社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＡＶＭ－59（13）
2007／12／14

ストリーム処理型 Web サービスを用いたＮＷサービス連携方式に関する研究

近藤 悟† 大西 浩行†

†NTT ネットワークサービスシステム研究所 〒180-8585 東京都武蔵野市緑町 3-9-11
E-mail: †{kondoh.satoshi,ohnishi.hiroyuki}@lab.ntt.co.jp

あらまし 近年、映像投稿サイトを始めとするメディア系サービスの普及が著しく、NW の広帯域化と高信頼化が進むに連れ、リアルタイムメディアストリームなどによるＮＷサービスが普及することが予想される。このようなストリーム型のデータ処理においてはデータサイズが巨大もしくは事前に決定できない等の理由から、フィルタ構造により処理を行う方式が一般的であるが、従来の Web サービスの方式とは大きく異なり連携が難しいという問題があった。そこで本研究では、①既存のアプリケーションに影響させずに、②ヘッダレベルを超えた様々なデータを認識及び変換処理可能にするという観点から、任意のストリームデータに対して各処理ノードを経由させながら最終宛先に配信させる処理連携方式を提案し、NW 上を流れる様々なデータの認識または合成処理等を Mashup サービスの連携要素にすることを可能にした。

キーワード ストリーム処理, メディア認識, Web サービス, サービス連携, SaaS.

A study of method to integrate stream-based Web services into NW services

Satoshi KONDOH† Hiroyuki OHNISHI†

† NTT Network Service Systems Laboratories, NTT Corporation
Midoricho 3-9-11, Musashino-shi, Tokyo, 180-8585 Japan
E-mail: †{kondoh.satoshi,ohnishi.hiroyuki}@lab.ntt.co.jp

Abstract Recently, the spread of the media service including the video upload site is remarkable. The NW services of the real-time media stream is expected to spread in the future as the NW systems will be made broadband and high-reliability. In general, it is known that stream data processing is implemented by filter functions because there are some problems that stream data size is huge and Non-determinate. But, it was difficult to make stream-processing cooperate because filter functions are greatly different from the conventional method of Web services. Then, to reduce the influence on existing applications, and to recognize or transform data except header data, we propose the method to make it reach a final destination through various processing nodes by intercepting stream data. As a result, application developers come to be able to create the Mashup service that deal various data which is streaming in NW easily.

Keyword Stream processing, Media recognition, Web services, Service orchestration, SaaS.

1. はじめに

近年，Youtube やニコニコ動画に代表される映像投稿サイトの爆発的な普及に伴い、一般ユーザによって魅力的な動画コンテンツが投稿されるようになってきている。しかし多数のユーザを惹きつけるコンテンツを作成しているのは特殊な編集機材・ソフトウェアを保有しているユーザに限定されているのが実情である。

一方、Web2.0 の隆盛とともに様々な Web サービスが登場しており、端末で特別なソフトウェアやライブラリを有しなくても、NW 上のサーバの機能を借りることで多様な処理を実現できるようになりつつある。

近年,Parlay-X[1]などによって、テレコム系機能や、プレゼンス情報、更にはデバイス制御など NW が持つ機能を Web サービスで外部に公開し、アプリケーション開発者に多様なサービスを自由に創出してもらう動きが盛んになりつつある。

また NGN などの広帯域 NW が実現されると、従来のような蓄積された動画コンテンツだけでなく、リアルタイムストリーミングによる映像・音声を用いたアプリケーションが増加することも予想される。これに対し、既存の関数型 Web サービスのように、データサイズが予め決められており、1 回の入出力対(若しくはその片方のみ)が授受されるのを待つ仕様では、次々に到着するストリームデータには対応することはできないという問題があった。

そこで本論文では、様々なデータの認識や合成を含

む処理サーバ及びプログラムを簡単にストリームに対応可能にするための方法と、それによって実現されたWebサービスを組み合わせる方式について提案し、ストリームデータをコンポーネントとしたMashupサービスを作成し易くする基盤を構築した。

