TTC技術レポート
Technical Report

TR-1095

WebRTCに関する技術報告書

Technical Report on WebReal-Time Communication (WebRTC)

第1版
2022年 5月 12日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目次

＜参考＞

１．国際勧告等の関連

本技術レポートは、RFC8825 と RFC8834 を調査したものである。

２．上記国際勧告等に対する追加項目等

なし

３．改版の履歴

版数	制定日	改版内容
第 1 版	2022 年 5 月 12 日	制定

４．参考文献

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/info/rfc8825>.

[RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, January 2021, <https://www.rfc-editor.org/info/rfc8834>.

５．工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、TTC ホームページでご覧になれます。

６．技術レポート作成部門

第 1 版　：　企業ネットワーク専門委員会

Ⅰ　本技術レポートの概要

近年、テレワークの推進により、Web 会議システムが急速に普及してきた。Web 会議システムにおいてはパソコンやスマートフォン、タブレットなどデバイスを選ばず、Web ブラウザからアクセスすることにより、いつでもどこでも会議を行うことができるというメリットがある。Web 会議システムの通信プロトコルはシステムによって WebSocket であったり独自仕様であったりと様々なプロトコルが使用されている。その中でも近年特に注目されているのが WebRTC である。

WebRTC はブラウザ同士の双方向通信のために 2012 年に規格が策定され、様々な Web ブラウザで実装されてきた。その後テレワークの推進により、さらに注目を浴び、2021 年に IETF による標準化が行われた。

本報告書では IETF によって標準化された WebRTC の仕様の概要となる RFC8825 と、WebRTC で使用される RTP について取り決められた RFC8834 について日本語に翻訳する。

II　RFC8825 原文の著作権について

III RFC8825 の和訳

概要

ここの文書は、ブラウザ上で実行するリアルタイムアプリケーションのための、アプリケーションの概要とコンテキストを示している。すなわち「Web 上でのリアルタイム通信」を指す。

これは、(1) この目標を達成するために必要なすべてのパーツが見つけられ、 (2) インターネット・プロトコルに属するパーツが完全に指定されており、すべてのパーツが公開トラックに含まれている、ということを理解する必要がある。

この文書は適用可能性を記述したものであり、これ自体はプロトコルを指定するものではないが、WebReal-Time Communication (WebRTC)に準拠するために実装で従うことが想定される他の仕様を指定している 。

1. はじめに

インターネットは、その歴史のごく初期の頃から、リアルタイムで対話的なアプリケーションを提供するために利用可能な手段と考えられていた。最も容易に想像できるのは、音声通話（別名「インターネット電話」）とビデオ会議である。

このようなアプリケーションを構築するための最初の試みは、特別なネットワーク、特別なハードウェア、そしてカスタムで構築されたソフトウェアに依存していた。

その後、利用可能な帯域幅が増加し、プロセッサーや他のハードウェアがより高速化するにつれて、参加への障壁が減少し、一般的に使用可能なハードウェアで満足のいくエクスペリエンスを提供できるようになった。

それでも、普遍的に通信する能力には多くの障害がある。そのひとつは、今のところ、すべての人が通信に利用することを合意した単一の通信プロトコルのセットが存在しないということである。もうひとつは、普遍的な識別システム（他の通信システムにおいて提供されている、電話番号やメールアドレスなどのようなもの）がまったくないことである。

しかし、「ユニバーサルソリューション」の開発は困難である。

ここ数年、サービスを提供するための新しいプラットフォームとして、ブラウザ組み込み型アプリケーション、つまり「Web アプリケーション」が増加している。ブラウザプラットフォームが必要なインタフェースを持っている限り、ほぼすべての種類のサービスを提供できることがわかっている。

従来、これらのインタフェースはプラグインによって提供されていたが、プラグインはブラウザとは別にダウンロードしてインストールする必要があった。 HTML 5 [HTML5] の開発において、アプリケーション開発者は、これらのインタフェースをブラウザ内で標準化された方法で利用できるようになる可能性に大きな期待を寄せている。

このドキュメントでは、 (1) ブラウザの JavaScript API を介してアクセス可能かつ制御可能であり、 (2) インターネット上のブラウザ間において直接通信するアプリケーションでインタラクティブなオーディオとビデオを使用できるようにするための十分な機能のセットを形成する一連のビルディングブロックについて説明する。結果として得られるプロトコルスイートは、WebRTC の「ユースケース」文書 [RFC7478] で必要なシナリオとして記述されているすべてのアプリケーションを有効にすることを目的としている。

その他の取り組み（たとえば、W3C の WebRTC、Web アプリケーションセキュリティ、デバイスとセンサーのワーキンググループ）は、HTML 5 の取り組みの中で、あるいは HTML5 の取り組みと並行して、これらの機能で標準化された API とインタフェースを利用できるようにすることに重点を置いている。このメモは、ネットワーク上で意思の疎通を指定するために必要なプロトコルとサブプロトコルの指定に焦点を当てている。

運用者は、WebRTC の提供により、ネットワーク上のリアルタイムメディアのシグナリングの性質が変化し、そのようなメディアの作成と消費に使用されるデバイスの種類が変化する可能性があることに注意する必要がある。シグナリングの場合、WebRTC セッションのセットアップは通常、アプリケーション固有のプロトコルを使用して TLS で保護された Web テクノロジーを介して行われる。セッション記述プロトコル（SDP）を解釈するためにネットワーク機器を設置することが要求される運用技術、例えば（1）ネットワークに SIP サーバ[RFC3361]を要求するエンドポイント、または（2）SIP アプリケーション層ゲートウェイ（ALG）の透過的な挿入、のようなシグナリングは機能しない。協調エンドポイントを使用するネットワークの場合、[RFC8155]で定義されているアプローチは、[RFC3361]の適切な代替として役立つ可能性がある。ブラウザベースの通信の増加は、SIP 固定電話機などの専用のリアルタイム通信ハードウェアからのシフトにもつながる可能性がある。これにより、トラフィックフィルタリングや QoS の適用などの目的で、専用のリアルタイムデバイスを独自のネットワークセグメント、アドレス範囲、または VLAN に配置する運用技術の有効性が低下する。

[RFC8837]に記載されているマーキングを適用することは、そのような技術の適切な代替となる可能性がある。

このドキュメントは、正式には[RFC8445]に依存しているが、発行時点では、WebRTC 実装の大部分は、[RFC5245]で説明されているバージョンの Interactive Connectivity Establishment（ICE）をサポートし、Trickle ICE の先行標準バージョンを使用している。[RFC8445] で定義されている「ice2」属性を使用すると、リモートエンドポイントで使用されているバージョンを検出でき、古い仕様から新しい仕様へのスムーズな移行を実現できる。

このドキュメントでは、「WebRTC」という用語（使用されている場合に注意）は、IETF と W3C の両方の取り組みで構成される全体的な取り組みのことを指す。

2 原則と用語

2.1 文書の目標

WebRTC のプロトコル仕様の目標は、記述されているすべての仕様が実装されている場合に、オーディオ、ビデオ、およびデータを使用して、別の実装との通信を実現できるようにする一連のプロトコルを指定することである。

この文書は、WebRTC の仕様へのロードマップとして機能することを目的としている。WebRTC のプロトコル仕様の他の部分で使用される用語を定義し、WebRTC のコンテキストで詳細に説明する必要のない他の仕様への参照をリストアップし、WebRTC スイートの一部である他のドキュメントへのポインタを提供する。

この文書とその参照先の文書を読むことで、WebRTC 互換を実装するために必要なすべての情報を入手できるはずである。

2.2 API とプロトコル間の関係

WebRTC の全体的な取り組みは、2 つの主要な部分で構成され、それぞれが複数のドキュメントで構成されている。

- IETF によって提供されるプロトコル仕様
- 一連の W3C ドキュメントで定義されている JavaScript API 仕様 [W3C.WD-webrtc] [W3C.WD-mediacapture-streams]

これらふたつの仕様がブラウザでサポートされている限り、任意のページに埋め込まれた JavaScript は、ユーザによって適切に許可されている場合に、オーディオ、ビデオ、および補助データを使用して通信をセットアップすることができる。ブラウザ環境は、この機能を使用できるアプリケーションの種類を制限しない。

プロトコル仕様はすべての実装がこの API 仕様を実装することを前提とはしていない。また、通信しているエンティティがブラウザなのか、プロトコル仕様を実装している別のデバイスなのかを相互運用で知る必要はない。

プロトコル仕様と API 仕様の間の連携の目標は、プロトコル仕様のすべてのオプションと機能に関して、そのオプションや機能を実行するためにどの API を呼び出すかを明確にすることである。同様に、API 呼び出しのシーケンスでは、どのプロトコルオプションと機能が呼び出されるかを明確にする必要がある。もちろん、どちらも実装の制約を受ける。

以下の用語は、WebRTC スイートで指定される文書全体で使用される用語であり、その意味がここで示されている。この文書の中でこれらすべての用語が使用されているわけではない。ここに記載されている以外の用語は、一般的に使用される意味に従って使用されている。

エージェント：未定義の用語。下記の「SDP エージェント」および「ICE エージェント」を参照。

Application Programming Interface (API)：一連の呼び出しとイベントの仕様。通常、プログラミング言語または WebIDL などの抽象的な形式仕様に関連付けられており、セマンティクスが定義されている。

ブラウザ : [HTML5]で定義されている「インタラクティブユーザエージェント」と同義で使用される。下記の「WebRTC ブラウザ」（別名「WebRTC ユーザエージェント」）の定義も参照のこと。

データチャネル : データをメッセージの形式で WebRTC エンドポイント間で送信できるようにする抽象概念。 ふたつのエンドポイントは、それらの間に複数のデータチャネルを持つことができる。

ICE エージェント：Interactive Connectivity Establishment（ICE）[RFC8445] プロトコルの実装。 ICE エージェントも SDP エージェントである可能性があるが、SDP を使用しない ICE エージェント（たとえば、Jingle [XEP-0166]を使用する ICE エージェント）が存在する。

インタラクティブ：複数の当事者間のコミュニケーション。ある当事者からのアクションが別の当事者によるリアクションを引き起こす可能性があり、そのリアクションは最初の当事者によって観察される可能性がある。アクション/リアクション/観察に必要な合計時間は数百ミリ秒以下のオーダーである。

メディア：オーディオおよびビデオコンテンツ。ワイヤーなどの「伝送媒体」と混同しないように注意。

メディアパス：メディアデータがひとつの WebRTC エンドポイントから別のエンドポイントまで送られるときの経路。

プロトコル：データユニットのセット、それらの表現方法、およびそれらの送信のルールの仕様と、それらの定義されたセマンティクス。プロトコルは通常、システム間を行き来するものと考えられている。

リアルタイムメディア：コンテンツの生成と表示が時間的に密接に発生することを目的としたメディア（数百ミリ秒以下のオーダー）。リアルタイムメディアは、インタラクティブなコミュニケーションをサポートするために使用できる。

SDP エージェント：[RFC3264]のセクション 3 で定義されているように、Session Description Protocol（SDP）のオファー/アンサーの交換に関連するプロトコルの実装。

シグナリング：メディアパスとデータパスを確立、管理、および制御するために発生する通信。

シグナリングパス：シグナリングに参加しているエンティティ間で使用され、シグナリングを転送する通信チャネル。メディアパスよりもシグナリングパスに多くのエンティティが存在する場合がある。

WebRTC ブラウザ（「WebRTC ユーザエージェント」または「WebRTC UA」とも呼ばれる）：上記のプロトコル仕様と JavaScript API 仕様の両方に準拠するもの。

WebRTC 非ブラウザ：プロトコル仕様に準拠しているが、JavaScript API の実装を要求していないもの。これは、「WebRTC デバイス」または「WebRTC ネイティブアプリケーション」と呼ばれることもある。

WebRTC エンドポイント：WebRTC ブラウザまたは WebRTC 非ブラウザのいずれか。プロトコル仕様に準拠している。

WebRTC 互換エンドポイント：WebRTC エンドポイントと正常に通信できるが、WebRTC エンドポイントの一部の要件を満たしていない可能性があるエンドポイント。これにより、ネットワーク内のどこにそのようなエンドポイントを接続できるかが制限されたり、他のエンドポイントに提供されるセキュリティ保証が制限されたりする場合がある。この仕様による制約はない。言及されている場合は、WebRTC エンドポイントに課せられた要件が WebRTC 互換エンドポイントに与える影響に注意する必要がある。

WebRTC ゲートウェイ：非 WebRTC エンティティへのメディアトラフィックを仲介する WebRTC 互換エンドポイント。

すべての WebRTC ブラウザは WebRTC エンドポイントであるため、WebRTC エンドポイントの要件はすべて WebRTC ブラウザにも適用される。

WebRTC 非ブラウザは、ブラウザが JavaScript アプリケーションをホストする方法と同じように、他の言語で API を提供することにより、アプリケーションをホストすることができる場合がある。例えば、アプリケーションからロードするための C++ API を提供するライブラリとして実装できる。この場合、JavaScript の場合と同様のセキュリティ上の考慮事項が必要になる場合がある。

WebRTC ゲートウェイについては、別のドキュメント [WebRTC-Gateways] で説明されている。

2.3 相互運用性とイノベーション

「IETF のミッションステートメント」 [RFC3935] は、「インターネットに関する標準の利点は相互運用性にあり、標準を実装する複数の製品が連携して、インターネットのユーザに価値のある機能を提供できるようになることである。」と述べている。

インターネット上での通信は、次の 2 つのフェーズで頻繁に発生する。

- ふたつの当事者は、何らかのメカニズムを介して、双方がサポート可能な通信方法を伝達する。
- それらは、共有された通信方法を使用して通信するか、共通点を見つけることができなかった場合は、通信をあきらめる。

多くの場合、通信機能のために多くの選択肢が用意されている。インターネットの歴史は、あらゆる種類のプロトコルにおいて、多くの種類のオプションの提案、標準化、実装、および成功または失敗に満ちている。

実装が必須の機能を設定する目的は、ネゴシエーションの失敗を防ぐことであり、ネゴシエーションを先取りしたり防止したりすることではない。

実装が必須の機能セットが存在することは、 (1) 仕様に準拠していて、 (2) 相手側がその仕様の基本レベルでの通信を受け入れる限り、正常に通信できることが保証されるという点で、市場への展開を大きく変える役割を果たす。

代替手段 (つまり、実装が必須の機能を持たないこと) は、通信できないということを意味するものではない。それは単に、通信のパートナーシップの一部になるために、標準と「いくつかの機能」を実装する必要があることを意味している。「いくつかの機能」は通常、ある種のプロファイルと呼ばれる。インターネットの精神に最も反するバージョンでは、「いくつかの機能」は、特定のベンダの製品のみで使用される。

2.4 用語

この文書のキーワードである「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「NOT RECOMMENDED」、「MAY」、「MUST」は、すべて大文字で表記されている場合にのみ、BCP 14 [RFC2119 RFC8174] で説明されている通りに解釈される。

3. アーキテクチャと機能グループ

ブラウザベースのアプリケーションの場合、リアルタイムサポートのモデルでは、ブラウザにアプリケーションのために必要なすべての機能 (電話やビデオ会議など) が含まれていることを前提とはしていない。ブラウザは、バックエンドのサーバと連動して動作する Web アプリケーション用の関数を実装することになる。

これは、ふたつの重要なインタフェースを指定する必要があることを意味する。つまり、ブラウザがサーバを介さずに相互に対話するプロトコルと JavaScript アプリケーションに提供される API である。

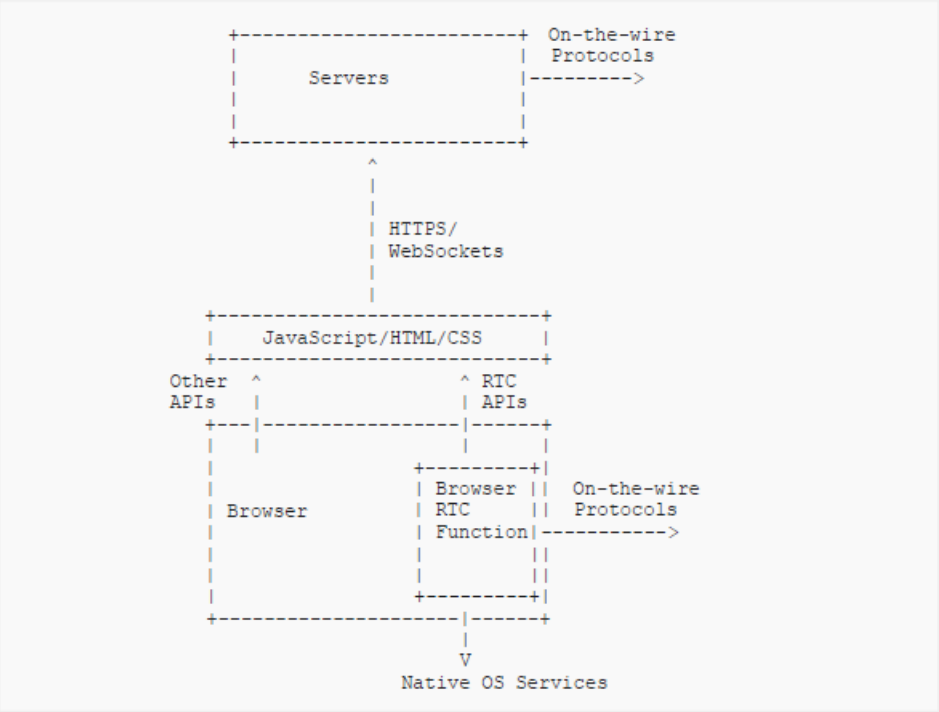

図 1：BrowserModel

注：HTTPS および WebSocket は、ブラウザ API を介して JavaScript アプリケーションに提供される。

すべてのプロトコルと API の仕様に関して、これらが他のブラウザとの通信にしか使えないという制限はない。これらは完全に指定されているため、任意のエンドポイントがプロトコルを正しく実装すれば、ブラウザ内で実行されているアプリケーションとの間で相互運用が可能になる。

