社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DPS－130 (50)
2007－CSEC－ 36 (50)
2007／3／2

アドホックネットワークのためのチェックポイントプロトコルと評価

東京電機大学 理工学部 情報システム工学科
小野 真和　桧垣 博章　古田 勝久
E-mail: {masa,hig}@higlab.net, furuta@k.dendai.ac.jp

分散コンピューティング環境において耐故障性を実現する手法として、従来の固定ネットワーク環境ではチェックポイントリカバリプロトコルが提案されてきた。チェックポイントリカバリプロトコルでは、状態情報を格納する安定記憶の存在と、メッセージの送信元コンピュータと送信先のコンピュータの同期による一貫性のないメッセージ(紛失メッセージ、孤児メッセージ)の検出、回避が十分に可能な通信帯域の存在が前提となっている。本論文では、これらの前提条件が成立しない、移動コンピュータのみで構成されるようなアドホックネットワークに注目し、アドホックネットワークにおけるチェックポイントリカバリプロトコルを提案する。本提案では移動コンピュータの状態情報を複数の隣接移動コンピュータに保存する。また、転送中に送信先において紛失メッセージとなる可能性のあるメッセージを紛失可能性メッセージとして中継移動コンピュータで保存する。このとき、中継移動コンピュータの状態情報の一部として記憶することにより、状態情報とメッセージログを同一の移動コンピュータに同時に保存することができる。これによって、チェックポイントプロトコルの開始から終了までに要する時間を短縮することが可能である。

Evaluation of Checkpoint Protocol for Mobile Ad Hoc Network

Masakazu Ono,　Hiroaki Higaki and Katsuhisa Furuta
Department of Computers and Systems Engineering
Tokyo Denki University
E-mail: {masa,hig}@higlab.net, furuta@k.dendai.ac.jp

For achieving mission-critical network applications, checkpoint recovery protocols have been researched and developed. In coventional protocols for wired networks, stable storages to store state information are assumed and enough bandwidth is assigned to synchronize a sender and a receiver computers of a message in order to avoid that the message becomes inconsistent, i.e. neither orphan nor lost. In this paper, we propose a novel checkpoint protocol in ad hoc networks without stable storage and enough communication bandwidth. Here, a checkpoint request message is delivered by flooding. State information of a mobile computer is carried by this message and stored into neighbor mobile computers. A candidate of a lost message is detected and stored by intermediate mobile computer on its transmission route. Here, communication overhead for taking global checkpoint is reduced.

1　はじめに

ノート型コンピュータやPDAなどの移動コンピュータをIEEE802.11 [3]やHIPERLAN [1]、Bluetooth [2]などの無線通信プロトコルを用いて相互に接続した無線LAN技術の研究開発が進み、その使用が普及している。また、無線基地局を介して有線ネットワークと接続されたインフラストラクチャネットワークだけでなく、移動コンピュータだけで構成されるアドホックネットワークへの要求が高まっている。アドホックネットワークの適用例として、一時的に構築されるイベント会場や災害現場などでのネットワーク、危険地帯で無線基地局が設置できない場所における自律移動型ロボットの協調動作のためのネットワーク、センサネットワーク [5,13]などがある。このようなアドホックネットワークにおけるミッションクリティカルアプリケーションの実行を考えたとき、耐故障性を実現するためのチェックポイントリカバリ手法を適用することが考えられる。しかし、有線ネットワーク環境を対象とした従来のチェックポイント手法 [9]は安定記憶 [12]がネットワーク上に存在することを前提としている。また、一貫性のないメッセージ(孤児メッセージと紛失メッセージ)を送信元コンピュータと送信先コンピュータの同期によって検出することが可能となる十分な帯域幅がネットワークによって提供されているとしている。そのため、アドホックネットワーク上の移動コンピュータは安定記憶を持つことができないという問題や、アドホックネットワークにおける隣接移動コンピュータ間の通信リンクが狭帯域幅で低信頼であるため、このネットワークを介した同期に要する通信オーバヘッドが大きいという問題を解決する必要がある。そこで、本論文ではアドホックネットワークにおいて安定記憶を実現し、一貫性のないメッセージの発生を回避するための送受信コンピュータ間における同期の実現に起因する通信オーバヘッドを回避する新たなチェックポイントプロトコルを提案する。

