修士論文

NDN上のビデオストリーミングにおける可用帯域推定を用いたビットレート適合化

2018年02月01日

指導教員 相田 仁 教授

東京大学大学院工学系研究科
電気系工学専攻

37-146519 趙亮

概要

コンテンツ配信を効率化する仕組みとして提案された新型ネットワークアーキテクチャ Named Data Network(NDN) 上でマルチメディアストリーミングアプリケーションを実装するにあたって、ビデオストリーミングビットレート適応 (bitrate adaptation) に必要な通信速度制御が NDN フレームワークでサポートされておらず、従来の TCP/IP のために設計されたビットレート適応アルゴリズムをそのまま NDN フレームワークに適応することができないという問題があった。このような問題を解決するために Interest 送信速度とパケット間間隔歪みの関係をモデル化し、カルマンフィルタを用いたネットワーク可用帯域推定手法を提案した。推定された可用帯域に基づき Interest 送信速度を制御し、さらにビットレート適応アルゴリズムにも推定された可用帯域幅を応用することで、NDN におけるビデオストリーミングビットレート適応を可能にした。提案手法を利用して NDN ビデオストリーミングアプリケーションを実装し、ネットワークシミュレータ NS-3 を用いて性能評価を行った。提案手法を用いると、利用可能な帯域幅を高精度に推定することができ、その結果を応用したビットレート適応アルゴリズムもバッファレベルを安定に維持させながら、従来の MPEG-dash より高いビットレートのコンテンツを選択できることを確認した。

In the NDN framework, the Interest rate control necessary for video streaming bitrate adaptation is not supported, and the bitrate adaptation algorithm designed for the conventional TCP/IP cannot be applied to the NDN framework. In this paper, a novel available bandwidth estimation method based on inter-packet strain is proposed in the NDN framework. Using this method, the available bandwidth can be estimated with high accuracy. By controlling Interest rate based on the estimated available bandwidth, and adapting the estimated available bandwidth to the bitrate adaptation algorithm, it was possible to select higher bitrate in video streaming without causing playback stall.

目次

図目次

第1章

序論

近年 Youtube や iTunes などのコンテンツ配信サービス、モバイル端ビデオアプリおよびストリーミングライブ配信の需要の増加に伴い、マルチメディアストリーミングがネットワークトラフィックの増加の主な原因となっている。マルチメディアストリーミングでは、クライアントがネットワークの利用状況や再生の状況に応じてリクエストするコンテンツのビットレートを調節するビットレート適応 (bitrate adaptation) アルゴリズム [4] が使われている。

ビットレート適応アルゴリズムでは、可用帯域 (available bandwidth) に制限がなければ最高品質のコンテンツを利用できるが、複数の利用者がいる場合に高いビットレートのコンテンツをリクエストするとネットワークリソースへの需要がネットワークの容量を上回り、輻輳状態になりやすい。ネットワーク輻輳によってストリーミング再生が一時停止（playback stall）することが発生し、ユーザーの QoE（quality of experience）の低下につながる。適した品質のコンテンツを選ぶためには、ネットワークの可用帯域を把握することが重要である。

一方、IP ネットワークはエンドツーエンドのコミュニケーションのために設計されたもので、今日のインターネットは支配的にコンテンツの配信ネットワークとして利用されている。このような利用形態の変化に対応して、コンテンツ配信を効率化する仕組みとして提案されたのは、インターネット全体をクリーンスレートで再設計した次世代のネットワークアーキテクチャ情報指向型ネットワーク (Information Centric Network) で、その中の代表的なプロジェクト Named Data Network(NDN)[1] が注目されている。

NDN ネットワークのエンジニアリング特性は IP ネットワークとほぼすべて共通で、NDN における通信および制御の問題はほとんど従来の手法を適用させることで効率的に解決することができる。しかし、NDN フレームワークには、個別のトランスポート層がないため、NDN ライブラリで実装されたパケットの再送のような一部の機能を除いて、輻輳制御、プロセス間の多重化、データの完全性と信頼性の保証など、標準的なトランスポート層機能の実現は各アプリケーションに任される。このため、ビデオストリーミングビットレート適応 (bitrate adaptation) に必要な通信速度制御が NDN フレームワークでサポートされておらず、従来の TCP/IP のために設計されたビットレート適応アルゴリズムをそのまま NDN フレームワークに適応することができない。

このような問題を解決するために、NDN の consumer-driven で pull 型通信の特性を考慮し、Interest 送信速度とパケット間間隔歪みの関係をモデル化し、カルマンフィルタを用いたネットワーク可用帯域推定手法を提案した。推定された可用帯域に基づき Interest 送信速度を制御し、ビットレート適応アルゴリズムにも推定された可用帯域幅を応用することで、NDN におけるビデオストリーミングビットレート適応を可能にした。

提案手法を利用して NDN ビデオストリーミングアプリケーションを実装し、ネットワークシミュレータ NS-3 を用いて性能評価を行った。可用帯域推定手法では高い精度でネットワークの可用帯域を推定できることと、それ用いた従来のビットレート適応アルゴリズムを適用したところ、安定して高いビットレートでストリーミング再生ができることをシミュレーションで確認した。

第2章

研究背景

2.1 NDN アーキテクチャ

今日のインターネットの砂時計アーキテクチャは、グローバルな相互接続性に最低限の機能を実装した、ユニバーサルなネットワーク層（すなわち、IP）を中心としている。この細いウエスト (thin waist) は、下層と上層の両方の技術を独立して革新することで、インターネットの爆発的な成長を可能にした。しかし、IP はエンドツーエンドのコミュニケーションのために設計されたもので、今日のインターネットは支配的にコンテンツの配信ネットワークとして利用されている [1]。

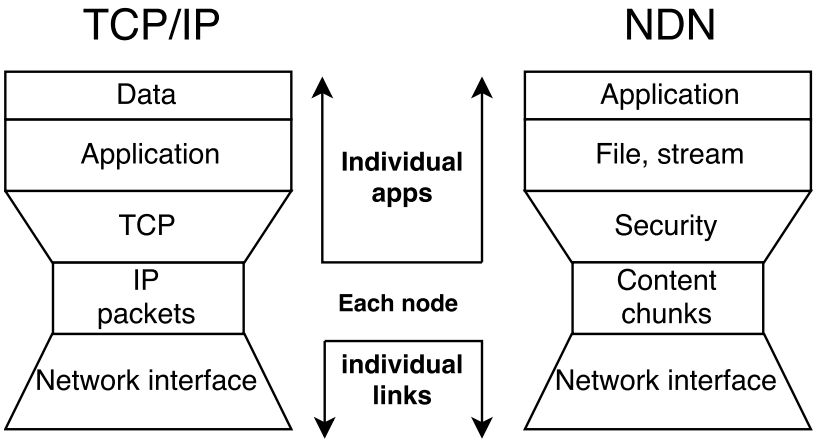

図 2.1: the hourglass model of TCP/IP and NDN[1]

こういった利用形態の変化に対応し、コンテンツの配信を効率化するために提案されたのは情報指向型ネットワーク (Information Centric Network) である。その中に代表的なものとして Named Data Netork(NDN) や Contents Centric Network(CCN) がある。

Named Data Networking（NDN）プロジェクトは、コンテンツデータのようなエンドノード以外のオブジェクトに名前を付けることができるように、この細いウエストの役割

を一般化し、IP アーキテクチャーよりも進化した新型ネットワークアーキテクチャを提案した。

具体的には、NDN は、指定された宛先アドレスにパケットを配送するネットワークサービスのセマンティクスを変更して、指定された名前で識別されるデータをリクエストを行い、名前は他にルーティングやキャッシングなどの役割を担っている。NDN パケットの名前は、エンドノード、映画や本のデータチャンク、ライトをオンにするコマンドなど柔軟に名前をつけることができる。

ネットワークアーキテクチャレベルで概念が変わったものの、NDN ネットワークのエンジニアリング特性は IP ネットワークとほぼすべて共通で、NDN におけるほとんどの通信および制御の問題は従来の手法を適用させることで効率的に解決することができる。

NDN を、従来の IP ネットワークであまり得意としていない分野のアプリケーションに応用し、問題解決のアーキテクチャフレームワークとして展開し、アーキテクチャの方向性を検証しながら、アプリケーションの設計で得られる経験をもとにアーキテクチャの方向性も修正していくことで、将来の実用化を目指している。

2.1.1　Interest と Data

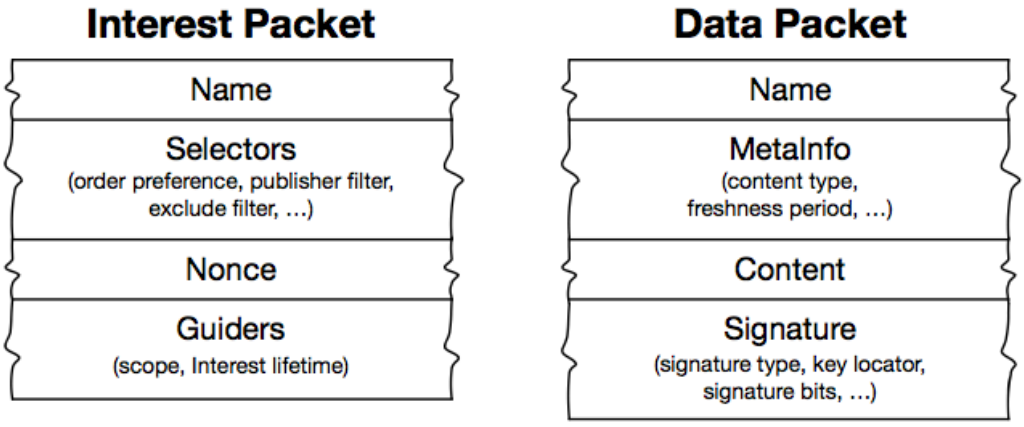

図 2.2: Interest/Data packet format

NDN データ配信において、Interest と Data という 2 つのタイプのパケットがあり図 2.2、その標準フォーマットの仕様が決められている。2 つのタイプのパケット Interest と Data をやりとりして通信することで、IP と同等の役割を果たせると考えられる。

コンテンツはパケット単位でユニークな名前を持っており、その名前でルーティングを行っている。ネットワークの利用形態としてコンテンツの消費者 (Consumer) が Interest パケットを出し欲しいコンテンツをリクエストし、Data パケットを入手するという Pull 型のサービスを想定している。NDN において通信はデータを受け取る側、つまり Data consumer から開始する。この 2 つのタイプのパケットにリクエストしているデータ、もしくは Data

に含まれているデータコンテンツを説明する名前 (Name) がつけられている。Interest は名前、セレクタ (Selector) とランダムな Nounce から構成されている。Selector を用いることで特定の発行者を選択することができる。Nounce が Interest につけられたランダムな数で、ルーターに同じ名前の Interest が来たら、Nounce で同じ Interest がループしているかを検出することができる。Data は名前とデータペイロード (Payload) とデジタル署名 (Signature) から構成されている。

Interest はリクエストメッセージで、Consumer がリクエストするコンテンツの名前を Interest に書き込み、ネットワークに送信する。ルーターが Interest を転送し、リクエストコンテンツを持った発行者 (Data producer) もしくはコンテンツのキャッシュを持ったコンテンツルーターにたどり着いたら、Data が送り返される。Interest が応答され、データを入手することを Interest が満たされるという。Data が PIT を参照して consumer に送り返されるので、往路と復路がいつでも同じパスである。

2.1.2 Forwarding model

Interest と Data を転送するために NDN のルーターが 3 つの表を持っている。キャッシングデータを保存する Content Store、Interest を転送するための経路表 Forwarding Information Base(FIB) と満たされていない Interest の表 Pending Interest Table(PIT) である。

PIT には転送されてまだ満たされていない Interest の情報を持っている。PIT が Interest の名前と Interest が来た Interface と転送先 Interface を記録している。Content Store は転送された Data のキャッシュで、Data の名前と Payload が保存されている。FIB は Interest を転送するための経路表で、コンテンツの Prefix と次の転送先の情報が保存されている。

まず Interest の処理の流れ図 2.3 を紹介する。ルーターに Interest が送られて来たら、まず Content Store で Data を探す。Content Store に対応する Data のキャッシュがあれば、Interest を受信した Interface に応答する。Content Store でキャッシュヒットしなければ、次は PIT を参照する。PIT に一致するエントリがあれば、エントリの Requesting Faces に Interface を追加し、Interest を棄却する。ルーターが同じ Interest を複数受信しても、その中の一つしか転送しない。その場合 Data が送り返されてきたら、Requesting Faces にあるすべての Interface に Data を送り出す。PIT にエントリがなければ、FIB を参照し Data producer に向けて Interest を転送し、PIT に新しいエントリーを作成し、その Interest が来た Interface と転送先の Interface を追加する。 FIB のルーティング情報テーブルはルーティングプロトコルによって構築され、FIB のエントリには複数の転送先 Interface を持つこともできる。Interest 名前で最長一致 Prefix lookup で次の転送先を決め、Forwarding Strategy によって一つまたは複数の Interface に送信する。

Data の処理方法は Data パケットを受信すると、Content Store を参照し、その Data がすでにキャッシングされていれば、その Data を棄却する。そうでなければ PIT を参照して、Requesting Faces に Data を転送し、Requesting Faces のエントリを消去する。Data が Caching Strategy に従いキャッシングされ、CS がいっぱいになったときに、Caching Strategy によって古いキャッシュを置き換える。

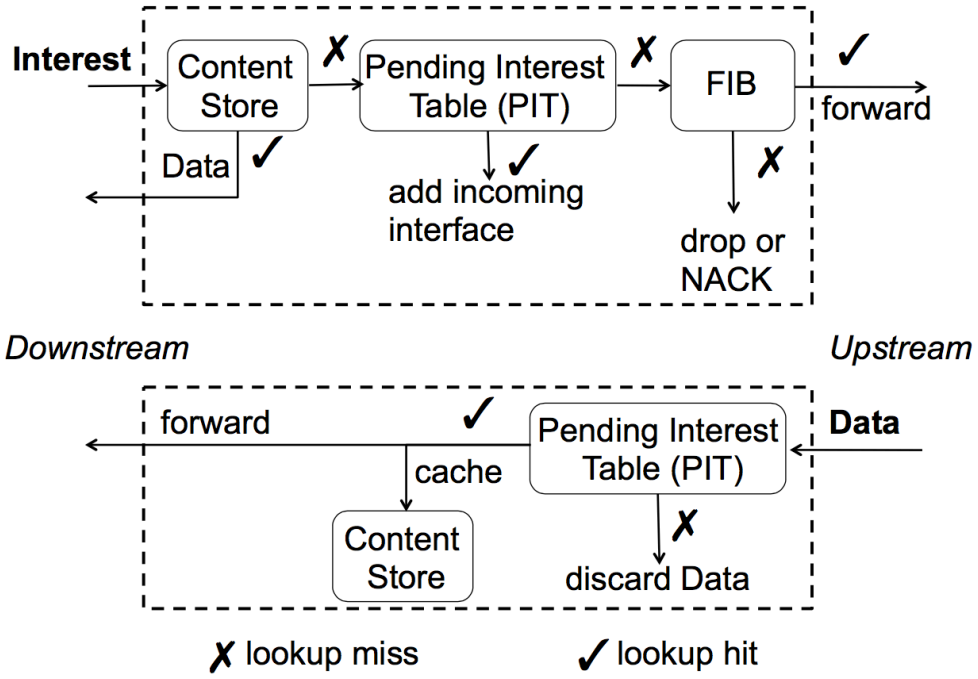

図 2.3: Interest/Data processing[2]

2.1.3　Interest 送信速度

従来の HTTP ベースの mpeg-dash では、クライアントは 1 つの HTTP リクエストででセグメント全体をダウンロードできる。 サーバ側 TCP 輻輳制御アルゴリズムに基づき、データパケットを適切な速度で返してもらえる。 しかし、NDN フレームワークでは、Interest パケットと Data パケットは一対一で対応している。通常、1 つのセグメントには複数の Data パケットが必要なため、複数の Interest パケットを送信する必要がある (図 2.4)。 Consumer は、Data パケットの受信速度を制御するために Interest パケットの送信速度を制御する必要がある。 Consumer 側で、Interest パケットを送信する間隔を調整することによって、Interest の送信速度を制御している。 Interest があまりにも速く送信すると、Data パケットのスループットが増加し、ボトルネックでネットワークの輻輳につながる。 Interest の送信速度が遅すぎると、Data パケットのスループットが減少し、ネットワークリソースを十分に活用することができない。

