TR-AC-0050

017

マルチエージェントによる適応的QoS制御方式

小菅 昌克　山崎 達也
荻野 長生*　松田 潤**

*KDD研究所　**大阪学院大学

2000.12.27

ATR環境適応通信研究所

マルチエージェントによる適応的QoS制御方式

A Multi-Agent Based Adaptive QoS Management Scheme.

小菅　昌克[1]　山崎達也[1]　荻野　長生[2]　松田　潤[3]

Masakatsu KOSUGA[1] , Tatsuya YAMAZAKI[1] , Nagao OGINO[2] , and Jun MATSUDA

[1]ＡＴＲ環境適応通信研究所

〒619-0288 京都府相楽郡精華町光台２丁目２番地

E-mail : {kosuga, yamazaki}@acr.atr.co.jp

[2]ＫＤＤ研究所

〒356-8502 埼玉県上福岡市大原２丁目１番１５号

E-mail : ogino@kddlabs.co.jp

[3]大阪学院大学

〒564-8511 大阪府吹田市岸部南二丁目３２番１号

E-mail : matsuda@utc.osaka-gu.ac.jp

あらまし 通信ネットワークおよび通信端末としてのパーソナルコンピュータやワークステーションの高速化に伴い，近年，ストリーム型メディアを含むマルチメディア通信を基本とした様々なアプリケーションが出現している．このような通信アプリケーションには，ユーザ要求および端末やネットワークの状況に応じて，自らの目的に合うようにストリームのQoSを決定し，調整する仕組みが必須である．本論文では，複数のエージェントによる直接的あるいは間接的な連携機能を利用して，様々な通信アプリケーションの共通基盤となりうる適応的QoS制御の仕組みを実現するためのフレームワーク(MARM:Multi-Agent Resource Management)を提案し，本フレームワークを実装して動作確認を行う．

目次

１．まえがき

高速ネットワーク技術の発展および通信端末となるパーソナルコンピュータやワークステーションの高速化に伴い，近年，ストリーム型メディアを含むマルチメディア通信を基本とした様々なアプリケーションが出現している．これらのアプリケーションに対しては，ネットワークだけでなく，エンドシステムまで含めた，エンドツーエンドでのQoS保証を行う必要がある．エンドツーエンドでのQoS保証を目的として，既に複数のQoSアーキテクチャが提唱されている(e.g. OMEGA[1], QoS-A[2])．これらのQoSアーキテクチャでは，ネットワークとエンドシステムのQoSを形式的に記述・構成可能とするQoSインタフェースセットを定義し，QoSの制御と管理メカニズムを統合するためのフレームワークを提供している[3]．このようなQoSアーキテクチャを利用することで，通信アプリケーションは，QoSの形でストリームを制御することが可能になる．また，ベストエフォートなIPネットワークにおいても，RSVP[4]やDiff-Serv[5]といった技術が出現し，マルチメディアストリームのQoS制御を行えるようになってきている．しかしながら，端末やネットワークの構成，ユーザ要求は千差万別であり，その変動を予測するのは困難である．例えば，デスクトップ端末と携帯端末では，利用できるリソース量も異なるし，ユーザが求めるQoSも異なる．また複数のアプリケーションが起動している場合は，利用できるリソースは，他のアプリケーションの動作に依存する．従って，通信アプリケーションには，ユーザ要求および端末やネットワークの状況に応じて，自らの目的に合うようにストリームのQoSを決定し，調整する仕組みが必要になる．このような適応的QoS制御の仕組みを個々の通信アプリケーションに組み込むと，通信アプリケーションの複雑化を招くので，様々な通信アプリケーションの共通基盤として，適応的QoS制御の仕組みを実現することが望ましい．近年，QoS制御にエージェント技術を使う試みが提案され始めた[6-8]．しかし，様々な通信アプリケーションの共通基盤となりうる適応的QoS制御の仕組みは提案されていない．本論文では，複数のエージェントによる直接的あるいは間接的な連携機能を利用して，このような適応的QoS制御の仕組みを実現するためのフレームワーク(MARM:Multi-Agent Resource Management)を提案する．

本論文では，まず２章で，マルチエージェントによる適応的QoS制御フレームワークの概要を示し，３章でフレームワーク上位層におけるQoS交渉と，フレームワーク下位層におけるQoS調整について説明し，適応的QoS制御フレームワークの詳細を述べる．そして４章で，実験システムへの本フレームワークの実装についてついて述べる．最後にまとめを行う．

２．マルチエージェントによる適応的 QoS 制御フレームワーク

２．１　マルチメディア通信システムのモデル

まず初めに，QoSマネージメントの観点から見たマルチメディア通信システムのモデルを図１に示す．各端末内ではいくつのかの送信アプリケーションと受信アプリケーションが同時に動作していることが考えられる．図１は，その一局面だけを取り出したものである．

図１に示すように，マルチメディア通信システムのレベルごとに異なったQoSが定義される．ユーザQoSはユーザが実感できるサービスの品質要求に相当し，おそらく自然言語による抽象的な表現で定義される（例えば，“非常に大きい”，“やや小さい”，“滑らか”等）．それぞれのマルチメディアアプリケーションは，そのアプリケーションが扱うメディアのQoSによって，アプリケーションQoSが

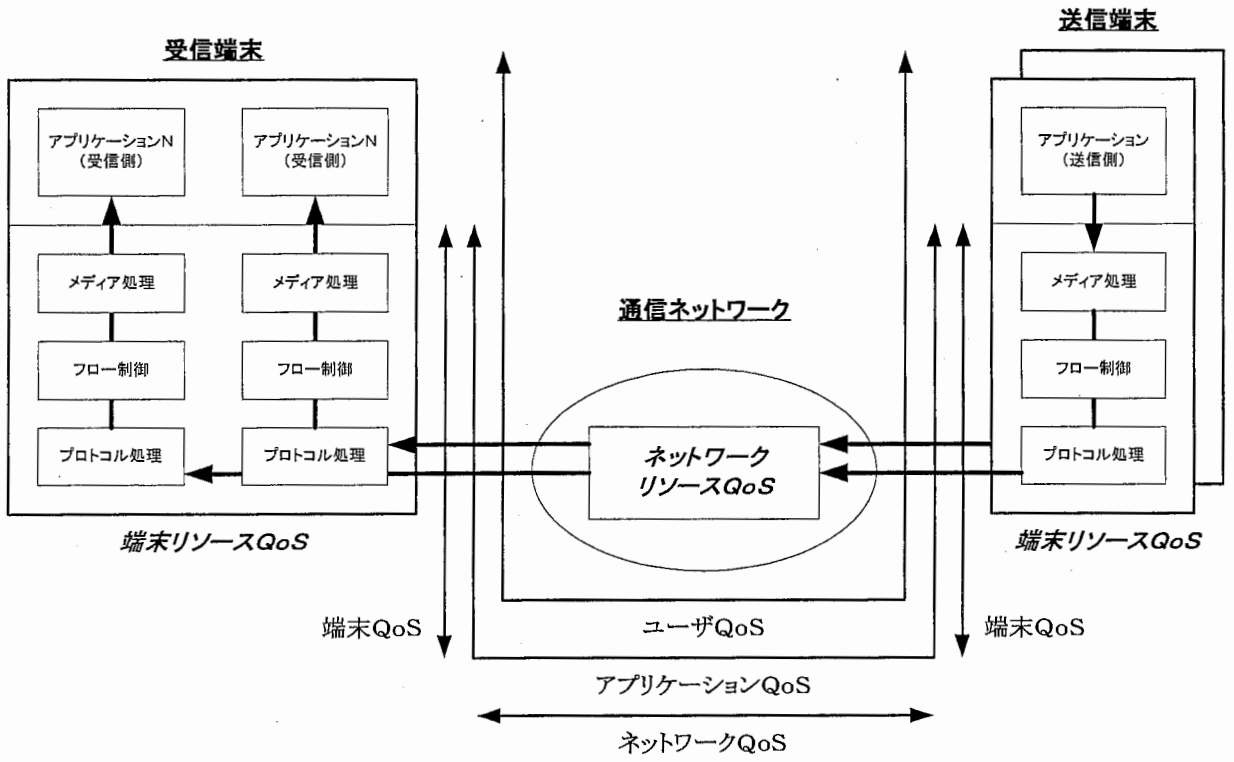


図1．マルチメディア通信システムのモデル

定義される．ビデオストリームの場合，アプリケーション QoS はフレームレート，フレームサイズ，量子化パラメータ等で定義される．遅延やジッターで定義されることもある．通信ネットワークはネットワーク QoS を伴ったメディアストリームを提供する．ネットワーク QoS は，スループット，遅延とその揺らぎ，ビット誤り率，パケット損失率などで定義される．端末 QoS はアプリケーション QoS とネットワーク QoS の両方に依存するが，アプリケーション QoS やネットワーク QoS とは別のものとして定義される．

通信リソースはまた，リソース QoS として定義される．端末リソース QoS はＣＰＵ使用率や割り当てられたメモリ量などに相当する．周期スレッドによってメディアデコード処理のようなメディア処理が行われるときには，スレッドの期限や長さが端末リソース QoS として含まれる [9]．ネットワークリソース QoS は，各通信リンクに割り当てられた帯域や，各通信ノードに割り当てられたバッファサイズなどによって定義することができる．