2. 関連研究及びその課題

2.1. 関連研究

CPU性能が貧弱な携帯端末のためのコンテンツ編集装置及びシステムの従来研究としては、XMLに則ったシナリオ記述に従いながら映像などのソースを撮影することで初心者でも簡単に編集が可能な「vlogテンプレートによる個人映像製作支援システム」[2]が存在する。但し一旦端末に蓄積した動画ファイルをサーバ側に送信して処理を行う方式のため、リアルタイムメディアストリームにエフェクトなどを施すことはできない。これに対し、RTPデータを、NW上に配置されたストリーム処理を行うサーバを経由させることにより、リアルタイムメディアにも対応した「ストリーム処理ノードシステム」[3][4]の研究が存在する。これにより、編集機材の共有化を図ることができるだけでなく、端末とサーバで処理毎にメディアデータを受け渡す方式のように、端末側で次に何の処理を行うかを決定するプログラムを用意する必要がなくなる。但し、このシステムを利用するためには、端末にRTPを拡張したFMTPを送受信可能なモジュールクラスを使用してプログラムを行う必要があるため、既存のアプリケーションを大きく改造する必要がある。また多段処理を行う場合には、端末から最初に行われるべき処理ノードに向けてデータを発信する必要がある上、最終的な宛先端末も最後の処理ノードの設定に依存するという問題があった。

他研究では、IVRやDTMFによるサービスを提供するメディアサーバを用いてストリーム処理を行う方式としては、メディアインスタンスと呼ばれる処理コンポーネントを、呼制御サーバを用いてSIP接続する検討[5]がある。しかし、通信をするためにSIP機能を端末で有する必要があり、RTSPや単純なRTPだけの通信を連携させることはできない。またやはり多段処理を行う際にデータの宛先を最初の処理サーバに設定する必要があるなど、端末側で処理機構を意識する必要があった。

また最近では、メディアストリームだけでなく、XMLストリームをNW上で処理する製品[6][7][8]も出現してきており、高速でXML解析するためのアルゴリズム[9][10][11]も盛んに研究されている。しかし、上記メディアストリームとXMLストリームは性質が異なり、それぞれ専用のシステムが検討されているため、互換性がないという問題があった。

2.2. 課題と要求条件

従来研究で挙げた課題を解決するために以下の要求条件を列挙する。

① 端末が処理機構の配置などを知る必要がなく、データドリブンで処理が行われる機構であること

② 多くの既存処理サーバに手を加えずに(若しくは改造ができるだけ簡単に)、処理ノードとして利用できる仕組みを有すること

③ 初心者でも簡単に、かつ動的に処理連携を実現できる仕組みを有すること

従来、NWにおいて様々なデータを認識するためには全パケットを監視する必要があり、負荷が問題であったが、本論文では、パケットなどL3レベルの解析を行う部分と、上位のアプリケーションデータを解析する部分を切り離して専念させることで、負荷を軽減しながら、上記の要求条件を実現し、NW側でデータを認識して多様なアプリケーション連携を可能とするためのストリーム処理型Webサービス連携システムを考案している。

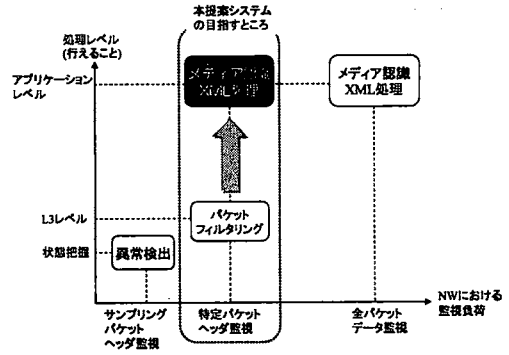

図1. 本研究の目的

3. 提案手法

本節では、まず対象とするデータを絞り込み、それを用いたシステムの概要及び実現できるサービス例を説明する。

3.1.1. 対象データ

本システムの適用先として考えているストリームデータとは、以下のような特性を持つものを想定している。

A) 情報源から能動的にデータが届けられる

B) 長時間に渡り、高速で連続に到達する

上記の特性を持つデータには、表1のようなものが挙げられる。表1のように、ストリームデータは高速性が重視され、UDPが用いられることが多いという特徴を持つ。そこで本論文では、前章の要求条件①②から汎用性を高めるために、RTPを含むUDPによるストリームを対象に検討を行った。