2.2　MPEG-dash のビットレート適応アルゴリズム

HTTP を利用した適応的ビデオストリーミング、HTTP Adaptive Streaming(HAS) には様々なプロトコルや規格が提案されているが、このセクションでは、その規格の一つである MPEG-dash（ISO/IEC 230009-1）[4]、およびビットレート適応アルゴリズムについて紹介する。RTP のような UDP ベースのメディアストリーミングプロトコルは、メディ

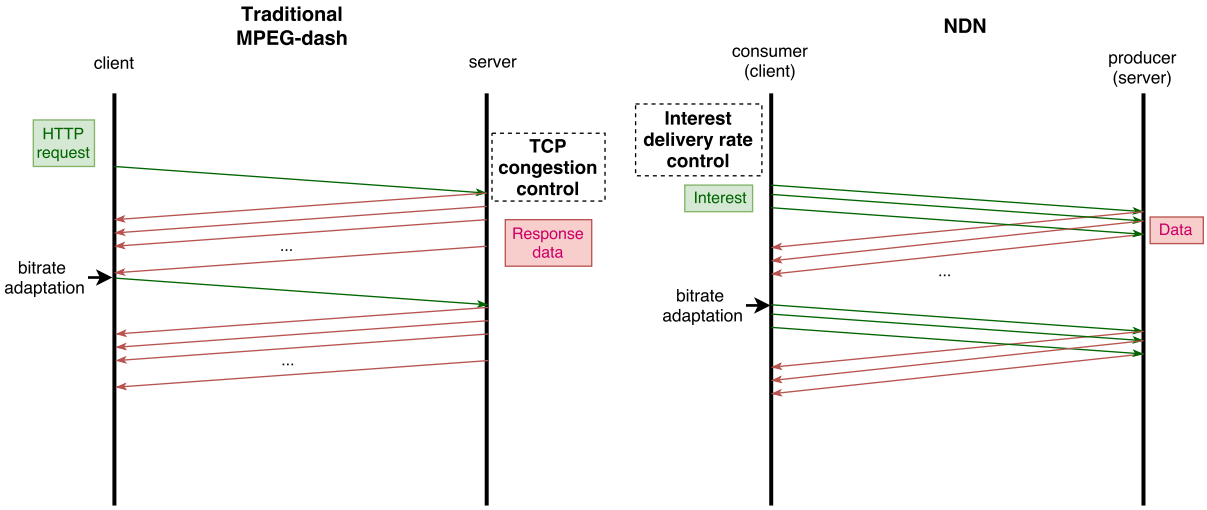

図 2.4: TCP/IP and NDN segment request comparsion

アストリーミングに適しているが、HTTP プロトコルの普遍性のために、HTTP ベースの MPEG-dash が徐々に主流になっている。 メディアストリーミングサービス事業者にとって MPEG-dash を使用すれば、既存の HTTP サーバをストリーミングメディアサーバとして使用でき、かつ異なるポリシーのファイアウォールによって制限されないといったメリットがある。

MPEG-dash において、コンテンツが事前にセグメント単位で異なるビットレートに符号化され、HTTP サーバーに保存されている。そしてクライアントが動画が再生中に途切れないように、ネットワーク帯域幅や再生の状況に応じて適応的にビデオのビットレートを切り替え、許容される最高のビットレートのコンテンツを取得するビットレート適応アルゴリズムが使われている。

[3] で提案されている Fdash はファジィ制御を利用して、プレーヤーのバッファレベルを一定の目標バッファレベルに維持させながら、一定時間内セグメントの平均スループットから次のセグメントのビットレートを求めるビットレート適応アルゴリズムである。

ビデオ再生中にバッファがなくなると再生が一時停止するので、スムーズな再生を実現するために、バッファの量を一定以上に維持する必要があり、逆に、ずっと低いビットレートを選択するとバッファがオーバーランするので、Fdash はスムーズなビデオ再生を保証するために、目標バッファレベルを設定し、バッファレベルを一定の目標バッファレベルに維持させながら、ビデオセグメントのビットレートを制御する。

再生前のバッファリング時間を最小限に抑えるために、プレーヤーはデータを受信した直後に再生を開始するため、最初のビデオビットレートが最も低いビットレートになる。目標バッファレベルに達した後、ビットレート適応アルゴリズムはより高いビットレートのセグメントを引き続き取得する。

ファジィ集合は、曖昧な表現を定量的に扱うために、従来の集合を拡張したもので、集合 U と U から単位閉区間 $[0,1]$ への函数 μ の対 (U,μ) で定義される。ファジィ集合のメンバシップ関数 μ は、各 $x \in U$ に対して、値 $\mu(x)$ は (U,μ) における x の帰属度 (grade of

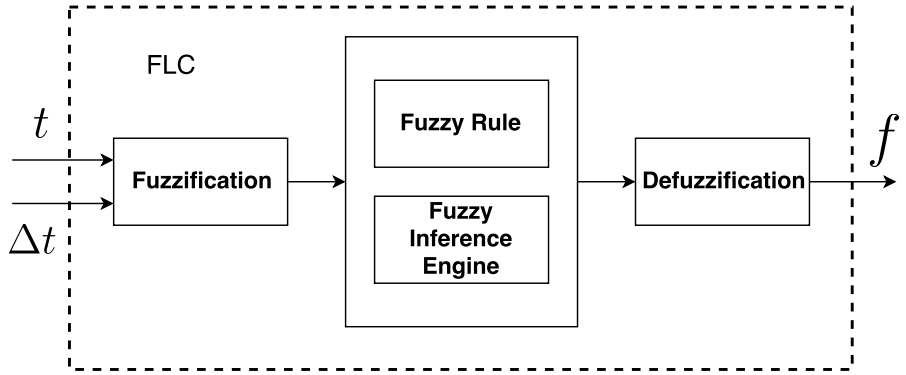

図 2.5: the fuzzy logic controller

membership) と呼ばれる。AND 演算や OR 演算などの論理演算をファジィ集合に拡張したものがファジィ論理と呼ばれる。

ファジィ集合を用いた制御をファジィ制御という。fdash ではファジィ制御のファジー論理コントローラー (FLC) は、以下の 4 つのコンポーネントから構成される。

1. ファジィ化 (fuzzfication)
ファジィ化では入力された数値をメンバシップ関数を用いてファジィ集合への帰属度を求める。

2. ファジィルールベース (fuzzy rule base)
if-then のような文法で表された論理表現で、制御対象の挙動や制御方法などを記述する。

3. ファジィ推論エンジン (buffer inference engine)
ファジィルールベースに基づきファジィ化された入力から出力を求める。

4. 非ファジィ化 (defuzzfication)
得られたファジィ集合をクリスプ集合へ変換する。Fdash では加重平均を使う。

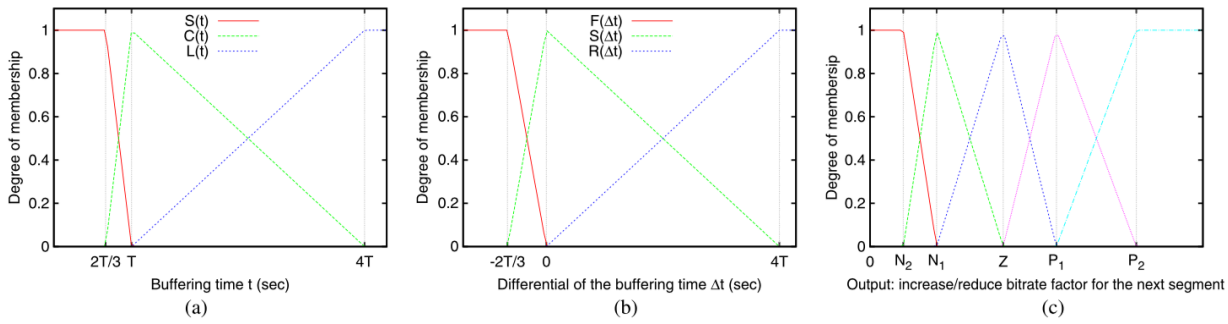

図 2.6: the member functions[3]

ファジー論理コントローラへの入力として、fdash はセグメントの受信から再生までのバッファリング時間（バッファレベル）t と前のバッファリング時間との差分 Δ_t を使用

する。

設定された目標バッファレベルと現在のバッファレベルの差を記述するために、三つの言語変数 short（低い）、close（安定）、long（高い）が使われ、バッファレベルの変化を記述するために falling（落下）、steady（定常）、および rising（上昇）の三つ言語変数が使われている。FLC の出力 f は、次のセグメントのビットレートの増減係数を表す。出力の言語変数は、R（reduce/縮小）、SR（small reduce/微縮小）、NC（no change/変化なし）、SI（small increase/微増加）、および I（increase/増加）として記述される。入力変数と出力変数のメンバシップ関数を図 2.6 で示している。

ファジー制御の if-then ルールは、次のとおりである。

Rule 1 (r1) : if (short) and (falling) then R
Rule 2 (r2) : if (close) and (falling) then SR
Rule 3 (r3) : if (long) and (falling) then NC
Rule 4 (r4) : if (short) and (steady) then SR
Rule 5 (r5) : if (close) and (steady) then NC
Rule 6 (r6) : if (long) and (steady) then SI
Rule 7 (r7) : if (short) and (rising) then NC
Rule 8 (r8) : if (close) and (rising) then SI
Rule 9 (r9) : if (long) and (rising) then I

非ファジィ化では、得られた I, SI, NC, SR, R を重みとして、事前に設定された増減係数 $\{N_2, N_1, Z, P_1, P_2\} = \{0.25, 0.5, 1, 1.5, 2\}$ を加重平均し、出力 f を次のように求められる。

$$f = \frac{N_2 \times R + N_1 \times SR + Z \times NC + P_1 \times SI + P_2 \times I}{SR + R + NC + SI + I} \tag{2.1}$$

$$I = \sqrt{r_9^2} \tag{2.2}$$

$$SI = \sqrt{r_6^2 + r_8^2} \tag{2.3}$$

$$NC = \sqrt{r_3^2 + r_5^2 + r_7^2} \tag{2.4}$$

$$SR = \sqrt{r_2^2 + r_4^2} \tag{2.5}$$

$$R = \sqrt{r_1^2} \tag{2.6}$$

b_i はセグメントのビットレートで τ はセグメントの再生時間で、t_i セグメント全体のダウンロード時間とすると、i 番目のセグメントのスループットを

$$r_i = \frac{b_i \times \tau}{t_i} \tag{2.7}$$

のように求めることができる。k セグメントの平均を r_d を

$$r_d = \frac{1}{k} \times \sum_{k}^{i=1} r_i \tag{2.8}$$

のように求めることができ、f が求められたら次のセグメントのビットレート b_{i+1} は

$$b_{i+1} = max\left\{b | b \leq f \times r_d, b \in B\right\} \tag{2.9}$$

選択できるコンテンツビットレート B の中で、$f \times r_d$ 以下で一番高いビットレートである。

2.3　可用帯域推定と MR-BART

ビットレート適応アルゴリズムでは、可用帯域 (available bandwidth) に制限がなければ最高品質のコンテンツを利用できるが、複数の利用者がいる場合に高いビットレートのコンテンツをリクエストするとネットワークリソースへの需要がネットワークの容量を上回り、輻輳状態になりやすい。ネットワーク輻輳によってストリーミング再生が一時停止（playback stall）することが発生し、ユーザーの QoE（quality of experience）の低下につながる。適した品質のコンテンツを選ぶためには、ネットワークの可用帯域を把握することが重要である。

パケットの送信側 (Transmitter) から受信側 (Receiver) までのネットワークパスに複数のノードが含まれている。各ノードに着信リンクと発信リンクを間を繋ぐ一定の長さのキューがあり、First In First Out(FIFO) である。ネットワークパスに複数のリンクがあり、i 番目のリンク L_i の容量 (Capacity) を C_i とする。リンクの容量は物理層の Interface によって決まり、容量 C_i の変化が非常にゆっくりで短い時間スケール (RTT) で見ると固定であると見なせる。リンク L_i には経時変化するクロストラフィック y_i が流れている。一定の観測時間内、リンク L_i の可用帯域 A_i は

$$A_i = C_i - y_i \tag{2.10}$$

で表せる。ネットワークパス全体の可用帯域 A は可用帯域が一番小さいリンクの可用帯域で決まる。

$$A = \min_{l} A_i \tag{2.11}$$

可用帯域が一番小さいリンクのことをボトルネックという。

正確に可用帯域を知るには、ネットワークパス上にあるノードにアクセスし、可用帯域に関する情報を集める必要があり、非現実的である。エンドユーザーにとっては、ノードに直接アクセスせずパケットの送受信情報を用いて可用帯域推定を行う必要がある。

可用帯域の手法にはパケット時間間隔の歪み、遅延の情報を用いる様々なものがある [5][6]。今回注目した MR-BART[7] でパケット時間間隔を用いて可用帯域推定を行う手法について紹介する。

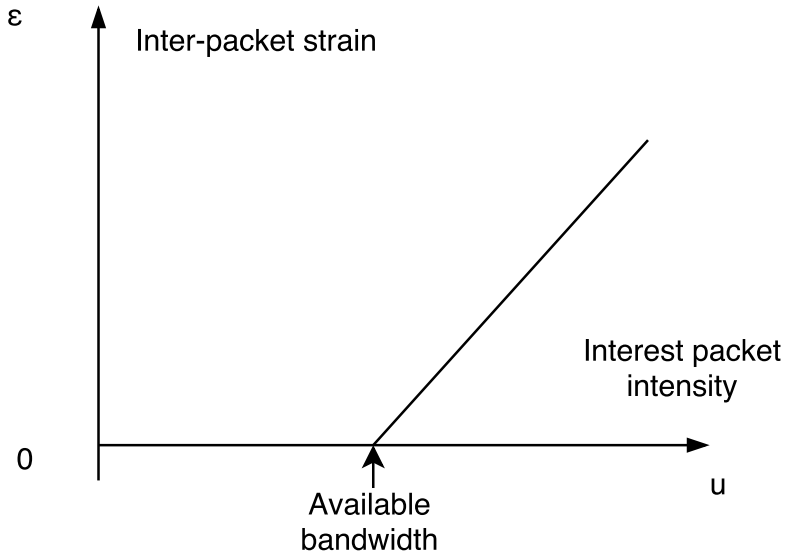

図 2.7: the model of u and ips

送信側から送り出されたパケットの間隔と受信した際のパケットの間隔に歪みが生じるので、その歪みと送信速度の関係で可用帯域を推定できる。送信速度がネットワークパスの可用帯域より小さいときは、パケットに遅延が生じず、歪みは発生しないが、送信速度が可用帯域より大きくなると、パケットがボトルネックリンクに溜まり、遅延が生じるので、パケット時間間隔に歪みが生じる。送信速度を大きくしていき、パケット時間間隔歪みがゼロより大きくなる送信速度で可用帯域を推定できる。そのモデルが図に示している。

送信側から送り出されたパケットの瞬時のレートを u、受信側で受け取ったパケット瞬時のレートを r とする。$u < C - y$ の場合、送受信のレートが変わらないので、つまり $u = r$。しかし $u > C - y$ の場合、ネットワークが輻輳状態になり、u と r の関係は

$$\frac{u}{r} = \max(1, \frac{u+y}{C}) = \begin{cases} 1 & u \leq C - y \\ \frac{1}{C}u + \frac{y}{c} & u > C - y \end{cases} \tag{2.12}$$

で表せる。パケット時間間隔歪みを ϵ とすると、ϵ は

$$\epsilon = \frac{u}{r} - 1 \tag{2.13}$$

と定義する。これによって $u > C - y$ の場合 ϵ は

$$\epsilon = \frac{1}{C}u + \frac{y - C}{C} = \frac{1}{C}u - \frac{A}{C} \tag{2.14}$$

と表せるので、係数を

$$\alpha = \frac{1}{C}; \beta = \frac{y - C}{C} = -\frac{A}{C} \tag{2.15}$$

と置くと、

$$\epsilon = \alpha u + \beta \tag{2.16}$$

になるので、ϵ と u は線形な関係にあるとわかる。$u < C - y$ の場合、u が大きくなるにつれ、パケット時間間隔の歪みも大きくなる。そしてパケットのサイズが大きくなるとパケット時間間隔の歪みも大きくなる。グラフの傾きからボトルネックの容量もわかる。