メディア処理はメディアのエンコード処理やデコード処理，メディア同期などを含む．フロー制御機能は，端末リソースの不足やネットワーク品質の劣化からメディアストリームを保護する．この機能は，適切なメディア情報フローの制御とアプリケーション QoS のわずかな劣化によって，メディア情報の破壊を回避する．ネットワークリソースは，受信端末内で端末リソースが不足した場合に，通信ネットワークにフローフィルタリング機能を追加することや，通信ネットワーク内のメディア情報フローを適切にフィルタリングすることで，効率的に利用される [10]．しかも，通信ネットワーク内のフローフィルタリング機能は端末リソースに加えて，ネットワークリソースの不足からもメディアストリームを保護できる．

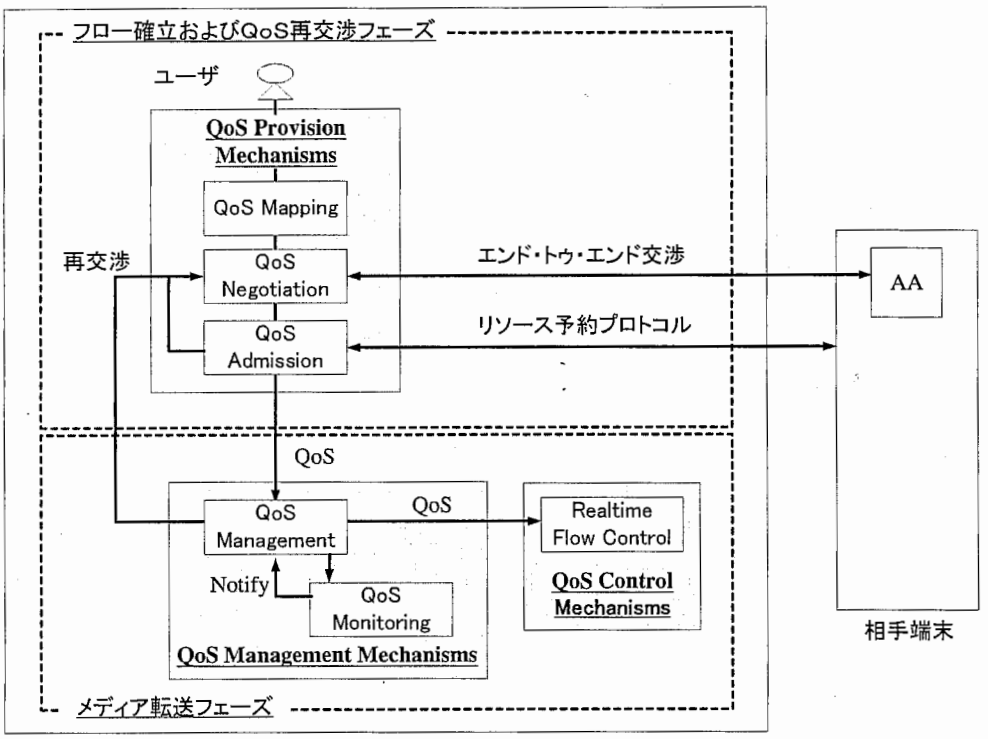


図2. QoS 制御フロー

2．2　機能からみた QoS 制御特性の分類

エンドツーエンドの QoS 制御を行う際に必要な機能は大きく以下の３つに分類することができる[3].

- QoS Provision 機能群
- QoS Control 機能群
- QoS Management 機能群

QoS Provision 機能群は,フローの確立と QoS 再交渉といった静的なリソースマネージメントを行う.一方, QoS Control 機能群と QoS Management 機能群は, メディア転送時の動的なリソースマネージメントを行う. QoS Control 機能群は QoS Management 機能群より処理のタイムスケールが短いリアルタイムなフロー処理,例えばフローのスケジューリングやフロー間の同期等を行う. QoS Management 機能群は, モニタリングやより制御間隔が長い処理を行い, QoS の維持や劣化, スケーラビリティについて責任を負う. 図２でこれらの機能群を中心に QoS 制御のフローを示す.

図２に示すように, QoS 制御は大きく“フロー確立および QoS 再交渉フェーズ”と“メディア転送フェーズ”の２つのフェーズに分けることができる. これらのフェーズでの QoS 制御特性は異なったものとなっている（表１）. つまり, フロー確立および QoS 再交渉フェーズでは, リソースマネージ

表1. 各フェーズにおけるリソースマネージメントの特徴

	フロー確立および QoS再交渉フェーズ	メディア転送フェーズ
特性	静的	動的
処理の範囲	グローバル	ローカル
処理のオーダー	数百ミリ秒～数秒	～数百ミリ秒

メントは静的であり相手端末を含めたグローバルな問題である．反面，メディア転送フェーズでは，リソースマネージメントは動的でありローカルな特性を持つ．また，処理の時間間隔は，フロー確立およびQoS再交渉フェーズでは数百ミリ秒～数秒に対して，メディア転送フェーズでは数百ミリ秒までであり，リアルタイムな処理を要求される．このことから，それぞれのフェーズにおいては異なったQoS制御方式が必要になると考えられる．

２．３　適応的QoS制御プラットフォームのモデル

図３に階層化されたマルチエージェントシステムにおける直接的および間接的な協調に基づいた，適応的QoS制御プラットフォームのモデルを示す．図３中には５種類のエージェントが存在する．異なったQoSのレベル間においては，それらのエージェントはQoSマッピング関数を持つ[11]．図４は異なったQoSのレベル間でのマッピングの関係を示している．

パーソナルエージェント（ＰＡ）はユーザを支援し，ユーザQoSとアプリケーションQoS間のQoSマッピングを行う．ＰＡはこのQoSマッピングをユーザの特性や好みを考慮して行う．その結果，ＰＡはアプリケーションQoSとそれに相当するユーザのユーティリティとの関係を表したユーティティ関数を生成する．

提案モデルでは，適応的QoSマネージメントは２つの階層を持ったマルチエージェントシステムによって実現される．マルチエージェントシステムの上位層は，アプリケーションエージェント（ＡＡ），ネットワークインタフェースエージェント（ＮＩＡ）およびネットワークエージェント（ＮＡ）から構成される．上位層では，フロー確立およびQoS再交渉フェーズの処理を担う．この層の目的は，現在の端末とネットワークの状況の中で，ユーザの好みを最大限に満たすQoSを選びだすことである．また，環境の変動を押さえるために，長期的な観点を含んだQoS交渉を行う必要もある．フローの確立時のQoS交渉は実時間性に対する要求はそれほど強くない．従って，分散人工知能の分野で研究されている高度な分散問題解決手法[12]が利用可能であり，熟考型エージェントであることが好ましい．これらのエージェント間では，直接的なメッセージの交換によるQoS交渉を行う．一方，マルチエージェントシステムの下位層はストリームエージェント（ＳＡ）によって構成される．下位層では，メ

