マルチメディア通信と分散処理 77－5
（1996. 7. 12）

ルールベースプログラミングを用いた プロトコル相互接続試験システムの設計

加藤 聰彦　　大岸 智彦　　井戸上 彰　　鈴木 健二

国際電信電話（株）研究所

〒356 埼玉県上福岡市大原 2-1-15

コンピュータ通信の普及に伴い、通信システムの相互接続試験技術の確立が重要となっている。特に相互接続試験では、通信回線上を監視して得られたPDUシーケンスを用いて、通信システムの動作を推測するため、実際のネットワーク環境における遅延や伝送誤りにより、監視したPDUの送受信順序と、被試験システムが実際にPDUを処理した順序が異なる場合がある。本稿では、筆者らが先に開発した、OSIインテリジェントリンクモニタを対象として、ルールベースプログラミングを用いて、これらの問題を解決する技法について述べる。

Design of Protocol Interoperability Testing System based on Rule Base Programming

Toshihiko Kato　Tomohiko Ogishi　Akira Idoue　Kenji Suzuki

KDD R & D Laboratories

2-1-15 Ohara, Kamifukuoka, Saitama 356, JAPAN

According to the progress of computer communication, the interoperability testing of communication systems are widely used. However, in order to perform the interoperability testing in the actual network environment, the propagation delay and the transmission error might introduce the problems such that the observed order of protocol data units by the testing system is different from the processing order in the system under test. This paper describes the application of the rule base programming to resolve these problems in the OSI intelligent protocol monitor which we implemented before.

1 はじめに

コンピュータ通信の普及に伴い、通信システムの試験技法の確立が重要な課題となっている。通信システムの試験は、適合性試験と相互接続試験に大別される。通信システムが実装されると、まず、それが通信プロトコルを正しく実装しているかの適合性試験が行われる。しかし、この試験ではすべての誤りを検出することはできないため、独立に開発された通信システムを通信させ、問題があった場合にそれを検出し、その原因を解析する相互接続試験が重要となる [1-3]。筆者らは、OSI 通信を対象とし、OSI システム間の通信回線を監視し、転送された PDU (Protocol Data Unit) の情報を用いて、通信しあう OSI システムの動作をエミュレートし、各システムのプロトコル動作の正当性を検査するインテリジェントリンクモニタを開発している [3]。

しかし、このような相互接続試験システムでは、回線監視により得られた PDU シーケンスから、被試験システムの動作を推測するため、実際のネットワーク環境で使用する場合にいくつかの問題がある。例えば、被試験システム間の伝送遅延のため、検出した PDU の送受信順序と、被試験システムが実際に PDU を処理した順序が異なる場合がある。また伝送誤りにより、試験システムが検出した PDU を被試験システムが受信処理できない場合がある。

このような問題を解決するためには、検出した PDU シーケンスに基づいてエミュレートされた被試験システムの動作に誤りが検出された場合も、それが通信システムに起因するか、ネットワーク上の遅延や伝送誤りによる可能性があるかを推定する必要がある。このような推定を行うためには、決定的なアルゴリズムを用いる方式よりは、考えられる可能性を適用し、最も適当と考えられるものを使用する診断手法 [4] が適していると考えられる。そこで本稿では、相互接続試験システムを実際のネットワーク環境で動作可能とするために、ルールベースプログラミングを適用する方法について述べる。特に、筆者らの開発したインテリジェントリンクモニタに、ルールベースプログラミング用いることにより、SUT において PDU を処理した順序を推定するための機能の設計について示す。

2 インテリジェントリンクモニタの概要と課題

2.1 インテリジェントリンクモニタの概要

図 1に、これまでに開発したインテリジェントリンクモニタのシステム構成を示す。本モニタは、被試験システム (SUT : System Under Test) の間の通信回線上を転送されるフレームを監視し、フレーム取込みモジュールがこれを取り込む。取り込まれたフレームは、SUT A と SUT B

