Multiagent based Adaptive QoS Control Mechanism in Flexible Videoconference System

Sung-Doke Lee Dong-Soo Han
School of Engineering
Information and Communications University
P.O.Box 77, Yuseong, Daejeon 305-600, Korea
{sdlee, dshan}@icu.ac.kr

Abstract — **In this paper, we propose adaptive QoS(Quality of Service) control mechanism and a architecture using multiagent framework to improve flexibility of a videoconference system. The proposed mechanism realizes more flexibility by changing their QoS control strategies dynamically. We implemented the mechanism, and our prototype system shows its capability of flexible problem solving against the QoS degradation, along with other possible problems within the given time limitation.**

Keywords — **Multiagent System, Adaptive QoS Control, Cooperation Protocol, Flexible Videoconference System**

1. Introduction

To use videoconference systems (VCSs) on heterogeneous computers and network environments, users have to consider the status of system resources, situations of other site over network, sledding of the meeting, and working condition of videoconference processes on machines, in other to maintain the comfortable system operation [1]-[4]. Usually these tasks burden novice users.

To reduce the users' burden of QoS control, automated parameter tuning mechanisms at application level have been introduced. These mechanisms are developed especially to maintain QoS adaptively in application layer on best effort type network, thus we call them adaptive QoS control mechanisms.

To reduce these various kinds of loads on users of desktop VCS, Flexible Videoconference System [5]-[7] has been designed in particular for novice users who want to use videoconference system efficiently. By adding some flexible features to traditional VCSs, FVCS can change its functions and performances autonomously in accordance with changes of user requirements and system/network environments. In the research area of adaptive QoS control based on the situations of environments, for example, VCS which can control its outgoing data rate considering. congestion condition of network has been developed. Though, static algorithms accomplish traditional QoS control mechanisms. This makes their problem solving capability monolithic, thus, flexible behavior considering importance or emergence of given problems is difficult to achieve.

In this paper, we propose adaptive QoS control mechanism to overcome this limitation explained above. Using this mechanism, we can deal with the problems such as exhaustion of resources, changing the QoS control dynamically based on the characteristics of the problem, status of problem solving process, user requirements, and so forth. In the concrete, we construct VCS as organization of intelligent software modules, i.e., agents, and also propose a new architecture of knowledge processing (M-INTER) in agents to switching problem solving strategy. After implementing the mechanism, our prototype FVCS shows its capability of solving the problems of QoS decreasing, along with other possible problems within the given time limit.

In section 2, we explain the basic concept of FVCS. Section 3 then presents adaptive QoS control mechanism and its architecture. The actual applying to VCS is also discussed. Finally, we illustrate the details of implementation and evaluate results of experiments using the prototype system.

2. Flexible Videoconference System

Flexible Videoconference System (FVCS) and flexible networking have been promoted aiming at providing a user-centered communication environment based on agent-based computing technology [5]-[8]. The primary objective of the project is to reduce lots of users' overloads in utilizing the VCSs, by effective use of traditional VCS software and expertise of designers/operators of the VCSs.

To lighten users' burdens of VCSs, FVCS is attained by embedding the following functionality to the existing VCSs, i.e., (F1) Service configuration function at the start of a session, and (F2) Service tuning function during the session. Here, (F1) composes the most suitable service configuration of VCS automatically by selecting best software modules and deciding their set-up parameters under the given conditions of environments and users. This function reduces users' burdens at the start up of videoconference session. (F2) adjusts the QoS autonomously according to the changes of network/computational environments or user requirements against the QoS. This function is realized by two phase tuning operations, i.e., parameter operation tuning for small-scale changes and reconfiguration of videoconference service for large-scale changes. In this paper we focus on (F2) and propose some improvement in (F2).

Figure 1 depicts the agent-based architecture of FVCS. Since in FVCS, high level process operations including task delegation and conflict resolution are required, the complex information exchanging among processes and intelligent information handling ability are also needed. These are

reasons why we applied agent-based computing technology to FVCS. Agents, i.e., intelligent software modules in VCS, and their cooperative behavior in accordance with inter-agent protocols realize the two functions described above [5].

