卒業論文　　2003年度 (平成15年度)

伝送特性に応じた
適応型映像・音声配信機構の設計と構築

慶應義塾大学 環境情報学部

氏名：三島 和宏

指導教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

南 政樹

平成16年1月30日

卒業論文要旨 2003 年度 (平成 15 年度)

伝送特性に応じた適応型映像・音声配信機構の設計と構築

映像・音声配信システムの多くは，パケット伝送にリアルタイム性が要求されるため，再送制御や輻輳制御を行わない UDP を転送プロトコルとして用いる．UDP では，これらの制御がなされないため，アプリケーションとしてこれらの制御を行う必要がある．高品質な映像・音声配信を行うシステムとして DVTS(Digital Video Transport System) があり，独自の輻輳制御手法を持つ．しかし，1) 中継ルータでの対応が必要，2) TCP の挙動にあわせた輻輳制御を行うなど，リアルタイムアプリケーションに適した輻輳制御が行えない．

本研究では，ネットワーク状態の変化がエンドノード間で取得可能な伝送特性として現れることに着目し，エンドノード間で取得可能な到達パケットに対するジッタの揺らぎを主指標に用いた新しい映像・音声配信のための輻輳制御手法を提案した．本手法は，送受信ノード間で必要な情報を交換し伝送特性の計測を行うため，中間ルータなどの特別な機器を必要としない End-To-End モデルでの輻輳制御が可能となる．

本研究の提案する手法の有用性を実証するため，既存の映像・音声配信システムとして DVTS を採用し，本手法を適用するための設計を行った．設計に基づき実装を行った DFCS(Dynamic Framerate Control) は，1) 伝送特性の計測，2) ネットワーク状態検知，3) フレームレート制御の 3 つの機能を有する．伝送特性の計測には，RTCP (Real-time Transport Control Protocol) を用い，送受信ノード間で協調しジッタ値ならびにパケット喪失率の計測を行う．ネットワーク状態の変化を検知した本機構は，送信ノードでのフレーム間引き率を変化させることにより，フレームレート制御を行う．

DFCS を用いて，本手法の有効性について評価を行った．評価結果より，伝送路状態の検知及び状態に適応したレート制御が行えることが確認された．特に，急激なネットワークの状態変化が発生した場合に，本機構が最も有効な動作をすることを確認できた．

本手法を用いることにより，中継ルータなどの特別な機器の準備をすることなく，ネットワークの状態にあわせた品質を動的に決定し，その品質で映像・音声を配信することが可能となった．

キーワード

1. DVTS，2. 輻輳制御，3. ジッタ，4. パケットロス，5. 伝送特性

慶應義塾大学 環境情報学部

三島 和宏

Abstract of Bachelor's Thesis

Academic Year 2003

Design and Implementation of Audio and Video Transport System with Transport Characteristic Adaptation

For real-time packet transport, most audio and video network transport applications use retransmission control and congestion control. Since these packet controlling mechanisms are not implemented on a protocol level, UDP requires to implement these control within application. DVTS(Digital Video Transport System) is a sample application which performs high quality video and audio transport. DVTS implements congestion control algorithms. But these algorithms resolves issues to be solved, 1) performing congestion control correspondence with relay routers, 2) designed to coop with TCP congestion control.

This research focuses on characteristics of the arriving packet in matters of network conditions. A new congestion control algorithm for audio and video transport system using interarrival packet jitter measurable by end node or application is being proposed. This algorithm controls the packet flow by exchanging the status between end nodes without special intermediate routers.

This research proposes the design, implementation and evaluation of the congestion control based on this algorithm, named DFCS(Dynamic Framerate Control). DFCS has three functions, 1) measurement of network characteristic, 2) detection of network status, 3) controlling video framerate. DFCS uses RTCP(Real-time Transport Control Protocol) to exchange information, measurement of the interarrival jitter and packet loss rate. DFCS changes frame drop rate by these status reports. DVTS was used in order to prove the usefulness of this algorithm.

DFCS's operation using this algorithm was evaluated. In the result of evaluation, DFCS tour performs rate control adaptable to the network status. Especially DFCS runs effectively especially on a condition of rapid change in network status.

Therefore this algorithm has achieved to determine the network status and to distribute video and audio with the quality united with the state of a network.

Keywords :
1. DVTS, 2. Congestion Control, 3. Jitter, 4. Packet Loss, 4. Transport Characteristic

Keio University , Faculty of Environmental Information

Kazuhiro Mishima

目 次

図 目 次

表 目 次

第1章　序論

本章では，本研究の背景として，インターネットの広帯域化と映像・音声配信システムの現状について述べ，本研究における問題意識，目的について述べる．

1.1　ネットワークの広帯域化と映像・音声配信

インターネットのデータリンクは急速に広帯域化した．バックボーンネットワークでは，図1.1に示すように 10Gigabit Ethernet(IEEE802.3ae) や Gigabit Ethernet(IEEE802.3ab) をはじめとする広帯域データリンクが普及しつつある．FTTH(Fiber To The Home) など理論上100Mbps の通信速度を提供する個人を対象としたサービスが登場し，エンドユーザレベルにおいても高速なネットワークを利用できるようになった．

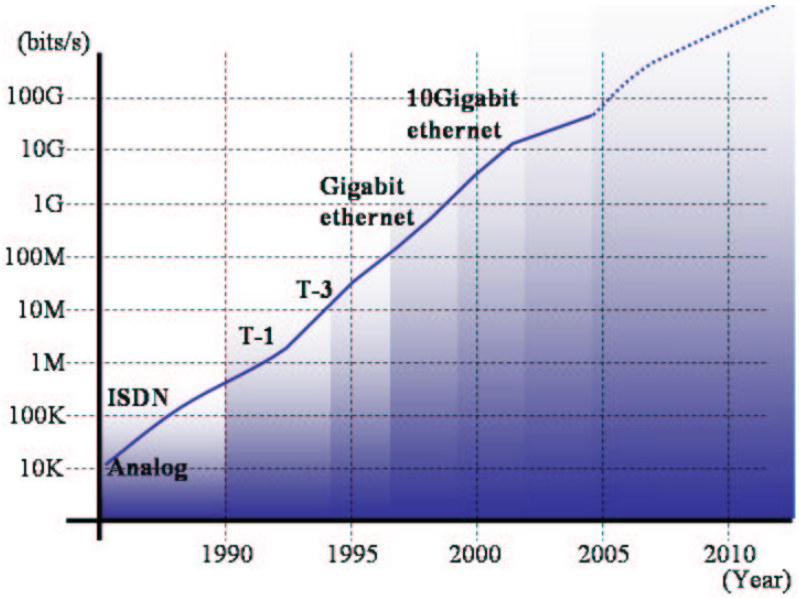

図 1.1: ネットワークの帯域変化

このようなネットワークの広帯域化に伴い，近年，様々な種類の映像・音声配信システムが開発され，広く利用されるようになった．映像・音声のインターネットを介した配信が容易となり，企業レベルだけでなく，個人レベルでも多種多様な配信が行われるようになった．表 1.1にその例を示す．

表 1.1: インターネット放送局の種類と実例

種類	実例
テレビ・ラジオ局によるニュース配信	TBS[1], 文化放送 [2], コミュニティ局 [3] など
企業によるインターネット放送局	Impress TV[4], 企業による広告 など
地域ベースのインターネット放送局	千葉県 [5], 茨城県 [6] など
個人ベースでのインターネット放送局	Triumphal Records[7] など
教育機関での利用	SoI[8], SFC-GC[9] など

1.2 既存の映像・音声配信における問題

映像や音声のインターネットを介した配信の多くは，多少のデータ損失よりもリアルタイム性がより重視されるため，再送制御や輻輳制御を考慮しない UDP(User Daragram Protocol) を用いる．また，WMT(Windows Media Technology)[10] などのように TCP を配信に用いるシステムも登場している．インターネットは伝送路品質が常に変化するため，UDP を用いる映像・音声配信では，トランスポート層ではなく，アプリケーション層で輻輳制御が行われる事が多い．制御が行われない場合，ネットワークに対して不適切なトラフィックを流してしまう可能性がある．

既存の配信システムには中継経路の輻輳状況を検知し，制御を行う機構がある．しかし，輻輳の検知において以下の 3 つの問題が挙げられる．

- 輻輳状況の検知にパケットロスやバッファリング時の問題の発生を主指標として利用する

  映像・音声受信時に行うバッファリングの際のバッファ不足やパケットロスの値によるフレームレート制御は，受信映像・音声の停止やノイズの発生などの要因となるため，映像・音声配信における制御手法としては好ましくない．

- 中間ルータなどの特別な機器によって輻輳状態を検知する

  中間ルータなどの特別な機器を必要とする制御は，ネットワークサービス提供者の対応なしでは行うことができない．映像・音声配信システムは，エンドユーザ間で利用を前提としており，このような輻輳制御には問題がある．

- 輻輳制御の挙動が映像・音声配信アプリケーションに適していない

  上述した 2 つの問題がある既存の輻輳制御手法は，End-To-End モデルを前提とする映像・音声配信システムには適さない．

1.3 本研究の目的

本研究は，UDP を用いた映像・音声配信システムを対象とする．本研究では，インターネットの輻輳状態が，パケット破棄率や到達時間の揺らぎ (ジッタ) といったエンドノード間で取得可能な伝送特性として現れることに着目し，エンドユーザ間のみでネットワーク状態の予測，状態に適応した輻輳制御を行う手法を提案する．

本手法は，アプリケーション層にて，送受信ノード間で時間情報を交換し伝送特性の計測を行い，その値に応じたフレームレート制御を行う．そのため，中間ルータなどの特別な機器を必要としない輻輳制御ができる．また，ジッタの揺らぎやパケット破棄率の値は，映像・音声配信システムでは受信品質に大きく関係する．そのため，これらの値に着目した輻輳制御手法は，End-To-End モデルを前提としたインターネットを介した映像・音声配信システムに適している．

本研究では，本手法の有用性を実証するために，既存の映像・音声配信システムに対し本手法を適用し，映像・音声配信時にネットワーク状態に適応した輻輳制御を行う機構を設計，実装する．また，実装を行った機構の動作を確認することで，本手法の評価を行う．

1.4 本論文の構成

本論文は，8 章により構成される．第 2 章では，本研究における要素技術として，現在のインターネットにおける映像・音声配信の現状，エンドユーザレベルで所得可能な伝送特性の種類とその特徴，TCP での輻輳制御手法について述べる．第 3 章では，本研究にて実装に用いる DVTS の特徴，既存の輻輳制御手法とその問題点について述べる．第 4 章では，その問題点を解決するための本研究のアプローチとその有用性について述べる．第 5 章では，アプローチに基づいたシステムの設計を述べる．第 6 章では，本システムの実装について述べる．第 7 章では，本システムの評価を行う．最後に，第 8 章では本研究のまとめを行う．

第2章　本研究における要素技術

本章では，本研究の要素技術として，インターネットにおける映像・音声配信技術，インターネットにおける伝送特性，インターネットにおける通信の輻輳制御に関する概要を述べる．

2.1　インターネットにおける映像・音声配信

インターネットを介した映像・音声配信時に用いる再生方式には 2 種類存在する．データ全体を受信した後に再生を行うダウンロード再生方式と，データの受信を再生と並行して逐次行うストリーミング再生方式である．図 2.1に，ダウンロード再生方式の概要図，図 2.2に，ストリーミング再生方式の概要図をそれぞれ示す．リアルタイムな映像・音声の配信には，ストリーミング再生方式が用いられる．

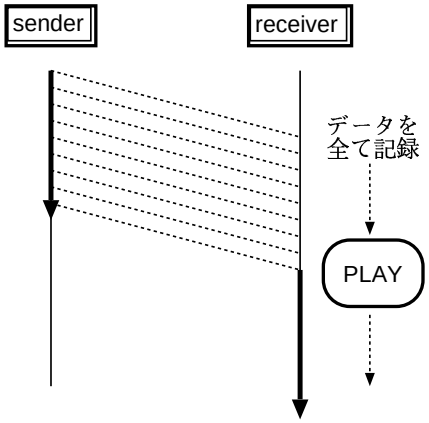

図 2.1: ダウンロード再生方式

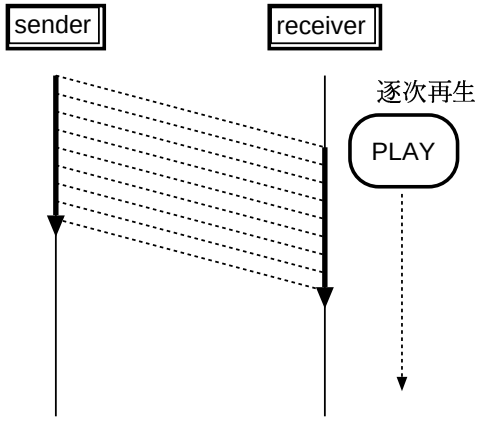

図 2.2: ストリーミング再生方式

2.1.1　ストリーミング再生方式ととジッタの揺らぎ

インターネット上のデータ転送では，パケットの到達時間が保証されない．このため，パケット到着時間に揺らぎが発生する．この揺らぎをジッタと呼ぶ．ストリーミング再生方式におけるジッタの発生と再生との関係図を以下に示す．

図 2.3に示したフローでは，全てのパケットが正しく受信側に到達している．この場合，受信側では正しく映像・音声の再生ができる．しかし，図 2.4に示したフローでは，1 つのパケットが中継ネットワークでの遅延の発生により到達時間が遅れている．この場合，再生のためのデッドラインまでにパケットが受信ノードに到達できず，受信ノードでは再生途中にパケット

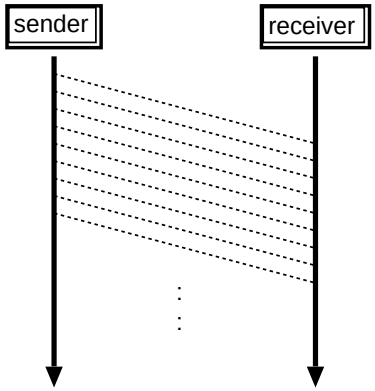

図 2.3: 正常な再生処理

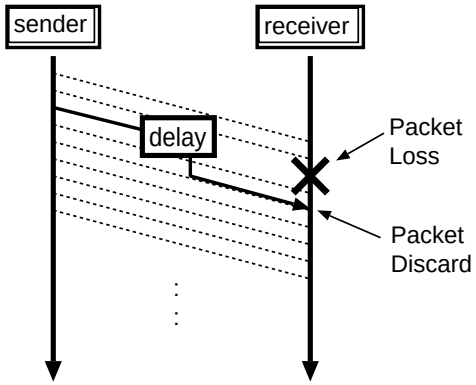

図 2.4: ジッタの発生と再生処理

ロスが発生している．また，デッドラインから遅れたパケットは，受信ノードに遅れて到達したとしても，再生には利用されず破棄される．このように，パケットの到達にジッタが発生すると，再生プログラムでのデータの消費の間で差が発生し，正しく再生できない．

この問題を解消するため，再生プログラムではバッファを用意し，受信時に一定量のデータを蓄積し，再生はその蓄積されたデータから消費する方法が用いられる．これをバッファリングと呼ぶ．バッファリングは，蓄積のためデータの消費を一時的に遅らせる．このため，バッファリングを行う量と再生にかかる遅延時間は比例関係にある．

2.1.2 RTP

映像や音声の配信を行うリアルタイムアプリケーションの多くは，パケットの配送に，TCP (Transmission Control Protocol)[11] ではなく，UDP[12] を用いる．その理由として，TCP による再送制御がある．

TCP では，受信者がパケットを受け取る度に確認応答 (acknowledgement) を送信者に対して送信することにより信頼性の確保をしている．送信者は，パケットを送信する際にタイマを設定し，そのタイマが切れるまでに確認応答を受け取らない場合，パケットロスが発生したと判断し，該当するパケットの再送 (retransmit) を行う．しかし，映像や音声の再生においては，ロスしたパケットが再送されたとしても，データの消費には間に合わないため，再送されたパケットは意味を失う．しかし，映像や音声の配信においても，送信データの信頼性の確保は非常に重要である．

UDP を用いた場合，TCP を用いた場合と比較して，トランスポート層におけるパケット到達順序の保証がなくなる．このため，パケットの到達順序が正しいかどうかの判断をアプリケーション層にて行う必要がある．

到達順序を知る主な手法として，RTP (Real Time Protocol)[13] の利用がある．RTP は，映像・音声といったリアルタイム性が重視されるデータの転送に利用されるプロトコルであり，下位層のトランスポートプロトコルとは独立した設計となっている．RTP は，パケットの順序番号やタイムスタンプなどの情報を付加する．この情報を元に，受信側は正しいパケットの順番を知ることができる．しかし，RTP は配送や配送の到達順序自体を保証しない．

2.1.3 RTCP

映像や音声の配信に主に用いられる RTP では，パケットの順序やパケットの喪失といった情報を計測することが可能であるが，伝搬遅延・ジッタなどのネットワーク状態を把握するために必要な情報を取得できない．これらの情報を取得するために，RTCP(Realtime Transport Control Protocol)[13] が用いられる．RTCP は，RTP セッションに対して遅延・ジッタ・帯域幅・輻輳状態などのフィードバック情報をセッション参加者間で交換する．これにより，映像・音声配信システムは，ネットワークの状態を把握することができる．

例えば，ネットワーク状態不良時には品質を下げ，逆に状態が改善したときには品質を上げるといった，ネットワーク状態に適応させた品質での映像・音声配信を行うことができる．

しかし，RTCP は，RTP のための通知プロトコルであり，データ配送や TCP のようなエンドノード間での到達性保証を行わない．

2.2 インターネットにおける伝送特性

インターネットにおける通信路の状態は，さまざまな要因により変化する．通信路状態の変化は，パケットの伝送特性の変化として表れる．この伝送特性の変化を観測することでネットワーク状態の予測や検知を行うことが可能となる．

特にエンドノード間で情報取得可能な伝送特性として実効帯域幅，往復伝搬遅延，遅延と伝搬時間の揺らぎ，パケット喪失率の 4 つが挙げられる．本節では，それらの特徴について述べる．

実効帯域幅 (実効スループット)

送信ホストから受信ホストまで実際に利用可能な帯域幅を実効帯域幅と呼ぶ．実効帯域幅は，パケットが送信ホストから受信ホストまで到達する時間とパケットサイズなどをもとに計算できる．その測定にはパケットペアスキームを用いたものが存在するが，測定に多大な時間と通信量を要する．

往復伝搬遅延時間 (RTT)

送信ホストと受信ホスト間でパケットが往復する時間を RTT(Round Trip Time) と呼ぶ．RTT によりネットワークの大まかな遅延を知ることができる．RTT は，Ping などのツールを用いて測定できる．

片方向遅延時間と伝搬時間の揺らぎ (ジッタ)

パケットが送信ホストから受信ホストまで到達するまでにネットワークの持つ帯域幅に応じた時間を要する．さらに，4.3節にて述べる要因によりパケットの到達時間は前後する．これを片方向遅延時間と呼ぶ．各パケットは送信された時間からある程度遅れて受信ホストに到達する．この伝搬時間の揺らぎは，到達時間に関して保証のないインターネットでは一定ではない．

パケット喪失率

送信パケット数と受信パケット数を比較することで，中継経路におけるパケット喪失の程度を知ることができる．喪失したパケットの割合を，パケット喪失率と呼び，喪失したパケット数を送信パケット数で割ることで求める．パケット喪失率によりネットワークのトラフィック状況を予測できる．

2.3 インターネットにおける通信の輻輳制御

インターネットは，End-To-End モデルでの通信を行う．エンドノード間では，宛先ホストに対してなるべくパケットを転送しようとする最善努力型の通信が行われる．しかし，その配送に対する保証は行われない．さらに，インターネット全体の通信を制御するための機構は存在しない．また，エンドノードでは，中継ネットワークの状態を確認できない．

中継経路上のトラフィック量増大により，ネットワーク全体のパフォーマンス低下や有効な通信が行われなくなる．この状態を輻輳と呼ぶ．中継ネットワーク自体は，輻輳状態から回復するための機構を持たない．このため，エンドノードが宛先ホストとの通信を行う際に，何らかの手法を用いてネットワークの状態を把握し，それに応じた通信を行う必要がある．これを輻輳制御と呼ぶ．

インターネットにおける多くの通信は，トランスポートプロトコルである TCP がネットワークの途中経路上での輻輳を検出し，送出するパケットのウィンドウサイズを動的に調整することで行う [14]．しかし，UDP は，輻輳制御機構を持たない．このためアプリケーションがこの輻輳制御を行う必要がある．

