マルチメディア通信システムにおける効果的なQoS調整戦略の決定手法

武田 敦志, 打矢 隆弘, 北形 元, 菅沼 拓夫, 白鳥 則郎
東北大学大学院情報科学研究科/電気通信研究所

インターネットをはじめとする通信ネットワークの充実や計算機端末の高性能化により，マルチメディア通信システム (MCS) が普及してきている．動作環境に適応するMCSとして多くの研究がなされてきたが，従来の手法では，QoS調整の場面に対して効果的なQoS調整戦略を選択実行することができなかった．本稿では，事例ベース推論を用いて自動的にQoS調整戦略決定知識を獲得することにより，効果的なQoS調整戦略を決定する手法を提案する．また，本手法を適用したプロトタイプシステムについて述べ，実験結果から本手法の有効性を検証する．

The method to derive an effective QoS Control Strategy on Multimedia Communication System

Atushi TAKEDA, Takahiro UCHIYA, Gen KITAGATA,
Takuo SUGANUMA and Norio SHIRATORI
Graduate School of Information Sciences, TOHOKU Univ.
/ Research Institute of Electrical Communication

Multimedia Communication Systems (MCS) have spread, because of expansion of communication network such as internet and high efficiency computers. MCS which adapts to the operating enviromnent has been researched. However, a traditional MCS cannot derive an effective QoS Control Strategy to perform for the condition. In this paper, we propose a method to derive an effective QoS Control Strategy through getting knowledge to switch strategies by using Case-Based Reasoning automatically. And we describe a prototype system applied the proposed method, and show availability of our proposal by results of examination.

1 はじめに

インターネットをはじめとするネットワーク網の充実や計算機端末の高性能化により，動画や音声のデータを送受信するマルチメディア通信システム (以下MCSと略記) が普及してきている．MCSではリアルタイム性が重要であり，特に動画メディアにおいては高い計算能力と広い通信帯域を必要とするため，限られた計算機資源 (CPU，メモリなどの端末資源や通信帯域などのネットワーク資源) を用いて可能な限り利用者の要求を満足させるために，サービス品質 (以下QoSと略記) の調整が必要となる．

1対1のMCSにおいては，計算機資源や利用者要求に適応したQoS調整を行なうやわらかいビデオ会議システム (FVCS) などが提案されている [1]．FVCSはQoSを調整する戦略 (以下QoS調整戦略と表記) を切替える機能を持つが，QoS調整戦略の決定方法は問題発生からの時間経過のみを使った単純なものである．効果的なQoS調整戦略を決定するには，QoS調整戦略を決定するための知識 (QoS調整戦略決定知識) に，あるアプリケーションが動作しているときは通信帯域の不足が頻繁に発生する等の計算機資源状態の変化特性を反映させなくてはならない．しかし，計算機資源の変化特性はMCSが動作する環境ごとに異なるため，システム設計時に予めこの変化特性を反映させた知識を記述することは極めて困難である．

そこで本稿では，QoS調整戦略決定のための知識をシステムが自律的に獲得するために，事例ベース推論 [2] を用いた手法を提案する．本稿で提案する手法では，システムより得られる情報から用いるべきQoS調整戦略に密接に関係している情報だけを事例とし，効果的な推論を可能とする．さらに，本手法をFVCSに適用したプロトタイプシステムとしてE-FVCSを設計し，その実装指針について述べる．また，プロトタイプシステムを用いた実験を通じ，従来の環境適応型MCSと比較することにより，従来より効果的なQoS調整戦略を決定できることを示す．

以下2章で関連研究と問題点について述る．次いで，3章で効果的なQoS調整戦略を決定するための手法を提案し，4章で本手法をFVCSに適用したプロトタイプシステム (E-FVCS) について述べる．最後に，5章で提案手法の有効性を実験結果から検証し，6章で本稿のまとめと今後の課題について述べる．

2 関連研究

マルチメディア通信システムにおけるQoS調整の研究として，イベントスケジューリングを用いて端末でのプロセス割り当てを行なう研究 [3] や，市場モデルを用いてネットワーク帯域割り当てを行なう研究 [4] がある．これらの方法では，割り当て対象

