知能と複雑系　129－9
(2002. 7. 26)

多地点間ビデオ会議システムのためのプロトコルとエージェントによる実装

前村　貴秀† 藤田　茂† 菅原　研次††

† 千葉工業大学 情報工学科
†† 千葉工業大学 情報ネットワーク学科

あらまし　我々がこれまでに開発したシステムでは，分散処理システムを動的に実現するための構成プロトコルがただ1つであった．本研究では，複数のプロトコルを実現するためにルールによる実装を行った．このプロトコルによる構成が実現されるまでの時間をPC（Pentimun3 1.13GHz RAM 1024MB JDK 1.4.0）上で計測し，200個のエージェントで，13秒以内に終了するという結果が得られた．また，3地点を結んだビデオ会議におけるエージェントを用いた支援システムの設計と実装を行った．実験の結果，VICそのものと比較してCPUやネットワークに負荷がある状況下でも，最大で2割、CPUのロードアベレージが改善される結果が得られた．

キーワード　TAF，マルチエージェントシステム，やわらかいシステム

A Case Study of the Agent-based Multi Point Video Meeting System Implementation

Takahide MAEMURA†, Shigeru FUJITA†, and Kenji SUGAWARA††

† Departmentt of Computer Science, Chiba Institute of Technology
†† Departmentt of Network Science, Chiba Institute of Technology

Abstract　The organization composition protocol based on a contract network protocol was realized by processing by the rule. Time until composition by this protocol is realized was measured, and when it was about 200 agents, the result of having ended within 12 seconds was obtained. he result of about twenty percent improved was obtained at the maximum also under the situation that load is given as compared with the case where this system is not operated, as a result of experiment which performed the design and mounting of the support system using the agent in a video meeting between multi point.

Key words　TAF, Multi Agent System, Flexible System

1. はじめに

我々がこれまでに開発したシステムでは，分散処理システムを組織するための構成プロトコルが固定的であった．そこで，柔軟な組織構成プロトコルを実現するための手段としてルールでによる処理でプロトコルを実装する方法がある．しかし，ルールで処理した場合，従来のシステムよりも組織が構成されるまでに必要な時間の増加することは明らかである．そのため，本研究では，ルールで処理した場合でも現実的な動作時間内に処理が終了するのかをPC上で調べることを目的とした．

まず始めに，エージェントシステムの設計・実装・運用を支援するツールであるTraining system for Agent Framework [1]を使用し，契約ネットプロトコルを基にした組織構成プロトコルをエージェントに実装し，評価を行った．次に，同じフレームワーク上でビデオ会議を支援するシステムの設計と実装を行い，有効性の検証を行った．これは，**VI**deo **C**onferencing tool（VIC）[2]を3地点間で柔軟に制御するためのシステムであり，情報を共有するためのプロトコルを設計した．また，負荷を与える実験を行い，VICそのものと比較して最大2割，CPUのロードアベレージが改善される結果が得られた．

2. エージェントフレームワーク

本研究で設計するエージェントが動作するフレームワークのモデル図を図1に示す．リポジトリは，開発者が設計したエージェントが蓄えられている場所である．利用者からの要求がくると，このリポジトリ内で契約ネットプロトコルを基にした組織構成が行われる．組織構成が実現するとワークプレース上にインスタンシエートされ，設計されたルールに従った動作を行う．動作中に利用者の要求に変化があると，新たな要求に基づいた組織の再構成が行われる．複数のリポジトリを利用した組

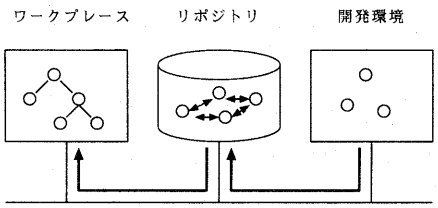

図 1 フレームワークのモデル図
Fig. 1 status changes figure

織構成も可能であり，エージェントは，ネットワークで接続された他のホスト上にあるワークプレースに移動する機能がある．さらに，エージェント名は一意となる名前が与えられるので，リポジトリに蓄えられているひとつのエージェントから複数のエージェントをインスタンシエートする機能がある．