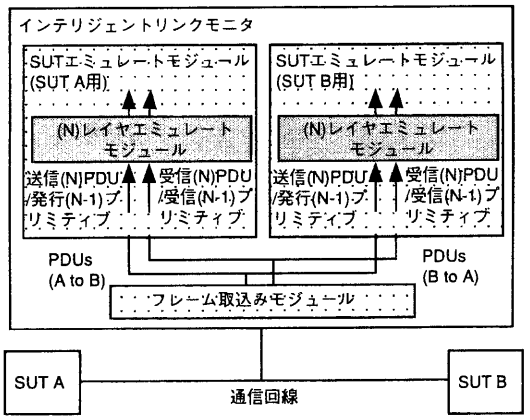

図 1: インテリジェントリンクモニタのシステム構成

の動作を模擬する SUT エミュレートモジュールに渡される。SUT A から B へ転送されたフレームに対して、SUT A の SUT エミュレートモジュールは、SUT A がそれを送出したとして扱い、そのための動作を模擬する。逆に、SUT A から B へ転送されたフレームに対しては、SUT A がそれを受信した場合の動作を模擬する。

SUT エミュレートモジュールは、各レイヤをエミュレートする (N) レイヤエミュレートモジュールに分割される。(N) レイヤエミュレートモジュールは、(N-1) レイヤエミュレートモジュールが生成した、送信または受信 (N)PDU および発行または受信 (N-1) プリミティブを入力される。これらの入力に対して、そのレイヤの状態が判定していない場合は、状態の推定を行い、状態が判定している場合は、(N) レイヤプロトコルの動作をエミュレートし、必要に応じて (N+1) レイヤエミュレートモジュールへの出力を生成する。

プロトコル動作のエミュレートは、通常の通信システムのプロトコル処理とは異なり、以下のような手順となる。

(1) 受信 PDU または受信プリミティブの処理

受信 (N)PDU または (N-1) プリミティブが与えられると、(N) レイヤエミュレートモジュールは、その入力を受信したとして、以下のように、(N) レイヤの状態遷移に従った処理を行う。

- 上位レイヤに通知すべきプリミティブが発生すれば、上位のエミュレートモジュールに渡す。
- 下位レイヤに出力する PDU／プリミティブがある場合は、次に期待される出力として保持する。
- 与えられた入力により、複数の遷移が可能である場合は、すべてを可能性のある遷移として保持する。
- 下位レイヤへの出力がある場合または複数の遷移がありうる場合は、実際の状態の遷移は行わず、後述のように、それに続く送信 PDU または発行プリミティ

ブの処理の中で、状態遷移を行う。ただし、一定時間以上送信PDUまたは発行プリミティブが存在しない場合は、出力無しのイベントが生じたと考えて遷移を行う。

(2) 送信PDUまたは発行プリミティブの処理

送信(N)PDUまたは発行(N-1)プリミティブに対しては、以下のような処理を行う。

- 出力が期待されるPDU／プリミティブが保持されている場合は、通知されたPDUまたはプリミティブがそれと一致するかを判断する。一致すれば、その出力が行われたと判断し、対応する状態遷移を選択する。
- 期待される出力が存在しない場合または一致しない場合は、(N)レイヤの状態遷移から、そのPDU／プリミティブを出力する遷移があるかを調べる。あれば、その遷移が行われたものと判断し、(N)レイヤの状態を遷移し、その遷移を生じさせる(N)プリミティブを(N+1)レイヤへ通知する。さらに、状態が変更された場合は、保持されている期待される出力をチェックし、その状態を変更し、必要ない場合は破棄する。
- 観測された出力を発生する遷移が存在しない場合は、SUTにプロトコル誤りが存在すると判断する。

2.2 課題

インテリジェントリンクモニタにでは、モニタが検出したPDUの順序に基づいて、SUTの動作を推定するために、SUTの中の各レイヤプログラム(IUT : Implementation Under Test)におけるPDUの処理とは異なる順序で処理することがある。これには以下のような原因が考えられる。

