インターネットで個人放送局を開くには (2)

正会員　甲藤二郎

早稲田大学理工学部　電子情報通信学科

“How to open personal broadcasting system on the Internet (2)”

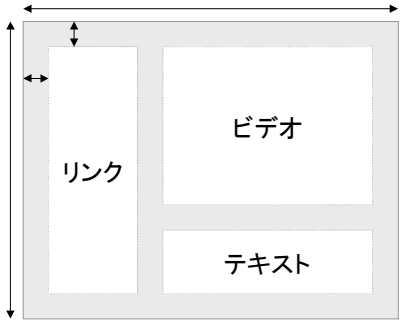
by Jiro Katto (Department of Electronics, Information and Communication Engineering, School of Science and Engineering, WASEDA University)

インターネット上で個人放送局を開設するための技術解説を行う。第 2 回はストリーミング技術の後半と、具体的なソフトウェアの実装方法について説明する。次回 (最終回) は、代表的なストリーミングソフトウェアの使用方法について説明する。

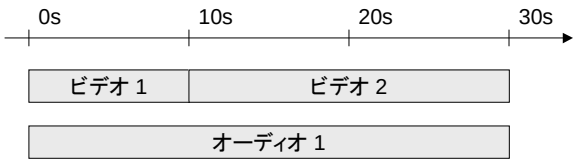
キーワード：インターネット放送、プレゼンテーション記述、ユニキャストとマルチキャスト、ストリーム・キャッシング、マルチメディア・プログラミング、ネットワーク・プログラミング

2.6 プレゼンテーション記述

インタラクティブ TV に代表されるように、テレビ番組に Web に似たインターフェースをかぶせることで、インターネット放送のコンテント面の拡張を実現できる。このために使用されるのがプレゼンテーション記述言語である。さまざまなフォーマットが提案されているが (D-HTML、MHEG、MPEG-4 等)、本稿ではその代表例として W3C (World Wide Web Consortium) で標準化された SMIL (Synchronized Multimedia Integration Language) について説明を行う [1]。



(a) レイアウト記述



(b) 同期記述

```
<smil>
<head>
    レイアウト記述
</head>
<body>
    同期記述
</body>
</html>
```

(c) SMIL 記述

図 1: SMIL によるプレゼンテーション記述

SMIL は XML に準拠しており、HTML に慣れた者なら誰もが容易に習得できる。図 1 に示すように、プレゼンテーション記述をレイアウト記述と同期記述に分類し、前者は

メディアの空間的な配置を、後者はメディアの時間的な関係を記述する。

具体的には、レイアウト記述は <layout> ... </layout> のタグで囲まれる。同期記述は <par> ... </par>、あるいは <seq> ... </seq> のタグで囲まれる。同期記述の <par> と <seq> は、それぞれ並列再生と逐次再生を表す。図 1(b) との対応で言えば、<par> を用いてビデオとオーディオの並列関係を記述し、<seq> を用いて二つのビデオの順番を記述する。なお、詳細は文献 [1] を参考にされたい。

プレゼンテーション記述に個々のストリーム情報 (SDP 等) を含めた構成も考えられるが、一般的に両者は分離される。具体的な配信は、

- プレゼンテーション記述 (SMIL 等)
- 個々のストリーム情報 (SDP 等)
- ストリーミング開始 (RTP 等)

の手順で行われる。リアルプレイヤや MPEG-4 のように、セッション中に動的にプレゼンテーション記述を更新できるものもある。これらはテキストやグラフィックスのストリーム転送もサポートしている。

なお、プレゼンテーション記述は各ストリームの時間関係を指定しているだけで、決してそれ自体がメディア同期を実現するものではない。前回説明を行ったように、あくまでもメディア同期は別のレイヤ (RTP 等) で扱われるものである。

