Swarm 間の協力を用いたトラッカーレスな BitTorrent システムの性能評価

M2013SC002 青木勇貴
指導教員：河野浩之

1　はじめに

BitTorrent のユーザは月々のアクティブユーザの数に基づいて 1 億 7000 万もいる．BitTorrent には，ファイルの入手困難問題と集中管理問題がある．ファイルの入手困難問題は人気のないファイルが入手困難になる問題である．この入手困難の問題を解決するために，各 swarm で共有されるファイルを一つにまとめる torrent bundle[1] やある swarm で共有されるファイルのピースを他の swarm 内の active peer に割り振り，ピースのアップロードに加担させる torrent inflation[2] の研究がある．また，トラッカーの管理下にある複数の swarm 同士を協力させてファイルの一部をキャッシュする multi-swarm collaboration(MSC)[3] の研究がある．集中管理問題はトラッカーによる集中管理が単一障害点を生む問題である．この集中管理問題を解決する方法として，トラッカーの機能を peer に委譲してトラッカーレスにする方法がある．

本研究では，BitTorrent のファイルの入手困難問題と集中管理問題を解決するために，MSC におけるトラッカーの機能を peer に分散するトラッカーレスな MSC を提案する．そして，それを実装して性能評価する．トラッカーレスな MSC を実現するためにマルチキャストと動的に変化するトラッカーの役割を持つ peer を使ったトラッカーレスのアーキテクチャ[4] を用いる．実験環境はネットワークシミュレータ ns-2 を用いる．実験に関して，我々は ns-2 上で従来の BitTorrent システムと MSC を用いたトラッカーレスな BitTorrent の両システムにおける耐障害性実験を行う．耐障害性実験では，システム動作中に peer をランダムに落とし，ダウンロード完了 peer の数を調べる．そして，その peer 数により，本研究が従来の BitTorrent よりシステムの耐障害性の面で向上していることを示す．

本論文の構成は以下の通りである．2 章ではトラッカーレスな MSC の BitTorrent システムの全体像について述べる．3 章ではトラッカーレスな MSC の設計を説明し，4 章でその実装を説明する．5 章では実験について述べる．最後に 6 章でまとめる．

2　トラッカーレスな MSC の全体像

本節では、本システムであるトラッカーレスな MSC の BitTorrent システムの全体像について述べる．図 1 は本システムの全体像を表す．

本システムでは，peer leader(PL) と呼ばれる peer がトラッカーの代理として動く．後述するが，新しい peer を swarm に参加させたり，swarm 内の peer と情報を授受したり，swarm 同士が協力するために他 swarm の PL との間で request/response を行う．PL は基本それぞれの swarm に 1 台ずつ存在する．MSC のトラッカーに代わって PL が swarm 内，swarm 間の通信をすることにより，トラッカーレスな MSC を実現する．

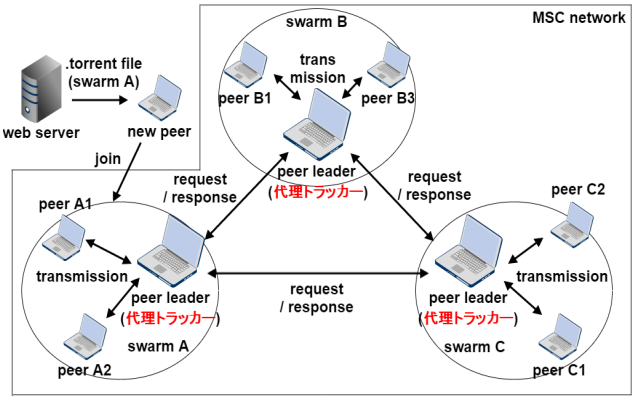

図 1　トラッカーレスな MSC の全体像

3　トラッカーレスな MSC の設計

本章では，トラッカーレスな MSC の設計について述べる．トラッカーレスな MSC を実現し耐障害性を向上させるために Chiara et al. のアーキテクチャ[4] を用いる．3.1 節では swarm 内の通信，3.2 節では swarm 間の通信，3.3 節では PL の移り変わりと復帰について述べる．

3.1　swarm 内の通信

本節では swarm 内の通信について述べる．図 2 は peer の参加から離脱の流れをシーケンス図で表したものである．図 2 の uni1，2，3，4 は peer の情報 (ユニキャストアドレスとポート番号) を表す．A1，X1，Y1 は各 swarm に割り当てられたマルチキャストアドレスを表す．