2　従来手法

2.1　チェックポイントプロトコル

アドホックネットワーク $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ とは、移動コンピュータの集合 $\mathcal{V}$ と互いに直接メッセージを交換する

ことが可能な移動コンピュータ $M_i, M_j \in \mathcal{V}$ の間の双方向リンク $\langle M_i, M_j \rangle$ の集合 $\mathcal{E}$ で定まるネットワークである。一般に、ネットワーク環境において、チェックポイントプロトコルによって各移動コンピュータ $M_i \in \mathcal{V}$ が設定したローカルチェックポイント c_i の集合であるグローバルチェックポイント $C_{\mathcal{V}}$ が一貫性を持つとは、次の性質を満たすことをいう [6]。

[定義]

1) 送信元移動コンピュータ M_s から送信先移動コンピュータ M_r へ配送されるメッセージ m が紛失メッセージであるとは、グローバルチェックポイント $C_{\mathcal{V}}$ に対して、$Send(m)$ が c_s に先行し、c_r が $Receive(m)$ に先行することである。なお、$Send()$ と $Receive()$ は、アプリケーション層におけるメッセージ送受信イベントである。
2) メッセージ m が孤児メッセージであるとは、$C_{\mathcal{V}}$ に対して、c_s が $Send(m)$ に先行し、$Receive(m)$ が c_r に先行することである。
3) 紛失メッセージ、孤児メッセージを含まないグローバルチェックポイントは、一貫性が保たれているという。□

ただし、紛失メッセージをリカバリ回復時に再送信することができれば、システム状態の一貫性を維持することが可能である。そこで、一貫性のあるグローバルチェックポイントを以下のように再定義する。

[定義]

4) 一貫性のあるグローバルチェックポイントとは、孤児メッセージを含まず、すべての紛失メッセージをリカバリ回復時に再送信可能であるものである。□

この定義にしたがって、M_r で紛失メッセージ m を記憶、保存するプロトコルとして [8] がある。

従来のチェックポイントプロトコルにおいては、m が $C_{\mathcal{V}}$ に対する紛失メッセージや孤児メッセージとなることを M_r でのみ判定することを前提としている。そのため、これらの発生を回避するには、システム全体での同期を必要としていた。例えば、Koo [11] のプロトコルにおいては、チェックポイント要求メッセージ *CReq* を受信してから、チェックポイント終了メッセージ *CFin* を受信するまでの間、アプリケーションメッセージの送信を禁止している。ところが、アドホックネットワークにおいては、無線通信の狭帯域幅、無線信号の減衰と複数無線信号の衝突による低信頼性、無線通信帯域の予約における競合、マルチホップ配送による伝達遅延などのために移動コンピュータ間の同期に要する通信オーバヘッドが大きくなる。この結果、アプリケーションの実行を一時停止する時間が長くなる、すなわち、チェックポイントプロトコルの開始から終了までの時間が長くなるという問題が発生する。本論文で提案するプロトコルでは、アドホックネットワークにおいて、m が紛失メッセージあるいは孤児メッセージとなる可能性を m の配送経路上にある移動コンピュータが判定し、必要に応じて m を記憶したり、m の転送を遅延させたりすることによってこの問題を解決する。ここでは、隣接する移動コンピュータ間における同期のみが必要であることから、アプリケーションの停止時間を短縮することが可能である。

2.2 モバイルチェックポイントプロトコル

モバイルチェックポイントプロトコルの実現にあたり、論文 [14] では、移動コンピュータを含むネットワークを以下の 4 つのモデルに分類している。

1) Centralized Networks
2) Cell-Dependent Infrastructured Networks
3) Cell-Independent Infrastructured Networks
4) Ad-hoc Networks