2.7 ユニキャスト、マルチキャスト、スプリッタ

インターネット放送の放送形態として、図 2 に示す三方式が考えられている [2, 3]。

- ユニキャスト
- マルチキャスト
- スプリッタ

ユニキャストはサーバとホスト (受信端末) を一対一に接続する方式で、インターネットで通常用いられる接続形態にほかならない。よって、現在のインターネット放送は、ほとんどがユニキャスト方式で運用されている。

しかし、ユニキャストではホスト数分だけコネクションを張らなければならない。このため、ホスト数の増加と共にサーバの負荷が増大し、さらにはネットワーク全体のトラヒックが爆発する危険性もある。これを解決するために、マルチキャスト方式とスプリッタ方式が提案されている。

マルチキャストでは、クラス D の特殊な IP アドレス (224.0.0.0〜239.255.255.255) を使用し、IP アドレスとポート番号によって決まるグループ (マルチキャストグループ) に属するホストだけにパケットを配信する。このとき、サーバから送出されるパケットは一つだけであり、途中のルータが特定の経路 (マルチキャストツリー) に向けてパケットをコピーしながら転送する。このため、サーバがホスト毎にコネクションを張る必要はなく、トラヒックの増加も必要最小限に抑えられる。

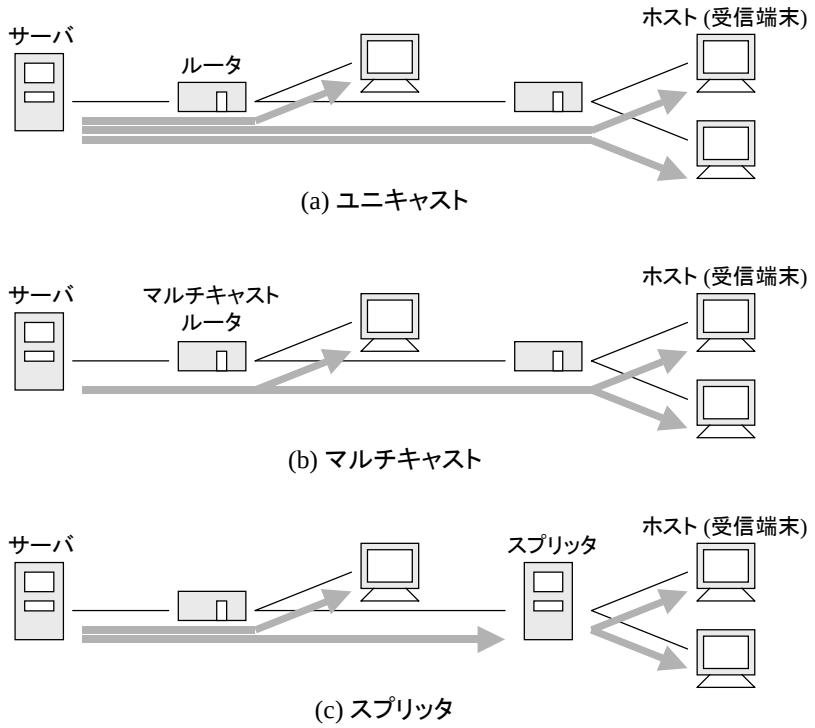


図 2: インターネット放送の配信形態の分類

マルチキャストのために使用されるのが、IGMP と各種のマルチキャスト経路制御プロトコルである。IGMP は、ホストのマルチキャストグループへの参加や離脱を通知する。また、経路制御プロトコルは、IGMP に応じて動的にマルチキャストツリーを決定する。具体的には、IGMP を用いてホストが参加要求を行うと、マルチキャストツリーが延長されてパケット転送が開始される。逆にホストが離脱すると、マルチキャストツリーが切り取られて転送が停止する。代表的な経路制御プロトコルとしては、DVMRP、MOSPF、PIM などが知られている。

ライブ放送や時間指定を行うオンデマンドサービスにとって、マルチキャストの有効性は明らかである。また、最近出荷されている大部分のルータや OS はマルチキャストに対応しており、インフラの準備も整っている。しかしながら、設定の手間や運用面の不安感のため、現在も mbone や Internet2 を介した評価実験が進められている [4]。