ここでは，インターネットにおける輻輳制御手法の例として TCP を用いた輻輳制御について述べる．TCP における主な輻輳制御方式として，TCP Tahoe，Reno，Vegas の 3 つが挙げられる．特に，TCP Vegas は 2.2節にて挙げた伝送特性のひとつである往復伝搬遅延時間を用いることから，本研究の視点に近い手法である．

TCP Tahoe

TCP Tahoe[15] は，1988 年に V.Jacobson により提唱された TCP における輻輳制御手法である．この手法は，スロースタートアルゴリズム，輻輳回避アルゴリズム，高速再転送アルゴリズムの 3 つのアルゴリズムが含まれる．

スロースタートアルゴリズムおよび輻輳回避アルゴリズムは，対向エンドから返される確認応答 (ACK) に基づき輻輳ウィンドウサイズ制御を行う．ウィンドウサイズの制御は，Slow Start Phase と Congestion Avoidance Phase の 2 つのモードがある．これらは，モード移行時の閾値である Slow Start Threshold($ssth$) の値によって切り替えられる．$ssth$ の値は，以下に示す式のように，セグメントロスが発生するごとに更新される．

$$ssth = \frac{cwnd(t)}{2} \tag{2.1}$$

転送を開始した直後は Slow Start Phase に入る．転送開始およびセグメントロス後の転送再

開時は，$cwnd$ を 1 とした上で，新しい ACK を受け取るたびに，以下に示す式のようにウィンドウサイズ ($cwnd$) を変更する．

$$cwnd(t + t_a) = cwnd(t) + 1(cwnd(t) < ttsh) \tag{2.2}$$

閾値 $ssth$ を超えると，Congestion Avoidance Phase に入り，以下に示す式のようにウィンドウサイズを変更する．

$$cwnd(t + t_a) = cwnd(t) + \frac{1}{cwnd(t)}(cwdn(t) > ssth) \tag{2.3}$$

セグメントロスは，各セグメントに設定されたタイマがタイムアウトする，もしくは，Duplicate ACK を受信することによって検出される．Duplicate ACK によるセグメントロスを検出し，セグメントの再送を行うアルゴリズムが，高速再転送アルゴリズムである．

TCP Reno

TCP Reno[16][14] は，1990 年に V.Jacobson により提唱された TCP における輻輳制御手法である．この手法には，TCP Tahoe の持つ高速再転送アルゴリズムが実行された際に転送速度を落としすぎる問題を回避するために，高速リカバリアルゴリズムが含まれている．

TCP Reno では，タイムアウトによるセグメントロスが検出された場合，TCP Tahoe と同様のウィンドウサイズの制御を行う．しかし，Duplicate ACK によりセグメントロスが検出された場合，ネットワークの輻輳状態が比較的軽微であると認識し，高速リカバリアルゴリズムに基づき Fast Recovery を行う．Fast Recovery の動作を以下に示す．

1. 連続した 3 つの重複 ACK を受信した場合，閾値 $ssth$ を現在のウィンドウサイズ $cwnd$ の $\frac{1}{2}$ にする
2. ロスしたパケットを再送し，ウィンドウサイズ $cwnd$ の値を 1 にて設定した $ssth + 3$ セグメントとする
3. それ以降，重複 ACK と同じ重複 ACK を受信するたびに $cwnd$ の値を 1 セグメントずつ増加させる
4. 新たなセグメントを要求する ACK を受信した場合，$cwnd$ の値を $ssth$ に変更する

この Fast Recovery に成功した後は，TCP Tahoe にて説明した Congestion Avoidance Phase に入る．

TCP Vegas

TCP Vegas[17] は，1994 年に Brakmo, Peterson らにより提唱された TCP における輻輳制御手法である．TCP Vegas では，これまでのセグメントロスを利用したウィンドウサイズの調整を行わず，送信した RTT(Round Trip Time) を正確に測定し，用いることで輻輳制御を行

う．この手法では，RTT が大きくなればネットワークが輻輳していると判断し，ウィンドウサイズを小さくし，逆に RTT が小さくなればウィンドウサイズを大きくする．

TCP Vegas においても，Slow Start Phase と Congestion Avoidance Phase があり，これらの切り替えには，計測された RTT(rtt) と計測中の最小 RTT($base\_rtt$) が用いられる．

Slow Start Phase では，以下に示す式のようにウィンドウサイズ $cwnd$ が変更される．

$$base\_rtt = rtt \tag{2.4}$$

$$cwnd(t + t_a) = cwnd(t) + \frac{1}{2} \tag{2.5}$$

RTT が $base\_rtt < rtt$ となった場合，すなわち RTT に遅延が発生した場合，Congestion Avoidance Phase に入る．

まず，推定スループットと実際のスループットの差を求め，その値と定数 α , β によりウィンドウサイズ $cwnd$ が以下のように変更される．

$$diff = \frac{cwnd}{base\_rtt} - \frac{cwnd}{rtt} \tag{2.6}$$

$$cwnd(t + t_a) = \begin{cases} cwnd(t) + 1 & (diff < \frac{\alpha}{base\_rtt}) \\ cwnd(t) & (\frac{\alpha}{base\_rtt} < diff < \frac{\beta}{base\_rtt}) \\ cwnd(t) - 1 & (\frac{\beta}{base\_rtt} < diff) \end{cases} \tag{2.7}$$

2.4 まとめ：要素技術

本章では，本研究の要素技術について述べた．まず，映像・音声配信システムの概要として，ストリーミング再生方式とオンデマンド再生方式の違い，ジッタが再生に及ぼす影響，RTP，RTCP についてそれぞれ述べた．次に，エンドノード間で取得可能な伝送特性を挙げ，その特徴について述べた．さらに，インターネットにおける輻輳制御手法として TCP での輻輳制御手法を挙げ，TCP Tahoe，Reno，Vegas の 3 手法の特徴について述べた．

3章では，既存の映像・音声配信システムのうち，DVTS を挙げ，その特徴，既存の輻輳制御手法の特徴と問題点について述べる．

第3章　既存技術：DVTS

本章では，既存の映像・音声配信システムの例として DVTS の概要について述べる．3.1節では，DVTS の持つ特徴について述べる．3.2節では，DVTS の既存の輻輳制御手法について述べ，3.3節にてその問題点のまとめを行う．

3.1　DVTS の持つ特徴

民生用デジタル AV 機器を用いた映像・音声配信を行うシステムとして，DVTS (Digital Video Transfer System)[18] がある．

DVTS は，IEEE1394 インタフェース [19][20][21] から DV フォーマット [22] のストリームを取得し，そのストリームに対し IP データグラム化を行い，インターネットを介して映像・音声の転送を行う．このシステムを利用することにより，リアルタイム，かつ，高品質な映像の配信が可能である．

DVTS は，以下の 4 つの機能を有する．

- IPv4/IPv6 への対応
- RTP を利用した通信
- フレーム間引き機能を有し，送出データ量の調整が可能 (3.1.1節にて後述)
- 複数地点へのストリーム送信が可能なマルチキャストへの対応

DVTS は，DV フォーマットの持つ以下の 5 つの特徴を持つ．

- フレーム内圧縮
- 解像度は，720pixel × 480pixel
- フルフレーム時のフレームレートは，29.97fps
- 圧縮率は一定
- フルフレーム時の送出データ量は，約 32Mbps

3.1.1 使用帯域の調整

フレーム内圧縮である DV フォーマットは，映像フレームがすべて送信されなくても映像の再生が可能である．このため，DVTS では，オプションで指定する値 (-f オプションとして整数値を指定) を分母とするフレーム間引き率により，映像の間引き処理を行う．この手法により，DVTS は使用するネットワーク帯域の調整を行う．

図 3.1は，最上段から順に，映像間引きを行わないストリーム，映像間引き率 $\frac{1}{2}$ のストリーム，映像間引き率 $\frac{1}{3}$ のストリームを示す．図中の長方形の枠は映像フレームを，黒い正方形の枠は音声フレームをそれぞれ示す．DVTS が送出する DV ストリームの構成を 3.1式に示す．間引き率の変化に伴い，DV ストリームに占める映像フレーム数が変化する．DVTS では，音声フレームの欠損は受信者において明らかなノイズや音飛びの原因となるため，音声フレームの間引きは行わない．

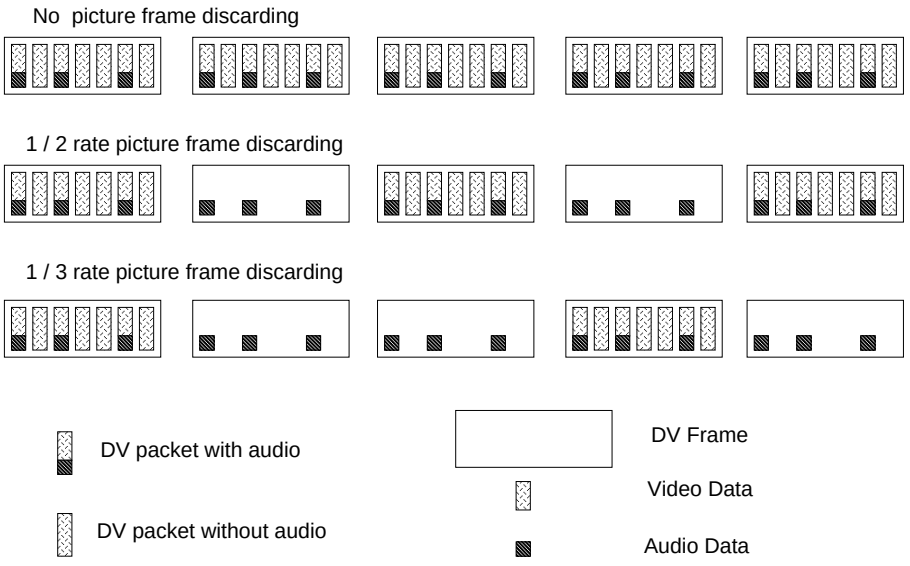

図 3.1: DVTS の映像フレーム間引き

$$F = AF + (VF \div DR) \tag{3.1}$$

F： DV ストリーム全体のフレーム数

AF： 音声フレーム数 (AudioFrame)

VF： 映像フレーム数 (VideoFrame)

DR： 映像フレーム間引き率 (VideoFrameDropRate)

最上段の映像間引きを行わない場合と中段の映像間引き率 $\frac{1}{2}$ のストリームを比較して，映像のフレーム数が半分となっている．DVTS では，フルフレーム時と比較して映像のフレームデータが不足した場合，受信側でひとつ前の映像フレームを再利用する機能を有している．この機能により，受信側ではノイズを発生することなく映像の再生を行う．

図 3.2は，DVTS における映像フレーム間引き率と使用帯域の関係を示す．フルフレーム時は 30Mbps を超えるデータ量が流れるが，フレーム間引き率を $\frac{1}{10}$ に設定すると，約 5Mbps のデータ量まで抑えることができる．いずれの場合も，音声フレーム間引きは行わないため，音声フレームが使用する帯域は変化しない．

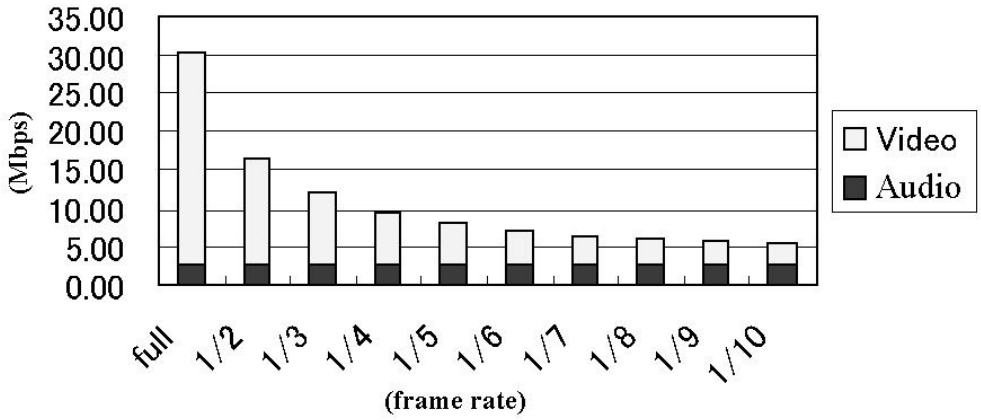

図 3.2: 映像フレーム間引き率と消費帯域

3.1.2 パケットフォーマット

DV(Digital Video) のための RTP パケットフォーマット [23](以降，DV/RTP) を図 3.3に示す．DV/RTP は，RTP ヘッダと DV データ部分により構成される．DV フォーマットは，映像・音声・システムなどのすべてのデータは 80byte 長の DIF ブロックに区切られている．DV/RTP には，DV データ部分に複数の DIF ブロックが詰められる形で構成されており，パケット中に含まれる DIF ブロック数を自由に設定できる．このため，DV/RTP のパケット長は 80byte 単位で変化する．

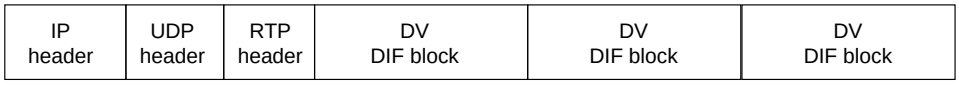

図 3.3: DV/RTP パケットフォーマット

3.2 DVTS における輻輳制御

DVTS は，UDP/RTP を用いた通信を行う．DVTS では，アプリケーションとして以下のような輻輳制御機能を持つ．

3.2.1 TCP-Friendly Congestion Control for DVTS

TCP との親和性のある UDP の研究に TCP Friendly[24] がある．DVTS では，この TCP Friendly の機能を持たせることによって，TCP と協調した輻輳制御 [25] を行う．

TCP Friendly は，1997 年に S.Floyd らによって提案された UDP 等による輻輳制御を行わないトラフィックを流すアプリケーションが TCP とネットワーク帯域を公平に共有するためのアルゴリズムである．UDP などの輻輳制御を行わないトラフィックは輻輳が発生してもデータ量を維持したまま，データ送信を行う．しかし，TCP は輻輳が発生するとウィンドウサイズを小さくして送信するデータ量を減少させる．このため，UDP トラフィックと TCP トラフィックが混在する環境では，お互いの使用するネットワーク帯域に不公平が生じてしまう．

TCP の最大スループットの計算は以下の式を用いて行う．

$$T < \frac{1.5\sqrt{\frac{2}{3}} \times B}{R \times \sqrt{P}} \tag{3.2}$$

T： トラフィック量

B： 接続リンクの MTU

R： RTT(Round Trip Time)

p： 最近のパケット喪失率

通常，TCP はこの値 T より低い値をとる．このため，この計算式より高いスループットを得ているフローは TCP よりも高いスループットを得ており，TCP との親和性がないと判断される．

TCP Friendly による適応的配信機構

DVTS では，3.2式の計算式を元に，DV/RTP ストリームに対して TCP との親和性を持たせている．DVTS によるトラフィックのスループットが，同一リンクにおいて TCP が得ることのできる最大スループットを超えた場合，映像間引き率を大きくし送出されるパケット数を減少させる．

TCP と同様に，DVTS においてもパケットの喪失が観測されない場合，利用可能なネットワーク帯域資源があると判断し，映像間引き率を減少させ送信レートを大きくする．しかし，DVTS では，映像間引き率と使用するネットワーク帯域の間には比例関係がない．例えば，$\frac{1}{5}$ から $\frac{1}{4}$ に変更した場合と $\frac{2}{1}$ から $\frac{1}{1}$ に変更した場合では，消費帯域の増加幅が大きく変化する．

このため，単純に映像間引き率を減少させるだけでは，ネットワークに対して急激な変化を発生させてしまう可能性がある．そのため，DVTS では，以下の式を用いて映像間引き率の小ささに反比例して映像間引き率を減少させにくくしている．

$$\frac{(f \times n)}{a} > 1 \tag{3.3}$$

f： 現在の映像間引き率 (-f オプションの値)

n： パケットロスが 0 であると RTCP メッセージを受信した回数

a： 定数 (この値が大きいほど間引き率が減少しにくくなる)

問題点

TCP Friendly を用いた協調的輻輳制御機構は，エンドノード間のみでネットワークの輻輳状態を検知し，フレームレートのコントロールを行う．しかし，TCP Friendly の目的は，UDP などの非 TCP フローを TCP の挙動のように制御することである．本手法を用いる場合，TCP との親和性は確保できるが，UDP フローが TCP フロー以上に帯域幅を得られない．このため，ネットワークが輻輳状態でない場合であっても，十分な品質で映像・音声配信を行うことができない．また，制御を行う際に用いる計算式にはパケットロスの値が主指標のひとつとして利用されており，パケットロスの発生が制御の前提となっている．パケットロスの発生は，受信者での映像や音声の乱れにつながるため，主指標として用いる手法は問題がある．

3.2.2 Packet Lossless TCP-Friendly DVTS using ECN

ECN(Explicit Congestion Notification)[26] は，中継ルータがエンドノードに対してネットワークの輻輳状態を明示的に通知する機構である．中継ルータは，転送待ちパケット数を検査し，その数が一定を超える場合，ネットワークが輻輳状態にあると判断し，中継パケットの TOS フィールドに CE(Congestion Experience) ビットを立てる．エンドノードでは，このビットを調査することで，ネットワークの輻輳状態を判断する．

3.2.1節にて挙げた TCP Friendly に基づく輻輳制御手法では，パケットロスの値が計算式内に含まれており，輻輳検知にパケットロスが起きることが前提となっている．しかし，この手法を用いることにより，パケットロスを起こすことなく，輻輳を検知できる．

ECN による適応的配信機構

DVTS では，3.2.1節の TCP Friendly による適応的配信機構に ECN 機能を持たせることによりパケットロスを起こさない適応的配信機構を実現している．

基本的な動作は前節の TCP Friendly による適応的配信機構と同様である．しかし，受信プ

ログラムが受信した DV/RTP パケットに CE ビットが立っている場合，DVTS は実際にパケットロスが発生していなくてもパケットロスが発生したと認識する．

DVTS における ECN による適応的配信の動作は以下の通りである．

1. 送信プログラム (以下，dvsend) が受信プログラム (以下，dvrecv) に対して，DV/RTP パケットを送信する
2. 中継ルータは，輻輳状態に応じて CE ビットを設定する
3. dvrecv は受信したパケットの CE ビットを検査し，CE ビットが立っている場合，そのパケット数を記録する
4. dvsend は，dvrecv に対して，RTCP SR メッセージを定期的に送信する
5. dvrecv は，RTCP SR メッセージを受信した後，前回 RTCP RR メッセージを送信した時間と RTCP SR メッセージを受信した時間の差・RTP パケットロス数・ECN マークされた RTP パケット数の 3 つの情報を RTCP RR メッセージとして dvsend に対して送信する
6. dvsend は，RTCP RR メッセージを受信し，情報を元に送出するフレームレート数を調整する

問題点

ECN を用いた協調的輻輳制御機構では，TCP Friendly による輻輳制御では不可能であった DV ストリームのパケットロスなしにフレームレートの制御ができる．しかし，ECN 機能を実現するためは，中間ルータにおいてネットワークの輻輳状態に応じたパケットの TOS フィールド設定が必要である．中間ルータが ECN に対応していない場合，この手法を用いた輻輳制御を行うことができない．

3.3 まとめ：既存の輻輳制御手法における問題点

3.2節において，DVTS の持つ既存の輻輳制御手法の問題点を述べた．以下に 3 つの問題点を整理する．

- UDP の挙動を TCP に近似させた挙動を取らせる手法は，UDP フローに対して最適化された輻輳制御手法でない
- 輻輳制御の指標にパケットロスの発生が前提となっているが，パケットロスは映像・音声の乱れの要因となるため指標として適切ではない
- エンドノード間のみでネットワークの輻輳状態を検知できない

DVTS における既存の協調的輻輳制御手法では，これらの問題全てを解決することができない．そこで，本研究では，4章にて新たな協調的輻輳制御手法を提案する．

第4章　伝送特性による協調的輻輳制御手法の提案

本章では，伝送特性による協調的輻輳制御手法の提案を行う．4.1節にて本研究に求められる要件を整理し，4.2節に最も適切な伝送特性について検討を行う．4.3節以降では，伝送特性としてジッタを用いる妥当性の検討ならびに処理に必要な時間精度について述べる．