3. 組織構成機能をもったエージェントの設計

3.1 契約ネットプロトコル

分散システムための組織構成プロトコルの基となった契約ネットプロトコルには，以下のメッセージがある．

- タスク通知（task-announcement）
  全エージェントにタスクを依頼する．
- 直接落札（directed-award）
  特定のエージェントにタスクを依頼する．
- 入札（bid）
  タスク通知に対する応答で条件を付けることも可能．
- 落札（award）
  複数の入札の中から依頼側が選んだ入札に対する応答．
- 交渉終了（break-off-negotitations）
  落札しなかった入札に対する交渉を終了するための応答．
- 動作可能（acceptance）
  初期化が終了し，動作可能状態になったことを通知する．
- 解約（cancellation）
  入札に対する応答として交渉終了を受けたとき，自分より下位のエージェントとの間で契約が成立していた場合はそれを解約するためのメッセージ．
- 拒否（refusal）
  直接落札に対して依頼が受けられないことを通知する．

次に契約ネットプロトコルの状態遷移図を図3に示す．エージェントが“タスク通知”（task-announcement）または，“直接落札”（directed-award）を受信するとタスクの評価を行う．この結果，依頼されたタスクが処理可能であれば，“入札”（bid）を返信する（直接落札の場合は，“動作可能”になる）．処理できないと判断したら，待機状態に戻る（直接落札の場合は，“拒否”（refusal）を返信する）．依頼側は入札を受信すると，入札評価を行い，最も良い条件を提示してきた“入札”に対して，“落札”（award）を返し，それ以外のエージェントには，“交渉終了”（break-off-negotitations）を返す．その後，依頼側のエージェントは acceptance 待ちの状態に入る．“落札”を受信したエージェントは，初期化を行い，“動作可能”（acceptance）を送信する．依頼側のエージェントが，“動作可能”を受信した時点で契約が成立する．

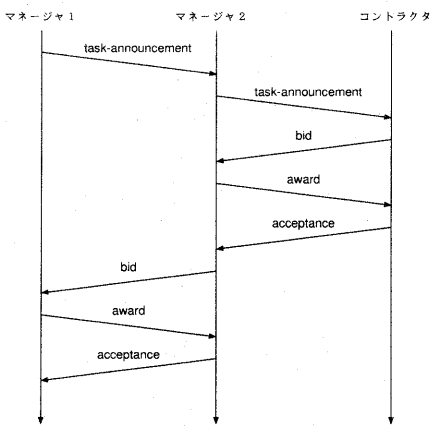

図 3 組織構成が実現された場合
Fig. 3 A flow when a contract is concluded

この契約ネットプロトコルでは，依頼されたタスクの一部を他のエージェントにさらに依頼することが可能である．タスクの一部を他のエージェントに依頼する場合，必要とする機能をもっているエージェントに関する知識を既に持っているときは，“直接落札”を使用し，特定できないときには，“タスク通知”を使用して，自分以外の全エージェントにメッセージを送信する．その後，契約が成立したとき始めて，自分に依頼してきたエージェントに対して“入札”を返す．本研究では，依頼する側をマネージャ，タスクを受ける側をコントラクタとする．深さについては，マネージャとコントラクタのみの場合を1段，コントラクタがさらにタスク分割を行い，他のエージェントに“タスク通知”を行った場合を2段と表現する．3段以降についても同様にタスク分割が行われていくものとする．