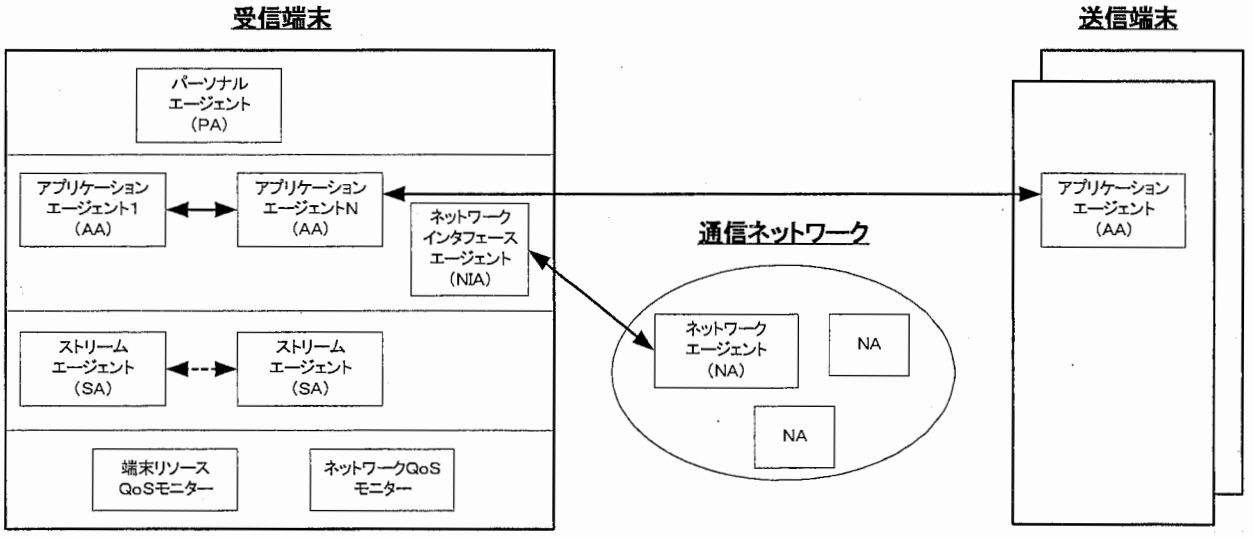


図3．適応的QoS制御プラットフォームのモデル

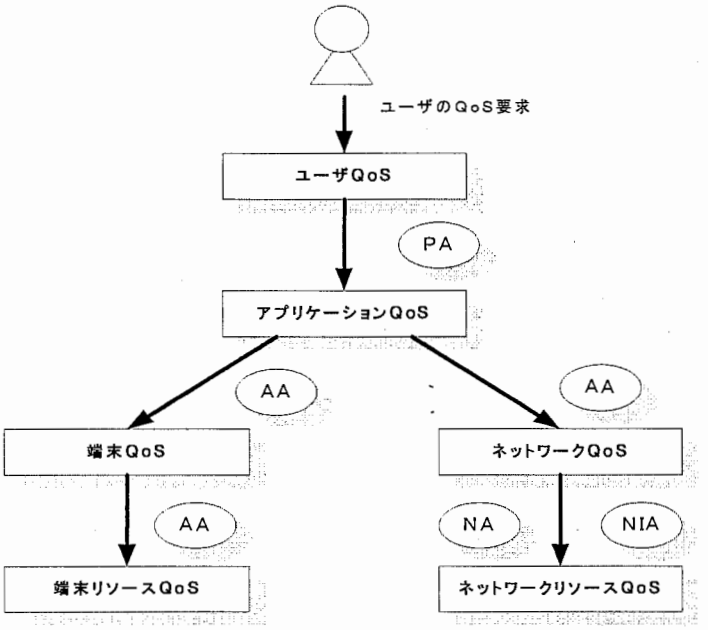


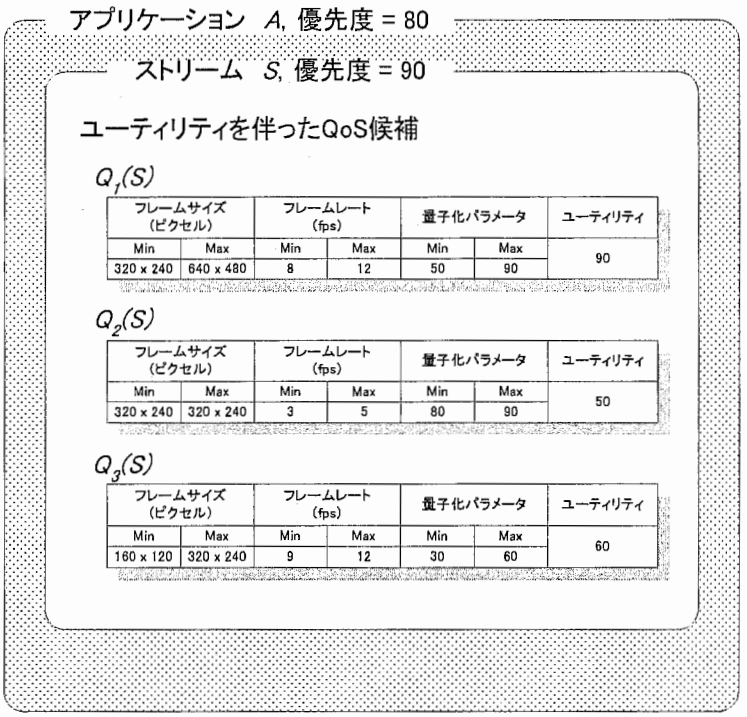
図4. QoS のレベルと QoS マッピング

ディア転送フェーズの処理を担う．この層の目的は，ユーザ要求を最大限に反映しながらも，効率よく QoS 調整を行うことである．QoS 調整はアプリケーション実行時に随時行われるため，実時間性への要求が非常に強い．よって，ＳＡはリアクティブエージェントであることが好ましい．リアクティブエージェントとは，記号知識を利用しない比較的簡単な構造を持つエージェントであり，エージェントの動作と外部環境を直接的に関連付け，分散的で非集中的なエージェント間の相互作用を利用することにより，外部環境に対して自律的に反応して動作する[13]．つまり，各エージェントは，共有メモリ等をつかった間接的な協調により QoS 調整を行う．このようにプラットフォームでは，上位層のマルチエージェントシステムにおける QoS 交渉によって，システム状態の大きな変化やユーザ要求の変化に柔軟に対応し，下位層のマルチエージェントシステムにおける QoS 調整によって，小さなシステム状態の変化にすばやく対応する．

ＡＡはマルチメディアアプリケーションごとに生成される．ＡＡはアプリケーション QoS から端末リソース QoS およびネットワーク QoS への QoS マッピングを行う．その後，各アプリケーションのための端末リソースが，端末内の関連するＡＡ間の端末内交渉によって割り当てられ，送信側ＡＡと受信側ＡＡとの端末間交渉によって送信側と受信側での QoS のギャップを解消する．

複数のＮＡが通信ネットワーク内に分散している．各ＮＡはローカルにネットワークリソースを管理し，ネットワーク QoS からネットワークリソース QoS への QoS マッピングを行う．ネットワーク QoS は，ＮＩＡとＮＡ間の QoS 交渉によって決定され，この交渉はネットワーク QoS ベースに行われる．ＮＩＡはネットワークと端末間の接続リンクを管理する．例えば，有線ネットワークから無線ネットワークへの繋ぎ代えや，通信リンクにおける制約などを管理している．ＡＡはネットワークに関する QoS 交渉をＮＩＡに委任する．

各ＳＡはメディアストリームごとに生成される．もし，マルチメディアアプリケーションが複数のメディアストリームを扱う場合，複数のＳＡが一つのＡＡに対して生成される．各ＳＡはそれぞれ端末リソースとネットワークリソースのモニターを行い，その変化に応じて QoS 調整を自律的に行う．



フレームサイズ (ピクセル)		フレームレート (fps)		量子化パラメータ		ユーティリティ
Min	Max	Min	Max	Min	Max	
320 x 240	640 x 480	8	12	50	90	90

フレームサイズ (ピクセル)		フレームレート (fps)		量子化パラメータ		ユーティリティ
Min	Max	Min	Max	Min	Max	
320 x 240	320 x 240	3	5	80	90	50

フレームサイズ (ピクセル)		フレームレート (fps)		量子化パラメータ		ユーティリティ
Min	Max	Min	Max	Min	Max	
160 x 120	320 x 240	9	12	30	60	60

図5. QoS パラメータセットの例

この QoS 調整は，あらかじめ決められた範囲内でしか行わず，ローカル内での QoS 調整機能によって実現される．この QoS 調整の範囲は，ＰＡによって事前に与えられ，ＡＡによって状況に応じて選択，変換される．この範囲内では，アプリケーション QoS はベストエフォートとして処理される．ストリームの優先度や QoS パラメータの優先度もまた，ＰＡによって事前に与えられる．

システム状態の変化が比較的大きく，あらかじめ与えられた QoS 調整の範囲を超えてアプリケーション QoS を調整する必要がある場合は，ＳＡは QoS の再交渉をＡＡに要求する．従って，もし QoS 調整の範囲が非常に狭い場合は，QoS 交渉は頻繁に発生し，通信はしばしば中断される．

２．４　QoS パラメータセットとユーティリティ

上位層で交渉によって最適に QoS 決め，下位層でその QoS を維持するように QoS を調整する方法は，動的な環境ではうまくいきそうにない．交渉完了時点での最適な QoS はその次の時点では最適とは限らないし，きっちりと決められた QoS は調整の幅が限られるため QoS の再交渉が頻発する．従って，再交渉の回数を減らしてプラットフォームの上位層と下位層との独立性を高め，ＳＡが自律的に動作できるようにするには，上位層から下位層に渡される QoS に工夫が必要である．そこで，その受け渡される QoS を，範囲を持ち，制御のポリシーを情報として付加したものとして考える．このような情報を付加された QoS を QoS パラメータセットと呼び，ユーザ要求として，ストリームに対して複数の QoS パラメータセットを指定できるとする．QoS パラメータセットを以下のように定義する（図５）．

- メディア種別に応じた複数の QoS パラメータを持つ．例えば動画では，圧縮方式，フレームレート，画像サイズ，表示色数，量子化レベル等が考えられる．
- QoS パラメータセットごとにユーティリティ値を持つ．ユーティリティ値は１から１００までの整数で指定し，ユーザ要求の度合いを表す．数値が大きいほうがユーザ要求の度合いが高く，この値が１００の場合はユーザ要求が保証型であることを表す．

- 各 QoS パラメータの値は最大値と最小値で指定し，その幅の間では同じユーティリティ値をとるものとする．つまりその範囲内ではベストエフォートとして QoS を制御してよい．実際，ユーザにとって多少の品質の変化は気にならない（例えば，フレームレートが５ｆｐｓから４ｆｐｓになったとして，気が付くであろうか？）ことからも，この前提が妥当であるとわかる．
- アプリケーション，ストリーム，QoS パラメータごとの優先度を持つ．

複数の QoS パラメータセットから得られるリソース QoS とユーティリティ値から，そのユーザのユーティリティ関数を導くことができる．表２は，ある QoS パラメータセットに各 QoS を実現するのに必要な通信帯域の値を加えて表にしたものである．この表の帯域とユーティリティの値は，ユーティリティ関数

$$u_1(x) = u_{\max} - \frac{u_{\max}}{1+\gamma x}$$

において，u_{max}=100，γ=1/100 とした場合（表３）と一致する．u_{max}は最大ユーティリティ，γは帯域の増加に対してユーザ満足がどれだけ向上するかを決定するパラメータである．図６はこのユーティリティ関数のグラフである．この他，ユーティリティ関数はさまざまなものが想定できる[14]．ま