一般的に想像される提供モデルを図 2 に示す。(「JS」は JavaScript を表す。)

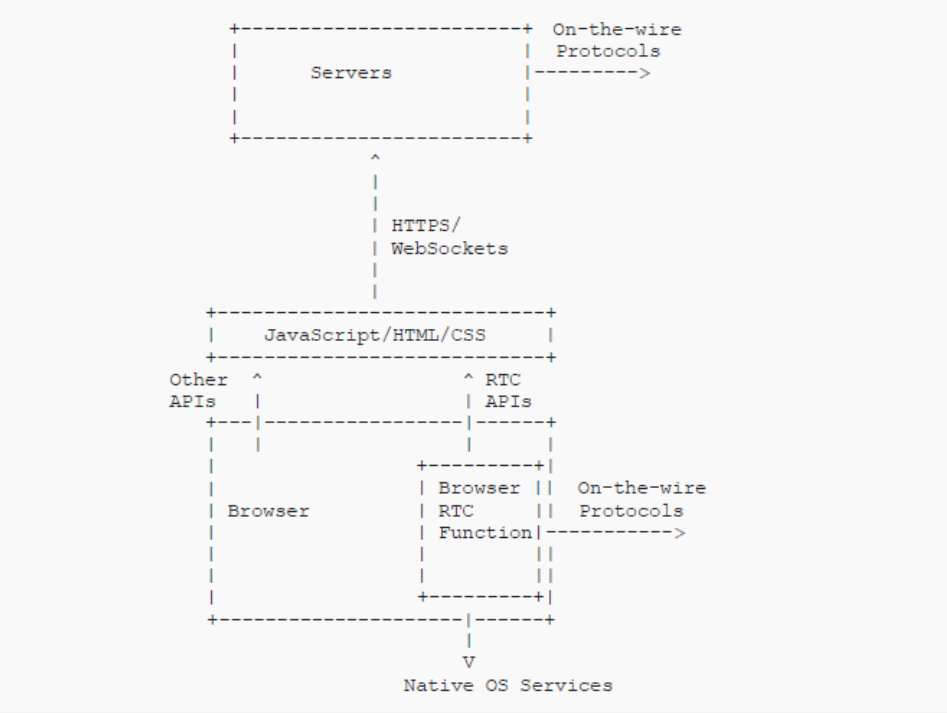

図 2：BrowserRTCT 台形

この図で注意すべき重要な部分は、メディアパス（「ローパス」）がブラウザ間で直接通過するため、WebRTC プロトコルスイートの仕様に準拠している必要があることである。シグナリングパス（「ハイパス」）は、必要に応じて信号を変更、変換、または操作できるサーバを経由する。

ふたつのサーバが異なるエンティティによって運用される場合には、標準化またはその他の合意手段によって、サーバ間のシグナリングメカニズムについて合意する必要がある。サーバ間では既存のプロトコル(例：SIP [RFC3261] や Extensible Messaging and Presence Protocol (XMPP) [RFC6120])を使用でき、ブラウザと Web サーバの間では標準ベースまたは独自のプロトコルを使用できる。

例えば、両方の運用者のサーバが SIP を実装している場合、サーバ間の通信では SIP を使用することができ、ブラウザ上で実行されるアプリケーションと Web サーバ間の通信では標準化されたシグナリングメカニズム(例：SIP over WebSockets)や、独自のシグナリングメカニズムのいずれかを使用できる。同様に、両方の運用者のサーバが XMPP を実装している場合、サーバ間の通信では XMPP を使用することができ、ブラウザ上で実行されるアプリケーションと Web サーバ間の通信では標準化されたシグナリングメカニズム(例：XMPP over WebSockets や Bidirectional-streams Over Synchronous HTTP (BOSH) [XEP-0124])や、独自のシグナリングメカニズムのいずれかを使用できる。

クライアント/サーバ間およびサーバ/サーバ間のシグナリングプロトコルの選択と、それらの間の変換の定義は、この文書の中で説明されている WebRTC プロトコルスイートの範囲外である。

ブラウザで必要な機能グループは、次のように多かれ少なかれボトムアップで指定できる。

データトランスポート：たとえば、TCP と UDP、エンティティ間の接続を安全にセットアップする手段、およびデータを送信するタイミングを決定する機能（輻輳管理、帯域幅推定など）。

データフレーミング：RTP、Stream Control Transmission Protocol（SCTP）、DTLS、およびコンテナとして機能するその他のデータ

フォーマット、およびデータの機密性と整合性のための機能。

データフォーマット：システム間で受け渡されるデータのコーデック仕様、形式仕様、および機能仕様。オーディオおよびビデオコーデック、およびデータとドキュメントの共有の形式は、このカテゴリに所属する。データフォーマットを利用するには、それらを説明する方法（セッションの説明など）が必要である。

接続管理：たとえば、接続の設定、データフォーマットの合意、通話中のデータフォーマットの変更などの管理。 SDP、SIP、および Jingle/XMPP はこのカテゴリに所属する。

プレゼンテーションとコントロール：インタラクションが問題なく動作することを保証するために必要なもの。これには、システムの一部が当事者間の協力を必要とする、フロアコントロール、画面レイアウト、音声起動イメージ切り替え、およびその他のそのような機能を含む。 Centralized Conferencing（XCON）[RFC6501] および Cisco/Tandberg の Telepresence Interoperability Protocol（TIP）は、この種の機能を指定するためのいくつかの試みだった。多くのアプリケーションは、これらの機能への標準化されたインタフェースなしで構築されている。

ローカルシステムサポート機能：一律に指定する必要のない機能。各参加者は他の参加者が認識しない方法で、通信に影響を与えることなく、彼らが選択したこれらの機能を実現する。このカテゴリの例には、エコーキャンセレーション（その一部の形式）、ローカル認証および承認メカニズム、OS アクセスコントロール、通話のローカル録音を行う機能が含まれる。

各機能グループ内では、イノベーションの自由とグローバルに通信する機能の両方を維持することが重要である。イノベーションの自由は、実装ではなくインタフェースの観点から仕様を作成することによって支援される。インタフェースに従って通信できる実装はすべて有効な実装である。グローバルに通信する機能は、(1)コア仕様が IPR の問題に邪魔されないこと、および(2)独立した実装を可能にするためにフォーマットとプロトコルが十分に指定されていることの両方によって支援される。

最初の 3 つのグループが 「メディア転送インフラストラクチャ」を形成し、最後の 3 つのグループが「メディアサービス」を形成すると考えることができる。多くの場合、ブラウザのメディア転送インフラストラクチャには共通の仕様を使用することが理にかなっている。この仕様は、ブラウザに組み込まれ、標準インタフェースを使用してアクセスできる。また、「メディアサービス」レイヤでは「Let a thousand flowers bloom」の考え方が主流である。ただし、相互運用可能なサービスを実現するには、少なくとも 6 つのグループのうち最初の 5 つを指定する必要がある。

4. データトランスポート

データトランスポートとは、ネットワークインタフェースを介したデータの送受信、通信の両端でのネットワーク層アドレスの選択、およびデータを処理するがデータを変更しない中間エンティティとの相互作用（Traversal Using Relays around NAT (TURN)など）を指す。

これには、輻輳制御、再送信、順序通りの配信に必要な機能が含まれる。

WebRTC エンドポイントは、 [RFC8835] で説明されているトランスポートプロトコルを実装しなければならない[MUST]。

5. データフレーミングと保護

メディアトランスポートの形式は RTP [RFC3550]である。 Secure Real-time Transport Protocol（SRTP）[RFC3711] の実装は、すべての実装に必要である(REQUIRED)。

RTP および SRTP の機能の使用方法に関する詳細な考慮事項は、[RFC8834] に記載されている。 WebRTC ユースケースのセキュリティに関する考慮事項は [RFC8826] に記載されており、結果として得られるセキュリティ機能は [RFC8827] に記載されている。

RTP 形式ではないデータの転送に関する考慮事項は [RFC8831] で説明されており、個々のデータチャネルを確立するためのサポートプロトコルは [RFC8832] で説明されている。 WebRTC エンドポイントは、これら 2 つの仕様を実装しなければならない[MUST]。

WebRTC エンドポイントは [RFC8834] 、[RFC8826] 、[RFC8827]、およびそれらに含まれる要件を実装しなければならない[MUST]。

6. データフォーマット

この仕様の目標は、各通信イベントが、接続の両側でサポートされる特定のインスタンスに最適なデータフォーマットを使用できるようにすることである。しかし、通信を確実に実現するには最低限の基準が大いに役立つ。この文書では、すべての実装でサポートされるべき最低限のベースラインを指定する。このベースラインは、この仕様に準拠しており、実装担当者がコードを記述することもできる。

オーディオやビデオをサポートするWebRTCエンドポイントは、[RFC7874] および [RFC7742] で必要なコーデックとプロファイルを実装する必要がある。

7. 接続管理

接続の設定、ネゴシエート、および破棄するための方法、メカニズム、および要件は、大きな項目であり、相互運用性とイノベーションの自由の両方を持つことが望ましい。

これには次の原則が適用される。

1. WebRTCメディアネゴシエーションは、SIPとWebRTCメディアネゴシエーションの間にシグナリングゲートウェイを構築できるようにするため、SIPで使用されるのと同じオファー/アンサーセマンティクス [RFC3264] で表すことが可能である。
2. メディアゲートウェイを使用せずに、ICEをサポートするレガシーSIPデバイスと、適切なRTP/SDPメカニズム、コーデック、およびセキュリティメカニズムとの間でゲートウェイを設定することが可能となる。 Web側のシグナリングとSIPシグナリングの間で変換するシグナリングゲートウェイが必要になる場合がある。
3. 新しいコーデックのSDPが指定される場合、そのコーデックをWebブラウザで使用できるようにするために、他の標準化は必要にはならない。新しいSDPパラメータを持つ可能性のある新しいコーデックを追加しても、ブラウザとJavaScriptアプリケーション間のAPIは変更されない。ブラウザが新しいコーデックをサポートするとすぐに、コーデックが指定される前に作成された古いアプリケーションでも、JavaScriptアプリケーションに変更を加えることなく、必要に応じて自動的に新しいコーデックを使用できるようになる。

WebRTCに対して行われた特定の選択、およびWebRTCを実装するブラウザによって提供されるAPIに対するそれらの影響については、[RFC8829] で説明されている。

WebRTCブラウザは [RFC8829] を実装しなければならない[MUST]。

WebRTCエンドポイントは、ネットワーク層に関連する [RFC8829] で説明されている機能（BUNDLE [RFC8843]、rtcp-mux [RFC5761]、Trickle ICE [RFC8838]など）を実装する必要があるが[MUST]、これらのエンドポイントは[RFC8829]で説明されているAPI機能をサポートする必要はない。

8. プレゼンテーションとコントロール

コントロールの最も重要な部分は、ブラウザの入出力デバイスや通信チャネルとの相互作用に対するユーザのコントロールである。重要なのは、ユーザが自分のオーディオやビデオ、テキストメッセージがどこに送信されているかを把握できることであり、つまりどのような名目で、制御チャネルの一部を構成する当事者によってどのような保証がなされるかを把握する方法を持っていることである。これは主に、ブラウザ、基盤となるオペレーティングシステム、およびユーザインタフェース間のローカル機能である。これは、ピア接続API [W3C.WD-webrtc] およびメディアキャプチャAPI [W3C.WD-mediacapture-streams] で指定されている。

WebRTCブラウザは、これらの仕様を実装しなければならない[MUST]。

9. ローカルシステムサポート機能

これらの機能は、実装の品質が主にユーザエクスペリエンスに大きく影響を与えるという事実によって特徴付けられるが、厳密なアルゴリズムは調整を必要としない。場合によっては、(例えば、以下で説明するエコーキャンセレーションのように、) システム全体の定義では、特定の方法で実装する必要なしに、これらの機能が役立ついくつかの特性をシステム全体に持たせる必要があることを指定する必要がある。

ローカル機能にはエコーキャンセレーションや音量調節、フォーカス・ズーム・パン/チルトコントロール（利用可能な場合）を含むカメラ管理などが含まれる。

システムの特定の当事者が特定のプロパティに準拠していることを確認したい場合がある。例えば以下のような場合がある。

- エコーキャンセレーションは、知覚的に目立つレベルより下の音響フィードバックループの抑制を達成するのに十分なものでなければならない。
- プライバシーの懸念は満たされなければならない[MUST]。たとえば、カメラのリモートコントロールが提供されている場合、API を使用して、ローカル参加者がカメラを制御しているユーザを特定し、カメラの使用許可を取り消すことができるようにする必要がある。
- Automatic Gain Control (AGC) が存在する場合は、話している音声を適切な dB 範囲内で正規化する必要がある。

オーディオ処理に関連する WebRTC システムの要件は、 [RFC7874] に記載されており、その文書には、エコーキャンセレーションと AGC に関する詳細なガイダンスが含まれている。ローカルデバイスを制御するための API は、[W3C.WD-mediacapture-streams] に記載されている。

WebRTC エンドポイントは、[RFC7874] の処理機能を実装しなければならない[MUST]。（セクション 6 の要件とともに、これは WebRTC エンドポイントがドキュメント全体を実装する必要があることを意味する。）

10. IANA の考慮事項

このドキュメントには IANA のアクションはない。

11. セキュリティに関する考慮事項

Web 対応のリアルタイム通信のセキュリティは、次のいくつかの要素で構成される。

コンポーネントのセキュリティ：ブラウザ、および関連するその他のサーバ。ここで最もターゲットが豊富な環境は、おそらくブラウザである。ここでの目的は、これらのコンポーネントの導入が追加の脆弱性を導入しないことである。

通信チャネルのセキュリティ：参加者は、参加しているリンクの暗号化パラメータを確認することで、通信のセキュリティに対して容易に安心できる必要がある。そして、他の当事者から、それらの当事者が適切な措置が取られることを約束された通信が行われていることで安心を得る必要がある。

パートナーの身元のセキュリティ：参加者が本人であることを確認する（肯定的な識別が適切な場合）、または参加者の身元を明らかにできない（匿名性がアプリケーションの目標である場合）。

セキュリティ分析と、その分析から導き出された要件は、 [RFC8826] に含まれている。

[W3C.WD-mediacapture-streams] および [W3C.WD-webrtc] のセキュリティセクションを読み込むことも重要である。

12. 参考文献

12.1 参考文献 (Normative)

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/ rfc2119>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC7742] Roach, A.B., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <https://www.rfc-editor.org/info/rfc7742>.

[RFC7874] Valin, JM. and C. Bran, "WebRTC Audio Codec and Processing Requirements", RFC 7874, DOI 10.17487/RFC7874, May 2016, <https://www.rfc-editor.org/info/ rfc7874>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/ rfc8174>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc- editor.org/info/rfc8445>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/ RFC8826, January 2021, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, January 2021, <https://www.rfc-editor.org/info/rfc8827>.

[RFC8829] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol (JSEP)", RFC 8829, DOI 10.17487/RFC8829, January 2021, <https:// www.rfc-editor.org/info/rfc8829>.

[RFC8832] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channel Establishment Protocol", RFC 8832, DOI 10.17487/RFC8832, January 2021, <https://www.rfc- editor.org/info/rfc8832>.

[RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, January 2021, <https://www.rfc- editor.org/info/rfc8834>.

[RFC8835] Alvestrand, H., "Transports for WebRTC", RFC 8835, DOI 10.17487/RFC8835, January 2021, <https://www.rfc-editor.org/info/rfc8835>.

[W3C.WD-mediacapture-streams] Jennings, C., Aboba, B., Bruaroey, J-I., and H. Boström, "Media Capture and Streams", W3C Candidate Recommendation, <https:// www.w3.org/TR/mediacapture-streams/>.

[W3C.WD-webrtc] Jennings, C., Boström, H., and J-I. Bruaroey, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Proposed Recommendation, <https:// www.w3.org/TR/webrtc/>.

12.2 参考文献 (Informative)

[HTML5] WHATWG, "HTML - Living Standard", January 2021, <https:// html.spec.whatwg.org/>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers", RFC 3361, DOI 10.17487/RFC3361, August 2002, <https://www.rfc-editor.org/info/rfc3361>.

[RFC3935] Alvestrand, H., "A Mission Statement for the IETF", BCP 95, RFC 3935, DOI 10.17487/RFC3935, October 2004, <https://www.rfc-editor.org/info/rfc3935>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <https://www.rfc-editor.org/info/ rfc5245>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <https://www.rfc- editor.org/info/rfc5761>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/ rfc6120>.

[RFC6501] Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", RFC 6501, DOI 10.17487/ RFC6501, March 2012, <https://www.rfc-editor.org/info/rfc6501>.

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <https://www.rfc-editor.org/info/rfc7478>.

[RFC8155] Patil, P., Reddy, T., and D. Wing, "Traversal Using Relays around NAT (TURN) Server Auto Discovery", RFC 8155, DOI 10.17487/RFC8155, April 2017, <https:// www.rfc-editor.org/info/rfc8155>.

[RFC8837] Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "Differentiated Services Code Point (DSCP) Packet Markings for WebRTC QoS", RFC 8837, DOI 10.17487/ RFC8837, January 2021, <https://www.rfc-editor.org/info/rfc8837>.

