リカバリオーバヘッドに基づいたマルチメディアチェックポイントプロトコル *

2N−7

東京電機大学理工学部情報システム工学科†
長田 慎司　平賀 研吾　森田 義徳　桧垣 博章‡ §

1 背景

ネットワーク技術とコンピュータ技術の発達にともない、ビデオオンデマンドや、テレビ会議システム、遠隔授業などのマルチメディアネットワークシステムの開発が進められている。ここでは、システムの一部が故障しても、ユーザへのサービス提供が停止しないことが要求される。これを満たす手法の1つとしてチェックポイントリカバリがある。チェックポイントとは、ネットワークに接続されたコンピュータ上でアプリケーションプログラムを実行する各プロセスが正常動作中の状態情報を安定記憶に保存したものである。あるプロセスで故障が発生した場合には、すべてのプロセスがチェックポイントの状態情報を安定記憶から読み出し、この状態からアプリケーションの実行を再開する。ここでシステムがアプリケーションの実行を正しく再開するためには、すべてのプロセスが設定したチェックポイントによって一貫性のあるグローバル状態が定められなければならない [1]。従来のデータ通信アプリケーションにおいては、紛失メッセージと孤児メッセージのないグローバル状態を一貫性があるという。また、メッセージ通信イベントは瞬時に終わると仮定している。チェックポイントはイベントが発生していないときにのみ設定される。メッセージ通信イベント発生中にチェックポイント取得を要求された場合には、これが終了するまで待たなければならない。

2 マルチメディアチェックポイントの一貫性

マルチメディアネットワークでは、送受信されるメッセージのサイズは大きく、メッセージ通信イベントを瞬時に終えることができない。そのため、従来のチェックポイント設定手法を適用すると、チェックポイントの設定をメッセージ通信イベント終了まで延期しなければならない。メッセージ通信イベント中に故障が発生すると、メッセージを最初から再送信することになる。そこで、我々はメッセージ通信イベントの発生中にプロセスがチェックポイントを設定した場合の一貫性について提案した [2]。データ通信ネットワークを対象とした従来の一貫性に対しては、上位互換性を保証した。すなわち、紛失メッセージや孤児メッセージのあるグローバル状態の一貫性は0であり、それらが存在しない状態の一貫性は1である。メッセージ通信イベントの発生中に設定されたチェックポイントの一貫性については、従来のデータ通信ネットワークを対象としたチェックポイントでは定義されていない。我々は、マルチメディアメッセージが複数のパケットからなることに注目した。メッセージ通信イベントは、パケット通信イベントのシーケンスからなる。チェックポイントは、2つの連続するパケット通信イベントの間に設定される。これによって、紛失パケットと孤児パケットが定義される。紛失パケットは、リカバリ時に再送信させることがなく紛失してしまう。マルチメディア通信アプリケーションでは、マルチメディアメッセージの一部が失われても受理することができる。そこで、紛失パケットが少ないほどグローバルチェックポイントの一貫性は高く、多いほど低くなる新しい一貫性の評価方法を定めた。ここでの評価値の値域は [0, 1] としている。

3 回復時間の評価

前章で述べたように、マルチメディア通信ネットワークにおいては、グローバルチェックポイントの一貫性は紛失パケットによって定まり、孤児パケットには依存しない。しかし、孤児パケットは、故障からのリカバリ時の時間および通信オーバヘッドを増加させる。孤児パケットは、プロセスがチェックポイントからアプリケーションの実行を再開した後に送信元プロセス p_i によって再送信される。また、孤児パケットの送信先プロセス p_j では、再送信されたパケットは既に受信済みの状態にあるため、これらを破棄する。p_i は、すべての孤児パケットの送信を終えることによってリカバリの手続きを終え、p_j はすべての孤児パケットの受信と破棄を終えることによってリカバリの手続きを終える。以上により、グローバルチェックポイントの新たな評価方法としてリカバリ時間を考えることができ、そのためには孤児パケットの再送受信に要する時間を計測すればよい。

[回復時間の評価]
本論文ではリカバリ時に再送信されるパケットの処理時間を回復時間として評価する。

各チャネルに存在する孤児パケットの送受信を終えるために要する時間は、孤児パケットの数に依存する。さらに、ある孤児パケットの処理を終えるためには、それに因果先行する孤児パケットの処理を終えていなければならない。図1において、プロセス p_4 がチェックポイント c_4 からリスタートするためには、m_3 の孤児パケットの送受信を終えなければならない。この時間を R_3 とする。一方、プロセス p_3 が c_3 からリスタートし、m_3 の孤児パケットの送信を行なうためには、m_2 の孤児パケットの送受信を終えなければならない。この時間を R_2 とする。さらに、プロセス p_2 が c_2 からリスタートし、m_2 の孤児パケットの送信を行なうためには、m_1 の孤児パケットの送受信を終えなければならない。この時間を R_1

---

*Multimedia Checkpoint Protocol Based on Recovery Overhead
†Tokyo Denki University
‡Shinji Osada, Kengo Hiraga, Yoshinori Morita and Hiroaki Higaki
§{shinji,hira, mine, hig}@higlab.k.dendai.ac.jp

とする。以上より、p_4 の回復時間は、$R_1 + R_2 + R_3$ となる。このように、各プロセスの回復時間は、すべてのプロセスが同時にチェックポイントからアプリケーションプログラムの実行を再開したとき、孤児パケットの送受信を終えるまでの時間となる。そこで、システムの回復時間は、すべてのプロセスの回復時間の最大値として評価される。