のアプリケーションは安定して動作するが，マルチメディア通信システムのパケット優先度やプロセスの処理優先度が高くなることにより，計算機資源を共有している他のアプリケーションが十分に計算機資源を利用できなくなるという問題が発生する．つまり，計算機資源割り当てを行なうにあたり，他のアプリケーションがより多くの計算機資源を必要とした場合は，マルチメディア通信システムは自身のQoSを調整して計算機資源に余裕を作る必要がある．

また，QoS調整に関する研究として，利用可能な計算機資源とQoSパラメータの関係知識についての研究[5][6]や，利用者要求とQoSパラメータの関係知識についての研究[7]がある．これらの研究では，計算機資源とQoSパラメータの関係や利用者要求とQoSパラメータの関係について論じられているが，利用可能な計算機資源の時間的変化への対応については述べられていない．

利用可能な計算機資源の時間的変化を考慮した研究として，やわらかいビデオ会議システム(FVCS)の研究[1]がある．この研究で述べられているFVCSは，時間的に変化する様々な計算機資源の状態に対応するためシステム内部に複数のQoS調整戦略を持ち，これを切替えることにより種々の状況に対応する．しかし，その切替えの判断基準は計算機資源の不足などの問題発生時からの経過時間のみであり，状況に則した効果的なQoS調整戦略を的確に選択できていないという問題がある．効果的なQoS調整戦略を決定するためには，計算機資源状態の変化特性に関する知識を用意する必要があるが，この変化特性はFVCSの動作環境により多種多様であるため，システム設計時にFVCSに対して予め与えることは非常に困難である．そのため，効果的なQoS調整戦略を決定するための知識をシステム動作中に自動的に獲得する手法の確立が求められている．

3 効果的なQoS調整戦略の決定手法

3.1 QoS調整戦略決定知識の自動的な獲得

計算機資源状態の変化によりQoSを変更する場合，計算機資源状態の変化特性と相性のよいQoS調整戦略を実行すれば，効果的なQoS調整が期待できる．例えば計算機資源が一時的に不足しているが直後に計算機資源の回復が期待できる場合，QoSパラメータを変更せずに計算機資源の回復を待つようなQoS調整戦略が効果的である．ここで，効果的なQoS調整戦略を決定するためには，システムが計算機資源状態の変化特性について知っていなくてはならない．先の例の場合，“直後に計算機資源の回復が期待できる”という計算機資源状態の変化特性を知っていれば，“QoSパラメータを変更しない”という効果的なQoS調整戦略を決定できる．

しかし，計算機資源状態の変化特性はシステムが動作する環境により多種多様であり，この特性を反映させた知識をシステム設計時に予め与えることは非常に困難である．よって本稿では，事例ベース推論[2]を用い，計算機資源状態の状態特性を考慮しつつ効果的なQoS調整戦略を決定する知識を自動的に獲得することにより，効果的なQoS調整戦略を決定するための手法を提案する．

事例ベース推論は過去の経験を事例として蓄積し，その事例を用いて未知の問題を推論する手法である．事例は条件と結論の対となっており，条件となっている情報と結論の関係が密であるかどうかが推論結果に大きく影響する．つまり，ただ単純にすべての情報を事例とするだけでは正答率の高い推論結果は期待できず，結論と関係のある情報だけを抜き出して事例とする処理が必要である．また，その事例の収集方法も重要である．

3.2 QoS調整戦略決定知識の内容

効果的なQoS調整を行なうためには，計算機資源の変化特性を反映させたQoS調整戦略を決定するための知識を用意しなくてはならない．そこで，次式で表されるQoS調整戦略決定知識Kを定義する．

$$S(t) = K(R(t), I_{mcs}(t), D(t)) \quad (1)$$

$R(t)$,$D(t)$は時刻tにおける計算機資源の状態と利用者要求であり，$I_{mcs}(t)$はマルチメディア通信システムの状態である．$S(t)$は時刻tに選ばれるQoS調整戦略であり，QoS調整戦略決定知識Kに求められる機能は，$R(t)$，$I_{mcs}(t)$，$D(t)$が与えられたときに最も効果的と思われるQoS調整戦略$S(t)$を導出することである．