MR-BART の帯域幅推定は、プローブパケット列を送信する送信端 (transmitter) によって開始され、送信端は異なるプローブ強度のパケットをネットワークに注入する。プローブパケットの長さは約 15−20 パケットであり、プローブパケット列は N 個のグループに分けられ、各グループは M 個のデータパケットを含み、各グループのプローブ強度 $u_k = \{u_1, ..., u_M\}$ は異なる u を使っている。受信端はパケット列を受信した後、各パケットの平均パケット時間間隔歪み ϵ_k を計算し、カルマンフィルタに u_k と ϵ_k を入力して利用可能な帯域幅を推定する。この研究では、MR-BART 帯域幅の推定方法に原理的には似ているが、以下の点が異なる。

- MR-BART がネットワークプローブパケットを使っている。これらのプローブパケットはネットワークオーバーヘッドを増加させる。しかし、本研究では実際のデータパケットの統計情報を用いて帯域幅推定を行い、ネットワークオーバーヘッドの増大を回避している。
- MR-BART 帯域幅推定は、送信側がプローブパケット列を送り、受信側がプローブパケットを受け取った後推定を行う。プローブパケット列の片道のパケット時間間隔歪みにより推定を行っているが、この研究での帯域幅推定は、Consumer によって開始され、実行されるので、往路と復路で発生したパケット時間間隔歪みを用いて推定を行っている。

2.4　カルマンフィルタ

カルマンフィルタ (kalman filter) は誤差のある観測値から動的システムの状態を推定するための逐次計算ベイジアンフィルタである [8]。時間の推移とともに変化する量について、複数の時点 $t = t_1, ..., t_n$ で観測して得られた確率変数列 $y_1, ..., y_n$ のことを時系列という。状態空間モデルでは与えられた時系列の観測値 $y = \{y_1, ..., y_n\}$ に基づいて状態 $\alpha_1, ..., \alpha_n$ を推定し、将来の観測値を予測あるいは欠陥値を補間することが主要な目的となる。フィリタリングは途中の時点 t までの観測値 $Y_t = \{y_1, ..., y_n\}$ を用いて当該時点の状態 α_t を推定するものである。線形ガウス状態空間モデルフィルタリングのための効率のよいアルゴリズムとして知られているのがカルマンフィルタである。カルマンフィルタは初期状態 α_1 の平均 a_1 および分散 P_1 から出発し、各時点の状態 α_t の一期先予測 $a_t = (\alpha_t |\, Y_{t-1})$ とその予測誤差分散 $P_t = Var\,(\alpha_t |\, Y_{t-1})$ を事前確率分布として、新しい観測値が観測されると、事後確率である状態のフィルタ化推定量 $a_{t|t} = E\,(\alpha_t |\, Y_t)$ とその推定誤差分散 $P_{t|t} = (\alpha_t |\, Y_t)$ を交互に求めていく。線形ガウス状態空間モデルは次のように定義される。

$$\begin{aligned} y_t &= Z_t \alpha_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, H_t) \\ \alpha_{t+1} &= T_t \alpha_t + R_t \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q_t), \quad t = 1, ..., n \end{aligned} \tag{2.17}$$

上の式は観測方程式と呼ばれ、観測値を観測されない潜在変数 α_t に正規ホワイトノイズ ε_t が乗っていることを表している。観測されない真の水準 α_t は状態（state）と呼ばれ、観測誤差 ε_t は観測値撹乱項とも呼ばれる。下の式は状態方程式と呼ばれ、状態 α_t が状態撹乱項 η_t が乗って、更新されていくことを表す。

第3章

問題点

従来のMPEG-dashのビットレート適応アルゴリズムでは過去のセグメントの平均スループットに基づき、ネットワークの可用帯域を推定を行っているが（式2.8)、その推定の精度が低く、可用帯域を過少評価する傾向がある（図5.13)。そのためビットレート適応アルゴリズムも低いコンテンツビットレートを選択することになる。ネットワーク可用帯域許容範囲内でもっとも高品質のコンテンツを取得するために、正確に可用帯域を推定することが重要である。

MPEG-dashではTCPの輻輳制御アルゴリズムとMPEG-dashのビットレート適応アルゴリズムが協調して同時に働くことで初めて正しく動作することができる。輻輳制御アルゴリズムがInterestの送信速度 (delivery rate) をコントロールし、ネットワーク輻輳によるパケットロスと遅延を防ぎ、ビットレート適応アルゴリズムは次のセグメントの最適なビットレートを決定している。TCPの輻輳制御アルゴリズムがネットワーク輻輳が起こらないように送信速度を決めた上で、ビットレート適応アルゴリズムが働いている。

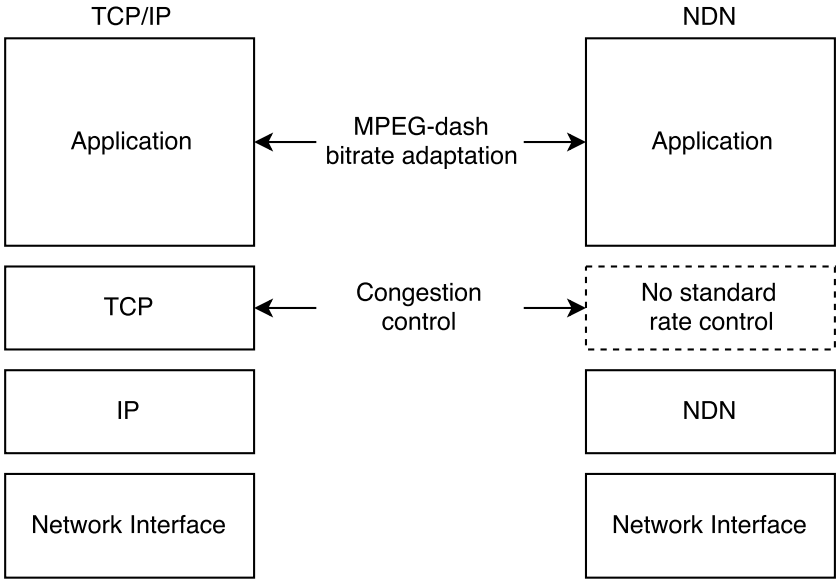

図3.1: TCP/IP and NDN network stack comparsion

ほかに NDN ネットワークには標準の輻輳制御アルゴリズムがないため、トランスポート層では送信速度は制御されておらず、Interest 送信速度制御の実現は各アプリケーションに任される。ビットレート適応アルゴリズムは、次のセグメントのビットレートを推定するために、過去のセグメントの平均スループットに基づいているので、基礎となる速度制御アルゴリズムは存在しなければ、ビットレート適応アルゴリズムも機能しない。ビットレート適応アルゴリズムはネットワーク輻輳を引き起こすことなくネットワーク帯域幅リソースを最大限に活用するという前提の下で働いているが、NDN におけるそのような前提は成り立たないことは明らかである。要するに、NDN には、基礎となる送信速度を制御するための方法が欠けており、過去のビットレート適応アルゴリズムを NDN フレームワークに適用することができない。

そこでこのような問題を解決するため、ビットレート適応アルゴリズムで Interest 送信速度をコントロールすることを試みた。ビットレート適応アルゴリズムでリクエストするセグメントのビットレート R を決めたら、そのセグメントを取得するのに必要な Interest の個数 m も決まる。m 個の Interest をセグメントの再生時間 T_{seg}(2 seconds) より少し短い時間内に均等に送り出すように Interest 送信速度を制御する。Interest の送信速度は $f = g \times \frac{m}{T_{seg}}$ になる。g は送信速度のゲインで、一定のゲインを設けることによって Interest 送信速度をセグメントのビットレートより高く設定した。ネットワークが輻輳していなければ、$g^{-1} \times T_{seg} + t_{rtt}$ で次のセグメントのダウンロードが完了する。次にビットレート適応アルゴリズムを適用して、ダウンロード時間や平均スループットで次のセグメントのビットレートを決める。

しかし、この方法ではネットワーク帯域リソースを十分活用できず、低いビットレートのコンテンツを選んでしまうという欠点がある。Data producer 側で事前に用意されているコンテンツのビットレートは連続的ではなく離散的な値で、選択したビットレートは一段上または下のビットレートと大きく離れている可能性もある。これによってセグメントダウンロード時間の短縮や平均スループットの上昇したことによって、次のセグメントが必ず一段上のビットレートにシフトするとは限らない。それにビットレート適応アルゴリズムはビデオプレイヤーのバッファレベルなどの情報も考慮して、適したビットレートを選んでいる。セグメントのダウンロード時間はその再生時間より長くなると、ビデオプレイヤーのバッファレベルが減少するので、セグメントのダウンロード時間が再生時間より長くても、ビットレート適応アルゴリズムはプレイヤーバッファレベルがゼロにならないことを保証しなければならない。そのような制約の上で、ビットレート適応アルゴリズムができるだけ高いビットレートを選んでいる。この状況ではセグメントのダウンロード時間や平均スループットが多少変化しても、ビットレート適応アルゴリズムが一定のビットレートを維持したままである。ビットレート適応アルゴリズムが違うビットレートにシフトしなければ、Interest 送信速度も変わることがないので、この手法ではネットワークの可用帯域の変化にうまく追従できず、Interest 送信速度の調節に柔軟性が欠けていることがわかった。

このような経験から上位層のアプリケーション層からトランスポート層の Interest 送信速度を制御するのではなく、トランスポート層で Interest 送信速度を決めた上でアプリケーション層でビットレート適応を行うべきことがわかった。ほかに次のセグメントのビット

レートは当該時点での可用帯域によって決めなければならない。前のセグメントのダウンロード所用時間や平均スループットでは正確にネットワークの可用帯域を推定することはできない。輻輳制御アルゴリズムもネットワークの実際の可用帯域によって Interest 送信速度を制御するので、正しくネットワークの可用帯域を推定することはメディアストリーミングのような大容量データ通信プロトコルでは必要不可欠ということが言える。本研究ではトランスポート層の可用帯域推定に基づいたレート制御アルゴリズムを提案し、過去のセグメントの平均スループットのかわりに推定した可用帯域を用いてビットレート適応を行うことで NDN でのメディアストリーミングシステムを提案した。

第4章

提案システム

本研究では上の章で紹介された問題点を解決するために、NDN ネットワークアーキテクチャにおいて利用可能な帯域幅を推定する方法を提案し、この帯域幅推定手法に基づき、トランスポート層の Interest 送信速度を制御する方法を提案する。そして、提案された手法に基づき、前のセグメントの平均スループットの代わりに推定された可用帯域を用いることで、従来のビットレート適応アルゴリズム Fdash を NDN フレームワーク上に適用させることができた。システムの全体像が図 4.1 に示されている。

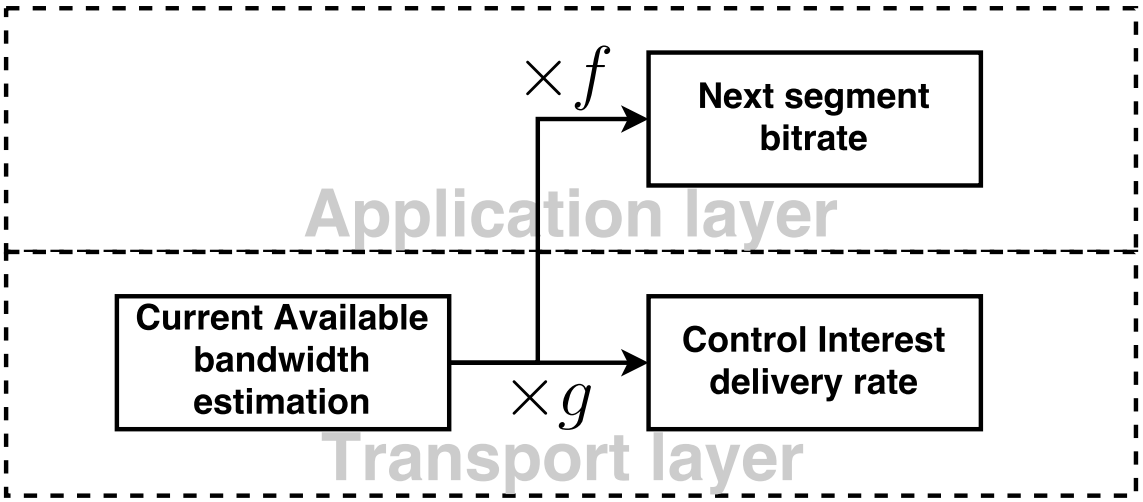

図 4.1: the overall picture of the system

4.1 可用帯域推定手法

この章では、本研究で提案された NDN フレームワークにおいて Interest および Data パケットの時間間隔歪みに基づき、カルマンフィルターを用いて利用可能なネットワーク帯域幅を推定する手法を紹介する。帯域幅推定方法のプロセスは図 4.2 に示されている。u_k

は予定された Interest の送信速度である。各 Data が受信された後、パケットの時間間隔の歪み ϵ_k を計算するために、ある時間ウィンドウ t_w 内の Data パケットの時間間隔 Δ_{out}、対応する Interest の送信時間 Δ_{in} を計算し、そして Δ_{in} から実際の平均送信速度 $\widehat{u}_k$ を計算する。$\widehat{u}_k$ および ϵ_k はカルマンフィルタに入力として入力され、モデルの状態を更新して利用可能な帯域幅の新しい推定値を得る。 利用可能な帯域幅の推定値を取得したら、次の Interest 送信速度 u_{k+1} をランダムのゲイン $g \sim U(g_{min}, g_{max}); g_{min} > 1$ をかけて利用可能な帯域幅よりもわずかに速く設定する。 Interest 送信速度の設定は利用可能な帯域幅よりも速いが、同時に Interest 送信速度は帯域幅遅延積によって制限されるため、帯域幅推定のための最小量の輻輳しか起こさない。またこのため u_k と $\widehat{u}_k$ は同じではない。 そのようなプロセスで、Interest 送信速度を制御し、ネットワークのボトルネックに輻輳を起こさせ、リアルタイムに利用可能な帯域幅を推定している。

$$u_{k+1} = g \times A_k$$

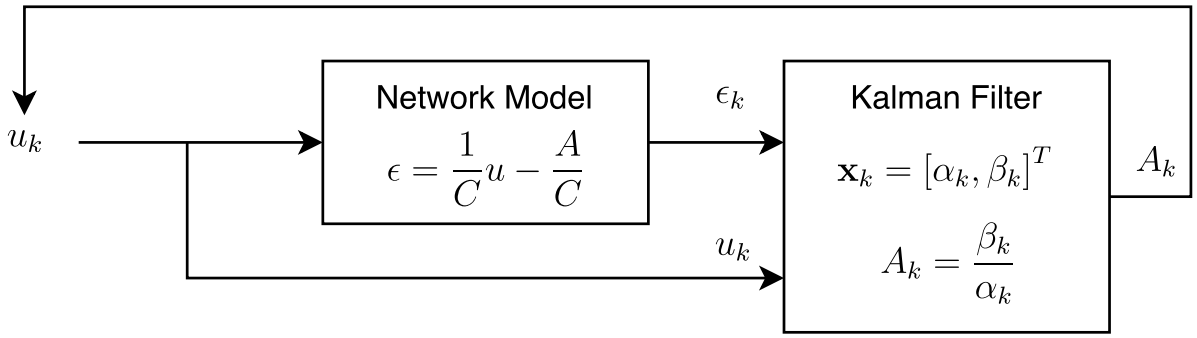

図 4.2: the estimation process of available bandwidth

4.2　パケット時間間隔歪みの計測

パケット時間間隔歪み（inter-packet strain、IPS）を計測するために、Interest を送り出す時間 $T = \{t_i | i = 1...N\}$ と、Data が到着した時間 $T' = \{t'_i | i = 1...N\}$ を記録する。新しい Data が到着するたび一定時間ウィンドウ t_w 内にある Data の記録 $T'_w = \{t'_i |\ t'_i > t_{now} - t_w\}$ 以外の古い観測データを棄却する。複数の Interest と Data の平均 IPS を求めるために、k 番目の Data の到着時間 t_k と k + M 番目の Data の到着時間 t_{k+M} の平均時間間隔

$$\Delta_{out} = \frac{1}{M}\sum_{T'_w}(t'_{i+1} - t'_i) = \frac{1}{M}(t'_{k+M} - t'_k) \tag{4.1}$$

対応する Interest の平均時間間隔は

$$\Delta_{in} = \frac{1}{M}\sum_{T_w}(t_{i+1} - t_i) = \frac{1}{M}(t_{k+M} - t_k) \tag{4.2}$$