4.1　伝送特性に応じた輻輳制御

映像・音声配信システムなどのリアルタイムアプリケーションでは，その性質上ネットワーク状態に応じた輻輳制御が必要不可欠である．また，輻輳制御を行うに当たり，エンドノード間で取得可能であり，かつ取得の容易な指標を用いることが好ましい．

本研究では，エンドノード間で取得可能なネットワークの特性値を利用したリアルタイムアプリケーションに最適化された輻輳制御手法を提案する．本研究に求められる要件は，以下の3つである．

- エンドノード間でのみ取得可能な情報を基にした輻輳制御手法
- なるべくパケットロスを発生させないで輻輳検知可能な輻輳制御手法
- 特定のRTPパケット伝送のために最適化された輻輳制御手法

4.2　要件を満たす伝送特性の検討

2.2節において，エンドノード間で取得可能な伝送特性について述べた．本節では，本研究にて確立する協調的輻輳制御手法に最も適切な伝送特性を検討する．それぞれの特徴を以下にまとめ，上述の要件を満たすことのできる伝送特性を検討する．

- **実効帯域幅 (実効スループット)**
  この値を利用することで最も正確な制御を行うことができる．しかし，計測にかかる時間や中継ネットワークにかける負荷が高いため，実用的ではない．
- **往復伝搬遅延 (RTT)**
  ICMPを用いることで容易に計測できる．しかし，映像・音声配信では片方向での値が重要であり，往復遅延時間まで求める必要がない．
- **遅延と伝搬時間の揺らぎ (ジッタ)**
  RTCPなどの手法を用いることにより容易に計測できる．通信状態に応じて値が変動するため，通信状態の予測ができる．しかし，予測精度が実効帯域幅を求めるよりも悪い．

- **パケット喪失率**
  RTP シーケンス番号などにより容易に計測できる．しかし，映像・音声配信においてパケット喪失の発生は，ノイズの発生につながるため，この値を主指標として用いた制御は好ましくない．

以上の結果を表 4.1にまとめる．

表 4.1: 各伝送特性の特徴

伝送特性	計測の容易さ	信頼性	有用性
実効帯域幅	×	◎	△
往復伝搬遅延時間	◎	○	×
遅延時間とジッタ	◎	○	○
パケット喪失率	◎	○	△

この結果から，伝送特性として” 遅延及び伝搬時間の揺らぎ (ジッタ)” は，計測の容易さや有用性の観点から適切である．また，2.1.1節より，リアルタイムアプリケーションにおける遅延やジッタは再生と密接な関係がある．この値を基に輻輳制御を行うことは，リアルタイムアプリケーションにおける輻輳制御手法として最も適している．

4.3 伝送特性としてのジッタの特徴

インターネットにおけるパケットの伝搬は，送信ホストや受信ホストの計算機資源，中継ネットワークの状況，その他の要因により影響を受ける．ここでは，遅延時間やジッタの変化を与える要因を整理する．特に，配信システムに DVTS を用いた場合の伝搬遅延の要因と発生箇所についてまとめる．

IEEE1394 アイソクロナス転送開始の遅延

IEEE1394 では，同期通信を行うアイソクロナス転送は優先的に実行される．アイソクロナス転送開始時には，バスに共通なクロック源を管理するサイクルマスターにアクセス権を与え，サイクルスタートと呼ばれる特殊なタイミング要求を送信する．通常，サイクルスタートは 125 マイクロ秒 $\pm$ 12.5 ナノ秒 または $8kHz \pm 10ppm$ に設定される．しかし，サイクルスタート送信時にデータ転送が行われている場合，サイクルスタートは延期され，アイソクロナスデータ転送にジッタが発生する．

符号化・複合化処理

映像・音声は，利用されるフォーマットによって符号化と複合化が行われる．DVTS では，ソフトウェア的な符号化・複合化処理は行われないが，DV カメラなどのハードウェアにおいてこれらの処理が行われる．これらの処理時間はデバイスの持つバッファのサイズなどによって大きく左右される．

計算機におけるパケット化処理

DVTS では，送信ホストにおいて，IEEE1394 バスを経由して DV パケットを受け取り，インターネットを介して転送するために DV/RTP パケットとして再構成する．また，受信ホストにおいては，到着した DV/RTP パケットの DV パケットへの復号化を行う．これらの処理を行う際に CPU の高負荷状態などの原因により，処理に遅延が発生する．

パケット伝搬時間の変化

IP パケット化された DV データは，インターネットを通じて転送される．このインターネットにおいて，物理層での伝送メディアでの物理的な信号伝搬やその他のレイヤーでの処理などにより，相手先ホストへのパケット到達時間が変化する．特にインターネットでは，これらの時間の保証がなされないため，伝搬遅延として現れる．

インタフェースでのバッファリング

エンドノードや中継ルータに備えられているネットワークインタフェースカードには，一定のバッファが用意されている．バッファでは，パケットを一時的に蓄え，一定以上たまった後に処理を行う．このバッファのサイズや処理のタイミングは，インタフェースカードによって異なる．そのため，インタフェースごとに異なる遅延が発生する．

中継ルータの性能とキューイング処理

中継ルータでは，一定のアルゴリズムに基づき到達したパケットのルーティング処理が実施される．すべてのパケットは，まずその処理列であるキューに入る．そして，ルータはキューに貯まったパケットを順に処理していく．ルータの性能不足やネットワークの輻輳によるパケット増などの要因により，パケット処理に時間がかかり，パケット転送に大きな遅延が発生する．

中継経路の変化

インターネットでは，常に中継経路が変化する可能性がある．経路が変化することにより経由するルータ数や中継ネットワーク環境 (帯域幅など) が変化する．これに伴い，パケットが相手ホストまで到達する時間が変化する．

4.4 ジッタによる輻輳制御

4.4.1 本手法の有効性検証

4.3節に挙げた要因により，パケットの伝送時間は変動する．ジッタの揺らぎの多くは，計算機やネットワーク状態が高負荷になることで発生する．本節では，通信状態の変化が実際にどのような形でジッタの揺らぎとして現れるかを確認することで，伝送特性としてジッタを用いた輻輳制御が可能かどうか検討する．

ジッタの発生とネットワーク状態の変化の関連性を確認するために，映像・音声配信システ

ムとして DVTS を用いた実験を行う．ジッタの値を測定するために，図 4.1に示す実験ネットワークを構築した．このネットワークは，擬似的に輻輳状態を発生させるため，中継ルータにおいて 35Mbps の帯域制限をかけている．

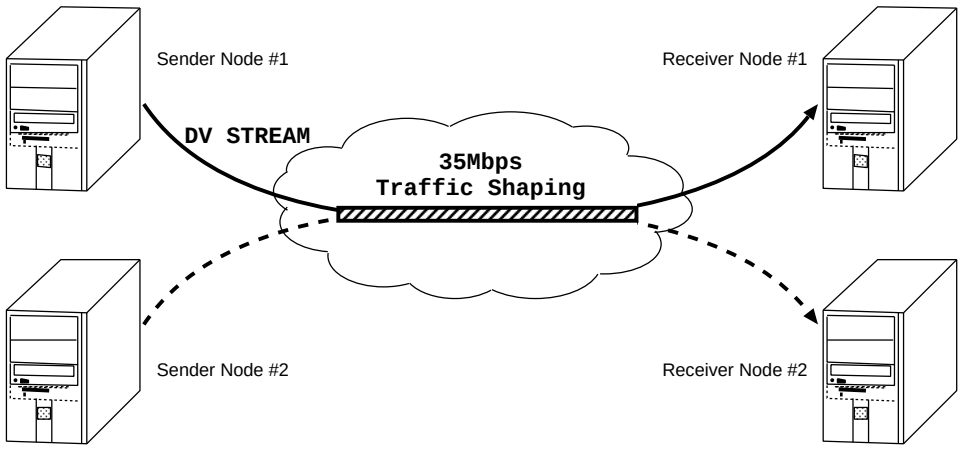

図 4.1: テスト時の環境概要図

図 4.2は，図 4.1に示したネットワークに DV ストリーム 1 本が流れている状態で，バーストトラフィックにより輻輳が発生した場合の $SenderNode\#1$ でのジッタの変動とパケットロスの値を示したグラフである．バーストトラフィックの発生には，Netperf[27] を用いた．Netperf は，パケットペアスキームを用いてネットワークの帯域幅を測定するアプリケーションであり，測定時にバーストトラフィックを発生させる．

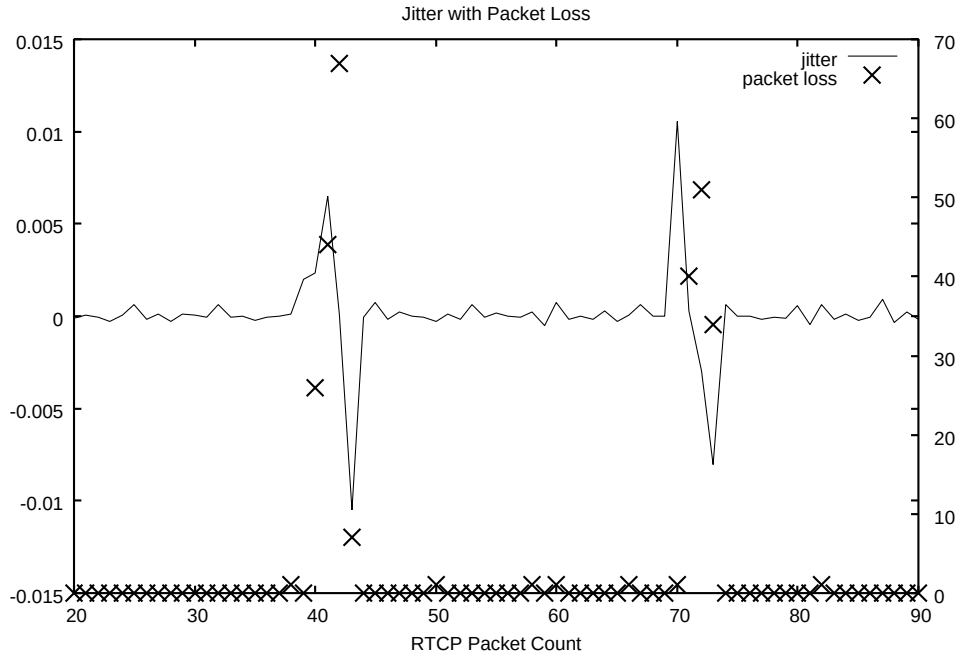

図 4.2: 輻輳時のジッタとパケットロス - DV ストリーム と Netperf

このグラフから，バーストトラフィックの発生とともにジッタは大きく増加し，バーストトラフィックの収束とともにジッタは大きく減少していることが確認できる．このように，ネットワークの状態に応じてジッタは，正負に変動する．この値を利用することで通信状態を把握できることが確認された．

4.4.2 ジッタのみによる輻輳制御の問題

次に図 4.3を以下に示す．このグラフは，先程と同様のネットワークにおいて，DV ストリームが 2 本流れたときの $SenderNode\#1$ でのジッタの変動とパケットロスの値を示したものである．

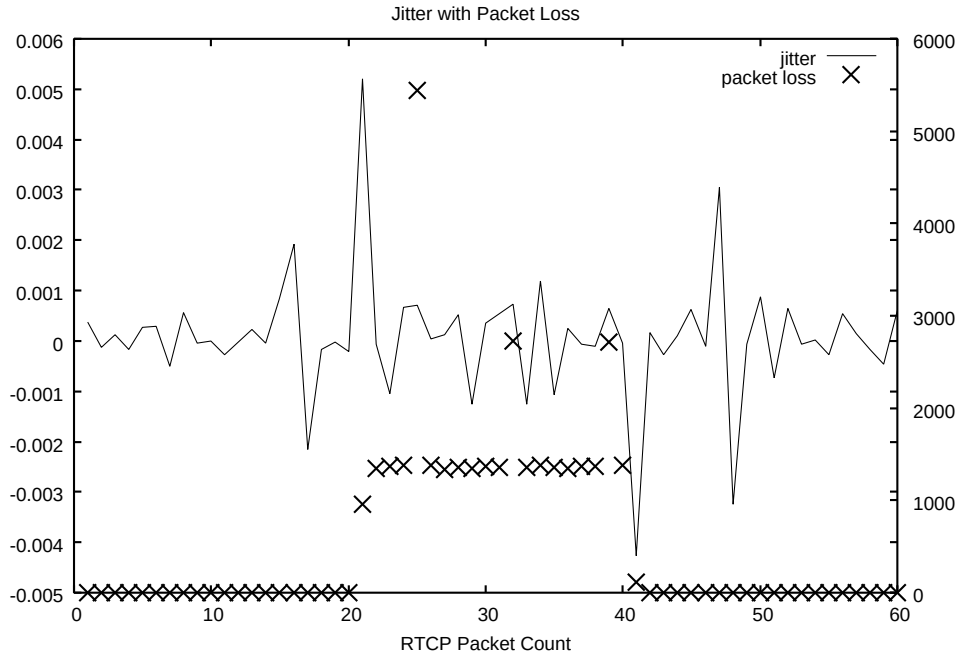

図 4.3: 輻輳時のジッタとパケットロス - DV ストリーム 2 本

この場合も，バースト的なトラフィックの発生ならびに収束とともにジッタが大きく正負に変動していることが確認できる．しかし，一定時間が経過すると，ジッタは大きな変化から小さな変化へと移行し，トラフィックが収束するまで一定以上の増減を繰り返していることも確認できる．

このように，ジッタの値は，輻輳状態になる直前やバーストトラフィックの収束直後などネットワークの状態が急激に変化する際に大きく変動するが，継続的な輻輳状態の間やネットワーク状態が緩やかに変化する際には緩やかな変動を続ける．このため，ジッタのみで全てのネットワークの変化を把握することは困難である．

問題の解決手法

ジッタの値のみで輻輳状態を把握しようとした場合，急激なネットワーク状態の変動には対応できるが，緩やかなネットワーク状態の変動には対応できない．そこで，ジッタの値だけで

なく，映像・音声配信の際の指標となるパケットロスの値を同時に補助的な指標とする．これにより，ネットワーク状態が緩やかな変化をしている場合のネットワーク状態を把握することができる．

4.4.3 処理に必要な時間精度

本研究では，伝送特性による輻輳制御を行う際，ネットワーク状態把握のためにジッタの計測を行う．ジッタ値の計測には，パケット伝送時の遅延時間を実際に計測しなければならない．計測処理において時間情報を利用する．この時間情報の精度は，ネットワーク状態把握の精度に影響を与える．例えば，100 マイクロ秒の時間変動は，1 ミリ秒単位での計測では変動が観測できない．

このため，輻輳制御を行おうとする映像・音声配信システムが用いる RTP パケットが送信ノードから受信ノードに伝送される時間がどの程度であるかを把握し，ジッタ値の計測に必要な時間精度をあらかじめ決定しておく必要がある．

4.5 まとめ：ジッタによる輻輳制御

本章では，エンドノード間で取得可能な伝送特性情報による輻輳制御手法の提案を行った．また，映像・音声配信システムにおける輻輳制御に適した伝送特性の検討を行い，ジッタの揺らぎやパケット喪失率を用いることを提案した．

ジッタの揺らぎが発生する要因についてまとめ，その変化がネットワーク状態の変化とともに現れることを実験により示すことで，本手法の有効性について検証した．さらに，本手法を用いた輻輳制御に必要な時間精度の検討の必要性について述べた．

5章では，本手法の実証例として DVTS を用いる場合の設計について述べる．

第5章　適応型映像・音声配信機構の設計

本章では，伝送特性による協調的輻輳制御手法を実証例として DVTS に対して適用する際の設計について述べる．5.2節では，本機構の設計要件と設計概要について述べる．5.3節以降では，本機構の各部分における詳細な設計について述べる．

5.1　伝送特性による輻輳制御手法の DVTS への適用

本研究では，ジッタやパケットロスを利用した輻輳制御が有効に機能するかを実証するため，既存の映像・音声配信システムに本手法を適用し，評価する．本研究では，その実証例として DVTS[18] を用いる．

5.1.1　処理に必要な時間精度

DVTS では，DV/RTP パケットを用いたフレーム伝送を行う．本節では，DV/RTP パケットの伝送にかかる時間を求め，ジッタ計測処理に必要な時間精度を検討する．

DVTS が送出する DV/RTP パケットの構成は，図 3.3にて示した．この DV/RTP パケットは，それぞれの部分が以下のサイズで構成される．ここで示した各サイズは，ネットワーク層プロトコルに IPv4 を用いた場合の値である．

- IP Header: $20bytes$
- UDP Header: $8bytes$
- RTP Header: $12bytes$
- ペイロード: $80bytes \times DIFblockNumber(default : 17)$

すなわち，DV/RTP パケットのサイズは，デフォルトの状態で $1400bytes$ である．このパケットが，理論値 $100Mbps$ のネットワークを通過するのに必要な時間 (δt) を以下に示す式により求める．

$$\begin{aligned} TransferTime(\delta t) &= PacketSize \div Bandwidth \\ &= (1400[bytes] \times 8) \div 100[Mbit/sec] \\ &= 11200[bits] \div 100000000[bits/sec] \\ &= 0.000112[sec] \\ &= 112[\,\mu\ sec] \end{aligned} \tag{5.1}$$

5.1式より，DV/RTP パケット 1 パケットが通過するのに必要な時間 (理論値) は，112 マイクロ秒である．すなわち，DV/RTP パケット伝送におけるジッタ変動はマイクロ秒単位で現れる．よって，DV/RTP パケットの伝送時のネットワーク状態把握には，少なくともマイクロ秒単位での計測が行えなければならない．

5.2 設計概要

3.3節において，DVTS における輻輳制御手法の問題点について整理した．また，4章にて伝送特性を用いた新たな協調的輻輳制御手法について検討した．

2章で述べたインターネット上での映像・音声配信技術，3章にて述べた DVTS の特徴，4章にて述べた要件を満たす機能について考慮し，以下の 5 つの機能を本機構の満たすべき機能とする．

- 1 対 1 の通信ノード間で協調したジッタ及びパケットロス値の計測が行える
- ジッタの計測がマイクロ秒の精度にて行える
- 通信ノード間で通信状態情報の交換が行える
- ジッタ及びパケットロスに基づく協調的輻輳制御が行える
- 輻輳制御手法にのっとった DVTS のフレームレート制御が行える

図 5.1にシステム概要を示す．

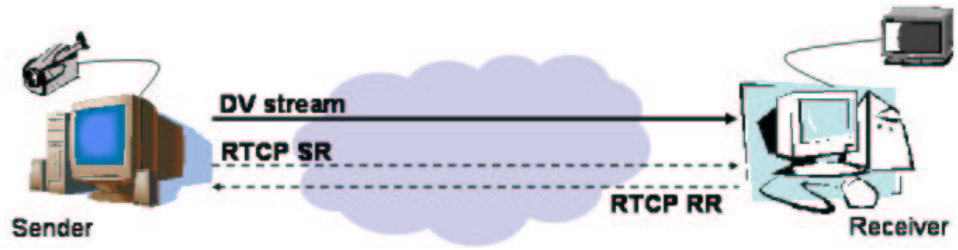

図 5.1: 全体概要図

本機構は，送信部および受信部により構成される．送信部および受信部は，それぞれ計算機および DV 機器を利用する．計算機と DV 機器間は，IEEE1394 インタフェースで接続を行い，データの送受信を行う．計算機間は，IP を用いてデータの送受信を行う．

また，送受信ノード間での情報の交換を行うために，RTCP(Real-time Transport Control Protocol)[A.2] を用いる．

5.2.1 全体構成

本機構を構成するモジュールを以下にまとめる.

- **送信部**
  - **送信者情報交換モジュール**
    RTCP SR(Sender Report) を用いた送信者情報の交換
  - **フレームレート 制御モジュール**
    受信者情報に基づいた映像レートの決定とセット
- **受信部**
  - **受信者情報交換モジュール**
    RTCP RR(Receiver Report) を用いた受信者情報の交換
  - **計測モジュール**
    送信者情報を基にジッタやフレームレートの値を計測

5.2.2 動作概要

本機構における各モジュールの関係及び動作概要を図 5.2に示す.