計算機資源状態の変化はその計算機資源を利用する複数のアプリケーションによって生じる．すなわち，計算機資源状態の変化特性は，その計算機資源を利用している複数アプリケーションの動作特性によって決定される．ここで，n個のアプリケーションが利用する計算機資源の状態$R(t)$を次式で表す．

$$R(t) = f(I_{app_1}(t), I_{app_2}(t), \cdots, I_{app_n}(t)) \quad (2)$$

$I_{app_i}(t)$とは，時刻tにおけるapp_iの動作状態である．

一般にそれぞれのアプリケーション動作の状態変化には固有の特徴がある．例えば，テキストエディタやSSHアプリケーション[8]のように瞬間的に計算機資源を利用するものや，マルチメディアアプリケーションのように一定量を継続して利用するもの，そして，コンパイラやシミュレータなどのようにある計算が終了するまで計算機資源を出来る限り使おうとするものがある．個々のアプリケーション毎に計算機資源利用状態を解析することは難しくないが，これらアプリケーションが計算機資源を共有する場合，共有された計算機資源の状態変化は複雑になる．

ここで，ある時刻$t = t_0$に計算機資源状態に変化が発生したとする．個々のアプリケーションの計算機資源利用量の変化は頻繁には発生しないものとすると，時刻$t = t_0$において計算機資源状態の変化に影響を与えるアプリケーションは少数であり，多くの場合，あるひとつのアプリケーションの計算機資源利用量の変化がシステム全体の計算機資源状態の変化に対して支配的となる．すなわち，式2はある時刻$t = t_0$において次式に近似できる．

$$R(t) = f(I_{app_m}(t), R_{total}(t)) \quad (3)$$

$I_{app_m}(t)$は，$t = t_0$において計算機資源$R(t)$の状態変化に対して最も影響を与えるアプリケーション

$Cases(app_m) := \{case(app_m)\}*$
$Case(app_m) := (S, I_{app_m}, I_{mcs}, R_{total}, D)$

S	:=	<QoS Control Strategy>
I_{app_m}	:=	(R_{app_m}, C_{app_m})
I_{mcs}	:=	(R_{mcs}, C_{mcs})
D	:=	<User Demand>
R_{total}	:=	< Resource Usage (all) >
R_{app_m}	:=	< Resource Usage (app_m) >
C_{app_m}	:=	< Condition (app_m) >
R_{mcs}	:=	< Resource Usage (MCS) >
C_{mcs}	:=	< Condition (MCS) >

MCS : Multimedia Communication System
app_m : The most dominant application to resources

図 1: QoS 調整戦略決定知識獲得のための事例に含む情報

Strategy (agent name)	=	Agent1a
MostDominantAppA	=	App1
DominantAppA(1-sec-average)	=	0.32
DominantAppA(2-sec-average)	=	0.02
DominantAppA(4-sec-average)	=	0.00
...		...
MostDominantAppB	=	App5
DominantAppB(1-sec-average)	=	0.42
...		...
MCS(1-sec-average)	=	0.20
...		...
Size	=	320x240
Fps	=	15.0
...		...
Total(1-sec-average)	=	0.98
...		...
Problem	=	too high

図 2: CPU 資源を対象とした QoS 調整戦略決定知識獲得のための事例

の動作状態であり，$R_{total}(t)$ はすべてのアプリケーションを統合して見た場合の計算機資源利用状態である．効果的な QoS 調整戦略を実行するためには，$I_{app_m}(t)$，$R_{total}(t)$，$I_{mcs}(t)$,$D(t)$ から効果的な QoS 調整戦略 $S(t)$ を導出する知識が必要となる．

3.3 QoS 調整戦略決定知識獲得のための事例

図 1 に提案手法で用いる事例データの内容を示す．また，図 2 に CPU 資源を対象とした事例の具体例を示す．提案手法は図 1 で示した情報を事例データとして収集蓄積する．I_{app_m} の情報には，それぞれのアプリケーションの計算機資源の利用状態だけでなく，アプリケーションの動作情報 C_{app_m} も含む．アプリケーションの動作情報とは，アプリケーションに対する利用者要求の情報やアプリケーションが提供している QoS の情報などを指す．

この事例を計算機資源の状態変化に最も影響を与えたアプリケーション app_m ごとに分けて蓄積する．これは app_m が異なると事例に含まれる情報の意味も異なるためである．

