基盤研究(B) P2P型ビデオストリーミング基盤のための理論的性能保証手法の研究

研究代表者　藤田　聡（広島大学），課題番号：16H02807

1. 背景

ここ数年，ビデオ配信サービスが急速に普及している．2015年秋にはAmazonプライムビデオとNetflixがVoD(Video on Demand)型のビデオ配信サービスを日本国内で開始し，2020年春にはNHKのネット同時配信サービス(NHK+)も始まった．それらのビデオ配信サービスのほとんどは，AkamaiやCloudFrontなどの **コンテンツ配信ネットワーク(CDN)** 経由で視聴者に届けられている．CDNの配信性能は世界中に分散配置されたサーバ群に依存しており，しかもそれらのサーバは，コンテンツ所有者との間のサービスレベル契約(SLA)によって規定されるサービス品質を保証するよう設計されている．そのためCDNの増設には多大な投資が必要となり，スケールアップは容易ではない．その一方で最近の報告によると，インターネット上のオンライン活動の約3分の1がビデオの視聴に費やされており，Netflixだけでも毎月10億時間分以上のビデオストリーミングがなされている(これはデータ量的にはほぼ720万テラバイトに相当する)．また現在の主流はHDクォリティでのビデオ配信であるが，今後数年のうちに4Kあるいは8Kクォリティでの配信へと移行していく可能性もある．

今後予想されるビデオ配信量の急増に対してCDNのみを使って対応するという伝統的な手法には限界があり，その根本的な解決策が強く求められている．その中で特に注目を集めているのが，視聴用に使われている端末の処理能力を補助的に利用して配信を行う **ピア支援型コンテンツ配信ネットワーク(PA-CDN)** である．PA-CDNに関しては，SplitstreamやCLive，ミストCDNなど，これまでにいくつかの先行事例がある．本研究課題ではそのような背景のもとで，PA-CDNにおける配信性能の保証問題に着目し，網羅的な研究を行った．研究成果の詳細については後述するが，本研究で得られた成果は以下のようにまとめることができる：

1. PA-CDNと通常のCDNの違いは中継ピアによるストリーム転送の有無であり，転送のホップ数が増えるほど配信能力(視聴者数)が増加するという特徴がある．本研究課題では，そのようなホップ数と配信能力のトレードオフの理論的な限界を明らかにした．
2. 前項の結果を拡張し，フラッシュクラウドやクラウドサーバによる配信サポートの有無など，様々な現実的な状況における性能限界を明らかにした．
3. ネットワークの混雑や中継ピアの離脱に伴う性能劣化を抑える手法を提案し，性能劣化期間の最小化を行った．
4. モバイルアドホックネットワークなどの特殊なネットワーク環境に対象を拡張し，ビデオストリーミングを安定して行うための手法の提案と実装・評価を行った．

従来のPA-CDNに関する研究の多くは，「この方法を使えばこのくらいの性能が出せる」という形の上界に関するものが多く，本研究課題のように性能の本質的な限界を解明していく下界に関する議論はほとんどなされていなかった．しかし下界に関する議論は，有限の研究資源をどこに投入すべきか(=どこに投入すべきではないか)を明確にし，より高機能なシステムの早期な実現に寄与するものである．本報告書では本研究課題で得られた具体的な結果の概要を説明する．証明や実験データの詳細については発表論文を参照されたい．