In this paper, we concentrate on the parameter tuning of (F2) as adaptive QoS control. (F2) is achieved by mainly Video-Conf-Manager (VCM) agents as shown in Figure 1. Each VCM agent maintains the videoconference services provided to one specific user.

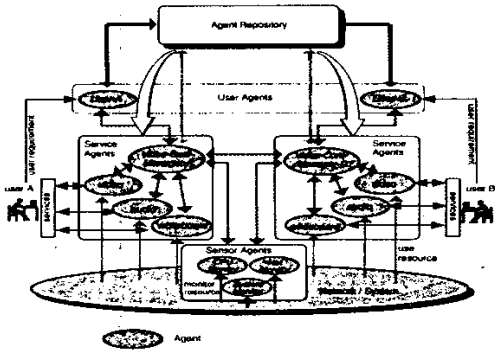


Figure 1. Agent-based Flexible Videoconference System

VCM agents exchange lots of data with User agents, Sensor agents, and Service agents frequently, and decide action sequence of QoS control onto videoconference process agents, i.e., video, audio, whiteboard agents. The parameter tuning in (F2) is driven by the following steps: 1) Changes are detected by Sensor agents or User agents, and they report the changes to VCM agents, 2) VCM agents negotiate each other to decide suitable operations against the videoconference process agents, 3) videoconference process agents set parameters of respective processes, 4) Sensor agents check recovery status and report it to VCM, 5) Repeat from 2) to 4) until the changes are recovered. By the try and error strategy explained above, FVCS can maintain QoS in a scalable manner.

3. Strategy-centric Adaptive QoS Control

3.1 Related Works

Some kinds of researches on application level QoS control are undertaken such as IVS[2] and framework-based approach[12]. IVS was developed aimed at the videoconferencing over Internet. IVS adjusts its outgoing data transmission rate by controlling the parameters of video coders based on feedback information about changing network conditions. It can also accept very simple user requirements by specifying policy of QoS control, while framework-based approach provides a skeleton to address two fundamental challenges for the construction of "network aware" applications, i.e., 1) how to find out about dynamic changes observed in network service quality and 2) how to map application-centric quality measures to network-centric ones.

In these approaches, we have found out several problems of the abilities concerning QoS control as follows.

(P1) Acceptable range of environmental changes are small: Both systems mentioned above are designed and customized in order to use on a specific network environment, i.e. the Internet environment. The QoS control behavior of them have been monolithic, so when we use them on an unexpected environment, QoS control abilities are extremely limited. The origin of the limitation is that they only have a single, fixed and deterministic QoS control strategy against lots of types of changes.

(P2) Load balancing capability is very limited: In VCS, each user terminal plays roles of both a sender and a receiver for videoconferencing. Since VCS imposes heavy load on CPU and I/O of each terminal node, the optimization of the load balancing operation considering status of both participants' terminal nodes are supported in a simple way, but not in IVS. The origin of the limitation should be 1) limited capability to deal with various types of changes, and 2) lack of negotiation mechanism between notes.

(P3) Meta level requirements of QoS are not considered: To achieve an effective and delicate QoS control, meta level requirements of QoS must be fulfilled by the systems. The basic level QoS requirements include video quality, video smoothness and audio quality, and the meta level requirements include deadline to resolve QoS degradation, accuracy of results of QoS control compared to the requirement, cost to resolve the problem, long term QoS policy and so on.

Our current version of FVCS provides partial solutions to these problems. Regarding (P1), the try and error strategy of parameter tuning in (F2) described in Section 2 is one of the solutions. For large-scale changes, reconfiguration of videoconference service in (F2) is another solution. In terms of (P2), we have defined some between VCM agents to coordinate load balancing of both user terminals. Monitoring capability for changes is also enriched by Sensor agents. However (P3) remains as our future work.