次に契約ネットプロトコルに基づいて組織構成が実現した場合の様子を図2に示す．

実験では，これらの流れの中から“タスク通知”（task-announcement）を受信してから“動作可能”（acceptance）を受信するまでを組織構成に必要な時間として計測を行った．今回，タスク分割を行った場合の時間を把握するため，あらかじめ“タスク通知”に対して“入札”を返してくるエージェントの数を調整した．これによりマネージャは，タイムアウトするまで“入札”を待たなくても落札を返せるようになり，最小の待ち時間で動作すると期待できる．これ以降は，この入札数を分岐数と呼ぶものとする．ここで，分岐数が5の場合のモデルを図4に示す．この図では，“Cnp（1）”のみが“タスク通知”を行っているが，実験では，“Cnp（0）”から“Cnp（4）”も“タスク通知”を行い，それぞれが入札を5ずつ受信する場合の計測を行うのでワークプレース上には，Cnp（0,0）から Cnp（4,4）までの25のエージェントが存在する．

3.2 組織構成に関する実験

今回設計したエージェントには，組織構成のためのルールし

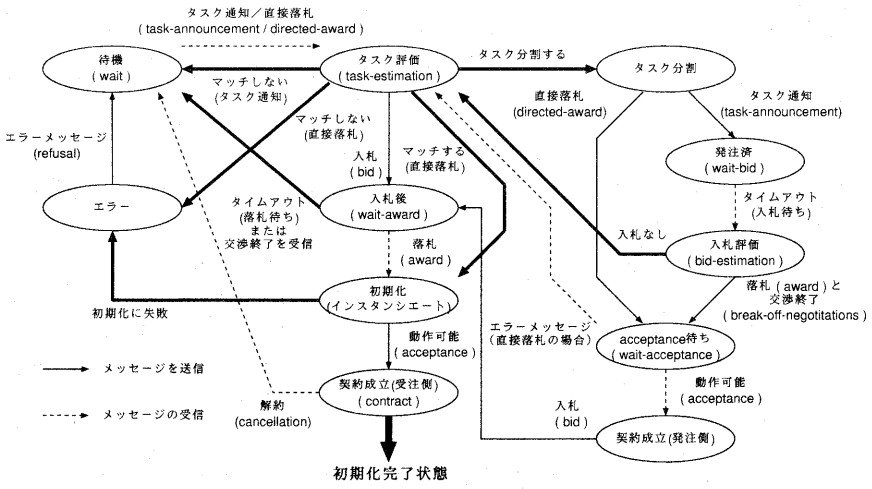

図 2　契約ネットプロトコルの状態遷移図

Fig. 2　status changes figure

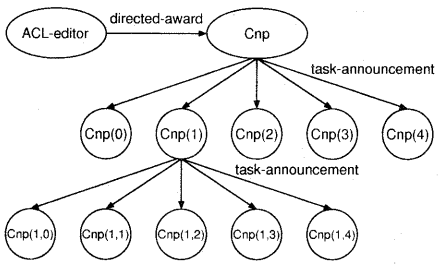

図 4　分岐数が 5 の場合

Fig. 4　When the number of branches is 5

か実装されていない．実際にエージェントを使用する場合，このルールに加えて開発者が設計したルールが追加されることになる．今回の実験条件を次に示す．グリーンスレッド，ネイティブスレッドの差は，JDK／OS の制約により生成できるエージェント数に上限があったためである．

- 組織構成プロトコルのみを実装したエージェント
  - ネイティブスレッドを使って動作させた場合
  - グリーンスレッドを使って動作させた場合
- 組織構成プロトコルのみを実装したエージェントと交渉には参加しないエージェントを同時に動作させた場合
- 組織構成プロトコルに加えてそれとは関連のないルールを実装したエージェントのみで動作させた場合

3.3　実験環境

実験環境を以下に示す．

(CPU) Pentium3 1.13GHz
(Memory) 1024MB
(OS) Linux 2.4.18
(Java) JDK 1.4.0，JDK 1.3.1

3.4　結　　果

組織構成プロトコルのみを実装したエージェントで実験を行った場合の結果を表 1，同じエージェントを用いてグリーンスレッドを使用した実験を行った場合の結果を表 2，組織構成プロトコルのみを実装したエージェントと交渉に参加しないエージェントを同時に動作させた場合の結果を表 3，組織構成プロトコルの前にそれとは関連のないルールを追加したエージェントのみで動作させた場合の結果を表 4，組織構成プロトコルの後にそれとは関連のないルールを追加したエージェントのみで動作させた場合の結果を表 5 に示す．また，これらを比較するためのグラフを図 5 に示す．これは，組織を構成するのに必要とされる時間の増加につながる要因を探るために表 1 から 5 をグラフ化したものである．

