

時間変化要素を含む分散型通信システムの記述法

太田 賢 渡辺 尚 水野 忠則

静岡大学工学部

分散型通信システムにおいて現在、マルチメディアアプリケーションが注目を集めているが、マルチメディアを扱う際に問題となるのは、その時間変化要素を再生時に保証することである。マルチメディア処理は、各メディアが再生するタイミングを絶対的、あるいは他のメディアに対し相対的に持つような、時間制約を持った並行プロセスである。本研究は、分散システムにおける通信サービス/プロトコルの仕様記述のために開発された形式記述技法を、マルチメディアアプリケーションの仕様記述に適用するものである。また、各メディア再生の時間関係の要求を満たすかの確認及び、そのアプリケーションの動作をシミュレータによりテストする。

Specification methodologies for distributed communication system including timed primitives.

Ken Ohta Takashi Watanabe Tadanori Mizuno

Faculty of Engineering, Shizuoka University
3-5-1, Johoku, Hamamatsu, 432 Japan

In distributed communication systems, many multimedia applications have been proposed and implemented. This paper applies FDT to the specifications of multimedia applications. Each of multimedia data has its timing for playing back absolutely or relatively and works in parallel with other multimedia data. This paper uses LOTOS-T and *T-PROMELA* to specify the multimedia system including timed primitives. Furthermore it uses *T-PROMELA* simulator to test user requirements for playing multimedia data and to trace the behavior of multimedia applications.

1 はじめに

分散型通信システムにおいて現在、マルチメディアアプリケーションが注目を集めている。その用途はTV会議、遠隔授業、マルチメディアデータベースアクセス、遠隔医療など幅広いものである。

マルチメディアを扱う際、問題となるのが時間変化要素 [1] である。マルチメディア処理とは複数メディアが時間軸に沿って並行に再生されるものであるが、各メディアはそれぞれ再生するタイミングを絶対的、あるいは他のメディアに対し相対的に持つような、時間制約を持った並行プロセスである。さらに分散環境下のマルチメディア通信においては、通信制御も加えたより複雑な機構が必要となる。その音声、映像通信等のマルチメディアアプリケーションに関しては、多くの制御機構が提案されている [2][3]。

本研究は、分散システムにおける通信サービス/プロトコルの仕様記述のために開発された形式記述技法 [4] を、マルチメディアアプリケーションの仕様記述に適用するものである。以下 2 章で時間拡張型形式記述言語について、3 章でマルチメディアの時間変化要素とその記述について述べ、4 章で *T-PROMELA* によるマルチメディアシステムへの適用を行う。5 章でまとめと今後の課題を述べる。

2 実時間型形式記述法

従来の通信プロトコル/サービスは、FDT の代表的な LOTOS の名が示すように時間順序を主体としたものであった。しかし近年、マルチメディア/FA などの分野において、マルチメディア同期や制御用通信などの実時間的なシステムの形式的記述が必要となってきている。LOTOS に関しては多くの時間拡張版が提案されているが、本章では、ISO/IEC/JTC1/SC21 のワーキングドラフト初版の LOTOS - T [5] と、我々が提案した *T-PROMELA* [6] について紹介する。*T-PROMELA* は、形式記述法の一つである *PROMELA* [7] を時間特性について拡張したものである。

2.1 LOTOS-T

LOTOS-T は LOTOS に、新たな時間に関するシンタックスを追加するとともに、これまでの表記のセマンティクスを一部定義し直している。

● 追加されたシンタックス

- $a \{ t \}; B$ アクション a は時間 t が経過した時点でまさにその時生起する。この時環境が a の生起をブロックした場合、 a は生起し得

ない。ただしその場合でも時間経過はブロックされない。

- $\text{exit} \{ t \}$ プロセスの正常終了が時間 t の経過時間に生起する。

● セマンティクスの定義

- $a; B$

* LOTOS における意味: アクション a が起きてプロセスは B として振る舞う。 a が生起する時間は不定。

* LOTOS-T における意味: アクション a は ASAP (できるだけ早く) に生起する。 a の生起後プロセスは B として振る舞う。 a の生起を環境がブロックしても時間は経過する。

- exit

* LOTOS における意味: プロセスの正常完了が生起する。いつ生起するかは不定。

* LOTOS-T における意味: プロセスの正常完了は ASAP に生起する。

タイムアウトの記述例を以下に示す。

$\text{ack}; \text{send_next_msg} [] i\{5\}; \text{retransmission}$

ack が生じたら send_next_msg へ移行する。 ack が生じる前に 5 単位時間経過したら retransmission へ移行する。

2.2 T-PROMELA

PROMELA (Protocol MEta LAnguage) は、プロトコル設計用の言語である。送受信や非確定処理のためのガードドコマンド、セレクトコマンドは CSP、変数の宣言や式などは C 言語の影響を受けている。*T-PROMELA* は *PROMELA* に対し、以下のような記述を加えた。