パケット時間間隔歪み IPS は

$$\epsilon = \frac{\Delta_{in}}{\Delta_{out}} - 1 = \frac{t_{k+M} - t_k}{t'_{k+M} - t'_k} \tag{4.3}$$

になる。そして Data のパケットサイズを L とすると実際の Interest 送信速度 $\widehat{u}$ は

$$\widehat{u} = \frac{M \times L}{\Delta_{in}} \tag{4.4}$$

で計算できる。ここで注意したいのは Data のサイズはすべて固定であると仮定し、payload

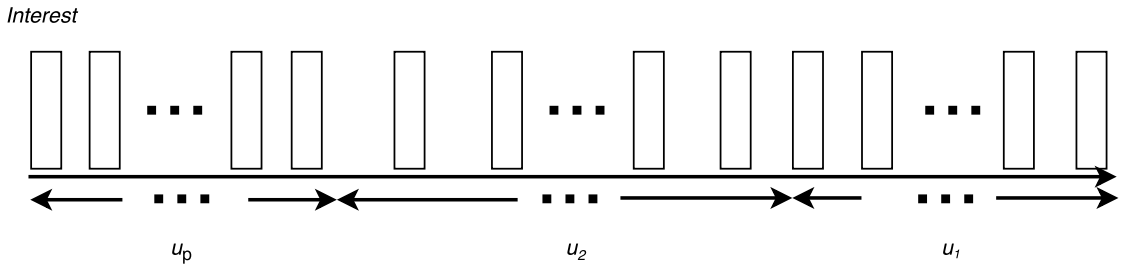

図 4.3: the interest sequence

size は 8000 バイトで、ヘッダを加えると 8032 バイトになる。そして packet 間間隔の歪みは Data が送り返される復路で生じたもので、Interest のサイズは Data と比べると非常に小さいので往路で発生した歪みは無視できるとしている。つまり、Interest が producer に到着した時のレートは u に等しい。そして Producer 側での処理時間も一定で、帯域推定には影響がでないものと仮定している。得られた u と ϵ はカルマンフィルターに入力し、帯域推定を更新していく。

4.3　カルマンフィルターの設計

この section ではカルマンフィルターを用いて、観測されたパケット時間間隔歪み ϵ と Interest delivery rate u から可用帯域推定を行う方法を紹介する。カルマンフィルターの設計は MR-bart[7] の設計を参考にしている。

前章で得られたら u と ϵ を p 個を 1 組として、$\mathbf{U}_k = \{(u_1)_k, ..., (u_p)_k\}$ と $\mathbf{E}_k = \{(\epsilon_1)_k, ..., (\epsilon_p)_k\}$ を入力としてフィルターに入れ、フィルタの一期先予測と観測値の差を計算し、フィルタの更新を行い、新しいシステム状態のフィルタ化推定量を得る。ネットワークのパスを線形システムとモデル化し、ネットワークリンクの容量 C と可用帯域 A をシステムの状態とする。u と ϵ の式 2.16 の係数、α, β を用いて推測の k 番目のステップでのシステムの状態ベクトルを

$$\mathbf{x}_k = [\alpha_k, \beta_k]^T \tag{4.5}$$

と定義する。状態方程式を

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} \tag{4.6}$$

と線形関数 $f()$ とノイズ $\mathbf{w}_{k-1}$ を用いて書くことができる。ネットワークの状態は各ステップで変化しないと仮定するので、

$$f(\mathbf{x}_k) = \mathbf{x}_k \tag{4.7}$$

を使用する。観測方程式を

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{4.8}$$

と定義する。$\mathbf{z}_k$ は観測されたパケット時間間隔歪みの $P \times 1$ のベクトルで、

$$\mathbf{z}_k = [(\epsilon_1)_k, (\epsilon_2)_k, ..., (\epsilon_p)_k]^T \tag{4.9}$$

各ステップ p 組の u と ϵ を入力する。ここで h() は

$$h(\mathbf{x}_k) = \mathbf{H}_k\mathbf{x_k} \tag{4.10}$$

$\mathbf{H}_k$ は

$$\mathbf{H}_k = \begin{bmatrix} (u_1)_k & 1 \\ (u_2)_k & 1 \\ \vdots & \vdots \\ (u_p)_k & 1 \end{bmatrix} \tag{4.11}$$

である。したがって $h(\mathbf{x}_k)$ は

$$h(\mathbf{x}_k) = \mathbf{H}_k\mathbf{x}_k = \begin{bmatrix} \alpha_k(u_1)_k + \beta_k \\ \alpha_k(u_2)_k + \beta_k \\ \vdots \\ \alpha_k(u_p)_k + \beta_k \end{bmatrix} \tag{4.12}$$

になる。そしてモデルのノイズは

$$\mathbf{v}_k \sim \mathcal{N}(0, R); \quad \mathbf{w}_k \sim \mathcal{N}(0, Q) \tag{4.13}$$

と仮定する。$\mathcal{N}(0, \theta)$ は平均 0 で共分散行列 θ の正規分布を表す。Q と R はそれぞれ状態遷移ノイズと観測ノイズの共分散行列である。$P \times P$ の行列 R は計測されたパケット時間間隔歪みの分散を用いて

$$R = \begin{bmatrix} (R_1)_k & 0 & \cdots & 0 \\ 0 & (R_2)_k & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & (R_p)_k \end{bmatrix} \tag{4.14}$$

と書くことができる。そして Q は 2×2 の行列で、システム内部の変動を表している。α と β はお互い独立なので、Q は

$$Q = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \tag{4.15}$$

と書くことができる。ここで λ は設定できるパラメータで、実験では 0.001 に設定している。

次にフィルタ更新の手順について見てみる。k 番目のステップでモデルの状態が $\mathbf{x}_k = [\alpha_k, \beta_k]^T$ で、$\mathbf{U}_k = \{(u_1)_k, ..., (u_p)_k\}$ と $\mathbf{E}_k = \{(\epsilon_1)_k, ..., (\epsilon_p)_k\}$ が入力として与えられたら、k+1 番目のステップの状態を $\mathbf{x}_{k+1} = [\alpha_{k+1}, \beta_{k+1}]^T$ を以下のように計算する。まずは観測値とモデルの予測の誤差

$$\mathbf{v}_k = \mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k \tag{4.16}$$

と一期先予測誤差

$$\mathbf{F}_k = \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{Q}_k \tag{4.17}$$

を求める。
観測値が与えられたらモデルの予測を修正するための係数カルマンゲインを

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}_k\mathbf{F}_k^{-1} \tag{4.18}$$

と計算することができる。
フィルタ化推定量 $\mathbf{x}_{k|k}$ と推定誤差分散 $\mathbf{P}_{k|k}$

$$\mathbf{x}_{k|k} = \mathbf{x}_k + \mathbf{K}_k\mathbf{v}_k \tag{4.19}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_k - \mathbf{K}_k\mathbf{F}_k\mathbf{K}_k^T \tag{4.20}$$

が得られる。次に状態方程式を用いて次のステップの一期先予測 $\mathbf{x}_{k+1}$ と予測誤差分散 $\mathbf{P}_{k+1}$ を求める。

$$\mathbf{x}_{k+1} = \mathbf{x}_{k|k} \tag{4.21}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_{k|k} + \mathbf{R} \tag{4.22}$$

この時の可用帯域は $\mathbf{x}_{k+1} = [\alpha_{k+1}, \beta_{k+1}]^T$ を用いて

$$A_{k+1} = \frac{\beta_{k+1}}{\alpha_{k+1}} \tag{4.23}$$

と可用帯域の推定値が得られる。

4.4　アプリケーション設計

可用帯域推定やビットレート適応アルゴリズムを評価するために、メディアストリーミング再生アプリケーションのクライアント側とサーバー側を実装した。

クライアントプログラムには、NDN consumer、streaming controller および MPEG-player の 3 つのコンポーネントから構成されており、3 つのコンポーネントにさらに様々なモジュールが含まれている。

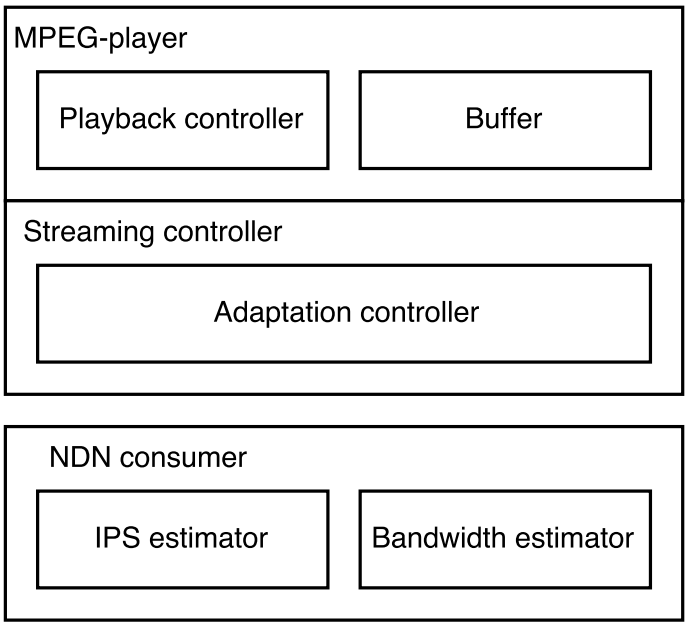

図 4.4: application structure

NDN consumer には、クライアントプログラムの最も基本的な機能、Interest 送信、Data 受信、タイムアウトした時に Interest の再送、受信 Data のリシーケンシングなど NDN の標準機能に加え、トランスポート層機能として、パケット間歪み推定器 (inter-packet strain estimator) と帯域幅推定器 (bandwidth estimator) の 2 つのモジュールを追加した。

パケット間歪み推定器は Interest/Data の送受信時間を管理し、パケット間歪み ϵ と Interest の実際の送信速度 $\hat{u}$ ををを推定し、それらの情報に基づき、帯域幅推定器がカルマンフィルターを用いてネットワークの可用帯域幅推定を行う。

streaming controller は、メディアストリーミングを制御している。 streaming controller は、periodID および adaptation set の ID を含む、取得すべき次のメディアセグメントの Interest 名を構築する。 streaming controller は、Interest 送信シーケンス番号を制御し、シーケンス番号が現在のセグメントの最大シーケンス番号 seqMax に等しい場合、前のセグメントの Interest がすべて送信されたことを示す。adaptation controller をアクティブにさせ、次のセグメントの representation を計算。

その後、adaptation controller は、使用されるさまざまな統計に基づいてビットレート適応を行い、用意されている選択可能なビットレートから、次のメディアセグメントの representation を求める。異なるビットレート適応アルゴリズムでは、通常、最後のビデオセグメントの平均スループット、バッファに残っているビデオ再生時間、推定された使用可能な帯域幅など、さまざまな統計情報が必要で、adaptation controller は対応するモジュールから必要な統計を取得する。たとえば、現在のネットワーク帯域幅推定値を bandwidth estimator から、現在のプレーヤーのバッファリング情報を MPEG-player から取得する。

Data を受信した後、まずはシーケンス番号で順番通りに並べ替え、ペイロードを取り出しフレーム解析用のバッファに入れ、Mpeg ヘッダを解析する。 MPEG ヘッダにフレーム

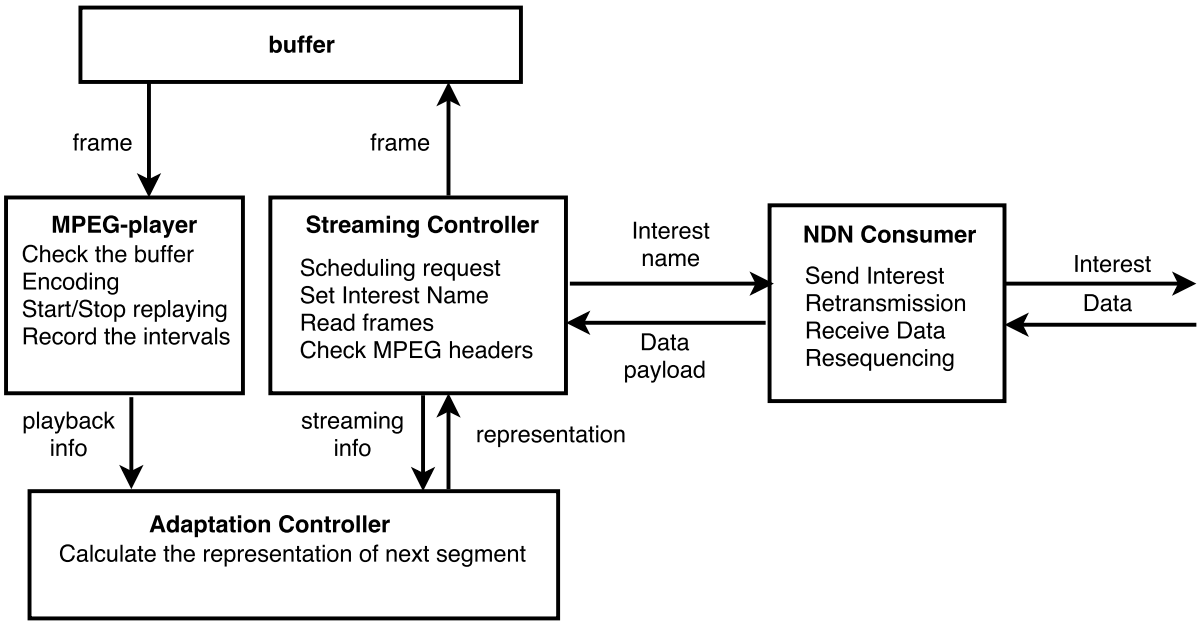

図 4.5: the client model

ID、フレーム長、およびフレームのタイプなどフレームのメタ情報が含まれている。フレームのメタ情報を得るために Mpeg ヘッダの内容を解析した後、フレーム長の長さのビットストリームをバッファの先頭から取り出し、フレームとして MPEG-player に渡す。バッファに残っているデータの量が 1 フレーム未満である場合、しばらく待機し、次の Data が到着した後にまたフレーム解析を再開する。

MPEG-player はストリーミング再生を制御している。MPEG-player は streaming controller からフレームを受け取ったらバッファリングキューに入れ、動画が一時停止している場合は再生を開始する。$\frac{1}{f}$(f:フレームレート) 秒ごとにバッファリングキューをチェックし、空であればビデオ再生を一時停止し、空でなければ 1 フレームを取り出し、再生する。また、MPEG-player は、ユーザーの満足度を評価するためにビデオが一時停止した回数および継続時間、ならびにビットレートが変化した回数などを記録する。

Producer 側プログラムは、異なる目標ビットレートのコンテンツファイルを、対応する MPD ファイルを使用してを管理する必要がある。 MPD ファイルに動画を一意に識別できる VideoID、period の開始時間や長さ、adaptation set、representation など、コンテンツファイルのメタデータ（meta data）が記述されている。一方、動画ファイルの各フレームを符号化した後、フレーム、フレーム長、及び情報フレームの種類の ID を含んでいる MPEG ヘッダをフレームに追加し、MPEG-header をつけたフレームを連続に並べ、Data のペイロードサイズに等しい長さに再分割される。Data のペイロードサイズは事前に設定される。本システムでは 8000 バイトに設定した。各セグメントは、異なる符号化方法および目標ビットレート決定によって必要な Interest/Data も異なる。簡単にするために、各 Data パケットはすべて同じサイズで、許容される最大長である同じサイズであると仮定した。Data の長さは、使用可能な帯域幅を計算するときにも使われる。

