マルチメディア通信と分散処理　108－18
（２００２．６．７）

相互通信環境の動的構成を可能とするミドルウェアの設計

橋本浩二† 原田雅博‡ 勝本道哲* 柴田義孝†

† 岩手県立大学 ソフトウェア情報学部
‡ 東京エレクトロン株式会社
* 通信総合研究所

コンピュータネットワークを利用して，複数の利用者が相互にマルチメディア通信を行う場合，通信システムは利用者の QoS(Quality of Service) 要求を考慮し，適切な通信資源を確保する必要がある．処理能力の異なるコンピュータや利用可能な帯域幅の異なるネットワークを想定した場合，RTP におけるトランスレータやミキサといったトランスコーディング機能が動的に利用可能であれば，より柔軟な相互通信環境を構成することが可能となる．本稿では，トランスコーディング機能を適切な中間ノードへ配置することによって，相互通信環境を動的に構成するためのミドルウェアの設計を行う．

Design of a Middleware for Flexible Intercommunication Enhanced by Linking Dynamically

Koji Hashimoto† Masahiro Harada‡ Michiaki Katsumoto* and Yoshitaka Shibata†

† Faculty of Software and Information Science, Iwate Prefectural University
‡ TOKYO ELECTRON LIMITED
* Communications Research Laboratory

Using distributed multimedia system that can integrate various realtime and non-realtime media data, when the system users communicate with each other by realtime audio video data, the system must guarantee end-to-end QoS (quality of service) according to requirements of the system users and available resources. If the system can dynamically use translator or mixer functions defined by RTP, more flexible peer-to-peer communication is realized. In this paper, we design a middleware system for flexible intercommunication enhanced by linking dynamically.

1　はじめに

コンピュータの処理速度向上と音声や動画像圧縮技術の進歩により，安価なパーソナルコンピュータでも複数のリアルタイムメディアを処理することが可能となった．また，xDSL や FTTH の普及により一般家庭においても数 Mbps 〜 100Mbps 程度の帯域幅を利用することが可能となり，リアルタイムメディアを利用した相互通信が現実的なものとなった．現在，IP パケットで DV ストリームを転送する技術 [1] や，プロダクション品質映像 (D1) を配信する技術 [2] も研究開発されており，利用可能な帯域幅とコンピュータ資源によっては，非常に質の高い映像による通信も実現可能となってきた．

しかしながら，マルチメディアを利用して相互通信を行うためには，(1) 利用するメディアに応じて帯域幅を確保する，または利用可能な帯域幅に応じてメディアを選択する．(2) ソフトウェアやハードウェアにより構成される通信システムを各通信拠点に配置する．といった作業が必要であるとともに，一度構築した相互通信環境を変更するためには多くのコストがかかる．また，例えば遠隔にある大学のキャンパス間で約 35Mbps の DV ストリームを用いた遠隔講義の通信環境を構築する際，自宅や出張先からでも講義を受講するための仕組みを実現することは困難である．なぜならば，全ての講義受講者の通信環境をあらかじめ把握することは現実的ではなく，例え講義に参加できたとしても，メディア処理速度や帯域幅の不足が生じる可能性がある．もし，RTP[3] で定義されているトランスレータやミキサといったトランスコーディング機能を利用することが可能ならば，DV ストリームを MPEG や M-JPEG などのストリームへリアルタイムに変換することでこの問題を解決することが可能となる．しかしながら，動的な相互通信環境を考慮すると，適切な場所にあらかじめトランスコーディング機能を配置しておくことは難しい．

そこで，利用者の通信環境とサービス品質 (Quality of Service) 要求に応じて，トランスコーディング機能を適切な中間ノードへ配置することにより，相互通信環境を動的に構成するためのミドルウェアを提案する．本稿では，そのシステム概要と設計について述べる．

