L-012

クライアントからの再送要求に基づく TCP インキャスト回避法
A TCP Incast avoidance method based on retransmission requests from a client

岸本 紫電† 長田 繁幸‡ 樽谷 優弥† 福島 行信† 横平 徳美†
Shiden Kishimoto Shigeyuki Osada Yuya Tarutani Yukinobu Fukushima Tokumi Yokohira

1. はじめに

分散ファイルシステムを採用しているデータセンタのネットワークにおいて，トランスポート層プロトコルとして TCP (Transmission Control Protocol) を用いている場合，分散ファイルシステムを構成するサーバ数が多いときにはスループットがほぼ 0 になってしまうという状況が起こり得ることが知られている．本研究では，この TCP インキャスト (以下，単にインキャストと呼ぶ) を回避するための方法を提案する．

2. TCP インキャストと従来の回避法

2.1 TCP インキャスト

分散ファイルシステムでは，大容量のファイルは複数のブロック (SRU: Server Requested Unit) に分割されて複数のサーバ計算機 (以下，単にサーバと呼ぶ) に分散的に格納される．ある計算機 (以下，クライアントと呼ぶ) がそのファイルを読み出す際には，これらの SRU がサーバ群からほぼ同時に一斉に送信される．これがバースト的なトラフィックを発生させ，ネットワーク内のスイッチではバッファオーバーフローが起きる可能性がある．オーバーフローによりロスしたパケットについては，サーバでは 3 つの重複 ACK の受信 (fast retransmit) か，あるいはタイムアウトの生起を契機として再送する．しかし，多くのサーバがパケットを送出しているため，前者の契機による再送は起こりにくく，後者の契機による再送の方が起こりやすい．通常の TCP の実装では，タイムアウト時間は 200 ミリ秒であるが，データセンタネットワークの RTT (Round Trip Time) は数 100 マイクロ秒であり，また，伝送速度も Gbps 級と大きいことから，SRU を構成する全パケットを送信するために使用する時間は，200 ミリ秒に比較すれば無視していいほど小さい．他のタイムアウトを経験するサーバについても同じようなことになる．従って，ネットワーク全体で見ると，全 SRU を構成する全パケットを送出するためにネットワークを使用する時間は，タイムアウトを待つためにネットワークを使用しない時間 (無通信時間) に比較して，無視していいほど小さくなり，結果的にスループットで見ると 0 に近い値になる．

2.2 従来の TCP インキャスト回避法

対策の 1 つとして，RTOmin のデフォルト値 200 ミリ秒を数 100 マイクロ秒の値に変更する方法 (FGTCP: Fine Grained TCP) が提案されている．しかし，この方法ではサーバ数が多くなった場合にはインキャストが回避できない．これは，タイムアウトが複数回起こった場合，次の再送までの間隔が指数的に増加するという標準の実装に起因しており，結果として無通信時間が延びることが原因である．そのため，FGTCP の再送間隔を線形増加とした方法 (LNRTCP: Linear TCP) や，指数増加と線形増加を組み合わせた方法 (HYBTCP: Hybrid TCP) も提案されている[1]．これらの方法によれば，標準実装の指数増加と比較して，無通信時間を低減することができるためインキャストを回避できることが多い．しかし，これらの方法では積極的に再送を行うことから，同一パケットがネットワーク上に流れる可能性が高くなり，結果として他アプリケーションのトラフィックを過度に妨害してしまう．すなわち，ネットワークに与える負荷が大きくなってしまう．

3. 再送要求に基づく TCP インキャスト回避法

3.1 アイデア

本研究では，クライアント側でスループットを測定し，その値が低くなったときにインキャストが発生したと判断し，サーバに向けて再送要求を送る．これにより，サーバのアイドル状態を解除させることで素早い再送を目指す．以下，このアイデアに基づくインキャスト回避法を AHTCP (ACK Holding TCP) と呼ぶ．