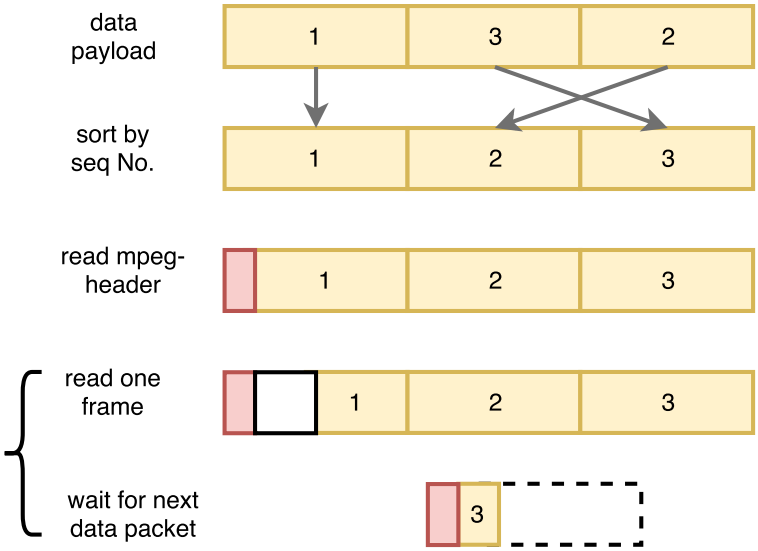

図 4.6: Consumer-side frame processing : analysis mpeg-header and read frames

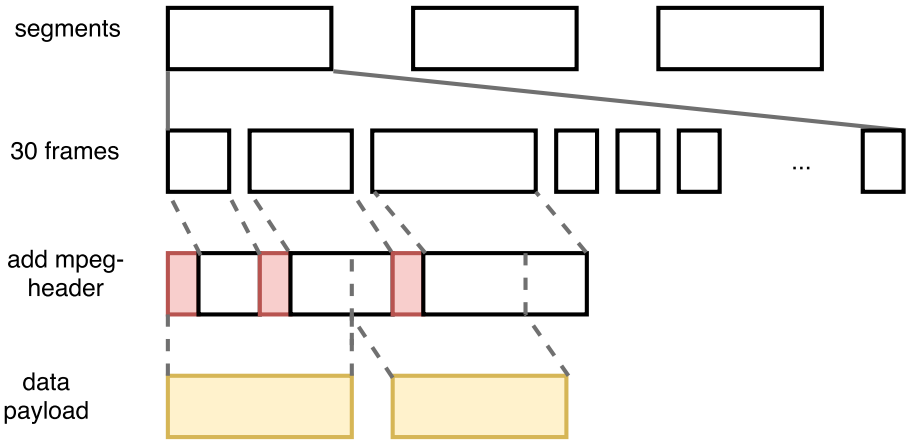

図 4.7: Server-side processing : Data assembling

4.5　Interest の命名規則

本研究でマルチメディアコンテンツを NDN で送受信するために Interest と Data にコンテンツの内容を記述する名前をつけるけれども、その命名規則を紹介する。この命名規則は MPEG-dash でマルチメディアコンテンツを階層的な分割法を参考にした。NDN でも同じような階層構造で Interest と Data を命名することによって MPEG-dash とも互換性を保つことができ、自然なやり方と考えられる。この命名規則においてひとつのマルチメディアコンテンツファイルを 5 つの階層に分けることができる。

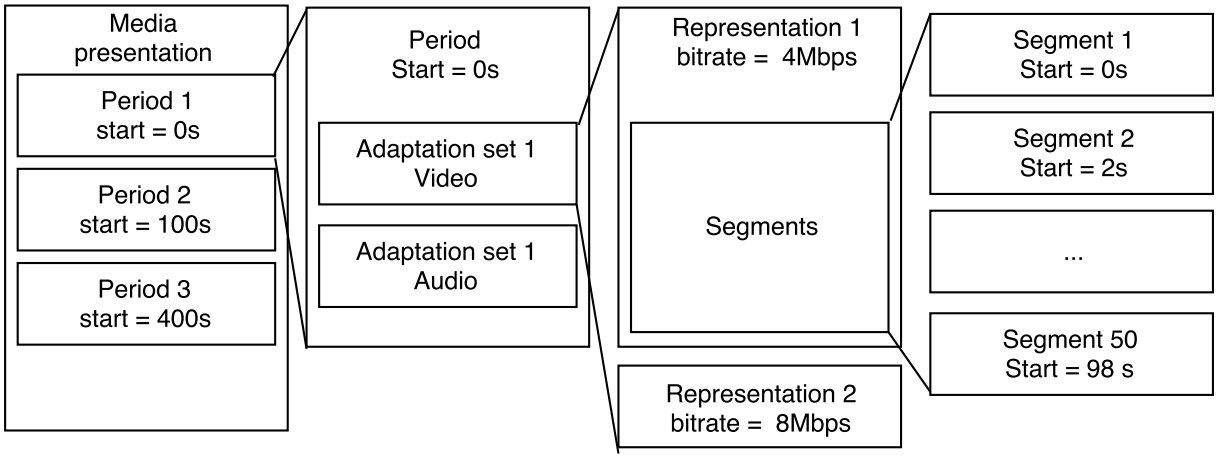

図 4.8: the structure of naming scheme

presentation

presentation は一番上層の階層で、ひとつのマルチメディアファイル全体を表している。この階層ではメディアファイルにユニークな VideoID をつけ管理している。

period

period はマルチメディアファイルを再生時間で分割し、動画の異なるチャプターやシーンを別 period に分けることができる。それぞれの period に開始時間と長さをメタ情報として MPD ファイルに記述し、メディアコンテンツの構成をわかりやすくすることができる。

adaptation Set

adaptation set は、マルチメディアファイルを構成する、動画、異なる言語の音声や字幕などを表現し、たとえば一つのマルチメディアファイルは動画コンテンツ、英語版音声トラック、中国語字幕などの adaptation set から構成される。

representation

representation は、動画のエンコード方式、ビットレートや解像度などを表現している。ビットレート適応にもっとも重要な情報である。MPD ファイルに記述し、アプリケーション側で解析することによってどのようなコンテンツが用意されているかがわかる。

segment

segment は 2 秒長さのメディアクリップで、ビットレート適応アルゴリズムは segment 単位で行われる。

たとえば、VideoID:2Df45A の英語版動画ファイルの一番目の period、目標ビットレートは 24Mbps で、その一番目の segment は次のように命名することができる。

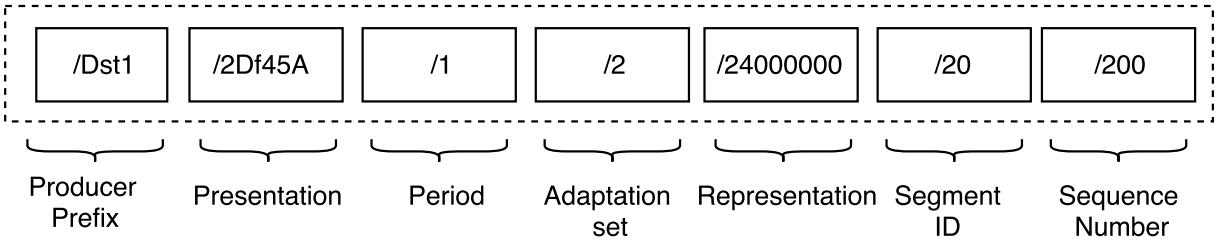

図 4.9: naming scheme

このような命名方式で segment 単位に分割されたマルチメディアコンテンツのすべての segment に一意な名前をつけることができ、新しい period、representation、segment が追加されても簡単に名前をつけることができる。Consumer 側からも解析された MPD ファイルから必要なコンテンツのメタ情報を取得し、階層的な名前を構成して、コンテンツをリクエストすることができる。

4.6　Interest 送信速度 u の選び方

前回の章で紹介したように、Consumer 側からの Interest 送信速度 u の大きさを調整することができる。利用可能な帯域幅に応じて u を選ぶ方法は異なる。u が利用可能な帯域幅よりも小さい場合測定 inter-packet strain は 0 に等しくなるので、利用可能な帯域幅とリンク容量の大きさを見積もるための必要な情報が含まれていない。入力として利用可能な帯域幅 A よりわずかに大きい u が望ましいと思われる。しかし、利用可能な帯域幅よりも大きい u を使用することは、観測するためにネットワークの輻輳を招く可能性があることを意味する。

本研究では、利用可能な帯域幅よりも大きい u を選択するために、推定された利用可能な帯域幅に倍数

$$g \sim U(g_{min}, g_{max}); g_{max} > 1 \tag{4.24}$$

をかけ、ランダムに Interest 送信速度を変化させている。ランダムに選んだ u はより豊かな統計情報を提供できると考えられる。g_{min}, g_{max} は Interest 送信速度を制御するパラメータで、可用帯域幅推定のために $u > C - y$ になるように u を選ぶ必要があるので、g_{max} は 1 より大きい値を設定する。g_{min} も 1 より大きい値を設定すると帯域推定の精度が上がるが、ボトルネックリンクも輻輳状態になるので、帯域幅遅延積を用いて Interest 送信速度を制限することが必要である。g_{min} を 1 より小さい値を設定すると輻輳状態になりにくい

が帯域推定の精度も落ちるというデメリットがある。シミュレーションではConsumerが一つの場合 $g_{min}, g_{max} = 1.2, 1, 4$、複数の場合は $g_{min}, g_{max} = 0.9, 1, 1$ に設定した。

4.7　パケットペアを用いた初期化

セクション5.1の数値実験は、カルマンフィルタが、初期値が小さいときに推定値を正しい利用可能な帯域に収束させるために多くのステップを必要とすることを示している。最初の収束を高速化し、より適切な初期値を設定するため、本研究では、パケットペア技術[6]を使用して利用可能な帯域幅を推定し、その推定結果をカルマンフィルタの初期値として使用する。具体的な方法は、プログラムの冒頭において、2番目のInterestは、1番目のInterestが送信された直後に送信する。最初と2番目のDataの到着後に、2つのDataの間の到着時間差 Δ_{out} とDataのサイズ L に基づいて、利用可能な帯域幅 A_1 を計算する。

$$A_1 = \frac{L}{\Delta_{out}} \tag{4.25}$$

得られた利用可能な帯域幅 A をカルマンフィルタの初期値として使用する。

$$\mathbf{x}_1 = \left[-\frac{1}{A_1}, 1\right]^T \tag{4.26}$$

4.8　帯域幅遅延積によるInterest送信速度制御

Inflightは、送信されたがまだ受信されていないパケット、つまりネットワーク伝送路上で送信中のパケットの量を指している。帯域幅遅延積（bandwidth-delay product,BDP）は、BDPは、ネットワークボトルネックリンクの帯域幅と、ネットワークが輻輳していないときの遅延RTpropとの積で、ネットワークリンクが収容できる最大データ量を表している。RTT_t は $RTprop_t$ にキューイング遅延などのノイズ η_t が乗ったものである[9]。

$$RTT_t = RTprop_t + \eta_t \tag{4.27}$$

Inflightがネットワーク内の帯域幅遅延積より大きい場合、パケットはボトルネックにバックログを起こし、ネットワーク遅延が発生する。Consumerは、一定の時間ウィンドウ W_R 内でInterest / Dataの往復時間（RTT）を評価して、時間ウィンドウ内の最小RTT、RTtropを得る

$$RTprop = min(RTT_t), \forall t \in [T - W_R, T] \tag{4.28}$$

ネットワーク帯域幅AとRTtropを計算した後、帯域幅遅延積を計算し、

$$BDP = A \times RTprop \tag{4.29}$$

ネットワークの利用可能な帯域幅に応じてInterest送信速度を算出した後、InflightとBDPを比較し制御する。BDPよりInflightより大きくなると、Interestの送信を遅らせ、次の送信タイミングまで待ってInterestを送信する。実際の送信速度 $\hat{u}$ は算出した u よりも小さいことがわかる。上記のように、consumerは、ネットワークの利用可能な帯域幅とRTTに従って、Interestの送信速度を制御している。

4.9　推定可用帯域を用いたビットレート適応

推定された利用可能帯域幅の結果を Interest 送信速度制御だけでなく、ビットレート適応にも使用した。従来のビットレート適応アルゴリズムでは前のセグメントの平均スループットを用いて可用帯域幅推定を行っているが、本研究の可用帯域幅推定方法は、各データパケットが到着した後に利用可能な帯域幅の推定値を更新するので、各セグメントごとに利用可能な帯域幅を推定するよりも正確で高速であると考えられる。さらに従来のビットレート適応における前のセグメントの平均スループットの代わりにリアルタイムに推定された可用帯域幅を使用することで、すべての Interest が送信された後、セグメントが完全にダウンロードされていなくてもビットレート適応アルゴリズムを実行でき、次のセグメントのビットレートを計算できる。図 4.10 にビットレート適応を行うタイミングを示している。

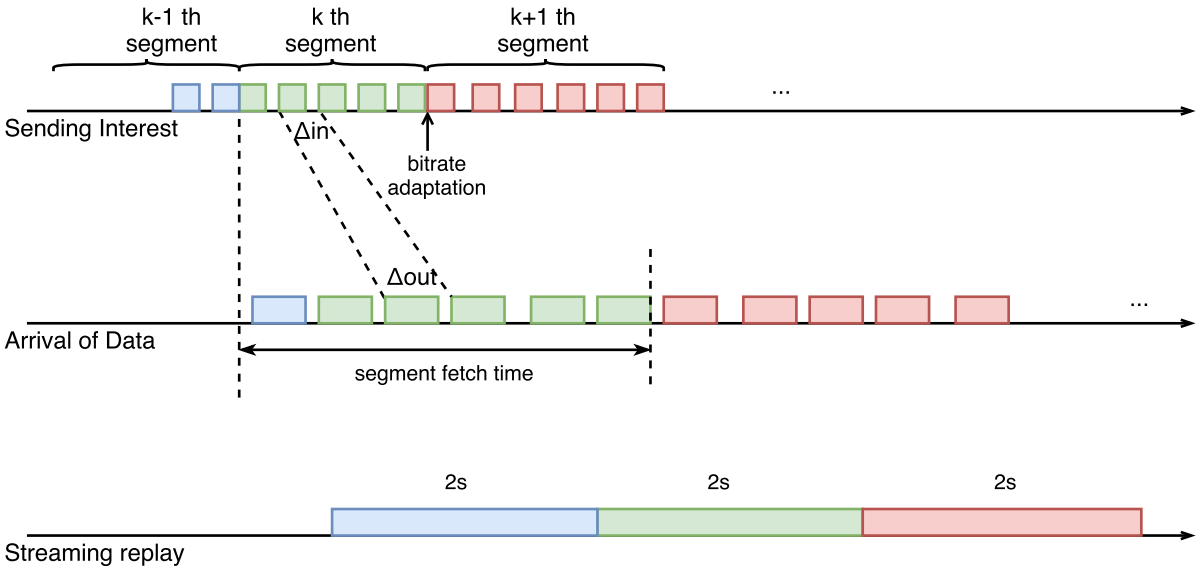

図 4.10: time chart of the system

第5章

評価実験

5.1 数値シミュレーション

利用可能な帯域幅の変更が続く場合の追従性など、可用帯域推定の精度を考察するために、まず、モデルに基づいていくつかのテストデータを生成し、フィルターにテストデータを入力して、フィルターが使用可能な帯域幅の変化を再現できるかどうかを確認する数値シミュレーションを行った。u の値を与えられた後、モデルに従ってあらかじめ設定された使用可能な帯域幅と容量から、対応する ips を計算する。カルマンフィルタに u と ips の値を入力して可用帯域幅の予測を出す。可用帯域幅の予測を得られたら、その予測に一定のゲイン $g \sim U(g_{min}, g_{max}); g_{max} > 1$ をかけたものを次のステップ入力 u とする。このように予測された可用帯域幅を、設定された可用帯域と比較する。テスト 1 では、設定リンク容量はずっと 1 で、最初の可用帯域は 1 で、30 ステップ後に 0.5 に減少し、また 30 ステップ後に 1 に戻る。この変化は図 5.1 の real abw に示される。テストで $g_{min}, g_{max} = 1.1, 1.4$ に設定した。テスト 2 ではフィルターの初期値を小さめ (0.001) に設定した。乱数の種を変え、複数回シミュレーションを行った。

数値シミュレーションの結果が図 5.1, 図 5.2 で示されている。図 5.1 では最初の立ち上がりが早く、30 ステップ後にほぼ可用帯域に収束した。初期値を小さめに設定したテスト 2 では、最初の立ち上がりが遅く、30 ステップ後もまだ可用帯域の予測誤差が大きかった、ステップ 30 60 の時でも大きく過少評価する傾向が見られる。数値シミュレーションからフィルターの初期値設定は可用帯域推定の速度に大きく影響が出ることがわかったので、アプリケーションではパケットペア帯域推定で得られた可用帯域を初期値として使っている。

5.2 NS-3 でのシミュレーション

5.2.1 帯域幅推定手法の評価

ns-3 は、インターネットシステム用の離散事象ネットワークシミュレータであり、主に研究や教育用に使用されている。ns-3 のもとで作られた ndnSIM[10] モジュールを使用すると、NDN ネットワークフレームワーク上でのシミュレーションが可能になる。提案シ