表２．QoS パラメータセットの例

Candidate	Size	Frame rate	Quality	Bandwidth (Kbps)※	Utility
#1	640 x 480	12	90	32555.9	100
#2	320 x 240	8	90	467.0	82
#3	320 x 240	8	50	169.5	63
#4	320 x 240	3	50	64.8	39
#5	160 x 120	3	50	33.4	25

表３．ユーティリティ関数の値
（u_{max}=100，γ=1/100）

帯域	ユーティリティ
33.4	25
64.8	39.3
169.5	62.9
467	82.4
32555.9	99.7

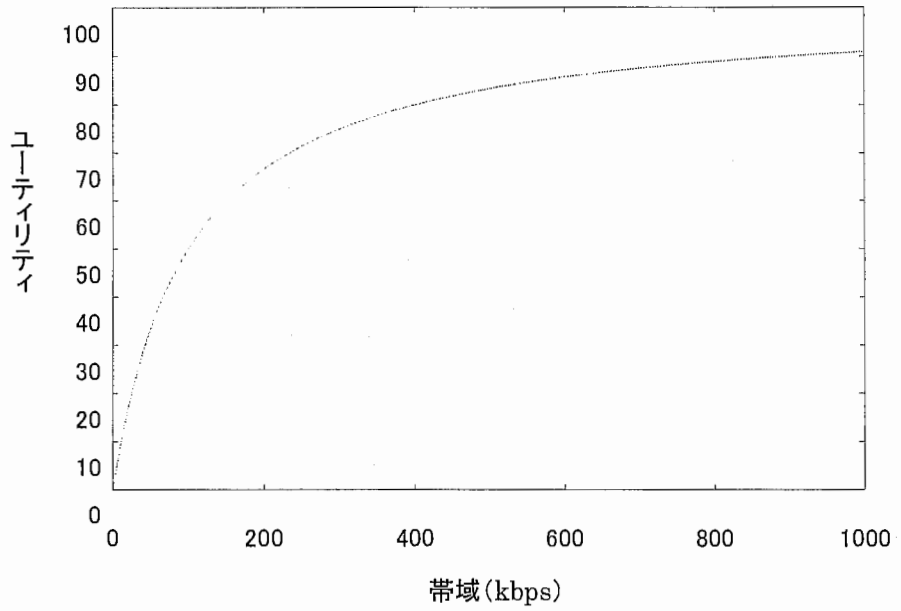


図6. ユーティリティ関数のグラフ

た我々は，ユーザとシステムのインタラクションを通してユーザの要求・嗜好を獲得し，ユーザアダプティブなQoSパラメータ調整を行う方法も検討し，提案している[15]．

3． 適応的QoS制御フレームワークの構成

3．1 QoS交渉

3．1．1 端末内QoS交渉

QoS交渉を必要とするＡＡ（例えば，新しく起動されたアプリケーションのＡＡ）は，QoS交渉の対象となる端末内のＡＡに，QoS交渉の要求メッセージを投げる．この要求メッセージを投げるエージェントをマスタエージェントと呼ぶ．要求メッセージを受け取ったＡＡは，QoS交渉が可能ならばその旨を返答する．その際，自分のQoSパラメータセットも通知する．各エージェントからの返答を受け取ったマスタエージェントは，それぞれのＡＡが持つQoSパラメータセットのユーティリティと優先度を利用して，式(3．2)で表されるリソース制約条件のもとで，統合ユーティリティ値Ｕを最大にするようなQoSパラメータセットを各ストリームに対して選択する．

$$U = \sum_{S} w(S)\log u(S,q) \qquad (3.1)$$

$$\sum_{S} r_m(S,q) \leq R_m \qquad (3.2)$$

ここで，$u(S, q)$ は，ストリームSのQoSがqであるときの個別ユーティリティであり，$w(S)$は，アプリケーションの優先度を考慮したストリームSの優先度である．また，$r_m(S, q)$ は，ストリームSをQoS qで処理するのに必要なリソースmの量である．R_mは，リソースmの利用可能限度量である．

3．1．2 端末間QoS交渉

端末内QoS交渉の結果決定したストリームの受信品質が，そのストリームの元々の送信品質を上回る場合には，そのストリームの送信端末との間でQoS交渉を行う必要が生じる．また本フレームワークに含まれないアプリケーション等が起動されて，利用できるリソース量が減少した時には，ストリームの送信品質を下げるために，そのストリームを受信している複数の受信端末との間でQoS交渉を行う必要が生じる．ここでは，前者の受信品質を上げるQoS交渉をp-p交渉，後者の送信品質を下げるQoS交渉をp-mp交渉と呼ぶ．端末間QoS交渉は，端末のユーティリティ値に基づいて行う．端末のユーティリティ値は，例えば，その端末で受信しているすべてのストリームの個別ユーティリティの総和として定義する．ここでは，あるストリームの受信側端末では，ストリーム受信品質の上昇とともにユーティリティ値も上昇するものと考える．逆に，あるストリームの送信側端末では，ストリーム送信品質が上昇すると，他のストリームを受信するために利用できるリソース量が減少し，ストリーム受信品質が落ちるので，ユーティリティ値は減少するものと考える．交渉は，受信側端末のユーティリティ値と送信側端末のユーティリティ値が，ある決められた関係を満足するようなストリー

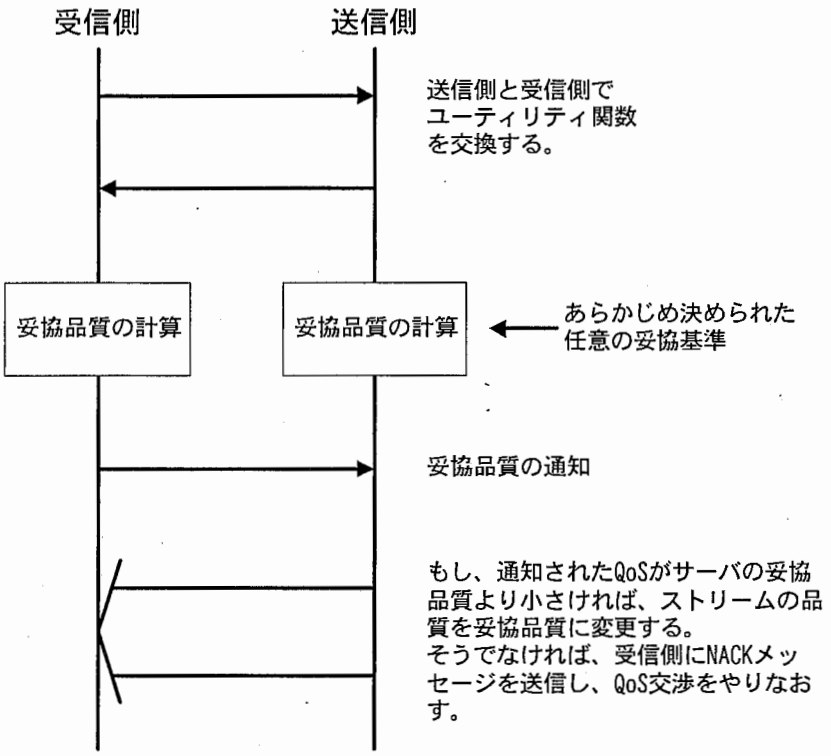


図7. p－p QoS 交渉プロトコル

ムの送受信品質を，目標の妥協品質として行われる．図７に p-p 交渉，図８に p-mp 交渉の具体的な交渉手順を示す[16]．交渉は，交渉対象のストリームを扱っている送信側アプリケーションエージェントと受信側アプリケーションエージェントとの間でメッセージをやり取りすることによって行われる．p-mp 交渉の場合には，交渉が必ず収束することを保証し，かつやり取りされるメッセージ数を削減するために，送信側アプリケーションエージェントをマスタエージェントとし，送信側アプリケーションエージェントと各受信側アプリケーションエージェントとの間でのみメッセージをやり取りする．図７および図８におけるユーティリティ関数情報とは，交渉対象のストリームの品質と端末におけるユーティリティ値との関係についての情報である．交渉開始時に，送受信端末間でユーティリティ関数情報を交換することによって，互いに独立に妥協品質を計算することができる．これによって，互いに相手が自分にとって不利な妥協品質計算を行っていないと言うことを確認した上で，ストリームの品質を妥協品質に変更できる．p-mp 交渉の場合には，すべての受信端末が納得できる送信品質を最終的な妥協品質として，送信側端末がこれをすべての受信端末に通知する．図７および図８に示された手順では，あらかじめ決められた任意の妥協基準に基づいて妥協品質の計算を行うことができる．妥協基準とは，妥協品質における受信側端末のユーティリティ値と送信側端末のユーティリティ値の関係を示すものである．妥協基準としては，例えば，協調型ゲーム理論における公平性の概念の利用が考えられる．公平性の概念は，ユーティリティ関数が凸関数であるという条件が必要だが，交渉目的に応じて，様々な交渉解を得ることができる[17]．例えば，送信側および受信側端末のユーティリティ値の積を最大にするストリーム品質を妥協品質とする場合，ユーティリティ値の和を最大にするストリーム品質を妥協品質とする場合，両者のユーティリティ値を等しくするストリーム品質を妥協品質とする場合等が考えられる．また，端末のユーティリティ値として，交渉開始時の要求品質に対応したユーティリティ値で正規化した値を使用することも考えられる．例として図９は，正規