3.2 再送要求の方法

AHTCP を利用するクライアント側では，インキャスト発生の有無を判断するために，直近 5 ミリ秒間において 1 ミリ秒間隔で測定したスループット 5 個分の平均値 (以下，移動平均スループットと呼ぶ) を利用する．本研究では，この移動平均スループットがあらかじめ決められた閾値より低くなったときに再送要求を送る．標準の TCP では，データ受信側は正しくパケットを受信したことを伝えるため，当該パケットのシーケンス番号とデータバイト数の和を確認応答番号として ACK を返す．ここで，MSS (Maximum Segment Size) は 1460 バイトであることを考慮し，その和から 1000 を引き，確認応答を保留した ACK を返す．また，再送要求を送るときには確認応答番号を 1 だけ増やした新たな ACK を送る．これにより，1000 回の再送要求が可能となる． サーバ側では，再送要求を受け取ったときに RTO を強制的にタイムアウトさせ再送する．また，再送の際には，確認応答が保留されていることを見越したシーケンス番号の調整が必要となる．すなわち，SRU が送信されるとき，最終セグメント以外のデータサイズは MSS となるため，再送要求を受け取ったときに再送するセグメントの先頭シーケンス番号は式 (1) のように表すことができる．式中の snd_una は送信済み，かつ確認応答が取れていない最初のデータバイト番号を表す．

$$snd\_nxt = \left\lceil \frac{snd\_una}{MSS} \right\rceil \times MSS + 1 \quad (1)$$

† 岡山大学 Okayama University
‡ 株式会社日本総合研究所 The Japan Research Institute, Limited (岡山大学在籍時，本研究に関する研究に従事)

4. NS2 によるシミュレーション

4.1 ネットワークモデル

NS2 (Network Simulator 2) を使い，図 1，2 のネットワークモデルを用いてシミュレーションを行った．ネットワークモデル A では 1 台のスイッチを用いてネットワークが構築されており，ネットワークモデル B では複数台のスイッチを用いてネットワークが構築されている． ボトルネックリンクはそれぞれクライアントリンク，スイッチ間リンクであり，ポートバッファは 40 パケット，それ以外のリンクのポートバッファは 10000 パケットに設定されている．

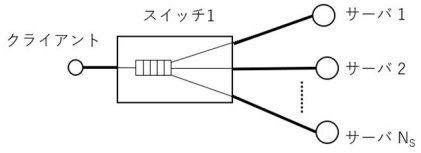

図 1　ネットワークモデル A

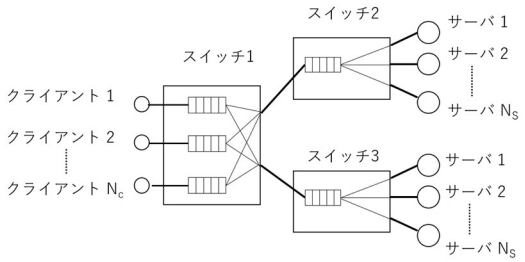

図 2　ネットワークモデル B

4.2 性能評価方法

性能評価の指標として，グッドプットとネガティブ影響率 (NIR: Negative Impact Rate) という値を利用する．前者はスループットからヘッダ分を除いたデータレートを表す．後者はパケット送信がネットワークに与える負荷を表し，これは式 (2) によって定義される．式中の P_{Sent} は全てのサーバが実際に送信したパケット数の合計を意味し，P_{SRU} はパケット消失が起こらない理想的な状態での送信パケット数を意味し，NIR の最小値は 1 となる．