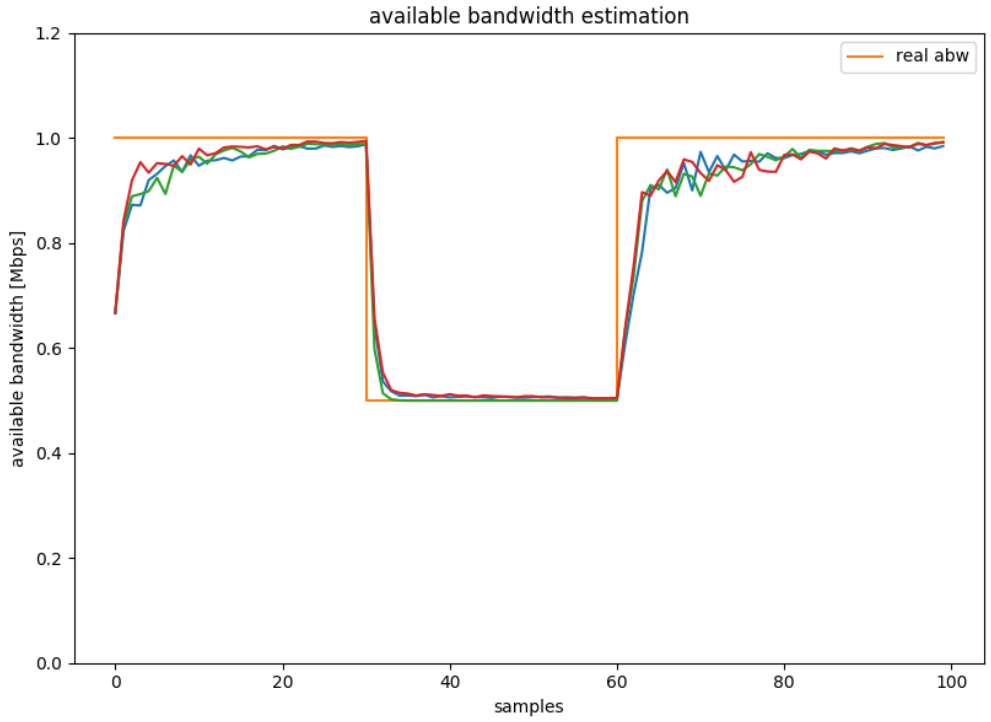

図 5.1: bandwidth estimation test

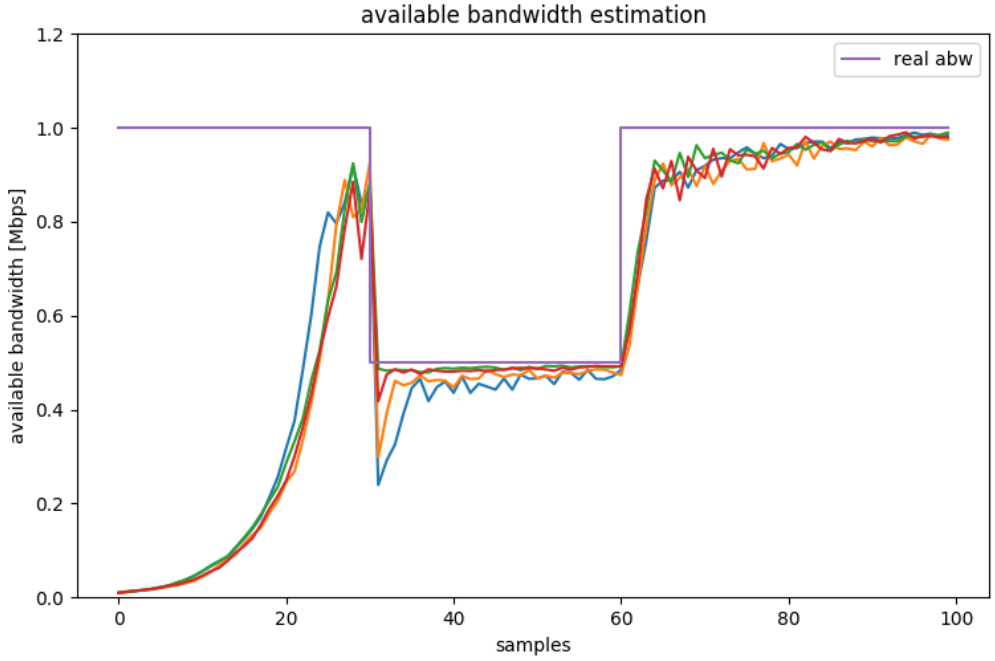

図 5.2: bandwidth estimation test with small initial value

ステムの性能を検証するために、ns-3 でシミュレーションシナリオを作り、ネットワークトポロジを構築し、シミュレーションを行う。シミュレーションで使われるネットワークトポロジには 6 つのノードが含まれる。 最も左の 2 つのノード src1 と src2 は、クライアントとクロストラフィックジェネレータのノードで、2 つの中間ノード rtr1 と rtr2 は 2 つのコンテンツルータで、2 つのコンテンツルータ間のリンクはボトルネックリンクである。実験のニーズに応じてボトルネックリンクの容量を変更できる。 右側の 2 つのノード dst1 と dst2 は、それぞれ対応する Prefix/dst1 と/dst2 を持つクライアントである。 各リンクの初期遅延設定は 10ms で、各ノードのキューは 2000 パケット長である。 ここでコンテンツルーターはキャッシュを考慮しないため、ルートキャッシュサイズは 0 に設定した。

実験システムをノード src1 にインストールし、src2 にクロストラフィックジェネレータをインストールした。 実験システムとクロストラフィックジェネレータはシミュレーションの最初から実行され、実験システムの推定ネットワーク帯域幅を記録する。 クロストラフィックジェネレータは、ネットワークの実際の動作をシミュレートするためにニーズに応じて使用されるさまざまなパターンのクロストラフィックを作っている。

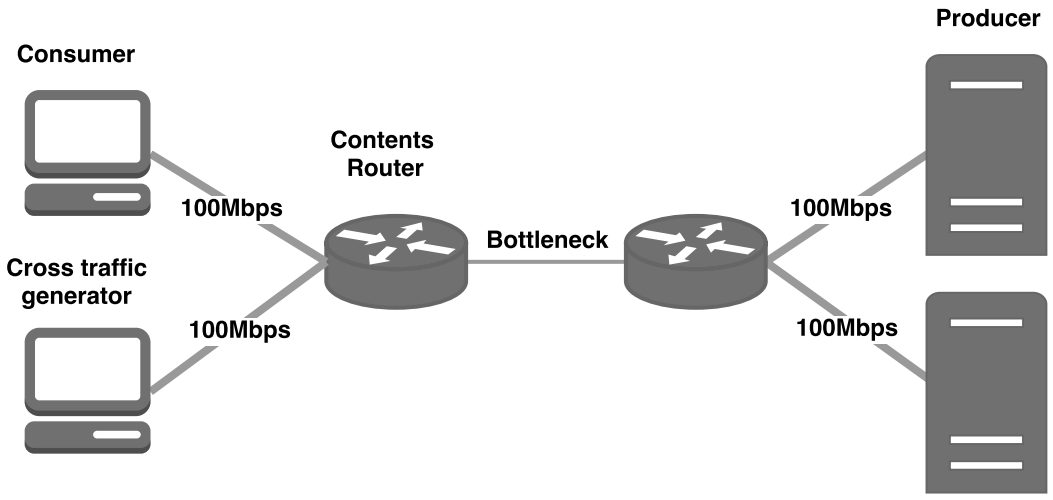

図 5.3: the simulation network topology

まずクロストラフィックは方形波のパターンで、クロストラフィックジェネレータは、合計実行時間シナリオ時間の 3 分の 1 から実行され、時間の 3 分の 1 後に実行を停止することで方形波のクロストラフィックを生成することができる。 平均 0.5Mbps のクロストラフィックを生成するために、トラフィックのジェネレータの設定を 1 秒あたり 7.125 の速度で Interest を送信する。 均等に Interest を送るだけでなく、ランダムに送信間隔を変更することもできる。 例えば、連続一様分布 $U\left(0, 2\times\frac{1}{f}\right)$ や平均 $\frac{1}{f}$、最大 $50\times\frac{1}{f}$ の指数分布に従って送信間隔を変化させることで、ランダムなクロストラッフィックを生成することができる。方形波クロストラッフィックを用いた実行の結果は図 5.15, 図 5.5 であり、選んだパケット間間隔歪み推定器の可用帯域推定ウインドウサイズ W_R は図 5.15 が 8 秒で、図 5.5 が 4 秒である。図 5.15 を見ると可用帯域の推定は過大評価する傾向があることが分かる。 図 5.15 と図 5.5 を比較すると、ウィンドウサイズ W_R が小さければ小さいほど、帯

域幅が変化したときにより速く推定されるが、誤差が大きくなり、オーバーシュートも大きくなるというトレードオフがある。

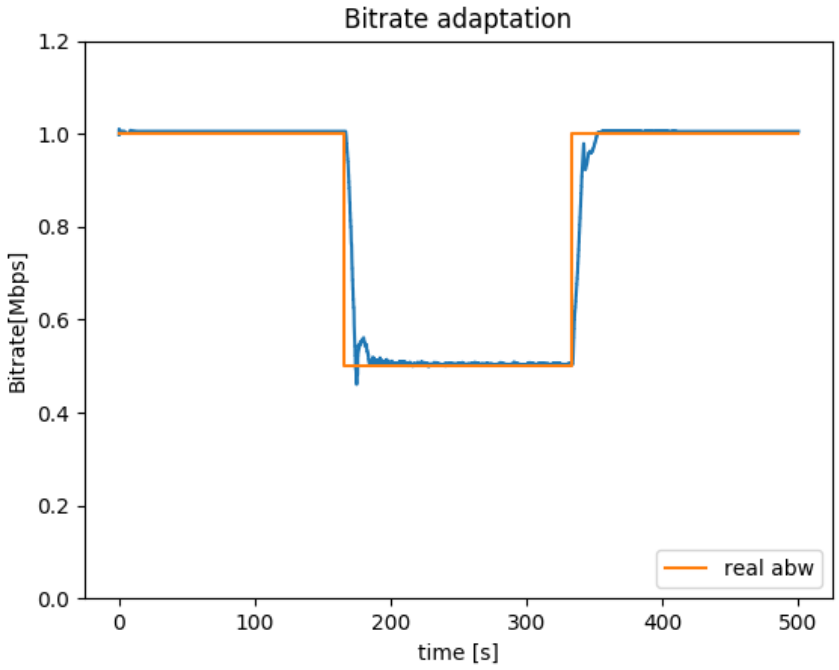

図 5.4: the result of bandwidth estimation with square wave cross traffic and $W_R = 8s$

次は、ランダムなクロストラフィックの場合である。図 5.6、図 5.7 は、指数分布に従うランダムクロストラフィックの場合の可用帯域の推定結果であり、図 5.7 は、図 5.6 の一部の拡大画像である。 3 つの波形は、実際の可用帯域幅、推定可用帯域幅、および実際の平均データスループットです。 データのスループットは実際の可用帯域幅の変化に従うことが分かり、2 つの間の差は非常に小さく、ボトルネックリンクの利用率が非常に高いことを示すことができる。 推定された可用帯域幅の変化は、比較的ゆるやかであり、データスループットは時には推定された可用帯域幅を超えることがあり、推定される可用帯域幅よりも小さい場合もある。続いて、二つの consumer があるシナリオでシミュレーションを行った。複数の Consumer があるシチュエーションではずっとネットワークを輻輳させるのではなく、1 より低い gMin を設定することで、過剰な輻輳を回避している。シミュレーションでは gMin,gMax を 0.9,1.1 に設定している。図 5.8 はボトルネックリンク帯域幅が 3Mbps で、二つの consumer が同時にストリーミングを開始するというシナリオでシミュレーションを行った結果である。図 5.9 は比較として従来の MPEG-dash(TCP/IP) を用いて同じシナリオでシミュレーションを行った結果である。図 5.8, 図 5.9 のように TCP/IP では推定した帯域幅が大きく変動するのに対し、提案システムは二つの consumer 間で差が出たものの、帯域の推定値が安定していると言える。

さらに二つの consumer のうち一つが途中からストリーミングを開始する場合図 5.10 や、途中からクロストラフィックが増加した場合図 5.12、複数の Consumer が 20Mbps のボト

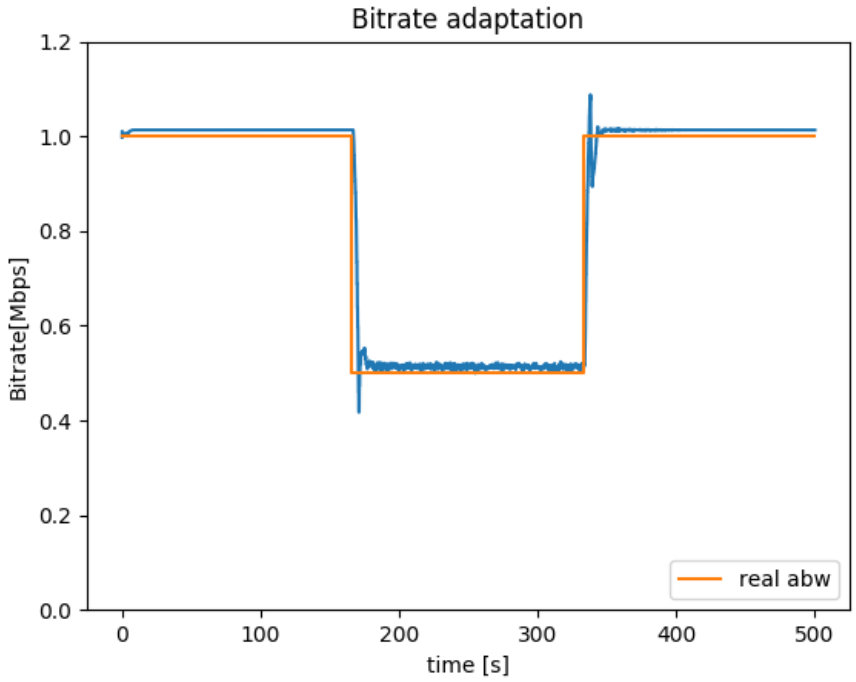

図 5.5: the result of bandwidth estimation with square wave cross traffic and $W_R = 4s$

ルネックをシェアする場合などにおいても、提案システムが安定して動作することが確認できた。

以上のように本研究で提案された可用帯域推定手法は、クロストラフィックの急激な変化（方形波クロストラフィック）やクロストラフィックのランダムな変化に関わらず、可用帯域を高精度に推定することができる。 同時に利用可能な帯域幅に応じて Interest 送信速度を制御し、高いボトルネックリンクの利用率が得られることを確認できた。

5.2.2　推定可用帯域を用いたビットレート適応実験

このセクションでは、使用可能な帯域幅の推定結果を使用して、ビットレート適応アルゴリズムの動作状況およびビデオ再生をシミュレートし、ビデオのビットレートとプレーヤーバッファのサイズの変化を追跡する。選択可能なビットレート B は、

$$\{45, 89, 131, 178, 221, 263, 334, 396,$$
$$522, 595, 791, 1033, 1245, 1547,$$
$$2134, 2484, 3079, 3527, 3840, 4220\}(Kbps)$$

と 20 段階のビットレートが含まれている。まずはボトルネックリンクが 1Mbps、consumer が一つで、クロストラフィックがないシナリオでシミュレーションを行った。その結果が図 5.14 に示されている。従来の MPEG-dash を用いて同じシナリオでシミュレーションを