これに対してスプリッタ方式は、マルチキャストのような大きな変更を伴わず、前述の問題を緩和する方式である。サーバとホストの間にスプリッタを配し、サーバからスプリッタまでは一本のユニキャストで、スプリッタからホストまでは複数本 (ホスト数分) のユニキャストでパケットを転送する。このため、スプリッタの適切な配置を行うこ

とで、ネットワークのトラヒック (特にバックボーントラヒック) を大幅に削減することができる。さらには、オンデマンド用途でスプリッタをミラーサーバとして運用すれば、後述するキャッシュサーバとしてパケット転送遅延の削減にも有効に機能する。

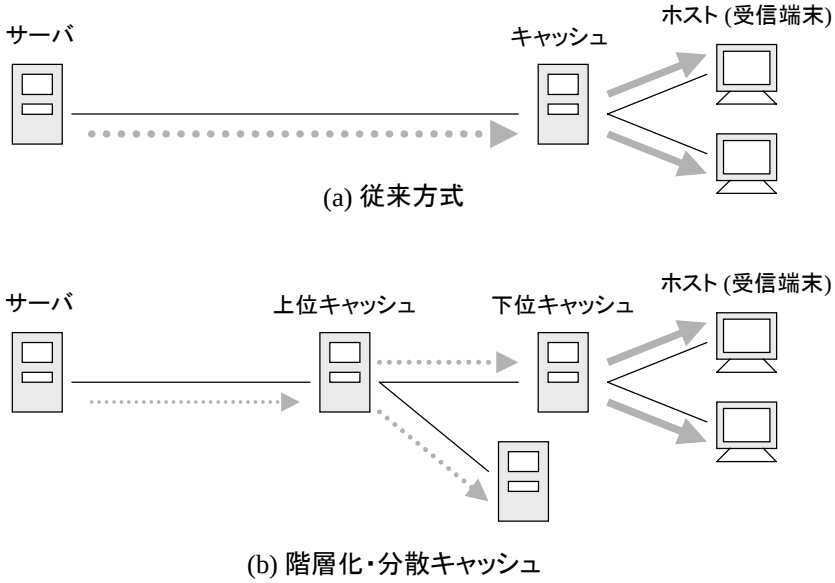


図 3: ストリーム・キャッシングの分類

2.8 ストリーム・キャッシング

オンデマンド用途で、スプリッタをミラーサーバとして運用し、さらにインテリジェントなキャッシュサーバとして運用することにより、ネットワーク全体のトラヒックとパケット転送時間を削減できる。これは通常の Web キャッシングと同様の効果である [5, 6]。

ただし、ストリームデータのサイズは非常に大きい場合があり、すべてをキャッシュするには限界がある。また、ストリームデータ特有の性質として、時間的に先行するデータは頻繁にアクセスされるが、後半のデータのアクセス頻度は下がるいう傾向がある。よって、ストリームデータに適したキャッシングを行うことで、パフォーマンスをさらに向上できることが期待される。

筆者らは

- キャッシュサーバの階層化と分散配置
- ストリームデータのセグメント分割
- セグメント単位のキャッシュ管理
- Web キャッシングとの親和性

を骨子とするストリームデータのキャッシュ方式に関する検討を進めている [7, 8]。図 3 には、ストリーム・キャッ

シュ方式の分類を示す。比較実験の結果として、階層化・分散キャッシュ方式によって、Web キャッシングの効率を下げることなく、トラヒック量、キャッシュヒット率 (転送遅延) 共に改善できることを明らかにしている。