1)-3) は、ネットワークの構成要素に固定コンピュータを含んでいる。そこで、固定コンピュータに実現した安定記憶に移動コンピュータの状態情報を保存することにより、チェックポイントを設定することができる。論文 [10] では、同期チェックポイント手法と非同期チェックポイント手法を組み合わせた複合チェックポイント手法を提案している。[10], [15] および [14] において、それぞれ 1)、2) に対するプロトコルを設計している。また、[4] と [16] も 1) を対象として固定コンピュータの安定記憶を使用するプロトコルを提案している。ところが 4) においては、ネットワークの構成要素が移動コンピュータのみであることから、安定記憶の実現が困難である。そこで、各移動コンピュータのチェックポイントの設定を、複数の移動コンピュータに状態情報を記憶させることによって実現する。

3 提案プロトコル

3.1 チェックポイントプロトコル

以下の条件のもとでプロトコルを構成する。

[前提条件]

1) アドホックネットワークに含まれるすべての移動コンピュータは、チェックポイントプロトコルの実行中、マルチホップ配送により互いにメッセージ交換が可能である。
2) 隣接する移動コンピュータ間の通信リンクは、動的に切断および確立されることがある。
3) 各移動コンピュータは、隣接する移動コンピュータのリストを保持している。
4) 隣接する移動コンピュータ間の通信リンクは双方向であり、半二重通信が行なわれる。また、ユニキャスト通信は、受信確認と再送機構により、メッセージの紛失なく実現されているものとする。□

チェックポイントプロトコルの基本形を示す。チェックポイントプロトコルの開始は、任意の移動コンピュータが任意のタイミングで行なうことができる。チェックポイント設定要求の伝達と、チェックポイント間の同期は、チェックポイント設定要求メッセージ *CReq* のフラッディング [7] によって実現される (図 1)。

CReq を受信した移動コンピュータ M_i は、現在の状態情報 S_i を獲得することによってローカルチェックポイント c_i を設定するとともに、*CReq* を隣接する移動コンピュータ群、すなわち、M_i の無線信号到達範囲内に存在するすべての移動コンピュータにブロードキャストする。これを繰り返すことによって、前提条件 1) により、すべての移動コンピュータにおいて、ローカルチェックポイントを設定することができる。

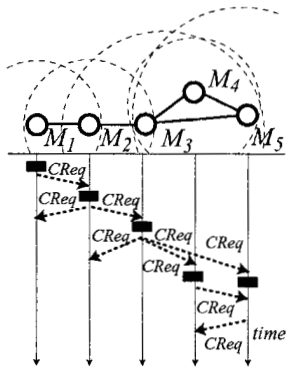

図 1: チェックポイントプロトコル

ここで、移動コンピュータ M_i の状態情報 S_i を保存する安定記憶を実現するために、S_i を複数の隣接移動コンピュータに記憶させる手法を用いる。各移動コンピュータは、ローカルチェックポイント c_i における状態情報 S_i を獲得してから $CReq$ のブロードキャストを行なうことから、S_i を $CReq$ によって配送することにより、追加のメッセージを要することなく S_i の配送が実現される。

[アドホックチェックポイントプロトコル (基本形)]

1) 任意の移動コンピュータ M_0 が、M_0 の状態情報 S_0 を獲得することでローカルチェックポイント c_0 を設定するとともに、S_0 を含み、M_0 が生成した ID が付与されたチェックポイント設定要求メッセージ $CReq$ を M_0 の無線信号到達範囲内にブロードキャストする。このとき、タイマ T_0 をセットする。
2) 移動コンピュータ M_i が送信したチェックポイント設定要求メッセージ $CReq$ を受信した移動コンピュータ M_j は、以下の処理を行なう。
   2-1) M_i から同一の ID を持つ $CReq$ を受信していないならば、受信した $CReq$ に含まれる M_i の状態情報 S_i を保存する。
   2-2) M_j がいずれの隣接移動コンピュータからも同一の ID を持つ $CReq$ を受信していないならば、M_j の状態情報 S_j を獲得するとともに、S_j を含み、受信した $CReq$ と同一の ID を付与した $CReq$ を M_j の無線信号到達範囲内にブロードキャストする。このとき、タイマ T_j をセットする。
3) 移動コンピュータ M_j が隣接移動コンピュータリスト L_j に含まれるすべての移動コンピュータから $CReq$ を受信する以前にタイマ T_j が時間切れとなったならば、M_j は、同じ $CReq$ を再度ブロードキャストする。□

