日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

形式手法を用いたバケット同期法の検証

永浦 尊信　千葉 勇輝　二木 厚吉

バケット同期法の正当性とは，複数ユーザ間でイベント受理の順序が同一に保たれることである．本論文では，形式手法を用いて，バケット同期法をモデル化し，正当性を検証するための枠組みを OTS/CafeOBJ を用いて構築する．バケット同期法の正当性を検証するためには，フレームというバケット同期法の持つ独特な時間の概念をモデル化する必要がある．そこでまず，フレームの概念を OTS/CafeOBJ の状態遷移としてモデル化する．その後，バケット同期法のモデルを作成し，そのモデルにおける正当性がどのような物か形式化する．

1　はじめに

バケット同期法 (Bucket Synchronization) [6] は，遅延のあるネットワーク通信を用いて複数のユーザ間でイベントの受理順序を共有するためのプロトコルであり，ネットワークゲームを中心に用いられている．バケット同期法を基礎にした応用プロトコル [1] [2] も作成されており，この分野の基礎的なプロトコルとなっている．しかしバケット同期法は，通信シミュレーションによってその正当性や効率性を論じられてこそいるが，形式化は行われていない．そのため，利用に対する条件が不明瞭であり，正確なバケット同期法の利用は困難である．そこで本論文では，バケット同期法を OTS/CafeOBJ [3] [5] [4] を用いて形式化することで，その正確な利用法を提示しプロトコルの応用を容易にすることを目的とする．

バケット同期法を用いるアプリケーションは，その内部でフレームという独特な時間の概念を用いている．そのため，フレームの概念を整理しなければ，バケット同期法も形式化することはできない．そこで副次的な目標として，フレームの概念を整理する．

これらを踏まえて，本論文は，下記の形式化と検証を検討する．

1. (**フレームの概念の整理**) バケット同期法を用いるアプリケーションに導入される時間の概念であるフレームについて考察を与える．
2. (**バケット同期法の形式化**) 1 で作成したフレームの形式化をふまえて，バケット同期法の形式化を行う．
3. (**正当性の定義と検証**) 2 までに得られた形式化の上で，どのような条件が満たされるならばバケット同期法が正当性を持つか考察し，定義する．また，この正当性の定義を用いた検証を検討する．

2　バケット同期法とフレーム

本節では，形式手法を用いてバケット同期法の正当性を検証するために，バケット同期法を用いるアプリケーションの構造を整理する．また，バケット同期法におけるフレームの概念を定義し，フレームの性質について説明する．ここで定義したフレームの概念に基づき，バケット同期法のプロトコルを定義する．

2.1　バケット同期法を用いるアプリケーション

バケット同期法を用いるアプリケーションは，以下

A Verification of Bucket Synchronization by Using Formal Method

Takanobu Nagaura, Yuki Chiba, and Kokichi Futatsugi, 北陸先端科学技術大学院大学情報科学研究科, School of Information Science, Japan Advanced Institute of Science and Technology.