2 MidField システム概要

相互通信環境の動的構成を実現するための，**MidField**[1] システムアーキテクチャを図 1 に示す．

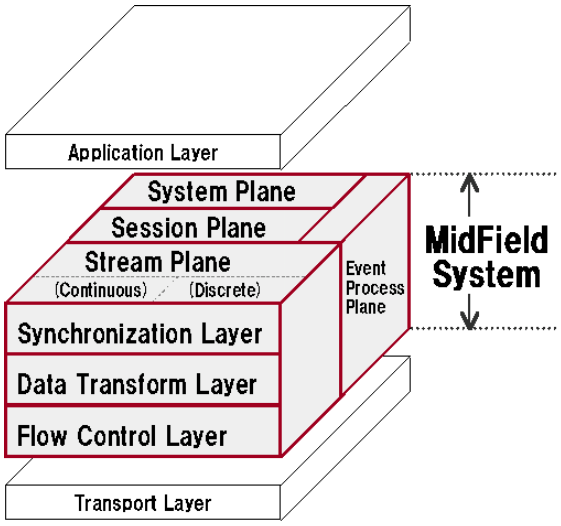

図 1: MidField システムアーキテクチャ

このシステムアーキテクチャはトランスポート層の上位に位置し，3 階層 4 プレーンで構成され，アプリケーションに対してマルチメディア通信機能を提供する．図中，同期層，データ変換層，フロー制御層により構成される Stream Plane はマルチメディアストリーム転送処理を行い，Session Plane では通信セッションの管理を行う．また，System Plane ではネットワークトラフィックや CPU 利用状況の監視を行い，システム利用者が要求する QoS に対するアドミッションテストを実行する．Event Process Plane では，システム内部で発生する各種のイベントを処理する．

2.1 MidField 通信セッション

利用者の通信環境とサービス品質 (Quality of Service) 要求に応じて，MidField システムは相互通信環境をコンピュータネットワーク上で動的に構成する．図 2 に MidField セッションの概要を示す．

システム利用者は相互通信を行うために，MidField セッションに参加する．図 2 では，システム利用者が 3 人 (MF1, MF2, MF3)，1 つの MidField セッションに参加している．ここで，MF1 と MF2 は共に，DV ストリームを送受信可能な通信環境を所有している．しかし MF3 は，帯域不足または DV 処理に対するコンピュータ資源不足のため，DV ストリームを用いた通信が不可能であり，MPEG4 ストリームによる送受信を要求している．この場合，MF1, MF2, MF3 が相互通信を行うコンピュータネットワーク上

[1] **Mid**dleware for **F**lexible **i**ntercommunication **e**nhanced by **l**inking **d**ynamically

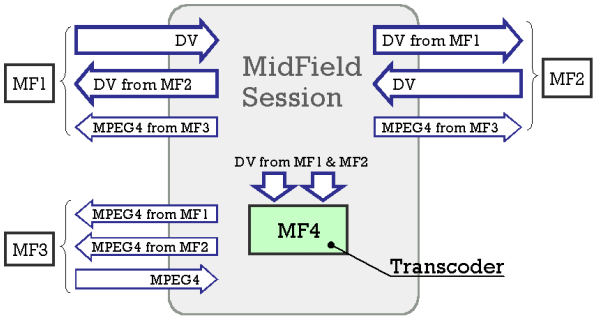

図 2: MidField Session

の適切な場所にトランスコーディング機能を配置する．図 2 では，MF4 が DV ストリームを MPEG4 へトランスコードすることにより，MF3 も相互通信が可能となる．

2.2 機能モジュール構成

MidField システム (図 1) の Session Plane におけるセッション管理機能は，MidField セッション参加者やメディアストリームの情報を取り扱う．これらの情報は，要求されるセッションの形態に応じて新しい機能が要求される可能性がある．これに対し，動的なトランスコーディング機能の配置を実現する際，管理情報やセッションの状態と管理機能をまとめて適切な中間ノードへ転送できれば，セッション管理機能の変更にも柔軟に対応できると考えられる．また，Stream Plane のメディア処理機能は，ローカルのコンピュータシステムに必要な機能が存在しない場合，他のコンピュータから取得できることが望ましい．これらを考慮し，MidField システムの Session Plane と Stream Plane を，移動エージェント [4, 5, 6, 7] により実現する．その機能モジュール構成を図 3 に示す．