ここで、チェックポイントプロトコルの実行と並行に送受信されたメッセージは、紛失メッセージや孤児メッセージとなる可能性がある。紛失メッセージは、いずれかの移動コンピュータに保存し、リカバリ回復時に、保存されたメッセージを再送信することによって、システム状態の一貫性を維持することができる。一方、孤児メッセージは、リカバリ再実行時に送信元移動コンピュータが同一のメッセージを再度送信する保障がないことから、その発生を回避しなければならない。移動コンピュータの移動による通信路の切断と接続が発生しない場合には、以下の性質が成り立つ。

[性質]

1) 紛失メッセージ m_l は、その配送経路上で以下のいずれかの条件を満足する。
   1-a) m_l の配送経路上にある 1 台以上の移動コンピュータ M_i において、$receive(m_l) \rightarrow c_i \rightarrow send(m_l)$ が成り立つ。
   ただし、$send()$ と $receive()$ は、ネットワーク層における送受信イベントである。また、$\rightarrow$ は、イベント間の happened-before の関係を表すものとする。
   1-b) m_l の配送経路上にある 2 台の移動コンピュータ M_i,M_j において、M_i,M_j は互いに直接通信可能であり、$send(m_l) \rightarrow c_i$ かつ $c_j \rightarrow receive(m_l)$ が成り立つ。
2) 孤児メッセージ m_o は、その配送経路上で以下の条件を満足する。
   2-a) m_o の配送経路上にある 2 台の移動コンピュータ M_i,M_j において、M_i と M_j は互いに直接通信可能であり、$c_i \rightarrow send(m_o)$ かつ $receive(m_o) \rightarrow c_j$ が成り立つ。□

性質 1) により、紛失メッセージとなる可能性があるメッセージ m_l を中継移動コンピュータ、すなわち m_l の送信元でも送信先でもない移動コンピュータが検出することができる。もし、この検出が送信先移動コンピュータ M_r でのみ検出可能であるならば (例えば、論文 [6], [11] では、M_r で m_l が紛失メッセージとなることを検出している。)、図 2 に示すように、M_r が m_l を受信した時点、すなわち $Receive(m_l)$ においては、M_r がすでに $CReq$ を隣接移動コンピュータへブロードキャスト送信済みであることが考えられる。この場合、m_l

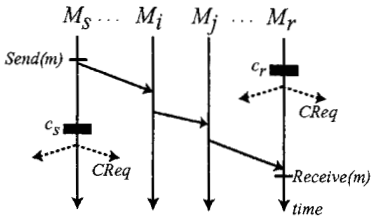

図 2: 紛失メッセージの検出とチェックポイント

を隣接移動コンピュータに記憶させるために、m_l を含む制御メッセージをブロードキャストする必要がある。しかし、これによって、以下の問題が発生する。

(1) M_r がリカバリで必要とする情報を隣接移動コンピュータに記憶させるために M_r が送受信するメッセージが増加する。
(2) M_r の状態情報 S_r を保存した隣接移動コンピュータがこの制御メッセージの送信時点においても M_r の無線信号到達範囲内に存在することは保証できない。すなわち、S_r のみを持つ移動コンピュータ、m_l

のみを持つ移動コンピュータが発生することがある。この結果、リカバリで必要とする情報が複数の移動コンピュータに分散することによって、リカバリ時に送受信されるメッセージ数が増加する。

(3) リカバリ回復時の再送信を必要とするすべての紛失メッセージを隣接移動コンピュータに記憶し、チェックポイントプロトコルが終結したことを各移動コンピュータが検出するためには、新しい同期メッセージの導入が必要となる。