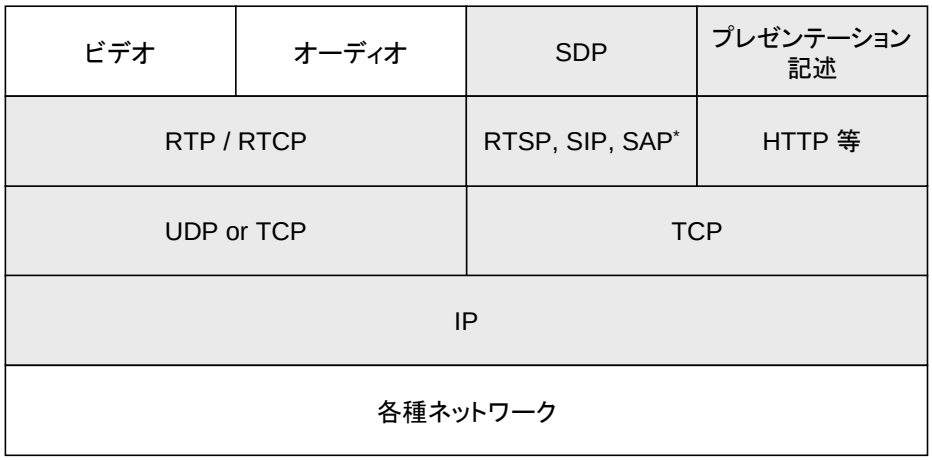


図 4: インターネット放送のプロトコル階層

2.9 総括

2.2 節から 2.6 節までの説明をまとめると、インターネット放送の典型的なプロトコル階層は図 4 のようになる。これらの実装によって、ストリーミングソフトウェアが完成する。さらに、2.7 節と 2.8 節で説明したスプリッタやキャッシュサーバの配置により、トラヒックや転送遅延を削減することができる。

表 1 は、インターネット放送で使われる代表的なソフトウェアの一覧を示している。世界的に広く使用されているのは、リアルネットワークス社、マイクロソフト社、アップル社の三製品である。数年前はより多くの企業が参入していたが、数多くの買収・合併を繰り返して現在に至っている。例として、Wavelet とスケーラブル符号化の VDONet 社、VxTreme 社、HTTP ストリーミングの Vivo 社、MPEG-1 ストリーミングの Xing 社、などが挙げられる。

図 4 の構成に最も忠実なのが、リアルネットワークス社の製品群である。これは各種制御プロトコルや SMIL の策定に直接関わっていたことが背景にある。同様に、アップル社の製品群も国際標準に忠実な実装を進めている。一方、マイクロソフト社の製品群は必ずしも国際標準に準拠していない。これは逆にオペレーティングシステムを含めたデファクト化を行える強みではあるが、ブラックボックスが増えることは気持ちが悪い一面もある。

表1: インターネット放送の商用ソフトウェア (ビューアは無償配布)

組織名	システム名称	URL
リアルネットワークス	RealSystem	http://www.real.com
マイクロソフト	Windows Media	http://www.microsoft.com/windows/windowsmedia/
アップル	QuickTime	http://www.apple.com/quicktime/
シスコシステムズ	IP/TV	http://www.cisco.com/warp/public/cc/pd/mxsv/
NTT	SoftwareVision	http://www.softwarevision.or.jp/
KDDI	QualityMotion	http://w3-mcgav.kddlabs.co.jp/qm/
東芝	MobileMotion	http://www2.toshiba.co.jp/mmotion/
キャノン	WebView	http://www.x-zone.canon.co.jp/WebView/

表2: プロトコル文書の入手先

組織名	プロトコル、標準名	URL
IETF	RTP/RTCP、ペイロードフォーマット	http://www.ietf.org/html.charters/avt-charter.html
	SDP、RTSP、SIP、SAP	http://www.ietf.org/html.charters/mmusic-charter.html
ITU-T	H.323、H.26X	http://standards.pictel.com/
ISO/IEC	MPEG	http://www.cselt.it/mpeg/
W3C	SMIL	http://www.w3.org/AudioVideo/

表3: 参照ソフトウェア (SMIL はバイナリのみ)

組織名	プロトコル	URL
UCB、UCL	RTP、SDP、SIP、SAP	http://www-mice.cs.ucl.ac.uk/multimedia/software/
コロンビア大他	RTSP	http://www.cs.columbia.edu/~hgs/rtsp/implementations.html/
オープンプロジェクト	H.323	http://www.openh323.org
リアルネットワークス他	SMIL	http://smw.internet.com/smil/smilhome.html