3.2 Adaptive QoS control with M-INTER model

To overcome these problems of traditional QoS control mechanisms described in Section 3.1 especially (P3), we propose a adaptive QoS control mechanism. The proposed mechanism is embedded to VCM agents in FVCS and realizes the service tuning function (F2).

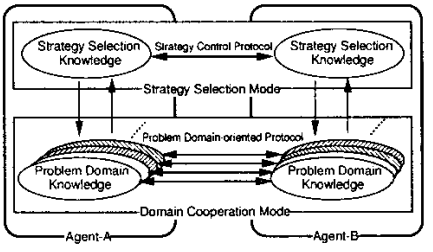


Figure 2. Conceptual Scheme

The adaptive QoS control mechanism is designed along with the following policies, i.e., (1) Introduce the knowledge representation scheme of meta level knowledge to control the

problem solving processes, (2) Give a design of function to incorporate multiple QoS control strategies, (3) Design a knowledge processing mechanism to switch the QoS control strategies using meta level knowledge, and (4) Realize the proposed mechanism as software modules to promote re-usability and maintenance during agent design and implementation. Figure 2 represents the concept of adaptive QoS control mechanism.

In this mechanism, the knowledge processing of QoS control will be performed in the following two different modes of agents in FVCS, namely Strategy Selection Mode and Domain Cooperation Mode.

(1) Strategy Selection Mode: This mode is important to overcome a limitation (P3) explained in the previous section. In this mode, agents monitor the meta-level conditions of cooperative behavior such as a class of given problem, a level of improvement during problem solving process, the rest period until deadline, and so forth. With these conditions, agents select the most adequate strategy by using Strategy Selection Knowledge. Moreover, each agent has to negotiate to select the best strategy, exchanging information of each cooperation statuses and problem domain knowledge of them. Hence, we have defined a Strategy Control Protocol between agents in order to promote the meta level cooperation. The essential idea of this approach is to introduce the meta level control of cooperation between agents based on Strategy Control Protocol which provides the effective coordination behavior for agents.

(2) Domain Cooperation Mode: In this mode, the problem domain specific cooperation is performed between agents. A problem domain means a class of problem to be resolved such as video QoS control problem domain and audio QoS control problem domain. In a problem domain, several strategies are prepared to be activated. Each strategy corresponds to a different method to resolve a specific problem, and the strategy is realized by a Problem Domain Knowledge in an agent and a Problem Domain-oriented Protocol (DoP) between agents. In this mode, one strategy is selected and activated during cooperation of agents with respect to situation of both the external environment and the cooperation status. The selection of strategy is in charge of Strategy Selection Mode.

The adaptive QoS control is realized by the alternative transition of these two modes. When a problem occurs, firstly, an agent begins to negotiate with other agents of FVCS to decide the most proper strategy on the given conditions in Strategy Selection Mode. Next, the agents transit to Domain Cooperation Mode, and they begin to perform the problem domain-oriented cooperation using specified Problem Domain Knowledge and DoP.

3.3 Meta-Interaction Architecture

We propose a Meta-Interaction (M-INTER) Architecture which consists of a new architecture of knowledge in VCM agents and a new protocol between agents to accomplish the strategy-centric adaptive QoS control (Figure 3). M-INTER is applied to VCM agents in FVCS.

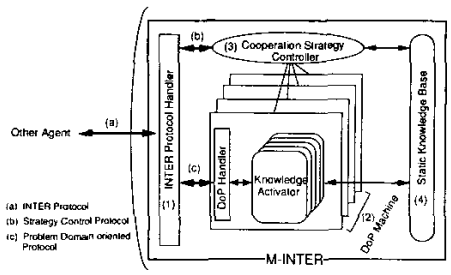


Figure 3. M-INTER Architecture

Table 1. Performatives Used in the INTER Protocol

Performative	Summary
RequestAction	S requests R to do something
Acceptance	S accepts the RequestAction
Refusal	S refuses the RequestAction
RequestInformation	S requests some information to R
Information	S sends some information to R replying RequestInformation
Report	S sends some information to R