そこで、紛失メッセージとなる可能性のあるメッセージ m_l を $CReq$ を送信する前に検出可能な移動コンピュータの存在が不可欠である。ここで、条件 1-a) を満たすメッセージ m_l については、移動コンピュータ M_i が m_l を $CReq$ の送信前に検出することができる。

[紛失メッセージとなる可能性の検出 (1)]
移動コンピュータ M_i が中継メッセージ (M_i を送信元、送信先としないメッセージ)m_l を $CReq$ の送信前に受信し、$CReq$ 送信時までに送信していないならば m_l は紛失メッセージとなる可能性のあるメッセージである。M_i は状態情報 S_i とともに m_l を $CReq$ に含めてブロードキャストし、これを受信した M_i の隣接移動コンピュータは、S_i と m_l を記憶する。□

一方、条件 1-b) を満たすメッセージ m_l については、m_l が紛失メッセージとなる可能性があるメッセージであることを検出できるのは M_j であり、検出したとき M_j はすでに $CReq$ メッセージを送信済みであることがある。そこで、以下の方法により m_l の検出を M_j が行ない、M_i が $CReq$ を送信する前にその結果を M_i に通知する (図 3)。

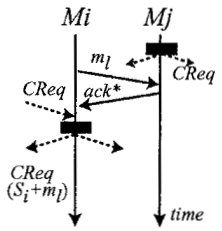

図 3: 中継移動コンピュータでの紛失メッセージの検出とチェックポイント

[紛失メッセージとなる可能性の検出 (2)]
$CReq$ を送信していない移動コンピュータ M_i から送信されたメッセージ m_l を移動コンピュータ M_j が $CReq$ 送信後に受信したならば、m_l の受信確認応答メッセージに m_l が紛失メッセージとなる可能性があることを示す情報を付加する。M_i は、これを受信したならば、状態情報 S_i とともに m_l を $CReq$ に含めてブロードキャストする。この手法を適用するために、各移動コンピュータは、メッセージ送信時から受信確認応答メッセージの受信までの間は $CReq$ を送信することができない。□

条件 1-a)、条件 1-b) はメッセージ m_l が紛失メッセージとなる必要条件であり、十分条件ではない。例えば、図 4 において、M_1 では $send(m) \rightarrow c_1$ であり、M_2 では $c_2 \rightarrow receive(m)$ であることから、M_2 によって紛失メッセージとなる可能性があると判定される。この結果、m は M_1 からブロードキャストされる $CReq$ に含まれる。しかし、M_1 では $Send(m) \rightarrow c_1$ であり、M_3 では $Receive(m) \rightarrow c_3$ であることから、m は紛失メッセージではない。

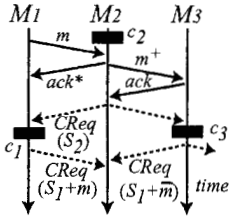

図 4: 誤検出紛失メッセージの多重受理回避 (誤検出の場合)

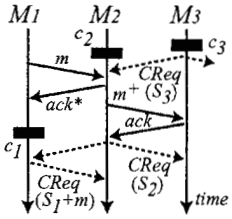

図 5: 誤検出紛失メッセージの多重受理回避 (誤検出でない場合)

もし、グローバルチェックポイント $C_V = \{c_1, c_2, c_3\}$ からリカバリしたならば、m は M_1 によって再送信されるが、M_3 では受信済みである。ここで、m が紛失メッセージとなる可能性のあるメッセージであることを検出した移動コンピュータである M_2(m を含む $CReq$ を送信した移動コンピュータである M_1 ではない) は、m がリカバリ回復時に再送信されるメッセージであることを示す情報を m に付与する。m を受信した送信先移動コンピュータ M_3 では、m の受信が $CReq$ の送信前であるならば、再送信時に m を受信しても破棄することを示す情報を $CReq$ に含めることができる (図 4)。また、m の受信が $CReq$ の送信後であるならば、m は紛失メッセージであるので、リカバリ回復後に再送信されたものを受信する必要がある (図 5)。