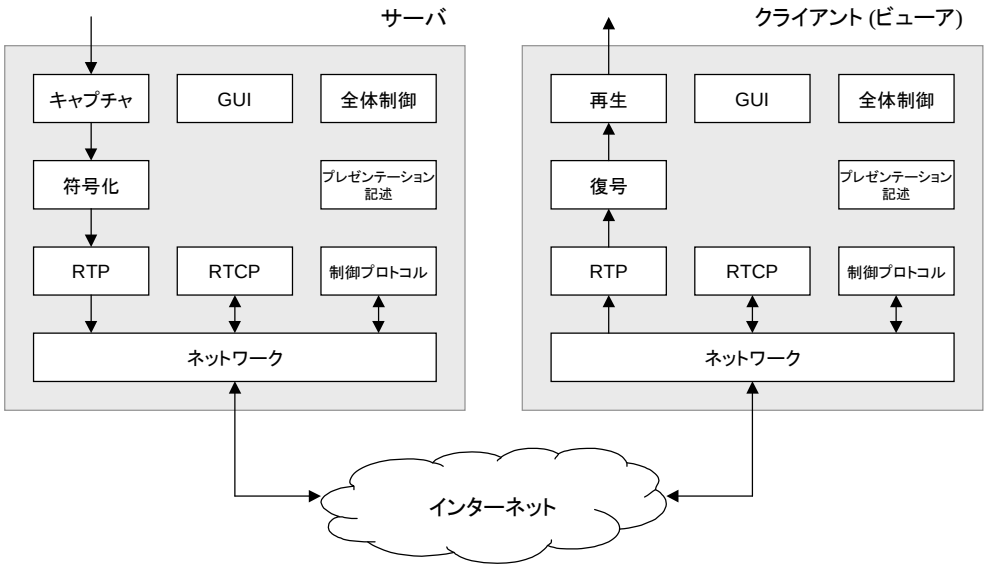


図5: ストリーミングソフトウェアの構成例

表2と表3は、各種標準化文書と対応する参照ソフトウェアの入手先を示す。SMIL を除き、参照ソフトウェアはすべて C/C++のソースコードを含む。ただし、SMIL に関しても、SGML/XML パーサのソースコードが各種公開されている (http://www.oasis-open.org/cover/ 等)。筆者自身は SMIL パーサを作成したことはないが、これらを拡張することで対応可能であると思われる。

3. ストリーミングソフトウェアの作成

3.1 概要

リアルプレイヤやメディアプレイヤを短期間に作成することは難しいが、プロトタイプであれば考えるほど難しいプログラミングではない。特に、大学院生等にとっては実践的な演習課題にもなる。そこで本章では、簡単なストリーミングソフトウェアの作成方法について説明する。

図5に、ストリーミングソフトウェアの構成例を示す。図4 のプロトコルスタックに加えて、AV データのキャプチャと再生、GUI (Graphical User Interface)、周辺要素としてのマルチスレッド、タイマイベントなどの実装が加わる。なお、以下ではライブ放送を前提とする。これは、オンデマンドサービス (VOD) は、制御プロトコルを除いてライブ放送のサブセットとして実現できるからである。また、制御系ではプレゼンテーション記述と制御プロトコルは扱わず、RTP/RTCP のみ説明する。これは、RTP/RTCP がストリーミングに必須の要素を含むからである。使用言語は C/C++、オペレーティングシステムは Windows と Linux を対象とする。また、サンプルプログラムは紙面では割愛し、筆者の Web ページに掲載する [9]。

3.2 要素技術

マルチスレッド

ひとつのプログラムを実行し、その中で複数のモジュール (スレッド) を並列動作させるために使用する。サーバ側のキャプチャ、符号化、送信、ビューア側の受信、復号、再生などは、マルチスレッド化することが望ましい。