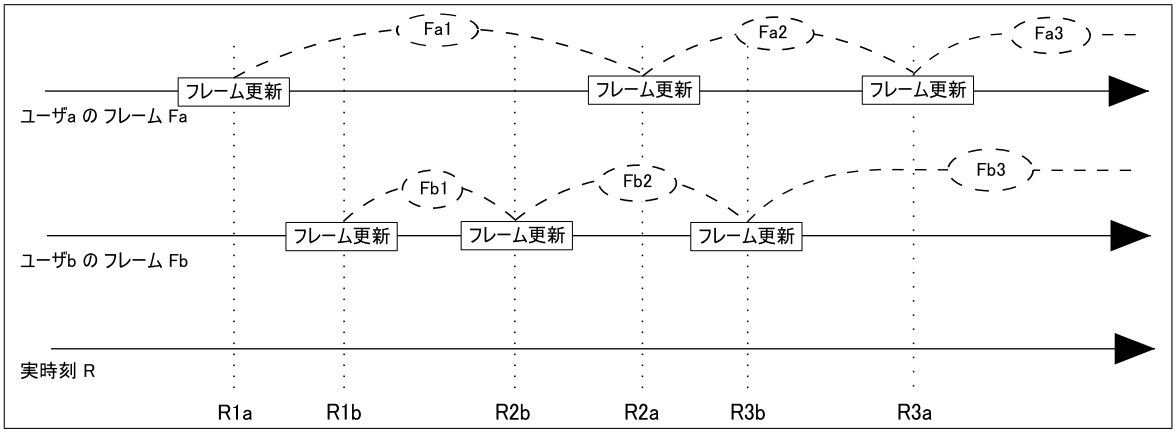

図 1　フレームの概念

に述べる構造をしている．

- 遅延のあるネットワークで接続された，複数のユーザが存在している．この全体をセッションと呼ぶ．
- セッションで情報を共有するために，すべてのユーザは，ユーザワールド，ユーザネットワーク，及び，ユーザバケットと呼ばれる情報を保持する．
- ユーザワールドは，ユーザが受理したイベントを受理したフレーム時間 (後述) 毎に区切って並べたリストである．
- ユーザワールドは，０個以上のイベントのリストを受理することによって更新される．
- 同一の要素を持ったイベントのリストを，同一の状態を持ったユーザワールドに対して受理すると，同一のユーザワールドが求まることが保証されている．
- ユーザネットワークは，ユーザに送信されているが，まだ受信していないメッセージの集合である．
- ユーザバケットは，ユーザが受信しているが，まだ受理していないイベントの集合である．

ここで，イベントとメッセージという二つの類似した概念が登場している．このイベントとメッセージの違いは，メッセージはイベントに送信時刻を付与し，ネットワークで送るデータのことであるという違いである．

2.2　フレーム

フレームは，バケット同期法を用いるアプリケーションに導入される時間管理の方式で，一般に以下に述べるような特徴を持つ．

- 離散的に時刻を刻む．この時刻の刻みを単純にフレーム，時刻の刻みによってユーザワールドを更新することをフレーム更新，開始から現在までのフレーム更新回数をフレーム番号，二つのフレームでフレーム番号の差をフレーム時間とそれぞれ呼ぶ．
- フレーム更新では，そのフレームにおけるイベントの受理作業を行うことにより，ユーザワールドを更新する．
- 基本的にフレームの長さは一定の実時間に対応した長さに調節される．しかし，実時間との対応と，フレーム更新の計算速度で，フレーム更新のほうが長い場合，フレーム更新に合わせてそのフレームは伸びる．つまり，ユーザワールドの整合性が，実時間との対応よりも優先される．

ユーザ a とユーザ b のいるセッションでのフレームの様子を図 1 に描いた．ここで注意するべきことは，互いに 1 フレーム目でも，ユーザ間でフレームの開始・終了の実時刻は一致していない ($\mathrm{R}_{na} \neq \mathrm{R}_{nb}$) ということである．そして，同一のフレーム番号では，同一のユーザワールドを全てのユーザが保有していることが要求されるが，その実時間は一致している必要はない．このことから，バケット同期法を用いるア

プリケーションにおいて重要なのは，実時刻で同時にメッセージを受理することではなく，同一のフレーム番号においてメッセージを受理することであることがわかる．

2.3 バケット同期法

バケット同期法では，次の方法によって，あるフレーム番号におけるイベントの受理を一致させる．また，イベントの発生と受理がそれぞれ別の操作であることに注意を要する．イベントが発生しても，受理を行わない限りユーザワールドは更新されない．

- 待ちフレーム時間 Wait を決定し，全ユーザ間で共有する．
- ローカルでイベントが発生した場合，直ちにリモートのユーザ全員に対して，送信時刻を付与したメッセージをブロードキャストする．自身はイベントに対して Wait を待機したのち，受理する．
- リモートからイベントメッセージを受信した場合，現在のフレーム番号と送信時のフレーム番号の差のフレーム時間 Delay を求め，Wait と Delay の差のフレーム時間待機したのち，そのメッセージに含まれるイベントを受理する．

図 2 では，ユーザ A の第 2 フレームで発生したイベントの受理時刻をユーザ A と B の間でバケット同期法が一致させる様子を示している．ここで横軸は，フレーム番号であるため注意を要する．

Delay が Wait を超える場合には，単純にバケット同期法を用いていると破綻する．そこで高い Delay が検出された場合，何らかの補償アルゴリズムを別途に利用したり，フレーム更新を停止して互いのメッセージを同期し終わるまで通信に専念するなど，実装には幅がある．しかし，いずれの場合もバケット同期法本体とは関係がないため，ここでは Delay は Wait 以下であることをバケット同期法の持つ仮定と考える．

