DOM 木の同期を支援する分散型 Web ブラウザの提案

河野 匠[†] 新城 靖[†]
佐藤 聡[†] 中井 央[†]

1. はじめに

離れたところに居る人と共同で作業を行うことができるアプリケーションを協調アプリケーションと呼ぶ．協調アプリケーションの多くは中央のサーバに依存しており，サーバの障害により可用性が低下するという問題がある．また，データを中央のサーバに保存するため，プライバシの侵害が発生することがある．

このような問題に対処するために，我々は中央サーバに依存しない協調アプリケーションを実行する基盤として分散型 Web ブラウザを開発している[1)]．これは Web ブラウザ上で動作する協調アプリケーションの実行環境である．従来の分散型 Web ブラウザは Remote Procedure Call (RPC) によるブラウザ間の通信機能とストレージ機能，そしてユーザの認証機能を提供する．協調アプリケーションは，この RPC を用いて他のブラウザで動作するインスタンスと通信を行い，データをやり取りする．

しかし，協調アプリケーションの中には RPC による通信が適していないものがある．その例として複数人で同時に編集できるテキストエディタが挙げられる．この場合はインスタンスの間で編集中のテキストの一貫性を確保する必要がある．しかし，現在の分散型 Web ブラウザが実装している RPC による通信機能では一貫性の確保はサポートされておらず，アプリケーションが自力で一貫性を確保する必要がある．

本研究では，このような問題を解決するため，RPC に代わる，Document Object Model (DOM) 木を用いた新たなデータのやり取りの仕組みを持つ分散型 Web ブラウザを提案する．具体的には，本研究では複数個所での更新を許す DOM 木の同期を可能にする．

2. DOM 木の同期を支援する分散型 Web ブラウザ

提案する分散型 Web ブラウザの全体の構成を図 1 に示す．この分散型 Web ブラウザは，協調アプリケーション，アプリケーション用の表示領域，API の実装，表示領域へ描画を行う描画機能，ストレージ機能，ユーザの認証機能，ブラウザ間の通信機能，および，

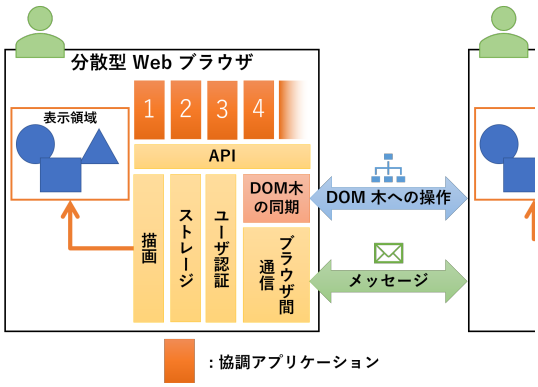

図 1 分散型 Web ブラウザの構成

DOM 木の同期機能から構成されている．DOM 木の同期機能はブラウザ間の通信機能を用いて実装する．

2.1 アプリケーションの構成

各アプリケーションは，ウインドウ・システムのアプリケーションと同様に，Web ブラウザの画面内にウインドウを持つ．その表示内容は，DOM 木で表現されている．

本研究ではアプリケーションのプロセスをブラウザの WebWorker 機能を用いて実現する．WebWorker は，JavaScript で記述された Web アプリケーションの処理をバックグラウンドで実行するための機能である[☆]．これを用いることで個々の協調アプリケーションを別々のスレッドで実行することができる．また，変数も分離されるため，協調アプリケーションの保護と隔離を実現することができる．

Web ブラウザの設計により，WebWorker から直接ウインドウの DOM 木に触れることはできない．また，WebWorker からは，直接通信のための API を呼ぶこともできない．そこで，本研究では，協調アプリケーションのプロセスからの表示領域の操作や外部との通信を可能にする API を提供する．この API を使用することで，協調アプリケーションはブラウザの機能を使用する．例えば，協調アプリケーションのプロセスがウインドウの DOM 木の innerHTML を変更するために，本研究で提供する API である setInner-

† 筑波大学

☆ https://html.spec.whatwg.org/multipage/workers.html

HTML() を呼び出す．この API は，同名のブラウザの DOM API を呼び出す．

2.2 ブラウザ間の通信路

従来の分散型 Web ブラウザは，ブラウザ間通信の実装にインスタント・メッセンジャ，または，ソーシャル VPN を使用していた．これに対してこの論文で提案する分散型 Web ブラウザは，ブラウザ間通信の実装に WebRTC を用い，Web ブラウザ同士を Peer-to-peer (P2P) で接続する．WebRTC では，中央サーバの障害の影響を受けず，サーバ管理者により通信内容が監視されることはない．