また図 6 のように複数の移動コンピュータが、1 つのメッセージを紛失メッセージとなる可能性のあるメッセージであると検出することがある。例えば、m は M_2 および M_4 において紛失メッセージとなる可能性のあるメッセージとして検出される。

これは、M_1 において $send(m) \rightarrow c_1$ であり、M_2 において $c_2 \rightarrow receive(m)$ であることと、M_3 において $send(m) \rightarrow c_3$ であり、M_4 において $c_4 \rightarrow receive(m)$ であることによるものである。もし、これらの移動コンピュータがそれぞれの $CReq$ に m を含めるならば、リカバリ時に送信先移動コンピュータ M_4 は M_1 と M_3 か

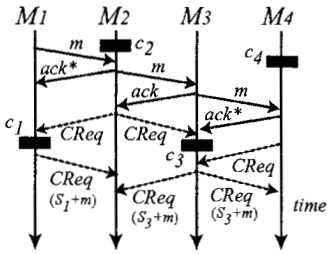

図 6: 紛失メッセージの多重検出

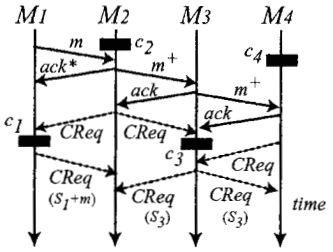

図 7: 紛失メッセージの多重検出の回避

ら再送信された 2 つの m を受信することになる。この問題を解決するために、前段で述べた再送信時に破棄すべきメッセージであることを示す情報を用いる (図 6)。もし、この情報を含むメッセージを紛失メッセージの可能性があるメッセージであると検出しても、これを以降に送信される $CReq$ に含めないことによって、最初に紛失メッセージとなる可能性を検出した移動コンピュータ (条件 1-a) を満たす場合) の送信する $CReq$、または、この移動コンピュータにこのメッセージを送信した移動コンピュータ (条件 1-b) を満たす場合) の送信する $CReq$ にのみ m を含ませることができる。

孤児メッセージとなる可能性のあるメッセージは、中継移動コンピュータでは対処を行わない。送信先移動コンピュータで受信を遅延させることによってのみ対処すれば十分である。

[メッセージ通信プロトコル]
移動コンピュータ M_i は、アプリケーションプログラムにおける送信要求、受信要求が発生したならば、アプリケーション層の送信イベント $Send()$ 、受信イベント $Receive()$ を実行する。また、ネットワークからメッセージ m を受信したならば $receive()$ を実行する。$send()$ は、送信元移動コンピュータにおける $Send()$ 実行時と中継移動コンピュータにおける $receive()$ 実行時に実行される。メッセージ m には、$m.source\_creq\_sent$、$m.creq\_sent$、$m.logged$ という 3 つのフラグが含まれる。

($Send(m)$ イベント)
1 $m.logged := false$ とする。
2 M_i が $CReq$ を送信済みであるならば、$m.source\_creq\_sent := true$, 未送信であるならば $m.source\_creq\_sent := false$ とする。
3 $send(m)$ を実行する。

($Receive(m)$ イベント)
1 m が $buffer_i$ に含まれないならば、$receive(m)$ の実行が終了するまで一時停止する。
2 もし、$m.source\_creq\_sent = true$ かつ M_d が $CReq$ を未送信であるならば、$CReq$ の送信が終了するまで一時停止する。
3 m を $buffer_i$ から取り出す。
4 $m.logged = true$ かつ、M_i が $CReq$ を未送信であるならば、リカバリ回復後に受信する m を破棄する情報を以降に送信される $CReq$ に含める。□

($send(m)$ イベント)
1 M_i が $CReq$ を送信済みであるならば、$m.creq\_sent := true$, 未送信であるならば $m.creq\_sent := false$ とする。
2 $CReq$ の送信を不可とする。
3 m を送信する。
4 $ack(m)$ を受信する。$m.logged = false$ かつ $ack(m).logged = true$ であるならば、以降に送信される $CReq$ に m を含める。
5 $CReq$ の送信を可とする。