グラフの “green-threads” は，グリーンスレッドを使用した場合，“with dummy” は，エージェントを 200 個追加した場合，“forward” は，組織構成用のルールの前に別なルールを追加した場合，“backward” は，組織構成用のルールの後ろに別なルールを追加した場合にそれぞれ対応している．また，表 3 から表 5 までの括弧内の数字は，分岐数と深さが同じときの表 1 の値を 1 としたときの比であり，この値が大きいほど，組織構成が完了するまでにより多くの時間がかかっていることになる．以降ではこの値を増加率として記述する．

3.5　考　　察

表 3 から，グリーンスレッドを用いた場合，契約が成立するまでの時間が明らかに増加しているが，ネイティブスレッドを用いた場合では実現できなかった分岐数が 10，深さ 3 段の場合，つまり，エージェントの数が 1111 個の場合であっても実験を目的とする範囲ではおおむね 10 分以下で動作することがわかった．増加率については深さが 3 段までの結果だけをみると，図 5 からエージェント数の増加に併せて増加率も増えている．

また，表 3 のどのエージェントとも契約を結ばないエージェ

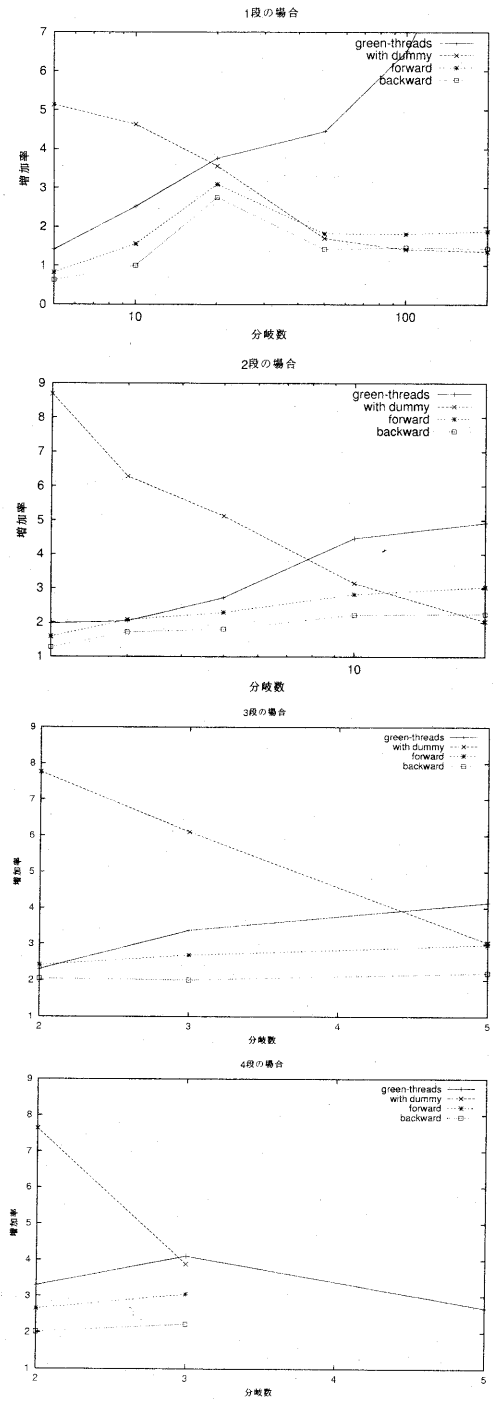

図 5　契約ネットプロトコルのみを実装したエージェントとの比較

Fig. 5　Comparison with the case of only the agent

表 1　契約ネットプロトコルのみでネイティブスレッドを用いた場合

Table 1　Contract Network Protocol with native threads

	深さ			
分岐数	1 段	2 段	3 段	4 段
2		68[7]	202[15]	502[31]
3		133[13]	547[40]	4,028[121]
5	51[6]	291[31]	4,214[156]	183,891[781]
10	54[11]	1,201[111]		
20	80[21]	8,183[421]		
50	243[51]			
100	513[101]			
200	1,168[201]			