3.4 QoS 調整戦略決定知識のための事例蓄積

図 3 に従来の FVCS の動作手順と，FVCS に提案手法を適応した場合の動作手順を示す．まず従来の FVCS の動作手順は以下の通りである．

(1) 計算機資源の利用状態が利用者要求を満たしていないことを発見する．
(2) 採用する QoS 調整戦略 $S(t)$ を決定する．
(3) QoS 調整戦略 $S(t)$ に従い QoS を変更する．
(4) QoS 変更により利用者要求を満たすことが出来ていれば QoS 調整を終了し，利用者要求が依然として満たされていない場合は手順 (1) に戻る．

次に，従来の FVCS に提案手法を適応した場合の動作手順を示す．提案手法では FVCS の手順 (1)，および，手順 (4) を図 3 のように拡張する．手順 (1-1) では QoS 調整戦略を決定するための推論を行ない，手順 (4-1)(4-2)(4-3) では事例の蓄積を行なう．

(1-1) それぞれのアプリケーションの動作状態情報 $I_{app_m}(t)$，$I_{mcs}(t)$，$R_{total}(t)$，および利用者要求の情報 $D(t)$ を収集する．$I_{app_m}(t)$，$I_{mcs}(t)$，$R_{total}(t)$，$D(t)$ を用いて事例ベース推論を行ない，最も効果的と思われる QoS 調整戦略を導出する．

(4-1) QoS 変更により利用者要求を満たすことが出来ていれば，収集する事例を $Case(t) = \{S(t), I_{app_m}(t), I_{mcs}(t), R_{total}(t), D(t)\}$ とする．

(4-2) 利用者要求が依然として満たされていない場合は，情報 $I(t+1)$，$D(t+1)$ から，時刻 t において用いるべきであった QoS 調整戦略 $S_{better}(t)$ を導出する．導出方法は以下の通りである．まず，ある QoS 調整戦略 S と情報 $I_{app_m}(t)$，$R_{total}(t)$，$I_{mcs}(t)$，$D(t)$ に従って QoS を変更したときに予想されるマルチメディア通信システムの状態 $I_{pre-mcs}(t+1)$，および，予想されるシステム全体での計算機資源利用状態 $R_{pre-total}(t+1)$ を導出する．そして，この導出処理をシステムに登録されている 全ての QoS 調整戦略に対して実行し，最も利用者要求を満足させた $\{I_{app_m}(t+1), I_{pre-mcs}(t+1), R_{pre-total}(t+1), D(t+1)\}$ を導出した S を $S_{better}(t)$ とする．収集する事例は $Case(t) = \{S_{better}(t), I_{app_m}(t), I_{mcs}(t), R_{total}(t), D(t)\}$ とする．

(4-3) 事例 $Case(t)$ をアプリケーション app_m で振り分けて蓄積する．

提案手法では，以上の手順により QoS 調整戦略決定に関する成功事例を収集する．この事例を用いて事例ベース推論を行なうことにより，効率的な QoS 調整戦略を決定するための知識を獲得する．

4 プロトタイプシステムの設計と実装

4.1 Effective FVCS(E-FVCS) の設計

3 章で提案した手法を従来の FVCS[1] に適用したプロトタイプシステムとして，E-FVCS を設計する．図 4 に E-FVCS のシステム構成を示す．図 4 において，<Knowledge> はデータベース等の知識蓄積モジュールであり，<Agent> はエージェントである．ここで，エージェントとは，他のソフトウェアモジュールとの関係が疎であり，システム設計後に追加交換可能なソフトウェアモジュールと定義する．