(1) 送信PDUと受信PDUのタイミングの相違

SUT間に伝送遅延がある場合は、転送方向の異なるPDUをモニタが検出する順序と、SUTが処理する順序が異なる場合がある。この例を図2のトランスポートレイヤの通信例で示す。SUT BはAK TPDUを送信した後に、DT TPDUを受信するが、モニタはDT TPDU、AK TPDUの順で検出する。このような状況は単に、固定的な物理回線の伝送遅延で生ずるだけでなく、その理由はより複雑である。

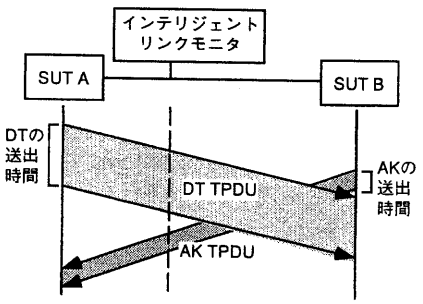

図2: 送受信PDUの検出順序の逆転

第一に、途中の通信装置の処理オーバヘッドや、処理方式によっても遅延が変動する場合がある。例えば途中に、転送すべきデータがある場合に動的にコネクションを確立するISDNルータが挿入されている場合、転送するPDUによってISDNコネクションの確立の遅延が追加される場合がある。

第二に、通信プロトコルの手順により処理順序が異なる場合もある。例えば、モニタはPDUの受信完了により、そのPDUを検出し処理するが、そのPDUを送信したSUTは、PDUの送信前にそのPDUの処理は終了している。このような送信側とモニタのPDU処理タイミングの相違は、フレーム長が長い場合や、下位レイヤでセグメンティングが行われた場合に問題となる。また、下位レイヤのフロー制御によりウィンドウが閉じていた場合は、注目しているレイヤがPDUを送出しても、下位レイヤでバッファリングされ、実際に回線に送出される時刻は遅れてしまう。

(2) PDUの紛失

伝送誤り、または途中の通信装置や受信側のバッファオーバフローにより、モニタが検出したPDUが、受信側のSUTで処理されない場合が考えられる。同様に、送信側SUTが送信したPDUが誤って、モニタでも受信側SUTでも検出されない場合も考えられる。

モニタは、SUTが、あるPDUに対して受信確認を送出したことを検出した時点で、はじめて、そのPDUの受信を確認できる。ただし、PDUに対する受信確認は、下位レイヤによりそのPDUを含む下位レイヤPDUの受信確認という形で行われる場合があることを考慮する必要がある。

(3) PDUの転送順序逆転

コネクションレスネットワークでは、PDUごとに独立にルーチングされ、到着順序が、送出順序と異なる場合がある。これにより、同一方向に転送されるPDUに対しても、送信側SUTにおける送出順序、モニタの検出順序、受信側SUTにおける受信順序が異なる場合がある。

3 PDU処理順序推定機能の設計方針

ルールベースプログラミングを用いて、インテリジェントリンクモニタに、SUTにおけるPDUの処理順序を推定する機能を実現するために、以下のような方針を立てた。

1. 上述の課題の内、固定的な伝送遅延と、送信側とモニタのPDU処理タイミングの相違については、決定的な手法を用いて解決し、既存のインテリジェントリンクモニタの(N)レイヤエミュレートモジュールの通常処理に組み込む。

2. その他の課題については、ルールベースプログラミングを利用し実現する。(N) レイヤエミュレートモジュールにおいて、プロトコル誤りが検出された場合に、ネットワークの不確定性の影響を考慮するために、本プログラムを起動する。

3. 各レイヤの状態遷移などの確定した情報は、決定的なプログラミングを用いて実現する。しかし、レイヤにまたがったフロー制御や受信確認の処理を実現するために、フロー制御で送信不可となった時間間隔や、受信応答の取れていないPDUの識別子などの情報を、必要に応じて、状態遷移の処理の実行中に、ルールベースのためのワーキングメモリに登録する。