- 時間型の変数: 時間を記述するための変数で、単位はクロックである。クロックは仕様内で実時間に対応付けられる。

例: $\text{time timeout} = 5000$

- 現在の時間を示す特別変数 gtime : シミュレーション開始時からの時間経過を示すシステム予約の変数

例: $\text{cur_time} = \text{gtime}$

- 文実行時間記述: 明示的な実行時間の記述

例: $\text{channel! data} @ 10$

チャンネルに data を送信するのに 10 単位時間 (クロック) かかる。

これらシンタックスの拡張の応用として、タイムアウトの記述例を以下に示す。

```
chout ! data ;
st_time = gtime ;      タイマのスタート
do
:: chin ? ack -> break    確認応答受信
:: (gtime-st_time>5000)-> タイムアウト
    chout ! data          再送
od
```

3 マルチメディア時間変化要素とその記述

マルチメディア情報の再生を行う時に問題となるのは、各マルチメディア情報の時間変化要素を正確に再現することである。時間変化要素は、動作定義の要素と時間関係に分けられる。以下でそれについて述べると共に、形式記述法を適用した記述例を示す。

なお、*T-PROMELA* の記述内では1クロックを1(ms)と定義したと仮定する(*T-PROMELA* においては、1クロックが時間の最小単位であり1クロックずつ時間が進む。ユーザは記述において1クロックを実時間、例えば1ms、1sec、1minに対応付けることができる)。また、メッセージの型を以下のように定義した。

```
mttype = {START, PAUSE, RESUME, TERMINATE,
LOCATION}
```

LOTOS-T 記述では、これらメッセージは送信側と受信側の同期するイベントとして扱われる。

3.1 動作定義の要素

1. イベントの順序：イベントとは画像、音声などの動作(再生)、さらにユーザの制御(早送り、停止など)を指す。イベントの順序により、アプリケーションの動作が規定される。ただしマルチメディアアプリケーションの記述の場合、イベントの順序だけでなく実時間的規定も必要である。
イベントの順序の記述については特に時間の概念がないので省略する。
2. 開始時間と終了時間：マルチメディアデータには、時間制約があるものとなないものがある。例えば、ニュースなど時間が正確に決まっているものは、開始時間と終了時間を指定し、さらにそれを守る必要がある。
1分間の映像を19:00に再生を開始し、正確に1分間で終了することを要求する仕様を記述する(シミュレーション開始時刻を0:00に設定しているとする)。以下に、*T-PROMELA*、*LOTOS-T*の順にこの仕様を示す。

T-PROMELA

```
#define JUSTIME 19*60*60*1000
proctype control(chan sig)      制御プロセス
{
    sig: 再生プロセスとの通信チャンネル
    time start_time ;      1時間は60*60*1000(ms)
    (gtime == JUSTIME )->      19:00  か?
    start_time = gtime ;
    sig ! START ;            再生プロセスへ開始MSG
    sig ? TERMINATE ;        再生プロセス終了
    assert( gtime - start_time == 1*60*1000 )
}                               正確に1分間で終了するという表明
```

LOTOS - T

```
process control[ START, TERMINATE, VIORATION ]:
exit :=
i{19*60*60*1000};              (* 19:00 に生起 *)
START{0};                      (* 再生プロセスへ開始MSG *)
i{60*1000};                    (* 1分間待つ *)
( TERMINATE{0} ; exit          (* 1分間で終了した *)
  □ VIORATION ; exit ) (* 要求を守れなかった *)
endproc
```

3. 同期点での同期：メディアの早送り、一時停止などのユーザの制御にも対応して、メディア間同期を実現するため、同期点を決める必要がある。

同期間隔を1秒間として、メディアの一時停止、再開などのユーザインタラクションにも対応して、メディア間同期を守ることを目的とする制御プロセスを記述する。以下に、*T-PROMELA*、*LOTOS-T*の順にこの仕様を示す。

T-PROMELA

```
proctype control(chan sigA, sigB, user)
{
    sigA, sigB: 2つの同期プロセスへのチャンネル
    user: ユーザインタラクション

    time sync ;
    sync = gtime + 5*1000 ;      同期間隔5秒間
    sigA ! START ; sigB ! START ;
    do
        両再生プロセス開始
    :: ( gtime>=sync ) ->      再生位置同期
        sync = gtime + 5*1000 ; 再生位置指定
        sigA ! LOCATION ; sigB ! LOCATION
    :: user ? PAUSE ->          一時停止
        sigA ! PAUSE ; sigB ! PAUSE ;
        user ? RESUME ->        再開
        sigA ! RESUME ; sigB ! RESUME
    od
}
```