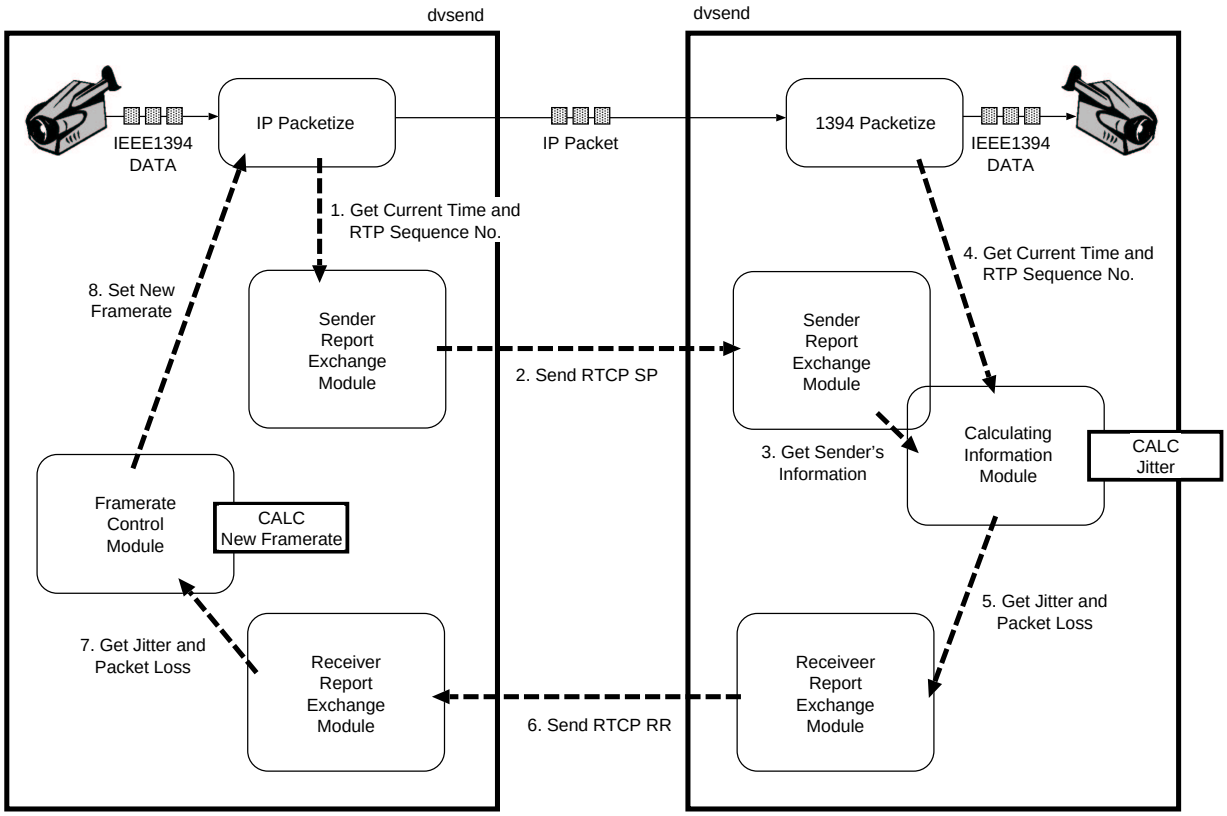

図 5.2: 各モジュール間関係概要

1. パケットの送信時間およびシーケンス番号を取得し，RTCP SR の必要なフィールドへセットする
2. RTCP SR パケットを送信する
3. RTCP SR パケットから送信者情報を取得する
4. パケットの受信時間及びシーケンス番号を取得する
5. ジッタ及びパケットロスの値を計測し，RTCP RR の必要なフィールドへセットする
6. RTCP RR パケットを送信する
7. ジッタ及びパケットロスの値を取得する
8. 新たなフレームレートの値を決定し，送信品質としてセットする

5.3 送信者情報交換モジュール

送信者情報交換モジュールは，受信部においてジッタおよびパケットロスの計測を行うために必要な送信者情報を受信ノードに対して送信する機能を有する．

5.3.1 情報の送信間隔

RFC1889 において，RTCP パケットによるバーストトラフィックを防止するために RTCP パケットの送信間隔は,「コントロールトラフィックは小さく，かつ既知のセッションバンド幅の一部程度に制御すべきである」とある．本機構では，RTCP による情報交換の重要性が高いことから，RTP パケットの送出頻度に比べて十分に少なく，かつなるべく短い送信間隔にて RTCP パケットを送出する．

本機構では，RTCP SR パケットの送信を，IEEE1394 アイソクロナスパケットを 1000 個受信し，さらに前パケットとの送信間隔が 1 秒以上空いている場合に行う．

DVTS が送出する DV/RTP パケット数と比較して，RTCP パケットが十分に少ない事を確認するため，DVTS が 1 秒間に送信するパケット数を 5.2式にて求める．送出レートは，間引きを行わないフルフレームレートとする．

$$\begin{aligned} PacketNumber(P) &= TrafficSize \div DV/RTPOnePacketSize \times Time \\ &= 32[Mbits/sec] \div (1400[bytes] \times 8) \times 1[sec] \\ &= 32000000[bits/sec] \div 11200[bits] \times 1[sec] \\ &\fallingdotseq 2857 \end{aligned} \tag{5.2}$$

計算の結果，DV/RTP パケットは 1 秒間に約 3000 パケット送出される．これと比較して，RTCP パケットは約 1 秒間に 1 度の割合で送出される．よって，RTP パケットの送出頻度と比較して十分に少ない．

5.3.2 送信時刻取得処理

DV/RTP パケット伝送におけるジッタ/遅延時間を計測するために，特定されたパケットが送信された絶対時刻を記録し，受信ノードに対して送信する．本処理では，DV/RTP パケットの送信時刻をマイクロ秒単位で取得し，そのパケットの RTP タイムスタンプとともに記録する．

5.3.3 送信者情報の送信

情報の交換は，RTCP パケットのうち，SR(Sender Report) パケットに必要な情報を格納し送信することで行う．図 5.3に情報を格納するブロックを示す．64 ビットの “NTP Timestamp” ブロックに取得した送信時刻情報を 32 ビットごとに格納し，32 ビットの “RTP Timestamp” ブロックに取得した RTP タイムスタンプを格納する．情報が格納されたパケットは，受信ノードの RTCP 受信用ポートに対して送信される．

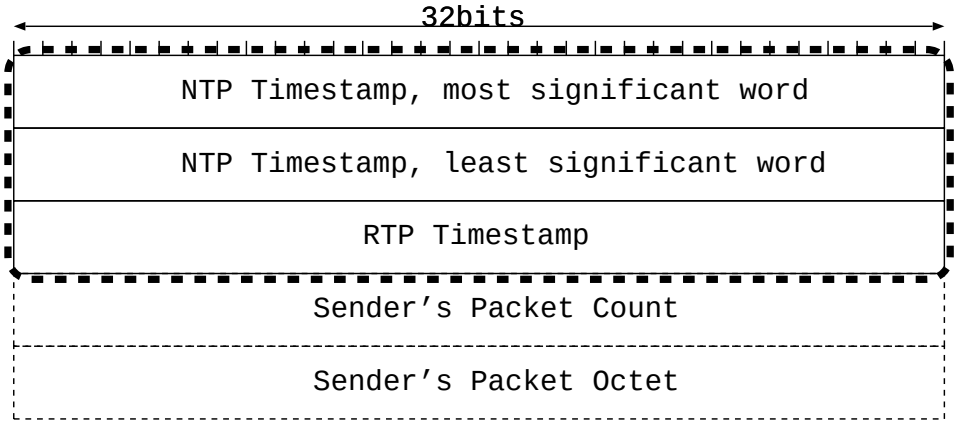

図 5.3: RTCP SR Sender Info Block

5.4 受信者情報交換モジュール

受信者情報交換モジュールは，受信した送信者情報を基にパケットロス数ならびにパケット到達時間の揺らぎの計測を行い，その情報を送信ノードに対してフィードバックを行う機能を有する．

5.4.1 情報の送信間隔

送信者情報交換モジュール同様，RTCP を用いる受信者情報交換モジュールでは，RTCP パケットによるバーストトラフィックの発生を防止する．

RTCP SR パケットを受信するごとに呼び出すことで，送信者と同様の間隔でパケットの送信を行う．

5.4.2 受信時刻処理

DV/RTP パケット伝送におけるジッタ/遅延時間を計測するために，特定パケットが到達した絶対時刻を記録する．受信した DV/RTP パケットのうち，RTCP SR パケット中の “RTP Timestamp” ブロックに記録された RTP タイムスタンプを持つパケットを受信した時刻をマイクロ秒単位で取得する．

5.4.3 ジッタ計測処理

RFC1889 では，ジッタの計測は 5.4式のように行われる．ジッタの値 J は，パケットの送信時と受信時の時間の差 D(Delay) を求め，その値の平均偏差により求められる．この手法を，RFC 準拠手法と呼ぶ．

$$\begin{aligned} D(i,j) &= (R_j - R_i) - (S_j - S_i) \\ &= (R_j - S_j) - (R_i - S_i) \end{aligned} \tag{5.3}$$

$$J = J + \frac{(|D(i-1,i)| - J)}{16} \tag{5.4}$$

しかし，RFC 準拠手法には以下のような問題がある．

- ジッタの値の安定に時間がかかることがある
- 平均偏差として求められるため，負の情報 (転送時間が早まった) を求められない

このため，本機構では 5.5式のように，単純に到達時間の差分を求めることでジッタの値 J を取得する．また，時間情報に正負の情報を付加することで，純粋な伝送時間の変化も取得する．この手法を時間差分取得手法と呼ぶ．

$$J = (R_j - S_j) - (R_i - S_i) \tag{5.5}$$

5.4.4 受信者情報の送信

情報の交換は，RTCP パケットのうち，RR(Receiver Report) パケットに必要な情報を格納し送信することで行う．図 5.4に情報を格納するブロックを示す．8 ビットの “Fraction Lost” ブロックに前 RTCP SR/RR パケット送受信後に失われた RTP データパケット数 (喪失パケット数を受け取るはずのパケット数で割ったもの)，24 ビットの “Cumulative Number of Packet Lost” ブロックに DV/RTP パケット送信中に失われたパケット数，32 ビットの “Interarirval Jitter” ブロックに本モジュールによって計測された DV/RTP パケット伝送時のジッタをそれ

ぞれ格納する．情報が格納されたパケットは，送信ノードの RTCP 受信用ポートに対して送信される

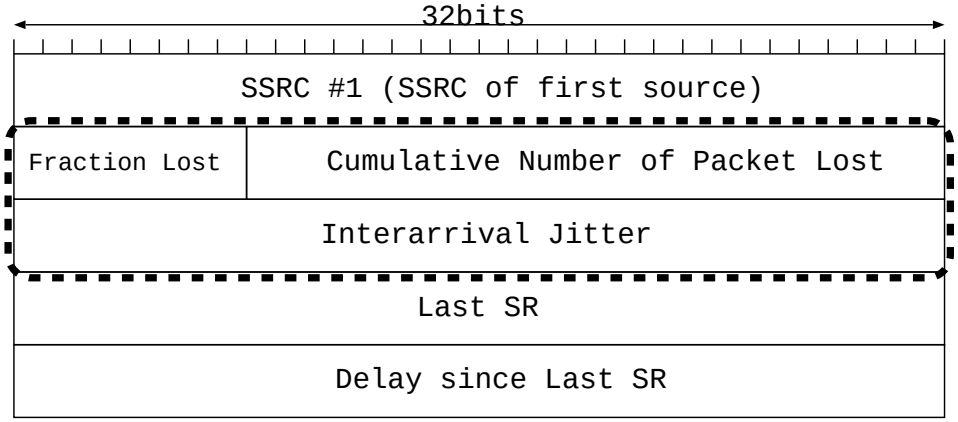

図 5.4: RTCP RR Report Block

ジッタ値の符号化

計測されるジッタは，5.5式の通り，正と負の値をとる．しかし，受信者情報送信時に構造体に格納される情報は，RFC では符号無し整数型を用いるよう規定されている．このため，ジッタ値を RFC に準拠する形で構造体に格納した場合，正の値も負の値も同じ値となる．本機構では，ジッタの値を構造体に格納する際に，1 ビット分左に演算を行い，最終ビットに正負の情報を格納できるようにする．最終ビットが 1 の場合，本機構はジッタが負の値が取ると認識し処理を行う．図 5.5にジッタが構造体に格納される際の配置を示す．

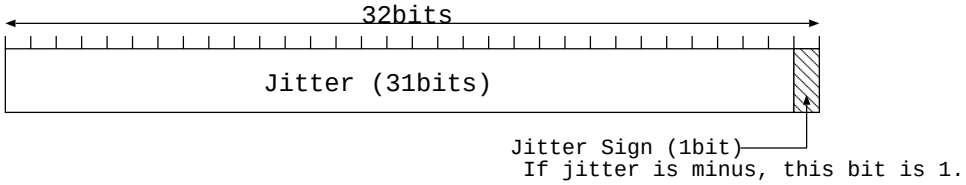

図 5.5: RTCP RR へのジッタ値の格納

5.5 フレームレート制御モジュール

フレームレート制御モジュールは，5.3節に示した情報交換モジュールによって取得した受信者の情報を基に，送信を行う際のフレームレートを動的に調整を行う機能を有する．本モジュールは，送信部に実装される．

送信レート制御は，送信フレームの間引き率の値を動的に変化させることで行う．フレーム間引き率の値は，$\frac{1}{2}$ であれば 2 というように，値が大きくなればなるほど間引き率が増加し，実際に必要な帯域幅は減少する．本モジュールで動的に変化する値の幅は，1 から 30 をとる．

5.5.1 変更限度値

間引き率の値を動的に変化させる際，ネットワーク状態的に問題のないレートとパケットロスを発生させてしまうレートの間で間引き率が常に上下する “Swing of Picture Frame Rate”[28] (図 5.6) が発生すると言われている．この発生を防止するため，本モジュールでは問題のあったレートを一時的に記憶し，それ以上値が変化させないようにする．この一時的に設定される値を，変更限度値と呼び，間引き率変更の際に動的に変更させる．

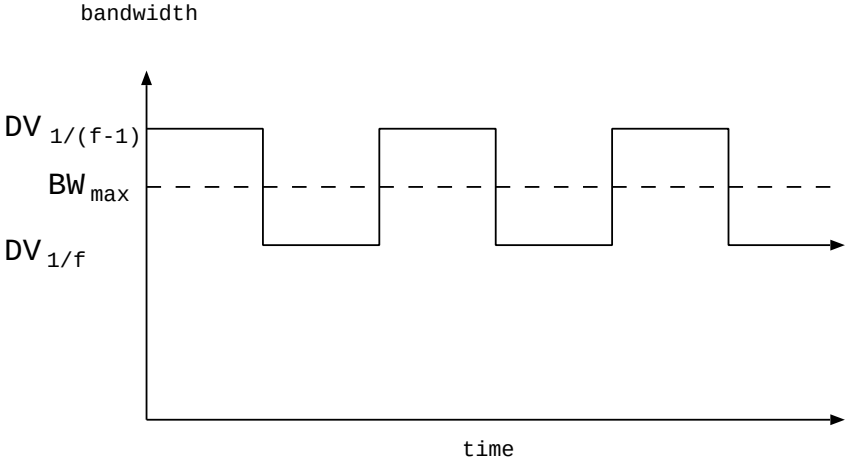

図 5.6: Swing of Picture Frame Rate

5.5.2 間引き率変更のアルゴリズム

本モジュールでは，次に挙げるアルゴリズムを利用し，送信時のレート制御を行う．制御には，変更限度値 $rate\_limit$ が設けられており，不必要なレートの変動を起こさないよう処理を行う．処理の流れは，図 5.7に示す．

- ジッタによる間引き率の加算処理
- ジッタによる間引き率の減算処理
- パケットロスによる間引き率の加算処理
- ジッタ及びパケットロスによる間引き率の減算処理

ジッタによる間引き率の加算

ジッタ値の急激な増加は，DV/RTP パケットが受信ノードに到達する時間が一時的に増加したことを示す．この急激な増加によりネットワーク中に何らかの問題が発生し，輻輳状態が起こったと認識する．

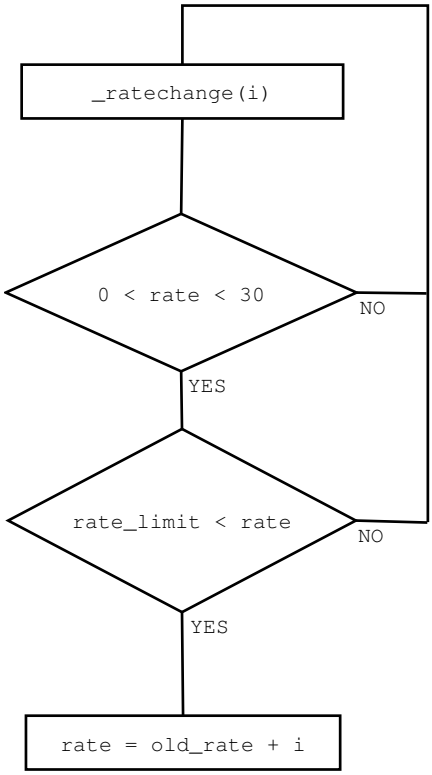

図 5.7: フレームレート制御処理

本モジュールでは，図 5.8に示すフローに従って間引き率を増加させる．処理には，受信した RTCP RR パケット 5 回分のジッタの平均値を用いる．変更閾値である Jth_max を超えた場合に間引き率を 1 増加させる．

ジッタによる間引き率の減算

ジッタの値が急激に増加した後に，ジッタ値の急激な減少が発生することは，ネットワークが輻輳状態からある程度回復したと認識する．

本モジュールでは，図 5.9に示すフローに従って間引き率を減少させる．処理には，受信した RTCP RR パケット 5 回分のジッタの平均値を用いる．変更閾値である Jth_min を超えた場合に間引き率を 1 減少させる．

パケットロスによる間引き率の加算

ネットワーク状態の変化が，ジッタの急激な変動として現れない場合，パケットロスの値によってフレームレートの制御を行う．本モジュールでは，図 5.10に示すフローに従って間引き率を増加させる．変更閾値である Lth_a を超えた場合に間引き率を 1 増加させる．

また，パケットロスがひどい場合，現在のフレーム間引き率ではネットワーク性能が追いつかない状態である．そこで，変更閾値 Lth_b を超えた場合，変更限度値 $rate\_limit$ を 1 増加させ，さらに間引き率も 1 増加させる．

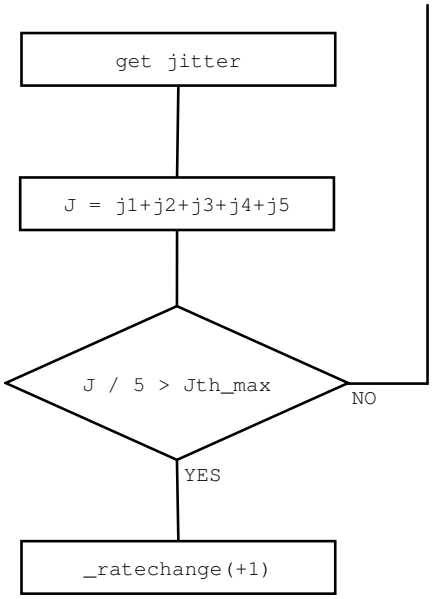

図 5.8: ジッタによる間引き率加算処理

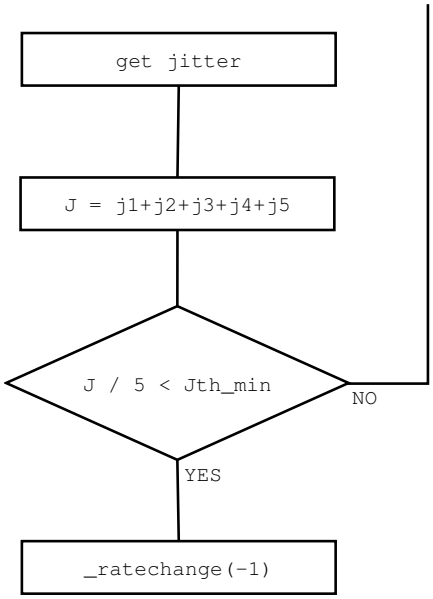

図 5.9: ジッタによる間引き率減算処理

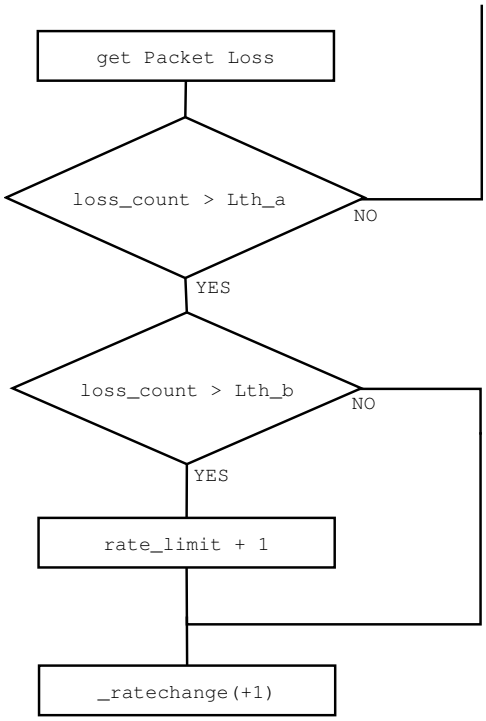

図 5.10: パケットロスによる間引き率変更処理

ジッタ及びパケットロスによる間引き率の減算

間引き率の減算処理においても，ジッタの急激な変動が現れない場合，パケットロスの値によってその制御を行う．主にパケットロスが連続して 0 を示した回数を利用するが，必要に応