3 OTS/CafeOBJ 上でのモデル化と検証

前節では，バケット同期法におけるフレームの概念，及び，バケット同期法のプロトコルを定義した．この定義に基づきバケット同期法の正当性を形式検証するために，OTS/CafeOBJ 手法を用いてバケット同期法の形式モデルを作成する．OTS/CafeOBJ において，状態は観測関数によって得られる観測値によって定義される．また，状態遷移は遷移関数によって行われる．そのため，観測関数と遷移関数を与えることによりバケット同期法をモデル化する．さらに，上記モデルに基づく形式検証について考察する．

3.1 オペレータとソートの定義

前節で示した特性を踏まえて，フレームおよびバケット同期法を OTS/CafeOBJ 上で記述した．以下はセッションの振る舞い仕様 SESSION から，オペレータの定義のみを抜き出した物である．

```
-- transition
op init :  Frame -> Session
bop send :  Session Uid Event -> Session
bop receive :  Session Uid Message -> Session
bop nextframe :  Session Uid -> Session

-- observations
bop wait :  Session -> Frame
bop frame :  Session Uid -> Frame
bop usernetwork :  Session Uid -> Network
bop userworld :  Session Uid -> UserWorld
bop userbucket :  Session Uid -> Bucket
```

ここで，いくつかのソートが導入されているが，それぞれ次のような意味を持つ．

- `Session` : セッションの状態を表す隠蔽ソート
- `Frame` : ペアノ整数で表現されたフレーム番号
- `Uid` : ユーザ ID
- `Event` : イベント
- `Message` : メッセージ．Event と Frame の組み
- `Network` : ネットワーク．Message のリスト
- `UserWorld` : ユーザワールド
- `Bucket` : Event を受理時刻まで保持するリスト

3.2 観測演算

OTS/CafeOBJ では観測演算と遷移演算を用いてモデルを記述する．オペレータのうち，観測演算は次のような意味を持つ．

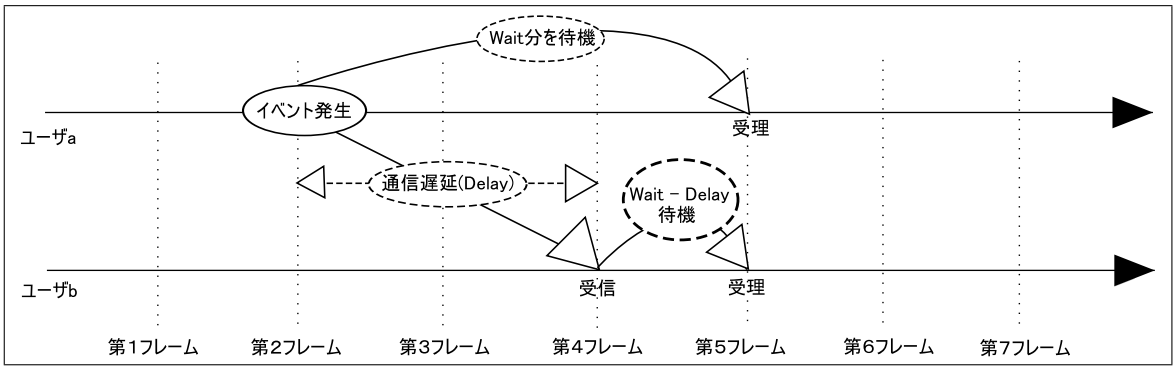

図 2　バケット同期法

- `wait(s)` : 共有されている Wait の値
- `frame(s, uid)` : uid のユーザの現在のフレーム番号
- `usernetwork(s, uid)` : uid のユーザの未受信 Message を持った Network
- `userworld(s, uid)` : uid のユーザの UserWorld
- `userbucket(s, uid)` : uid のユーザの Bucket

3.3　遷移演算

遷移演算は，OTS/CafeOBJ でのモデル化において中核的な位置をなす．そのため，以下では遷移演算について詳しくその意味を解説する．

3.3.1　init(delay)

delay を Wait として共有し，Session を開始したことを示す．init では，すべてのユーザの frame は 0 に，usernetwork，userworld，userbucket は空に初期化する．また，wait の値は delay として初期化される．ここで，init の後には wait の値に触れるアクションは存在しない．そのため init(delay) から到達可能なすべての Session に対して，wait の値は delay に一致する．