4. ルールベースプログラミングのための言語としてOPS83 [5] を使用する。

4 詳細設計

4.1 処理の流れ

図3に、ルールベースプログラミングを用いたエミュレートモジュールの制御の流れを示す。エミュレートモジュールは、決定的アルゴリズムを実行する部分と、ルールベースプログラミングによる部分とに区分される。決定的アルゴリズム部では、PDU／プリミティブのイベントを検出すると、固定的な伝送遅延および送受信のタイミングの相違を考慮して、そのイベントの各SUTにおける処理時刻を計算する。さらにそれに従ってイベント列を並び替え、従来のインテリジェントリンクモニタで採用されている方法でSUTの動作をエミュレートする。ただし、その際、ルールベースプログラム部で必要な情報は、ワーキングメモリに追加するようにする。

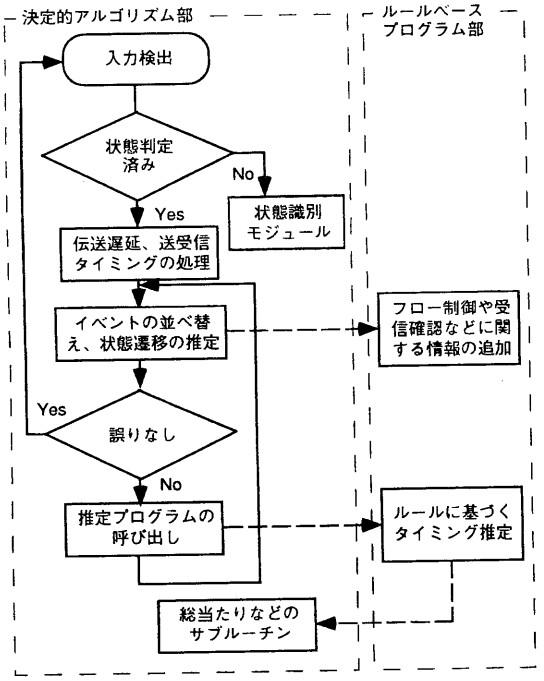

図 3: エミュレートモジュールの制御の流れ

決定的アルゴリズム部で、SUTのプロトコル誤りが検出された場合は、ネットワークの不確定性を考慮に入れるために、ルールベースプログラム部を呼び出す。その結果、可能性のあるイベント列を生成し、再度、SUTの動作をエミュレートする。また、ルールベースプログラム部は、全ての可能性の検索など特定のアルゴリズムをする場合は、決定的アルゴリズム部のサブルーチンを呼び出す。

4.2 決定的アルゴリズム部の処理

SUTが送受信したと検出されたPDUを、モニタが取り込んだ時刻から、伝送遅延や各レイヤの処理順序などを考慮してSUTの各レイヤが処理した時刻を推定する。その手順は以下のようになる。

(1) 固定的な伝送遅延に対する処理

フレーム取込みモジュールは、モニタでフレームの最後のバイトを検出した時点で、フレームを取り込む。例えば、図4に記述されたあるレイヤのTDT 1(シーケンス番号が1のDT TPDU)は時刻t2において、検出される。しかし、SUT Aにおける該当レイヤは、少なくともそのフレームが送信開始された時刻、すなわち、

t1(モニタがフレームの取り込みを開始した時刻) - dt(A)(SUT Aとモニタの間の伝送遅延)

には処理を終えている。一方、SUT Bにおける該当レイヤは、同じPDUを、SUT Bが受信を完了した時刻、すなわち、

t2(モニタがフレームの最後を検出した時刻) + dt(B)(SUT Bとモニタの間の伝送遅延)

より後に処理する。ここで、時刻t1は、t2より、検出したフレームの長さと回線の伝送速度により求められる。また、dt(A)およびdt(B)は別途測定するものとする。

(2) セグメンティングに対する処理

セグメンティングが行われた場合は、モニタは上位レイヤのPDU(例えば、図4におけるSDT1)を、そのPDUの最後のセグメントを運ぶ下位レイヤのPDUの最後を検出した時点、SDT1の場合は時刻t6に検出する。しかし、そのPDUを送信したSUTでは、そのPDUの最初のセグメントを含む下位レイヤのPDUの先頭が送出された時刻には、そのPDUの処理を終えている。すなわち、SUT AにおけるSDT1の処理時刻は、

t1 - dt(A)