(1) INTER Protocol Handler: It is a simple message handling module to cope with inter-agent communication messages. The messages are driven by INTER Protocol, the primary protocol used by cooperation between agents. Table 1 represents performatives, that is, means of communication primitives, of this protocol. In the table, "S" stands for a sender of a message, while "R" stands for a recipient of a message, respectively. When an agent A asks an agent B to do something, the agent A sends a RequestAction message to the agent B. After receiving the message, the agent B decides whether it can accept the request or not, and replies an Acceptance or a Refusal message to the agent A. When a agent wants to obtain some information from another agent, a RequestInformation message and an Information message are used for the request and the answer, respectively. A Report message is used for information exchanging without any requests.

(2) Problem Domain-oriented Protocol Machine (DoP Machine): A protocol handling module to achieve the problem domain-oriented cooperation in Domain Cooperation Mode. The DoP Machine is an implementation model of the Problem Domain Knowledge in Figure 2. There are several DoP machines in the M-INTER model, and each of them handles different DoPs respectively which are dedicated to a specific problem domain. A DoP Machine consists of a DoP Handler and several Knowledge Activators(KAs). DoP Handler is a simple parser of DoP, while Knowledge Activator decides actions of an agent based on the static knowledge, when it receives a message from other agent. There can several KAs in a DoP machine. Each KA can be realized by different knowledge processing implementations. For instance, a KA can be implemented by a rule-based inference engine, on other hand, other KAs can be implemented by a simple procedural-type inference module. A single KA is activated during a cooperation.

(3) Cooperation Strategy Controller: A strategy control module activated in Strategy Selection Mode. This module is

charged with selection of DoP Machine and Knowledge Activator, negotiating with other agents using Strategy Control Protocol (Table 2). The selection is done in accordance with the meta level requirements of QoS and cooperation status. Make-Coop related messages (Request, Acceptance, Refusal-Make-Coop messages) are used when two agents begin cooperative problem solving. In these messages, some kinds of information about required cooperation, such as goal and deadline, are exchanged. Close-Coop related messages are used to close the cooperation. Change-Protocol related messages are used in case that change of DoP machine is required during a cooperation. Moreover Change-Coop-Status related messages are used to change cooperation status such as goal and deadline.

Table 2. Performatives Used in the Strategy Control Protocol

Performative	Summary
Request-Make-Coop	S requests R to start cooperation
Acceptance-Make-Coop	S accepts a request from R to start cooperation
Refusal-Make-Coop	S refuses a request from R to start cooperation
Request-Close-Coop	S requests R to terminate cooperation
Acceptance-Close-Coop	S accepts a request from R to terminate cooperation
Refusal-Close-Coop	S refuses a request from R to terminate cooperation
Request-Change-Protocol	S requests R to change protocol
Acceptance-Change-Protocol	S accepts a request from R to change protocol
Refusal-Change-Protocol	S refuses a request from R to change protocol
Request-Change-Coop-Status	S requests R to change cooperation status
Acceptance-Change-Coop-Status	S accepts a request from R to change cooperation status
Refuse-Change-Coop-Status	S refuses a request from R to change cooperation status

(4) Static Knowledge Base: A container of expert knowledge that is used by Cooperation Strategy Controller and Knowledge Activators. This knowledge is represented by a set of frame descriptions.

3.4 Applying M-INTER Model to FVCS

To apply M-INTER model to FVCS, we have defined four types of DoP Machines. These DoP machines are designed to be used for QoS control of video process in FVCS.

(1) Basic Protocol Machine: A simple protocol machine to control QoS of video in both sites. Using this protocol, VCM agents direct videoconference processes rotatably and repeatedly to increase/decrease values of QoS parameters in a fixed range, until resource conditions are recovered. There are five kinds of Knowledge Activators, which have each range of change respectively.