WebRTC の通信路を形成するとき，分散型 Web ブラウザは接続相手のユーザを認証する．また，IP アドレスなどの接続情報の交換を行うシグナリングと呼ばれる作業を行う．本研究では，このシグナリングを，共通ランデブ・ポイント・インターフェースを通して行う[2]．

基礎となる P2P の通信路を形成した後，本研究では，その通信路の中に複数の論理通信路を形成できるようにする．これにより，アプリケーションが複数の種類の通信を同時に利用することが容易になり，また，通信路の開設要求ごとに WebRTC における重たい通信路の開設を行う必要がなくなる．

2.3 DOM 木の同期機能を利用した投票アプリケーション

本研究で提供する DOM 木の同期機能を使うと，協調アプリケーションは DOM 木を複数人で簡単に共有することができる．そのようなアプリケーションの例として，図 2 の投票アプリケーションを考える．このアプリケーションでは，各利用者は，ある提案について賛否を投票する．アプリケーションは得票数と投票者を表示する．利用者が賛否のボタンを押した場合，そのインスタンスは自分の持っている DOM 木にその利用者の名前を入れる．すると，DOM 木の同期が発生し，他の利用者においても投票の結果が現れる．

本研究の DOM 木の同期機能は，DOM 木への追加，削除，および変更などの操作 (差分) を，ブラウザ間の通信路を通して送受信することで実現する．DOM 木の一貫性の確保には Operational Transformation (OT)[3] を用いる．これは，データの一貫性を保つためにある操作を別の操作へ変換する手法である．OT の利点は，複数のノードでデータの更新を許すことである．本研究では，OT の実装として，dom-ot☆を用いる．

3. 関 連 研 究

Browsix[4] は Web ブラウザ上で単一 PC 用アプリケーションを動作させる試みである．本研究は複数の PC で動作する協調アプリケーションを動作させる．

☆ https://github.com/marcelklehr/dom-ot

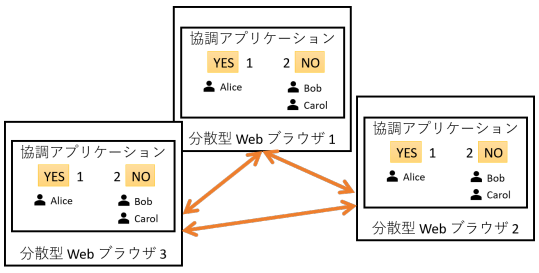

図 2 投票アプリケーション

そのために，複数のブラウザで DOM 木の同期を行えるようにする．

4. ま と め

この論文では，RPC に代わる，DOM 木を用いた新たなデータのやり取りの仕組みを持つ分散型 Web ブラウザを提案した．提案した分散型 Web ブラウザは，OT による DOM 木の同期機能を提供する．

現在までに，JavaScript で記述された簡単なアプリケーションを実装し，ブラウザ間に通信路を開設し，OT を用いて DOM 木が同期できることを確認した．今後の課題としては，各通信機能の実装を行うこと，実用的なアプリケーションの実装を通して提案手法を評価することが挙げられる．また，アプリケーションの記述言語として JavaScript 以外の言語も使用できるようにしたい．

参 考 文 献

1) Y. Shinjo, Fei Guo, N. Kaneko, T. Matsuyama, T. Taniuchi, and A. Sato, “A distributed web browser as a platform for running collaborative applications,” 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011), pp.278–286, 2011.
2) 蛸井博, 新城靖, 佐藤聡, 中井央, “分散型 SNS のための共通ランデブ・ポイント・イン ターフェースの実装,” 情報処理学会 コンピュータシステム・シンポジウム論文集, Vol. 2016, pp.110-120, 2016.
3) C. A. Ellis and S. J. Gibbs, “Concurrency Control in Groupware Systems,” Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data (SIGMOD ’89)，pp.399–407, 1989.
4) B. Powers, J. Vilk, and E. D. Berger,“Browsix: Bridging the gap between unix and the browser,” the 22th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’17), pp.253–266, 2017.