である。一方、SUT B での SDT1 の処理時刻は、

$$t6 + dt(B)$$

となる。

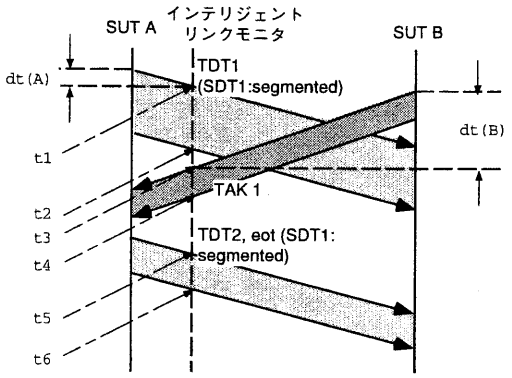

図 4: 時刻推定の例

4.3 ルールベースプログラム部の処理

(1) エレメントのデータ型

OPS83 において使用される、ワーキングメモリ上のエレメント [5] として、次のようなデータ型を導入する。

```
type eventSeq = element
  ( ....
    Events      : array(MAX: event) ;
    .... )
type event = record
  ( ....
    Id          : eventId ;
    Earliest    : time ;
    Estimated   : time ;
    EventInfo   : .... ;
        -- イベント名、パラメータ値等
    LossProced  : logical ;
    Lossed      : logical ; .... )
type unacked = element
  ( Id          : eventId ;
    Estimated   : time ;  )
type windowClosed = element
  ( LayerId     : .... ;
    From        : time ;
    To          : time )
```

ここで、eventSeq エレメントは推定されたイベント発生順序を示すデータ型で、各イベントは時刻、イベント情報などを含む。ここで、Estimated は決定的アルゴリズム部で推定された生起時刻である。Earliest, LossProced, Lossed などの各要素はルールベースプログラム部で使用される。また、unacked エレメントはある SUT が受信する PDU を検出した時点で、それが受信確認されていないことを表すために使用される。さらに、windowClosed エレメントは、あるレイヤにおいてウィンドウが閉じてデータの転送が止められていた時間を表すために使用される。

(2) 決定的アルゴリズム部からの呼び出し

決定的アルゴリズム部において状態遷移のエミュレートなどを行っている際に、ルールベースプログラム部の処理において必要となる情報は、エレメントとしてワーキングメモリに入れる。具体的には次のような処理を行う。

- あるイベント発生順序でプロトコル誤りを検出すると、対応する eventSeq エレメントをワーキングメモリに入れる。
- レイヤ処理で、ある受信イベントを検出すると、unacked エレメントをワーキングメモリに入れる。同様に、その受信イベントの応答に相当する送信イベントを検出すると、対応する unacked エレメントを取り除く。
- あるレイヤにおいて、フロー制御によりユーザデータを持つ PDU を送信できなくなると、次に、ウィンドウが開いてデータが出力された時点で、閉じた時刻と、開いた時刻を規定した windowClosed エレメントをワーキングメモリに入れる。

(3) タイミング推定のためのルール

決定的プログラム部においてプロトコル誤りが検出されると、次のようなルールを用いて、ネットワークの不確定性を考慮して、別の PDU 処理タイミングを推定する。

```
rule PduLoss
  {
    &1 (unacked);
    &2 (eventSeq including receiving Events
        with LossProced = 0b and Id = &1.Id);
    [currenttime - &1.Estimated]
  -->
    find &I such as &2.Events[&I].Id = &1.Id;
    modify &2 (Events[&I].LossProced = 1b);
    try eventSeq with Events[&I].Lossed = 1b;
  }
rule WindowClosed
  {
    &1 (windowClosed);
    &2 (eventSeq including sending Events
        with Estimated = &1.To);
  -->
    find &I such that &2.Events[&I].Estimated
        = &1.To;
    modify &2 (Events[&I].Earliest = &1.From);
```

```
    try new eventSeq in which Events[&I] may
        happen as early as &i.From;
}
```

