様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　　元 年　　６ 月　　５ 日現在

機関番号：１５４０１
研究種目：挑戦的研究（萌芽）
研究期間：2017～2018
課題番号：１７Ｋ１９９８１

研究課題名（和文）分散ファイル共有システムのための超細粒度アクセス制御方式の研究

研究課題名（英文）Fine grained access control method for distributed shared file systems

研究代表者
藤田　聡（Fujita, Satoshi）

広島大学・工学研究科・教授

研究者番号：４０２２８９９５

交付決定額（研究期間全体）：（直接経費）　　4,900,000円

研究成果の概要（和文）：対象となるドメインを手書き情報のリアルタイム共有に絞って課題について取り組んだ．まず対象をキャンバスへの手書き情報などの二次元情報に限定し，手書き情報のリアルタイム共有という文脈のもとで細粒度アクセス方式の検討を行った．具体的には，ストローク情報の交換方式と整合性の維持方式に関する網羅的な検討を行い，Firebaseによるリアルタイムデータベースによる実装とWebRTCによる任意の端末間での情報転送を通して，十分短いレイテンシで高い分解能での情報共有が行えることを実証した．また実装システムでは，アクセス権限の簡潔な指定と，セキュアな情報交換がともに実現されている．

研究成果の学術的意義や社会的意義
まず本課題の背景は，研究開始後の１年間に大きく変化した．まず汎用のテキストエディタであるAtom エディタを共同編集に用いるためのプラグインが公開され，本課題で目指していた方向のプロダクト がWebRTCを用いて実現された．そして手書き情報の共同編集に関しても，Chrome canvasなどのクラウドベースのソリューションが数多くリリースされた．本研究課題の成果はそれらのアプローチをさらに発展させるものであり，社会的な意義は大きい．また分散システムの分野における最近の潮流の一つであるCRDの新しい適用分野としての意義もあり，学術的な価値も高い．

研究成果の概要（英文）：We focused on real-time sharing of handwritten information as the target domain. First, the object was limited to two-dimensional information such as handwritten information on canvas, and the fine-grained access method was examined in the context of real-time sharing of handwritten information. Specifically, we comprehensively study the exchange method of stroke information and the maintenance method of consistency, and by using real-time database by Firebase and information transfer between arbitrary terminals by WebRTC, with sufficiently short latency and high resolution. In the implementation system, both simple specification of access authority and secure information exchange are realized.

研究分野：情報工学

キーワード：分散ファイルシステム　アクセス制御　WebRTC　P2P

1．研究開始当初の背景

既存の分散ファイル共有システムにおけるアクセス制御は，数種類のアクセス権限を数個のユーザクラスに対して設定するものがほとんどであり，SNS などのアプリケーションで求められるレベルの（細粒度の）アクセス制御は実現されてこなかった．そのためアプリケーション側からの要求は，実質的に各ユーザの運用によってカバーせざるを得なかった．

応募者は 2013 年に，数百人規模のユーザによる非同期かつ頻繁なファイル更新を実時間で行うことのできるピアツーピア(P2P)型の分散ファイル共有方式を開発した．この手法では，各ユーザのもつ端末上に共有ファイルのレプリカを置き，ユーザによってなされたファイルの更新をすべてのレプリカに高速に伝播してレプリカ間の一貫性を保つという方法がとられている．したがって，この手法のもとでは，既存システムのような「アクセス権限の集中管理」は困難であるが，そのいっぽうで，更新情報の伝播経路をうまく制御することで，アクセス制御が分散的に実現されうるという側面もあった．このことから，更新情報の伝播をシステムに参加しているノードの協力により選択的に実現することで，従来システムよりはるかに細かく動的な（超細粒度の）アクセス制御が実現できるのではないかと着想した．

２．研究の目的

本研究の目的は，分散ファイル共有システムのための超細粒度アクセス制御方式を実現し，その性能を実験的に評価することである．既存の分散ファイル共有システムにおける共有ファイルへのアクセス制御は，数種類のアクセス権限を数個のユーザクラスに対して設定するものがほとんどであり，SNS などのアプリケーションで求められるレベルの（細粒度の）アクセス制御は実現されてこなかった．それに対し本研究では，そのファイルにアクセスできるユーザの集合とアクセス権限がファイルの所有者によって任意に設定できるような超細粒度のアクセス制御の実現を目指す．

３．研究の方法

本研究では上記の課題に対し，１）効率的な更新情報配信アルゴリズムの開発，２）細粒度アクセス制御方式の匿名性とセキュリティの向上，３）アクセス権限を簡潔 に指定するためのインターフェース開発，４）複数の既存のファイルシステムとの融合の 4 つの観点からアプローチした．