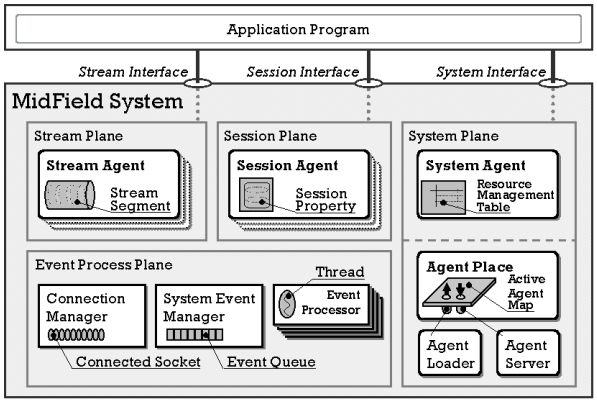

図 3: 機能モジュール構成

MidField システムの各プレーンの機能は，*Straem Interface*, *Session Interface*, *System Interface* によりアプリケーションプログラムへ提供される．これ

らのインターフェースに対応するシステムの処理は，Stream Agent, Session Agent, System Agent が対応する．Stream Agent は，RTP ストリーム処理を実現するために Stream Segment を所有し，RTP ストリームの送受信およびトランスコーディングを行う．Session Agent は，MidField セッション情報である Session Property を所有し，必要に応じて MidField システム間を移動する．また，System Agent はローカルコンピュータシステムの CPU 資源やネットワークトラフィックを監視し，システム利用者が MidField セッションへ参加する際にはアドミッションテストを実行する．

AgentPlace はエージェントの生成・起動，移動，終了処理を行い，ローカルシステム内のエージェントを管理する．そして，エージェントの移動を実現するための Agent Loader と Agent Server を所有する．Agent Place も MidField システムにおけるエージェントの 1 つである．

これらのエージェントはそれぞれ，コンピュータネットワーク上で稼動する MidField システム間でメッセージの送受信が可能である．Event Process Plane の Connection Manager は，MidField システム間を接続したソケットインターフェースを所有し，エージェント間メッセージ通信の排他的制御を行う．また，System Event Manager は，各エージェントが発行するシステム内部イベントを Event Queue に格納し，Event Processor の Thread でイベントを処理する仕組みを実現する．これにより，システム内部のイベント処理を一元管理するとともに，優先順位に基づくイベント処理が可能となる．

3 相互通信環境の動的構成

MidField システムは必要に応じて MidField セッション内に複数の RTP セッションを生成する．そして，生成された RTP セッション間にトランスコーディング機能を配置することにより，相互通信環境の動的構成を実現する．

3.1 送受信開始時の処理

MidField システムにおける RTP セッションは IP マルチキャストセッション上に実現される．従って，RTP ストリームを送信するためには，送信側のみならず，IP マルチキャスト受信側全ての帯域幅とメディア処理機能を考慮する必要がある．一方，既に流れている RTP ストリームを受信する際は，受信側の帯域幅とメディア処理機能を考慮し，必要に応じてトランスコーディング機能を利用する．これら送受信開始時の処理を図 4,5 に示す．

MidField セッション生成時には，デフォルトの通信セッションとして RTP セッションが生成される．RTP ストリームを送信する際は，図 4 に示す通り，

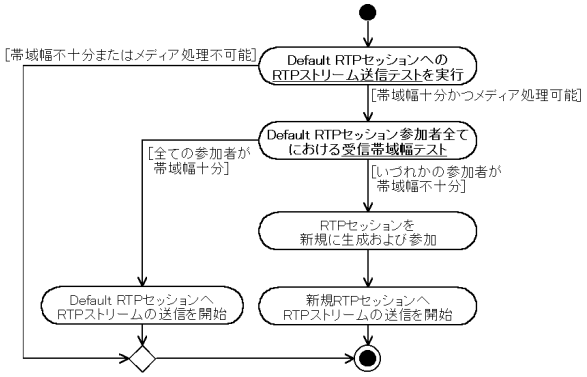

図 4: RTP ストリーム送信開始処理フロー

まずはじめにデフォルトの RTP 通信セッションへの送信が可能かどうかをテストする．必要となる送信帯域幅に対し，デフォルトセッション参加者の受信帯域幅が十分に確保できない場合，RTP セッションを新規に生成し，そのセッションへ RTP ストリームを送信する．もし要求された送信フォーマットでの RTP ストリーム送信が不可能な場合，送信フォーマットの QoS を下げるか，フォーマットを変更し，図 4 の処理を再度繰り返す．