表 1：主なストリームデータの種類

	内容	用途	データ形式	プロトコル
センサ	光,音,位置	監視,異常検出	RAW	UDP
コンテンツ	映像/音声	配信, format 変換	MPEG, G711	HTTP, RTP
その他	株価,為替, 伝票	金融決済業務	XML	TCP, UDP

3.1.2. システムの構成

本論文で考案するストリーム型 Web サービス連携システムは、図 2 に示すように大きく分けて以下の 2 つから構成される。

[処理ノード]：入出力の設定などが Web サービスで公開され、その設定に基づいて、転送されるメディアや XML などのデータを認識または変換処理するノード

[統合指令サーバ]：アプリケーション開発者がブラウザなどからアクセスし、各処理ノードの接続関係を構築することで、入出力設定を各処理ノードに配信する指令サーバ

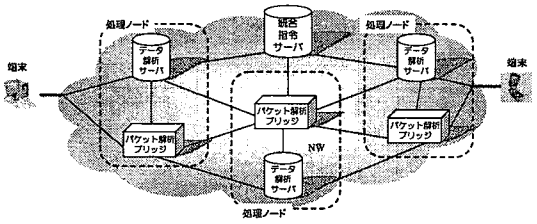

図 2.システムの構成図

2.2 の課題①②に対しては、NW 側においてデータを L3 レベルで認識し、特定の処理ノードへインターセプトすることによって解決の手段としている。また課題③に対しては、parlay-X による API のように、通信設定が Web サービス化された新たなブリッジ装置や、多段処理を GUI 上で簡単に設定する機能を考案することで解決への手段としている。次に本システムを用いて実現できるサービス例を簡単に示す。

3.2. 実現できるサービス例

従来、IP 再送信においてユーザ毎の嗜好に合わせた広告を挿入する場合、メディアサーバ側で主映像と広告を合成してユニキャストする必要があった。しかし、本システムを利用することにより、図 3 のように NW 側で映像自体を認識し、マルチキャストで分岐した後に広告を合成できる。このため、配信サーバからマルチキャスト配信するだけで、ユーザコンテクストを反映したサービスを提供することが可能となる。

その結果、配信サーバが個別にメディアを合成する必要がなくなるためサーバの負担が軽減され、またストリームも 1 つになるため NW の輻輳が回避できるという利点もある。

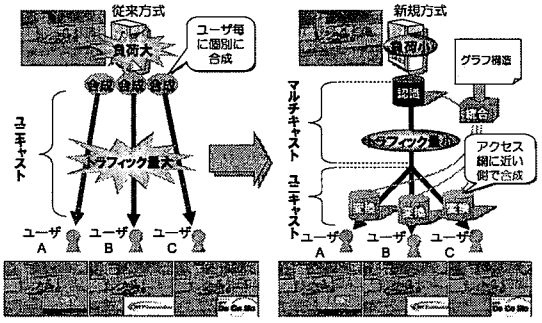

図 3. マルチキャストによる広告挿入配信

3.3. 処理ノードの構成

本提案システムにおける処理ノードは、以下の 2 つによって構成される。

① パケット解析ブリッジ

② データ解析サーバ

上記 2 つは NW 上において別々に存在するが、仮想的にセットで 1 つのノードとして動作する。

3.3.1. パケット解析ブリッジ

従来、サーバ同士を接続して多段処理を実現する場合はサーバを意識したデータの送出の仕方する必要があった。しかし、NW 側においてストリームデータを自由に取得できる機能(Web サービス)が存在すれば、アプリケーション側で完全にデータを操作可能となるため、送信端末は通常通り宛先端末に送ることだけで所望の処理が行われる。また最終宛先も、元の送信データのものを利用できるため、受信端末を変える毎にアプリケーション側から設定を行う必要はなくなる。