単位はミリ秒，括弧内の数字はワークブレース上のエージェント数

表 2　契約ネットプロトコルのみでグリーンスレッドを用いた場合

Table 2　Contract Network Protocol with green threads

	深さ			
分岐数	1 段	2 段	3 段	4 段
2		134(1.97)	466(2.31)	1,654(3.29)
3		271(2.04)	1,843(3.37)	16,514(4.10)
5	72(1.41)	793(2.73)	17,453(4.14)	490,902(2.67)
10	136(2.52)	5,366(4.47)	464,282(-)	
20	301(3.76)	40,260(4.92)		
50	1,082(4.45)			
100	3,342(6.52)			
200	12,600(10.79)			

単位はミリ秒，括弧内の数字は表 1 と比較したときの増加率

表 3　どのエージェントとも契約を結ばないエージェントを 200 個，ワークブレース上に追加した場合

Table 3　Contract Network Protocol with dummy agent

	深さ			
分岐数	1 段	2 段	3 段	4 段
2		591(8.69)	1,568(7.76)	3,830(7.63)
3		835(6.27)	3,336(6.10)	15,602(3.87)
5	262(5.14)	1,490(5.12)	12,785(3.03)	
10	250(4.63)	3,777(3.14)		
20	285(3.56)	16,609(2.03)		
50	413(1.70)			
100	725(1.41)			
200	1,586(1.36)			

単位はミリ秒，括弧内の数字は表 1 と比較したときの増加率

ントを追加した場合と表 1 の追加しなかった場合とを比較すると，図 5 からもわるようにどの深さについても増加率は分岐数が増えるにつれて減少している．

表 3 のどのエージェントとも契約を結ばないエージェントを追加した場合と表 4 と表 5 のルールを追加したケースとを比較すると，前者の方が，増加率が小さいことから組織を構成するまでに必要な時間のうち，エージェント間通信に使われている時間のほうが推論のために必要とされている時間よりもより多くの時間がかかることがわかった．

ルールを追加する位置については，図 5 より，位置を変えても 1 倍以上の差は見られなかったが，組織構成用のルールより

表 4　ファクトとルールを追加した場合（契約ネットプロトコルの前）
Table 4　When a rule is added ahead

	深さ			
分岐数	1 段	2 段	3 段	4 段
2		108(1.59)	488(2.41)	1,333(2.66)
3		276(2.08)	1,469(2.69)	12,274(3.05)
5		668(2.30)	12,538(2.98)	
10	84(1.56)	3,389(2.82)		
20	247(3.09)	24,811(3.03)		
50	443(1.82)			
100	929(1.81)			
200	2194(1.88)			

単位はミリ秒，括弧内の数字は表 1 と比較したときの増加率

表 5　ファクトとルールを追加した場合（契約ネットプロトコルの後）
Table 5　When a rule is added back

	深さ			
分岐数	1 段	2 段	3 段	4 段
2		87(1.28)	413(2.04)	1,012(2.02)
3		228(1.71)	1,097(2.01)	8,947(2.22)
5		525(1.80)	9,249(2.19)	
10	54(1.00)	2,666(2.22)		
20	220(2.75)	18,421(2.25)		
50	345(1.42)			
100	749(1.46)			
200	1,681(1.44)			

単位はミリ秒，括弧内の数字は表 1 と比較したときの増加率

も前に記述するよりも後ろに記述したほうが結果が良くなることがわかった．これは，今回利用したプロダクションシステムがファーストマッチ戦略をもちいているためである．

以上のことから，今回実装した組織構成プロトコルは，200 個程度のエージェントであれば，単一のリポジトリ内で同時に動作させた場合でも，13 秒以内で組織を構成できる．