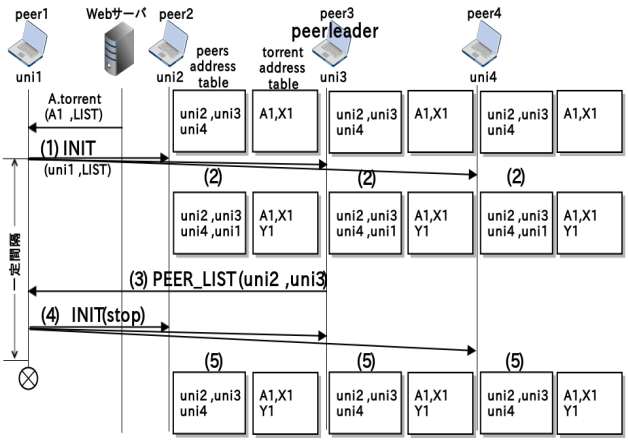

図 2　peer の参加，離脱のシーケンス図

(1)peer は swarm に参加するとき，参加してから一定期間ごとに INIT メッセージをマルチキャストで送信する．INIT メッセージには peer の情報や他の swarm 情報のリストが含まれる．(2) 各 peer はメッセージの情報を使って，torrent address table(TAT) と peers address table(PAT) を更新する．TAT と PAT については 4.1 節で述べる．(3)PL は table の更新に加え，PAT の中から peer をランダムにいくつか選んで，INIT メッセージを送信した peer に peer list として返信する．(4)peer は swarm から退出するときも INIT メッセージをマルチキャストする．この INIT メッセージの中には swarm 退出フラグと peer の情報が含まれる．(5) それを受信した peer は送信した peer の情報を削除する．また，一定期間内に INIT メッセージが届かなくても peer の情報を削除する．

3.2 swarm 間の通信

swarm 間の通信について述べる．swarm 間で通信を行う処理には (1) から (3) のような処理がある．処理は (1) から (3) の順に行われる．(1) から (3) の処理手順を踏むことで，swarm 間の協力が実現される．

(1) collaboration swarm の決定

(2) collaboration swarm へのキャッシュ

(3) collaboration swarm からのキャッシュ復元

collaboration swarm(CS) は swarm と協力関係を持つ swarm である．CS の決定は 4.4 節で詳しく述べる．CS へのキャッシュは 4.5 節で詳しく述べる．CS からのキャッシュ復元は 4.6 節で詳しく述べる．

3.3 動的なトラッカーの移り変わりと復帰

耐障害性を向上させるために，トラッカーの役割を果たす PL が動的に移り変わり，障害時に復帰する．基本的に PL は swarm ごとに 1 台だけ存在し，どの peer も PL になる可能性がある．seed と呼ばれる完全なファイルを持った peer が最初の PL になる．図 3 は PL の移り変わりと復帰の流れを表す．図 3 の (1) から (2) は PL の swarm 離脱のシーケンス，(3) から (6) は PL 復帰のシーケンスを表す．図 3 の seed と peer1 から peer4 は同じ swarm に存在するものとする．

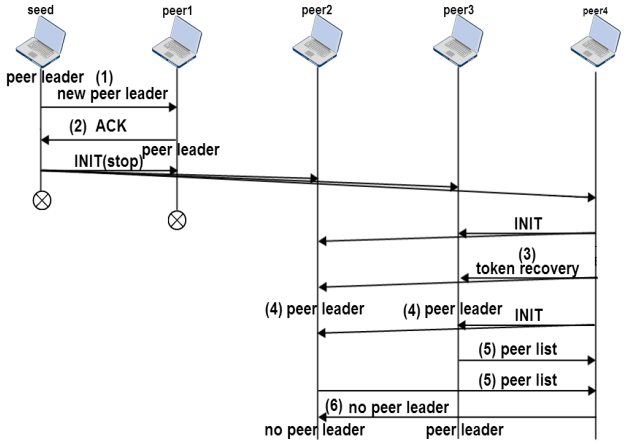

図 3 peer leader の移り変わりと復帰のシーケンス図

PL は他の peer に PL の権利を譲渡してから swarm を去る．(1)PL はランダムに譲渡する peer を選んで，その peer に new peer leader メッセージを送る．(2) その peer から返答が来たら，その peer に PL の役割を譲渡する．返答が来ない場合，再度譲渡する peer を選ぶ．

swarm に PL がいない時，peer の中から PL が自動的に選出される．(3)swarm の peer が INIT メッセージを送っても peer list が返ってこないなら，その peer はトークン復帰メッセージを swarm 内にマルチキャストする．(4) そのメッセージを受信した各 peer は一定確率で PL になる．(5) もし 2 台以上の PL が復帰するなら，それ以降 INIT メッセージの返答として複数の peer list が返信されることになる．(6)PL を 1 台に保つために，複数の peer list を受け取った peer は複数の PL の中から 1 台だけランダムに選出し，それ以外の PL に no peer leader メッセージを送信して PL を退任させる．