ただし、マルチスレッドでは各モジュール間のデータの受渡しに競合が発生する (データの読出しと書込みが同時に発生する)。これを回避するために、モジュール間にバッファを設け、mutex 等の排他制御を用いてバッファアクセスを行う必要がある。

タイマイベント

特定の処理を、時刻を指定して実行する場合に使用する。ビューアにおける同期再生のための復号処理、定期的な RTCP パケットの送信、などに有効である。Windows の場合はマルチメディアタイマ、Linux の場合は後述する GUI ツールキットが提供するタイマ関数が使いやすい。

GUI (Graphical User Interface)

アプリケーションの GUI インタフェースを提供するために使われるのが一連の GUI ツールキットである。Windows の場合は Win32 と MFC、Linux の場合は GTK、Qt、Tcl/Tk などが代表例である。

GUI プログラムは一般的にイベント駆動構成となる。すなわち、プログラムを起動して無限ループに入るメイン関数と、発生したイベントを処理するイベント処理部から構成される。このとき、メイン関数の無限ループはイベント待ちループにほかならない。

3.3 マルチメディアプログラミング

ビデオ、オーディオ信号のキャプチャと再生は OS に依存する部分が多く、必然的に OS 依存のプログラムを書き分けなければならない。これに対して符号化、復号は OS に依存する部分が少なく、好対照をなしている。

また、通常のファイル形式によるキャプチャ、再生とは異なり、ストリーミング用の API (Application Program Interface) はほとんど提供されていない。このため、より低レベルのプログラミングが必要になる。この点が、ダウンロード (ファイル形式) とストリーミングの大きな違いでもある。

キャプチャと再生

ビデオキャプチャ API では、Windows では Video for Windows や DirectShow、Linux では Video4Linux 等が使用される。オーディオキャプチャ API では、Windows では MCI (Media Control Interface) や DirectSound、Linux では OSS (Open Sound System) 等が使用される。ビデオ再生 API では、Windows、Linux 共に GUI ツールキットが利用でき、あるいは Windows の場合は DirectDraw も用いられる。オーディオ再生 API では、Windows では MCI や DirectSound、Linux では OSS 等が使用される。

マルチスレッドで述べたように、キャプチャと符号化部、復号と再生部のデータの受渡しには注意が必要である。特にオーディオの場合、キャプチャバッファからの読出し、再生バッファへの書込み時に、現在のキャプチャ書込み位置、再生読出し位置を超えてアクセスを行わないようにするなど、十分な制御を加える必要がある。

符号化と復号

符号化と復号は、オペレーティングシステムの違いを意識することなく、C/C++の標準的な関数を用いて実装することができる。無論、MMX 等のアセンブラ命令を組み込む場合は OS (及び CPU) 依存となる。筆者自身の経験は十分とは言えないが、世界初のソフトウェアデコーダである mpegplay 以来有効な高速化手法はテーブル参照である。以前は有効とされた整数演算と浮動小数点演算の時間差は、最近の StreamingSIMD や 3Dnow!の登場によって顕著ではなくなってきている。一方、Web 上に存在する参照ソースはファイル対応のものが多い。これをストリーミング対応に修正するためには、すべてのファイル I/O 記述を取り除き、ネットワークアプリケーションに適したインターフェースとなるように関数を再定義すると使いやすい。

3.4 ネットワークプログラミング

TCP/IP の処理モジュールはカーネルに組み込まれており、通常はソケットと呼ばれる抽象化されたインタフェースを介してパケットを送受信する [10]。もちろん、Linux のようにソースコードが公開されている場合は、必要に応じてカーネル部分をカスタマイズしても構わない。

ソケットプログラミング

ソケットとは TCP/IP プロトコルを実行するための抽象化されたインターフェースのことである。これを使うことで TCP/IP の詳細を記述する必要がなくなり、プログラムの作成が簡略化される。