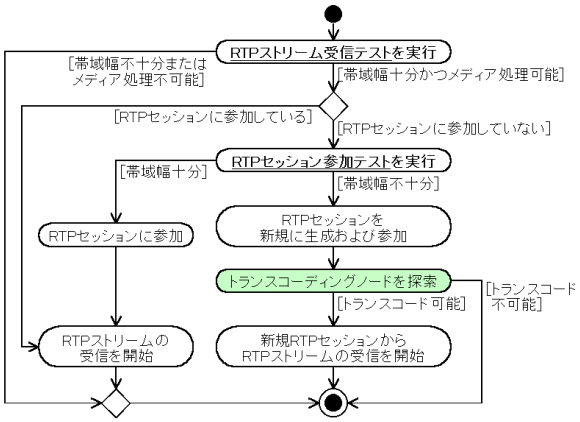

図 5: RTP ストリーム受信開始処理フロー

既に流れている RTP ストリームを受信する場合，まず始めに必要となる受信帯域幅とメディア処理機能のテストを行う．ここで，もし所望する RTP ストリームの RTP セッションに参加していない場合，そのセッションへ参加可能かどうかをテストする．RTP セッションは IP マルチキャストセッションの上に構成されるので，不用意に参加すると，所望する RTP ストリーム以外の RTP トラフィックにより受信オーバーフローを起こす可能性があるからである．受信帯域幅が十分に確保できない場合，RTP セッションを新規に生成し，そのセッションに参加する．そしてトランスコーディングノードを探索し，トランスコーディングによる RTP ストリームの受信が可能ならば，新規に生成した RTP セッションからトランスコードされた RTP ストリームを受信する．もし

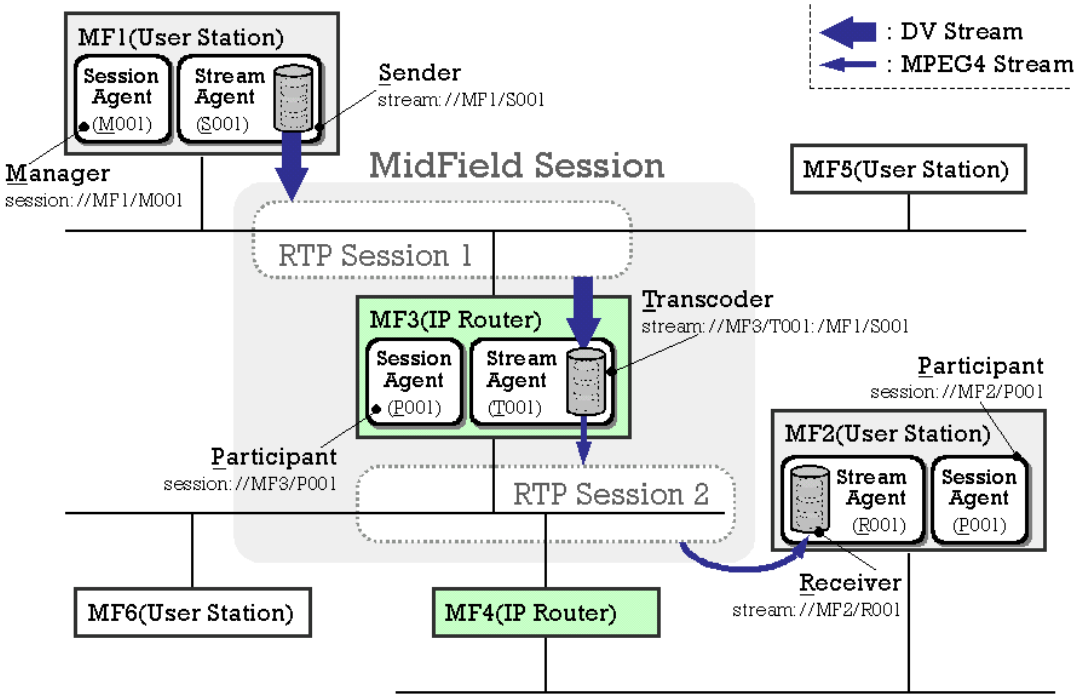

図 6: 相互通信環境の動的構成例

要求された受信フォーマットでのメディア処理が不可能な場合，送信開始時と同様に，受信フォーマットの QoS を下げるか，フォーマットを変更し，図 5 の処理を再度繰り返す．