[RFC8838] Ivov, E., Uberti, J., and P. Saint-Andre, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", RFC 8838, DOI 10.17487/RFC8838, January 2021, <https://www.rfc-editor.org/info/ rfc8838>.

[RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <https://www.rfc-editor.org/info/rfc8843>.

[WebRTC-Gateways] Alvestrand, H. and U. Rauschenbach, "WebRTC Gateways", Work in Progress, Internet-Draft, draft-ietf-rtcweb-gateways-02, 21 January 2016, <https:// tools.ietf.org/html/draft-ietf-rtcweb-gateways-02>.

[XEP-0124] Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J., Stout, L., and W. Tilanus, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, November 2016, <https://xmpp.org/extensions/xep-0124.html>.

[XEP-0166] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, September 2018, <https://xmpp.org/extensions/xep-0166.html>.

IV　RFC8834 原文の著作権について

V　RFC8834 の和訳

概要

WebReal-TimeCommunication (WebRTC) フレームワークは、Web ブラウザ間での音声、ビデオ、テキスト、コラボレーション、ゲームなどによる直接対話型通信をサポートする。このメモでは、WebRTC フレームワークのメディアトランスポートの側面について説明する。WebRTC コンテキストでの Real-time Transport Protocol (RTP) の使用方法を指定し、RTP 機能、プロファイル、および拡張機能がサポートする必要がある要件を示す。

1. はじめに

The Real-timeTransport Protocol (RTP) [RFC3550]は、オーディオやビデオのテレビ会議データやその他のリアルタイムメディアアプリケーションを配信するための、フレームワークを提供する。これまでの作業では、多数のプロファイル、ペイロードフォーマット、その他の拡張によって RTP プロトコルを定義してきた。これらを適切なシグナリングと組み合わせると、多くのテレビ会議システムの基礎となる。

WebReal-TimeCommunication (WebRTC) フレームワークは、2 つのピアの Web ブラウザ間で、直接、対話的、リアルタイムの通信を使用して、音声、ビデオ、コラボレーション、ゲームなどのプロトコルを作成する。このメモでは、RTP フレームワークを WebRTC コンテキストで使用する方法について説明する。すべての WebRTC エンドポイントで実装される RTP 機能のベースラインセットを提案する。

この文書は、WebRTC フレームワーク内での使用を目的としたプロトコルを指定するが、そのコンテキストに限定されない。WebRTC フレームワークの概要は [RFC8825] にある。

この文書の構造を以下に示す。セクション 2 は、このメモを作成し、これらの RTP 機能を選択する方法の概要。セクション 3 は用語の定義。コア RTP プロトコルの要件はセクション 4 で説明されており、サポートされている RTP 拡張はセクション 5 で説明されている。

セクション 6 はネットワーク問題に対する堅牢性を高めるメカニズムの概要、セクション 7 では、会議制御レート適応メカニズムについて説明している。必須の RTP メカニズムの議論は、セクション 8 でパフォーマンス監視とネットワーク管理ツールのレビューで終わる。セクション 9 は、他の RTP および RTP 制御プロトコル（RTCP）拡張機能をこのフレームワークに将来組み込むためのガイドラインを示している。セクション 10 は、シグナリングチャネルに課せられる要件を記述する。セクション 11 は、 RTP フレームワークの機能と WebRTC アプリケーションのプログラミング・インターフェース (API) との関係について説明している。セクション 12 は、 RTP 実装の考慮事項について説明している。セキュリティに関する考慮事項 (セクション 13) と IANA に関する考慮事項 (セクション 14) で終わる。

2　理論的根拠

RTP フレームワークには、RTP データ転送プロトコル、制御プロトコル、および多数の RTP ペイロード形式、プロファイル、拡張機能がある。この add-on の範囲は、オリジナルのプロトコル設計者が想定していなかったさまざまなニーズに対応し、新しいメディア符号化をサポートする。しかし、新しい実装でサポートされる拡張については疑問がある。WebRTC フレームワークの開発は、すべての WebRTC エンドポイントについて、使用可能な RTP 機能および拡張機能を確認する機会を提供する。このビルドは、20 年前の RTP 開発で、拡張機能の使用方法と、インストールされている RTP 実装のベースと互換性がある場合には、拡張機能の使用方法を指定する。

この文書は WebRTC エンドポイントによって実装することができるが、新しいユースケースには有利である。しかし、[RFC7478] で指定されている WEB RTC のユースケースや要件に対応する必要はない。

この文書で定義されている RTP 機能と拡張機能のベースラインセットは、WebRTC フレームワークの要件を対象としているが、他の会議関連の RTP の使用にも広く役立つことが期待されている。特に、この RTP 機能と拡張機能のセットは、他のデスクトップまたはモバイルビデオ会議システム、またはルームベースの高品質テレプレゼンスアプリケーションに適している可能性がある。

3. 用語

この文書のキーワード「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「NOT　RECOMMENDED」、「MAY」、「MUST」は、ここに示すように、すべての大文字が出現した場合、および出現した場合にのみ、BCP 14 [RFC2119][RFC8174]で記述された内容で解釈される。これらのキーワードの小文字または大文字小文字混合の使用は、このメモでは特殊な意味を持つと解釈されない。

以下の追加の用語を定義する。

WebRTC メディアストリーム：WebRTC API [W3C.WD-mediacapture-streams]の W3C によって定義されたメディアストリームの概念。メディアストリームは、0 個以上のメディアストリームトラックで構成される。

メディアストリームトラック：WebRTCAPI の W3C によって定義されたメディアストリームの概念の一部[W3C.WD-mediacapture-streams]。メディアストリームトラックは、マイクやカメラなどのあらゆるタイプのメディアソースからのメディアの個別のストリームであるが、オーディオミックスやビデオコンポジションなどの概念的なソースも可能である。

トランスポート-レイヤーフロー：送信元 IP アドレス、送信元ポート、宛先 IP アドレス、宛先ポート、およびトランスポートプロトコルの特定の 5 タプルによって識別されるトランスポートパケットの単方向フロー。

双方向トランスポート-レイヤーフロー：双方向のトランスポート層のフローは、トランスポート層のフローである。それは対称的である。つまり、逆方向のトランスポート層フローには 5 タプルがあり、順方向パスのトランスポート層フローと比較して、送信元と宛先のアドレスとポートが交換され、トランスポートプロトコルは同じである。

この文書では、[RFC7656] と [RFC8825] の用語を使用している。その他の用語は、RTP 仕様 [RFC3550] の定義に従って使用される。特に、頻繁に使用される用語として、RTP ストリーム、RTP セッション、およびエンドポイントに注意。

4. RTP の使用した WebRTC：コアプロトコル

4.1 RTP と RTCP

リアルタイムトランスポートプロトコル（RTP）[RFC3550]は、WebRTC のメディアトランスポートプロトコルとして実装する必要がある[REQUIRED]。RTP 自体は、RTP データ転送プロトコルと RTP 制御プロトコル（RTCP）の 2 つの部分で構成されている。RTCP は RTP の基本的かつ不可欠な部分であり、すべての WebRTC エンドポイントで実装および使用する必要がある[MUST]。

次の RTP と RTCP 機能は、制限された機能を RTP に実装しながら、すべての WebRTC エンドポイントに必須のものである[REQUIRE]。

- 単一の RTP セッションで複数の同時同期ソース (SSRC) 値の使用をサポートする ([RFC3550] および [RFC8108] に従って多数の SSRC 値を同時に送信する RTP エンドポイントのサポートを含む) 。[RFC8861] で定義されたマルチ SSRC セッションの RTCP 最適化はサポートされている[MAY]; サポートされている場合は、使用法を通知する必要がある[MUST]。
- セッションへの参加時に SSRC をランダムに選択。SSRC 値の衝突検出と解決（セクション 4.8 も参照）。
- RTP ミキサーによって生成されたコントリビューションソース（CSRC）リストを含む RTP データパケット、および CSRC に関連する RTCP パケットの受信のサポート。
- 正しい同期情報を RTCPSenderReports に送信すると、受信側はリップ同期を実装できる。高速 RTP 同期拡張のサポートについては、セクション 5.2.1 を参照。
- 複数の同期コンテキストのサポート。複数の同時 RTP パケットストリームを送信する参加者は、すべてのストリームに単一の RTCP CNAME を使用し、受信者が同期してストリームを再生できるように、単一の同期コンテキストの一部として送信する必要がある[SHOULD]。送信および受信の RTCPSenderReport (SR) 、ReceiverReport (RR) 、SourceDescription (SDES) 、および BYE パケットタイプをサポートする。他の RTCP パケットタイプのサポートは、他の仕様ではサポートされていないことに注意してください。追加の RTCP パケットタイプは、RTP/SAVPF プロファイル (セクション 4.2) およびその他の RTCP 拡張 (セクション 5) で使用される。複数の同時 RTP パケットストリームを送信する参加者は、すべてのストリームに単一の RTCP CNAME を使用し、受信者が同期してストリームを再生できるように、単一の同期コンテキストの一部として送信する必要がある[MUST]。この仕様では、状況によっては RTP ストリームを送信するときに単一の CNAME を使用することが義務付けられている。セクション 4.9 を参照。
- RTCP の送信者レポート（SR）、受信者レポート（RR）、ソース記述（SDES）、および BYE パケットタイプの送受信のサポート。この仕様の他の部分で義務付けられていない限り、他の RTCP パケットタイプのサポートはオプションであることに注意してください[OPTIONAL]。追加の RTCP パケットタイプは、RTP/SAVPF プロファイル（セクション 4.2）およびその他の RTCP 拡張機能（セクション 5）によって使用されることに注意してください。セッション記述プロトコル（SDP）バンドルネゴシエーション拡張機能を実装する WebRTC エンドポイントは、SDP グループ化フレームワークの「mid」属性を使用してメディアストリームを識別する。このようなエンドポイントは、[RFC8843]で説明されている RTCP SDES メディア識別（MID）アイテムを実装する必要がある。[MUST]
- データベース幅の分割のための RTCP 帯域幅の設定、および、例えば SDP 「b=」行 [RFC4566] [RFC3556] を使った RTCP 帯域幅の分割の設定をサポートする。
- [RFC3550]のセクション 6.2 で説明されている短縮された最小 RTCP レポート間隔のサポート。短縮された最小 RTCP レポート間隔を使用する場合、参加者タイムアウト間隔を計算する際に (短縮されていない) 固定の最小間隔が使用される

[MUST] ([RFC3550] のセクション 6.2 と 6.3.5 を参照してください) 。最初の複合 RTCP パケットを送信する前の遅延は、ゼロに設定できる ([RFC8108] によって更新された[RFC3550]のセクション 6.2 を参照。) 。

- 不連続送信のサポート。 RTP を使用すると、エンドポイントはいつでも送信を一時停止および再開できる。再開すると、RTP シーケンス番号は通常どおり 1 ずつ増加するが、RTP タイムスタンプ値の増加は一時停止の期間によって異なる。不連続送信は、一部のオーディオペイロード形式で最も一般的に使用されるが、オーディオ固有ではなく、任意の RTP ペイロード形式で使用できる。
- 不明な RTCP パケットタイプと RTP ヘッダー拡張を無視する。これは、将来の拡張機能、ミドルボックスの動作などの堅牢な処理を保証するためであり、信号が送信されなかった RTP ヘッダー拡張機能または RTCP パケットタイプを受信する可能性がある。

RTP 実装のうち、特に VoiceoverIP (VoIP) のみを使用した引数データシステムの数が重要であることが判明したが、これらの機能はすべてサポートされていない。実装者は、既存の実装との相互運用時に、適切な分解を行うための要件を考慮することをお勧めする。

その他の実装に関する考慮事項についてはセクション 12 で説明している。

4.2 RTP プロファイルの選択

特定のアプリケーションドメインの RTP を完全に指定するには、RTP プロファイルを選択する必要がある。WebRTC を使用するには、[RFC7007]によって拡張された RTCP ベースのフィードバック（RTP/SAVPF）[RFC5124]用の拡張されたセキュア RTP プロファイルを実装する必要がある[MUST]。RTP/SAVPF プロファイルは、基本的な RTP/AVP プロファイル[RFC3551]、RTCP ベースのフィードバック用の RTP プロファイル（RTP/AVPF）[RFC4585]、およびセキュア RTP プロファイル（RTP/SAVP）[RFC3711]を組み合わせたものである。

改善された RTCP タイマーモデルには、RTCP ベースのフィードバック拡張機能[RFC4585]が必要である。これにより、厳密に帯域幅に応じてではなく、イベントに応答して RTCP パケットをより柔軟に送信できる。これは、メディアイベントだけでなく輻輳信号も報告できるようにするために不可欠である。これらの拡張機能により、RTCP 帯域幅を節約することもできる。エンドポイントは通常、フィードバックを必要とするイベントが多数ある場合にのみ、完全な RTCP 帯域幅割り当てを使用する。セクション 5.1 で説明した RTP 会議拡張機能を利用するには、タイマールールも必要である。

> 注意：RTP/AVPF プロファイルで定義された拡張 RTCP タイマーモデルは、RTCP 帯域幅値や「trr-int」などのパラメータ設定に関する制約がある場合、RTP/AVP または RTP/SAVP プロファイルだけを実装する従来のシステムと下位互換性がある。ゲートウェイを介した RTP/ (S) AVP エンドポイントとのインターワーキングで最も重要な要素は、「trr-int」パラメータを 4 秒に設定することである。[RFC8108]のセクション 7.1.3 を参照。

セキュア RTP（SRTP）プロファイル拡張[RFC3711]は、メディア暗号化、整合性保護、リプレイ保護、および限定された形式のソース認証を提供するために必要である。WebRTC エンドポイントは、基本的な RTP/AVP プロファイルまたは RTP/AVPF プロファイルを使用してパケットを送信してはいけない[MUST NOT]。生成されるすべての RTP および RTCP パケットを保護するために、完全な RTP/SAVPF プロファイルを使用する必要がある[MUST]。言い換えると、実装では SRTP とセキュア RTCP（SRTCP）を使用する必要がある。RTP/SAVPF プロファイルは、暗号スイート、DTLS-SRTP 保護プロファイル、キーイングメカニズム、および[RFC8827]で説明されているその他のパラメータを使用して構成する必要がある[MUST]。

4.3 RTP ペイロードフォーマットの選択

WebRTC エンドポイントの必須の実装オーディオコーデックと RTP ペイロード形式は、[RFC7874]で定義されている。WebRTC エンドポイントの必須の実装ビデオコーデックと RTP ペイロード形式は、[RFC7742]で定義されている。WebRTC エンドポイントは、RTP ペイロード形式と関連するシグナリングが定義されている他のコーデックを追加で実装してもよい[MAY]。

WebRTC エンドポイントは、RTP セッションの他の参加者が、どれほど一般的であっても、RTP ペイロード形式を理解していると想定することはできない。RTP ペイロードタイプ番号と特定の RTP ペイロードフォーマットの特定の構成との間のマッピングは、それらのペイロードタイプ/フォーマットを使用する前に合意する必要がある[MUST]。SDP コンテキストでは、これは、「m=」行に関連付けられた「a=rtpmap:」および「a=fmtp:」属性と、RTP ペイロード形式を構成するために必要な他の SDP 属性を使用して実行できる。

エンドポイントは、一意の RTP ペイロードフォーマット構成ごとに異なる RTP ペイロードタイプ番号を使用している限り、複数の RTP ペイロードフォーマットまたは単一の RTP ペイロードフォーマットの複数の構成のサポートを通知できる。セクション 4.8 で概説されているように、RTP ペイロードタイプ番号は、RTP パケットストリームをシグナリングコンテキストに関連付けるために使用されることがある。この関連付けは、各コンテキストで一意の RTP ペイロードタイプ番号が使用されている場合に可能である。たとえば、RTP パケットストリームで使用される RTP ペイロードタイプ番号を、SDP のメディアセクションの「a=rtpmap:」行で通知されるペイロ

ードタイプと比較することにより、RTP パケットストリームを SDP「m=」行に関連付けることができる。これにより、次の考慮事項が発生する。

RTP パケットストリームが関連付けされている場合、RTP ペイロードタイプに基づいてコンテキストの通知が行われると、RTP ペイロードタイプの番号は、コンテキストの通知を一意に行わなければならない[MUST]。

RTP ペイロードタイプの設定が複数のコンテキストで使用されている場合、異なる RTP ペイロードタイプの列挙型は、指定されたコンテキストに関連して一意性を保証する。

RTP ペイロードタイプ番号を使用して RTP パケットストリームを指定されたコンテキストに関連付けない場合は、同じ RTP ペイロードタイプ番号を使用して、同じ RTP ペイロードフォーマット設定を複数のコンテキストに指定できる。

単一の RTP セッション内で、単一の RTP ペイロードタイプ番号を異なる RTP ペイロード形式または同じ RTP ペイロード形式の異なる構成に割り当ててはならない（SDBPUNDLE グループ[RFC8843]の「m=」行は単一の RTP セッション）[MUST NOT]。

複数の RTP ペイロード形式のサポートを通知したエンドポイントは、以前にデコード機能の制限を通知していない限り、いつでもこれらのペイロード形式のデータを受け入れることができなければならない[MUST]。複数のタイプのメディア（オーディオやビデオなど）が同じ RTP セッションで送信される場合、この要件は制約される。このような場合、ソース（SSRC）は、そのソースによって送信されているメディアのタイプに対して通知された RTP ペイロード形式間の切り替えのみに制限される。セクション 4.4 を参照してください。コーデックを変更することによる高速レート適応をサポートするために、RTP は、セッションセットアップ中にシグナリングされた単一の SSRC によって使用される RTP ペイロードフォーマット間の変更に対する事前シグナリングを必要としない。