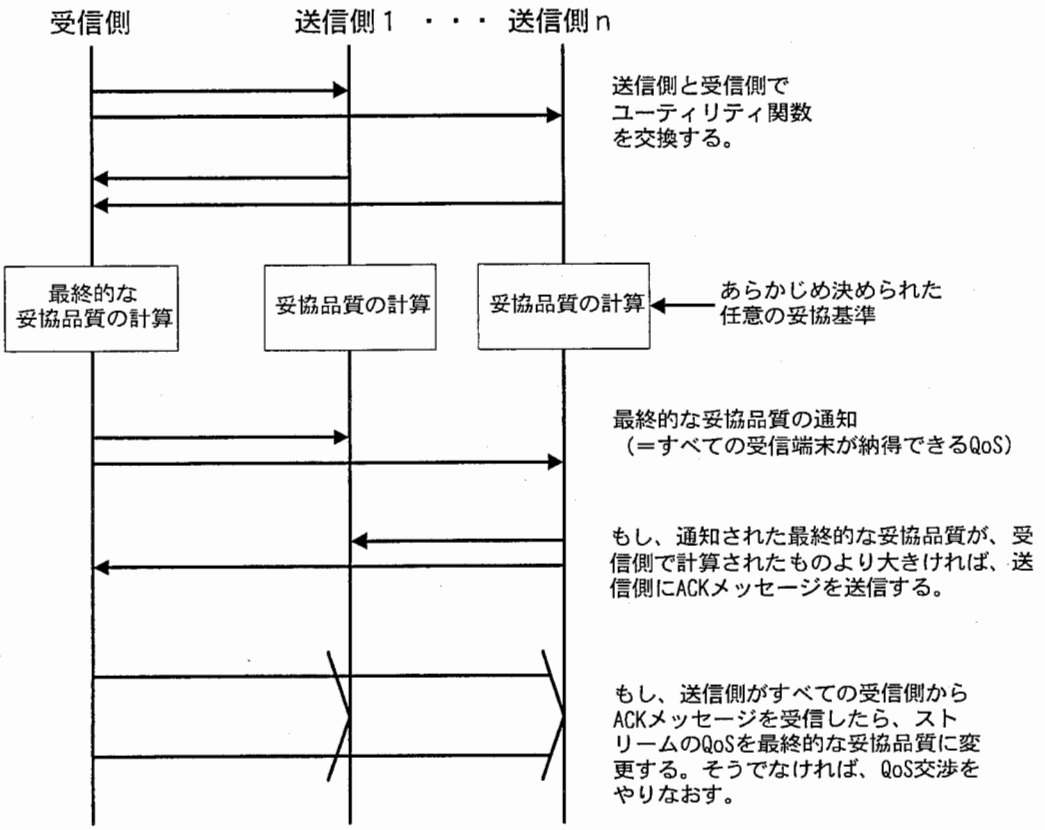


図8. p－mp QoS 交渉プロトコル

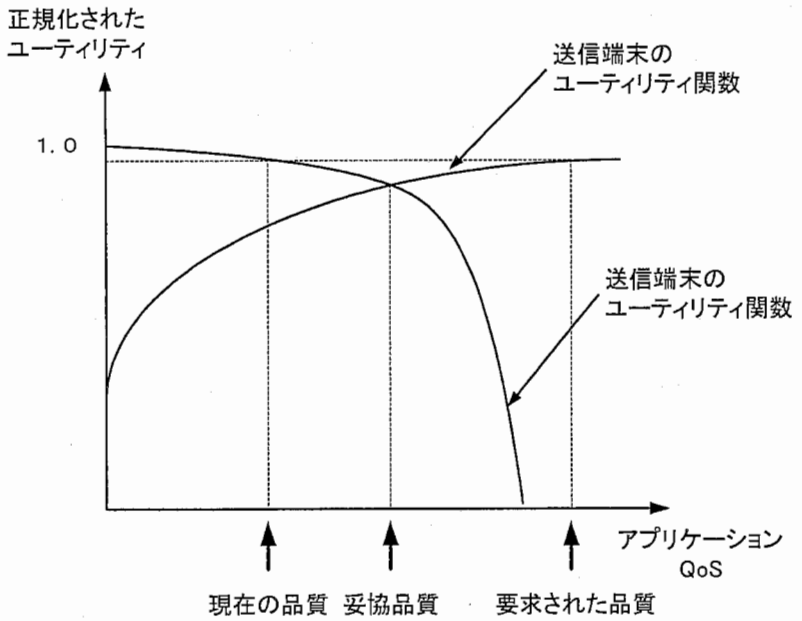


図9. 妥協品質計算法の例

化された送受信端末のユーティリティ値を等しくするようなストリーム品質を妥協品質とする妥協基準を用いた際の，p-p 交渉における妥協品質計算法を示している．

３．１．３　ネットワークとの交渉

ネットワークリソース制約（ネットワークリソースにおける（３．２）式の右辺 R_m の値）は，ＮＡと交渉を行うことで決定する．ただし，ＡＴＭネットワークやＲＳＶＰのような保証型のコネクションを提供するネットワークでは，ネットワークのアドミッション制御を通して，ネットワークリソースの割り当てを行うことになるため，ネットワークエージェントは存在せず，アドミッションコントロールの結果をネットワークリソース制約として利用する．

企業ネットワーク等の利用者のポリシーが明確であり，ネットワーク管理者が全体の統括を行うようなネットワークにおけるネットワークリソースマネージメントとして，ポリシー制御型ネットワーク (Policy-based networking：PBN) が提案されている [18]．PBN では，管理者がネットワークを運用する際に予め知識や経験として持っている運用ポリシーをネットワークシステム内で保持し，その保持情報に従って，自律的に運用・管理を行っていく．ＰＢＮでは，ポリシーサーバがネットワークエージェントとなりうる．

ホームネットワークのようなネットワークでは，ネットワークを利用する人々の行動の制約が少なく，その利用目的が多種多様であり，計画性がないことが考えられる．従って，明確なポリシーを決定することも困難であると考えられる．そのようなネットワークにおいて効率的なネットワークリソースの割当てを行うためには，ユーザ端末の自律分散的な処理に基づき，ボトムアップ的にネットワークリソース割り当てを行う仕組みが必要となる．近年，分散資源割当てを実現するためのメカニズムとして市場モデルが注目され [12]，ネットワークリソース割り当てへの適用も試みられている [19]．市場モデルにおいては，交渉管理システムがネットワークエージェントになるであろう．

このように，ネットワークの性質によりネットワークリソースマネージメントはさまざまであり，従ってネットワークエージェントもさまざまなものがあると考えられる．接続されるネットワークエージェントに応じたネットワークインタフェースエージェントを使用することで，さまざまなネットワークに対応することが可能となる．

３．１．４　再交渉

ＡＡが生成したすべてのＳＡで QoS 調整が不可能になった場合，新しい状況のもとでＡＡは再び QoS 交渉を行い，新しい QoS を選出する．

３．１．５　シミュレーション実験

動作確認のために，端末間および端末内交渉をシミュレータ上で実装し，その有効性を検証するために，以下のシナリオで動作実験を行った．この実験では，３．１．２に述べた端末間交渉の方法を用いず，送信側と受信側で選ばれた QoS のうち低い QoS を両側で採用する単純な方法とした．

受信端末では２つのアプリケーションＡＰＬ１とＡＰＬ２がそれぞれビデオサーバーＡおよびＢから２つのビデオストリームを受信する（図１０）．アプリケーションの優先度はＡＰＬ１が 50，ＡＰＬ２が 100 とした．ＡＰＬ１は Stream1 と Stream2 を，ＡＰＬ２は Stream3 と Stream4 を受信する．Stream1, 2, 3, 4 の優先度はそれぞれ，90, 50, 90, 50 とした．各ストリームの QoS パラメータを表３に示す．ＣＰＵ利用率と帯域は，Pentium II 266MHz のマシン上でのデータをもとにスプライン関数を用いた QoS マッピング [20] を用いて算出した．

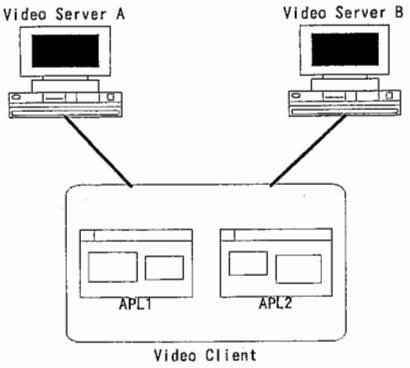


図10. システムイメージ

表3. ストリームの QoS

Candidate	Size	Frame rate	Quality	CPU(%)	Bandwidth(Kbps)
#1	640 x 480	12	90	3812.8	32555.9
#2	320 x 240	8	90	85	467.0
#3	320 x 240	8	50	78	169.5
#4	320 x 240	3	50	30	64.8
#5	160 x 120	3	50	13	33.4