じてジッタの値も利用する．本モジュールでは，図 5.11に示すフローに従って間引き率を減少させる．

パケットロスが連続して 0 を示した回数が変更閾値 Lth_c を超え，さらに RTCP RR パケットにより取得した 1 回分のジッタ値が jth を超えた場合，間引き率を 1 減少させる．また，パケットロスが連続して 0 を示した回数が変更閾値 Lth_d を超えた場合，ジッタの値にかかわらず，間引き率を 1 減少させ，さらに変更限度値 $rate\_limit$ を 1 減少させる．

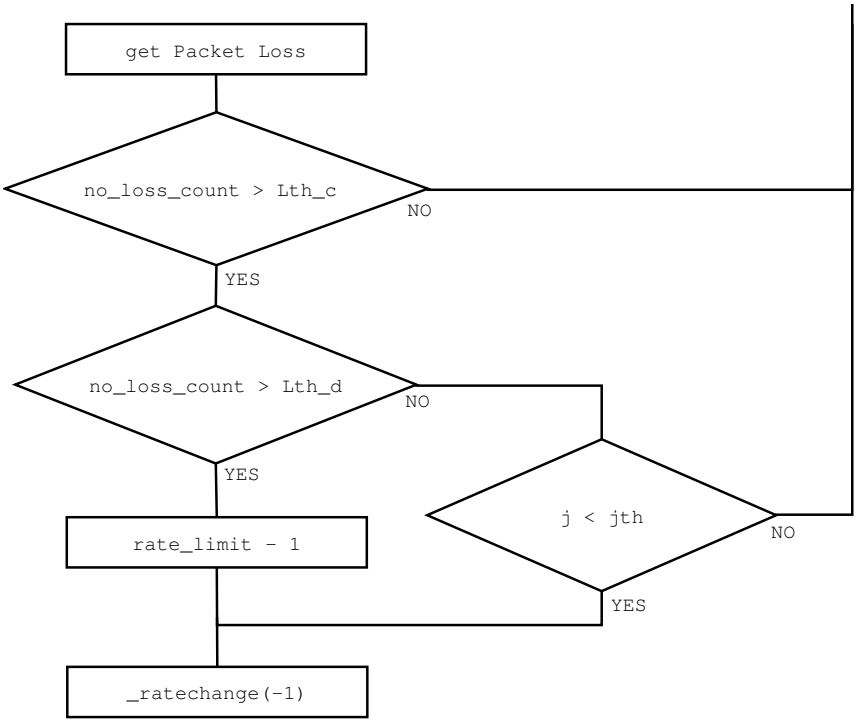

図 5.11: ジッタ及びパケットロスによる間引き率変更処理

5.6 まとめ：適応型映像・音声配信機構の設計

本章では，伝送特性を用いた輻輳制御手法を DVTS に対して適用するための設計要件を挙げ，本機構が以下の 4 つのモジュールで構成されることを述べ，それぞれの詳細な設計について述べた．

- 送信者情報交換モジュール
- フレームレート制御モジュール
- 受信者情報交換モジュール
- 計測モジュール

6章では，本設計に基づいた適応型映像・音声配信機構の実装について述べる．

第6章　適応型映像・音声配信機構の実装

本章では，5章にて行った設計に基づいて実装を行った適応型映像･音声配信機構 DFCS(Dynamic Framerate Control System) について述べる．6.1節では，実装概要と実装を行った環境について述べる．6.2節以降では，本機構の各部分における詳細な実装について述べる．

6.1　実装概要

インターネット上で DV ストリームの送受信を行う DVTS を対象に，伝送特性による協調的輻輳制御機能を有する適応型映像・音声配信機構 DFCS を設計し，その設計を基づき実装を行った．

DFCS では，送信部と受信部が協調し，DV パケット送信にかかる伝送時間の揺らぎ（ジッタ）･パケットロスの値を計算し，その情報を基に送信部が受信部に対して送信を行う DV ストリームの映像フレーム間引き率の値を動的に制御する．DV ストリームの送出映像フレーム間引き率を調整することにより，使用する帯域を変化させる．

ジッタの計算に当たって，時間情報の取得には gettimeofday 関数を用い，取得した値はマイクロ秒単位で double 型変数として扱う．マイクロ秒単位での値取得を行うため，使用する OS は gettimeofday 関数によりマイクロ秒単位での値取得が行えるものでなければならない．

本機構の実装を行ったソフトウェア環境を表 6.1に示す．

表 6.1: 実装ソフトウェア環境

OS	FreeBSD 4.7 RELEASE
プログラミング言語	C 言語
コンパイラ	gcc 2.95.4

本機構の実装を行った機器環境を表 6.2に示す．

表 6.2: 実装ハードウェア環境

CPU	Pentium III 800MHz
メモリ	384MB
ハードディスク	40GB
NIC	100Base-TX, 1000Base-SX
DV 機器 (1)	Panasonic DV カメラ MV-EX21
DV 機器 (2)	Sony メディアコンバータ DVMC-DA1
DV 機器 (3)	Sony メディアコンバータ DVMC-DA2
DV 機器 (4)	Canopus メディアコンバータ ADVC-100

6.2 関数一覧と処理の流れ

6.2.1 送信部

送信部における DFCS に関係する関数とその処理の流れを図 6.1に示す．送信部では，パケット送出時刻の取得処理，RTCP RR パケットによる受信者情報によってフレーム間引き率の変更処理を行う．

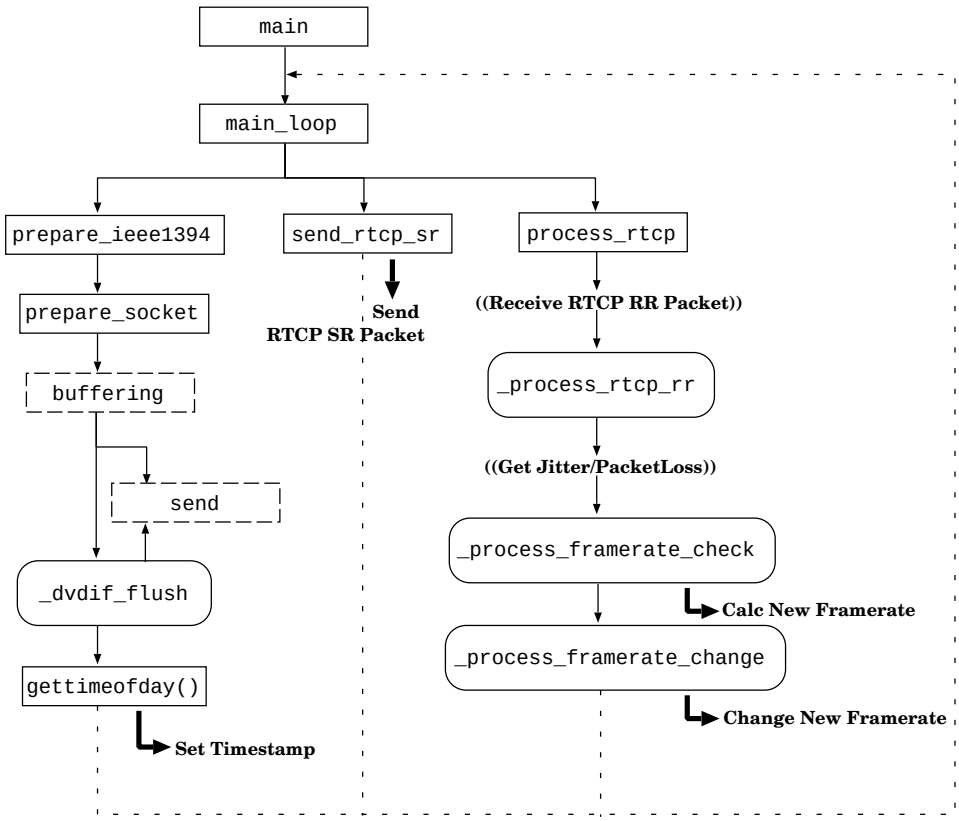

図 6.1: 送信部における処理の流れ

6.2.2 受信部

受信部における DFCS に関係する関数とその処理の流れを図 6.2に示す．受信部では，パケット受信時刻の取得処理，ジッタ・パケットロス値の計測処理，送信部への情報のフィードバック処理を行う．

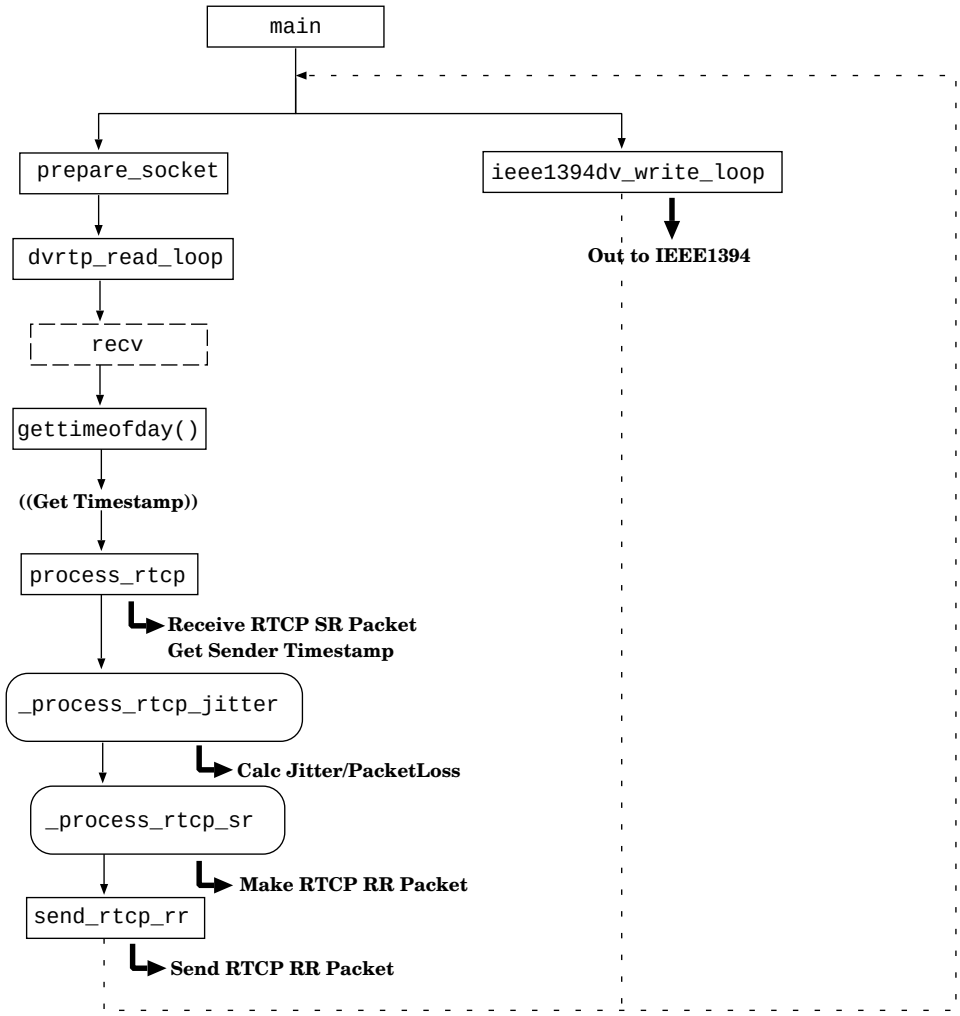

図 6.2: 受信部における処理の流れ

6.3 送信部

6.3.1 *dvsend_param* 構造体

図 6.3に，送信部 (dvsend) が用いる *dvsend_param* 構造体のうち，本実装のために独自に定義したものを示す.

```
struct dvsend_param {
    (省略)
    /* NTP timestamp uses for RTCP SR */
    u_int32_t ntp_sec;
    u_int32_t ntp_frac;

    /* DFCS enabled or disabled */
    int enable_dfcs;

    /* No Packet Loss Counter */
    int noloss_count;

    /* Previous Jitter Count */
    int jitter_prev;
    int jitter_p;
    int jitter_pp;
    int jitter_ppp;

    /* Over Jitter Counter */
    int oj_count;
    int sj_count;

    /* Frame Drop Rate Limit */
    int frame_limit;
};
```

図 6.3: *dvsend_param* 構造体

dvsend_param 構造体には，送信部が全体として必要とする変数が格納される．本実装では，取得した時間情報，フレーム間引き率の決定に必要な値，フレーム間引き率の変更限度値などの情報を新たに格納できるようにした.

6.3.2 DV/RTP パケット送出時刻取得

図 6.4に，送信部での DV/RTP パケットの送出時刻取得を行う実装を示す．時間情報の取得は，*gettimeofday* 関数を用いてマイクロ秒単位で行う．

パケット送出時刻は，実際に RTP パケットとして受信部に対して送信される瞬間に取得しなければならない．この実装は，DIF ブロックの送出を行う *_dvdif_flush* 関数内に実装されている．また，*_dvdif_flush* 関数が実行されると RTCP SR パケットに格納される RTP タイムスタンプが変更される．

```
static void
_dvdif_flush(struct dvsend_param *dvsend_param)
{
     (省略)
     /* NTP timestamp for RTCP SR */
     if (gettimeofday(&rtp_now, NULL) < 0) {
       printf("gettimeofday failed\n");
       return;
     }

     /* Timestames(NTP/RTP) use for RTCP SR */
     dvsend_param->ntp_sec = rtp_now.tv_sec;
     dvsend_param->ntp_frac = rtp_now.tv_usec;
     (省略)
}
```

図 6.4: *_dvdif_flush* 関数での時刻取得

6.3.3 ジッタ値の取得処理

受信部において正負の値を取るジッタ値をビット演算により RTCP RR パケットに格納した．この情報を取得するための実装を図 6.5に示す．

本実装では，取得した値の最終ビットを確認した上で 1 ビット分演算を行い符号無し整数型のジッタを取得する．最終ビットに 1 が入っていた場合にのみ，取得したジッタ値にマイナスを掛け合わせ，負の値とする．

6.3.4 フレーム間引き率の決定

図 6.6から 6.10 に，送信部でのフレームレートの制御を行う際に必要な間引き率の値を決定する実装を示す．これらの実装は，*_process_framerate_check* 関数として実装されている．本実装では，5.5.2に示すアルゴリズムに基づき間引き率の値の変動処理を行う．

```
static int
_process_rtcp_rr (struct dvsend_param *dvsend_param,
                  char *buf, int n,
                  struct sockaddr *addr, int length)
{
     (省略)
       /* check jitter signed from last 1bit */
       jitter_minus = jitter & 0x00000001;
       jitter >>= 1;
       if(jitter_minus){ jitter = -jitter; }
  (省略)
}
```

図 6.5: *_process_rtcp_rr* 関数でのジッタ値のビット演算

パケットロスによる決定

図 6.6は，パケットロスの値によって間引き率値の減算処理およびフレーム間引き率の変更限度値の決定を行う実装である．本実装は，5.5.2項にて示したアルゴリズムに基づき実装されており，変更閾値 Lth_d を 10 と設定している．

```
     /* limit_rate change by packet loss */
     if (dvsend_param->noloss_count > 10) {
       if (dvsend_param->frame_limit > 1) {
         dvsend_param->frame_limit--;
       }
       /* frame drop rate change by packet loss */
       _process_framerate_change(dvsend_param, -1);
       return(1);
     }
```

図 6.6: パケットロスによる決定 1

図 6.7は，パケットロスの値によって間引き率値の加算処理及びフレーム間引き率の変更限度値の決定を行う実装である．本実装は，5.5.2項にて示したアルゴリズムに基づき実装されており，変更閾値 Lth_a を 50，Lth_b を 100 と設定している．

ジッタによる決定

図 6.8は，ジッタの値によって間引き率値の加算処理を行う実装である．変数 $jitter\_ave$ に過去 5 回分の RTCP RR パケットに格納されたジッタの値の平均値が入る．本実装は，5.5.2項にて示したアルゴリズムに基づき実装されており，変更閾値 Jth_max を 4000 と設定している．

```
/* limit_rate change by packet loss */
if (pkt_loss_sum > 50) {
  if (pkt_loss_sum > 100 && dvsend_param->frame_limit <= 30){
    dvsend_param->frame_limit++;
  }
  /* frame drop rate change by packet loss */
  _process_framerate_change(dvsend_param, 1);
  return(1);
}
```

図 6.7: パケットロスによる決定 2

また，アルゴリズムの精度を高めるため，この処理が呼び出された回数を格納するカウンタ変数を設定し，一定の回数呼びされた時にのみ間引き率の値を変動させるように実装している．

```
/* frame drop rate change by jitter */
if(jitter_ave / 5 > 4000){
  dvsend_param->oj_count++;
  if(dvsend_param->oj_count > 3){
    _process_framerate_change(dvsend_param, 1);
    dvsend_param->oj_count = 0;
    return(1);
  }
} else if(dvsend_param->oj_count) {
  printf("counter reset!\n");
  dvsend_param->oj_count = 0;
}
```

図 6.8: ジッタによる決定 1

図 6.9は，ジッタの値によって間引き率値の減算処理を行う実装である．変数 $jitter\_ave$ に過去 5 回分の RTCP RR パケットに格納されたジッタの値の平均値が入る．本実装は，5.5.2項にて示したアルゴリズムに基づき実装されており，変更閾値 Jth_min を -2400 と設定している．

また，本実装においても，間引き率の変動を抑制するカウンタ変数を設定している．

ジッタ及びパケットロスによる決定

図 6.10は，ジッタ及びパケットロスの値によって間引き率値の減算処理を行う実装である．本実装は，5.5.2項にて示したアルゴリズムに基づき実装されており，前者の実装では変更閾値 Lth_c を 3，変更閾値 jth を -1500，後者の実装では変更閾値 Lth_c を 1，変更閾値 jth を -5000 とそれぞれ設定している．

```
/* frame drop rate change by jitter */
if(jitter_ave / 5 < -2400){
  dvsend_param->sj_count++;
  if(dvsend_param->sj_count > 3){
    _process_framerate_change(dvsend_param, -1);
    dvsend_param->sj_count = 0;
    return(1);
  }
} else if(dvsend_param->sj_count) {
  printf("counter reset!\n");
  dvsend_param->sj_count = 0;
}
```

図 6.9: ジッタによる決定 2

```
/* frame drop rate change by jitter and packet loss */
if (jitter < -1500 && dvsend_param->noloss_count > 3) {
  _process_framerate_change(dvsend_param, -1);
  return(1);
}

/* frame drop rate change by jitter and packet loss */
if (jitter < -5000 && dvsend_param->noloss_count > 1){
  _process_framerate_change(dvsend_param, -1);
  return(1);
}
```

図 6.10: ジッタ及びパケットロスによる決定

6.4 受信部

6.4.1 *dvrecv_param* 構造体

図 6.11に，受信部 (dvrecv) が用いる *dvrecv_param* 構造体のうち，本実装のために独自に定義したものを示す.

dvrecv_param 構造体には，受信部が全体として必要とする変数が格納される．本実装では，取得した時間情報，ジッタ・パケットロスの計測に必要な値などの情報を新たに格納できるようにした.

```
struct dvrecv_param {
    (省略)
    /* NTP time - RTCP SR */
    u_int32_t sr_ntp_sec;
    u_int32_t sr_ntp_frac;

    /* NTP timestamp for RTCP RR */
    u_int32_t rtcp_now_ntpsec;
    u_int32_t rtcp_now_ntpusec;

    /* previous transit time of DV/RTP packet */
    double jitter_prev_transit;

    /* delay time of DV/RTP packet */
    double jitter_delay;

    /* jitter count in _process_rtcp_jitter */
    double jitter_jitter;

    /* Interarrival Jitter for RTCP RR */
    u_int32_t rr_jitter;
};
```