3.2 トランスコーディングノードの探索

図 6 は，トランスコーディングノードを含む相互通信環境の動的構成例を示している．この図では，MF1 が送信する DV/RTP ストリームを MF3 がトランスコードし，MPEG4/RTP ストリームとして MF2 が受信している．以下，図 6 に従い，MidField セッションの生成からトランスコーディングノードの探索，受信までの流れを示す．

(1) MidField セッションと送信ストリームの生成

MF1 の Session Agent (session://MF1/M001) が MidField セッションを生成する．また，デフォルトの通信セッションとして，RTP セッション (RTP Session 1) を生成する．そして，Stream Agent (stream://MF1/S001) が送信ストリームを生成する．図 4 に示す送信開始時の処理を経て，デフォルトセッションへ DV ストリームの送信を開始する．

(2) MidField セッションへの参加と受信要求

次に，MF2 の利用者が MidField セッションへの参加を希望する．図 6 の MidField セッション管理者である Session Agent (session://MF1/M001) は，現在のセッション情報を含む自分自身のクローンを生成して，そのクローンが MF2 へ移動する．これが，session://MF2/P001 となる．MF2 の利用者は，既に流れている DV ストリームの受信要求を発行する．ここで，図 5 の処理の結果，DV ストリームの受信は不可能だが，MPEG4 のストリーム受信処理は可能であることがわかり，新規に RTP セッション (RTP Session 2) を生成する．そして，トランスコーディングノードの探索を開始する．

(3) トランスコーディングノードの探索

MidField システムは起動時に，稼動中のシステムを把握するための IP マルチキャストセッションへ参加している．MF2 はその IP マルチキャストセッションを利用して，稼動中の MidField システムを探す．ここでは，MF3, MF4, MF5, MF6 が稼動中であることが MF2 へ通知される．

また，MF2 は MF1 へルーティングパスを要求する．MF1 は MF2 までのルーティングパスを調べ，MF2 へ返す (MF1 – MF3 – MF4 – MF2)．

MF2 は，MF1 から取得したルーティングパス内に稼動中の MidField システムを探し，トランスコーディング可能かどうかの問合せを行う．ここでは，ルーティングパス内に MF3, MF4 が稼動しており，共にトランスコーディング可能であるとする．この場合，送信者である MF1 により近い MF3 を選択し，Session Agent (session://MF2/P001) が自身のクローンを生成し，そのクローンが MF3 に移動する．これが session://MF3/P001 となる．

ここで，稼動中の MidField システムがルーティングパス内に存在しない場合，パス内のネットワーク上で稼動している MidField システム (図中 MF5, MF6) もトランスコーディングノードの対象とする．

(4) RTP ストリームの受信

MF3 では，トランスコーディングを行うために Stream Agent (stream://MF3/T001:/MF1/S001) が生成される．そして，MF1 からの DV ストリームを受信し，MPEG4 にトランスコードして RTP Session 2 への送信を開始する．最後に，RTP Session 2 に流れ始めた MPEG4 のストリームを，MF2 の Stream Agent (stream://MF2/R001) が受信する．

こうして，トランスコーディング機能を動的に配置し利用することによって，DV ストリームを MPEG4 のストリームで受信するための環境が動的に構築される．

4 システムの設計

MidField システムは，開発言語として Java 1.3 を想定しており，クラス構成図の記述には UML 1.3 を使用している．以下，主要なエージェントの構成と，Session Agent, Stream Agent について述べる．

4.1 主要エージェント

MidField システムを構成する主要な機能モジュールはエージェントとして実現される．図 7 は，MidField システムにおけるエージェントのクラス構成を示している．