3.3.2　send(session, uid, event)

uid のユーザが Event を発生させたことを示す．他のすべてのユーザの usernetwork に，Message を追加する．この Message は送信時刻を保持するために，event と frame(s, uid) の組みとして生成する．また userbucket(session, uid) に event を追加する．ここではイベントが発生しているが，受理は行わないため userworld は変更しない．さらに，frame の値は増やさず，そのままの値を保つ．これは，実時間が過ぎているとしても，あくまでフレーム更新によってのみフレーム番号は増えると考えるためである．

3.3.3　receive(session, uid, message)

uid のユーザが message を受信したことを示す．message を usernetwork(session, uid) から削除する．また，userbucket(session, uid) に message のもつ event を追加する．send の場合と同様に，receive においても frame の値は変更しない．また，ここでもイベントの受理は行わないため，userworld は変更しない．

3.3.4　nextframe(session, uid)

uid のユーザがフレーム更新を行ったことを示す．userbucket(session, uid) から受理すべき Event を探し出し，userworld(session, uid) に対して受理を行う．また，frame(session, uid) の値を次に進める．ここで，nextframe は usernetwork に対して，wait 値よりも古いメッセージが存在しない場合のみ発生する．これは，別途の保障アルゴリズムや，更新停止によって実現することを期待される条件で，本モデル化では仮定として導入している．この仮定は OTS/CafeOBJ 上で以下のように記述した．

```
ceq nextframe(session, uid) =
  session if not( c-nextframe(
    usernetwork( session, uid ),
    wait( session ) ) ) .
```

ただし，c-nextframe は wait よりも古いメッセージが Network に存在しないことを調べるオペレータである．

3.4 検証

この形式化の上で，バケット同期法に対する形式検証を行うことを考えた．バケット同期法の正当性は，セッションが継続している間，同一のフレーム番号で，すべてのユーザの持つユーザワールドが一致していることである．これを形式的に記述すると以下のようになる．

$$\forall s \in Session\ \exists s' \in Session\ \forall u1, u2 \in Uid.$$
$$s' \triangleright s \wedge frame(s, u1) = frame(s', u2) \rightarrow$$
$$userworld(s, u1) = userworld(s', u2)$$

ここで $s' \triangleright s$ は，s' から s に対して到達可能であることを示す．検証は，この命題を証明するような証明譜を作成することに帰着される．その証明譜は現在開発中である．

4 まとめと今後の課題

OTS/CafeOBJ 上で，バケット同期法および，フレーム概念の形式化を行った．その過程において，バケット同期法の持つ明示されていなかった条件や構造を抽出できた．また，正当性を命題の形で明確に定義し，正当性の検証を CafeOBJ における証明譜の作成に帰着した．証明譜の作成と，それに基づく考察が今後の課題である．

参 考 文 献

[1] Eric Cronin, Anthony R. Kurc, Burton Filstrup, and Sugih Jamin. An efficient synchronization mechanism for mirrored game architectures. *Multimedia Tools and Applications*, 23(1):7–30, 2004.

[2] Stefano Ferretti and Marco Roccetti. Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 405–412, New York, NY, USA, 2005. ACM.

[3] Kokichi Futatsugi. Verifying specifications with proof scores in cafeobj. In *ASE*, pages 3–10. IEEE Computer Society, 2006.

[4] Elie Najm, Uwe Nestmann, and Perdita Stevens, editors. *Formal Methods for Open Object-Based Distributed Systems, 6th IFIP WG 6.1 International Conference, FMOODS 2003, Paris, France, November 19.21, 2003, Proceedings*, volume 2884 of *Lecture Notes in Computer Science*. Springer, 2003.

[5] Kazuhiro Ogata and Kokichi Futatsugi. Proof scores in the ots/cafeobj method. In Najm et al. [4], pages 170–184.

[6] Multi-Player Game On, Laurent Gautier, and Christophe Diot. Design and evaluation of mimaze,. In *Proceedings of IEEE Multimedia Systems Conference*, pages 233–236, 1998.