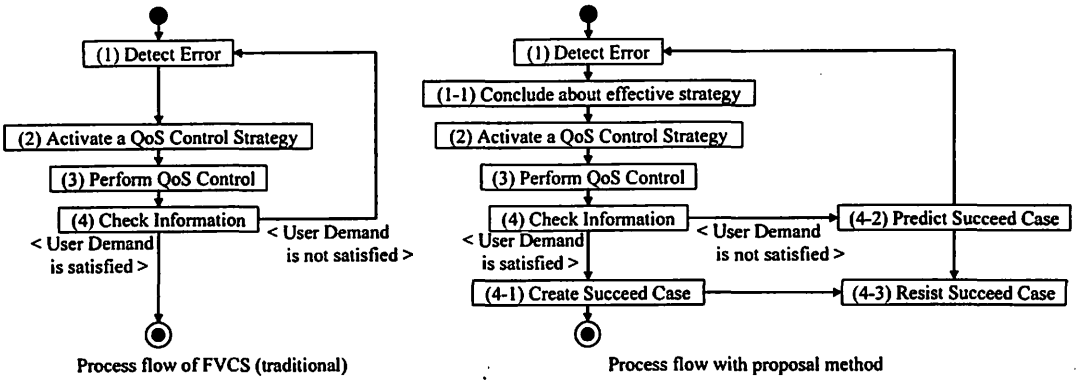

図 3: 従来手法 (FVCS) と提案手法における QoS 調整の処理手順

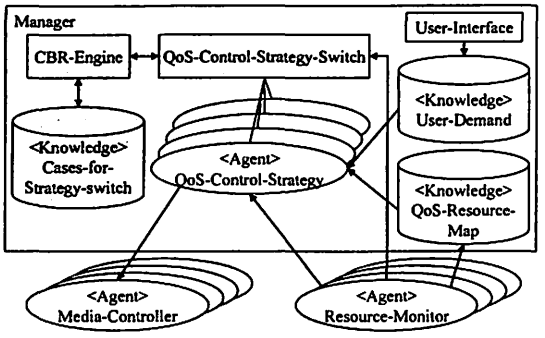

図 4: E-FVCS のシステム構成

Agent name	QoS Control Strategy
Agent1a	$q(t+1) = k(r(t))$
Agent1b	$q(t+1) = k(\frac{1}{2}r(t) + \frac{1}{2}r(t-1))$
Agent2a	$q(t+1) = k(\frac{1}{3}\sum_{t-2}^{t} r(k))$
Agent2b	$q(t+1) = k(\frac{1}{7}\sum_{t-6}^{t} r(k))$
Agent2c	$q(t+1) = k(\frac{1}{15}\sum_{t-14}^{t} r(k))$
Agent3a	$q(t+1) = q(t)$
Agent3b	$q(t+1) = \frac{1}{2}q(t) + \frac{1}{2}k(r(t))$
Agent3c	if $r(t)$ is too high $\Rightarrow q(t+1) = \min$

$r(t)$:< CPU Usage of applications without E-FVCS >
$q(t)$:< QoS Parameter Value >
$k(r)$:< CPU Resource to QoS Conversion >

図 5: プロトタイプシステムに登録した QoS 調整戦略

従来の FVCS に新たに追加した構成要素は，以下に述べる CBR-Engine，Cases-for-Strategy-Switch，QoS-Control-Strategy-Switch の 3 つである．

QoS-Control-Strategy-Switch QoS 調整戦略を切替えるモジュール．CBR-Engine で推論した結果を元に，そのとき最も効果的と思われる QoS-Control-Strategy を活性化・実行する．また，Resource-Monitor からの情報をもとに事例を生成し，Cases-for-Strategy-Switch に登録する．

Cases-for-Strategy-Switch,CBR-Engine QoS 調整戦略決定のための事例を蓄積し，その事例を使って事例ベース推論を行なうことにより，その時々で最も効果的と思われる QoS 調整戦略を推論する．

また，FVCS とは異なり，QoS-Control-Strategy をエージェントとして設計したことにより，E-FVCS システム設計後に追加交換が可能である．E-FVCS は新しく QoS-Control-Strategy を追加していくことで，QoS 調整戦略を拡充することができる．

4.2 プロトタイプ実装

提案手法の実現性を検証するために，プロトタイプシステムとして E-FVCS を Java，および，C++ を用いて実装した．また，事例ベースの推論エンジンには C4.5[9] を使用した．E-FVCS で監視・適応する計算機資源は CPU 資源を対象とした．また，通信するメディアは動画メディアとし，調整する QoS パラメータは動画像のサイズとフレームレートとした．

図 5 にプロトタイプシステムに登録した 8 個の QoS 調整戦略を示す．これらは，CPU 利用率の変化を種々の方針を元に予測し QoS を変更するものと，CPU 利用率の変化には関係なく QoS を変更するものである．