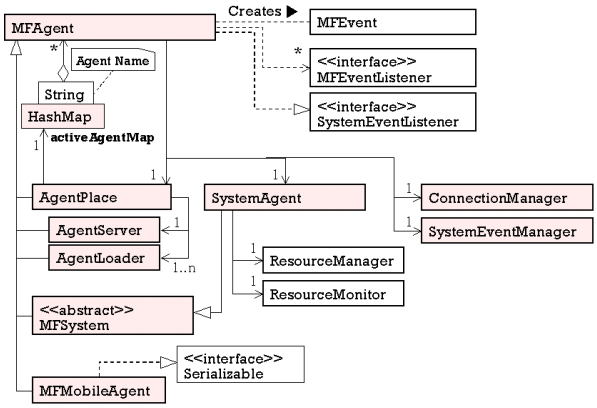

図 7: エージェントのクラス構成

MFAgent は全てのエージェントの基盤クラスであり，エージェント間メッセージ通信機能を所有する．また，MFMobileAgent は移動エージェントの基盤クラスであり，移動の際にオブジェクトシリアライゼーション機能を利用するため，Serializable インターフェースを実装している．

図 3 に示した System Plane と Event Process Plane の各機能モジュールは，AgentPlace, SystemAgent, ConnectionManager, SystemEventManager により実現される．また，アプリケーションに対する System Plane のインターフェースは，MFSystem により実現される．MFSystem は抽象クラスであり，SystemAgent の機能をアプリケーションから利用するためのメソッドで構成される．

AgentPlace は，ローカルの MidField システム内でアクティブなエージェントを管理するために HashMap を所有し，MFMobileAgent の転送を行うために AgentServer と AgentLoader を利用する．AgentLoader は，Java 1.3 における ClassLoader を拡張した NetworkClassLoader を所有し，ネットワーク上でオンデマンド型のクラスロードを実現する．

4.2 Session Agent

Session Agent は，上述した MobileAgent クラスを拡張して実現される．図 8 は，そのクラス構成を示している．

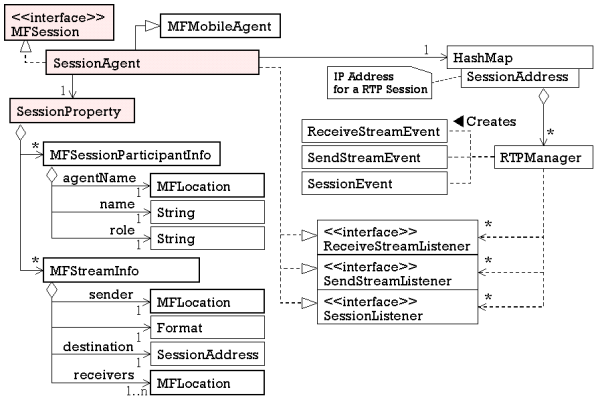

図 8: SessionAgent のクラス構成

SessionAgent は，SessionProperty を 1 つ所有し，SessionProperty は参加者情報とストリーム情報を複数所有する．また，MidField 内に存在する RTP セッション毎に，RTPManager を管理する．RTPManager は，JMF 2.0[8] のクラスであり，RTP のセッション管理機能を所有している．RTPManager は，送受信ストリームと RTP セッションにおける各種のイベントを生成し，そのイベントを SessionAgent が処理する．

アプリケーションに対する Session Plane のインターフェースは，MFSession により実現される．MFSession は，SessionAgent の機能の一部をアプリケーションに公開するためのインターフェースである．アプリケーションは，MFSession を利用して，MidField セッションの生成やセッションへの参加が可能となる．

4.3 Stream Agent

Stream Agent は，RTP ストリームを所有し，ストリームの送受信およびトランスコーディング機能を実現する．MidField システムにおける RTP ストリームは，JMF 2.0 を利用して実現される．図 9 は，その概要を示している．