($receive(m)$ イベント)
1 $m.logged = false$ かつ $m.creq\_sent = false$ かつ M_i が $CReq$ を送信済みであるならば、$m.logged := true$、$ack(m).logged := true$ として $ack(m)$ を返送する。
2 それ以外の場合は、$ack(m).logged := m.logged$ として $ack(m)$ を返送する。
3 m の送信先が M_i であるならば、m を $buffer_i$ に格納する。
4 それ以外の場合は $send(m)$ を実行する。

各移動コンピュータ M_i は、$CReq$ に状態情報 S_i とともに含ませるメッセージの集合をメッセージログ ML_i として保存することとする。

[アドホックチェックポイントプロトコル]
1) 任意の移動コンピュータ M_0 が、M_0 の状態情報 S_0 を獲得することでローカルチェックポイント c_i を設定するとともに、S_0 とメッセージログ ML_0 を含み、M_0 が生成した ID が付与されたチェックポイント設定要求メッセージ $CReq$ を M_0 の無線信号到達範囲内にブロードキャストする。このとき、タイマ T_0 をセットする。
2) 移動コンピュータ M_i が送信したチェックポイント設定要求メッセージ $CReq$ を受信した移動コンピュータ M_j は、以下の処理を行なう。
   2-1) M_i から同一の ID を持つ $CReq$ を受信していないならば、受信した $CReq$ に含まれる M_i の状態情報 S_i とメッセージログ ML_i を保存する。
   2-2) M_j がいずれの隣接移動コンピュータからも同一の ID を持つ $CReq$ を受信していないならば、M_i の状態情報 S_i とメッセージログ ML_i を獲得するとともに、S_j を含み、受信した $CReq$ と同一の ID を付与した $CReq$ を M_j の無線信号到達範囲内にブロードキャストする。このとき、タイマ T_j をセットする。
3) 移動コンピュータ M_j が近隣する移動コンピュータ

リスト L_j に含まれるすべての移動コンピュータから $CReq$ を受信する以前にタイマ T_j が時間切れとなったならば、M_j は、同じ $CReq$ を再度ブロードキャストする。

4) $ML_i := \phi$ として終了する。□

4 評価

アプリケーションプロセス間(エンド-エンド)の論理チャネルにおいて紛失メッセージと孤児メッセージが存在しない状態を保障するためにすべてのチャネルに同期メッセージを送信する従来手法と、本提案を比較した。図8に結果を示す。x 軸はフィールドサイズ、y 軸は任意の移動コンピュータから開始されたチェックポイントリクエストを受信しチェックポイントプロトコルの実行した移動コンピュータの数、z 軸が移動コンピュータ間で送信された同期メッセージ数を示す。

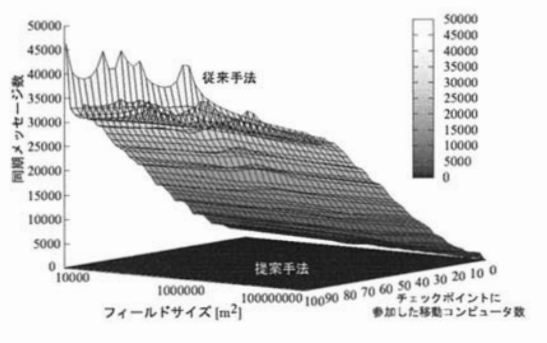

図 8: 同期メッセージ数の比較

従来手法はエンド-エンド間の論理チャネルにのみ注目している。無線アドホックネットワークでは1つの同期メッセージが複数の移動コンピュータを経由して配送される。この同期メッセージはすべてのアプリケーションプロセス間のチャネルで送受信されるため、同期メッセージの配送オーバヘッドが大きくなる。それに対し、アプリケーションをマルチホップで配送する中継移動コンピュータ間の通信路(ホップバイホップ)に注目した提案手法では、1回のフラッディングで紛失メッセージの検出と保存、孤児メッセージの発生回避を保障することが可能である。以上のことから、エンド-エンド間で一貫性のある状態を維持する方法よりも、ホップバイホップで一貫性のある状態を維持する提案手法の方が少ない同期メッセージで実現できる。