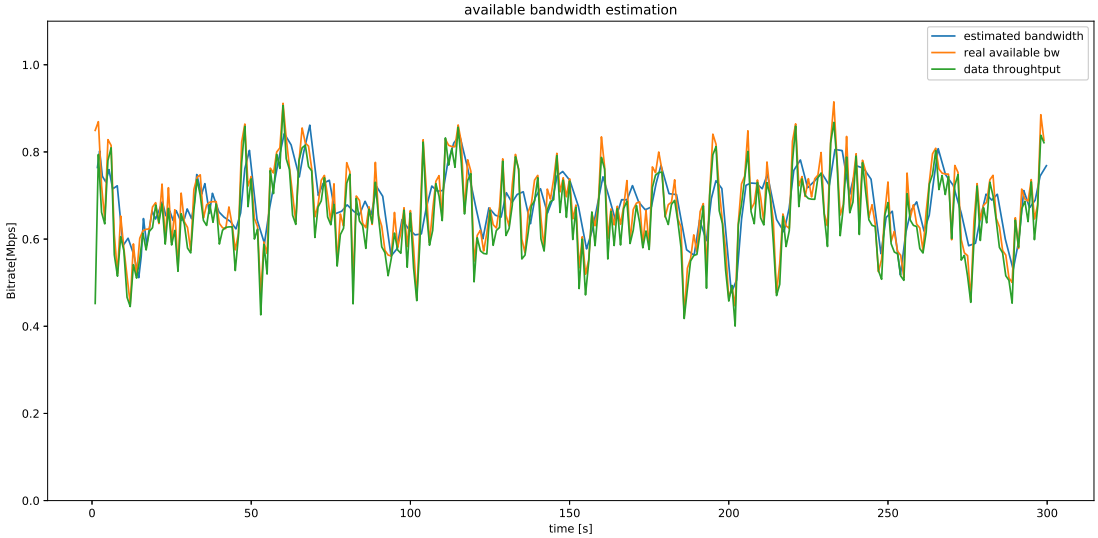

図 5.6: available bandwidth estimation with random cross traffic

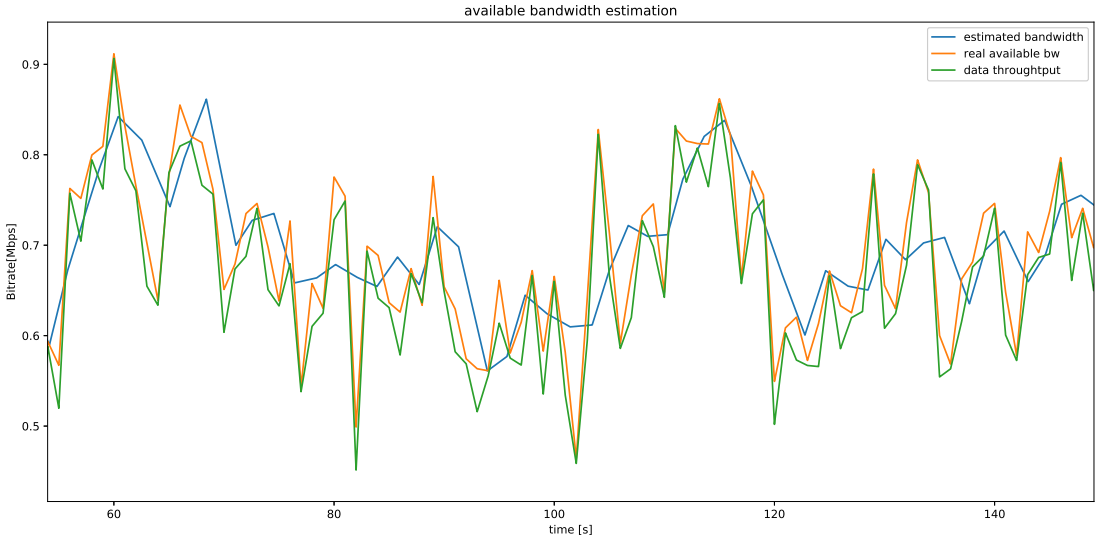

図 5.7: available bandwidth estimation with random cross traffic(2)

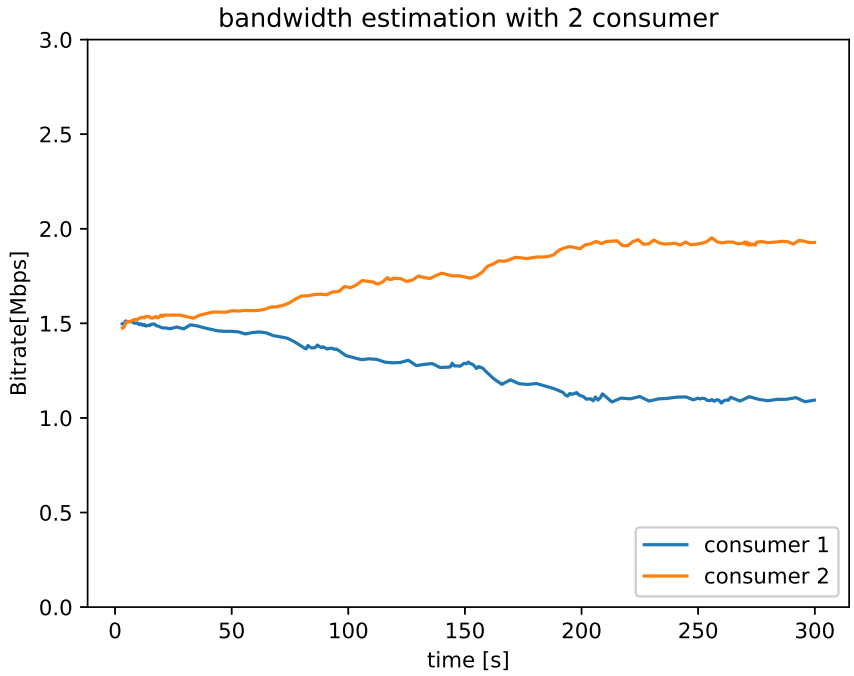

図 5.8: available bandwidth estimation with 2 consumers

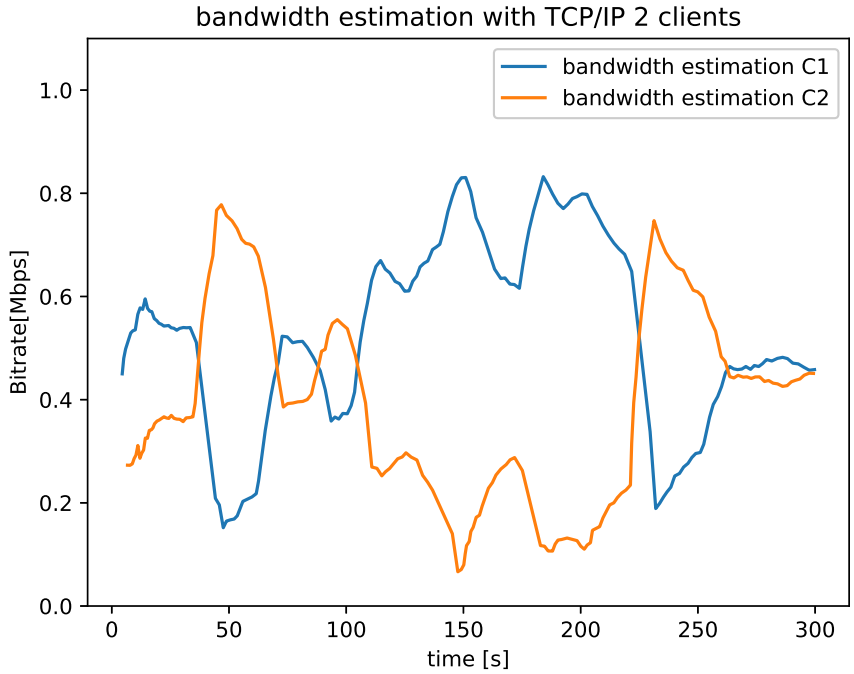

図 5.9: available bandwidth estimation with 2 consumers with MPEG-dash (TCP/IP)

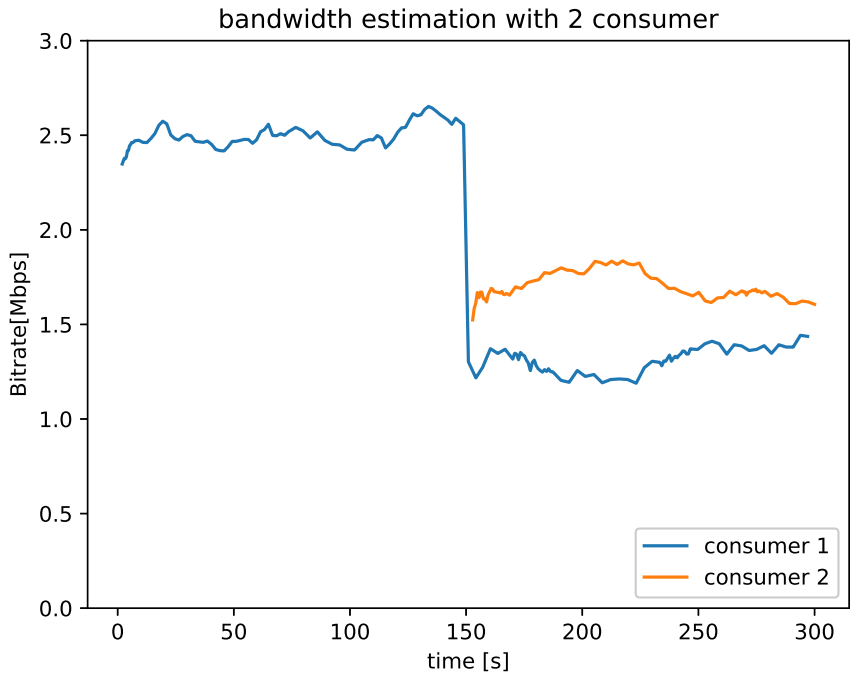

図 5.10: available bandwidth estimation with 2 consumers one starts at 150s

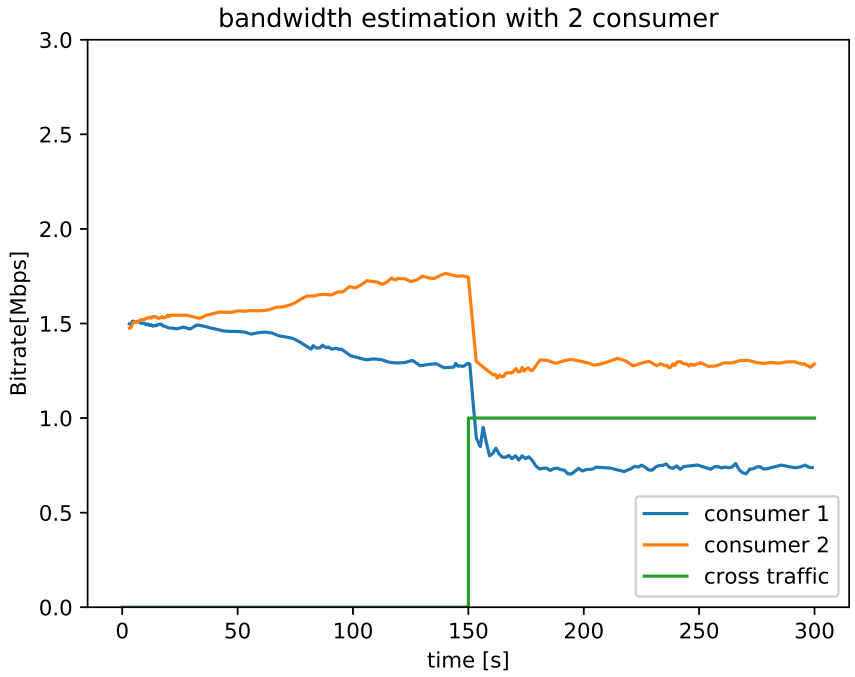

図 5.11: available bandwidth estimation with 2 consumers cross trffic starts at 150s

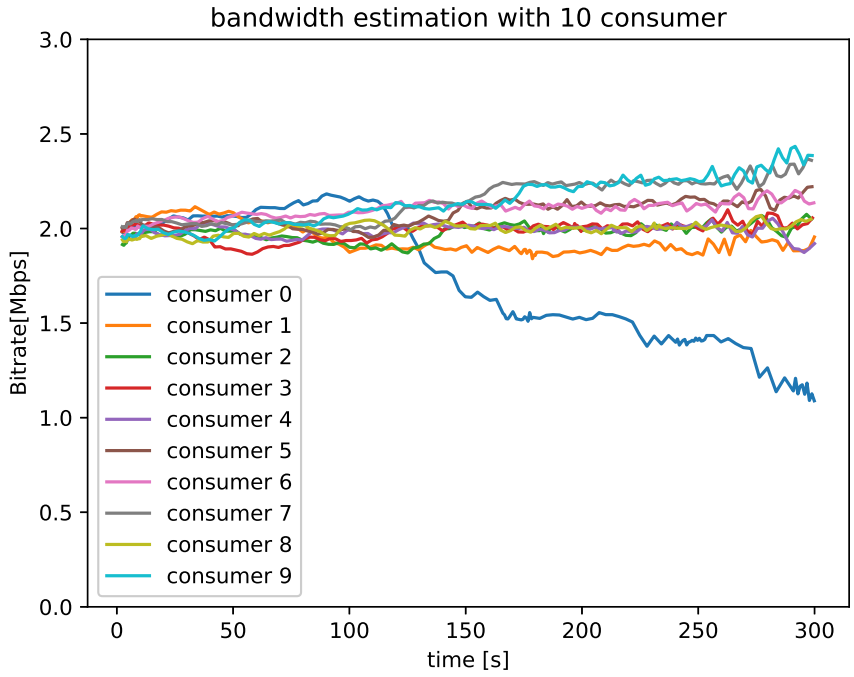

図 5.12: available bandwidth estimation with multiple consumers and bottleneck bandwidth of 20Mbps

行った結果が図 5.13 に示されている。二つの結果を比べると、従来の MPEG-dash では可用帯域を過小評価したせいで、ビットレート適応アルゴリズムはほぼずっと可用帯域より低いビットレートを選択した。しかし、提案システムでは可用帯域を正確に推定し、全体を通じて MPEG-dash より高いビットレートを選択したことが確認できる。このシミュレーションで正確に帯域推定を行う重要性が示された。

次にクロストラッフィックがあるシナリオでシミュレーションを行った。クロストラフィックは図 5.15 のように方形波と指数分布のランダムトラフィックを組み合わせたものを使用した。 図 5.15, 図 5.16 のシミュレーション結果を見ると、最初のターゲットバッファレベルに到達するまでの約 50 秒間に低いビットレートが選択され、ターゲットバッファレベルに達した後、利用可能な帯域幅に応じてもっと高いビットレートが選択される。グラフで分かったように、利用可能な帯域幅よりも高いビットレートを選択すると、ダウンロード速度より再生速度が速くなり、セグメントのダウンロードは再生よりも時間がかかるので、バッファーレベルは減少する。バッファレベルがあるレベルまで低下すると、より低いビットレートに切り替え、バッファレベルを増加させる。このように、バッファレベルの安定性を制御しながら、利用可能な帯域幅によって許容される最高のビットレートのコンテンツを取得している。

図 5.17 はボトルネックリンク帯域幅が 3Mbps で、選択できるビットレートを半分に減らし (奇数番目だけを使用した)11 段階のビットレートが選択可能な場合、シミュレーションを行った結果である。図 5.18 はさらに選択できるビットレートを 5 段階まで減らしたシ

ナリオでシミュレーションを行った結果である。選択できるビットレートの差が大きくなると、ビットレート適応アルゴリズムはバッファレベルを安定に維持するために、選択するビットレートの変動も頻繁になり、図 5.19 でバッファレベルの変動幅も大きくなることがわかった。