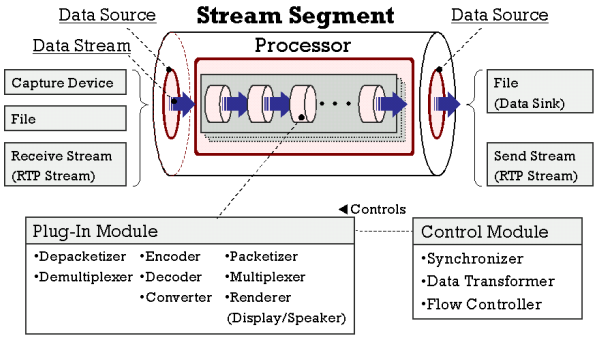

図 9: ストリームセグメントの構成

Stream Segment は，Data Source と Processor により構成される．Processor は 1 つ以上の入力 Data Source に対してメディア処理を施し，1 つの出力 Data Source を生成する．Processor は，複数の Plug-In Module をつなぎ合せることにより，要求されたメディア処理を実現する．各種の Plug-In Module には，制御用の Control Module が存在し，MidField システムの Stream Plane における各層の制御は，これらの Control Module により実現される．

図 10 は，Stream Segment を含む Stream Agent のクラス構成を示している．実装における StreamSegment クラスは抽象クラスであり，Sender, Transcoder, Receiver といった各 StreamSegment に共通な機能とクラスを所有する．Sender および Receiver は，それぞれ 1 つの RTPManager を所有し，RTP の送受信ストリームに対する RTCP メッセージをイベントとして処理することが可能となる．Transcoder は，Sender と Receiver 両方の機能を所有する．

アプリケーションに対する Stream Plane のインターフェースは，MFStream により実現される．MFStream は MFSession 同様，StreamAgent の機能の一部をアプリケーションに公開するためのインターフェースである．

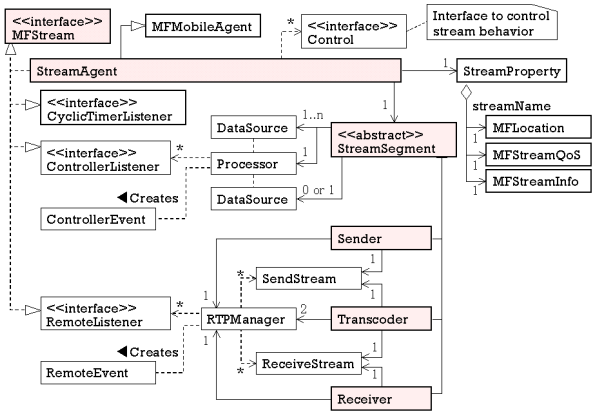

図 10: StreamAgent のクラス構成

5 まとめ

本稿では，トランスコーディング機能を適切な中間ノードへ配置することによって，相互通信環境を動的に構成するためのミドルウェアの設計を行った．ミドルウェアのアーキテクチャとして，エージェント指向の MidField システムの概要を示し，動的な相互通信環境の構成について述べ，システムの主要なクラス構成を図示した．現在，クラスメソッド呼び出しレベルでの詳細設計を進めている．

今後，JGN(Japan Gigabit Network) 上にプロトタイプシステムを構築し，相互通信環境の動的構成機能の評価を行う予定である．

参考文献

[1] 杉浦，小川，中村，村井，”民生用 DV を用いたインターネットビデオ会議システム”，情報処理，Vol.40, No.7, pp.698-702, Jul. 1999.

[2] 勝本道哲，原田雅博，中川晋一，”D1 over IP による高品位動画像転送・蓄積システムの設計”，情報処理学会 マルチメディア通信と分散処理研報，No. 95, pp.85-90, 1999.

[3] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson.,”RTP: A Transport Protocol for Real-Time Applications”, RFC 1889, January 1996.

[4] Shiratori N., Sugawara K., Kinoshita T. and Chakraborty G.,”Flexible Network:Basic Concepts and Architecture”, IEICE Trans. Communication, Vol.E77-B, No.11, pp.1287-1294, 1994.

[5] Pauli
Misikangas and Kimmo Raatikainen.,”Agent Migration between Incompatible Agent Platforms”, Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS2000), pp.4-10, Apr. 2000.

[6] 橋本浩二, 柴田義孝, 白鳥則郎, ”移動エージェントによる QoS 保証機能”, 情報処理学会論文誌, Vol.42, No.11, pp.2771-2779, Nov., 2001.

[7] 冨川, 高井, ”移動プロクシエージェントによるアクセス経路選択”, 電子情報通信学会論文誌 D-I Vol.J84-D-I, No.2, pp.183-190, Feb., 2001.

[8] http://java.sun.com/products/java-media/jmf/