異なる RTP クロックレートを使用する 2 つの RTP ペイロードタイプ間で変更を実行する場合、RTP 送信者は[RFC7160]のセクション 4.1 の推奨事項に従わなければならない[MUST]。RTP 受信機は、RTP セッションでクロックレートを切り替えるソースをサポートするために、[RFC7160]のセクション 4.3 の推奨事項に従わなければならない[MUST]。受信者に対するこれらの推奨事項は、送信者が単一のクロックレートのみを使用する場合と下位互換性がある。

4.4 RTP セッションの使用

RTP を使用して通信する一連のエンドポイント間の関連付けは、RTP セッション[RFC3550]と呼ばれる。エンドポイントは、同時に複数の RTP セッションに関与できる。マルチメディアセッションでは、通常、各タイプのメディアは個別の RTP セッションで伝送される（たとえば、オーディオ用に 1 つの RTP セッションを使用し、ビデオ用に異なるトランスポート層フローを使用する個別の RTP セッションを使用する）。WebRTC エンドポイントは、この方法でマルチメディアセッションのサポートを実装する必要があり[REQUIRED]、レガシーシステムとの互換性のために異なるトランスポート層フローを使用して各 RTP セッションを分離する（これはセッション多重化と呼ばれることもある）。

しかし、現代のネットワークでは、ネットワークアドレス/ポート変換器 (NAT/NAPT) とファイアウォールを広範囲に使用する場合、RTP アプリケーションがトランスポート・レイヤ・フローをサポートするのが望ましい。これは、単一のトランスポート層フローを構成する単一の RTP セッションですべての RTP パケットストリームを送信することによって実行できる。これにより、セクション 12.1.3 で説明されているように、一部のサービス品質メカニズムの使用が妨げられる。したがって、[RFC8860]（これは SSRC 多重化と呼ばれることもある）によると、単一のトランスポート層フローを使用する単一の RTP セッションで、メディアタイプに関係なくすべての RTP パケットストリームのトランスポートをサポートするための実装も必要である[REQUIRED]。1 つの RTP セッションで複数のタイプのメディアを使用する場合は、その RTP セッションのすべての参加者がこの使用法に同意する必要がある[MUST]。SDP コンテキストでは、[RFC8843]で説明されているメカニズムを使用して、単一の RTP セッションを形成する RTP パケットストリームのそのようなバンドルに信号を送ることができる。

あとで、別の RTP セッションの構造と多重化メソッド別のシナリオの有用性について説明する。 [RFC8872]。

4.5 RTP と RTCP 多重化

これまで、RTP と RTCP は、別々のトランスポート層フローで実行されてきた（たとえば、RTP セッションごとに 2 つの UDP ポート、1 つは RTP 用、もう 1 つは RTCP 用）。ネットワークアドレス/ポート変換（NAT / NAPT）の使用が増えると、複数の NAT バインディングを維持するのにコストがかかる可能性があるため、これは問題になる。また、RTP トラフィックを許可するには複数のポートを開く必要があるため、ファイアウォールの管理が複雑になる。これらのコストとセッションのセットアップ時間を削減するには、単一のトランスポート層フローでの RTP データパケットと RTCP 制御パケットの多重化をサポートする実装が必要である[RFC5761][REQUIRED]。このような RTP および RTCP 多重化は、使用する前にシグナリングチャネルでネゴシエートする必要がある[MUST]。 SDP がシグナリングに使用される場合、このネゴシエーションは[RFC5761]で定義されたメカニズムを使用する必要がある。実装では、RTP と RTCP を別々のトランスポート層フローで送信することもサポートできるが、これはオプションで実装できる[OPTIONAL]。実装が別々のトランスポート層フローで送信される RTP と RTCP をサポートしていない場合は、[RFC8858]で定義されているメカニズムを使用していること

を示さなければならない[MUST]。

単一のトランスポート層フローに多重化された RTP と RTCP を使用すると、たとえトラフィックがアクティブでなくても、そのポートでのトラフィックが定期的に生成されることに注意してください。これは、NAT バインディングを存続させるのに役立つ[RFC6263]。

4.6 縮小サイズ RTCP

RTCP パケットは通常、複合 RTCP パケットとして送信され、[RFC3550]では、これらの複合パケットが SR または RR パケットで始まる必要がある。RTP / AVPF プロファイル[RFC4585]で頻繁に RTCP フィードバックメッセージを使用する場合、これらの統計はすべてのパケットで必要になるわけではなく、平均 RTCP パケットサイズが不必要に大きくなる。これにより、RTCP 帯域幅共有内で RTCP パケットを送信できる頻度が制限される可能性がある。

この問題を回避するために、[RFC5506]は、平均 RTCP メッセージサイズを減らし、より頻繁なフィードバックを可能にする方法を指定している。頻繁なフィードバックは、リアルタイムアプリケーションがネットワークの状態の変化をすばやく認識し、送信とエンコードの動作を適応できるようにするために不可欠である。実装は、非複合 RTCP フィードバックパケットの送受信をサポートする必要がある[RFC5506][MUST]。非複合 RTCP パケットの使用は、シグナリングチャネルを使用してネゴシエートする必要がある[MUST]。SDP がシグナリングに使用される場合、このネゴシエーションは[RFC5506]で定義された属性を使用する必要がある[MUST]。下位互換性のために、リモートエンドポイントがシグナリング交換での非複合 RTCP の使用に同意しない場合、複合 RTCP フィードバックパケットの使用をサポートするための実装も必要である[RQUIRED]。

4.7 対称 RTP/RTCP (シンメトリ RTP/RTCP)

NAT およびファイアウォールデバイスのトラバーサルを容易にするために、対称 RTP [RFC4961]を実装および使用するための実装が必要である[REQUIRED]。対称 RTP を使用する理由は、主に、送信および受信 RTP パケットストリームと RTCP が実際に双方向のトランスポート層フローであることを確認することにより、NAT とファイアウォールの問題を回避するためである。これにより、NAT とファイアウォールのピンホールが維持され、受信方向が目的の受信者が実際に望んでいるトランスポート層フローであるという同意を示すのに役立つ。さらに、NAT マッピングやファイアウォールの状態が不必要に肥大化することがないため、リソース、特にエンドポイントのポートだけでなく、ネットワークのポートも節約できる。ネットワークに保持されるフローごとの QoS 状態の量も削減される。

4.8 RTP 同期ソース（SSRC）の選択

シグナリングされた RTP 同期ソース（SSRC）識別子をサポートするには、実装が必要である[REQUIRED]。SDP を使用する場合、これはセクションで定義されている「a=ssrc:」SDP 属性を使用して実行する必要がある[MUST]。[RFC5576]の 4.1 と 5、および[RFC5576]のセクション 6.2 で定義されている「previous-ssrc」ソース属性。[RFC5576]で定義されている他の SSRC ごとの属性がサポートされる場合がある[MAY]。

シグナリングされた SSRC 識別子のサポートが義務付けられているが、RTP セッションでのそれらの使用はオプションである[OPTIONAL]。実装は、事前に明示的に通知されていない SSRC を使用して RTP および RTCP パケットを受け入れるように準備する必要がある[MUST]。[RFC3550]によると、実装はランダムな SSRC 割り当てをサポートし[MUST]、SSRC 衝突検出と解決をサポートする必要がある[MUST]。シグナリングされた SSRC 値を使用する場合、[RFC5576]のセクション 5 で説明されているように、衝突検出を実行する必要がある[MUST]。

多くの場合、RTP パケットストリームを非 RTP コンテキストに関連付けることが望ましい。WebRTC API のユーザの場合、SSRC と MediaStreamTracks 間のマッピングがセクション 11 に従って提供される。ゲートウェイまたはその他の使用法の場合、RTP パケットストリームを、SDP を使用してフォーマットされたセッション記述の「m=」行に関連付けることができる。SSRC が通知される場合、これは簡単である（SDP では、「a=ssrc:」行はメディアレベルになり、「m=」行との直接関連付けが可能になる）。SSRC がシグナリングされていない場合、RTP パケットストリームで使用される RTP ペイロードタイプ番号は、多くの場合、そのパケットストリームをシグナリングコンテキストに関連付けるのに十分である。たとえば、このメモのセクション 4.3 で説明されているように RTP ペイロードタイプ番号が割り当てられている場合、RTP パケットストリームで使用される RTP ペイロードタイプは、メディアレベルの SDP「a=rtpmap:」行の値と比較できる。SDP で、「m=」行にマップする。

4.9 RTCP 正規名（CNAME）の生成

RTCP 正規名（CNAME）は、RTP エンドポイントの永続的なトランスポートレベル識別子を提供する。RTP エンドポイントの SSRC 識別子は、衝突が検出された場合、または RTP アプリケーションが再起動された場合に変更される可能性があるが、その RTCP CNAME は、RTCPeerConnection [W3C.WebRTC]の期間中は変更されないため、RTP エンドポイントを一意にすることができる。関連する一連の RTP セッション内で RTP パケットストリームが識別され、関連付けられる。

各 RTP エンドポイントには少なくとも 1 つの RTCPCNAME が必要であり[MUST]、その RTCPCNAME は RTCPeerConnection 内で一意である必要がある[MUST]。RTCP CNAME は、特定の同期コンテキストを識別する。つまり、単一の RTCP CNAME に関連付けられているすべての SSRC は、共通の参照クロックを共有する。エンドポイントに複数の非同期基準クロックに関連付けられた SSRC があり、したがって異なる同期コンテキストがある場合は、同期コンテキストごとに 1 つずつ、複数の RTCPCNAME を使用する必要がある。

セクション 11 の説明を考慮に入れると、WebRTC エンドポイントは単一の RTCPeerConnection に属する RTP セッションで複数の RTCP CNAME を使用してはならない（つまり、RTCPeerConnection は同期コンテキストを形成する）[MUST NOT]。RTP ミドルボックスは、複数の RTCP CNAME に関連付けられた RTP パケットストリームを生成する場合がある[MAY]。これにより、マルチパーティ RTP セッションの一部である複数の異なるエンドポイントからのメディアを再同期する必要がなくなる。

RTP 仕様[RFC3550]には、一意の RTP CNAME を選択するためのガイドラインが含まれているが、NAT デバイスが存在する場合はこれらのガイドラインでは不十分である。さらに、長期的な永続的識別子はプライバシーの観点から問題になる可能性がある（セクション 13）。したがって、WebRTC エンドポイントは、1 つの例外を除いて、[RFC7022]に続いて、RTCPeerConnection ごとに新しい一意の短期永続 RTCPCNAME を生成する必要がある[MUST]。作成時に明示的に要求された場合、RTCPeerConnection は、共通の同一生成元コンテキスト内で既存の RTCPeerConnection と同じ CNAME を使用できる[MAY]。

WebRTC エンドポイントは、RTP 仕様[RFC3550]で指定された構文制限に一致する CNAME の受信をサポートする必要があり[MUST]、上記の形式に従って CNAME が選択されるとは想定できない。

4.10 うるう秒の処理

RTP メディアの再生と同期への影響を制限するためのうるう秒の処理に関する[RFC7164]に記載されているガイドラインに従う必要がある[SHOULD]。

5. WebRTC による RTP の使用：拡張

WebRTC コンテキストでは、完全な機能を得るために必要な、またはベースラインパフォーマンスを改善するために非常に有用な RTP 拡張機能が多数ある。これらの拡張機能の 1 つは会議に関連しているが、他の拡張機能はより一般的なものである。次のサブセクションでは、WebRTC 内での使用が必須または推奨されるさまざまな RTP 拡張について説明する。

5.1 会議の拡張とトポロジ

RTP は、本質的にグループ通信をサポートするプロトコルである。グループを実装するには、各エンドポイントから RTP パケットストリームを RTP ミドルボックスに送信してトラフィックを再分配するか、エンドポイント間でユニキャスト RTP パケットストリームのメッシュを使用するか、または IP マルチキャストグループを使用して RTP パケットストリームを分配する。これらのトポロジは、[RFC7667] で論じられているように、多くの方法で実装することができる。

IP マルチキャストグループの使用は IPTV システムでは一般的であるが、 RTP ミドルボックスに基づくトポロジは対話型ビデオ会議環境では支配的である。共通の RTP セッションを作成するためのユニキャストトランスポート層フローのメッシュに基づくトポロジは、これまで広く導入されていなかった。したがって、WebRTC エンドポイントは、IP マルチキャストグループに基づくトポロジや、単一の RTP セッションとして構成されたポイントツーマルチポイントメッシュなどのメッシュベースのトポロジをサポートすることは期待されていない（[RFC7667]の用語で「Topo-Mesh」）。

ただし、独立した RTCPeerConnections [W3C.WebRTC] を使用して WebRTC に実装された、複数の RTP セッションを使用して構築されたポイントツーマルチポイントメッシュは、WebRTC での使用が期待されるため、サポートする必要がある。

この文書に従って実装された WebRTC エンドポイントは、 [RFC7667] で記述されているすべてのトポロジをサポートすることが期待されている。ここで、RTP エンドポイントは、ピアが RTP パケットストリームの輻輳制御の実行に参加できる場合に限り、一部のピアデバイスとの間でユニキャスト RTP パケットストリームを送受信する。ピアデバイスは、別の RTP エンドポイントにすることも、RTP パケットストリームを他の RTP エンドポイントに再配布する RTP ミドルボックスにすることもできる。この制限は、RTP ミドルボックスベースのトポロジの一部が WebRTC での使用に適していないことを意味する。具体的には：

- ビデオスイッチングマルチポイントコントロールユニット (MCU) (Topo-Video-switch-MCU) は、輻輳制御および Quality of Service レポートに RTCP を使用することが問題になるため、使用すべきではない[SHOULD NOT] ([RFC7667] セクション 3.8 を参照) 。
- Relay-Transport Translator (Topo-Ptm-Trn-Translator) トポロジを安全に使用するには、輻輳制御アルゴリズムまたは、まだ標準化されていないポイントツーマルチポイントを処理する RTP サーキットブレーカーが必要であるため、使用すべきでは

ない[SHOULD NOT]。

次のトポロジを使用できるが、注意が必要な問題がいくつかある。

- RTCP 終端のコンテンツ修正 MCU (Topo-RTCP-terminating-MCU) を使用してもよい[MAY]。この RTP トポロジでは、RTP ループの検出とアクティブな送信者の識別は、WebRTC アプリケーションが担当する。クライアントは RTP レイヤで互いに分離されているため、RTP はこれらの機能を支援できない ([RFC7667] セクション 3.9 を参照) 。

セクション 5.1.1 から 5.1.6 で説明する RTP 拡張は、集中型会議で使用されるように設計されている。集中型会議では、RTP ミドルボックス(例えば、会議ブリッジ)が参加者の RTP パケットストリームを受信し、他の参加者に配信する。これらの拡張は相互運用性のために必要ではない。これらの拡張を実装していない RTP エンドポイントは正常に動作するが、パフォーマンスが低下する可能性がある。リストされた拡張機能のサポートにより、エクスペリエンスの品質が大幅に向上する。妥当なベースライン品質を提供するために、WebRTC エンドポイントでサポートする必要がある拡張機能もある。

RTCP 会議拡張は、「Real-time Transport Control Protocol (RTCP) ベースのフィードバックのための拡張 RTP プロファイル (RTP/AVPF)」 [RFC4585] と「フィードバック付き RTP 音声ビジュアルプロファイルのコーデック制御メッセージ (AVPF)」 [RFC5104] で定義されている。これらは、このプロファイルのセキュアなバリエーション (RTP/SAVPF) [RFC5124] によって完全に使用できる。

5.1.1 完全なイントラリクエスト (FIR)

「完全なイントラリクエスト」(FIR) メッセージは、セクション 3.5.1 および 4.3.1 (Codec Control Messages [RFC5104])で定義する。セッションの参加者から新しいイントラ画像をミキサーに要求するために使用される。これは、ソースを切り替えるときに使用され、受信者が長い予測チェーンでビデオまたは他の予測メディアエンコーディングをデコードできるようにする。メディアを送信している WebRTC エンドポイントは、受信した FIR フィードバックメッセージを理解して対応しなければならない[MUST]。これにより、集中ミキサーベースの会議を使用する場合のユーザエクスペリエンスが大幅に向上するためである。FIR メッセージの送信のサポートはオプションとなる[OPTIONAL]。

5.1.2 画像損失表示 (PLI)

「画像損失表示」(PLI)メッセージは、セクション 6.3.1 RTP/AVPF プロファイル [RFC4585] で定義する。これは、デコーダのコンテキストが失われ、何らかの方法で修復してほしいことを送信者エンコーダに伝えるために、受信者によって使用される。要求を満たすには複数の方法があるため、これは上記の FIR とは意味的に異なる。メディアを送信している WebRTC エンドポイントは、損失耐性メカニズムとして PLI フィードバックメッセージを理解し、応答しなければならない[MUST]。受信者は PLI メッセージを送信してもよい[MAY]。

5.1.3 スライス・ロス表示 (SLI)

「スライス・ロス表示」(SLI)メッセージは、セクション 6.3.2 RTP/AVPF プロファイル [RFC4585] で定義する。これは、1 つ以上の連続するマクロブロックの損失または破損を検出したこと、およびこれらを何らかの方法で修復したいことをエンコーダに伝えるために、受信者によって使用される。マクロブロックの概念をサポートするコーデックを使用するときにスライスが失われた場合、受信者は SLI フィードバックメッセージを生成することが推奨される[RECOMMENDED]。SLI フィードバックメッセージを受信した送信者は、失われたスライスの修復を試みるべきである[SHOULD]。