図１１は受信端末のＣＰＵが，ストリーム数の変化に対してどのように確保されたかを示している．ＡＰＬ１は，開始３秒後に，サーバＡから Stream2 を，１２秒後に，サーバＢから Stream1 を受信する．その後，ＡＰＬ２は，開始２６秒後にサーバＢから Stream3 を，３６秒後には，サーバＡから Stream4 を受信する．図１１より，４ストリームを受信の際（３７～６３秒）に，優先度の高いアプリケーションの２ストリームが優先されており，ユーザ要求を反映してＣＰＵが確保されていることがわかる．しかし，２６秒後にサーバＢから Stream3 を受けた際，CPU に余裕があるにもかかわらず，Stream3 の品質が低くなってしまっている．これは，今回用いた端末間交渉では両側の QoS のうちで低い方の QoS が選ばれる上に，各端末間交渉がそれぞれのメディアごとに独立でおこなわれているため，端末間交渉における端末全体としてのリソースの利用効率を考慮した仕組みがないためである．

３．２　QoS 調整

３．２．１　QoS 調整の動作方式

ＳＡは，ＡＡから受け取った QoS パラメータセットをもとに QoS 調整を行う．ストリーム内の QoS 調整では，各 QoS パラメータの優先度とリソース状態に応じて調整を行う QoS パラメータの順番を決定し，QoS パラメータセットで指定されている QoS の範囲内で，QoS を変化させる．各 QoS パラメータの最大値と最小値の設定法によって SA の動作は変化する．つまり，最大値と最小値の範囲が広い

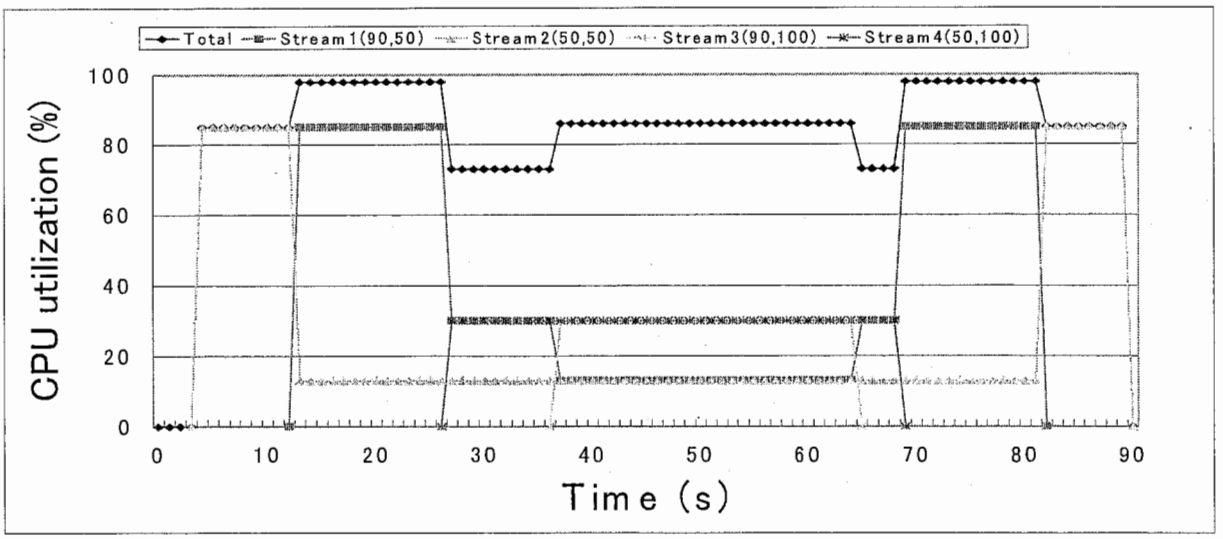


図11. 受信端末でのCPU利用状況

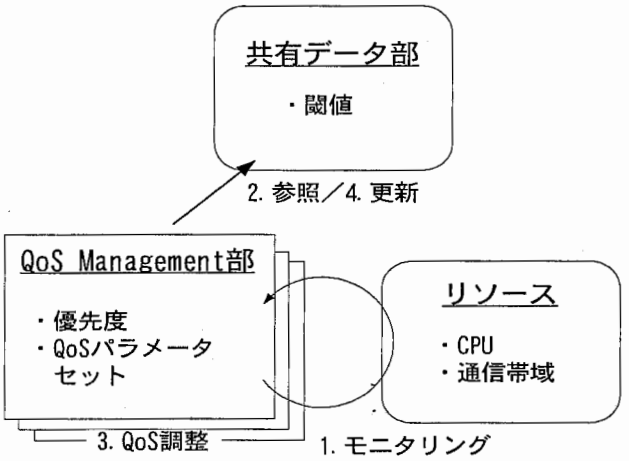


図12. SAによるQoS調整方式

程，SAの挙動はベストエフォート的な性質を持つ．逆に，最大値と最小値の範囲が狭い場合は，保証型の性質を持つが，QoS再交渉の頻度は高くなる恐れがある．最大値と最小値は，ユーザ要求に基づいて，PAによって決定され，設定される． 次に，複数のＳＡによって自律分散的にQoS調整を行うための方式を示す．各ＳＡは以下の特徴を持つ．（図１２）

- リソースモニタリングにより，システムのリソース状態を監視する．
- 各SA は，ストリーム間のQoS調整順序の指標となる優先度を持つ．
- 各SAが非同期に動作するための共有データ部を持つ．

共有データ部には全SAから参照および変更可能な可変パラメータである閾値が設定される．閾値はSAの持つ優先度との比較に利用され，優先度と同じ範囲の実数値をとる．ＳＡはモニタリングにより，システムのリソースに余裕がある，もしくは逆に切迫していることを感知できる．そのような変化を感知したＳＡは共有データ部にアクセスし，ＳＡが持つ優先度と共有データ部の閾値との比較により，QoS調整を行うべきかどうかの判断を行い，実際にQoS調整を行った場合は閾値を更新する．すなわち，リソースが切迫している場合，自分の優先度が閾値より低ければ，使用リソース量を減らすようにQoS調整を行い，その後，より優先度が高いSAもQoS調整に加わるように閾値を上げる．逆に，リソースに余裕がある場合は，自分の優先度が閾値より高ければ，使用リソース量を増やすようにQoS調整を行い，その後，より優先度が低いSAもQoS調整に加わるように閾値を下げる．このような動作メカニズムによって，ＳＡが非同期に動作することが可能となる．閾値の初期値は，全てのＳＡの優先度の最大（小）値とする場合や平均値とする場合等が考えられ，閾値の更新方法としては一定値を加減する方式や，あるいはQoS調整に参加していないSAの優先度の平均値へ更新する方式等が考えられる．

３．２．２　ＳＡの動作実験

リソース変動時のＳＡの非同期QoS調整動作を確認するため，シミュレーション実験を行った．各ＳＡは周期的にCPU使用率のモニタリングを行い，前節で述べた方式によりQoS調整を行う．１つのQoSパラメータ（表４）の値を最大値から最小値まで下げることを１回のQoS調整とし，ＳＡはユーザ要求に従ってQoSパラメータを変化させるが，本実験においては，圧縮クオリティ，フレームレート，画像サイズの順番にQoSを調整することとする．すべてのQoSパラメータを下げてしまったＳＡ

はQoS調整を行わずに，閾値の更新のみを行う．閾値の初期値は全てのSAの持つ優先度の最小値とし，閾値の更新は一定値（ΔTh）を加減していく方式を採用する（図１３）．QoS調整は閾値より低い優先度を持つ複数のSAが非同期に行うが，本実験においてはその動作順序は特に制御しておらず，実質的にはOSのタスク処理に依存している．

シミュレーション実験結果の一例を図１４，図１５に示す．実験では３つのＳＡを用いた．各ストリームは表４に示す共通のQoSを持つが，優先度は１０，５０，９０と異なる値を持つ．初めに全ＳＡが最大のQoSパラメータで安定に動作している状態（安定状態）で新たなＣＰＵ負荷をかけることにより，ＳＡの利用可能リソースを減らす．各ＳＡは再度安定に動作できる状態（再安定状態）に到達するまで，非同期にQoSパラメータを調整する．SAの動作の評価値としてQoS調整時間と優先度コストを用いた．QoS調整時間とは安定状態から再安定状態に至るまでに要した時間であり，単位はモニタリング周期である．例えば，モニタリング周期が100msである場合，ΔThが10，CPU負荷が６０％の時の実際にQoS調整にかかる時間は，9×100 = 900msとなる．QoS調整時間は短い方が実時間処理の観点から望ましい．モニタリングの周期を短くすることで，QoS調整時間は短くなるが，モニタリングによる負荷がCPUにかかるため，トレードオフが発生する．また，優先度コストとはＳＡがQoS調整を行う都度，そのＳＡの優先度を加算していくことにより得られる値である．この値が大きい程，優先度が高いＳＡがQoS調整を行ったことになるので，優先度コストは小さい方が望ましい．

新たなＣＰＵ負荷が１０％，３０％，６０％の場合に，ΔThとQoS調整時間の関係を図１４に，ΔThと優先度コストとの関係を図１５に示す．両図には，閾値を用いず，３つのＳＡが優先度の低い順に順番にQoS調整を行った場合（Sequential）とすべてのＳＡが並列にQoS調整を行った場合（Parallel）のデータもあわせて示してある．

図１４および図１５からΔThを大きくするとSAの並列性が増し，逆にΔThを小さくするとＳＡの動作の逐次性が増すことがわかる．すなわちΔThが大きい場合はQoS調整時間は短くなるが，反面，優先度コストが高くなりストリームの優先度が守られていない．一方ΔThが小さい場合は優先度コストは低く抑えられるが，QoS調整時間が非常に長くなる．従って，実時間処理と優先度を守る処理のどちらに重点を置くかにより，ΔThを適応的に設定しSAの動作を制御する必要がある．しかしながら図１５の６０％の例が示すように，ＣＰＵに急激な負荷が加わる場合には最終的に全てのSAがQoS調整にかかわることになるため，ΔThにかかわらず優先度コストがほぼ一定となり，このような場合にはΔThを大きくしてできる限りSAの並列性を増すことが望ましい．