図 6.11: *dvrecv_param* 構造体

6.4.2 DV/RTP パケット受信時刻取得

図 6.12に，受信部での DV/RTP パケットの受信時刻取得を行う実装を示す．時間情報の取得は，*gettimeofday* 関数を用いてマイクロ秒単位で行う．

パケット受信時刻は，実際に受信部が RTP パケットを受信する瞬間に取得しなければならない．この実装は，RTP パケットの受信を行う *dvrtp_read_loop* 関数内に実装されている．6.3.2項より，送出時刻が取得される瞬間に RTP タイムスタンプが変更されるため，受信部においても RTP タイムスタンプが変化した瞬間に時刻の取得を行う．

6.4.3 ジッタ値計測

図 6.13に，受信部でのジッタ値の計測を行う実装を示す．この実装は，*_process_rtcp_jitter* 関数として実装されている．

RTCP SR パケット及び受信部でのパケット受信時刻の取得によって得られた時間情報によりパケット到達時間の差分を求め計測を行う．計測によって得られたジッタ値は，図 6.11に示した *dvrecv_param* 構造体に格納し，RTCP RR パケットが送信される際に参照される．

```
void
dvrtp_read_loop (struct dvrecv_param *dvrecv_param)
{
     (省略)
       /* (if rtp timestamp is change) get current time */
       if (rtphdr->ts != rtp_ts_prev) {
         if (gettimeofday(&rtp_now, NULL) < 0) {
           printf("gettimeofday failed\n");
           return;
         }

         /* get NTP timestamp for RTCP RR */
         dvrecv_param->rtcp_now_ntpsec = rtp_now.tv_sec;
         dvrecv_param->rtcp_now_ntpusec = rtp_now.tv_usec;
       }
     (省略)
}
```

図 6.12: *dvrtp_read_loop* 関数での時刻取得

```
  /* calc transit time of DV/RTP packet */
  transit_sec =
      dvrecv_param->rtcp_now_ntpsec - dvrecv_param->sr_ntp_sec;
  transit_usec =
      dvrecv_param->rtcp_now_ntpusec - dvrecv_param->sr_ntp_frac;
  transit =
      (double)(transit_sec + (double)(transit_usec / 1000000.0));

  /* calc delay time of DV/RTP packet */
  delay = transit - dvrecv_param->jitter_prev_transit;
  dvrecv_param->jitter_delay = delay;

  /* set previous transit time of DV/RTP packet */
  dvrecv_param->jitter_prev_transit = transit;

  /* jitter implementation based on time-diff-mesurement */
  dvrecv_param->jitter_jitter = dvrecv_param->jitter_delay;
```

図 6.13: *process_rtcp_jitter* 関数でのジッタ値計測

ジッタ値の格納

ジッタ値の格納は，5.4.4節の設計に基づき，符号無し整数型の RTCP RR パケットの該当ブロックに正負の値の格納ができるようビット演算を行う．その実装を，図 6.14に示す．

ジッタが負の値を取る場合，*jitter_minus* 変数に 1 が入る．その後，ジッタ値を 1 ビット分演算を行い，最終ビットに値を格納できるようにする．*jitter_minus* の値により，最終ビットを 1 にするか決定する．

```
/* check jitter sign */
if (dvrecv_param->jitter_jitter < 0) {
  dvrecv_param->jitter_jitter = -dvrecv_param->jitter_jitter;
  jitter_minus++;
}

/* jitter change from double val. to int val. */
for_rr_jitter = dvrecv_param->jitter_jitter * 1000000.0;
for_rr_jitter <<= 1;

/* (if jitter is minus) change 1bit for notice minus */
if(jitter_minus){ for_rr_jitter |= 0x00000001; }
```

図 6.14: *_process_rtcp_jitter* 関数でのジッタ値のビット演算

6.5 まとめ：適応型映像・音声配信機構の実装

本章では，5章の設計に基づき実装を行った DFCS の概要ならびに以下に示す各部の動作の詳細について述べた．

- 送信部
  - dvsend param 構造体
  - DV/RTP パケット送出時刻取得
  - ジッタ値の取得処理
  - フレーム間引き率の決定
- 受信部
  - dvrecv param 構造体
  - DV/RTP パケット受信時刻取得
  - ジッタ値計測

7章では，本機構の動作について検証し，本手法の有用性について評価する．

第7章　評価

本章では，本論文にて設計，実装を行った DFCS の評価について述べる．7.2節で本機構が実現した機能，7.3節でフレームレート制御機能の評価についてそれぞれ述べる．また，7.4節では，本評価のまとめを述べる．

7.1　評価概要

本評価は，本研究の提案する輻輳制御手法の有効性の検証を目的とする．定性評価ならびに定量評価は，以下の項目について行う．

- **定性評価**：設計に基づき実装した DFCS の実現した機能
- **定量評価**：DFCS によるフレームレートコントロールの実現 (4 つの評価実験と結果考察)

7.2　本機構の実現した機能

本機構は，5.2節にて述べた満たすべき 5 つの機能を満たし，DVTS を用いた映像・音声配信における伝送特性に応じた動的な映像フレームレート制御機能を実現した．

- 1 対 1 の通信ノード間で協調したジッタ及びパケットロス値計測
- マイクロ秒の精度でのジッタ計測
- 通信ノード間で通信状態情報の交換
- ジッタ及びパケットロスに基づく協調的輻輳制御手法の確立
- 輻輳制御手法にのっとった DVTS のフレームレート制御

また，動作確認を行った機器の組み合わせを，表 7.1ならびに表 7.2に示す．

表 7.1: 動作を確認した組み合わせ (DVcamera-MediaConverter)

	種類	メーカ	機種
送信側	DV カメラ	Panasonic	MV-EX21
受信側	メディアコンバータ	Canopus	ADVC-100

表 7.2: 動作を確認した組み合わせ (MediaConverter-MediaConverter)

	種類	メーカ	機種
送信側	メディアコンバータ	Sony	DVMC-DA1
受信側	メディアコンバータ	Sony	DVMC-DA2

7.3 フレームレートコントロールの実現

DFCSを搭載したDVTSの挙動を確認し，フレームレートコントロールが実現できたかどうか評価する．本評価は，以下に挙げる実験を行い，フレーム間引き率の変動やネットワークにかかるトラフィック量の変化を確認することで行う．

- **帯域幅の不足**
  中継ネットワークの帯域幅が減少し，フルレートでのDVストリーム送信が困難になった際の本機構の挙動の確認を行う
- **遅延時間の変化**
  パケット伝送にかかる時間を増加させ，片方向遅延時間が大きく増加した際の本機構の挙動の確認を行う
- **DVストリーム2本での協調動作**
  DFCSを搭載したDVTS2組が帯域制限を行ったネットワーク上に同時にトラフィックをかけた際の本機構の挙動の確認を行う
- **DVストリームとTCPストリームの協調動作**
  DVCSを搭載したDVTSによるトラフィックとTCPによるトラフィックが帯域制限を行ったネットワーク上に同時にトラフィックをかけた際の本機構の挙動の確認を行う

7.3.1 実験1：帯域幅の不足

実験1では、DVストリーム送信中に通信経路上の帯域幅が変化した際の本機構の挙動を確認する．

実験環境

実験1は，図7.1に示したように送信ノード及び受信ノードの2台のPCを用いて行った．2台のPC間には，PCで作成したルータを設置し，片一方のインタフェースにdummynetを設定し，擬似的に帯域幅を制限した．

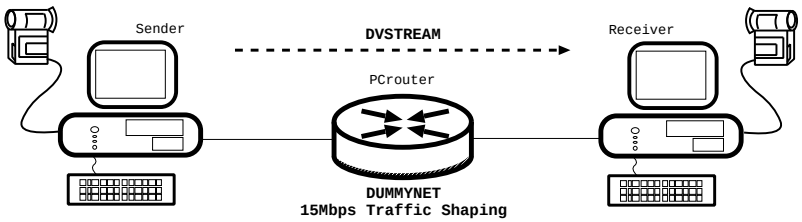

図 7.1: 実験1のネットワークトポロジ

実験は，図7.2に示すタイミングで行った．帯域幅の制限を行った時間は，図中2-3間及び4-5間となる．

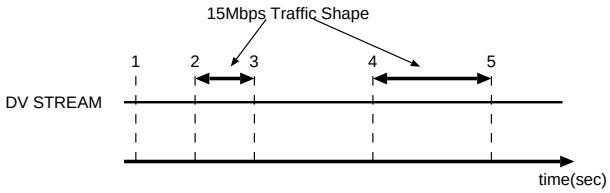

図 7.2: 実験 1 のタイミング

1. DV 送信ノードにて DV 送信を開始
2. 15Mbps に帯域幅を制限
3. 帯域幅の制限を解除
4. 再度， 15Mbps に帯域幅を制限
5. 帯域幅の制限を解除
6. DV 送信ノードにて DV 送信を終了

実験結果

送信ノードにおける映像間引き率ならびにトラフィック量の推移を図 7.3 に示す.

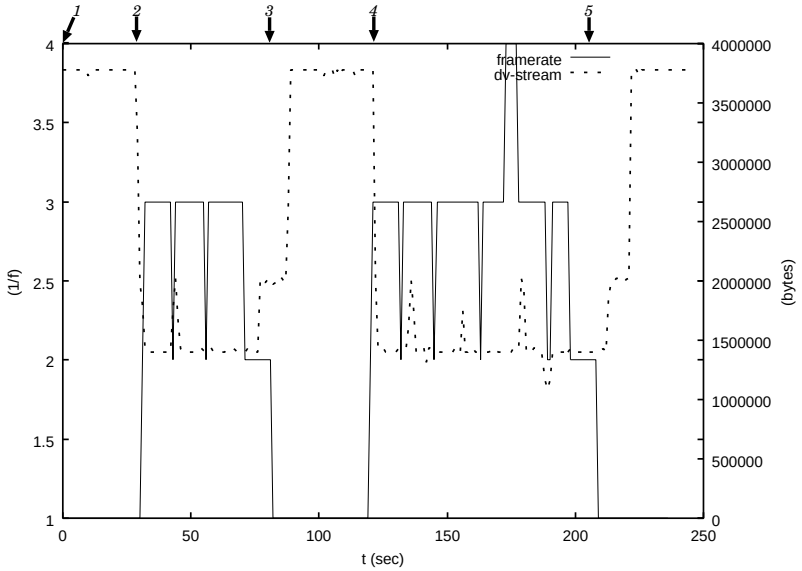

図 7.3: 実験 1 の結果

帯域幅を制限した 2，4 の段階から，本機構は映像間引き率を $\frac{1}{1}$ から $\frac{1}{2}$，さらに $\frac{1}{2}$ から $\frac{1}{3}$ に増加させている．また，帯域幅の制限を解除した 3，5 の段階から，映像間引き率を減少させている．それらに伴い，トラフィック量も増減していることが確認できる．

この実験結果より，利用可能な帯域幅の増減に対して本機構が適応しフレームレートの調整を行うことができるということが確認できた.

7.3.2 実験 2：遅延時間の変化

実験 2では，DV ストリーム送信中に通信経路上の遅延時間が変化した際の本機構の挙動を確認する．

実験環境

実験 2は，実験 1と同様に図 7.4に示したように送信ノード及び受信ノードの 2台の PC を用いて行った．PC ルータでは dummynet を用いて擬似的に遅延を発生させた．

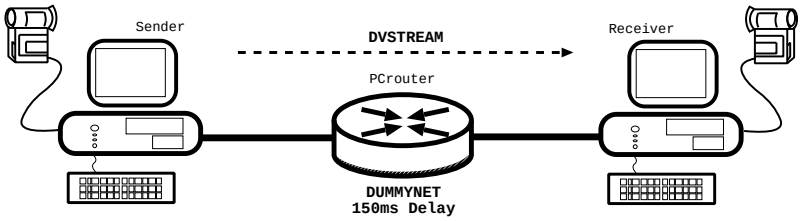

図 7.4: 実験 2のネットワークトポロジ

実験は，図 7.5に示すタイミングで行った．遅延時間を変化させた時間は，図中 2-3 間及び 4-5 間となる．

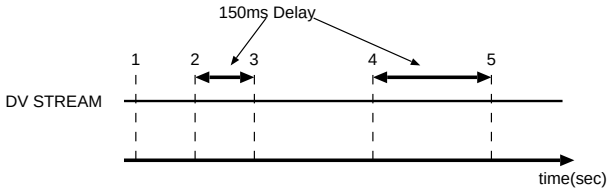

図 7.5: 実験 2のタイミング

1. DV 送信ノードにて DV 送信を開始
2. 遅延時間を 150ms に設定
3. 遅延時間の設定を解除
4. 再度，遅延時間を 150ms に設定
5. 遅延時間の設定を解除
6. DV 送信ノードにて DV 送信を終了

実験結果

送信ノードにおける映像間引き率ならびにトラフィック量の推移を図 7.6 に示す．

帯域幅を制限した 2の段階で，遅延時間の増大に伴うジッタの増加を検知し，映像間引き率を $\frac{1}{1}$ から $\frac{1}{2}$ に変更している．しかし，遅延時間のみの変動で，ネットワークの帯域幅は不足していないため，映像間引き率は $\frac{1}{2}$ から $\frac{1}{1}$ に変更されている．

また，この実験では，遅延時間を $150ms$ から $0ms$ に戻した際に，映像間引き率の変動が発生している．これは，本機構が遅延時間の変化をネットワークの変化として誤認識してしまっているため発生していると予想される．

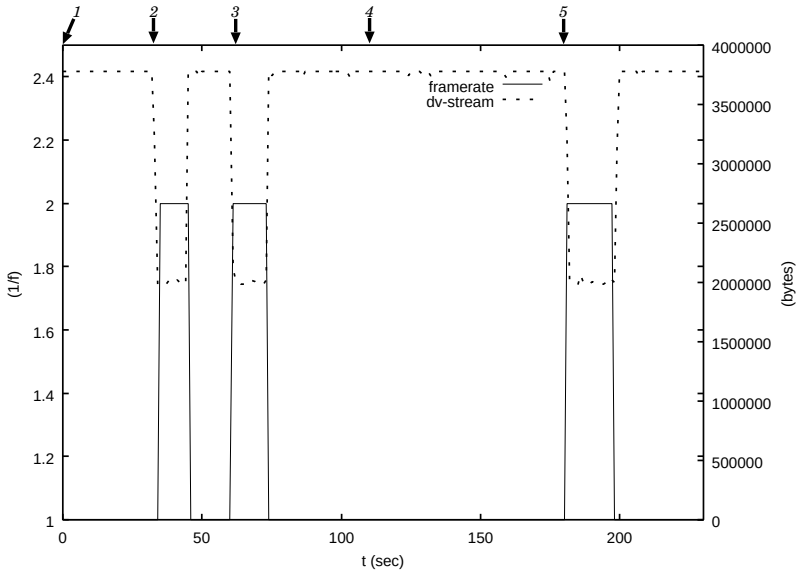

図 7.6: 実験 2 の結果

7.3.3 実験 3：DV ストリーム 2 本での協調動作

実験 3 では，同一通信経路上に本機構を搭載した DVTS が 2 台存在した場合の相互の挙動を確認する．

実験環境

実験 3 は，図 7.7に示したように，2 台の DV 送信ノードと 2 台の DV 受信ノードの 4 台の PC を用いて行った．4 台の PC 間には，PC で作成したルータを設置し，片一方のインタフェースに dummynet を設定し，擬似的に 35Mbps に帯域幅を制限した．

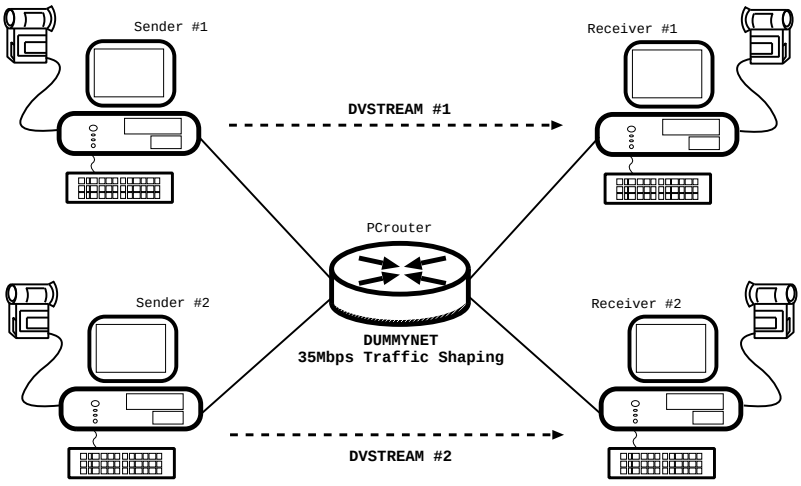

図 7.7: 実験 3 のネットワークトポロジ

実験は，図 7.8に示すタイミングで行った．実験ネットワークに DV ストリームが 2 本流れた時間は，図中 2-3 間となる．

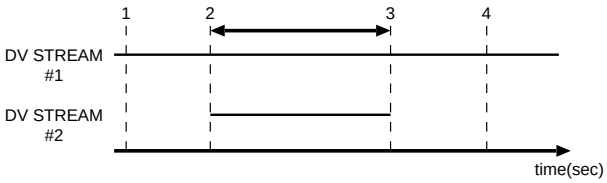

図 7.8: 実験 3 のタイミング

1. DV 送信ノード #1 にて DV 送信を開始
2. DV 送信ノード #2 にて DV 送信を開始
3. DV 送信ノード #2 にて DV 送信を終了
4. DV 送信ノード #1 にて DV 送信を終了

実験結果

2 台の送信ノードにおける映像間引き率の推移を図 7.9に，トラフィック量の推移を図 7.10にそれぞれ示す．

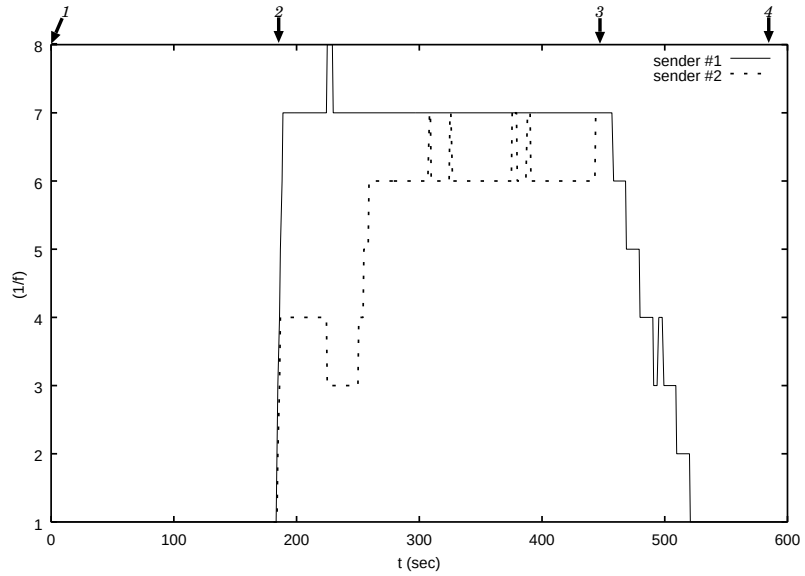

図 7.9: 実験 3 の結果 - 送信ノードにおける映像間引き率

図 7.9より，2 本目の DV ストリーム送信を開始した 2 の段階で，先行してストリームを送信していた送信ノード #1 がネットワークの状態変化を検知し，映像間引き率を $\frac{1}{1}$ から $\frac{1}{7}$ まで変更していることが確認できる．また，後よりストリームを送信開始した送信ノード #2 も同様にネットワーク状態を予測し，映像間引き率を $\frac{1}{1}$ から $\frac{1}{4}$ まで変更した後，徐々に間引き率を増加させ，最終的に $\frac{1}{6}$ 程度で間引き率を安定させていることが確認できる．それに伴い，図 7.10に示すように，それぞれのトラフィックがほぼ同程度で通信帯域を共有するように変化していることが確認できる．

しかし，2 本目の DV ストリームを開始した直後の映像間引き率の値に注目すると，明らかに後から送信を開始した送信ノード #2 の間引き率の方が低い値を取っている．これは，本機構がレート制御を開始するまでに RTCP パケット 5 個分の送受信を行う時間を要し，この時間の間に先行してレート制御を行っていた送信ノード #1 のレート制御が先に実行されてしまったために発生したものと予想される．