基本的な使い方としては、はじめに TCP、UDP のいずれかを指定したソケット (ソケット識別子) を作成する。これと IP アドレスとポート番号を指定した sockaddr 構造体を結び付け、一連のソケット関数を実行する。簡潔に言えば、ソケットを介して送受信端末を結合し、TCP/IP によるデータ転送を実行する。インターネット放送の場合は、メディアチャネル毎、制御チャネル毎、すなわちポート毎に複数のソケットを作成する必要がある。必然的にマルチスレッド化が求められる。

マルチキャスト

マルチキャストに対応するためには、ネットワーク部の修正が必要である。ただし、追加量はわずかであり、サーバ側、クライアント共に ip_mreq 構造体と若干のオプションの追加によってマルチキャスト化は完了する。これはソケットの恩恵に他ならない。これによって、アプリケーション実行時に IGMP が実行されることになる。

むしろ厄介なのはマルチキャストサーバの設定である。ストリーミングソフトウェアをマルチキャスト対応にしたところで、所属するドメインのルータがマルチキャストルータでなければ、単にローカルにブロードキャストしているに過ぎない。このため、まずは管理者間で連絡を取り合い、外部のマルチキャスト接続先を決定する必要がある。これはイントラネットかもしれないし、あるいは mbone 等の実験ネットワークかもしれない。その上で設定ファイルを用意し、ルータのマルチキャスト機能を有効にする (すなわち、IGMP パケットを送受し、マルチキャスト経路制御プロトコルが動作するようにする)。これによって、初めてマルチキャスト放送が可能になる。

3.5 RTP/RTCP

最後に RTP/RTCP について説明する。ネットワークと順番が前後するが、符号化・復号部とネットワークの間に位置するモジュールである。基本的に RTP、RTCP それぞれの構造体を定義し、必要なパラメータを設定した上でパケット送受信を繰り返す。一連の同期再生、誤り制御、フロー制御に活用する。

RTP

サーバ側の例として、ビデオ 1 フレーム分の圧縮データから複数個の RTP パケットを構成する場合を考える。この場合、例えば以下の処理を実行する。

- 圧縮データの分割
- RTP パケットの作成

圧縮データの分割はパケット廃棄対策を兼ねる。つまり、RTP パケット毎に復号可能とするために、再同期位置に合わせてパケットを分割する。最近は圧縮側で再同期をサポートする傾向にあり (MPEG-4、H.263+ 等)、圧縮データ中のユニークワード探索によって分割位置を決定できる。あるいは、圧縮側で再同期データが揃うたびに RTP モジュールを呼び出す構成にしてもよい。そして、タイムスタンプ、シーケンスナンバ、フレーム区切り (M-bit) 等を設定し、分割データに添付して RTP パケットを構成する。

クライアント側では、パケット受信のたびに以下の処理を実行する。

- RTP ヘッダのチェック
- 圧縮データの復元
- 復号実行時刻の決定

まずシーケンスナンバのチェックを行い、番号が不連続な場合は廃棄とみなし、再同期情報に従って圧縮データを復元する。このとき、復号部との整合を図り、復元データを復号器がそのままデコードできるようにしておくと楽である。

よくインターネットではパケットの到着順序が逆転すると言われるが、経路制御は概してスタティックであり、順序逆転はほとんど発生しない。また、順序の逆転が起きた場

合には、パケットは再生時間に間に合わないことが多い。よって、インターネット放送のようなリアルタイム系アプリケーションでは、順序逆転を廃棄とみなしてもあまり深刻な問題にはならないことが多い。

またタイムスタンプや M-bit をチェックし、タイマイベント等を用いて該当フレームの復号実行時刻を決定する。特に、タイムスタンプ時刻から実際に復号を開始するまでの時間はプレイアウト遅延と呼ばれ、ストリーミングソフトウェアの即時性の尺度になる。つまり、一度設定すると、同期再生のために同じ値に保たなければならず、プレイアウト遅延を超えて到着したパケットは廃棄するしかない。よって、即時性と廃棄率のトレードオフを考慮し、適切なプレイアウト遅延を決定するメカニズムが重要である。