$$NIR = \frac{P_{Sent}}{P_{SRU}} \quad (2)$$

4.3 シミュレーションとその結果

ネットワークモデル A，B 上の全てのリンクを 1Gbps に設定し，10Mbps の UDP バックグラウンドトラフィックを発生させた．グッドプットの最大値は，リンクの帯域からヘッダ分のオーバヘッドとバックグラウンドトラフィックを除いたものとなり，ネットワークモデル A で約 970 Mbps, ネットワークモデル B で約 1,940 Mbps となる． SRU を 64KB，Nc=4，Ns=256 として，SRU を送信するサーバ (アクティブサーバ) 数を 2 から 256 まで増加させた．図 3 から図 6 に，シミュレーション 10 回分の実行結果の平均値を示す．結果から，ネットワークモデル A での AHTCP は，FGTCP と比較するとグッドプットが向上しており，LNRTCP や HYBTCP と比べても，多少低いがほぼ同じような値になっている．一方，NIR は低く抑えられており，ネットワークに与える負荷が小さい．ネットワークモデル B では，LNRTCP や HYBTCP と同等のグッドプットを記録しながらも NIR を抑えることができている．

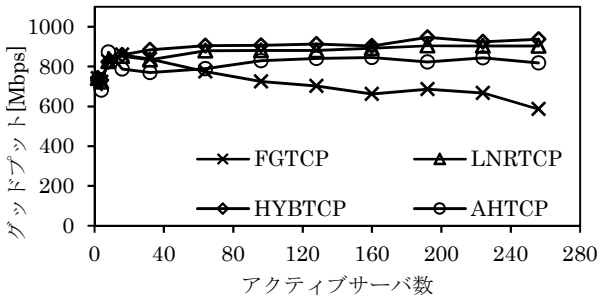

図 3 ネットワークモデル A におけるグッドプット

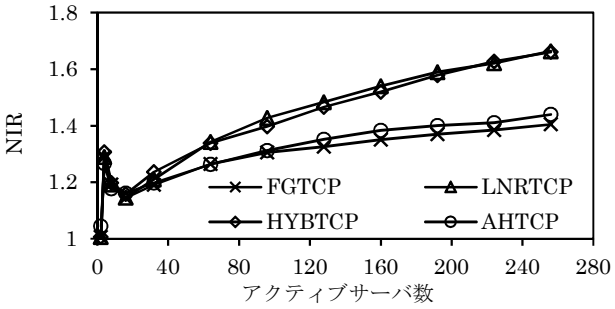

図 4 ネットワークモデル A における NIR

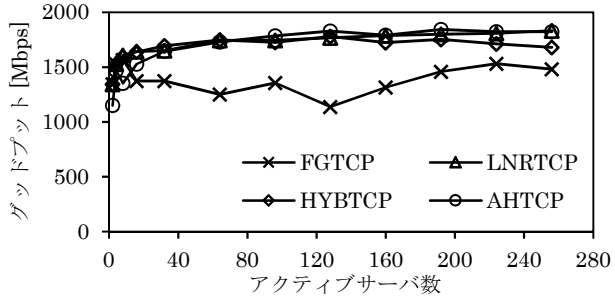

図 5 ネットワークモデル B におけるグッドプット

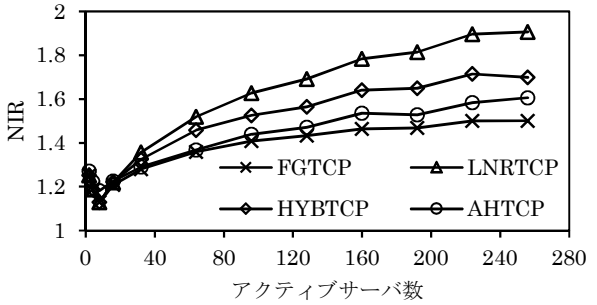

図 6 ネットワークモデル B における NIR

5. おわりに

本論文では，インキャスト回避法として，TCP における確認応答を拡張し，再送要求として利用する AHTCP を提案した．現在はインキャスト検知のための閾値は固定値であるが，様々なネットワーク形態に適用させるためには，閾値を動的に決定する必要があり，今後，その方法を検討する所存である．

参考文献

[1] Shigeyuki Osada, Daichi Izumi, Shiden Kishimoto, Yukinobu Fukushima, Tokumi Yokohira, “Backoff Algorithms to Avoid TCP Incast in Data Center Networks”, International Conference on ICT Convergence 2018, pp. 515–520, (2018).