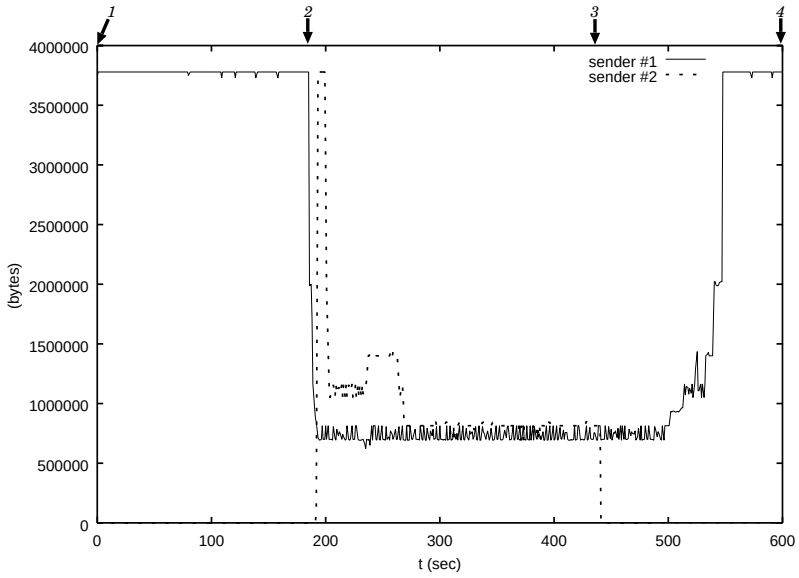

図 7.10: 実験 3 の結果 - 送信ノードにおけるトラフィック量

7.3.4 実験 4：DV ストリームと TCP ストリームの協調動作

実験 4 では，同一経路上に本機構を搭載した DVTS によるトラフィックと TCP によるトラフィックが存在する場合の本機構の挙動を確認する．

実験環境

実験 4 は，図 7.11に示したように DV 送信ノード，DV 受信ノード，TCP 送信ノード，TCP 受信ノードの 4 台の PC を用いて行った．4 台の PC 間には，PC で作成したルータを設置し，片一方のインタフェースに dummynet を設定し，擬似的に 35Mbps に帯域幅を制限した．また，TCP 送受信ノード間は FTP(File Transfer Protocol) を用いたファイルの転送を行った．

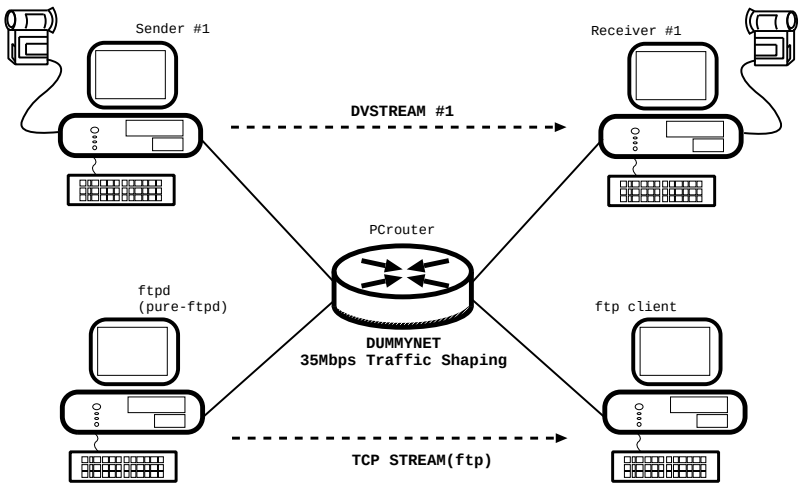

図 7.11: 実験 4 のネットワークトポロジ

実験は，図 7.12に示すタイミングで行った．実験ネットワークに DV ストリームと TCP ストリームが同時に流れた時間は，図中 2-3 間及び 4-5 間となる．

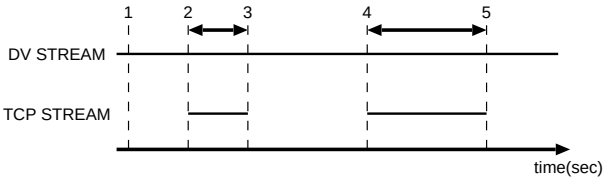

図 7.12: 実験 4 のタイミング

1. DV 送信ノードにて DV 送信を開始
2. FTP によるファイル転送を開始
3. FTP によるファイル転送を終了
4. FTP によるファイル転送を開始
5. FTP によるファイル転送を終了
6. DV 送信ノードにて DV 送信を終了

実験結果

実験の際の DV 送信ノードにおける映像間引き率ならびにトラフィック量の推移を図 7.13に示す．また，DV 送信ノードおよび TCP 送信ノードでのトラフィック量の推移を図 7.14に示す．

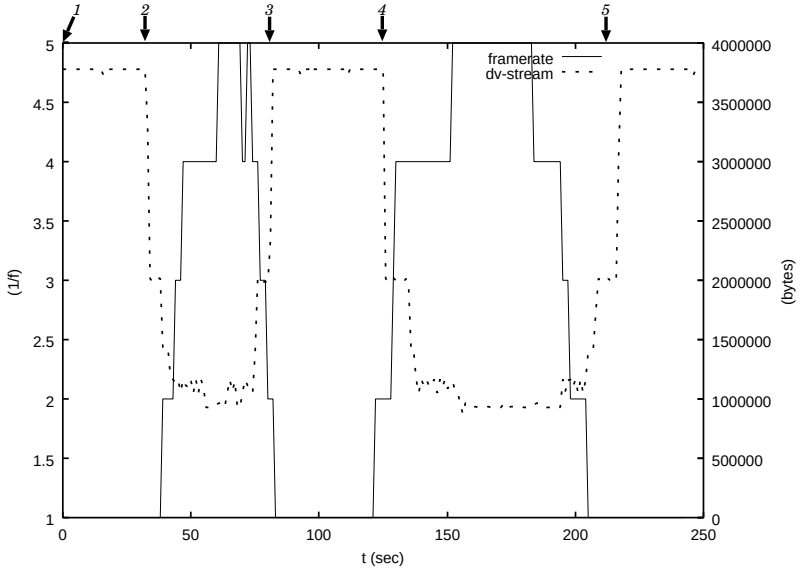

図 7.13: 実験 4 の結果 - 映像間引き率とトラフィック量

図 7.13より，FTP を用いてファイル転送を開始した 2，4 の段階で，本機構がネットワークの変化を検知し，映像間引き率を $\frac{1}{1}$ から $\frac{1}{5}$ まで変更している．それに伴い，トラフィック量も減少している．また，図 7.14より，DV ストリームのトラフィック量が減少した後は，DV ストリームと TCP ストリームの使用する帯域幅がほぼ同程度に変化している．この実験結果より，DV ストリームと TCP ストリームが公平に帯域を分配でき，TCP Friendly による協調的輻輳制御と同様の制御が本機構のみで可能であることが確認できた．

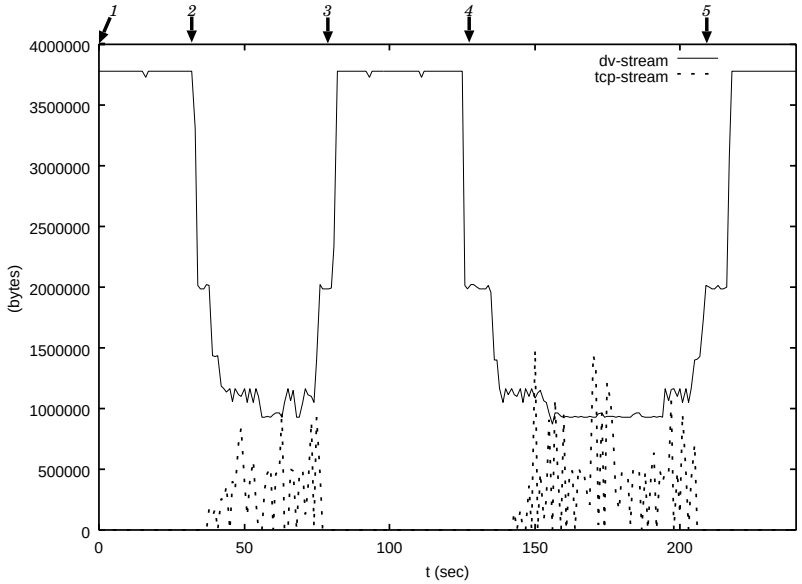

図 7.14: 実験 4 の結果 - DV ストリームと TCP ストリームの協調

7.4 まとめ：適応型映像・音声配信機構の評価

以上の評価結果より，本機構が提供する伝送特性に応じた輻輳制御手法は，以下に挙げるネットワーク状態の変化を検知し，状態に適応した映像間引き率の設定を行い配信可能であることを確認できた．

- 通信経路上の帯域幅の変動
- 通信経路上の伝搬遅延時間の変動
- 複数 DV ストリームにおける協調動作
- TCP ストリームとの協調動作

本機構を利用することにより，ネットワーク状態に適応し，必要に応じて映像間引き率を制御することにより，適応的映像・音声配信を行うことが可能となる．

あわせて，レート制御の際のネットワーク状態認知精度の向上，レート変動安定性の向上，間引き率以外の制御レートの検討といった課題も確認できた．

8章では，本評価結果をもとに本研究での成果ならびに今後の課題について述べる．

第8章　まとめ

本章では，まとめとして，8.1節にて本研究での成果を述べ，8.2節および 8.3節にて今後の課題と展望について述べる．

8.1　結論

リアルタイム性の高い映像・音声配信システムでは，再送制御や輻輳制御を行わない UDP を用いた配信を行っており，ネットワーク状態に適応した配信が難しい問題があった．そこで，本研究では，ネットワーク状態の変化がエンドノード間で取得可能な伝送特性として現れることに着目した．本研究では，エンドノード間で取得可能な伝送特性のみを用いて，中間ルータなどの特別な機器を必要としない映像・音声配信のための新しい輻輳制御手法を提案した．

エンドノード間で取得可能な伝送特性を用いた輻輳制御を行うに当たり，映像・音声配信システムに最適な伝送特性の検討を行った．そして，本研究では，ジッタの揺らぎを主指標として用いた輻輳制御手法を提案した．本手法では，送信ノードならびに受信ノードが協調して，パケットの伝送にかかる時間を計測し，その遅延時間の差分を求めることでジッタを計測する．そして，その値に基づいた映像・音声のフレームレート制御を行う．

また，ジッタの揺らぎを用いた輻輳制御の有効性を確認するために DVTS を用いた伝送実験を行った．その結果，ジッタの揺らぎだけでなく，パケット喪失率の値を補助的に用いることで，急激なネットワーク状態の変化から緩やかなネットワーク状態の変化まで広い範囲のネットワーク状態変化を検知できることが確認された．ジッタの揺らぎならびにパケット喪失率は，映像・音声配信の品質に影響を及ぼしやすい．その特長を活かし，それらの値を指標にした輻輳制御手法は，映像・音声配信システムにとって適している．

本研究では，ジッタの揺らぎおよびパケット喪失率を用いた輻輳制御手法の有用性を実証するために，映像・音声配信システムとして DVTS を用いた伝送特性による適応型映像・音声配信機構 DFCS の設計，実装を行った．本機構 DFCS は，送受信ノード間で RTCP パケットを用いてジッタの揺らぎの情報と，補助情報としてのパケットロスの情報を交換し，伝送路の状態を検知し，その状態に応じたレート制御を行う．

DFCS を組み込んだ DVTS の実際の挙動を確認し，評価を行った．評価の結果，伝送路状態の検知及び状態に適応したレート制御が行えることが確認された．特に，急激なネットワークの変化が発生した場合に，本機構は最も有効な動作をすることも確認できた．

本研究により，中継ルータなどの特別な機器を必要とせず，ネットワークの状態に適応させた映像・音声配信が可能となった．この手法を応用することにより，DVTS 以外の映像・音声配信システムに対してもネットワーク状態に適応した配信を行う機構を実現できる．

8.2 今後の課題

考え得る本研究の今後の課題は以下のものが挙げられる．

フレームレート制御の精度向上・安定化

本機構では，フレームレート制御を行うアルゴリズムを定義した．しかし，そこで用いられる変更閾値の値は今回の実装にあわせて定義したもので，実際のネットワークにおいて DVTS を用いた映像・音声配信を行うことに確実に最適化されたものではない．そのため，間引き率の変更を行う際に，安定した動作をしない場合も考えられる．そこで，その閾値の最適化を行うことが必要であると言える．また，本機構が用いているアルゴリズム自体の最適化も必要であると考えられる．これらの改良が本研究の今後の課題である．

レート制御の手法

本機構では，レート制御を行う際，映像のフレーム間引き率を動的に変化させることで実現している．DVTS では，映像のフレーム間引き率以外の値も使用帯域を制御するために利用可能である．例えば，1 パケット中の DIF ブロック数や音声の冗長化設定などが挙げられる．これらの値の動的変更に対応することで，現在以上に細かい帯域制御を行うことが可能となり，輻輳制御の精度も向上する．そのため，これらの値の制御も動的に行えるよう本機構を改良することも本研究の今後の課題である．

Embedded DVTS への組み込み・活用

現在，DVTS コンソーシアム [29] を通じて行われている共同研究として，ビデオデッキなどの AV 機器に DVTS を組み込む”Embedded DVTS” の開発が行われている．これらの機器への本機構の組み込み・活用も今後の課題である．

特に，組み込み機器では，ジッタなどの情報に依らずネットワークインタフェースの負荷状況や計算機の負荷状況をリアルタイムに把握することができる．そのため，レート制御の指標としてこれらの値を用いることにより，レート制御の精度向上や制御の安定化といった今後の課題の解決も含めて対応できる．

8.3 将来的展望：汎用的な映像・音声の適応的配信

本機構は，DVTS における協調的輻輳制御手法を提供する．しかし，現在 DVTS 以外の映像・音声配信システムも多く開発・利用されている現状があることから，DVTS 以外のシステムに対して協調的輻輳制御手法を提供できるように本機構を改変していくことが考えられる．そのために本機構が解決すべき点を以下に挙げる．

制御パラメータの抽象化

本機構では，DVTS を映像・音声配信システムとして利用しているため，DVTS の持つ品質値であるフレームレートについてのみ言及している．しかし，DVTS 以外の映像・音声配信システムでは，ビットレートや解像度などといった品質値も持っている．そのため，本手法を汎用的に適用するためには，制御を行うパラメータを増やすことが必要である．また，これらのパラメータを抽象化し，複合的に扱うことのできる手法の確立も必要である．

制御を行うための指標の抽象化

本機構では，伝送特性としてジッタの揺らぎならびにパケットロス値を利用した．ネットワークの状態を検知するための指標はこれらの情報に限定されない．このため，ジッタやパケットロス以外の指標を制御時に参照する情報として取り込むことで本手法のさらなる精度向上が可能となる．

謝辞

本研究を進めるに当たり，ご指導頂きました，慶應義塾大学環境情報学部教授 村井純博士ならびに徳田英幸博士，同学部助教授 楠本博之博士ならびに中村修博士，同学部専任講師 南政樹氏ならびに重近範之博士に感謝いたします．

また，絶えずご指導とご助言を頂きました，独立行政法人通信総合研究所 杉浦一徳博士，慶應義塾大学政策メディア研究科修士課程 入野仁志氏，SFC 研究所 小川浩司氏に深い感謝の意を表します．

評価をお手伝い頂いた千代佑氏，松園和久氏をはじめとする慶應義塾大学 徳田・村井・楠本・中村・南合同研究会の諸氏には，本研究を進めていく上でさまざまな励ましとご助言を頂きました．特に，STREAM 研究グループならびに DVTS プロジェクトの皆様にはさまざまなご協力を頂きました．ここに，深い感謝の念を表します．

提出期限が 1ヶ月も早められながらも一緒に苦楽を共にした RG-00 の皆さんありがとう．特に，論文中の図の元となった tgif ファイルを提供頂いた久松”ringo” 剛氏，現実逃避に窓際プログラミングをさせてもらいました堀場”qoo” 勝広氏，お陰様で飯時のネタには困りませんでした工藤”kudo” 紀篤氏など，本当にお疲れ様でした．

参考文献

[1] *TBS News i*. URL:http://news.tbs.co.jp/.

[2] **文化放送インターネット ラジオ** *BBQR*. URL:http://www.joqr.co.jp/bbqr/.

[3] *Shonan BeachFM 78.9*. URL:http://www.763.fm/fmlive.html.

[4] *impressTV*. URL:http://impress.tv/.

[5] **千葉県インターネット 放送局**. URL:http://www.pref.chiba.jp/stream/index.html.

[6] **いばらきインターネット 放送局**. URL:http://www.pref.ibaraki.jp/movie/.

[7] *Triumphal Records*. URL:http://www.triumphal.jp/.

[8] *SOI(School of Internet)*. http://www.soi.wide.ad.jp/.

[9] *SFC-GC*. URL:http://gc.sfc.keio.ac.jp/.

[10] *Windows Media 9 Series*. URL:http://www.microsoft.com/japan/windowsmedia/.

[11] J. Postel. RFC793 Transmission Control Protocol. *http://www.ietf.org/rfc/rfc0793.txt*, pages 1–85, September 1981.

[12] J. Postel. RFC768 User Datagram Protocol. *http://www.ietf.org/rfc/rfc0768.txt*, pages 1–3, August 1980.

[13] Audio-Video Transport Working Group. RFC1889 RTP: A Transport Protocol for Real-Time Applications. *http://www.ietf.org/rfc/rfc1889.txt*, pages 1–75, January 1996.

[14] M. Allman, V. Paxson, and W. Stevens. *TCP Congestion Control*, April 1999. RFC 2581.

[15] Jacobson V. Congestion Avoidance and Control. *Proceedings of ACM SIGCOMM'88*, August 1988.

[16] Jacobson V. Modified TCP Congestion Avoidance Algorithm. *email to the end2end list*, April 1990.

[17] Brakmo L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications, Volumn 13, Number 8*, October 1995.

[18] A.Ogawa. *DVTS (Digital Video Transport System) WWW page*, November 2001. URL:http://www.sfc.wide.ad.jp/DVTS/.

[19] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394-1995 High Performance Serial Bus. Aug 1996.

[20] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394a-2000, IEEE Standard for a High Performance Serial Bus,, Amendment 1. *http://standards.ieee.org/reading/ieee/std/busarch/1394a-2000.pdf*, Jun 2000.

[21] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394b-2002, Amendment to IEEE Std 1394-1995. *http://standards.ieee.org/reading/ieee/std/busarch/1394b-2002.pdf*, Dec 2002.

[22] HD DIGITAL VCR CONFERENCE. Specifications of Consumer-Use Digital VCRs PART2 SD Specifications of Consumer-Use Digital VCRs. *Specifications of Consumer-Use Digital VCRs using 6.3mm magnetic tape*, pages 1–380, Dec 1994.

[23] K, Kobayashi, A. Ogawa, S. Casner, C. Bormann. RFC3189 RTP Payload Format for DV (IEC 61834) Video. *http://www.ietf.org/rfc/rfc3189.txt*, Jan 2002.

[24] S.Floyd and K.Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, August 1998.

[25] A.Ogawa, K.Sugiura, O.Nakamura, and J.Murai. Implementing tcp-friendliness in digital video over ip. **情報処理学会論文誌 第** *43* **巻 第** *2* **号**, February 2002.

[26] K. Ramakrishnan and S. Floyd. *A Proposal to add Explicit Congestion Notification (ECN) to IP*, January 1999. RFC 2481.

[27] *The Public Netperf Homepage*. URL:http://www.netperf.org/netperf/NetperfPage.html.

[28] Akimichi Ogawa. Design and implementation of dv based video transportation with congestion control. 2003.

[29] *DVTS* **コンソーシアム**. URL:http://www.dvts.jp/.