5.1.4 基準画像選択表示 (RPSI)

「基準画像選択表示」(RPSI) メッセージは、セクション 6.3.3 RTP/AVPF プロファイル [RFC4585] で定義する。一部のビデオエンコーディング標準では、予測コーディングに最新のものよりも古い参照画像を使用できる。このようなコーデックが使用されており、エンコーダとデコーダの同期が失われたことをエンコーダが学習した場合、正しいと分かっている参照画像を将来のコーディングのベースとして使用することができ、RPSI メッセージによって、これをシグナリングできる。エンコーダとデコーダの同期が失われたことを検出する受信者は、使用されているコーデックが参照画像選択をサポートする場合、RPSI フィードバックメッセージを生成する必要がある[SHOULD]。そのような RPSI メッセージを受信する RTP パケットストリーム送信者は、利用可能な帯域幅制約内で使用されているコーデックと共に、参照画像を変更することが可能であればそのメッセージに基づいて動作すべきである[SHOULD]。

5.1.5 時間的-空間的トレードオフ要求 (TSTR)

「時間的-空間的トレードオフ要求」(TSTR)および通知は、セクション 3.5.2 および 4.3.2 [RFC5104] で定義する。この要求を使用して、ビデオエンコーダに時間的解像度と空間的解像度との間のトレードオフを変更するように要求することができる。例えば、高い空間的画質を好むが、低いフレームレートを好むように要求することができる。TSTR 要求および通知のサポートはオプションである[OPTIONAL]。

5.1.6 一時的な最大メディアストリームビットレート要求 (TMMBR)

「一時的な最大メディアストリームビットレート要求」 (TMMBR) フィードバックメッセージは、3.5.4 および 4.2.1 (Codec Control

Messages [RFC5104]で定義する 。この要求とそれに対応する一時最大メディアストリームビットレート通知（TMMBN）メッセージ[RFC5104]は、この受信者が使用できる帯域幅の量に現在の制限があることを送信者に通知するためにメディア受信者によって使用される。

これにはさまざまな理由が考えられる。たとえば、RTP ミキサーはこのメッセージを使用して、他のセッション参加者に向かって存在するボトルネックに合うように、ミキサーによって転送される送信者のメディアレートを制限できる (メディアトランスコーディングは行われない) 。メディアを送信する WebRTC エンドポイントは、TMMBR メッセージのサポートを実装するために必要であり[REQUIRED]、SSRC 用に受信した TMMBR メッセージによって設定された帯域幅制限に従う必要がある[MUST]。TMMBR メッセージの送信はオプションである[OPTIONAL]。

5.2 ヘッダー拡張

RTP 仕様 [RFC3550] は、帯域内データを含む RTP ヘッダー拡張を含める機能を提供するが、拡張のフォーマットとセマンティクスは十分に規定されていない。ヘッダー拡張の使用は WebRTC では任意であるが、もしそれが使用されるなら、RTP ヘッダー拡張に明確な意味を与えるので、 [RFC8285] で定義された RTP ヘッダー拡張の一般的なメカニズムに従ってフォーマットされ、シグナリングされなければならない[MUST]。

[RFC8285] で述べられているように、ヘッダー拡張が「ヘッダー拡張が無視されるように設計されている」であるという RTP 仕様からの要件 [RFC3550] は成り立つ。具体的には、ヘッダー拡張は、相互運用性に影響を与えることなく受信者によって安全に無視され得るデータに対してのみ使用されなければならず[MUST]、拡張の存在によってストリームがシグナリングされる方法と互換性がない方法で、パケットの残りの部分の形式または性質が変更したときに使用されてはならない[MUST] (例えば、ペイロード型によって定義されるように)。RTP ヘッダー拡張の有効な例としては通常の RTP 情報に追加されるが、相互運用性を損なうことなく無視できるメタデータがある。

5.2.1 高速同期

多くの RTP セッションでは、オーディオ、ビデオ、およびその他のコンテンツ間の同期が必要である。この同期は、RTP 仕様 [RFC3550] に記述されているように、RTCP SR パケットに含まれる情報を使用して受信者によって実行される。しかし、この基本的なメカニズムは遅くなる可能性があるので、 [RFC6051] で記述されている高速 RTP 同期拡張が RTCP SR ベースの同期に加えて実装されることが推奨される[RECOMMENDED]。

このヘッダー拡張は [RFC8285] で記述されている一般的なヘッダー拡張フレームワークを使用するので、使用する前にネゴシエートする必要がある。

5.2.2 クライアントからミキサーへのオーディオレベル

クライアントからミキサーへのオーディオレベル拡張 [RFC6464] は、ヘッダーが付加されたパケット内のオーディオアクティビティのレベルをミキサーに通知するためにエンドポイントによって使用される RTP ヘッダー拡張である。これにより、RTP ミドルボックスは、ペイロードのデコードまたは詳細な検査を行わずにミキシングまたは選択の決定を行うことができ、一部のタイプのミキサーの複雑さが軽減される。また、受信者のデコードリソースを節約でき、オーディオアクティビティレベルに基づいて最も関連性の高い RTP パケットストリームだけをデコードできる。

クライアントからミキサーへのオーディオレベルヘッダー拡張 [RFC6464] を実装しなければならない[MUST]。それは

実装が [RFC6904] に従ってヘッダー拡張を暗号化できることが要求される[REQUIRED]。なぜならこれらのヘッダー拡張に含まれる情報は機密であると考えられるためである。この暗号化の使用が推奨される[RECOMMENDED]。ただし、API またはシグナリングを使用して暗号化の使用を明示的に無効にできる。

このヘッダー拡張は [RFC8285] で記述されている一般的なヘッダー拡張フレームワークを使用するので、使用する前にネゴシエートする必要がある。

5.2.3 ミキサーからクライアントへのオーディオレベル

ミキサーからクライアントオーディオレベルヘッダ拡張 [RFC6465] は、RTP ミキサーによって共通ソースストリームに混合された異なるソースのオーディオレベルを有するエンドポイントを提供する。これにより、CSRC フィールドに含まれるかどうかだけでなく、各セッション参加者の相対的なアクティビティレベルをユーザインタフェースで指定できる。これは重要でない関数の純粋な最適化であるため、実装するのは任意となる[OPTIONAL]。このヘッダー拡張が実装される場合、実装が [RFC6904] に従ってヘッダー拡張を暗号化できることが要求される[REQUIRED]。なぜなら、これらのヘッダー拡張に含まれる情報は機密であると考えられるからである。API またはシグナリングによって暗号化が明示的に無効にされていない限り、この暗号化を使用することがさらに推奨される。[RECOMMENDED]

このヘッダー拡張は [RFC8285] で記述されている一般的なヘッダー拡張フレームワークを使用するので、使用する前にネゴシエートする必要がある。

5.2.4 メディアストリームの識別

SDP バンドルネゴシエーション拡張を実装する WebRTC エンドポイントは、SDP Grouping Framework 「mid」属性を使用してメディアストリームを識別する。そのようなエンドポイントは、 [RFC8843] で記述される RTP MID ヘッダー拡張を実装しなければならない [MUST]。

このヘッダー拡張は [RFC8285] で記述されている一般的なヘッダー拡張フレームワークを使用するので、使用する前にネゴシエートする必要がある。

5.2.5 ビデオオリエンテーションの調整

ビデオを送受信する WebRTC エンドポイントは、セクション 4 [RFC7742] で示すビデオオリエンテーションの調整 (CVO) RTP ヘッダー拡張を実装しなければならない [MUST]。

このヘッダー拡張は [RFC8285] で記述されている一般的なヘッダー拡張フレームワークを使用するので、使用する前にネゴシエートする必要がある。

6. WebRTC による RTP の使用 : トランスポートの堅牢性の向上

RTP パケットストリームをパケット損失に対して堅牢にし、損失によるメディア品質への影響を軽減できるツールがある。ただし、一般的に非堅牢なストリームと比較してオーバーヘッドが増加する。オーバーヘッドを考慮する必要があり、集約ビットレートはネットワークの混雑を避けるためにレート制御されなければならない [MUST] セクション 7) 。その結果、堅牢性を向上させるには、より低い基本エンコーディング品質が必要になる場合があるが、より少ないエラーでその品質を提供できる可能性がある。次のサブセクションで説明するメカニズムを使用して、パケット損失に対する耐性を向上させることができる。

6.1 否定応答と RTP 再送信

RTP/SAVPF プロファイルをサポートする結果として、実装では RTP データパケット [RFC4585] に対して否定応答 (NACK) を送信できる。このフィードバックを使用すると、RTCP フィードバックチャネルの容量制限に応じて、特定の RTP パケットの損失を送信者に通知できる。送信者は、この情報を使用して既知の損失パケットを補うためにメディアエンコーディングを適合させることにより、ユーザエクスペリエンスを最適化できる。

RTP パケットストリーム送信者は、セクション 6.2.1 ([RFC4585]で定義された一般的な NACK メッセージを理解する必要があるが [REQUIRED]、このフィードバックの一部または全部を無視してもよい [MAY] (セクション 4.2 [RFC4585] を参照) 。受信者は損失した RTP パケットに対して NACK を送信してもよい [MAY]。NACK をいつ送信するかについてのガイドラインは [RFC4585] で提供される。受信者が損失した RTP パケットごとに NACK を送信することは期待されない ; むしろ、パケット損失イベントについて送信者に伝える価値があるかどうかについて情報に基づいた決定を行うために、NACK フィードバックを送信するコストと損失パケットの重要性を考慮する必要がある。

RTP 再送信ペイロードフォーマット [RFC4588] は、NACK フィードバックに基づいて損失したパケットを再送信する機能を提供する。再送信は、再送信されたパケットが有効な時間内に到着するように、インタラクティブなリアルタイムアプリケーションでは注意して使用する必要があるが、ネットワーク RTT が比較的低い環境では効果的である。(RTP 送信者は、セクション 6.4.1 ([RFC3550]の最後で説明されているように、RTCP SR および RR パケットの情報を使用して受信者への RTT を推定できる。再送信を使用すると、転送 RTP 帯域幅も増加する可能性があり、元のパケット損失がネットワーク輻輳によって引き起こされた場合は、パケット損失が増加する可能性がある。しかし、デコーダ状態を修復するための重要な損失パケットの再送信は、完全なイントラフレームを送信するよりもコストが低い可能性があることに留意したい。NACK に応答して RTP パケットをむやみに再送信することは適切でない。RTP 再送信を使用する前に、失われたパケットの重要性とパケットが有効な時間内に到着する可能性を考慮する必要がある。

受信者は、SSRC 多重化を使用して送信される RTP 再送パケット [RFC4588] のサポートを実装することが要求され [REQUIRED]、セッション多重化を使用して送信される RTP 再送パケットもサポートしてもよい [MAY]。RTP 再送ペイロードフォーマットのサポートがネゴシエートされ、送信者が NACK で参照されるパケットの再送が有用であると考えている場合、送信者は NACK に応答して RTP 再送パケットを送信してもよい。送信者は、すべての NACK パケットを再送信する必要はない。

6.2 前方誤り訂正 (FEC)

順方向誤り訂正 (FEC) の使用は、一定の帯域幅オーバーヘッドを犠牲にして、ある程度のパケット損失に対して効果的な保護を提供できる。RTP で使用するために定義されている FEC 方式がいくつかある。これらの方式の中には、特定の RTP ペイロード形式に固有のものもあれば、RTP パケット全体で動作し、任意のペイロード形式で使用できるものもある。冗長エンコーディングまたは FEC を使用すると、再生遅延が増加することに注意すること。これは、FEC 方式とそのパラメータを選択するときに考慮する必要がある。

WebRTC エンドポイントは、 [RFC8854] で示されている FEC の使用に関する推奨事項に従わなければならない[MUST]。WebRTC エンドポイントは、他の種類の FEC をサポートしてもよいが[MAY]、使用する前にネゴシエートしなければならない[MUST]。

7. WebRTC による RTP の使用：レート制御とメディア適応

WebRTC は、有線と無線の両方を含むさまざまなリンクテクノロジーを使用して、世界中の大規模なユーザグループを相互接続する異種ネットワーク環境で使用される。その結果、ユーザ間のネットワークパスには、一方向の遅延、使用可能なビットレート、負荷レベル、およびトラフィック混合が幅広く変化する可能性がある。個々のエンドポイントは、1 つ以上の RTP パケットストリームを各参加者に送信でき、複数の参加者が存在する場合もある。これらの RTP パケットストリームにはそれぞれ異なるタイプのメディアを含めることができ、メディアタイプ、ビットレート、RTP パケットストリームの数、およびトランスポート層フローは非常に非対称になる。非 RTP トラフィックは、RTP トランスポート層フローとネットワークパスを共有できる。ネットワーク環境は予測不可能または安定していないため、WebRTC エンドポイントは、生成する RTP トラフィックが利用可能なネットワーク容量の変化に適合できることを保証しなければならない[MUST]。

WebRTC のユーザのエクスペリエンスの品質は、ネットワークの制限に対するメディアの効果的な適応に大きく依存する。エンドポイントは、非常に短い時間を除き、ネットワークパスがサポートできるよりも大幅に多くのデータを送信しないように設計する必要がある。そうしないと、高レベルのネットワークパケット損失または遅延スパイクが発生し、メディア品質が低下する。ネットワークパスのキャパシティに関する制限要因は、リンクの帯域幅である場合もあれば、リンク上の他のトラフィックとの競合である場合もある(これは、WebRTC 以外のトラフィック、他の WebRTC フローによるトラフィック、または同じセッション内の他の WebRTC フローとの競合である可能性がある)。

したがって、効果的なメディア輻輳制御アルゴリズムは、WebRTC フレームワークの重要な部分である。しかし、この文書の執筆時点では、WebRTC のフローのような対話型メディアアプリケーションに使用できる標準的な輻輳制御アルゴリズムはない。RTCPeerConnections の輻輳制御アルゴリズムのいくつかの要件は [RFC8836] で議論されている。もしこれらの要求を満たす標準化された輻輳制御アルゴリズムが将来開発されるなら、この文書はその使用を強制するために更新される必要があるだろう。

7.1 境界条件とサーキットブレーカー

WebRTC エンドポイントは、 [RFC8083] で記述されている RTP サーキットブレーカーアルゴリズムを実装しなければならない[MUST]。RTP サーキットブレーカーは、アプリケーションが極度のネットワーク輻輳状態を認識して対応できるように設計されている。しかし、RTP サーキットブレーカーは、輻輳が極端になるまでトリガされない可能性があるため、輻輳制御の代替とみなすことはできず、アプリケーションはネットワーク容量の変化に適応できるように輻輳制御も実装しなければならない[MUST]。輻輳制御アルゴリズムは、標準化された輻輳制御アルゴリズムが利用可能になるまで独自のものでなければならない。将来の RTP 輻輳制御アルゴリズムは、サーキットブレーカーによって許容されるエンベロープ内で動作することが期待される。

セッション確立シグナリングは、必然的にメディアビットレートが準拠する境界も確立する。メディアコーデックの選択は、アプリケーションが有用な品質を提供するために利用できるサポートビットレートの上限と下限、および存在するパケット化の選択を提供する。さらに、シグナリングチャネルは、例えば SDP 「b=AS:」または「b=CT:」ラインおよび RTP/AVPF TMMBR メッセージを使用して、最大メディアビットレート境界を確立することができる (セクション 5.1.6) 。RTP パケットストリームを送信するときは、ピアから受信した SDP 「b=AS:」または「b=CT:」回線のような、シグナリングされた帯域幅の制限に従う必要がある[MUST]。メディアを受信する WebRTC エンドポイントは、その帯域幅制限を通知すべきである[SHOULD]。これらの制限は、エッジリンクの容量など、既知の帯域幅制限に基づく必要がある。

7.2 輻輳制御の相互運用性とレガシーシステム

WebRTC との相互運用を希望するすべてのエンドポイントは、RTCP を実装し、定義された RTCP レポートメカニズムを介して輻輳フィードバックを提供しなければならない[MUST]。

RTP/AVP プロファイル [RFC3551] を使用して RTCP をサポートするレガシー実装とインターワーキングする場合、輻輳フィードバックは数秒ごとに RTCP RR パケットで提供される。

このようなエンドポイントと相互作用しなければならない実装は、発生する可能性のある輻輳を制限するために、RTP サーキットブ

レーカー [RFC8083] 制約内に維持することを保証しなければならない[MUST]。

レガシーエンドポイントが RTP/AVPF をサポートする場合、これにより、頻繁なレポートのための重要なパラメータ「trr-int」のネゴシエーションが可能になり、エンドポイントが TMMBR [RFC5104] のような輻輳制御目的のためのいくつかの有用なフィードバックフォーマットをサポートする可能性がある。このようなエンドポイントと相互作用しなければならない実装は、発生する可能性のある輻輳を制限するために、RTP サーキットブレーカー [RFC8083] 制約の範囲内に留まることを保証しなければならないが[MUST]、利用可能なフィードバックの量に依存して、より良い輻輳応答を達成できることが分かるかもしれない。