2. 研究成果

2.1 基本モデル：マルチツリー型ビデオ配信

ビデオストリームには動画品質に応じたビットレートがあり(例えば4K動画の場合，68Mbps以上が推奨値)，各ノードのアップロード容量も一定値に制限されている(例えばフレッツ光などでは回線ごとに1Gbpsまで)．したがってビデオストリームをP2P技術を使って普通に配信した場合，各ノードが持つことのできる子ノードの数(出次数)が制限されることになり，指定された視聴ノード集合にビデオを配信するためには，次数制約付き全域木(もしくは次数制約付きシュタイナ木)を構成することになる(WebRTCなどのビデオストリーミングのための既存ライブラリには環境に合わせたビットレートの自動調整機能があるため，次数制約は実際にはそれほど強くならない)．その場合の評価のメトリックとしては，視聴ノードまでの転送回数，すなわち配信木の深さを考えるのが適切である．なぜならばPA-CDNに特徴的な性質はノードによる転送であり，転送回数の増加はそれだけでビデオの質の低下につながる可能性を生むからである．本研究課題ではそのようなツリー型配信の中でも特に，Splitstreamというシステムで提案された，与えられたビデオストリームを複数のサブストリームに分割して配信するというモデルを考える(その結果，各ノードの容量制約の範囲内で，サブストリームごと独立に全域木を構成する問題を考えることになる)．サブストリームへの分割については，Multiple Description Codingなどの既存手法を使うことを想定する．

2.2 定常状態におけるPA-CDNによる配信

最初に次のような問題を考える：
問題１ a個のサブストリームに分割されたビデオストリームをn台のノードに配信したいとする．各ノードのアップロード容量が均一であると仮定したとき，各サブストリームを高々1回の転送ですべてのノードに配信するのに必要十分なアップロード容量をaとnの関数で表現せよ．
考察の結果，aとnの組み合わせによっては，a個のサブストリームを同時に送信できるだけの容量で必要十分であるが，特定の組み合わせのもとではaでは不十分で，一定の追加分が必要十分であることが明らかになった(発表論文[1])．転送回数に制限をつけず，チェーン状に転送を行う場合の容量の自明な上界がaであることから，この追加分が，P2P配信のもとでビデオの品質を維持するために必要なコストであると考えられる．

次に，この結果の単純な拡張として，最大転送回数を1から一般のkに拡張した場合の考察を行った(発表論文[2])．その結果，マルチツリー型ビデオ配信における転送ホップ数の自明な下界値が（ほぼ）達成できることが明らかとなった．ここで「ほぼ」と言うのは，得られた上界と自明な下界のギャップが高々１であり，もし改良できるとしても１ホップ減らせるかどうかの水準まで迫って来ていることを意味している（自明な下界がタイトになるケースとそうならないケースについてはある程度解明したが，不明なケースも残されている）．結果の詳細は以下の通りである：

与えられたビデオストリームがa個の均一なサブストリームに分割されており，各ピアのアップロード容量が，a個のストライプを他のピアに転送できるだけの（ギリギリの）大きさであると仮定する．このとき，一つのストライプをサーバからk(>=3)ホップ以内に受け取ることのできる最大のピア数は，等比級数を用いて1+a+a^2+…+a^{k-1}のようにあらわすことができる（完全a分木を考えれば良い）．簡単のため，この値をBkとあらわすことにしよう．視聴者数NがN>Bkを満たすとき，すべての視聴者がストライプをkホップ以内に受け取ることは不可能である．一方，

1. N<=B_kであってもkホップ配信が不可能なケースがある（自明な下界である「kホップ」はタイトではない）が，
2. 任意のN<=B_kに対して，k+1ホップ配信は常に可能である（上界とのギャップは高々１である）

上記の結果のさらなる拡張として，固有のビットレートを持つ **複数のビデオストリーム** を，それぞれの視聴者グループに向けて個別に配信する問題について考察した(発表論文[3])．基本アイデアは，論文[1][2]と同様，与えられたビデオストリームを複数のサブストリームに分けて配信することなのであるが，この論文では特に，動画品質の（質的な）違いをサブストリーム数の（量的な）違いに転嫁することでストリーム間の負荷分散も図っている．各サブストリームは高さの制限された「コアツリー」と呼ばれる均一なオーバーレイを使って配信される．コアツリーのルート(root)にはメディアサーバから直接サブストリームが送られ，ルートに送られたサブストリームは，高々ツリーの高さ分の転送を繰り返して葉ノードに到達する（ホップ数を制限することでレイテンシの増加を抑えている）．ツリーの次数は，各ピアのアップロード容量から決まる値である（論文では説明をわかりやすくするため容量を均一と仮定している）．各視聴者は自分が視聴するビデオストリームから生成されたサブストリームを配信するコアツリーに（視聴者・転送者として）参加すると同時に，もし容量に余力があれば，他のストリームに関するコアツリーにもヘルパーとして参加することができる．