4 トラッカーレスな MSC の実装

本章では，3.1 節に出てきた TAT と PAT について，3.2 節で示した CS 決定からキャッシュ復元までの実装について述べる．4.1 節では TAT と PAT について述べ，4.2 節ではビットフィールドを説明する．4.3 節では CS 要否条件を示す．4.4 節では CS の決定，4.5 節では CS へのキャッシュ，4.6 節では CS からのキャッシュ復元について述べる．

4.1 TAT と PAT

TAT は swarm 毎に割り当てられた一意なマルチキャストアドレスのリストである．本研究の実装では，swarm 毎のマルチキャストアドレスを swarm 番号に置き換える．PAT は同じ swarm に属する peer のユニキャストアドレスとポート番号のリストである．本研究の実装では，これらのセットを peer id に置き換える．

4.2 ビットフィールド

ビットフィールドは peer のピース所持状況を表す配列である．各要素には 0 か 1 が格納され，ピースの不所持と所持を表す．各 peer はビットフィールドを所持している．

4.3 multi-swarm collaboration function

MSC で用いられる関数について述べる．関数は HyunYong et al. の MSC の研究で使われたものを用いる．swarm S_i 内のピースの分散状況を表す関数として

$$A(S_i) = \frac{\sum_{n=1}^{C_i} OR(V_{mn})}{C_i}$$

を用いる．C_i はファイルを分割したピースの個数を表す．V_{mn} はピース所持のビットを表し，peer m がピース n を持っていたら $V_{mn} = 1$，持っていないなら $V_{mn} = 0$ となる．$OR(V_{mn})$ は m 台の peer 全員のピース n に関するビットの論理和である．なお，$A(\tilde{S}_i)$ を求める際に seed と全体のピースの β%以上を持つ peer を除く．β は任意の値である．条件 $A(S_i) = 1$ と $\alpha < A(\tilde{S}_i)$ を満たす時，swarm S_i は CS が必要であると判断する．α は N/L で表され，N，L は seed の数，seed 以外の peer 数である．

4.4 collaboration swarm の決定

本節では，swarm が CS を必要と判断してから，CS を決定するまでの処理部分について詳しく述べる．図 4 は CS 決定に至るまでのシーケンス図である．図 4 のように，swarm が 0 から n まで n+1 つ存在するものとする．

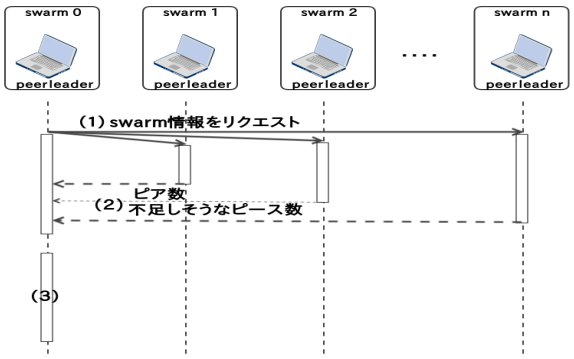

図 4 CS 決定のシーケンス

(1) swarm 0 の PL は swarm 1 から swarm n までの PL に対して swarm 情報 (peer 数と入手困難になりそうなピース数) をリクエストする．

(2) swarm 1 から swarm n の PL はピース保有割合 β%未満である peer のビットフィールドだけで論理和演算したビットフィールドから入手困難になりそうなピース数を求め，peer 数と共に返す．

(3) swarm 0 の PL は各 swarm の $a*|e_0-e|+b*|f_0-f|$ を求め，それが一番小さい swarm を CS とする．

e_0 は swarm 0 の入手困難になりそうなピース数を表し，f_0 は swarm 0 の peer 数を表す．また，e と f は他の swarm におけるそれらの数である．