独自の輻輳制御アルゴリズムを使用すると、通信セッションで異なるアルゴリズムと実装が相互作用する場合に問題が発生する可能性がある。異なる実装が適応のタイプに関して異なる選択を行った場合、たとえば、1 つの送信者ベースと 1 つの受信者ベースの場合、一方の方向が制御されていないときに一方の方向が二重に制御される状況に陥る可能性がある。この文書は、独自の輻輳制御アルゴリズムの動作を命令することはできないが、そのようなアルゴリズムを使用する実装は、この問題を認識する必要があり、効果的な輻輳制御が双方向に流れるメディアに対してネゴシエートされるようにする必要がある。IETF が将来、WebRTC トラフィックの送信者ベースと受信者ベースの両方の輻輳制御アルゴリズムを標準化する場合、相互運用性、制御、およびメディアフローの両方向が輻輳制御されていることの保証の問題も考慮する必要がある。

8. WebRTC による RTP の使用：パフォーマンスモニタリング

セクション 4.1 で説明したように、実装は送受信する RTP パケットストリームに関連する RTCP 送信者レポート (SR) と受信者レポート (RR) パケットを生成するために必要である[REQUIRED]。これらの RTCP レポートには基本的なパケット損失とジッタの統計情報が含まれているため、パフォーマンスモニタリングの目的で使用できる。

RTCP 拡張レポート (XR) フレームワークでは、多数の追加パフォーマンス・メトリックがサポートされている。[RFC3611] と [RFC6792] を参照。この文書の執筆時点では、WebRTC での使用に適した拡張メトリックは明確ではないため、RTCP XR パケットを生成する実装は、必要はない。しかし、詳細な性能モニタリングデータを使用できる実装は、適切に RTCP XR パケットを生成してもよい[MAY]。RTCP XR パケットの使用は通知されるべきである[SHOULD] ; 実装は、予期しないか理解されていない RTCP XR パケットを無視しなければならない[MUST]。

9. WebRTC による RTP の使用：将来の拡張

この文書で指定された RTP プロトコルと RTP 拡張のコアセットは、WebRTC の将来のニーズに対して不十分であることが証明される可能性がある。この場合、この文書の将来の更新は、「RTP ペイロード形式仕様の作成者のガイドライン」 [RFC2736] 、「RTP ペイロード形式の記述方法」 [RFC8088] 、「RTP Control Protocol (RTCP) 拡張のガイドライン」 [RFC5968] に従って行われなければならない。また、開発された RTP と関連プロトコルを拡張するための将来のガイドラインを考慮すべきである[SHOULD]。

将来の拡張の作成者は、拡張を推奨する際に RTP が使用される広範囲の環境を考慮するように促される。これは、一部のシナリオで適用可能な拡張が、他のシナリオで問題になる可能性があるためである。可能であれば、WebRTC フレームワークは、一般的なユーティリティである RTP 拡張を採用し、特定の WebRTC ユースケースに限定されたメカニズムを採用するのではなく、RTP を使用する他のアプリケーションへのゲートウェイの簡単な実装を可能にする。

10. シグナリングに関する考慮事項

RTP は、外部シグナリングチャネルが存在することを前提に構築されており、RTP セッションとその機能の設定に使用することができる。RTP セッションの基本設定は、次のパラメータで構成される。

RTP プロファイル : セッションで使用される RTP プロファイルの名前。RTP/AVP [RFC3551] および RTP/AVPF [RFC4585] プロファイルは、基本レベルで相互運用でき、セキュアなバリエーションである RTP/SAVP [RFC3711] および RTP/SAVPF [RFC5124] と同様である。プロファイルのセキュアバリエーションは、SRTP パケット内の認証用の追加ヘッダーフィールドとペイロードの暗号変換が存在するため、非セキュアバリエーションと直接相互運用しない。WebRTC は RTP/SAVPF プロファイルの使用を要求しており、これはシグナリングされなければならない[MUST]。インターワーキング機能は、RTP/SAVPF が使用されることを WebRTC エンドポイントに示し、「trr-int」値を 4 秒に設定することによって、これをレガシーユースケースの RTP/SAVP プロファイルに変換できる。

トランスポート情報 : RTP および RTCP の送信元と宛先の IP アドレスとポートは、各 RTP セッションに対してシグナリングされ

なければならない[MUST]。WebRTC では、これらの転送アドレスは候補に信号を送信し、指名された候補アドレスペアに到達する Interactive Connectivity Establishment (ICE) [RFC8445] によって提供される。RTP および RTCP 多重化[RFC5761]を使用して単一のポート（つまり、トランスポート層フロー）を RTP および RTCP フローに使用する場合は、これを通知する必要がある[MUST]（セクション 4.5）。

RTP ペイロードタイプ、メディアフォーマット、およびフォーマットパラメータ：メディアタイプ名 (使用される RTP ペイロードフォーマット) と RTP ペイロードタイプ番号との間のマッピングを通知しなければならない[MUST]。各メディアタイプには、コーデックおよび RTP ペイロードフォーマット(SDP の「a=fmtp:」)を設定するために通知されなければならない[MUST]いくつかのメディアタイプパラメータを含んでも良い[MAY]。この文書のセクション 4.3 は、ペイロードタイプの一意性のための要件を論じている。

RTP 拡張：必要なパラメータを含む、追加の RTP ヘッダー拡張および RTCP パケットタイプの使用は、通知されなければならない[MUST]。このシグナリングにより、WebRTC エンドポイントの動作 (特に送信時) が予測可能で一貫していることが保証される。ゲートウェイを介して WebRTC セッションに接続される可能性のある非 WebRTC システムとの堅牢性および互換性を確保するために、不明な RTCP パケットおよび RTP ヘッダー拡張(セクション 4.1)を無視する実装が必要である[REQUIRED]。

RTCP 帯域幅：エンドポイントとの RTCP 帯域幅値の交換をサポートする必要がある。これは、SDP または意味的に同等なものを使用する場合、「RTP Control Protocol (RTCP) 帯域幅のセッション記述プロトコル (SDP) 帯域幅修飾子」 [RFC3556] で説明されているように実行する必要がある[SHALL]。これにより、エンドポイントは RTCP 帯域幅の共通のビューを持つことも保証される。相互運用性の問題を引き起こす RTCP パケットのタイミングとタイムアウト間隔の違いを防ぐには、異なるエンドポイント間の RTCP 帯域幅の共通のビューが重要である。

これらのパラメータは、オファー/アンサー交換内で伝達される SDP メッセージで表現されることが多い。RTP は SDP またはオファー/アンサーモデルに依存しないが、必要なすべてのパラメータが合意され、RTP 実装に提供される必要がある。WebRTC では、これらのパラメータの設定方法はシグナリングモデルと API によって異なるが、API で設定するか、ピア間で明示的にシグナリングする必要がある。

11. WebRTC API に関する考慮事項

WebRTC API [W3C.WebRTC] と Media Capture and Streams API [W3C.WDmediacapture-streams] は、0 個以上の MediaStreamTracks で構成される MediaStream の概念を定義および使用する。MediaStreamTrack は、マイクやカメラなどのあらゆる種類のメディアソースからの個々のメディアストリームであるが、オーディオミックスやビデオコンポジションなどの概念的なソースも可能である。MediaStream 内の MediaStreamTracks は、再生中に同期する必要がある場合がある。

RTP における MediaStreamTrack の実現は、RTCPeerConnection のコンテキストにおいて、RTCPeerConnection の一部である RTP セッション内で送信される SSRC によって識別されるソースパケットストリームで構成される。MediaStreamTrack を使用すると、同じ RTP セッションでパケットストリーム (SSRC) を追加することもできる。これらは、そのようなメディアエンコーダが使用される場合、MediaStreamTrack に関連するソースストリームのスケーラブルなエンコーディングからの依存パケットストリームである可能性がある。冗長パケットストリームにすることもできる。これらは、ソースパケットストリームに Forward Error Correction (セクション 6.2)または RTP 再送信(セクション 6.1)を適用するときに作成される。

同じメディアソースが複数の MediaStreamTracks を供給できることに注意する必要がある。異なるセットの制約または他のパラメータを MediaStreamTrack に適用することができるので、RTCPeerConnection に追加された各 MediaStreamTrack インスタンスは、関連するパケットストリームの独自のセットとなる異なる SSRC を持つ独立したソースパケットストリームを生成する必要がある[SHALL]。ソースストリームとエンコード設定が、同じメディアソースを共有する異なる MediaStreamTracks 間で同一である場合、適用される制約とパラメータに依存する。エンコーディングパラメータと制約が同じである場合、実装では、異なる RTP パケットストリームを作成するためにエンコードされたストリームを 1 つだけ使用することを選択できる。このような最適化では、MediaStreamTracks の 1 つに対する制約がいつでも変更される可能性があることを考慮する必要があることに注意する。つまり、エンコーディング設定が同一でなくなり、2 つの異なるエンコーダインスタンスが必要になる可能性がある。

同じ MediaStreamTrack を複数の MediaStreams に含めることもできる。したがって、複数の MediaStreams セットで同じ同期ベースを暗黙的に使用する必要がある。これがすべてのケースで機能し、進行中のパケットストリームの配信中に同期ベースと CNAME を変更してエンドポイントにメディアを中断させないようにするには、同じエンドポイントから発信されるすべての MediaStreamTrack とその関連 SSRC を、1 つの RTCPeerConnection 内で同じ CNAME を使用して送信する必要がある。これがセクション 4.9 で単一の CNAME を使

用する動機となっている。

同じエンドポイントから発信されるすべての SSRC に同じ CNAME を使用するという要件は、複数のエンドポイントからのトラフィックを転送して 1 つの CNAME だけを使用するミドルボックスを必要としない。

通常、セクション 4.9 で指定されているように、異なる RTCPeerConnection インスタンスに対して異なる CNAME を使用する必要がある。同じ CNAME で 2 つの通信セッションを持つことで、異なるサービス間でのユーザまたはデバイスの追跡が可能になる (セクション 4.4.1 (詳細は [RFC8826]) 。Web アプリケーションは、異なる RTCPeerConnections (同一起点コンテキスト内) で使用される CNAME が同じであることを要求できる。これにより、異なる RTCPeerConnections 間でエンドポイントの RTP パケットストリームを同期できる。

注意 : 一致する CNAME の作成は単一のオリジン内の既存のトラッキングに依存するため、これはトラッキングの問題にはならない。

上記は、現在、1 つの RTCPeerConnection 上で MediaStreamTrack を受信し、それを任意の RTCPeerConnection 上の発信 1 つとして追加する WebRTC エンドポイントに強制的にストリームの再同期を実行させる。送信者は CNAME を使用するものに変更する必要があるため、同期のタイムベースとしてローカルシステムクロックを使用する必要がある。したがって、入力ストリームのタイムベースと送出システムの相対的な関係を定義する必要がある。この関係は、クロック・ドリフトと同期化の調整の監視も必要である。送信エンティティは、送信ストリームの輻輳制御も行う。パケット損失の場合、着信データの損失も処理する必要がある。これにより、発信元パケットストリームで問題または中断を引き起こす可能性が最も低い方法は、修復を含む完全なデコードのモデルであり、その後、発信パケットストリームにメディアを再度エンコードすることが観察される。この方法の最適化は明らかに可能であり、実装特有である。

WebRTC エンドポイントは複数の MediaStreamTrack の受信をサポートしなければならず[MUST]、異なる MediaStreamTrack (および関連するパケットストリームのセット) のそれぞれが異なる CNAME を使用する。ただし、異なる CNAME で受信された MediaStreamTrack には同期が定義されていない。

注意 : 複数の CNAME の受信をサポートする目的は、エンドポイントがストリームをリレー/転送するときに、より効率的なストリーム処理を可能にする将来の変更との前方互換性を可能にすることである。また、エンドポイントが特定のタイプのマルチストリームミドルボックスまたは WebRTC 以外のエンドポイントと相互運用できることも保証される。

「JavaScript セッション確立プロトコル (JSEP)」 [RFC8829] は、WebRTC MediaStreams、MediaStreamTracks、SSRC 間のバインディングが「セッション記述プロトコルでの WebRTC MediaStream の識別」[RFC8830] で規定されているように行われることを規定している。MediaStream Identification (MSID) ドキュメント [RFC8830] のセクション 4.1 でも、不明な SSRC を持つソースパケットストリームを MediaStreamTracks および MediaStreams にマッピングする方法を定義している。これは、レガシー相互運用性のいくつかのケースを処理するのに関連する。一般に、着信パケットの RTP ペイロードタイプによって、パケットストリームが送信元ストリームであるか、冗長であるか、依存するパケットストリームであるかが明らかになる。正しい送信元パケットストリームへの関連付けは、パケットストリームに使用されているペイロード形式によって異なる。

最後に、この仕様では WebRTC API に CSRC リスト(セクション 4.1)およびミキサーからクライアントへのオーディオレベル (セクション 5.2.3) (サポートされている場合) を決定する方法を実現するための要件を課している。このための基本的な必要条件はセクション 12.2.1 でさらに議論される。

12. RTP 実装に関する考慮事項

以下の説明では、このメモで説明されている RTP 機能の実装に関するガイダンスを提供する。焦点は WebRTC エンドポイントの実装の観点にあり、ミドルボックスの動作についても言及されているが、それはこの文書の焦点ではない。

12.1 RTP セッションの構成と使用

WebRTC エンドポイントは、1 つ以上の RTP セッションに同時に参加する。各 RTP セッションは、複数のメディアソースを伝達し、複数のエンドポイントからのメディアデータを含めることができる。以下では、WebRTC エンドポイントが RTP セッションを構成および使用できるいくつかの方法の概要を説明する。

12.1.1 RTP セッション内での複数のメディアソースの使用

RTP はグループ通信プロトコルであり、すべての RTP セッションに複数の RTP パケットストリームが含まれる可能性がある。これが望ましい理由はいくつかある。

- 複数のメディアタイプ：
  WebRTC 以外では、メディアソースのタイプごとに 1 つの RTP セッションを使用するのが一般的である（たとえば、オーディオソース用に 1 つの RTP セッション、ビデオソース用に 1 つ、それぞれが異なるトランスポート層フローを介して送信される）。しかしながら、使用される UDP ポートの数を減らすために、WebRTC のデフォルトでは、セクション 4.4 で説明されているように、すべてのタイプのメディアを単一の RTP セッションで送信する。RTP および RTCP 多重化（セクション 4.5）を使用して必要な UDP ポートの数をさらに減らすことができる。この RTP セッションは、1 つの双方向トランスポート層フローのみを使用するが、それぞれが異なるタイプのメディアを含む複数の RTP パケットストリームを含む。一般的な例は、2 つの RTP パケットストリーム（1 つのビデオと 1 つのオーディオ）を単一の RTP セッションに送信するカメラとマイクを備えたエンドポイントである。

- 複数のキャプチャデバイス：
  WebRTC エンドポイントには、複数のカメラ、マイク、またはその他のメディアキャプチャデバイスが含まれる場合があるため、同じメディアタイプの複数の RTP パケットストリームを生成したい場合がある。または、単一のキャプチャデバイスから複数の異なる形式または品質設定でメディアを一度に送信することもできる。どちらの場合も、単一のエンドポイントが同じメディアタイプの複数の RTP パケットストリームを同時に単一の RTP セッションに送信する可能性がある。

- 関連する修復データ：
  エンドポイントは、別のストリームに何らかの方法で関連付けられた RTP パケットストリームを送信できる。たとえば、FEC または別のストリームに関連する再送信データを含む RTP パケットストリームを送信する場合がある。RTP ペイロード形式の中には、この種の関連付けられた修復データを送信元パケットストリームの一部として送信するものと、別のパケットストリームとして送信するものがある。

- 階層化または複数記述のコーディング：
  単一の RTP セッション内で、エンドポイントは、階層化されたメディアコーデック（たとえば、H.264 スケーラブルビデオコーディング（SVC））、またはそれぞれが個別の RTP SSRC を持つ複数の RTP パケットストリームを生成する複数記述コーデックを使用できる。

- RTP ミキサー、トランスレータ、およびその他のミドルボックス：
  WebRTC コンテキスト内の RTP セッションは、エンドポイントと他のピアデバイスとの間のポイントツーポイントの関連付けであり、これらのデバイスは共通の SSRC スペースを共有する。ピアデバイスは別の WebRTC エンドポイントである場合もあれば、RTP ミキサー、トランスレータ、またはその他の形式のメディア処理ミドルボックスである場合もある。後者の場合、ミドルボックスは複数の参加者から混合された RTP ストリームまたはリレーされた RTP ストリームを送信する可能性があり、WebRTC エンドポイントはこれをレンダリングする必要がある。したがって、WebRTC エンドポイントが単一の RTP セッションのメンバーにすぎない場合でも、ピアデバイスはその RTP セッションを拡張して他のエンドポイントを組み込むことができる。WebRTC はグループ通信環境であり、エンドポイントは、単一の RTP セッションであっても、一度に複数の RTP パケットストリームを受信、デコード、および再生できる必要がある。

12.1.2 複数の RTP セッションの使用

単一の RTP セッション内で複数の RTP パケットストリームを送受信することに加えて、WebRTC エンドポイントは複数の RTP セッションに参加する場合がある。 WebRTC エンドポイントがこれを選択する理由はいくつかある。

- レガシーデバイスと相互運用するには：
  WebRTC 以外の世界では、異なるタイプのメディアを別々の RTP セッションで送信するのが一般的である。たとえば、音声用に 1 つの RTP セッションを使用し、ビデオ用に別のトランスポート層のフローで別の RTP セッションを使用する。すべての WebRTC エンドポイントは、さまざまなタイプのメディアをさまざまな RTP セッションで送信して、そのようなレガシーデバイスと相互運用できるようにするオプションをサポートする必要がある。これについてはセクション 4.4 で詳しく説明している。