事例ベース推論に用いた事例の例を図 2 に示す．全体の CPU 利用率の変化に対して (A)CPU 利用率が最も変動したアプリケーション，(B)CPU 利用率が最も大きいアプリケーションの 2 種類を支配的であるアプリケーションとした．また計算機資源状態の変化の履歴を表す i 秒間平均 i-sec-average は，CPU 利用率を $r(t)$ とした場合，i-sec-average $= \frac{1}{i}\sum_{t-2i+2}^{t-i+1} r(k)$ とした．

QoS 設定値とその時使用する計算機資源との関係についての知識蓄積方法については様々な研究 [5][6] がなされているが，プロトタイプシステムでは過去の QoS と CPU 利用状態の値を収集し平均値を用いる単純な方法を採用した．

5 実験と評価

5.1 比較対象のシステム

E-FVCS のプロトタイプシステムを用いた実験を行ない，その結果を評価する．本稿では，QoS 調整戦略切替え機能を持たない適応型 MCS を「適応型 MCS」と呼び，QoS 調整戦略切替え機能は持つがその決定知識が静的に記述されている適応型 MCS を

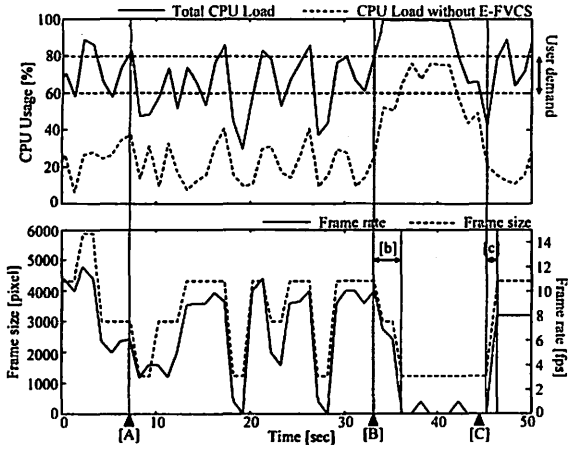

図 6: 一般的な適応型 MCS による QoS 調整

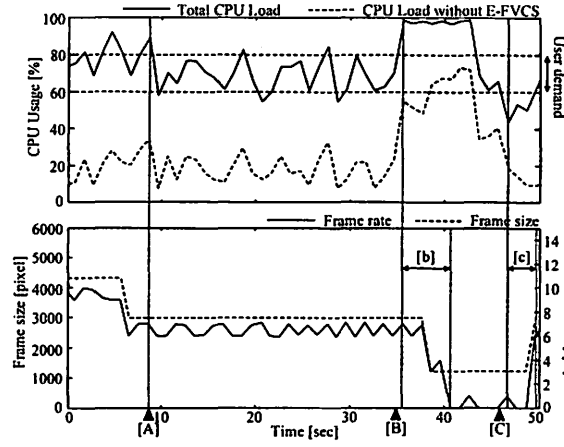

図 7: 従来の FVCS による QoS 調整

FVCS と呼ぶことにする．以下，従来の適応型 MCS や FVCS と E-FVCS の比較を行ない，E-FVCS が効果的な QoS 調整戦略を選択実行できることを検証する．

適応型 MCS で用いる QoS 調整戦略は E-FVCS に登録されている QoS 調整戦略 (図 5) の中で最も汎用的な Agent1a とする．また FVCS では，問題が発生すると最初に Agent3a を実行し，問題が解決出来なければ続いて Agent3b，Agent1a の順で QoS 調整戦略を実行こととした．

5.2 実験環境

実験では 2 つの計算機端末 Computer-A，および，Computer-B が 100Mbps のネットワークで接続された環境で，それぞれの端末において E-FVCS を動作させて行なった．Computer-A は CPU として Pentium2.4，メモリとして DDRRAM-512MB を搭載し，Computer-B は CPU として Pentium2.8C，メモリとして DDRRAM-1GB を搭載している．それぞれの端末には USB カメラを接続し，そのカメラから取得された動画像をお互いに送信する．

上述の実験環境において，Computer-A の CPU に負荷をかけ，Computer-A の CPU 利用率とそこで動作する E-FVCS の QoS 調整の動作を観測する．Computer-A の CPU にかける負荷は，15 個のアプリケーションを実際に使用したときの CPU 負荷をエミュレートしたものである．