4. 多地点間ビデオ会議支援システム

本システムは，これまで 2 点間でしか行われていなかったビデオ会議システムの動的な調整を多地点を結んだ場合でも動作するようにしたもので，やらわかいビデオ会議システム [3] を拡張したシステムである．

4.1 実験環境

スイッチングハブとホストの間を 100Mbps の Ethernet で接続した 3 台のホストを使用し実験を行った．各ホストで，VIC と支援システムを動かし，あるホストに，CPU とネットワークに負荷をかけた場合における CPU のロードアベレージとネットワークのトラフィックの 2 つのパラメータ値の変動と VIC から出力されるフレームレートと画質の計測を行った．図 6 に実験で使用したハードウェアの構成を示す．

4.2 システム構成

本システムを構成するエージェントは次の通りである．

- チーフエージェント (Chief Agent)

システム全体の動作状態と接続先のホスト管理を行う．

- メッセージ送信エージェント (Sender Agent)

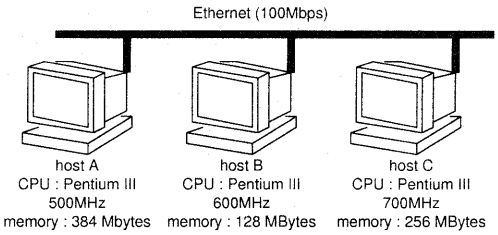

図 6　ビデオ会議システムの実験環境
Fig. 6　Experiment environment

表 6　設計したエージェントのルール数とメソッド数
Table 6　Number of rules and methods of the agent

エージェント名	ルール数	ベースプロセスのメソッド数
チーフエージェント	26	0
送信エージェント	11	8
受信エージェント	15	12
検証エージェント	9	14
計測エージェント	5	10
VIC エージェント	9	13

他のホストにメッセージを送信する．

- メッセージ受信エージェント (Receiver Agent)

他のホストからのメッセージを受信する．

- 検証エージェント (Inspection Agent)

受信したホストの情報を基にビデオ会議システムのパラメータ値を変更するかどうかを検証する．

- 計測エージェント (Measurement Agent)

主にホストの情報を収集する．

- VIC エージェント (VIC Agent)

検証エージェントからの指示を VIC に通知する．

次に，本システムで設計した各エージェントのルール数とエージェントのベースプロセスである Java のメソッド数を表 6 に示す．

4.3 結　　果

システムを動作させない状態で CPU に負荷をかけた場合の結果を図 7，システムを動作させた状態で CPU に負荷をかけた場合の結果を図 8，システムを動作させない状態でネットワークに負荷をかけた場合の結果を図 9，システムを動作させた状態でネットワークに負荷をかけた場合の結果を図 10 に示す．各グラフにおいて x 軸は経過した時間（ミリ秒），y 軸は，各パラメータ値について以下それぞれの値を 100%としたときの比率を示している．

- フレームレート（Actual Frame Rate）: 15fps
- CPU のロードアベレージ（CPU LOAD）: 5.00
- ネットワークのトラフィック（Traffic）: 5.00Mbps

まず，CPU に負荷をかけた場合については，システムを動作させた場合の結果を示した図 7 とシステムを動作させなかった場合の結果を示した図 8 とを比較すると，動作させなかった場合のロードアベレージは，250 秒から 300 秒の間，90% を

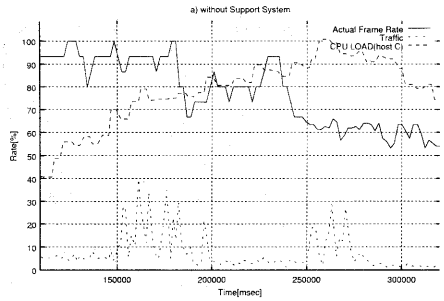

図 7 システムを動作させない状態で CPU に負荷をかけた場合
Fig. 7 The test which applies load to CPU without system