この手法によって，ピアの参加や離脱に伴うオーバーレイの更新処理を見通しよく与えることが可能になった．新規ピアが参加するコアツリー群の選択は，リクエストを最初に受け取るメディアサーバによって集中的にコントロールされる必要があるが，そのあとのオーバーレイの更新手続きは，各コアツリーに最近参加したピアのみによって分散的に実行可能である．ただしこの方法には弱点もあって，ピアの参加・離脱が頻繁に起こる状況では更新のオーバーヘッドのため性能が大きく低下する恐れがある．この点については，次節で述べる結果でカバーされている．

2.3 いくつかの拡張

本研究課題では，上記の他にいくつかの拡張を行った．

最初にPA-CDNにおけるメディアサーバ（CDNのエッジサーバ）の配信負荷が，P2P支援の併用によってどの程度下がるのかを理論的に明らかにした(発表論文[4])．結果の概要は以下の通りである：与えられたビデオストリームがa個の均一なサブストリームに分割されており，各ピアのアップロード容量が，a個のサブストリームを他のピアに転送できるだけの（ギリギリの）大きさであると仮定する．このとき，サーバの容量がN/a以上あれば，N台のピアにa個のサブストリームを２ホップ以内に届けることができる．１ホップ配信に必要なエッジサーバのアップロード負荷は自明にN以上なので，１）サブストリームへの分割と，２）配信ホップ数の（１ホップから２ホップへの）緩和を許すことによって，サーバコストはa分の1へと劇的に下がることになる．数値例として56Mbpsでエンコードされている4K動画の配信を考えてみると，上記の結果が意味するのは，200名の視聴者に１ホップで配信するときのエッジサーバのアップロード負荷が1.12Gbpsとなるのに対し，32個のストライプに分割して２ホップで配信する際の負荷は（オーバーヘッドを無視すれば）350Mbpsにまで減らせると言うことである．なお，上述の各ピアのアップロード容量が「均一にaである」と言う強い仮定のもとで得られた上記の結果は，「平均でaである」と言う弱い仮

定のもとでの結果に拡張できる．具体的には，そのように仮定を弱めた場合にも，サーバコストは高々3N/aで抑えられることが証明されている．

次にCDNの **フラッシュクラウド問題** について考察した(発表論文[5])．フラッシュクラウドとは，CDNのエッジサーバに対して多数の配信要求が集中したときに発生する一時的な性能劣化現象のことであり，論文では，性能劣化期間を最小化するようなP2P支援手法の提案を行なっている．一般にP2Pシステムにおけるアップロード容量とダウンロード要求は，参加ピア数に比例して増加することが知られている．したがって，もし各参加ピアが自分が受けたのと同等以上のサービスを他ピアに与えるようにシステム全体をうまく構成できれば，(BitTorrentのように)外部からわずかなサービスを与えてやるだけで，システム内にサービスが溢れた状況を作り出すことができる．ただしビデオストリーミングの場合，ダウンローダとして参加したピアがアップローダとして機能するようになるまでには一定の時間が必要であり，この部分の遅延が，フラッシュクラウド発生時の性能劣化の原因となる．もちろんアップロード容量は少しずつ追いついてくるため配信性能はしばらく経てば回復するが，ユーザビリティを保つには，性能劣化期間は短い方が望ましい．この論文では，ダウンローダがアップローダになるまでの遅延時間を短縮する手法を提案している．具体的には，到着する配信要求の数に応じて動的に成長するオーバーレイ構造（フラッシュクラウドアブソーバー．以下，FCAと略記）を用意し，このオーバーレイでフラッシュクラウドを適宜吸収すると言うアプローチをとっている．ここでFCAの成長は，メディアサーバ，もしくはその代理サーバによって集中的に管理される．FCAの具体的な形状は，各ピアのアップロード容量とその時点での参加ピア数によって決まるが，任意の時刻において，その時点の参加ピア数への（ほぼ）最小の転送回数でのチャンクストリーム配信がなされるように設計されている．またこの手法で作られるオーバーレイ（FCA）はあくまでも緊急避難的なものであり，実用的には，ダウンロード要求の増加がある程度落ち着いた時点で定常的なオーバーレイに移行することを想定している．また転送回数の最小化に拘らなければ(すなわち理論限界でのトレードオフを考えないのであれば)，配信方法はずっと簡単になる．