(2) Compromise Level Protocol Machine: A deliberative type protocol machine to adjust QoS by trial and error strategy. With this protocol, VCM agents have each mental state on limitations of degradation of QoS parameters, namely compromise level [13]. VCM agents perform negotiation to find the compromise point each other, changing their compromise level dynamically. This strategy is rather costly, but it can achieve QoS tuning precisely.

(3) Time Restricted Protocol Machine: A protocol machine that can cooperate considering time restrictions such as deadline in the first priority. With this protocol, cooperation between agents is terminated forcedly regardless of degree of problem solvency.

(4) Reactive Protocol Machine: A reactive type protocol machine to reduce communication overhead between agents. Although accuracy of parameter tuning is not guaranteed, quick response against the changes is enabled. This strategy is used on the unstable environment where resources are expected to be changed at very short time interval. It is also used as their last card when deadline comes nearby.

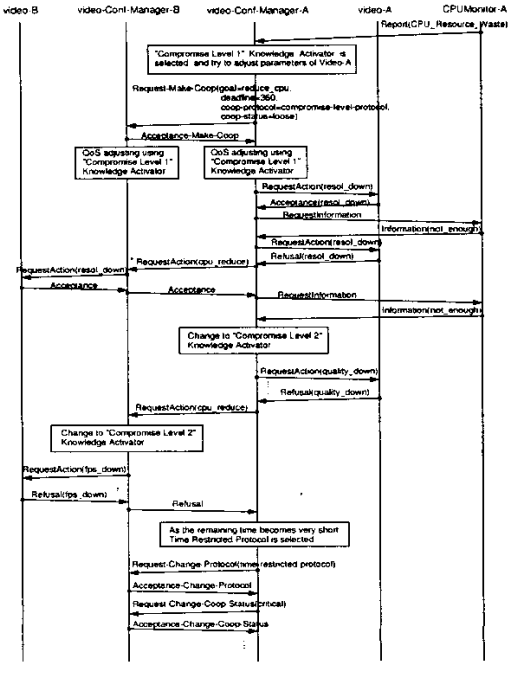


Figure 4. an Example of Agents' Cooperation with M-INTER Architecture

The behavior of agents based on M-INTER architecture against the change of CPU resource is illustrated in Figure 4.

(1) Detection of resource degradation: CPUMonitor-A agent detects deviation of CPU resources from acceptable range, and reports to VideoConfManager-A (VCM-A) agent with Report message.

(2) Selection of initial strategy: Cooperation Strategy Controller in VCM-A selects the Compromise Level Protocol Machine because there is temporal allowance to deadline. Firstly, "Compromise Level 1" Knowledge Activator in Compromise Level Protocol Machine of VCM-A is activated and tries to adjust parameter of Video-A agent within its compromise level.

(3) Cooperative QoS control with Compromise Level Protocol Machine: If VCM-A can not release the resource, it requires the collaboration to VCM-B by issuing

Request-Make-Coop message to VCM-B to make cooperation relation. The "Compromise Level 1" Knowledge Activator of VCM-B is activated and tries to adjust parameter of Video-B agent within his compromise level as well. When VCM-B can not release the resource too, Cooperation Strategy Controller switches Knowledge Activator to "Compromise Level 2".

(4) Change of DoP Machine: In case that the specified deadline comes nearby, Cooperation Strategy Controller switches DoP Machine to Time Restricted Protocol Machine to keep the deadline. With this protocol machine, requirements on time constraints are added in a message of DoP, so punctual behavior of agents is enabled.

(5) Termination of cooperative action: When CPU resource is released, cooperation relation of VCM-A and VCM-B is closed.

4. Experiments and Evaluation

4.1 Implementation

The proposed architecture based on M-INTER model is embedded to the VCM agents of FVCS, described in section 3. We have used ADIPS Framework [9]-[10] as an agent-based computing infrastructure. The proposed architecture of M-INTER model is written in Tcl/Tk programming language [12] extending the agent's knowledge architecture provided by original ADIPS Framework.