- サービス品質を向上させるためには：
  いくつかのネットワークベースのサービス品質メカニズムは、トランスポート層フローの粒度で動作する。これらのメカニズムを使用して一部の RTP パケットストリームに差別化されたサービス品質を提供する必要がある場合、これらの RTP パケットストリームは、別のトランスポート層フローを使用し、適切なサービス品質マーキングを使用して、別の RTP セッションで送信する必要がある。これについては、セクション 12.1.3 で詳しく説明する。

・ 異なる目的でメディアを分離するには：
エンドポイントは、ピアデバイスがそれらを簡単に区別できるようにするために、異なる RTP セッションで異なる目的を持つ RTP パケットストリームを送信したい場合がある。たとえば、一部の集中型マルチパーティ会議システムは、アクティブスピーカーを高解像度で表示するが、他の参加者は低解像度の「サムネイル」で表示する。このようなシステムは、RTP ミドルボックスの操作を簡素化するために、別々の RTP セッションを使用してビデオの高解像度バージョンと低解像度バージョンを同時に送信するようにエンドポイントを構成する場合がある。 WebRTC コンテキストでは、これは現在、1 つ（または複数）の RTCPeerConnection に同じメディアソースを持つ複数の WebRTC MediaStreamTracks を確立することで可能である。次に、各 MediaStreamTrack は、特定のメディア品質、つまりメディアビットレートを提供するように構成され、その RTCPeerConnection のコンテキストで具体的に合意されたコーデックパラメータを使用して、独立してエンコードされたバージョンを生成する。 RTP ミドルボックスは、SSRC、RTP ペイロードタイプ、または RTP ペイロード、RTP ヘッダー拡張、または RTCP パケットに含まれるその他の情報を検査することにより、低解像度ストリームと高解像度ストリームに対応するパケットを区別できる。ただし、RTP パケットストリームが別々のトランスポート層フロー上の別々の RTP セッションに到着する場合は、RTP パケットストリームを簡単に区別できる。

・ 複数のピアに直接接続するには：
マルチパーティ会議では、RTP ミドルボックスを使用する必要はない。それよりも、図 1 に示すように、複数の個別の RTP セッションで構成されるマルチユニキャストメッシュを作成できる。各参加者は、個別の RTP セッションを介して（つまり、独立した RTCPeerConnection オブジェクトを使用して）RTP トラフィックを他のすべての参加者に送信する。このトポロジには、メディアデータへのアクセスと操作を信頼できる RTP ミドルボックスノードを必要としないという利点がある。欠点は、送信者自体を超えて同じセッションの一部である参加者ごとに RTP パケットストリームのコピーを 1 つ要求することにより、各送信者で使用される帯域幅が増えることである。

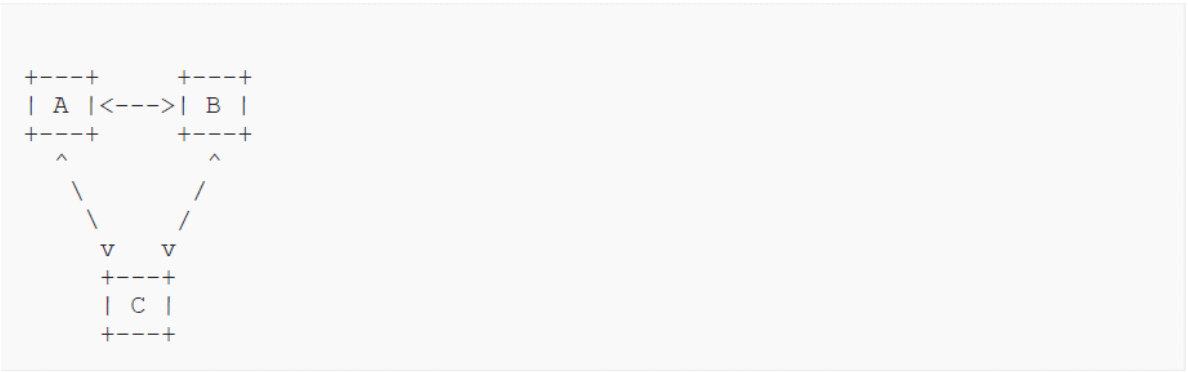

図 1：複数の RTP セッションを使用したマルチユニキャスト

マルチユニキャストトポロジは、複数のピアツーピアトランスポート層接続にまたがる単一の RTP セッションとして実装することも、またはピアの各ペア間に 1 つずつある複数のペアワイズ RTP セッションとして実装することもできる。 RTP セッションと RTCPeerConnection オブジェクト間の関係の一貫したマッピングを維持するために、これをいくつかの個別の RTP セッションとして実装することをお勧めする[RECOMMENDED]。唯一の欠点は、エンドポイント A が B と C の間で発生する伝送の品質を認識しないことである。これは、B と C の間の RTP セッションの RTCP レポートが表示されないためであるが、3 人の参加者全員が単一の RTP セッションの一部である場合は表示される。 Mbone ツール（1990 年代後半の実験的な RTP ベースのマルチキャスト会議ツール）の経験から、サードパーティの RTCP 受信品質レポートを、非対称ネットワークの問題を理解するのに役立つ方法でユーザに提示できることが示されている。個別の RTP セッションを使用するアプローチでは、これを防ぐことができる。ただし、個別の RTP セッションを使用する利点は、異なるピア間で異なるメディアビットレートと RTP セッション構成を使用できるため、A から C への転送に制限がある場合に、B が C と同じ品質低下に耐えることを強制されないことである。これらの利点は、デバッグ能力の制限を上回ると考えられる。

・ 複数のピアに間接的に接続：
マルチパーティ会議の一般的なシナリオは、RTP ミキサー、トランスレータ、またはその他のタイプの RTP ミドルボックスを使用して、複数のピアへの間接接続を作成することである。図 2 は、4 人の集中型会議で使用される可能性のある単純なトポロジの概要を示している。ミドルボックスは、受信した RTP パケットストリームの一部のみを特定の受信者に送信するか、一連の寄与ストリームから結合された RTP パケットストリームを提供することにより、特定の観点から RTP パケットストリームの送信を最適化するように機能する。

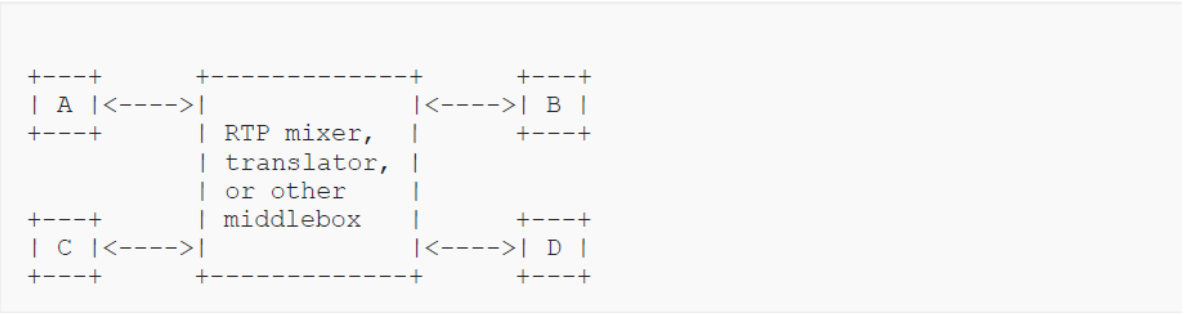

図 2：ユニキャストパスのみの RTP ミキサー

ミドルボックスの実装にはさまざまな方法がある。標準の RTP ミキサーまたはトランスレータとして実装されている場合、単一の RTP セッションはミドルボックス全体に拡張され、1 つのマルチパーティセッション内のすべてのエンドポイントを包含する。他のタイプのミドルボックスは、各エンドポイントとミドルボックスの間で個別の RTP セッションを使用する場合がある。一般的な側面は、これらの RTP ミドルボックスが、WebRTC エンドポイントによって提供されるメディアエンコーディングを制御するために多くのツールを使用できることである。これには、エンコーディングチェーンの切断を要求したり、エンコーダにいわゆるイントラフレームを生成させたりするなどの機能が含まれる。もう 1 つの一般的な側面は、混合出力によりよく一致するようにストリームのビットレートを制限することである。他の側面は、最適なフレームレート、画像解像度、およびフレームレートと空間品質の間のトレードオフを制御することである。ミドルボックスは、アプリケーションに適切なメディア最適化を提供しながら、輻輳制御を正しく実行し、ソースを識別し、同期を管理する責任がある。ミドルボックスは、RTP ヘッダーまたは 1 つのエンドポイントから受信したメディア自体（または両方）を操作してからエンドポイントに送信するため、セキュリティに関しても信頼できるノードである必要がある。したがって、RTP パケットストリームを送信する前に、復号化してから再暗号化できる必要がある。

ミキサーは、RTP パケットストリームに関する RTCP レポートをミキサー間で転送しないことが期待されている。これは、異なるエンドポイントに提供される RTP パケットストリームの違いによるものである。元のメディアソースには、さまざまな受信者に送信される前のミキサーの操作に関する情報がない。このシナリオでは、エンドポイントのフィードバックまたはリクエストがミキサーに送られる。ミキサーがそれ自体でこれに対処できない場合、受信者の要求を満たすために元のメディアソースに移動することを余儀なくされる。これは、必ずしも RTP および RTCP トラフィックに明示的に表示されるとは限らないが、相互作用とそれらを完了する時間は、そのような依存関係を示す。

マルチパーティシナリオでソース認証を提供することは課題である。ミキサーベースのトポロジでは、エンドポイントのソース認証は、最初に、暗号化検証によってメディアがミキサーからのものであることを検証し、次に、ミキサーを信頼してエンドポイントに向かうソースを正しく識別することに基づいている。複数のエンドポイントがエンドポイントに直接表示される RTP セッションでは、すべてのエンドポイントが互いのマスターキーに関する知識を持っているため、セッション内の別のエンドポイントから発信されたと主張するパケットを注入できる。リレーを実行するノードは、以前に他のエンドポイントから送信された SSRC フィールドを持つパケットの転送を防止することにより、暗号化されていない軽減を実行できる。ソースの暗号化検証のために、SRTP は、基本 WebRTC 標準の一部ではない追加のセキュリティメカニズム（たとえば、SRTP [RFC4383]の Timed Efficient Stream Loss-Tolerant Authentication（TESLA））を必要とする。

- 複数のピア間でメディアを転送するには：
  RTP パケットストリームを受信するエンドポイントが、その RTP パケットストリームをサードパーティに転送できることが望ましい場合がある。これをサポートする上でのセキュリティとプライバシーへの明らかな影響がいくつかあるが、潜在的な用途もある。これは、受信およびデコードされたメディアを取得し、それを新しいストリームとして再エンコードおよび送信されるメディアソースとして使用することにより、W3C API でサポートされる。

  RTP 層では、メディア転送は連続した RTP 受信者および RTP 送信者として機能する。受信側は RTP セッションを終了してメディアをデコードし、送信側は完全に別個の RTP セッションを使用してメディアを再エンコードして送信する。元の送信者にはメディアの受信者が 1 人しか表示されず、RTP 層の情報に基づいて転送が行われていることを知ることができない。転送されたメディアの送信に使用される RTP セッションは転送しているノードがメディアを受信した RTP セッションに接続されていないためである。

  転送を実行しているエンドポイントは、転送に適した RTP パケットストリームを生成する責任がある。転送されたメディアを送信するために使用される発信 RTP セッションは、メディアが受信された RTP セッションとは完全に分離されている。

これには、輻輳制御の目的でメディアトランスコーディングが必要になり、発信 RTP セッションに適したビットレートが生成され、メディア品質が低下し、転送エンドポイントがリソースをトランスコーディングに費やすようになる。メディアトランスコーディングにより、2 つの異なる RTP セッション(発信 RTP セッションと受信 RTP セッション)が分離され、ほとんどすべての依存関係が削除され、転送エンドポイントがメディアトランスコーディング操作を最適化できるようになる。コストは、転送ノードでの計算の複雑さが大幅に増加する。転送されたストリームの受信者は、転送デバイスをストリームの送信者と見なし、転送デバイスによって生成されたまったく新しい RTP パケットストリームではなく、転送されたストリームを受信していることを RTP 層から知ることができない。

12.1.3 RTP ストリームの差別化された処理

RTP パケットストリームの差別化された処理のユースケースがある。このような差別化は、システムのいくつかの場所で発生する可能性がある。第一に、メディアを送信するエンドポイント内の優先順位付けである。これは、送信される RTP パケットストリームと、輻輳制御によって決定される現在利用可能な集約からのビットレートの割り当ての両方を制御する。

WebRTC API [W3C.WebRTC]により、アプリケーションはさまざまな MediaStreamTracks の相対的な優先順位を示すことができると期待されている。これらの優先順位は、特に輻輳制御のために RTP パケットストリーム間で使用可能な帯域幅を分割する方法を決定する場合に、ローカル RTP 処理に影響を与えるために使用できる。 RTP 再送信や FEC の冗長ストリームなど、メイン RTP パケットストリームに関連付けられている RTP パケットストリームについても、相対的な優先順位の変更を考慮する必要がある。このような冗長 RTP パケットストリームの重要性は、そのコーデックがパケット損失に対してどれだけ堅牢であるかに関して、使用されるメディアタイプとコーデックに依存する。ただし、デフォルトのポリシーでは、冗長 RTP パケットストリームにソース RTP パケットストリームと同じ優先度を使用する場合がある。

次に、ネットワークは、RTP パケットストリームを含むトランスポート層フローとサブフローに優先順位を付けることができる。通常、差分処理には 2 つのステップが含まれる。1 つは IP パケットが異なる処理が必要なクラスに属しているかどうかを識別することで、2 つ目はパケットに優先順位を付けるための実際のメカニズムで構成される。 IP パケットを分類するための 3 つの一般的な方法は次のとおりである。

DiffServ：エンドポイントは、パケットが特定のクラスに属していることをネットワークに示すために、DiffServ コードポイントでパケットをマークする。

フローベース：特定の処理を行う必要があるパケットは、IP アドレスとポートアドレスの組み合わせを使用して識別される。

ディープパケットインスペクション：ネットワーク分類子（DPI）はパケットを検査し、パケットが優先される特定のアプリケーションとタイプを表しているかどうかを判断しようとする。

フローベースの差別化は、トランスポート層フロー内のすべてのパケットに同じ処理を提供する。つまり、相対的な優先順位付けは不可能である。さらに、リソースが限られている場合、WebRTC セッションで使用されるすべての RTP パケットストリームに対してベストエフォートと比較して、差別的な処理を提供できない可能性がある。フローベースの差別化の使用は、WebRTC システムとネットワークの間で調整する必要がある。 WebRTC エンドポイントは、フローベースの差別化を使用して、RTP パケットストリームをさまざまな UDP フローに分離し、フローベースの差別化をよりきめ細かく使用できるようにする可能性があることを認識する必要がある。使用されたフロー、それらの 5 タプル、および優先順位付けは、ネットワークに通信して、優先順位付けを有効にするためにフローを正しく識別できるようにする必要がある。これに対する特定のプロトコルサポートは指定されていない。

DiffServ は、エンドポイントまたは分類子のいずれかが適切な Differentiated Services Code Point（DSCP）でパケットをマークできることを前提としているため、パケットはそのマークに従って処理される。エンドポイントがトラフィックをマークする場合、WebRTC コンテキストで 2 つの要件が発生する。1）WebRTC エンドポイントは、使用する DSCP を認識し、RTP パケットストリームのセットでそれらを使用できることを認識している必要がある。2）パケットを送信するときに、情報をオペレーティングシステムに伝達する必要がある。このプロセスの詳細は、このメモの範囲外であり、 「WebRTC QoS の Differentiated Services Code Point（DSCP）パケットマーキング」[RFC8837]でさらに説明されている。

SRTP メディアの暗号化にもかかわらず、ディープパケットインスペクターは RTP ストリームをかなり分類することができる。その理由は、SRTP が RTP ヘッダーの最初の 12 バイトを暗号化しないままにするためである。これにより、SSRC を使用した RTP ストリームの識別が容易になり、たとえばストリームのメディアタイプを判別するために相関させることができる有用な情報が分類子に提供される。パケットサイズ、受信時間、パケット間隔、RTP タイムスタンプの増分、およびシーケンス番号を使用して、かなり信頼性の高い分類が実現される。

パケットベースのマーキングスキームの場合、RTP ペイロードの相対的な優先度に基づいて、個々の RTP パケットを異なる方法でマーキングできる場合がある。たとえば、I、P、および B の画像を含むビデオコーデックは、B フレームのみを伝送するペイロードを優先することができる。これは、これらのペイロードの損失が少ないためである。ただし、QoS メカニズムと適用されるマーキングによっては、パケットドロップの確率が異なるだけでなく、パケットの並べ替えも発生する可能性がある。詳細については、[RFC8837]および[RFC7657]を参照。デフォルトのポリシーとして、RTP パケットストリームに関連するすべての RTP パケットには、同じ優先順位を付ける必要がある。パケットごとの優先順位付けはこのメモの範囲外であるが、将来的に他の場所で指定される可能性がある。

特定の RTP パケットストリームに関連付けられた RTCP パケットをどのようにマークする必要があるかを考慮することも重要である。送信者レポート（SR）を使用する RTCP 複合パケットは、RTP パケットストリーム自体と同じ優先度でマークする必要があるため、RTCP ベースのラウンドトリップ時間（RTT）の測定は、RTP パケットストリームが経験するのと同じトランスポート層フロー優先度を使用して行われる。RR パケットを含む RTCP 複合パケットは、報告された RTP パケットストリームの大部分で使用される優先度で送信する必要がある。タイムクリティカルなフィードバックパケットを含む RTCP パケットは、より高い優先度を使用して、そのようなフィードバックの適時性と配信の可能性を向上させることができる。