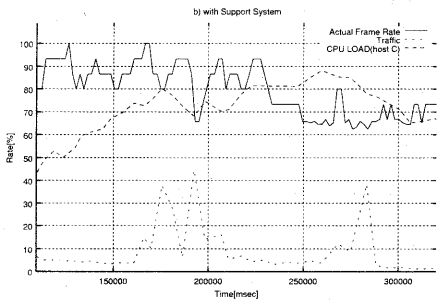

図 8 システムを動作させた状態で CPU に負荷をかけた場合
Fig. 8 The test which applies load to CPU with system

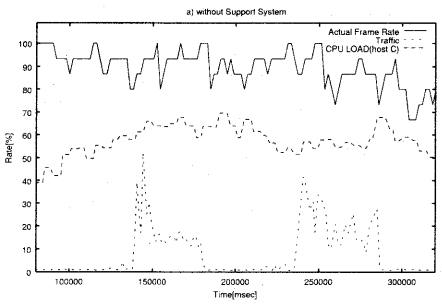

図 9 システムを動作させない状態でネットワークに負荷をかけた場合
Fig. 9 The test which applies load to a network without system

越える値が計測されたの対して動作していた場合は，最大でも 90% 程度に押えられている．

また，ネットワークに負荷をかけた場合については，システムを動作させた場合の結果を示した図 9 とシステムを動作させなかった場合の結果を示した図 10 とを比較すると，負荷をかけ始めている時点で既に CPU のロードアベレージは，動作させた場合のほうが低い値を示している．300 秒付近のロードアベレージは，共に 50%から 60%の間の示しているが，フレームレートは，動作させた場合のほうが高く，より動きの滑らかな画像が表示されていたことになる．

以上のことから，本支援システムを導入することでビデオ会議システム以外のロードアベレージやネットワークのトラフィック

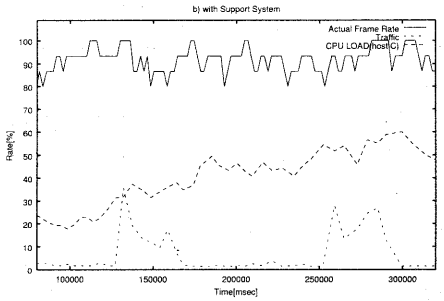

図 10 システムを動作させた状態でネットワークに負荷をかけた場合
Fig. 10 The test which applies load to a network with system

を増加させるような処理を新たに行ったとしても，ビデオ会議システム自体が利用する CPU やネットワークのリソースを減らすことによりホストの処理速度が極端に低下することを防ぐことが可能であり，利用者は，会議自体やその他の作業に集中することができる環境を実現することが可能である．

4.4 考　　察

前節では，支援システムを導入することによる有効性を確認することができた．しかしながら，このシステムは，実験環境に依存する部分が多く，次の問題点がある．

- VIC のパラメータ値を決定する条件文と閾値がハードコーディングされているのため，実験を行った環境とは状況が異なる場合，期待される結果が得られないことがある．

この問題点を解決するための方法として，条件文や閾値を動的に変更し，その環境に適した値を見つける手法が考えられ，学習を行うエージェントによるシステムとして設計中である．

5. おわりに

本稿では，ルールにより分散システムの構成／再構成を行うプロトコルを実装し，PC 上での評価実験について述べた．今回の実験から単独のエージェントリポジトリを用いて 200 個のエージェントが存在する場合に 13 秒で構成が完了することを示した．

また，同じフレームワーク上でこれまで 2 点間で行われてきた実験を 3 点間に拡張し，VIC 単独で用いるよりも CPU のロードアベレージに関して 2 割の性能向上を実現した．

文　　献

[1] 木下哲男，桑原和宏，菅沼拓夫，菅原研次，服部文夫，原英樹，藤田茂：エージェントシステムの作り方，社団法人 電子情報通信学会 (2001)

[2] Steven McCanne, Van Jacobson：Video Conferencing Tool http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/

[3] 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，情報処理学会論文誌，Vol.38，No.6，pp.1214-1224(1997).