検証技術は、今後のコンピュータネットワークの発展にとって不可欠のものになると思われる。

謝辞

日頃御指導をいただく当研究所データ通信網研究室の苗村室長はじめDCNA関係各位に厚く感謝いたします。

[参考文献]

(1) 浅野,北川,大友 他：パケット交換網における端末接続試験，コンピュータネットワーク研資 CN16-2 (1978)

(2) Bochmann,G.V.:A Unified Method for the Specification and Verification of Protocols, IFIP Congress Proc.(1977).

(3) Danthine,A. et al.:Modelling and Verification of End-to-End Transport Protocols, Symposium on Computer Network Protocols (IFIP/ACM),Liege Belgium (1978).

(4) Gouda,M.G. et al.:Protocol Machines---A Concise Formal Model and its Automatic Implementation, Proc.ICCC (1976).

(5) Harangozo,J.:An Approach to Describing a Data Link Level Protocol with a Formal Language, Proc. Fifth Data Communication Symposium (1977).

(6) 井手口,水野,松永：通信プロトコルの抽象機械化とその応用，信学会 電算機研資 EC77-30 (1977)

(7) Merlin,P.M. et al.:Recoverability of Communication Protocols---Implications of a Theoretical Study, IEEE tran. Comm. Sept. (1976).

(8) 森野,高橋,田島,苗村：ハイレベルデータリンク制御手順の規定方法について，情学全大17 (1976).

(9) Stenning,N.V.:A Data Transfer Protocol, Computer Networks 1, (1976).

(10) Sunshine,C.A.:Interprocess Communication Protocols for Computer Networks,Ph.D. thesis Stanford Univ. (1975).

(11) 友永,阿部,宮沢,河岡：プロトコル記述法の一検討，コンピュータネットワーク研資 CN12-2 (1977)

(12) 友永：プロトコルの状態遷移と処理手順の検証法，情学全大 19 (1978)

(13) West,C.H.:General Technique for Communications Protocol Validation, IBM J. Res. Develop. Vol.22 No.4 (1978).

(14) Zafiropulo,P.:A New Approach to Protocol Validation, International Conference on Communications (1977).