本機能は、上記を実現させるために、通過するパケットの内容を見て、転送やミラーリング及びカプセル化を行うものである。また処理対象とするパケットの条件や、後述するデータ解析サーバとの連携のためのモード設定が Web サービスで公開されており、表 2 に示す様な API 仕様になっている。

int setBridgeProperty(int mode, p_cond* in_c, p_cond* c_c, p_cond* out_c)

表 2：setBridgeProperty API 仕様

引数名	型	設定内容
戻り値	int	設定番号(負値はエラーコード)
in_c	p_cond*	パケット入力条件リスト
server	D_set*	中継先設定リスト
o_c	p_cond*	パケット出力設定リスト

なお p_cond は以下のような 5 タプル(送受信 IP アドレス・ポート番号・プロトコル)やカプセル化フラグを纏めた構造体である。

class p_cond {//条件設定用の構造体

```
    ip_address saddr; //送信元 IP アドレス
    short sport;      //送信元ポート番号
    ip_address daddr; //宛先 IP アドレス
    short dport;      //宛先ポート番号
    string protocol;  //プロトコル
    int mode;         //0:透過,1:転送,2:複製
    boolean isCup;    //カプセル化フラグ
}
```

上記の設定 Web サービス(＝WebAPI)を実行することで、ブリッジの内部テーブルに設定がブリッジを止めることなく動的に 1 つ追加される。第 6 フィールドとしてある通り、動作としては大きく分けて図 4 に示すような 3 つのモードが存在する。

透過モード 0：デフォルトのモードで、入力から出力へそのまま転送する。

転送モード 1：第 2 引数の入力条件に合致するパケットを中継先(データ解析サーバ)に転送し、処理されて戻ってきたパケットを第 4 引数の出力設定に応じて送信する。

複製モード 2：入力条件に合致するパケットを中継先に転送すると同時に、複製して元の宛先端末または他のパケット解析ブリッジに送信する。

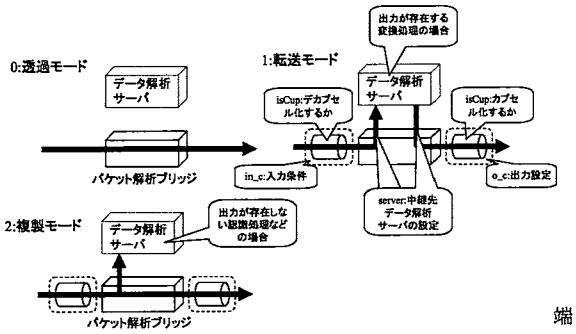

図 4. パケット解析ブリッジのモード

p_cond 構造体の最後のフィールドはカプセル化フラグであり、入力条件におけるカプセル化フラグが真の場合は、デカプセル化を行って、パケットを中継先に転送する。また出力設定におけるカプセル化フラグが真の場合は、パケットをカプセル化して送信する。

図 5 は、サービス例に挙げた処理グラフ例の実行シーケンスである。この例では以下の設定で説明する。

処理ノード①：音声認識(1 入力 1 出力)

処理ノード②：文字合成(2 入力 2 出力)

またストリームは以下のような想定である。

ストリーム①：端末 A→端末 C(音声 RTP)

ストリーム②：端末 B→端末 X(映像 RTP)

まず処理グラフの設定として、統合指令サーバに HTTP 等でアクセスし GUI 上で後述する処理グラフを作成することにより、setBridgeProperty を用いて以下のような設定が 2 つの処理ノードに成される。

処理ノード 1(ブリッジ 1)：

① 入力条件：[入力 1...RTP,転送,カプセル化なし]

② 中継先設定：[サーバ 1 の IP アドレス,入出力ポート]

③ 出力設定：[出力 1...宛先:ブリッジ 2 の IP アドレス及び入力条件ポート番号、カプセル化あり]

処理ノード 2(ブリッジ 2)：

① 入力条件：[入力 1...ブリッジ 1 の送信元 IP アドレス/ポート、宛先ポート番号、複製、カプセル化あり],[入力 2...RTP、転送、カプセル化なし]

② 中継先設定：[サーバ 2 の IP アドレス,入出力ポート]