表4. QoSパラメータ

	最大	最小
フレームレート	5fps	3fps
サイズ	320 x 240	288 x 216
品質	80	50

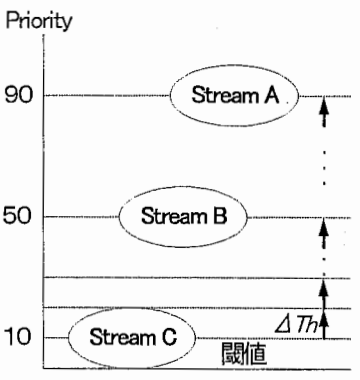


図13. 閾値の変化

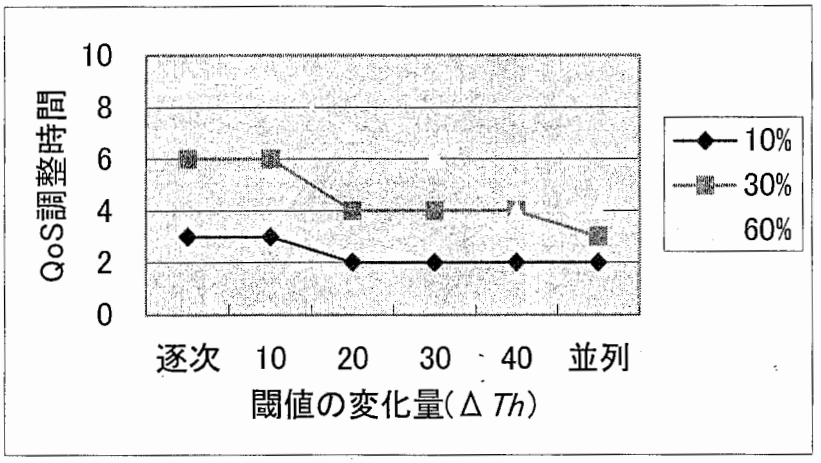


図14. QoS 調整時間

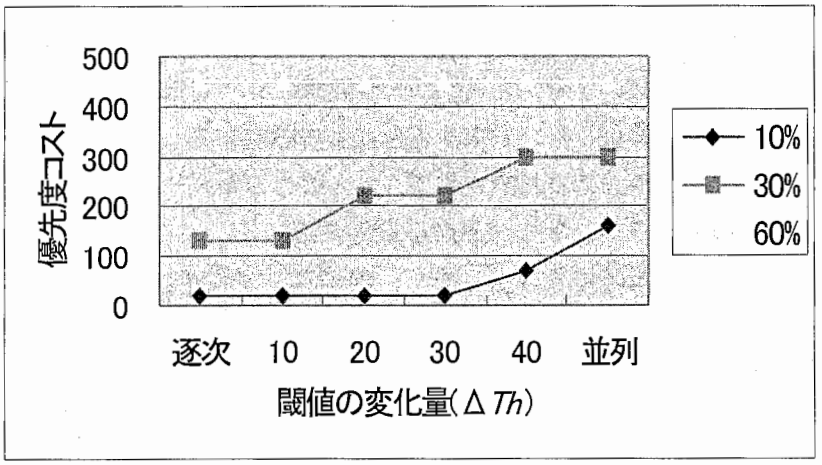


図15. 優先度コスト

4． 適応的 QoS 制御フレームワークの動作実験

4．1 システム構成

図１６に実験システムの構成を示す．各端末にはPC（OS:Windows2000）を使用し，通信にはIPを使用し，有線および無線で同じサブネットに接続されている．また，各端末にはＣＣＤカメラ，キャプチャカード，サウンドカードが接続されており，動画像及び音声のリアルタイム圧縮・伸長が可能となっている．この実験システム上に適応的QoS制御フレームワークとそれを利用する通信アプリケーションの実装を行った．今回作成した通信アプリケーションは，送信端末では，ＣＣＤカメラから取り込んだ動画像をＭ－ＪＰＥＧ方式でリアルタイム圧縮し，受信端末に向けて送信する．受信端末では，送信端末から受信したデータを伸長して別々のウィンドウで表示する．

4．2 ソフトウェア構成

図１７にフレームワークのソフトウェア構成を示す．本実験ではＡＡの動作確認を行うこととし，

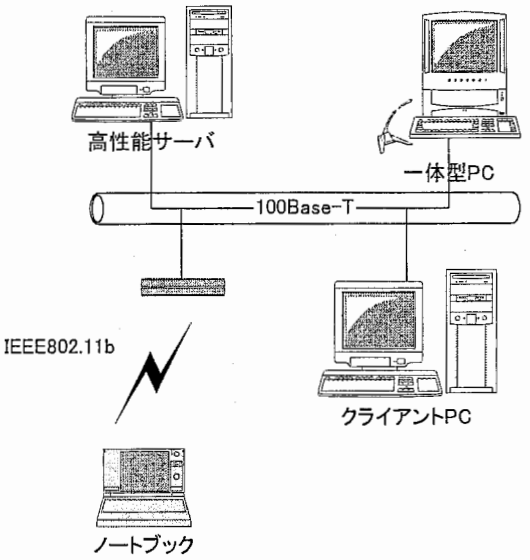


図16. システム構成

ＰＡとＳＡの実装は行っていない[1]．そのため，ユーザ要求はＰＡを用いずにQoS設定フロントエンド（図１８）を作成し，直接要求QoSを入力することにより設定する．入力されたQoSパラメータセットは名前を付けられてデータベースに格納される．ＡＡはこのQoSパラメータセット名を使って対応するQoSパラメータセットをフロントエンドから取得する．

通信アプリケーションは起動時にＡＡを生成し，終了時に消滅させている．ＡＡ生成時には，QoS設定フロントエンドが自動的に起動される．また，通信アプリケーションはフレームワークを利用する為に専用の通信ライブラリを利用する．ＡＡは，リソースマネージャからＣＰＵの使用量を知ることができ，また通信ライブラリからは単位時間の送受信データ量を知ることができる．ＳＡは，通信ライブラリのopen関数内で自動的に生成され，close関数内で自動的に消滅する．ＳＡが生成された時に，ＡＡはQoS設定フロントエンドからQoSを取得してQoS交渉を行う．また，QoS設定フロントエンドからは，ＡＡにユーザ要求が変更されたことを通知でき，通知を受けたＡＡはQoSの再交渉を行う．ＡＡにおけるQoSマッピングはスプライン関数によるQoSマッピング[20]を用い，QoSマッピング用のデータはアプリケーションを動作させてあらかじめ測定した．ＣＰＵリソースの有効利用の観点から，エージェントはイベント駆動方式をとり，再交渉要求等のエージェントへのメッセージ送信は非同期に行われる．

フレームワークでは，できるだけアプリケーション側でQoSを意識しないように考慮しているが，決定されたQoSをアプリケーションが反映できるようにアプリケーション側にQoS設定用のAPIが必要となる．また，現在は，各メディアストリームのQoSパラメータセットを接続時にアプリケーションから入力している．これについては，フレームワーク側で通信開始をハンドリングして，フレームワーク側からQoSパラメータの指定を要求する，もしくはユーザやサービスから自動に決定することも可能である．

ＡＡはJavaで実装され，エージェント間はRMIによって通信を行う．交渉のメッセージフォーマットはKQMLを使用したが、プリミティブは独自に定義した。アプリケーションおよび通信ライブラ