RTCP

RTCP パケットは、例えばタイマイベントを使用し、送受信端末間で定期的に交換する。これを用いた処理として特に重要なのが、同期再生用の時間軸の設定と輻輳制御の実行である。なお、RTP/RTCP に使用するポート番号は対であり、RTP に偶数 2N 番を用いた場合、RTCP には奇数 2N+1 番を使用する。

前回説明を行ったように、メディア内同期は RTP ヘッダのタイムスタンプで実現できるが、メディア間同期には RTCP パケットの NTP タイムスタンプが必要である。このため、セッション開始時にはメディア間同期が取れず、個々の RTCP パケットが到着するまでは時間軸の対応付けができない。これは現在の RTP/RTCP の制約上、やむをえないことである。

より重要なのが輻輳制御である。UDP ストリーミングを行う場合、輻輳制御を行わないと他の TCP トラヒックのスループットが低下する。これは、TCP が自身の輻輳制御メカニズムによってレートを削減するためである。さらには UDP ストリーム自身が網全体の輻輳を引き起こす危険性がある。そこで、TCP との公平性を保ちながら帯域を有効利用する UDP の輻輳制御に関する検討が進められている。これは TCP フレンドリと呼ばれ、代表例として

$$Th = 1.22 \cdot \frac{MTU}{RTT \cdot \sqrt{loss}}$$

に従って TCP と等価な転送レートを推定し、レート制御を行う方式が知られている [11]。このとき、*Th* は推定レート、*MTU* はリンクの最大パケットサイズ、*RTT* はラウンドトリップ遅延、*loss* はパケット廃棄率である。

商用ソフトウェアの輻輳制御としては、リアルネットワークス社の SureStream やマイクロソフト社のインテリジェント・ストリーミングが有名である。共に技術的な詳細は不明であるが、受信レートの測定に基づくフィードバック制御、あるいは上記の TCP フレンドリの原理を用いているものと推測される。

表 4: オペレーティングシステムと各種 API の例

目的	Windows	Linux
マルチスレッド	Win32, MFC	pthread
GUI	Win32, MFC	X11, GTK, QT, Tcl/Tk, ...
ビデオキャプチャ	Video for Windows, DirectShow	Video 4 Linux
オーディオキャプチャ	MCI, DirectSound	Open Sound System
ビデオ再生	Win32, MFC, DirectDraw	X11, GTK, QT, Tcl/Tk, ...
オーディオ再生	MCI, DirectSound	Open Sound System
ネットワーク	ソケット	

3.6 総括

本章ではストリーミングソフトウェアの作成方法について説明を行った。まとめとして、表 4 ではプロトタイプ実装に有用な API の一覧を示している。API は日々変更されるものであり、陳腐化する危険性や、筆者の知らない API があることも注意願いたい。

(第 3 回に続く)

参考文献

[1] W3C: “Synchronized Multimedia Integration Language (SMIL) 1.0 Specification,” http://www.w3.org/TR/REC-smil (Jun.1998).

[2] 大澤光編著： “インターネットストリーミング,” 共立出版 (2000)

[3] 苅田監訳: “マスタリング TCP/IP: IP マルチキャスト編,” オーム社 (1999).

[4] K.C.Almeroth et al.: “The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment,” IEEE Network (Jan.2000).

[5] 藤本訳: “Web サーバ完全技術解説,” 日経 BP 社 (1997).

[6] インクトミ・ジャパン監修: “キャッシュ技術入門,” 小学館 (2000).

[7] 海老沢他: “マルチメディア・ストリームのための分散型キャッシングに関する検討,” 2000 信学会ソサイエティ大会, B-7-63 (Sep.2000).

[8] 海老沢他: “Web キャッシングと親和性を持つストリーミング・キャッシング方式,” 2001 信学会総合大会, B-7-206 (Mar.2001).

[9] http://www.katto.comm.waseda.ac.jp/~katto/streaming/.

[10] W.R.Stevens: “UNIX Network Programming,” Prentice Hall (1990).

[11] The TCP-Friendly Website: http://www.psc.edu/networking/tcp_friendly.html.