本研究課題では上述のマルチツリー型モデルの他に，チャンクのストリームを最小の時間(MBT)でブロードキャストする問題についても考察した(発表論文[6])．この問題は，メッシュ型PA-CDN上でストリームの配信をチャンクレベルで細かく制御する状況に対応している(ツリー型配信ではチャンクの流れはサブストリームごとに固定されているが，メッシュ型配信では，ストリーム中で連続する二つのチャンクが異なる経路で配信されることがありうる)．各ノードが単位時間あたり一つのチャンクを他のノードに向けてアップロードできるような状況下で単一のチャンクを最短時間でブロードキャストする方法は，二項ツリー(binomial tree)と呼ばれるオーバーレイ構造に沿ってチャンクを配信することであり，この方法を使ったときのブロードキャスト時間は，ノード数をNに対してlog Nとなることが知られている．しかし二項ツリーの根に近いノードは，チャンクを受け取ってからそのチャンクのブロードキャストを完了するまでの間，自身のアップロード帯域をそのチャンクのために使い続けなくてはならないため，同一のオーバーレイにチャンクのストリームを流すと，チャンクの衝突のため大きな遅延が発生してしまう．この論文では，ノードのアップロード容量が均一ではない(ヘテロな)ネットワークを想定し，そのようなチャンクストリームのブロードキャスト時間を最小化する手法を提案している．提案方式のアイデアは，大容量ノードをツリーの上流に配置し，そのアップロード容量を有効に利用することである．シミュレーションの結果，この手法はブロードキャスト時間の理論的な下限をほぼ達成することが明らかになった．

2.4 一時的な帯域幅減少に対して耐性のあるツリー状オーバーレイ

本研究課題では，定常状態以外での性能限界についても考察を行った．

PA-CDNの弱点の一つは，アップロードの一部を担う中継ノードが高い処理能力を有するマシンとは限らないため，ノードから出力されるストリームの流れが一時的に停止したり不安定になったりすることである．この問題に対する単純な解決法は，そのノードがシステムから完全に離脱したとみなしてオーバーレイの再構築を行うことであるが，帯域が不安定になる期間が短いときには，この方法は逆に復旧までの時間を長くしてしまう可能性がある．この点を解決するため，PA-CDNのためのツリー構造のP2Pオーバーレイを，ノードのアップロード容量の時間的な減少に対してレジリエントになるように構成する手法を提案した(発表論文[7])．提案手法の基本的な考え方は，

1. ツリー構造のオーバーレイに冗長性を導入し，各ノードのアップロード容量の一部を兄弟ノードへの接続に利用すること（その分，下流への配信に使うことのできる帯域が少なくなり，ツリーの高さが増える可能性があることに注意）
2. 兄弟ノードへの接続リンクを利用して兄弟ノードへのビデオストリームの転送を行うこと（親からのストリームが途絶えたときにも兄弟から受け取れるように前もって準備しておく）

の二点である．具体的には，あるノードの子の最大数が mからm-kへと一時的に減少しても，提案手法を用いることで，$m-k ¥geq m/2^x$ を満たす最小の整数である xで，最大 2^x ホップで m個の子ノードへのビデオストリームの転送を継続することができることが明らかになった．また前述のように，そのような冗長性の導入によってツリーの高さが増える可能性があるが，この論文では，高さの増加が定数で抑えられるための十分条件も併せて導出している．この十分条件により，例えば各ノードがオーバーレイ内に少なくとも6つの子ノードを持つことができるケースでは，オーバーレイ内のノード数が9331以下であれば，提案手法によるレイテンシの増加が高々1ホップで抑えられることが明らかになった．