12.2 メディアソース、RTP ストリーム、および参加者の識別

12.2.1 メディアソース識別

各 RTP パケットストリームは、一意の同期ソース (SSRC) ID によって識別される。SSRC ID は、 RTP パケットストリームを構成する各 RTP パケットで伝送され、対応する RTCP レポートでそのストリームを識別するためにも使用される。SSRC は、セクション 4.8 で説明したように選択される。WebRTC エンドポイントの単一のトランスポート層フローで受信された RTP および RTCP パケットを逆多重化する最初の段階は、SSRC 値に基づいて RTP パケットストリームを分離することである。それが完了すると、追加の逆多重化手順により、メディアをレンダリングする方法と場所を決定できる。

RTP を使用すると、ミキサーまたはその他の RTP 層のミドルボックスで、複数のメディアソースからのエンコードされたストリームを組み合わせて、新しいメディアソース（ミキサー）からの新しいエンコードされたストリームを形成できる。その新しい RTP パケットストリーム内の RTP パケットには、コントリビューティングソース（CSRC）リストを含めることができる。どの元の SSRC が結合されたソースストリームに貢献したかを示す。セクション 4.1 で説明したように、実装は、CSRC リストを含む RTP データパケットと、CSRC リストに存在するソースに関連する RTCP パケットの受信をサポートする必要がある。CSRC リストは、実行されているミキシング操作に応じて、パケットごとに変更される可能性がある。ユーザインタフェースがセッションでアクティブな参加者を示している場合、特定の RTP パケットにどのメディアソースが寄与しているかを知ることが重要になる可能性がある。アプリケーションがセッション参加の変更を追跡できるようにするには、パケットに含まれる CSRC リストの変更を何らかの API を使用して WebRTC アプリケーションに公開する必要がある。SSRC / CSRC 名前空間が WebRTC アプリケーションに公開されないように、CSRC 値をこの API と交差するときに WebRTC MediaStream ID にマップして戻すことが望ましい。

12.2.2 SSRC 衝突検出

RTP 標準では、SSRC 衝突の検出と処理をサポートする RTP 実装が必要である。つまり、2 つの異なるエンドポイントが同じ SSRC 値を使用する場合に競合を解決できる（[RFC3550]のセクション 8.2 を参照）。この要件は、WebRTC エンドポイントにも適用される。 SSRC の衝突が発生する可能性のあるシナリオはいくつかある。

- 各 SSRC が 2 つのエンドポイントのいずれかに関連付けられ、メインメディアを伝送する SSRC ID がシグナリングチャネルでアナウンスされるポイントツーポイントセッションでは、使用されている SSRC に関する情報により、衝突が発生する可能性が低くなる。 SDP が使用されている場合、この情報は、ソース固有の SDP 属性 [RFC5576] によって提供される。それでも、ピアにシグナリングしてシグナリングメッセージで確認応答を受信する前に、両方のエンドポイントが新しい SSRC ID の使用を開始すると、衝突が発生する可能性がある。 「セッション記述プロトコル（SDP）のソース固有のメディア属性」[RFC5576]には、エンドポイントが SSRC 衝突をどのように解決したかを通知するメカニズムが含まれている。

- シグナリングされていない SSRC 値は、RTP セッションにも表示される可能性がある。一部の RTP 機能は、機能を提供するために追加の SSRC を使用するため、これは見た目よりも可能性が高くなる。たとえば、再送信データは、ソース RTP パケットストリーム[RFC4588]の SSRC とは別に、独自の SSRC を必要とする別の RTP パケットストリームを使用して送信される場合がある。このような場合、エンドポイントは、RTP レベルと RTCPeerConnection レベルの両方で正しく機能するために、シグナリングチャネルを介して厳密にアナウンスする必要のない新しい SSRC を作成できる。

- マルチパーティ会議の複数のエンドポイントは、新しいソースを作成し、それらを RTP ミドルボックスに向けて通知できる。SSRC / CSRC が RTP ミドルボックスからの異なるエンドポイント間で伝播される場合、衝突が発生する可能性がある。

・ RTP ミドルボックスは、エンドポイントの RTCPeerConnection を同じエンドポイントからの別の RTCPeerConnection に接続して、エンドポイントが独自のトラフィックを受信するループを形成する可能性がある。明らかにバグと見なされるが、エンドポイントがケースを認識して処理できることが重要である。このケースは、受信したストリームが別のストリームであるが、このクライアントの入力が含まれているメディアミキサーなどが関係している場合に、さらに問題になる。

これらの SSRC / CSRC の衝突は、同じ RTP セッションが RTP ミドルボックスによって複数の RTCPeerConnections に拡張されている場合にのみ RTP レベルで処理できる。複数の RTCPeerConnections が相互接続されているより一般的なケースを解決するには、複数の相互接続された RTCPeerConnection 間で伝播される MediaStreamTrack の一部である 1 つまたは複数のメディアソースの識別をこれらの相互接続間で保持する必要がある。

12.2.3 メディア同期コンテキスト

エンドポイントが複数のメディアソースからメディアを送信する場合、これらのメディアソースを同期するかどうか（およびどのメディアソースを同期するか）を検討する必要がある。 RTP / RTCP では、同期は、RTP パケットストリームのセットが、同じ RTCP CNAME ID を使用して、同じ同期コンテキストおよび論理エンドポイントからのものとして示されるようにすることで提供される。

次の規定は、すべてのメディアソースの内部クロック (つまり、RTP タイムスタンプを駆動するもの) を NTP フォーマットでエンコードされた RTCP 送信者レポートで提供されるシステムクロックに関連付けることができるということである。すべての RTP タイムスタンプをすべてのすべてのソースの共通システムクロックに相関させることにより、複数の RTP セッション間でも、さまざまな RTP パケットストリームのタイミング関係を受信者で導出でき、必要に応じてストリームを同期できる。要件は、メディア送信者が相関情報を提供することである。情報が使用されているかどうかは受信者次第である。

13. セキュリティに関する考慮事項

WebRTC の全体的なセキュリティアーキテクチャは[RFC8827]で説明されており、WebRTC フレームワークのセキュリティに関する考慮事項は[RFC8826]で説明されている。これらの考慮事項は、このメモにも当てはまる。

RTP 仕様、RTP/SAVPF プロファイル、およびこのメモで説明されている完全なプロトコルスイートを形成するさまざまな RTP / RTCP 拡張機能と RTP ペイロード形式のセキュリティに関する考慮事項が適用される。これらのさまざまなプロトコル拡張の組み合わせから生じる新しいセキュリティ上の考慮事項はないと考えられている。

「リアルタイムトランスポートコントロールプロトコル（RTCP）ベースのフィードバック（RTP / SAVPF）用の拡張セキュア RTP プロファイル」 [RFC5124]] は、機密性、整合性、および部分的なソース認証を提供することによって、基本的な問題の処理を提供する。[RFC8827]のセクション 5.5 で定義されているように、この保護された RTP プロファイルと DTLS-SRTP キーイング[RFC5764]を組み合わせることにより、実装と使用が必須のメディアセキュリティソリューションが作成される。

RTCP パケットは、関連する RTP セッション間で同期する必要がある RTP パケットストリームを関連付けるために使用される Canonical Name (CNAME) ID を伝達する。CNAME 値の不適切な選択はプライバシーの問題になる可能性がある。このメモのセクション 4.9 では、[RFC7022]で指定されているように、短期間の永続的な RTCP CNAME の生成が義務付けられており、このリスクを軽減する追跡不可能な CNAME 値が生成される。

RTCP レポート間隔が不適切な値に設定されている場合、潜在的なサービス拒否攻撃が存在する。これは、SDP 「b=RR:」行または「b=RS:」行[RFC3556]または同様のメカニズムを使用して、RTCP 帯域幅の割合を過度に大きな値または小さな値に設定するか、過度に大きいまたは小さい値を選択することによって実行できる。

RTP / AVPF 最小レシーバーレポート間隔の値（SDP を使用している場合、これは「a=rtcp-fb:... trr-int」パラメータである）[RFC4585]。リスクは次のとおりである。

1. RTCP 帯域幅は、定期的なレポート間隔を非常に大きくして、効果的な輻輳制御を維持できず、メディアトラフィックによって引き起こされる輻輳によるサービス拒否につながる可能性があるように構成できる。
2. RTCP 間隔を非常に小さい値に設定すると、エンドポイントが高速 RTCP トラフィックを生成し、RTCP トラフィックが輻輳制御されていないためにサービス拒否につながる可能性がある。および
3. RTCP パラメータは、エンドポイントごとに異なるように設定できる。一部のエンドポイントでは大きなレポート間隔を使用し、一部のエンドポイントでは小さなレポート間隔を使用する。これにより、レポート間隔に基づくタイムアウト期間の不一致に

よる参加者のタイムアウトが早すぎるため、サービス拒否につながる。[RFC8108]のセクション 6.1 で説明されているように、エンドポイントが RTP / AVPF 最小レシーバーレポート間隔（trr-int）[RFC4585]に小さいがゼロ以外の値を使用する場合、これは特に懸念される。

参加者のタイムアウトを計算するときに、固定された（短縮されていない）最小間隔を使用することで、参加者の時期尚早なタイムアウトを回避できる（このメモのセクション 4.1 および[RFC8108]のセクション 7.1.2 を参照）[SHULD]。他の懸念に対処するために、エンドポイントは、[RFC3550]で指定されたデフォルトの 5 秒間隔よりも大幅に長くなるように RTCP レポート間隔を構成するパラメータを無視する必要がある（メディアデータレートが非常に低く、長いレポート間隔がメディアデータレートの約5%に相当する場合を除きく）、または RTCP 帯域幅がメディア帯域幅を超えるように RTCP レポート間隔を十分に小さく設定する。

[RFC6562]のガイドラインは、Opus などの可変ビットレート（VBR）オーディオコーデックを使用する場合に適用される（必須のオーディオコーデックの説明については、セクション 4.3 を参照）。[RFC6562]のガイドラインも適用されるが、クライアントからミキサーへのオーディオレベルヘッダー拡張機能（セクション 5.2.2）またはミキサーからクライアントへのオーディオレベルヘッダー拡張機能（セクション 5.2.3）を使用する場合は、それほど重要ではない。ヘッダー拡張の暗号化の使用をお勧めする [RECOMMENDED]。既知の理由がない限り、音声アクティビティベースのソース選択やサードパーティのモニタリングを実行する RTP ミドルボックスのように、情報から大きなメリットが得られる。これは、API またはシグナリングを使用して表現されている。情報漏えいが音声レベルの表示から重大であることを示すさらなる証拠が作成された場合、その時点で暗号化の使用を義務付ける必要がある。

RTP ミドルボックスを使用するマルチパーティ通信シナリオでは、セッションのセキュリティを維持するために、これらのミドルボックスに多くの信頼が置かれる。ミドルボックスは、機密性と整合性を維持し、ソース認証を実行する必要がある。セクション 12.1.1 で説明したように、ミドルボックスは、会議に参加しているエンドポイントが別のエンドポイントになりすますことを防ぐチェックを実行できる。マルチパーティトポロジに関するいくつかの追加のセキュリティに関する考慮事項は、[RFC7667]にある。

14. IANA の考慮事項

このドキュメントには IANA アクションはない。

15 参考文献

15.1 参考文献 (Normative)

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/ rfc2119>.

[RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, DOI 10.17487/RFC2736, December 1999, <https://www.rfc-editor.org/info/rfc2736>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <https://www.rfc-editor.org/info/rfc3551>.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, DOI 10.17487/RFC3556, July 2003, <https://www.rfc-editor.org/info/rfc3556>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/ rfc4566>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/ rfc4585>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc- editor.org/info/rfc4588>.

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, DOI 10.17487/RFC4961, July 2007, <https://www.rfc-editor.org/info/rfc4961>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <https://www.rfc-editor.org/info/rfc5104>.

[RFC5124] J Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/ RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <https://www.rfc-editor.org/info/rfc5506>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <https://www.rfc- editor.org/info/rfc5761>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/ rfc5764>.

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, DOI 10.17487/RFC6051, November 2010, <https://www.rfc-editor.org/info/rfc6051>.

[RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <https://www.rfc-editor.org/info/rfc6464>.

[RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <https://www.rfc-editor.org/info/ rfc6465>.

[RFC6562] Perkins, C. and JM. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP", RFC 6562, DOI 10.17487/RFC6562, March 2012, <https://www.rfc- editor.org/info/rfc6562>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <https://www.rfc- editor.org/info/rfc6904>.

[RFC7007] Terriberry, T., "Update to Remove DVI4 from the Recommended Codecs for the RTP Profile for Audio and Video Conferences with Minimal Control (RTP/AVP)", RFC 7007, DOI 10.17487/RFC7007, August 2013, <https://www.rfc-editor.org/info/ rfc7007>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/ RFC7022, September 2013, <https://www.rfc-editor.org/info/rfc7022>.

[RFC7160] P Petit-Huguenin, M. and G. Zorn, Ed., "Support for Multiple Clock Rates in an RTP Session", RFC 7160, DOI 10.17487/RFC7160, April 2014, <https://www.rfc- editor.org/info/rfc7160>.

[RFC7164] Gross, K. and R. Brandenburg, "RTP and Leap Seconds", RFC 7164, DOI 10.17487/ RFC7164, March 2014, <https://www.rfc-editor.org/info/rfc7164>.

[RFC7742] Roach, A.B., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <https://www.rfc-editor.org/info/rfc7742>.

[RFC7874] Valin, JM. and C. Bran, "WebRTC Audio Codec and Processing Requirements", RFC 7874, DOI 10.17487/RFC7874, May 2016, <https://www.rfc-editor.org/info/ rfc7874>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https:// www.rfc-editor.org/info/rfc8083>.

[RFC8108] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", RFC 8108, DOI 10.17487/RFC8108, March 2017, <https://www.rfc-editor.org/info/rfc8108>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/ rfc8174>.

[RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <https://www.rfc- editor.org/info/rfc8285>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/info/ rfc8825>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/ RFC8826, January 2021, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, January 2021, <https://www.rfc-editor.org/info/rfc8827>.

[RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <https://www.rfc-editor.org/info/rfc8843>.

[RFC8854] Uberti, J., "WebRTC Forward Error Correction Requirements", RFC 8854, DOI 10.17487/RFC8854, January 2021, <https://www.rfc-editor.org/info/rfc8854>.

[RFC8858] Holmberg, C., "Indicating Exclusive Support of RTP and RTP Control Protocol (RTCP) Multiplexing Using the Session Description Protocol (SDP)", RFC 8858, DOI 10.17487/RFC8858, January 2021, <https://www.rfc-editor.org/info/rfc8858>.

[RFC8860] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", RFC 8860, DOI 10.17487/RFC8860, January 2021, <https:// www.rfc-editor.org/info/rfc8860>.

[RFC8861] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session: Grouping RTP Control Protocol (RTCP) Reception Statistics and Other Feedback", RFC 8861, DOI 10.17487/RFC8861, January 2021, <https://www.rfc-editor.org/info/rfc8861>.

[W3C.WD-mediacapture-streams] Jennings, C., Aboba, B., Bruaroey, J-I., and H. Boström, "Media Capture and Streams", W3C Candidate

Recommendation, <https:// www.w3.org/TR/mediacapture-streams/>.

[W3C.WD-webRTC] Jennings, C., Boström, H., and J-I. Bruaroey, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Proposed Recommendation, <https:// www.w3.org/TR/webrtc/>.

15.2 参考文献 (Informative)

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <https://www.rfc-editor.org/info/ rfc4383>.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <https://www.rfc-editor.org/info/rfc5576>.

[RFC5968] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP)", RFC 5968, DOI 10.17487/RFC5968, September 2010, <https://www.rfc- editor.org/info/rfc5968>.

[RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, DOI 10.17487/RFC6263, June 2011, <https://www.rfc-editor.org/info/rfc6263>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <https:// www.rfc-editor.org/info/rfc6792>.

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <https://www.rfc-editor.org/info/rfc7478>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https:// www.rfc-editor.org/info/rfc7656>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https:// www.rfc-editor.org/info/rfc7657>.

[RFC8088] Westerlund, M., "How to Write an RTP Payload Format", RFC 8088, DOI 10.17487/ RFC8088, May 2017, <https://www.rfc-editor.org/info/rfc8088>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc- editor.org/info/rfc8445>.

[RFC8829] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol (JSEP)", RFC 8829, DOI 10.17487/RFC8829, January 2021, <https:// www.rfc-editor.org/info/rfc8829>.

[RFC8830] Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", RFC 8830, DOI 10.17487/RFC8830, January 2021, <https://www.rfc- editor.org/info/rfc8830>.

[RFC8836] Jesup, R. and Z. Sarker, Ed., "Congestion Control Requirements for Interactive Real-Time Media", RFC 8836, DOI 10.17487/RFC8836, January 2021, <https:// www.rfc-editor.org/info/rfc8836>.

[RFC8837] Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "Differentiated Services Code Point (DSCP) Packet Markings for WebRTC QoS", RFC 8837, DOI 10.17487/ RFC8837, January 2021, <https://www.rfc-editor.org/info/rfc8837>.

[RFC8872] Westerlund, M., Burman, B., Perkins, C., Alvestrand, H., and R. Even, "Guidelines for Using the Multiplexing Features of RTP to Support Multiple Media Streams", RFC 8872, DOI 10.17487/RFC8872, January 2021, <https://www.rfc-editor.org/info/ rfc8872>.