③ 出力設定：[出力 1...指定なし],[出力 2...指定なし]

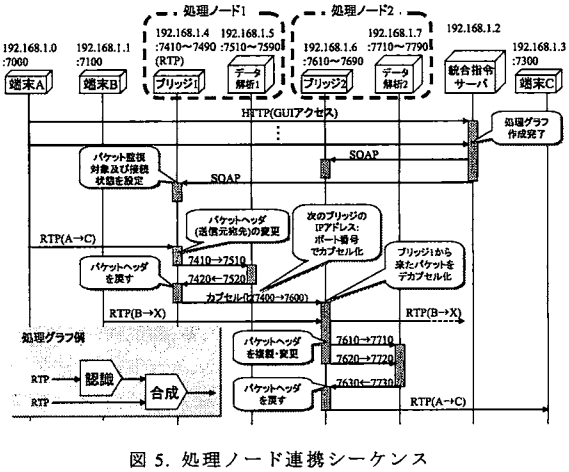

図 5. 処理ノード連携シーケンス

端末 A 及び端末 B から送信された RTP ストリームは、まず第 1 段階として、ブリッジ 1 に届くと入力条件に合致する RTP パケットか解析され、宛先 IP アドレスとポート番号がデータ解析サーバ 1 に変更されて転送される。次に、中継先のデータ解析サーバ 1 は処理結果のデータをブリッジ 1 の宛先ポートへ返信する。続いてブリッジ 1 は送信元がデータ解析サーバ 1 であるパケットを元パケットの送信元/宛先 IP アドレス:ポートに戻し、更に出力設定に応じて、何も指定がなければそのまま NW 上に送信(→端末 B)、カプセル化の指定があれば宛先 IP アドレス:ポートでカプセル化して送信する。

第 2 段階として、ブリッジ 2 に届いたカプセル化パケットは入力条件に従って「送信元がブリッジ 1」かつ「宛先ポートの合致」が解析され、デカプセル化された

後、入力設定に従って複製を行い、元パケットはそのまま通過して端末Xに送出される。複製パケットと入力条件に合致したもう1つのデータはデータ解析サーバ2へ処理ノード1と同様に処理される。

この例ではRTPを取り上げたが、本方式は基本的にUDP で送信されるデータであれば同様に扱うことが可能である。

3.3.2. データ解析サーバとの連携

データ解析サーバは、特殊なものではなく、一般的なソケットによる入出力を有するサーバまたはプログラムであるならば前述のブリッジと連携することが可能である。また、データ解析サーバの入出力の数により、以下のように連携するパケット解析ブリッジのモードが異なる。

変換型：エフェクトやコーデック変換など、型を変えずにデータを変換して転送する場合であり、転送モードのブリッジと連携する。

終端型：音声認識・映像認識など、結果を他に転送せずにDBなどに蓄積する場合は、複製モードのブリッジと連携する。

パケット解析ブリッジと連携するには、統合指令サーバ等が、前述の setBridgeProperty に以下の D_set 構造体を用いて登録する。

```
class D_set{//データ解析サーバ設定
        ip_address s_ip;   //サーバの IP アドレス
        short* in_p;       //入力ポート番号リスト
        short* out_p;      //出力ポート番号リスト
}
```

上記について、異なるサーバ IP アドレスの D_set インスタンスを混在させて登録させた場合、1つのパケット解析ブリッジと複数のデータ解析サーバが連携することになる。また同一 IP アドレスで IP アドレスだけが同じで他のポート番号が全て異なるインスタンスを複数登録した場合、ブリッジは到達するストリームに対して現在使用されていないポート番号を動的に割り当てる機能を有している。これは同時に複数のストリームを処理可能にするためである。もし全てのポート番号が使用されている場合は透過モードで通過させるが、これはストリームを処理待ちで遅延させないようにするためと、元々端末が処理システムを意識しないでデータを発信しているので基本的には影響がないためである。