3. 発表論文のリスト

ジャーナル論文

[1] Hironori Ando, Satoshi Fujita: Tight Bounds on the Upload Capacity to Enable Two-Hop Delivery in Peer-to-Peer Video Streaming Systems. International Journal of Foundations of Computer Science, 31(3): 341-354 (2020).

[2] Satoshi Fujita: Multi-Tree-Based Peer-to-Peer Video Streaming with a Guaranteed Latency. IEICE Transactions on Information and Systems, 102-D(9): 1707-1714 (2019).

[3] Satoshi Fujita:Peer-to-Peer Video Streaming of Non-Uniform Bitrate with Guaranteed Delivery Hops. IEICE Transactions on Information and Systems, 102-D(11): 2176-2183 (2019).

[4] Satoshi Fujita: Cloud-Assisted Peer-to-Peer Video Streaming with Minimum Latency. IEICE Transactions on Information and Systems, 102-D(2): 239-246 (2019).

[5] Satoshi Fujita: Flash Crowd Absorber for P2P Video Streaming. IEICE Transactions on Information and Systems, 102-D(2): 261-268 (2019).

[6] Satoshi Fujita: Broadcasting a Stream of Chunks in Heterogeneous Networks with a Short Maximum Broadcast Time. Journal of Interconnection Networks 18(2-3): 1850007:1-1850007:16 (2018).

[7] Satoshi Fujita: Resilient Tree-Based Video Streaming with a Guaranteed Latency. Journal of Interconnection Networks 19(4): 1950009:1-1950009:20 (2019).

[8] Satoshi Fujita: Three-Tier Delaunay Network as a Topology for Peer-to-Peer Overlay. Journal of Interconnection Networks 19(4): 1950010:1-1950010:16 (2019).

[9] Naoki Takeuchi, Satoshi Fujita: Semi-Deterministic Construction of Scale-Free Networks with Designated Parameters. Journal of Interconnection Networks 18(1): 1850001:1-1850001:18 (2018).

[10] Bahaa Aldeen Alghazawy, Satoshi Fujita: Low Cost Cloud-Assisted Peer to Peer Live Streaming. TIIS 10(4): 1732-1750 (2016).

国際会議（拡張版がジャーナル論文になったものを除く）

[11] Tsuyoshi Yoshihara, Satoshi Fujita: Fog-Assisted Virtual Reality MMOG with Ultra Low Latency. CANDAR 2019: 121-129.

[12] Shogo Inoue, Satoshi Fujita: Collaborative Illustrator with Android Tablets. CANDAR Workshops 2019: 208-214.

[13] Masayuki Kawakami, Satoshi Fujita: Live Streaming over Wi-Fi Direct Multi-Groups. CANDAR Workshops 2019: 221-227.

[14] Yuta Yamada, Satoshi Fujita: Load Balancing in P2P Video Streaming Systems with Service Differentiation. CANDAR Workshops 2018: 539-543.

[15] Takuya Shoji, Satoshi Fujita: Tight Lower Bounds on the Maximum Number of Hops in P2P Video Streaming through Multiple Spanning Trees. CANDAR 2017: 30-37.

[16] Satoshi Fujita: Reliability Calculation of P2P Streaming Systems with Bottleneck Links. IPDPS Workshops 2017: 1238-1244.

[17] Shuji Jinbo, Bahaa Aldeen Alghazawy, Satoshi Fujita: Batch-based flash crowd relaxation in cloud-assisted P2P live streaming. SNPD 2017: 95-100.

[18]]Martin Thodi, Satoshi Fujita: Collaborative Development Environment in Peer-to-Peer Networks. CANDAR 2016: 586-589.