４．研究成果

現在利用されている代表的な共同編集システムとして Google ドキュメントが挙げられる．Google ドキュメントはウェブブラウザ内で動作するオフィスソフトであり，一つの書類を複数人で同時に編集することができる．書類の変更箇所は他のユーザにリアルタイムに通知され，例えば報告書の執筆作業と校正作業を(メールに添付して互いに送信するなどの余分な手間をかけずに)並行して進めることができる．また文書ファイル(*.doc ファイル)だけでなく，スプレッドシート(*.xls)やプレゼンテーション(*.ppt)の共同編集もサポートされており，通常業

務で使われる書類のほとんどをカバーしているといって良い．類似のサービスとしてマイクロソフトが OneDrive 上で提供している Office Web Apps などがあり，iCloud や Dropbox などのオンラインストレージサービスでも，オフィスソフトを使った共同編集サービスが利用可能である．ただしこれらの共同編集サービスのほとんどはクラウド技術によって実現されている．したがって，たとえそれが小学校の教室内で行われるローカルな共同作業であったとしても，共有ファイルは常にクラウドデータセンター内のストレージに格納され，あるユーザが行なったキー入力は，往復数百キロメートルかけて目の前のユーザに届けられることになる．

GitHub Inc．が 2017 年 11 月に公開した Teletype for Atom は，ローカルな共同編集をローカルに実現する具体的な手法の一つである．このソフトウェアは，オープンソースのテキストエディタである Atom のパッケージの一つであり，トークン交換などの前処理をサーバを介して行なっておくことで，暗号化されたピアツーピア(P2P)通信による文書共有と共同編集が可能となる．P2P 通信は，WebRTC のデータチャネルで実現されており，NAT 超え技術を使うことで，ファイアウォールの内側にいるユーザとの共同作業も可能である．

Teletype for Atom は，編集操作の並行実行に伴う競合や矛盾を回避するため，CRDT (Conflict-free replicated data type)と呼ばれるデータ構造を用いてテキストデータの管理を行っている．テキストに対する編集操作は，文字列の挿入，削除，置換の組み合わせで実現されることが多く，そのサポートは比較的容易である．実際，それらの基本操作を効率よくサポートする具体的な CRDT がこれまでにいくつか提案されている．本研究ではその拡張として，手書き情報やイラストなどの二次元データの共同編集に着目している．テキストなどの一次元データとは異なり，二次元データの場合は，オブジェクトの相対位置に加えて，各オブジェクトに割り当てられる絶対座標に意味が与えられることが多い．また手書き文字などでは，一つのオブジェクトが連続するポイント列として存在しており，それらの二次元空間内での位相的な性質が意味を持つことも多い(例えば英文字の a と d の違いなど)．そのため，単純な挿入，削除以外にも，オブジェクトの持つ位相的な性質を変化させる操作や，オブジェクトの位相的な性質を維持したままその形状を変化させる操作なども含まれることになる．

Android を含む多くのシステムでは，キャンバス上の描画（書き込み）は線分の列として与えられることが多い．提案システムではそのような描画を，画素単位ではなくストローク単位で取り扱うことにした．また各ストロークに対して（X 軸に平行な）外接長方形を考え，二つのストロークの外接長方形どうしが空でない交わりを持つとき，それらのストロークは互いに「干渉する」とみなすことにする．通常の（共同編集型ではない）描画システムでは，各ユーザが行なった書き込み操作の系列は線形リストを使って管理され，このリストとスタックを用いることで，操作の Undo と Redo が実現される．しかし共同編集においては，ストローク間の干渉が発生するため，その取り扱いを慎重に設計する必要がある（例えばユーザ A が顔の輪郭を描画した後でユーザ B が表情を描画した場合，ユーザ A の操作はユーザ B の操作によってブロックされるとみなすのが自然である．テキスト情報の共同編集では，一つの行の次に新しい行を挿入するという操作に関する競合が主として起こっていたが，二次元データの場合，そのような競合のあり方ははるかに複雑になる）．本課題によって得られたシステムでは，共同描画のセマンティクスの検討とストロークを基本単位とした三種類の共同描画モードの提案と実装を行っている．提案手法では，ストロークのキャンバスへの追加によって生成される複数バージ