[1] ＳＡはダミーの雛型だけ作成されるが，QoS調整は行わない．

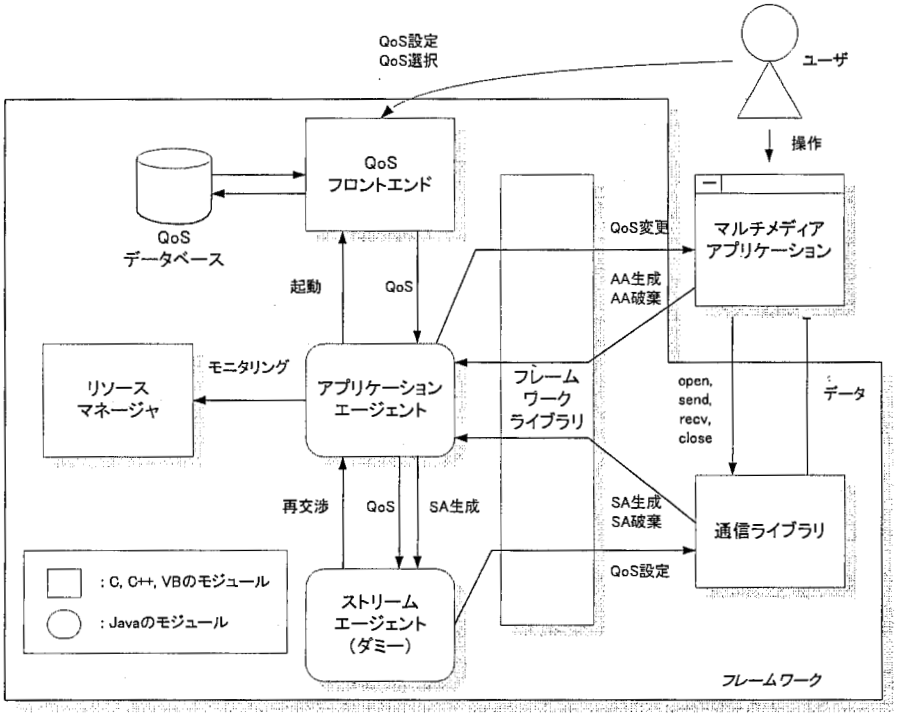


図17. ソフトウェア構成

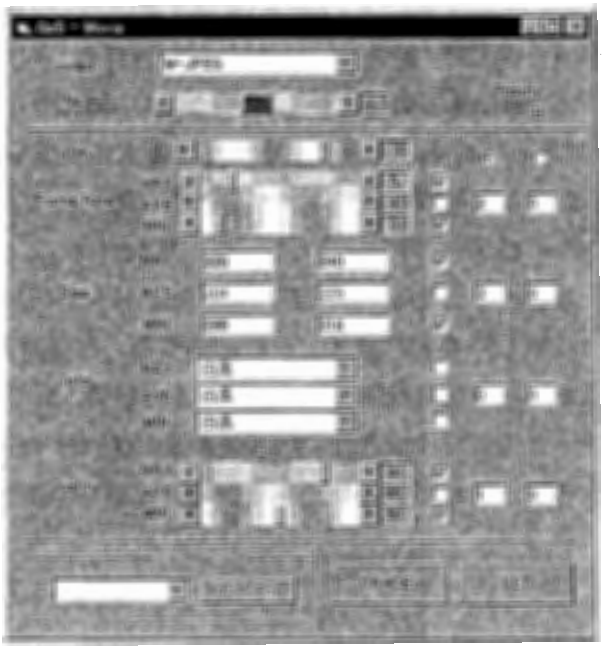


図18. QoS フロントエンド

リは C++で実装を行った．フレームワークとアプリケーションを結びつけるフレームワークライブラリ内で，JNI を用いて Java の VM 起動や Java と C++間でのメソッドの呼び出し変換を行っている．

4．3　実験システムの動作例

次に，ＡＡ間の QoS 交渉によってユーザの要求が画像調整に反映されることを確認するために，本システムを用いて行った動作実験の一例を示す．まず，図１９(a)に示すように動画像 A（画面左のウ

(a)　　(b)

図19. 動作実験例

ィンドウに表示）を小さいサイズで表示し，動画像B（画面右のウィンドウに表示）を大きいサイズで表示している状況から開始する．ここで，動画像AのQoS設定画面から，動画像Aの画面を大きいサイズで表示するようにQoSパラメータをセットし，そのユーティリティを大きい値に設定する．このとき，受信端末では，設定したQoSパラメータが，動画像AのＡＡに通知され，必要リソース量の値に変換される．今回の設定では画像のサイズが大きくなったので処理量が増えCPU使用率が増加する．そこで，動画像AのＡＡと動画像BのＡＡ間の端末内QoS交渉によって動画像Bをこのままのサイズで表示することは不可能と判断され，CPU使用率を100%以内に収めるために動画像Bを小さいサイズで表示するようにリソース(CPU 使用率)の再配分がなされる．（この実験例では，両エージェントの交渉が成立するように，動画像BのQoSについては小さい画像サイズのユーティリティも大きな値に設定している.）受信端末でのQoS交渉の結果，受信端末の動画像AのＡＡと送信端末AのＡＡ，受信端末の動画像BのＡＡと送信端末BのＡＡがそれぞれ個別に端末間QoS交渉を行い，受信端末から送信端末に新しいQoSパラメータ(フレームレート，画像サイズ，圧縮クオリティ)が通知される．本実験システムでは，送信端末と受信端末の間にフィルタリング機能が存在しないので，受信品質を落す場合も，送信端末に新しいQoSパラメータを通知する．送信端末では新たに設定されたQoSパラメータに従って動画像を圧縮し受信端末へ送信する．その結果，図１９(b)に示すように動画像Aはユーザの要求どおり大きいサイズで表示され，動画像Bは小さいサイズの表示に変わる．このように，ＡＡ間のQoS交渉によって，ユーザの要求が画像調整に反映されることを確認した．

5．　まとめ

本文では，マルチエージェントの直接的あるいは間接的な連携機能を利用した適応的QoS制御フレームワークを提案した．本フレームワークは，様々な通信アプリケーションのための共通基盤を提供することを目的としている．マルチエージェントシステムを，通信アプリケーション対応に設けるアプリケーションエージェント群とストリーム対応に設けるストリームエージェント群に階層化することにより，高度で知的な適応的QoS制御と実時間的な適応的QoS制御の両者を同時に実現している．実験システムおよびシミュレーションを用いて，本フレームワークの簡単な動作実験を行い，動作確認を行った．今後は，ＳＡやＰＡ，ＮＡを含めたフレームワーク全体を実験システムに組み込み，そ

の有効性を確認する必要がある．また，同時に，適応的 QoS 制御を有効活用したマルチメディア通信アプリケーションを提案していく必要もある．現在，適応的 QoS 制御における市場モデルを用いたネットワークリソースの割り当て方法[21]や複数端末を利用したマルチメディアサービス[22]の研究も行っており，それらの成果も盛り込んでいく必要がある．

参考文献

[1] K. Nahrstedt, and J.M. Smith, "Design, Implementation and Experiences with the OMEGA End-point Architercture," IEEE JSAC, September 1996.

[2] A.T. Campbell, G. Coulson, F. Garcia, D. Hutchison, and H.Leopold, "Integrated Quality of Service for Multimedia Communications," Proc.IEEE INFOCOM'93, San Francisco, USA, April 1993.

[3] C. Aurrecoechea, A.T. Campbell, and L. Hauw, "A Survey of QoS Architectures," ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol.6 No.3, May 1998

[4] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin ,"Resource reSerVation Protocol (RSVP)", RFC2205, IETF, 1997

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss,"Architecture for Differetiated Services (Diffserv)", RFC2475, IETF, 1998

[6] 中沢実, 須田飛志, 酒井宏三, 服部進実, "エンドユーザネットワーキング－ユーザアダプティブエージェントによるQoS制御," 信学技報 CQ97-6, pp.39-46, May. 1997.

[7] A.Puliafito, O.Tomarchio, and H.de Meer, "An agent-based framework for QoS management," Proceedings of WCSS'97, Singapore, pp392-396, Sept. 1997.

[8] 菅沼拓夫, 藤田茂, 菅原研次, 木下哲男, 白鳥則郎, "マルチエージェントシステムに基づくやわらかいビデオ会議システムの設計と実装," 情処学論,vol.38, no.6, pp.1214-1224, June 1997.

[9] J. Tokuda, T. Nakajima and P. Rao, "Real-Time Mach: Towards a predictable real-time system," Rroc. Of USENIX Mach Workshop, Oct. 1990.

[10] J. C. Pasquale, G. C. Polyzos, E. W. Anderson and V. P. Kompella, "the multimedia multicast channel," Proc. Of 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, pp.197-208, 1992.

[11] N. Nishio and H. Tokuda, "Simplified method for session coordination using multi-level QoS Specification and translation," Building QoS into Distributed Systems, A. Campbell and K. Nahrstedt Eds, Chapman & Hall, 1997

[12] 石田享, 片桐恭弘, 桑原和宏, "分散人工知能," コロナ社, 東京, 1996

[13] 木下哲男, 菅原研次, "～エージェントの基礎と応用～ エージェント指向コンピューティング," ソフト・リサーチ・センタ, 東京, 1995

[14] O. Angin, A. T. Campbell, M. E. Kounavis, R. R.-F. Liao, "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking," IEEE Personal Communications, pp.32-43, Aug. 1998.

[15] 中岡謙, Antoine RAUX, 小菅昌克, 松田潤, "ユーザアダプティブなQoSパラメータ調整方法の一提案", 信学技報 IN99-62(CQ99-40), pp.37-42, Oct. 1999.

[16] 荻野長生, "分散型通信アプリケーションにおけるQoS交渉の一方法," 1998信学会ソ大, B-11-11, pp.513, Sept. 1998.

[17] X. Cao, "Preference function and bargaining solutions," Proc. of the 21th IEEE Conf. on Decision and Control, WA6-10:15, pp.164-171, 1982

[18] R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control",

RFC2753, IETF, 2000
[19] 荻野長生, “市場経済モデルに基づく通信リソース割り当て”, 電子情報通信学会誌, Vol.82 No.9, pp.967-976, 1999
[20] 山﨑達也, 松田潤, ”スプライン関数を用いた QoS マッピング”, 信学総合大会, SB-11-2, pp.728-729, 1998
[21] 小菅昌克, 荻野長生, 蓮池和夫, “適応的 QoS 制御方式へのリソース制約を考慮した市場モデルの適用”, 情報処理学会第61回全国大会 講演論文集(3) 2G-3, pp.327-328, Aug. 2000.
[22] 益崎将一, 小菅昌克, 蓮池和夫, “適応的 QoS 制御方式に基づく複数端末でのマルチメディアサービスの検討”, 情報処理学会第61回全国大会 講演論文集(3) 2G-2, pp.325-326, Aug. 2000.