3.4. 統合指令サーバ

本節では、統合指令サーバが、前述の処理ノードの WebAPI を用いて、各処理ノードが連結したグラフ構造を構築する様子について説明する。

3.4.1. 処理グラフ作成 GUI 機能

まずアプリケーション開発者は、ブラウザなどから統合指令サーバにアクセスし、GUI を用いてストリームデータがどの処理を経て最終的に端末に送信されるかを決定するグラフ構造を構築する流れについて説明する。なお本論文では、このグラフ構造をフィルタグラフと定義する。図6に示すように、GUI 上では各処理は1つのフィルタとして表され、IP アドレスやポート番号などの詳細な部分は隠蔽されている。

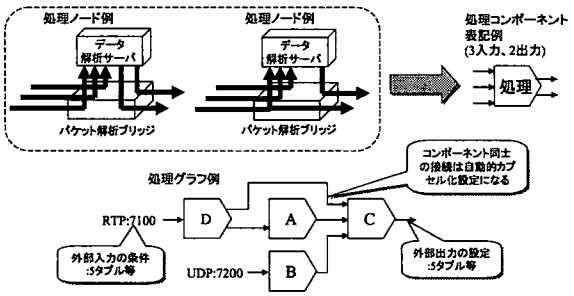

図 6. GUI による処理グラフの作成

アプリケーション開発者は、各処理を選択してその入力及び出力を接続することでフィルタグラフを構築する。構築後フィルタグラフを確定すると、統合指令サーバは予め登録された処理ノードのブリッジに対してそれぞれ setBridgeProperty を実行する。ブリッジの設定はルータの設定とは異なり動的に可能なため、ペット処理中でも処理グラフを追加可能である。

この際、処理グラフ上を流れるストリームが、別の処理グラフを流れることがないように、カプセル化で用いるポート番号は既に使用されていないものを利用する。これにより処理グラフ同士の独立性を保つことができる。

3.4.2. 処理グラフテンプレート推定機能

粒度が小さな処理を組み合わせると、処理グラフの構造が煩雑になりがちであり、一般ユーザにとって効果的な組合せがどのようなものであるかが判りにくいという問題がある。そこで統合指令サーバにおいて、過去のアプリケーション開発者のフィルタグラフを蓄積し、そこからアプリケーション開発者が指定した条件に合う最適なテンプレートを推定する方式[12]を検討している。これは個々の処理の入出力同士の接続される度合いを遷移確率として記憶し、それを用いた構造カーネル計算を行うことで最適なテンプレートを推定する方式となっている。

4. 既存方式との比較評価

本論文で提案したシステムの評価として、図7に示す(関連研究で挙げた)以下の 2 つの方式と比較したものを表3に示す。

① FMTP による処理ノード方式[2]

② メディアサーバに SIP 接続を行って実現する

(B2BUA) 方式[5]

②においては、TV 電話会議(カンファレンスルーム)を流用して処理ノードを実現する方式を想定している。対象データの汎用性については、①は RTP のオプションに情報を埋め込むため、既に RTP オプションを別目的で利用している場合は適用できす、汎用性が高いとは言えない。また処理ノードの追加の容易性については、SIP 接続方式は SIP-IF を実装する必要があるため敷居が高い。また同様に①は FMTP 送受信モジュールを生成し、コールバック関数を作成する必要があるが、提案方式では Web サービスで公開されたパケット解析部を後から連携させることにより、複雑なモジュールをプログラムレベルで組み込まなくても良いという利点がある。

また処理の組合せの柔軟性については、②ではメディアサーバの仕様上、片方向のストリームセッションの生成が難しい上、カンファレンスルームという粒度の大きい機能を流用するため扱い辛い面があるが、①や提案方式では、そもそも 1 つのフィルタのように扱えるため、入出力が複数で数が異なっているような処理も扱うことが可能である。端末との独立性については、①や②では、最初の処理ノードに対して送信する必要があるため自由な最終的な宛先に送信できないという制限があったが、提案方式では元パケットの情報が保存されるため、所望の処理を行った後に元の宛先に届くという利点がある。