本実験における利用者要求は “計算機端末全体での CPU 利用率が 60%以上 80%以下” とした．また，図 4 で示した User-Demand と QoS-Resource-Map の知識は予め用意し，適応型 MCS，FVCS，E-FVCS 共に同じ知識を用いた．

5.3 実験結果

5.3.1 従来手法 (適応型 MCS，FVCS) と提案手法 (E-FVCS) との QoS 調整動作の比較

図 6 に適応型 MCS，図 7 に従来の FVCS，そして図 8 に E-FVCS(提案手法) の QoS 調整動作の結果を示す．それぞれ，時刻 [A] において断続的に CPU を利用するアプリケーションが動作を開始し，時刻 [B] において CPU 資源を最大限に使うアプリケーションが動作を開始している．

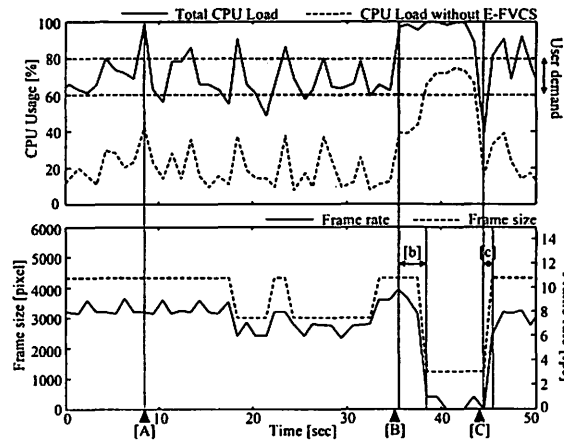

図 8: E-FVCS(提案手法) による QoS 調整

利用者要求により，それぞれのシステムは CPU 利用率が 80%より高くなれば QoS を下げ，60%より低くなれば QoS を上げようとする．従来の適応型 MCS では図 6 の時刻 [A] 以降のように，断続的に CPU を利用するアプリケーションが動いている場合に，QoS が乱高下する現象が発生する．この問題を解決するために従来の FVCS では，最初に QoS を変更しにくい QoS 調整戦略を用いているが，図 7 の時刻 [B] や [C] のように素早い QoS 変更が求められる場合に調整が遅れてしまう問題が発生する．

対して E-FVCS では，図 8 の時刻 [A] 以降は QoS を変更しにくい QoS 調整戦略を用い，時刻 [B]，および [C] では素早く QoS を変更する QoS 調整戦略を用いることにより，時刻 [A]-[B] において適応型 MCS より安定した動作を，時刻 [B][C] において従来の FVCS より素早い QoS 調整を実現している．

5.4 評価

5.4.1 E-FVCS の知識獲得効果

図 9 に E-FVCS の知識獲得時間と QoS 調整の的確さ関係を示す．ここで，横軸は事例を収集した時間を示し，縦軸は 1 分間あたりの QoS 調整失敗数である．QoS 調整失敗とは，図 3 で示した提案手法を適用した場合の動作手順 (4) において利用者要求

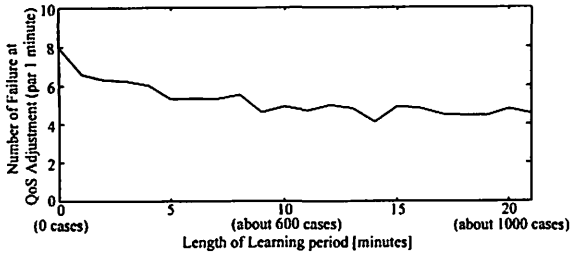

図 9: E-FVCS における知識獲得時間の長さと QoS 調整の的確さの関係

	Number of Failure at QoS Adjustment (par 1 minute)
E-FVCS (proporsal)	**4.53**
FVCS (traditional)	5.38
Adaptive-MCS (traditional)	5.46

図 10: E-FVCS(提案手法) と従来手法のとの QOS 調整の的確さについての比較

を満足できない場合を指す．的確な QoS 調整が出来ていれば，単位時間の QoS 調整失敗数は減少する．図 9 のデータは 50 回測定したときの平均値を示している．