4.5 collaboration swarm へのキャッシュ

CS が決まったら，swarm は CS にピースをキャッシュする．図 5 はある swarm から CS に対してキャッシュ処理が行われるまでのシーケンス図である．図 5 のように，CS 内には PL と peer 1 から peer n までの n 台の peer が存在するものとする．

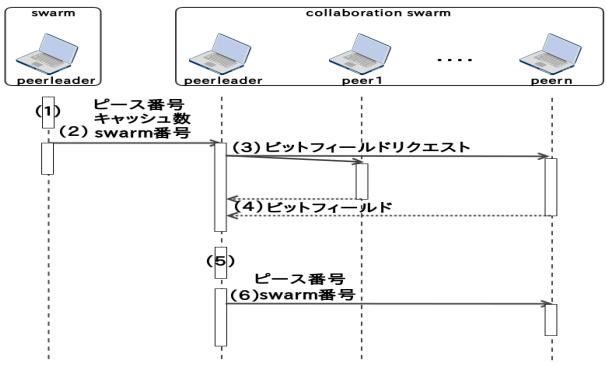

図 5 CS へのキャッシュのシーケンス

(1) swarm の PL は全 peer で論理和演算したビットフィールドの中でビットが 1 であり，β%未満である peer だけで論理和演算したビットフィールドの中でビットが 0 であるピース番号を探し，それを入手困難になりそうなピースの番号とする．

(2) swarm の PL は CS の PL にピースをキャッシュするようにリクエストを送る．そのリクエストには入手困難になりそうなピースの番号，キャッシュ数，swarm の番号が含まれている．

(3) CS の PL は CS 内の n 台の peer にビットフィールドをリクエストする．

(4) 各 peer は PL にビットフィールドを返す．

(5) CS の PL は各ビットフィールドに対してピース保有割合を求め，ピース保有割合 β%以上の peer とピース保有割合 β%未満の peer に分ける．

(6) CS の PL はピース保有割合 β%未満の peer の中からキャッシュ数分の peer を選び，ピースをキャッシュするように指示する．その指示にはピース番号と swarm 番号が含まれている．要求されたキャッシュ数をピース保有割合 β%未満の peer だけで満たすことができない場合，ピース保有割合 β%以上の peer から選ぶ．それでもキャッシュ数を満たすことができない場合は，ピースがそれ以上キャッシュされない．

4.6 collaboration swarm からのキャッシュ復元

CS にピースをキャッシュしたら，swarm はピースが不足した時にピースを復元してもらう．図 6 はある swarm が CS からキャッシュを復元してもらうまでのシーケンス図である．図 6 のように，CS 内には PL と peer 1 から peer n までの n 台の peer が存在するものとする．

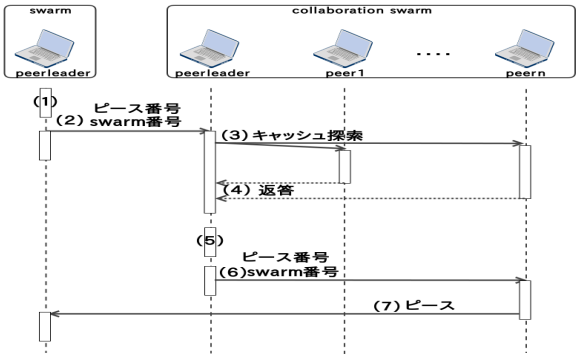

図 6 CS からのキャッシュ復元のシーケンス

(1) swarm の PL は swarm 内の全 peer のビットフィールドを論理和演算したビットフィールドを求め，ビットが 0 であるピース番号を不足ピースとする．

(2) swarm の PL は CS の PL に不足ピースをキャッシュ復元するようにリクエストを送る．そのリクエストには不足ピース番号，swarm の番号が含まれる．

(3) CS の PL は CS 内の n 台の peer にキャッシュ探索メッセージを送る．このメッセージには swarm から送ら

れてきたピースの番号，swarm の番号が含まれる．

(4) PL からキャッシュ探索メッセージを受け取った peer はキャッシュをもっているかどうかを返答する．

(5) CS の PL はキャッシュをもっている peer の中からランダムに peer を選ぶ．

(6) CS の PL は選んだ peer にキャッシュを復元するように指示する．その指示にはピース番号とキャッシュ番号が含まれる．

(7) 指示された peer は swarm 番号の swarm 内の peer に対してキャッシュしているピースを送る．

5 実験

我々は実装したトラッカーレスな MSC を評価するために実験を行った．

5.1 実験環境

本節では，実験環境について述べる．我々はホスト OS 上に仮想化環境を構築し，その仮想化環境上で実験した．下の表 1 は仮想化環境に関係する仕様を表す．

表 1 仮想化環境に関する仕様

ホスト OS の CPU	Intel(R) Core(TM) i7-3770 3.40GHz
ゲスト OS のコア数	2
memory	4.0GB
ゲスト OS	Ubuntu 12.04 32bit