第一のルールは、決定的アルゴリズム部の状態遷移の処理において、受信確認が取れていないPDUが存在することが指摘されており、かつそれがeventSeqエレメント中に含まれる場合に、そのPDUが紛失したというPDU処理順序を新たに考慮するためのものである。ここで、PDUの選択は最も古いものを優先するように設定する。

その後、決定的アルゴリズム部を用いて、紛失したPDUを考慮し、最下位のレイヤからエミュレートをやり直し、SUTの動作を調べる。PDUの紛失により、SUTの動作がプロトコル誤りなくエミュレートできた場合は、それをPDU処理順序の1つの候補として扱い、さらに、PDU紛失を仮定したことを、新たなエレメントとして登録する。

第二のルールは、下位レイヤのフロー制御によりウィンドウが閉じた時刻があり、かつウィンドウが開いた時刻にeventSeqエレメントの中のPDUが送信されていれば、そのPDUは下位レイヤにより送出が止められていたと考えられ、起こりえた最も古い発生時刻を、ウィンドウが閉じた時刻に設定する。

その後、決定的アルゴリズム部において、想定するレイヤにおいて、PDUの処理順序の入れ換えを行い、そのレイヤの動作をエミュレートし直す。プロトコル誤りなくエミュレートできた場合は、それをPDU処理順序の1つの候補として扱う。

5 考察

本検討では、次のような目的で発見的なアルゴリズムが規定可能な、ルールベースプログラミングを用いた。すなわち、相互接続試験システムを実際のネットワーク環境で動作させた場合は、ネットワークの伝送遅延などにより、検出したPDUの順序と、SUTの中の各レイヤのプロトコルプログラムが実際に処理した順序が異なる場合がある。このような場合に対処するためには、ルールベースプログラミングを用いた発見的アルゴリズムが有効である。

さらに、プロトコルの相互接続試験システムにおいては、以下の理由で、ルールベースプログラミングが有効であると考えられる。

- 相互接続試験システムにおいては、SUTの動作を検出されたPDUのシーケンスから推測する必要がある。場合によっては、あるPDUのシーケンスに対して、複数の動作を取りうることが考えられる。このような最適な動作を予想するためには、ルールベースプログラミングが有効であると考えられる。
- ルールベースを用いて推定された、PDUの送受信タイミングのずれ、伝送誤りなどの結果を、再度評価して、その推定が妥当であったかを評価するためにもルールベースプログラミングが有効であろう。

6 おわりに

本稿では、通信回線上を監視して得られたPDUシーケンスを用いて、通信システムの動作を推測するプロトコル相互接続試験に対して、ルールベースプログラミングを適用する方法について検討した。具体的には、伝送遅延や伝送誤りを持つ実際のネットワーク環境で使用可能とするために、監視したPDUシーケンスから、被試験システムの処理したシーケンスを推定するためにルールベースプログラミングを用いることが有効であることを示した。最後に日頃御指導頂くKDD研究所 村上所長に感謝する。

参考文献

[1] O. Rafiq and R. Castanet, "From conformance testing to interoperability testing," Proc. of the 3rd International Workshop on Protocol Test Systems, pp.371-385, 1990.

[2] K. Takahashi, et.al,, "Design and Implementation of an Interconnectability Testing System - AICTS," Proc. of the 7th International Workshop on Protocol Test Systems, pp.119-134, 1994.

[3] T. Ogishi, et.al., "Design of Intellignet OSI Protocol Monitor," Proc. of the 8th International Workshop on Protocol Test Systems, pp.351-366, 1995.

[4] 橋本他, "履歴診断得キスポートシステムのための状態推定方式の提案," 信学技法, AI94-4, May 1995.

[5] 田中, 下井, "エキスパートシステム構築の方法," パーソナルメディア社, July 1987.