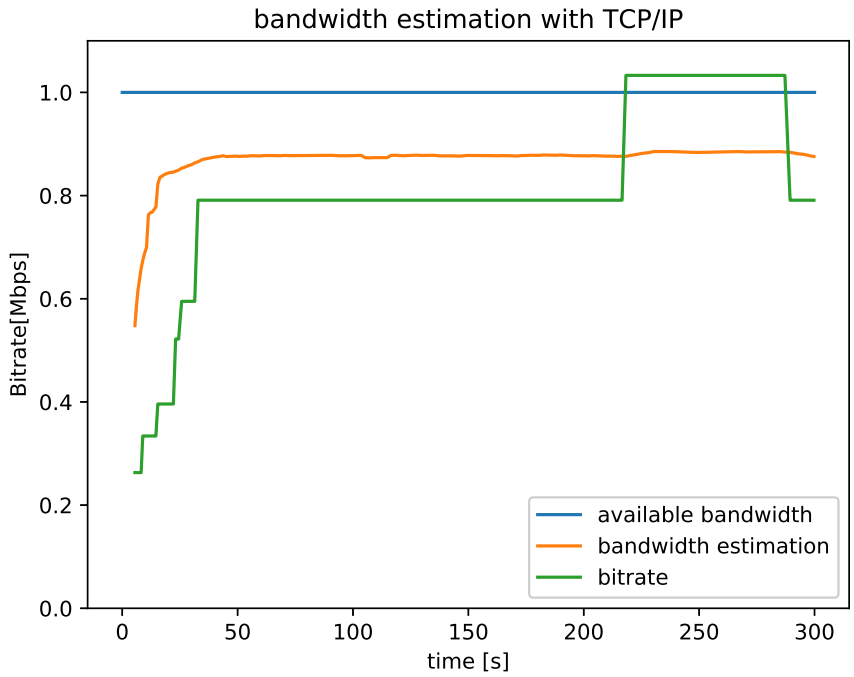

図 5.13: bitrate adaptation with traditional MPEG-dash over a 1Mbps bottleneck

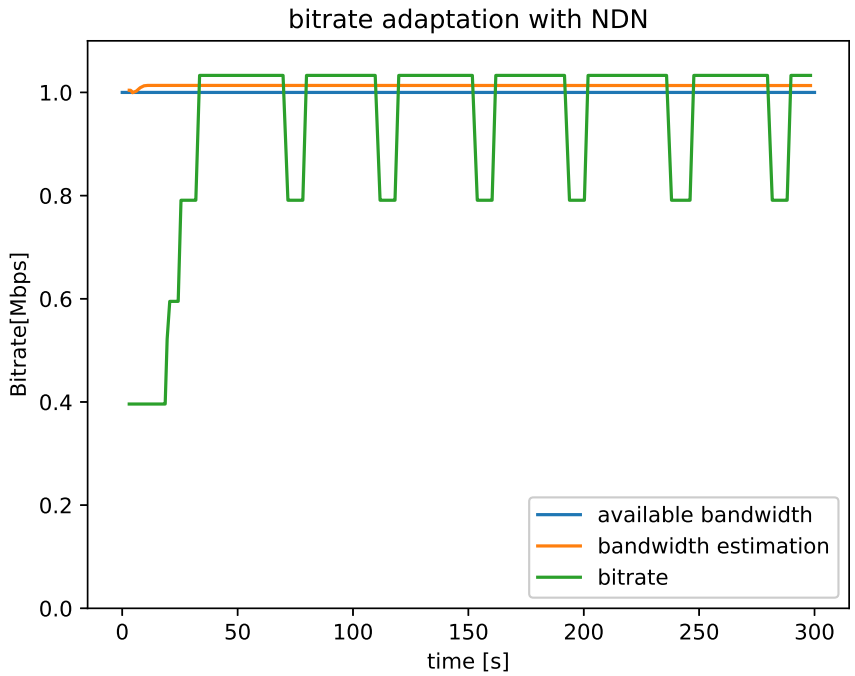

図 5.14: bitrate adaptation with proposed system over a 1Mbps bottleneck

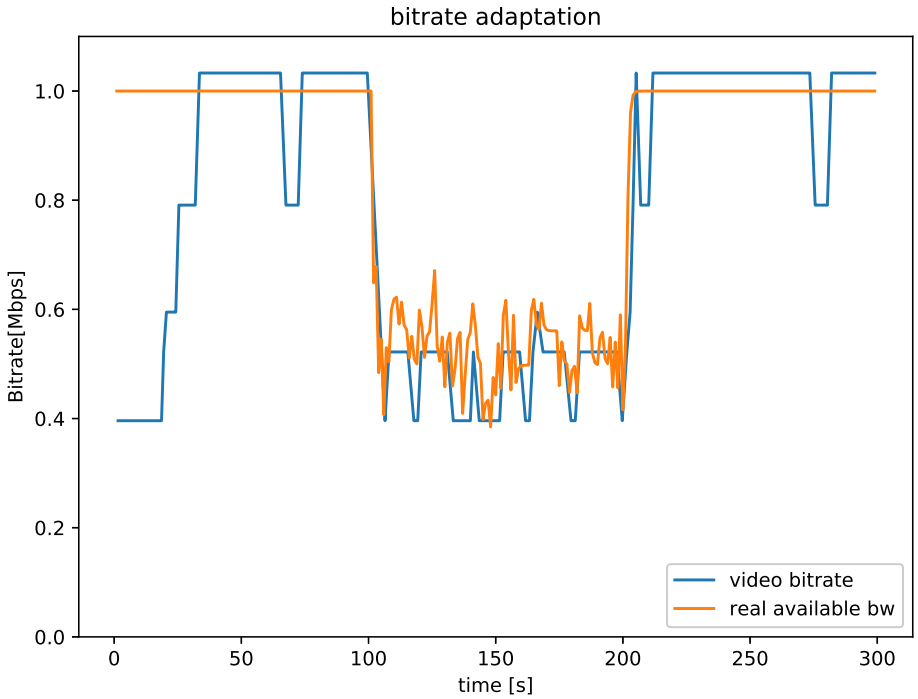

図 5.15: bitrate adaptation with square wave cross traffic

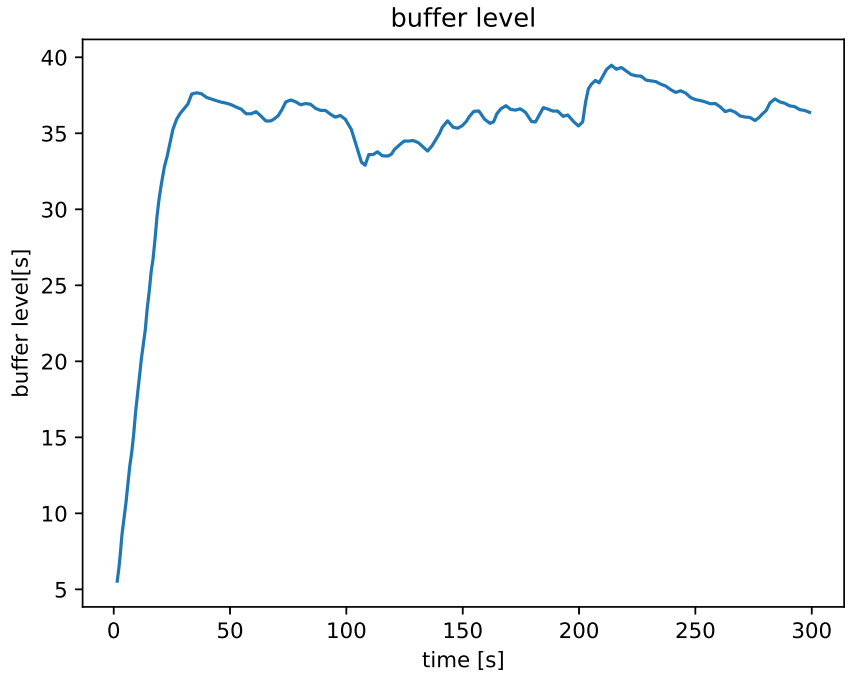

図 5.16: the buffer level in bitrate adaptation with square wave cross traffic

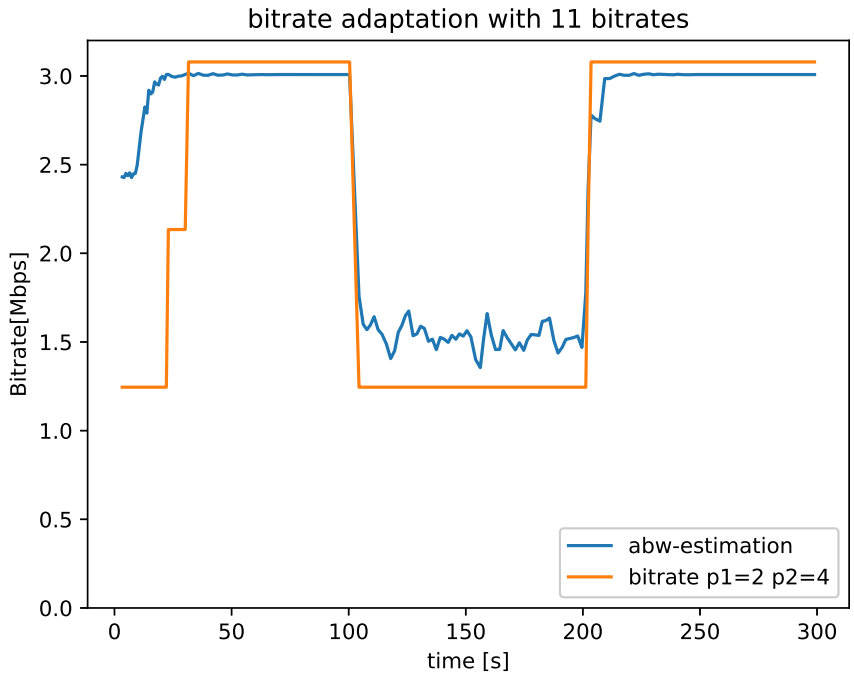

図 5.17: bitrate adaptation with square wave cross traffic and 11 available bitrates

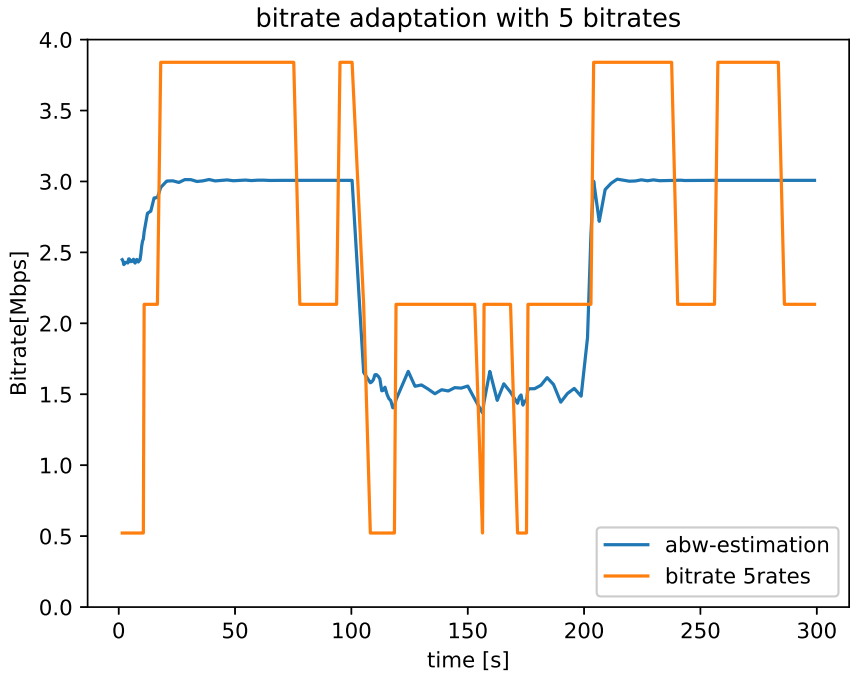

図 5.18: bitrate adaptation with square wave cross traffic and 5 available bitrates

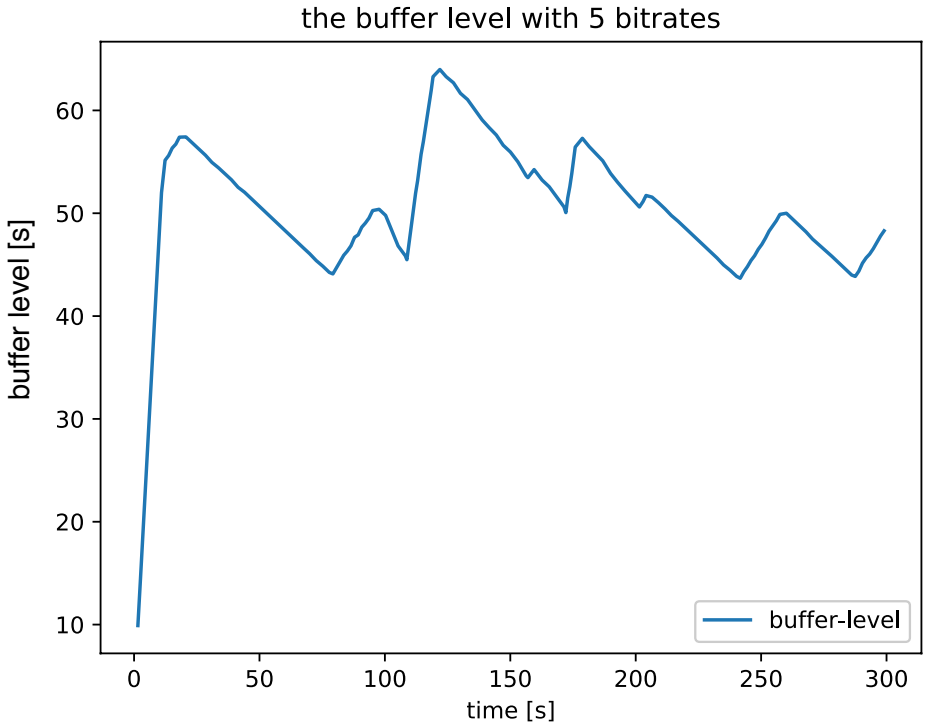

図 5.19: the buffer level of bitrate adaptation with square wave cross traffic and 5 available bitrates

第6章

結論

本研究では NDN ネットワークアーキテクチャにおいて利用可能な帯域幅を推定する方法を提案し、この帯域幅推定手法に基づき、トランスポート層の Interest 送信速度を制御する方法を提案した。

提案手法を使用すれば、ネットワークの利用可能な帯域幅をフルに活用することができ、ネットワークの状況が急激に変化する場合にも対応できることをシミュレーションで確認された。また提案手法はプローブパケットではなく、実際のアプリケーションのパケットを使用してネットワークのボトルネックに輻輳を起こさせ、これにより生じたパケット時間間隔歪みを用いて帯域幅推定を行っている。これは、この帯域幅推定と Interest 送信速度制御方法はビデオストリーミングアプリケーションのような大量のデータを転送する必要があるアプリケーションに適していることを意味している。

この帯域幅推定方法を、ストリーミングメディア再生のためのビットレート適応アルゴリズムと組み合わせることで、 従来のビットレート適応アルゴリズムのように、過去の一定期間の平均ダウンロードスループットを計算して、ネットワークの利用可能な帯域幅を推定するのではなく、各 Data パケットが到着した後にリアルタイムで推定値を更新し、ネットワーク帯域幅の変化により迅速に反応することができる。また従来のビットレート適応アルゴリズムにおいてセグメント単位に行う帯域推定よりも、ネットワークの可用帯域幅を高い精度で推定できると考えられる。

シミュレーションの結果より、提案手法を使用すれば、ボトルネックリンクの利用可能な帯域幅を最大限に活用することができ、同時に Fdash ビットレート適応アルゴリズムは、バッファレベルを目標バッファレベルの近傍で安定に維持させながら、可能な限り最高ビットレートを選択できていることがシミュレーションで確認された。この研究は、NDN フレームワークにおけるトランスポート層 consumer 側可用帯域推定と速度制御の実装にインスピレーションを与え、また、NDN フレームワーク上におけるビデオストリーミングアプリケーションの実現可能性や可用性を示唆した。

参考文献

[1] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3, pp. 66–73, 2014.

[2] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang, and Lixia Zhang. A case for stateful forwarding plane. *Computer Communications*, Vol. 36, No. 7, pp. 779 – 791, 2013.

[3] Dimitrios Vergados, Angelos Michalas, Aggeliki Sgora, Dimitrios Vergados, and Periklis Chatzimisios. Fdash: A fuzzy-based mpeg/dash adaptation algorithm. pp. 1–10, 12 2015.

[4] Thomas Stockhammer. Dynamic adaptive streaming over http -: standards and design principles. In *MMSys*, 2011.

[5] Shilpa Shashikant Chaudhari and Rajashekhar C. Biradar. Survey of bandwidth estimation techniques in communication networks. *Wireless Personal Communications*, Vol. 83, No. 2, pp. 1425–1476, Jul 2015.

[6] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, Vol. 17, No. 6, pp. 27–35, Nov 2003.

[7] Mahboobeh Sedighizad, Babak Seyfe, and Keivan Navaie. Mr-bart: Multi-rate available bandwidth estimation in real-time. *Journal of Network and Computer Applications*, Vol. 35, No. 2, pp. 731 – 742, 2012. Simulation and Testbeds.

[8] 野村俊一. カルマンフィルター—R を使った時系列予測と状態空間モデル. 共立出版, 2016.

[9] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control. *ACM Queue*, Vol. 14, September-October, pp. 20 – 53, 2016.

[10] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim 2.0: A new version of the ndn simulator for ns-3. 01 2015.

発表文献

[11] 趙亮, 相田仁. NDN 上のビデオストリーミングにおける可用帯域推定を用いたビットレート適合化, **情報ネットワーク研究会（*IN*）**, 2018 年 3 月 1 日.