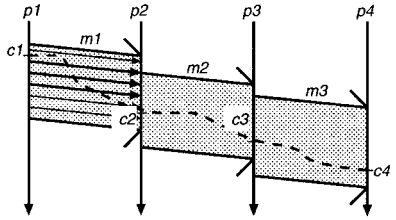

図 1: 回復時間評価

4 チェックポイントプロトコル

ここでは、前章で提案した回復時間の評価に基づいたチェックポイントプロトコルを提案する。マルチメディアアプリケーションでは実時間性を要求される場合が多く、正常動作時のオーバヘッドが小さいことが求められる。このため、チェックポイントを取得する際にプロセスがアプリケーションの実行を一時中断することは許されない。本論文で提案するプロトコルでは、チェックポイントの設定を要求し、グローバルチェックポイントの一貫性、回復時間を評価するコーディネータープロセスが存在すると仮定する。

4.1 一貫性評価に基づいたプロトコル

我々は紛失パケットの発生確率を減少させることによって一貫性を向上させるプロトコルを提案した [2]。このプロトコルでは、以下のような手順でチェックポイントが設定される。

[チェックポイントプロトコル $\mathcal{P}_\mathcal{E}$]

1) コーディネータープロセス p_c がアプリケーションで要求される一貫性を指定し、チェックポイント設定要求 *Req* を各プロセスに送信。
2) *Req* を受け取った各プロセスは、仮チェックポイントを設定し p_c に確認応答 *Ack* を返信する。ただし、プロセスがメッセージを受信中であるならばチェックポイントプロトコルを終了させるまでの時間 τ から p_c との間の RTT2δ を引いた時間 $\tau - 2\delta$ だけチェックポイントの設定を遅延させる。また、このとき、各プロセスは p_c が一貫性を計算するのに必要な情報を *Ack* に付加する。
3) すべてのプロセスから *Ack* を受信したならば、p_c は一貫性を評価する。これが要求される一貫性より高いならば *Done* メッセージを、要求を満たしていなければ *Cancel* メッセージを各プロセスに送信し、チェックポイントプロトコルを終了する。

このプロトコルでは、2) のようにメッセージを受信中のプロセスがチェックポイントを設定するのを遅らせることで紛失メッセージの発生確率を減少させ、高い一貫性を実現している。

4.2 拡張チェックポイントプロトコル

3 章で述べたように、リカバリ時に大量の孤児パケットが存在すると、実時間処理の要求を満たすことができなくなる。そこで、3 章の回復時間の評価に基づいたチェックポイントプロトコルを提案する。

[拡張チェックポイントプロトコル $\mathcal{P}_\mathcal{E}^+$] $\mathcal{P}_\mathcal{E}$ では、メッセージを受信中のプロセスがチェックポイントの設定を遅延させることによって、孤児パケットの発生確率を高めている。拡張プロトコル $\mathcal{P}_\mathcal{E}^+$ では、p_c が受信側プロセスがチェックポイントの設定を遅延することによってどれだけの孤児パケットが発生しているのかを把握することによって、メッセージ受信側プロセスの待ち時間を以下のように最適化する。

- 各プロセスは *Ack* メッセージを p_c に返信する際に既に送信 (受信) の終了したパケットの情報を付加する。
- あるチャネルで孤児パケットが発生していた場合、p_c はチャネルの両端のプロセスから受信した *Ack* に付加された情報によりそのチャネルの回復時間を算出する。このような回復時間をチャネルごとに保存しておく。
- p_c は *Req* メッセージをネットワーク上の各プロセスに送信する際に各チャネル毎の回復時間の平均値 $\bar{\tau}$ を付加する。
- *Req* を受けとったプロセスがメッセージを受信中であった場合、$max(\tau - 2\delta - \bar{\tau}, o$ だけチェックポイント設定を遅延させる。ただしこのとき、このプロセスが複数のメッセージを受信中であるならば各チャネルごとの回復時間の平均値の中で最も短い回復時間を $\bar{\tau}$ とする。

4 番目の拡張で、複数チャネルよりメッセージを受信している場合は受信を行なっているチャネルの統計情報の中から最小の回復時間を選択している。これは、最小の回復時間以外を選択するとそれ以下の回復時間を要するチャネルにおいて、紛失パケットの発生確率が増加するためである。これらの拡張を行なうことにより、紛失パケットの発生確率を増大させることなく、かつ孤児パケットの発生による回復時間の増大も減少させることができる。

5 まとめと今後の課題

本論文で提案した手法では、紛失パケットによる一貫性の低下を防ぐ手法に加え、孤児パケットの再送信に要する回復時間を減少させる手法を提案した。今後は、今回提案した手法の評価を MPEG-2 データをネットワークに送信するシミュレーションにより行ない、提案手法の評価を行なう。

参考文献

[1] Chandy, K.M. and Lamport, L., "Distributed Snapshot: Determining Global States of Distributed Systems," ACM Trans. on Computer Systems, Vol. 3, No. 1, pp. 63–75 (1985).

[2] Osada, S., Hiraga, K. and Higaki, H., "Qos based Checkpointing Protocol in Multimedia Network Systems," The Annual IEEE International Workshop on Fault-Tolerant Parallel and Distributed Systems, CD-ROM (2001).