図 9 より，E-FVCS は知識獲得時間が長くなるにつれて知識獲得を行ない，より的確に QoS 調整を実行していることがわかる．また，図 9 のグラフは 10 分を過ぎた辺りから勾配がなだらかになっており，これは E-FVCS が 10 分程度で十分な知識獲得を行なっていることを示している．

5.4.2 E-FVCS と適応型 MCS，FVCS の適応能力

図 10 に，適応型 MCS，FVCS，および E-FVCS をそれぞれ 50 分間動作させたときの 1 分間あたりの QoS 調整失敗数を示す．図 10 より，E-FVCS は，従来の適応型 MCS や FVCS よりも的確な QoS 調整を行なっていることがわかる．

5.4.3 議論

5.4.1 より，事例を蓄えていくことにより QoS 調整戦略の決定がより的確になることが示された．これらの結果は本研究の目的である効果的な QoS 調整戦略を決定するための知識の自動的な獲得を実現していること示している．

また 5.4.2 から，従来の FVCS と適応型 MCS による QoS 調整の的確さに大きな差がないことがわかった．これは，従来の FVCS が複数の QoS 調整戦略を持っているにも関わらず，それらを効果的に切替えることが出来ていないことを示している．これに対して，提案手法のプロトタイプシステムである E-FVCS は従来の FVCS や適応型 MCS よりも的確な QoS 調整を行なっている．これは，QoS 調整戦略の切替えが従来の FVCS より効果的に行なわれていることを示している．

6 おわりに

本稿ではマルチメディア通信システムにおいて効果的な QoS 調整戦略を決定することを目的とし，QoS 調整戦略決定知識を事例ベース推論を用いて獲得するために，事例の振り分け方法と事例に登録される情報，および，事例の収集方法について提案した．また，提案手法を導入した E-FVCS を用いて実験を行ない，実験結果より E-FVCS が効果的な QoS 調整戦略を決定するための知識を蓄積できることを確認し，蓄積後には従来のシステムより効果的な QoS 調整戦略が選択実行されていることを検証した．

今後の課題として，事例を蓄えるときに用いる QoS 調整の結果を評価する方法を改良することが挙げられる．例えば，QoS を変化させない QoS 調整戦略を用いて QoS 調整が失敗した場合と，QoS を大きく変化させる QoS 調整戦略を用いて QoS 調整が失敗した場合では，後者の方が利用者に与える不快感は大きい．また，計算機資源が余っている状態よりも，計算機資源が不足している状態の方が利用者にとっては不快である．本稿ではこれらを同じ失敗として評価したが，利用者の観点から見た場合，上記のような QoS 調整失敗時のリスクを考慮にいれ，QoS 調整の結果を評価する必要がある．

参考文献

[1] Takuo SUGANUMA, SungDoke LEE, Tetsuo KINOSHITA, and Norio SHIRATORI. An agent architecture for strategy-centric adaptive qos control in flexible videoconference system. *New Generation Computing*, Vol. 19, pp. 173–191, 2001.

[2] J.L. Kolodner, R.L. Simpson, and K. Sycara. A process model of case-based reasoning in problem solving. *IJCAI-85*, pp. 284–290, 1885.

[3] 阿部睦, 鳩野淳子, 渥美幸雄. 端末におけるサービス品質保証方式の提案と評価. 電子情報通信学会論文誌 B, Vol. J82-B No.5, pp. 711–721, 1999.

[4] 八槇博史, 山内裕, 石田亨. 市場モデルによるアプリケーション qos の制御:実装上のトレードオフ. 情報処理学会論文誌, Vol. 40 No.1, pp. 142–149, 1999.

[5] 小菅昌克, 山崎達也, 荻野長生, 松田潤. マルチエージェントによる適応的 qos 制御方式. 電子情報通信学会論文誌 B, Vol. J82-B No.5, pp. 702–710, 1999.

[6] 中岡謙, 松田潤. ファジークラシファイアシステムによる qos マッピングルールの獲得. 電子情報通信学会論文誌 D-1, Vol. J85-D-1 No.1, pp. 69–78, 2002.

[7] 中沢実, 須田飛志, 阿部倫之, 服部進実. ユーザアダプティブエージェントによる自律的ネットワーク. 電子情報通信学会論文誌 B-I, Vol. J81-B-I No.2, pp. 58–69, 1998.

[8] Open SSH `http://www.openssh.com/`.

[9] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.