LOTOS - T

```
process system[START, LOCATION, PAUSE, RESUME]:
noexit:=
```

```

control[START,LOCATION,PAUSE,RESUME]
| [ START, LOCATION ] | playA[ START,LOCATION ]
| [ START, LOCATION ] | playB[ START,LOCATION ]
where
process control[START,LOCATION,PAUSE,RESUME]:
noexit := (* 制御プロセス *)
START ; (* A,B 再生開始 *)
ctrlsub[LOCATION,PAUSE,RESUME]
where
process ctrlsub[LOCATION,PAUSE,RESUME]:
noexit :=
LOCATION{5*1000}; (*5 秒間隔で同期 MSG *)
ctrlsub[LOCATION,PAUSE,RESUME]
[] PAUSE ; RESUME ; (* 一時停止、再開 *)
ctrlsub[LOCATION,PAUSE,RESUME]
endproc
endproc
endproc

```

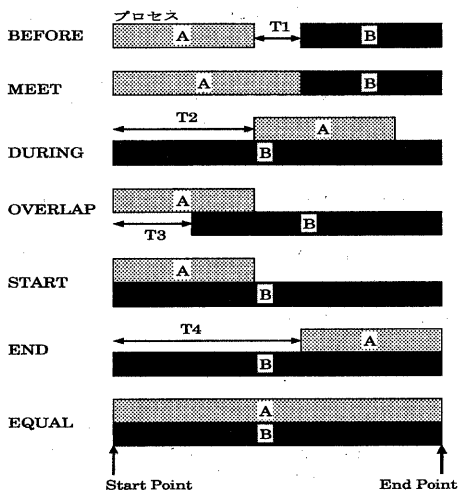


図 1: 時間関係の概念

3.2 時間関係

図 1に、イベントの基本的な時間関係を示す。A、B はマルチメディア再生プロセス (図 2) である。これらは上位の制御プロセスからの制御メッセージにより、動作を決定する。制御プロセスはシナリオに基づき、再生プロセスに制御メッセージを送り続ける。さらに、ユーザからの制御要求があった場合も同様にそれを伝える。

例として BEFORE 関係を再生する制御プロセスを、*T-PROMELA*、*LOTOS-T* を用いて記述する。以下がその仕様である。

T-PROMELA

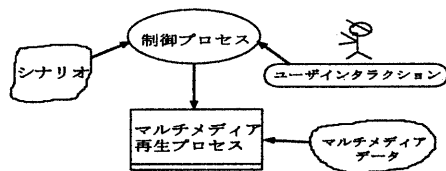


図 2: マルチメディア再生プロセス

```

proctype before(chan sigA,sigB)
{
    sigA,sigB: 再生プロセスとの通信チャンネル
    time t ;
    sigA ! START ;          プロセス A 再生開始
    sigA ? TERMINATE ;
    t = gtime ;
    ( t + T1 <= gtime ) ->  T1 待ち
    sigB ! START ;          プロセス B 再生開始
    sigB ? TERMINATE ;
}

```

LOTOS - T

```

process before[ST_A,TRM_A,ST_B,TRM_B]:exit :=
sig[ST_A,TRM_A] | [ST_A,TRM_A] | play[ST_A,TRM_A]
>> i{T1}; (* T1 待ち *)
sig[ST_B,TRM_B] | [ST_B,TRM_B] | play[ST_B,TRM_B]
where
process sig[START,TERMINATE]:exit :=
START ; TERMINATE ; exit
endproc
process play [ START, TERMINATE]:exit :=
hide media_play in
START ; media_play ; (* 再生 *)
TERMINATE ; exit
endproc
endproc
endproc

```

4 マルチメディアシステムへの適用

本章では、3章で示した記述を実際のマルチメディアシステムに適用し、仕様記述、動作確認、要求検査を行う。使用する形式記述技法としては *T-PROMELA* を選んだが、それは、*LOTOS-T* にはまだその処理系 (検証やシミュレータのツール) がいないことと、時間型の変数を利用できる *T-PROMELA* の方が記述がしやすいという理由からである。