5 まとめと今後の課題

本論文では、アドホックネットワークにおけるチェックポイントプロトコルを示した。紛失メッセージとなる可能性のあるメッセージを、エンド-エンドではなく、ホップバイホップで検証し、メッセージログに保存する機構を導入することにより、各移動コンピュータが隣接移動コンピュータに対して状態情報とメッセージログを含むチェックポイント要求メッセージを一度だけ送信することにより、同期オーバヘッドを削減することができる。今後の課題は、実際に提案されている無線ネットワークにおけるチェックポイントプロトコルとの比較検討を行なっていく。

参考文献

[1] "Radio Equipment and Systems (RES); HIPERLAN," ETSI,Functional Specifications (1995).

[2] "The Official Bluetooth Wireless Info Site," http://www.bluetooth.com .

[3] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," Standard IEEE 802.11 (1999).

[4] Acharya, A. and Badrinath, B.R., "Checkpointing Distributed Applications on Mobile Computers," Proc. of 3rd International Conference on Parallel and DistributedInformation Systems, pp. 73–80 (1994).

[5] Callaway, E.M., "Wireless Sensor Networks," Auerbach Publications (2003).

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

[7] Corson, M.S. and Ephremides, A., "A Distributed Routing Algorithm for Mobile Wireless Networks," ACM Journal of Wireless Networks, vol. 1, No. 1, pp. 61–81 (1995).

[8] Elnozahy, E.N. and Zwaenepoel, W., "On the use and Implementation of Message Logging," Proc. of the Fault-Tolerant Computing Symposium, pp. 298–307(1994).

[9] Gendelman, E., Bic, L.F. and Dillencourt, M.B., "An efficient checkpointing algorithm for distributed systems implementing reliable communication channels," Proc. of 18th International Symposium on Reliable Distributed Systems, pp. 290–291 (1999).

[10] Higaki, H. and Takizawa, M., "Checkpoint-Recovery Protocol for Reliable Mobile Systems," Proc. of the 17th International Symposium on Reliable Distributed Systems, pp.93–99 (1998).

[11] Koo, R. and Toueg, S., "Checkpointing and Rollback-Recovery for Distributed Systems," IEEE Trans. on Software Engineering, Vol. SE-13, No. 1, pp. 23–31 (1987).

[12] Lampson, B.W., Paul, M. and Siegert, H.J., "Distributed Systems – Architecture and Implementation," Springer–Verlag, pp. 246–265 (1981).

[13] Lesser, V., Ortiz, C.L. and Tambe, M., "Distributed Sensor Networks," Kluwer Academic Publications (2003).

[14] Miyazaki, M., Morita, Y. and Higaki, H., "Hybrid Checkpoint Protocol for Mobile Networks with Unreliable Wireless Communication Channels," Proc. of the 2nd Asian International Mobile Computing Conference, pp.164–171 (2002).

[15] Morita, Y. and Higaki, H., "Checkpoint-Recovery for Mobile Computing Systems," Proc. of the 21st Internation Conference Distributed Computing Systems Workshops, pp.479–484 (2001).

[16] Neves, N. and Fuchs, W.K., "Adaptive Recovery for Mobile Environments," Communications of the ACM, Vol. 40, No. 1, pp. 69–74 (1997).

[17] Strom, R. and Yemini, S., "Optimistic Recovery in distributed systems," ACM Trans. on Computer Systems, Vol. 3, No. 3, pp. 204–226 (1985).

[18] Yao, B. and Fuchs, W.K., "Message logging optimization for wireless networks," Proc. of the 20th International Symposium on Reliable Distributed Systems, pp. 182–185 (2001).

[19] 小野, 桧垣, "アドホックネットワークのためのチェックポイントプロトコル," 情報処理学会 MBL 研究会, 情報研報, Vol. 2003, No. 93, pp. 91–96 (2003).