上記の仮想環境上にシミュレーション環境を構築し，そのシミュレーション環境の上で BitTorrent のプログラムとトラッカーレスな MSC のプログラムを実行してシミュレーションを行った．シミュレーション環境として，我々はネットワークシミュレータ ns-2(version 2.29) を用いた．

5.2 耐障害性実験

従来の BitTorrent システムと比べて，MSC を用いたトラッカーレスな BitTorrent システムに耐障害性があることを調べる．そのために，それぞれのシステムで，peer をランダムに落としていき，最終的にダウンロードを完了した peer の数を調べる．これを 1000 回ずつ行い，各システムの平均ダウンロード完了 peer 数を比較する．ランダムに落とされる peer の割合を 25%，50%，75%と増やしていき，それぞれにおいて平均ダウンロード完了 peer 数を比較する．平均ダウンロード完了 peer 数を調べるにあたって，swarm の数は 4 つ，各 swarm に存在する peer の数は 25(全 peer は 100)，各 swarm で交換されるファイルのサイズは 100MB，ピースの大きさは 256KB，peer のアップロード速度は 1Mbps，β は 90 とした．各 swarm には最初に 1 台の seed がいるものとする．それ以外の peer は最初ビットフィールドのビットがすべて 0 とする．

5.3 実験結果

実験結果を示す．図 7 の実験結果は全 peer の 25%，50%，75%を落とす頻度で peer を落とした時のダウンロード完了 peer 数の平均を表す．各項目において，左は BitTorrent の結果を表し，右は MSC を用いたトラッカーレスな BitTorrent の結果を表す．全 peer の 25%を落とす頻度では 18peer，50%を落とす頻度では 18.8peer，75%を落とす頻度では 11.6peer 多くなっている．

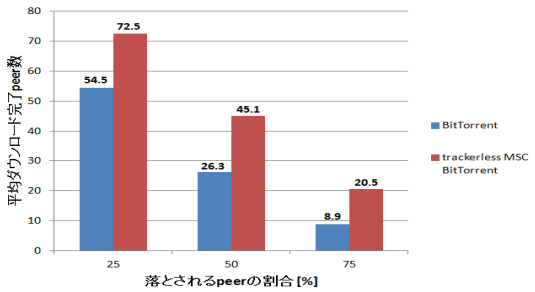

図 7 平均ダウンロード完了 peer 数比較

5.4 考察

実験結果から，BitTorrent と比べて MSC を用いたトラッカーレスな BitTorrent の方がダウンロード完了 peer 数が多くなっていることがわかる．システム動作中に peer が落ちるという障害時にダウンロード完了 peer 数が多くなっていることから，MSC を用いたトラッカーレスな BitTorrent は，BitTorrent と比べてシステムの耐障害性が向上していると言える．

6 おわりに

本研究では，MSC を用いたトラッカーレスな BitTorrent システムを実装し，その性能を評価した．そして，ns-2 上で，BitTorrent と MSC を用いたトラッカーレスな BitTorrent の両システムにおける耐障害性実験を行った．耐障害性実験では，全 peer の 25%，50%，75%を落とす頻度で peer を落とした時，BitTorrent と比べて，ダウンロード完了 peer 数の平均が 18，18.8，11.6 多くなっており，システムの耐障害性が向上していることを示した．

参考文献

[1] D. Menasche, et al., “Content availability and bundling in swarming systems,” Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies, pp. 121-132 (2009).

[2] N. Carlsson, et al., “Using torrent inflation to efficiently serve the long tail in peer-assisted content delivery systems,” Proc. of the 9th IFIP TC 6 International Conference on Networking, pp. 1-14 (2010).

[3] H. Lee, et al., “Multi-swarm collaboration for improved content availability in BitTorrent-like systems,” Proc. of IEEE Consumer Communications and Networking Conference, pp. 565-569 (2011).

[4] C. Taddia, et al., “A multicast-anycast based protocol for trackerless BitTorrent,” Proc. of 16th International Conference on Software, Telecommunications and Computer Networks, pp. 264-268 (2008).