4.1 仕様記述

図 3 のようなシナリオ再生を行うマルチメディア再生アプリケーションを記述する。マルチメディアアプリケーションの構成は図 4 のとおりである。以下に、メイン制

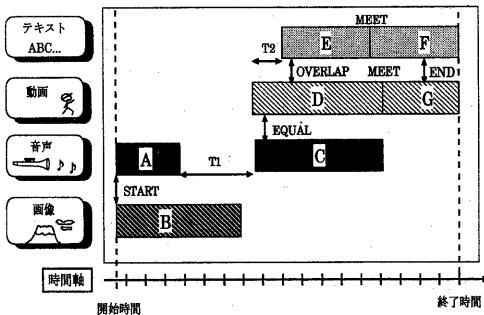


図 3: シナリオ例

御プロセス、各メディア制御プロセス、再生プロセスの記述を示す。

```

1 #define START_TIME 19*60*60+1000
2 #define END_TIME (19*60+10)*60+1000 ;
3 /* 開始 19:00 終了 19:10 */
4 #define T1 90*1000    T1 は 90 秒
5 #define T2 60*1000    T2 は 60 秒
6 mtype = { START, TERMINATE, IND } ;
7 /* チャネル宣言 */
8 chan sigA = [0] of { int } #0 ;
9 chan sigB = [0] of { int } #0 ;
10 chan ch_snd = [0] of { int, chan } #0 ;
11 chan ch_img = [0] of { int, chan } #0 ;
12 .....
13 proctype control()    ★ メイン制御プロセス ★
14 {
15     time t ;
16     (gtime == START_TIME) -> 開始時間か?
17     ch_snd ! START(sigA) ;    A 再生開始
18     ch_img ! START(sigB) ;    B 再生開始
19     ch_snd ? IND(sigC) ;      C より指示
20     ch_vid ! START(sigD) ;    D は C に同期
21     ch_vid ? IND(sigD) ;      D より指示
22     t = gtime ;
23     ( gtime == t+T2 ) ->    D の開始後 T2 後に
24     ch_txt ! START( sigE ) ; E を再生開始
25 }
26 proctype sound_ctrl() ★ 音声制御プロセス ★
27 {
28     time t ;
29     ch_snd ? START( sigA ) ; 再生指示
30     t = gtime ;
31     sigA ! START ;           A 再生開始
32     sigA ? TERMINATE ;       A は完了?
33     ( gtime == t+T1 ) ->    A 完了後 T1 後に
34     sigC ! START ;           C 再生開始
35     ch_snd ! IND( sigC )     C 再生通告

```

```

36 sigC ? TERMINATE
37 }
38 proctype image_ctrl()    ★ 画像制御プロセス ★
39 .....
40 proctype play(sig)      ★ 再生プロセス A,B... ★
41 {
42     LOAD_DATA@LOAD_TIME ;    データロード
43     sig ? START ;
44     DELAY #0 ;               実際の再生までの遅延
45     MEDIA_PLAY@MEDIA_LENGTH ; メディア再生
46     sig ! TERMINATE ;
47 }

```

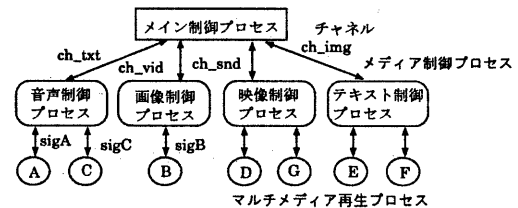


図 4: マルチメディア再生アプリケーションモデル

この仕様は、3つの仮定の上で記述されている。まず、各再生プロセスは、制御プロセスの再生指示メッセージを受信した後、すぐに再生を開始することを仮定している(44行目)。しかし、実際はそのメッセージ受信後から表示するのに全く遅延がないわけではない(ウィンドウを開くとか、データの転送など)。また、再生開始前の準備(42行目)、メディアの実際の再生の部分(45行目)を時間的に確定的な処理として扱っているが、実際はCPU、通信といった資源の状態に依存するものである。データロードに関しては、例えば通信によりデータを取得することを考えた場合、その転送時間は予測できても確定的であるとは言えない。メディア再生に関しては、一定の再生レートを維持するのはCPUの負荷などの要因から不可能であるから、実時間同期処理を行わない場合、やはり確定的な実行時間は分らない。さらに、映像を再生する場合、ディスクあるいはメモリに全データを読み込んでおくのは難しいから、随時、通信によりデータを取得すると共に再生を行うことになり、CPUと通信の両方の資源の状態に影響を受けることになる。本来はそれらのメカニズムを記述した上で、以下のシミュレーションを行うべきであるがそれは今後の課題としておく。