付 録A　RTP

本章では，インターネットを介した映像・音声配信に用いられるRTPについて述べる．A.1.2節では，本機構が受信者からジッタなどの情報を取得するために用いるRTCPについても併せて述べる．

A.1　RTP(Real-time Transport Protocol)

RTPは，インターネット上で映像や音声などのリアルタイム性が重視されるデータの転送に利用されるプロトコルである．RTPは，RFC1889としてIETFにて定義されている．RTPの持つサービスとしては，以下の5つの機能がある．

- ペイロードタイプ識別
- シーケンス番号付与
- タイムスタンプ付与
- ジッタ対策
- パケット喪失対策

RTPは，主にUDP上で動作するプロトコルとして設計されているが，UDP以外のプロトコル上でも動作可能である．RTPは，マルチキャストにも対応している．

RTP自身は，到着時間の保証といったQoS制御は行わない．また，RTPは配送や配送の到達順序自体を保証するプロトコルではない．RTPパケットの受信者は，RTPパケットに含まれるシーケンス番号を利用して，パケットの並び替えや喪失の検知を行うことができる．

A.1.1 RTP パケットフォーマット

RTP パケットヘッダのフォーマットを図 A.1に示す.

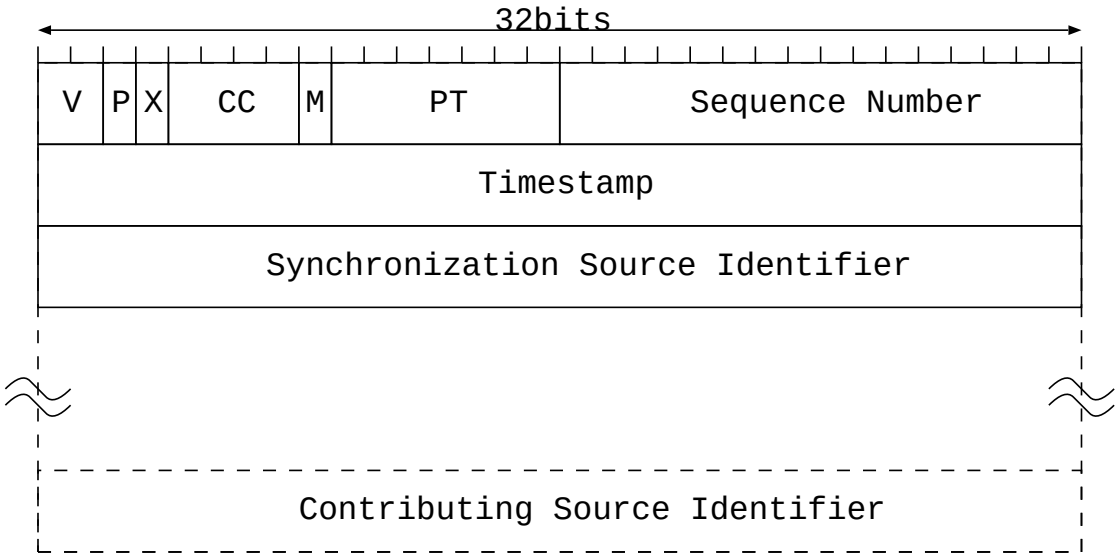

図 A.1: RTP パケットフォーマット

RTP パケットヘッダの各項目の内容を以下に示す.

- V(Version):2bits
  - RTP のバージョン番号を示す．値は 2
- P(Padding):1bit
  - パディングの有無を示す.
- X(eXtension):1bit
  - 拡張ヘッダの有無を示す．拡張ヘッダがある場合，RTP ヘッダの直後に拡張ヘッダが付加される.
- CC(CSRC Count):3bits
  - CSRC カウント．RTP ヘッダの直後に CSRC がいくつあるかを示す.
- M(Marker):1bit
  - マーカービット．ペイロード・タイプごとに使用法が異なる.
- PT(Payload Type):7bits
  - ペイロードタイプを示す．RTP ペイロードの種類を指定する.
- Sequence Number:16bits

  - シーケンス番号を示す．シーケンス番号の初期値は乱数によって指定する．

- Timestamp:32bits
  - タイムスタンプを示す．タイムスタンプのためのクロック基準周波数はペイロード・タイプごとに異なる．

- SSRC:32bits
  - ソース識別子を示す．異なる SSRC 値を用いることにより送信者のネットワークアドレスが同一でも区別できる．

- CSRC:32bits
  - 寄与ソース識別子を示す．RTP ミキサによって統合されたストリームの送信者リスト．

A.2 RTCP(Real-time Transport Control Protocol)

A.1節に挙げた RTP には，以下のような 4 つの問題点がある．

- フロー制御
- クロック同期
- メディア間同期
- 情報ソース識別

RFC1889 には，RTP と同時に RTCP も定義している．RTCP は，RTP セッションのモニタリングとその情報のフィードバックをセッション参加者に伝えるためのプロトコルである．RTCP では，送信者と受信者の間で RTCP パケットを交換することで，上記問題の解決を図る．しかし，RTCP は，RTP のための通知プロトコルであるため，RTCP 自体が何らかのデータを配送したり，TCP のようにエンドノード間での到達性を保証することはできない．

送受信者間で交換される RTCP パケットには以下の 5 つの種類がある．

- SR(Sender Report):
  - RTP セッションにおける送信者での状態通知に使用される RTCP メッセージ

- RR(Receiver Report):
  - RTP セッションにおける送信者以外の参加者における状態通知に使用される RTCP メッセージ

- BYE:
  - セッションからの離脱を通知する RTCP メッセージ

- SDES(Source DEScription):
  - RTP パケットの SSRC/CSRC の値とユーザ情報との関係を通知する RTCP メッセージ
- APP(APPlication):
  - RTCP 規定外アプリケーション固有の制御情報を通知する RTCP メッセージ

A.2.1 RTCP SR/RR パケットフォーマット

RTCP SR パケットのフォーマットを図 A.2に示す.

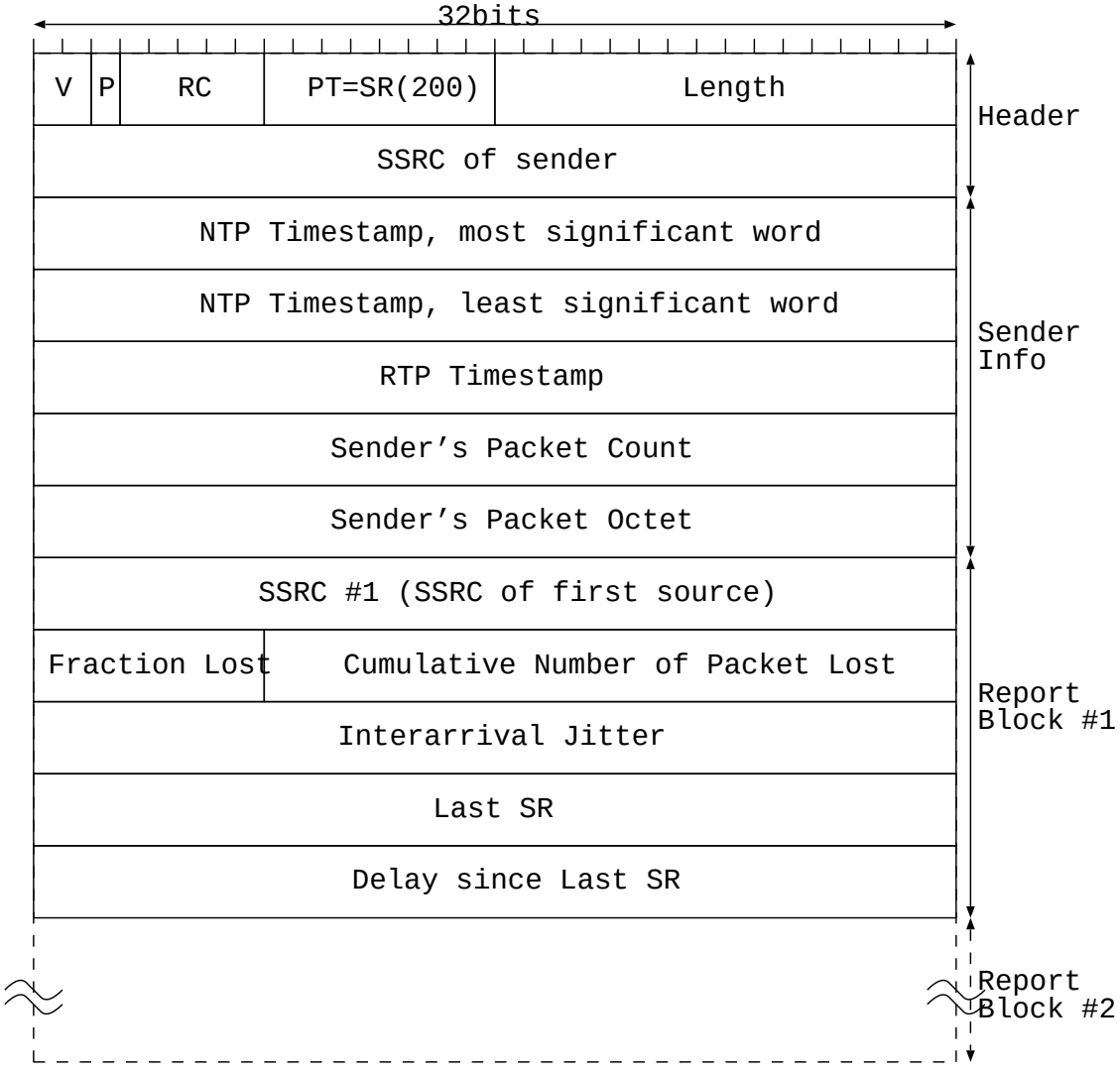

図 A.2: RTCP SR パケットフォーマット

RTCP RR パケットのフォーマットを図 A.3に示す.

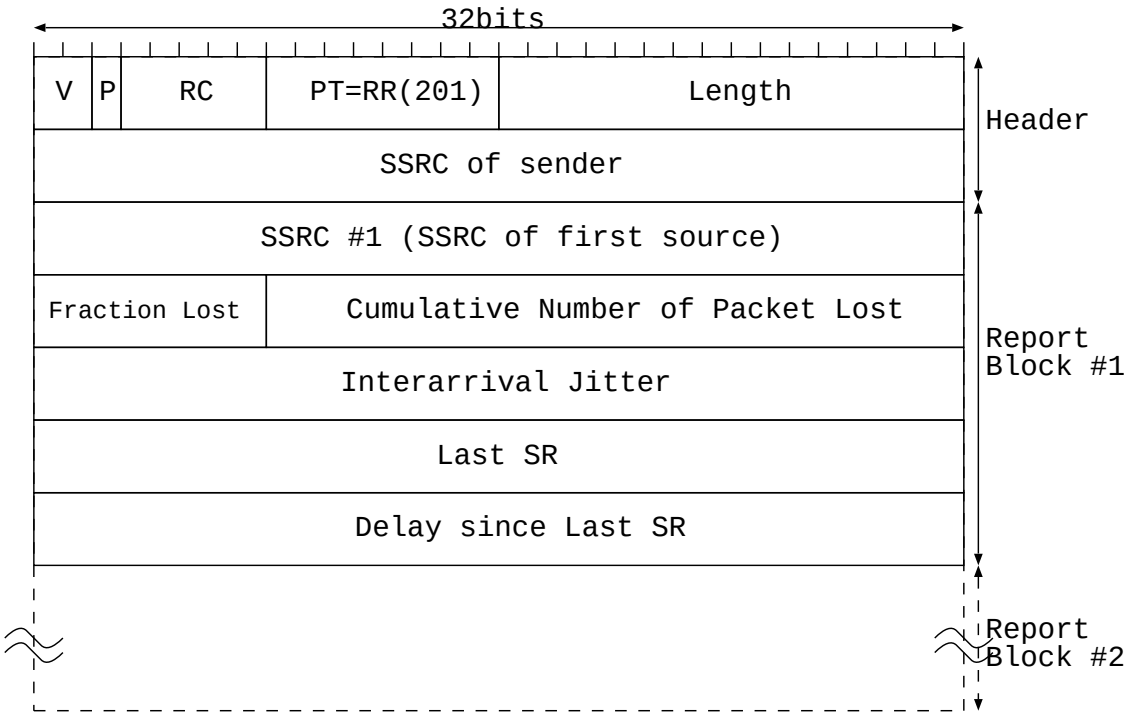

図 A.3: RTCP RR パケットフォーマット

RTCP SR/RR パケットは複数のセクションから構成される．1 つ目のセクションは，ヘッダ部であり，8bytes で構成される．ヘッダ部の構成を以下に示す．

- V(Version):2bits
  - RTP のバージョン番号を示す．値は 2
- P(Padding):1bit
  - パディングの有無を示す．
- RC(Reception Report Type):5bits
  - パケットに含まれる Reception Report 数を示す．
- PT(Packet Type):8bits
  - RTCP パケットの種類を示す．RTCP SR は”200”．RTCP RR は”201”．
- Length:16bits
  - RTCP パケット長を示す．
- SSRC:32bits
  - RTCP SR パケット送信者を特定するための番号を示す．

2つ目のセクションは，SR メッセージと RR メッセージにより異なる．SR メッセージには，送信者から送信されるストリームに関する情報 (Sender Info) が含まれ，20bytes で構成される．なお，RR メッセージの場合，この送信者情報は含まれない．Sender Info 部の構成を以下に示す．

- NTP Timestamp:64bits
  - この RTCP SR パケットが送信された時間を NTP による時間で表したもの．絶対時間も経過時間も取得できない場合は，0 を代入してよい．
- RTP Timestamp:32bits
  - この RTCP SR パケットが送信された時間を RTP タイムスタンプによって表現したもの．
- Sender's Packet Count:32bits
  - セッションが開始されて，この RTCP SR パケットが送信されるまでに送出された RTP パケット数．
- Sender's Octet Count:32bits
  - セッションが開始されて，この RTCP SR パケットが送信されるまでに送出された RTP パケットのペイロード部分の累積バイト数．

これ以降のセクションには，0 個もしくは RTCP ヘッダ部の RC が示す個数分のストリームを受信する受信者に関する情報 (Reception Report) が含まれる．Reception Report 部は，24bytes で構成される．Reception Report 部の構成を以下に示す．

- SSRC:32bits
  - この Reception Report を提供した RTP セッション参加者の SSRC を示す．
- Fraction Lost:8bits
  - 喪失した RTP パケット数を示す．喪失したパケット数を受け取るはずのパケット数で割った値が使用される．
- Cumulative Number of Packet Lost:24bits
  - 喪失した RTP パケット数の合計を示す．
- Extended Highest Sequence Number Received:32bits
  - 上位 16 ビットはシーケンス番号が初期値に戻った回数，下位 16 ビットは最後に受け取った RTP パケットのシーケンス番号をそれぞれ示す．
- Interarrival Jitter:32bits
  - 受け取った RTP パケットの到着時間の揺らぎ (ジッタ) を示す．

- LSR(Last SR timestamp):32bits
  - 最後に受信した RTCP SR パケットに含まれていた NTP タイムスタンプの値の 16 ビット目から 47 ビット目までの 32 ビットを示す.
- DLSR(Delay since Last SR):32bits
  - 最後に RTCP SR パケットを受信してからこのパケットを送信するまでの時間を, 1/65536 秒単位で表したもの.

付 録B　DVTSの各関数

本章では，DVTS に追加された DFCS に関わる関数のソースコードを提示する．

B.1 dvsend

B.1.1 *_process_framerate_check* 関数

```
static int
_process_framerate_check (struct dvsend_param *dvsend_param,
                          int jitter, int pkt_loss_sum)
{
     int jitter_ave;

     /* discard until getting 5 jitter count */
     if (dvsend_param->log_count <= 4) {
       return(1);
     }

     /* average previous 5 jitter */
     jitter_ave = dvsend_param->jitter_ppp +
                  dvsend_param->jitter_pp +
                  dvsend_param->jitter_p +
                  dvsend_param->jitter_prev + jitter;

     /* print debug message */
     printf("Sum of Jitter: %d+%d+%d+%d+%d=%d(%d)\n",
            dvsend_param->jitter_ppp,
            dvsend_param->jitter_pp,
            dvsend_param->jitter_p,
            dvsend_param->jitter_prev,
            jitter,
            jitter_ave,
            jitter_ave/5);

     /* limit_rate change by packet loss */
     if (dvsend_param->noloss_count > 10) {
```

```
    if (dvsend_param->frame_limit > 1) {
      dvsend_param->frame_limit--;
    }
    /* frame drop rate change by packet loss */
    _process_framerate_change(dvsend_param, -1);
    return(1);
  }

  /* limit_rate change by packet loss */
  if (pkt_loss_sum > 50) {
    if (pkt_loss_sum > 100 && dvsend_param->frame_limit <= 30){
      dvsend_param->frame_limit++;
    }
    /* frame drop rate change by packet loss */
    _process_framerate_change(dvsend_param, 1);
    return(1);
  }

  /* frame drop rate change by jitter */
  if(jitter_ave / 5 > 4000){
    dvsend_param->oj_count++;
    if(dvsend_param->oj_count > 3){
      _process_framerate_change(dvsend_param, 1);
      dvsend_param->oj_count = 0;
      return(1);
    }
  } else if(dvsend_param->oj_count) {
    printf("counter reset!\n");
    dvsend_param->oj_count = 0;
    return(1);
  }

  /* frame drop rate change by jitter */
  if(jitter_ave / 5 < -2400){
    dvsend_param->sj_count++;
    if(dvsend_param->sj_count > 3){
      _process_framerate_change(dvsend_param, -1);
      dvsend_param->sj_count = 0;
      return(1);
    }
  } else if(dvsend_param->sj_count) {
    printf("counter reset!\n");
    dvsend_param->sj_count = 0;
```

```
      return(1);
    }

    /* frame drop rate change by jitter and packet loss */
    if (jitter < -1500 && dvsend_param->noloss_count > 3) {
      _process_framerate_change(dvsend_param, -1);
      return(1);
    }

    /* frame drop rate change by jitter and packet loss */
    if (jitter < -5000 && dvsend_param->noloss_count > 1){
      _process_framerate_change(dvsend_param, -1);
      return(1);
    }

    return(1);
}
```

B.1.2 *_process_framerate_change* 関数

```
static int
_process_framerate_change (struct dvsend_param *dvsend_param, int change)
{
    /* check allowed frame drop rate */
    if (dvsend_param->frame_limit <= dvsend_param->frame_drop + change
        && dvsend_param->frame_drop + change <= 30) {

      printf("FrameRate Changed!: %d -> %d\n",
             dvsend_param->frame_drop,
             dvsend_param->frame_drop + change);

      /* change frame drop rate */
      dvsend_param->frame_drop += change;
      dvsend_param->noloss_count = 0;

      return(1);
    } else {
      printf("Over Allowed Framerate\n");
      return(-1);
    }

    return(1);
```

```
}
```

B.2 dvrecv

B.2.1 *_process_rtcp_jitter* 関数

```
static int
_process_rtcp_jitter (struct dvrecv_param *dvrecv_param)
{

  int transit_sec, transit_usec, jitter_minus = 0;
  double transit, delay;
  u_int32_t for_rr_jitter;

  /* calc transit time of DV/RTP packet */
  transit_sec = dvrecv_param->rtcp_now_ntpsec - dvrecv_param->sr_ntp_sec;
  transit_usec = dvrecv_param->rtcp_now_ntpusec - dvrecv_param->sr_ntp_frac;
  transit = (double)(transit_sec + (double)(transit_usec / 1000000.0));

  /* calc delay time of DV/RTP packet */
  delay = transit - dvrecv_param->jitter_prev_transit;
  dvrecv_param->jitter_delay = delay;

  /* set previous transit time of DV/RTP packet */
  dvrecv_param->jitter_prev_transit = transit;

// (RFC based jitter implementation)
//  /* jitter implementation based on RFC */
//  dvrecv_param->jitter_jitter +=
//    (1./16.) * (dvrecv_param->jitter_delay -
//    dvrecv_param->jitter_jitter);
//
//  /* print jitter */
//  printf("%lf\n",dvrecv_param->jitter_jitter);

  /* jitter implementation based on time-diff-mesurement */
  dvrecv_param->jitter_jitter = dvrecv_param->jitter_delay;

  /* print jitter */
  printf("%f\n",dvrecv_param->jitter_jitter);

  /* check jitter sign */
  if (dvrecv_param->jitter_jitter < 0) {
```

```
    dvrecv_param->jitter_jitter = -dvrecv_param->jitter_jitter;
    jitter_minus++;
  }

  /* jitter change from double val. to int val. */
  for_rr_jitter = dvrecv_param->jitter_jitter * 1000000.0;
  for_rr_jitter <<= 1;

  /* (if jitter is minus) change 1bit for notice minus */
  if(jitter_minus){ for_rr_jitter |= 0x00000001; }

  /* set Interarrival Jitter for RTCP RR */
  dvrecv_param->rr_jitter = for_rr_jitter;

  return(1);
}
```