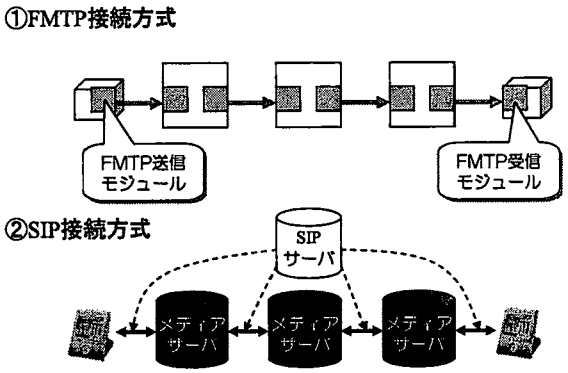

図 7. 比較方式

最後に高速性については、①や②は、データヘッダなどの変更が微量であるため高速である。提案方式ではパケット毎にカプセル化を行うためやや低速になるが、トンネル化等で既に実用的に利用されている手法であるため問題になるほどではない[13]。

上記の比較評価から、提案方式ではカプセル化という軽量な手法を適用することにより、対象データの汎用性や処理ノードの追加の容易性など、様々なユーザが簡単にストリーム処理を扱って連携サービスを構築可能にしていることが確認できる。

5. まとめ

本論文では、様々なストリームデータを処理するシステムを構築する上で、処理ノードにおいてパケット解析部分と、データ解析部分を切り離し、パケット解析部分を Web サービス化することで、従来よりも簡便に、ストリーム連携サービスを構築することが可能になった。今後の課題としては、RTP と RTCP のように他のプロトコルと連動するものに対するシーケンス番号一致の対応や、TCP への対応を図ることを検討している。

表 3：方式比較による評価

	FMTP 接続方式	SIP 接続方式	提案方式
対象データの汎用性	×(RTP のみ)	○（データに非依存）	○ (UDP 全般)
処理ノード追加の容易さ	△(特殊クラスを実装する必要あり)	×(SIP-IF を搭載する必要あり)	○
処理の組合せの柔軟さ	○	△(カンファレンスルームを流用)	○
端末との独立性	×(宛先は処理ノードにする)	×(宛先は処理ノードにする)	○(宛先は自由に設定可能)
転送の高速性		◎	○(解析とカプセル化がある)

文　献

[1] http://portal.etsi.org/docbox/TISPAN/Open/OSA/ParlayX30.html

[2] 狩塚・小西・森本・佐藤・谷口:"構造化撮影テンプレートを用いた個人映像製作支援システム",信学技報,IE2007-48(2006,9)

[3] 金子・竹内・和泉:"同期伝送プロトコルを使ったストリーム処理ノードの実装と多段接続における性能評価",信学技報,IN2007(2007,9)

[4] S.Takeuchi,Y.kaneko,M.Yamamoto,M.shibata,A.Narumi,and S.Sunasaki:"A Program Production System using ID and File-data over IP Networks", SMPTE Motion Imaging Journal, vol.114,No3,pp.132-138,Apr.2005

[5] 入江・金子・片山:" メディアサービス開発のためのコンポーネントベースのメディア処理モデル",信ソ大,B-6-38,2007,9.

[6] Cisco Service application oriented network

[7] Open Automatic Networking:Alaxala

[8] ネットワンシステムズ:" メッセージング・ネットワークが切り開く「ネットワーク」の新たな可能性", http://www.netone.co.jp/product/cat_network/msg_router/tfa9q10000005hqz-att/InternetWeek2006_print_v2.pdf, 2006

[9] 横山:"Web サーバ・ブラウザ間における XML ストリーム通信の実装",DBSJ Letters,Vol.6 No.1,pp89-92,2007.6

[10] 鈴木:"データスカッシング",情報処理 vol.46 No.1,2005

[11] 江田,鬼塚,山室:"XML データの要約情報を用いた高速な XPath 処理方法",信学会論文誌 vol.J89-D pp139-144,2006

[12] 近藤・森谷・大西:"フィルタ型 WebAPI を用いたグラフ構造の統合に関する一検討",信ソ大,B-19-21,2007.9.

[13] 入野,片山:" IPFIX における能力通知用オプションテンプレートの提案",信学総大,B-7-25,2007