4.2 動作確認、要求検査

ユーザの要求は以下の3つがあるとする。

1. BがCの再生開始時には終了していること
2. このシナリオが19:10に正確に終了すること

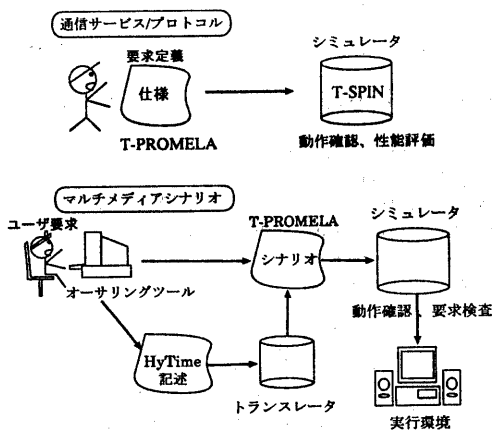


図 5: マルチメディアシナリオ記述システム

3. C と D は開始点同期、終了点同期すること。

検査のために以下のような検査プロセスを記述する。

```

1 proctype testing()
2 {
3   assert( end_time_B < start_time_C );
4   assert( scenario_end_time == END_TIME );
5   assert( start_time_C == start_time_D &&
6         end_time_C == end_time_D );
7 }

```

要求 1、2、3 に対し 4、5、6 行目の各表明文が対応する。start_time_?, end_time_? は A、B.. 等各再生プロセスが与える開始時刻、終了時刻であり、scenario_end_time は各再生プロセスの終了時刻の最も遅いものである。この検査プロセスも含めてシミュレーションを実行すると以下のような結果が報告される。1 の要求以外は満足されたこと、そして各再生プロセスの再生状況が分かる。

```

proc A *sound* [start 19:00:00]
proc B *image* [start 19:00:00]
proc A *sound* [end 19:02:00]
proc C *sound* [start 19:03:30]
proc D *video* [start 19:03:30]
proc B *image* [end 19:03:40]
.....
proc G *video* [start 19:07:30]
proc F *text* [end 19:10:00]
proc G *video* [end 19:10:00]
assert'end_time_B < start_time_C' : Violated.
assert'scenario_end_time == END_TIME': Satisfied.
assert'start_time_C == start_time_D' : Satisfied.

```

5 おわりに

マルチメディアアプリケーションの時間変化要素の仕様記述に、時間拡張をした形式記述技法である T-PROMELA、LOTOS-T を適用し記述を行い、それらの言語仕様及び記述能力がマルチメディアアプリケーションに対し有効であることが確かめられた。

また、マルチメディアアプリケーションのシナリオ再生を制御するプロセスの仕様を T-PROMELA で実際に記述し、その後シミュレータにより動作確認、要求検査を行うことで、そのシナリオがユーザーの要求通りに動作することを確かめることができた。

今後の課題として 2 つ挙げられる。まずは、図 5 のようなシステム環境を構築することである。仕様の記述はかなりの労力を必要とする作業であるから、マウスでアイコンを配置することによりシナリオを構築し、仕様記述を自動生成するようなオーサリングツールが必須である。あるいは、HyTime[8] の記述を読み込み、それから T-PROMELA の記述を生成することも考えられる。また、シナリオを読み込み実際に再生を行う実行環境を構築する必要がある。

もう 1 つは、4.1 節で述べたように、分散環境上のマルチメディアアプリケーションのメカニズムをより詳細に記述し、動作確認、要求検査をすることである。

参考文献

- [1] 坂下 善彦: マルチメディア処理の要素, 情報処理学会マルチメディア通信と分散処理, pp.183-190(1994.6).
- [2] 篠田、石井、桑名: クロスプラットフォームなコンピュータネットワーク上での音声通信の実現, 情報処理学会マルチメディア通信と分散処理, pp.17-24(1993.7).
- [3] 橋本、渡辺、柴田: パケット オーディオ・ビデオシステムの QoS 保証及び交渉機構について, 情報処理学会マルチメディア通信と分散処理, pp.67-72(1994.5).
- [4] 水野 忠則: プロトコル言語, カットシステム (1994.7).
- [5] Initial draft on Enhancements to LOTOS, part 6 ISO/IEC/JTC 1/SC21 Project JTC1.21.57(1993.11).
- [6] 太田、渡辺、水野: プロトコル記述言語 PROMELA の時間特性の拡張, 情報処理学会 第 48 回全国大会講演論文集 (1), pp.221-222(1994.3).
- [7] 水野 忠則: コンピュータプロトコルの設計法, カットシステム (1994.11).
- [8] 小町 祐史: マルチメディア/ハイパメディア情報交換の標準化動向, 情報処理, Vol.35, No.7(1994.7).