ョンの取り扱いに関して，以下の三つのケースを明確に区別している（実装の詳細については発表論文で述べている）：

a. ユーザごとに描画領域が独立に与えられており，ストロークの競合がそもそも起きないケース．このケースでは，競合をシステム側でエラーとして検知し，共有キャンバスに一切反映させない仕様になっている（権限の厳重な管理によってエラーの発生は予防可能である）．提案手法では，このケースをカバーするモードとしてレイヤー管理モードを用意している．レイヤー管理モードでは，キャンバスを複数レイヤーの重なりと捉え，レイヤー間に静的な順序関係による優先度付けを与えて描画を制御する．同一レイヤー内の優先度付けは，後述のグラフィティモードと同様，ストローク単位で行われる．具体的なユースケースとして，イラストの共同描画がある．

b. ユーザの描画領域には重なりがあるが，複数のストロークはすべてキャンバスに反映させても構わないケース．「ストロークが重ね書きされた一つの作品」を作ることがゴールとなる．競合するストローク間の順番付けはeventuallyに整合性が取れていればよく，時間の経過によって見え方が若干変化しても構わない．いわゆるグラフィティモードがそれに該当する．グラフィティモードとは，各ユーザがキャンバスに対して行った描画データが他のユーザのキャンバスにリアルタイムに反映するモードである．書き込みに関する制約は一切なく，複数ユーザによる同時書き込みは，ストローク単位で書き込み時刻の昇順に共有キャンバスに反映される(過去になされた描画は後から書き込まれた情報によって上書きされる)．またこのモードでは，オフライン時の書き込みは許容せず，他のユーザが後から与えたストロークと干渉するストロークに関しては，Undo，Redo操作はすべてロックされる（この部分の実装では，CRDTに関する知見が応用されている)．

c. 競合するストロークの追加を敢えて行い，異なるバージョンを「意図して」作るケース．分岐を行うユーザは，分岐直前のバージョンをチェックポイントとして明示的に記録する必要がある（Gitと同様の考え方）．チェックポイントの記録はUndo不可であり，チェックポイント以前のストロークは，チェックポイントを記録した時点ですべて「確定」される．チェックポイント群はすべてのユーザで共有され，指定されたチェックポイントへのロールバックは誰でもいつでも行うことができる(ただしチェックポイントの記録には合意が必要)．チェックポイントから始まるストローク列は，内部的にはすべて「新しい絵」として取り扱われる．そのためチェックポイントは複数バージョンを派生することになるが，チェックポイントを記録したユーザが派生させたバージョンのことを特に，そのチェックポイントの「最新バージョン」と呼ぶ(最新バージョンは唯一に決まり，時間とともに成長する)．各ユーザには，指定したチェックポイントの最新バージョンをアクセスする手段も与える．「他のユーザがキャンバスに追加したストロークと競合するストローク」がユーザAによって追加されたことを検知したシステムは，ユーザAに対してチェックポイントの記録の要否を問い合わせ，記録を選択した場合は追加されたストロークを新たな分岐として受け入れ，しなかった場合はストロークの追加そのものを拒否する．

各モードは，想定するユースケースに必要な追加機能とともに Android アプリとして実装されている．実装システムでは，書き込みによって与えられたストローク情報は JSON 形式にリアル

タイムに変換され，ネットワークを介して他のユーザに届けられる．ストロークのサンプリング周期と転送周期は 30fps に設定されている（それ以上頻度を上げても，受け取り側でバッファ溢れを起こしてしまう）．また各ユーザの操作内容を分散環境で共有するためのモデルとして共有メモリとメッセージ交換に着目し，特に後者に関しては，WebRTC のデータチャネルで相互接続された分散共有メモリとして実装されている．実測されたレイテンシは，WebRTC による実装の同一ネットワーク内での共同編集では 40ms，Firebase による集中型の実装では 200ms ほどであった．

５．主な発表論文等

〔雑誌論文〕（計　4 件）

1. Satoshi Fujita: Cloud-Assisted Peer-to-Peer Video Streaming with Minimum Latency. IEICE Transactions 102-D(2): 239-246 (2019). 査読あり

2. Satoshi Fujita: Flash Crowd Absorber for P2P Video Streaming. IEICE Transactions 102-D(2): 261-268 (2019). 査読あり

3. Naoki Takeuchi, Satoshi Fujita: Semi-Deterministic Construction of Scale-Free Networks with Designated Parameters. Journal of Interconnection Networks 18(1): 1-18 (2018). 査読あり

4. Satoshi Fujita: Broadcasting a Stream of Chunks in Heterogeneous Networks with a Short Maximum Broadcast Time. Journal of Interconnection Networks 18 (2-3): 1850007 (2018). 査読あり

〔学会発表〕（計　4 件）

1. Satoshi Fujita: Resilient Peer-to-Peer Video Streaming with a Guaranteed Latency. CANDAR 2018: 11-18.

2. Tokihiro Naito, Satoshi Fujita: Acquiring Nearly Optimal Peer Selection Strategy through Deep Q-Network. CANDAR 2018: 120-125.

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

4. Satoshi Fujita: On the Cost of Cloud-Assistance in Tree-Structured P2P Live Streaming. IPDPS Workshops2018: 820-828.