4.2 Experiments on Agents' Behavior

The FVCS based on M-INTER architecture has been implemented and experimented under the environment shown in Figure 5. To evaluate the flexibility of agents provided by our model, we have changed the CPU resources forcedly, and have monitored the system's behavior. The result clearly shows that the experiment with the proposed architecture acts much more flexibly than the existing systems. Firstly, some extra load on CPU of WS-B has been added externally, and observed the changes of QoS parameters of video process, i.e., frame rate, encoding quality and resolution. At that time, on WS-A in Figure 5, User-A represents his requirements of smoothness in movement of video to the highest priority, second highest priority to video quality and lowest priority to video resolution. On the other hand, User-B on WS-B represents its requirements with the highest priority to video quality, second highest priority to smoothness, and lowest priority to resolution. When the problem of CPU resources insufficiency is occurred, we provided a limited time to solve the problem to the system. The given time limits were 120 and 180 seconds, respectively.

Figure 6 and 7 represents the transition of the parameters' values controlled by the agents. In the graph, x-axis represents the time (second) and y-axis represents each parameter values observed at the recipient site. The parameter values are expressed in percentage when the following values are regarded as 100%;

- CPU load: 100%
- Smoothness in movement: 35-fps
- Quality: 32-level
- Resolution: 3-level

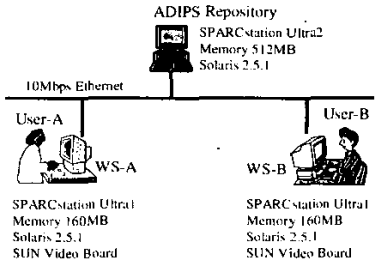
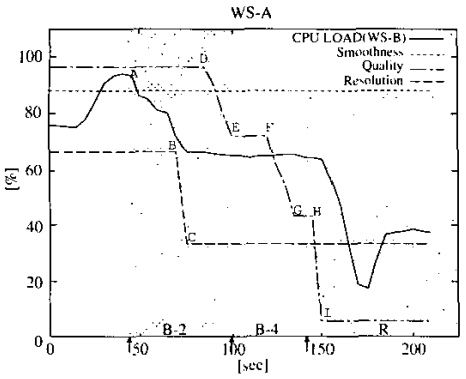


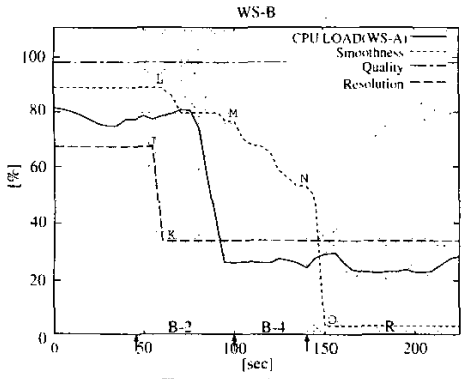
Figure 5. Experiment Environment

In the graph, symbol (↑) indicates the switching time of DoP machines or Knowledge Activators (KAs). 'B-1', 'R' etc. represent the types of DoP machines and the KAs used in the time slot.

Figure 6 represents the transition of the parameter's value controlled by the agents when the given time limitation is 120 seconds. When the CPU load of WS-B increases (at point A or after 40 seconds), agents of FVCS began cooperative actions. At first, the KA 2 of the Basic Protocol Machine (B-2) is selected. There exist five types of KAs in Basic Protocol Machine. If this numerical value of this KA becomes big, the slope becomes sharper.



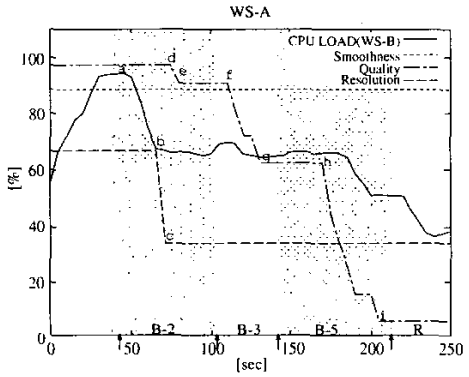
(a) Change of QoS at User-A



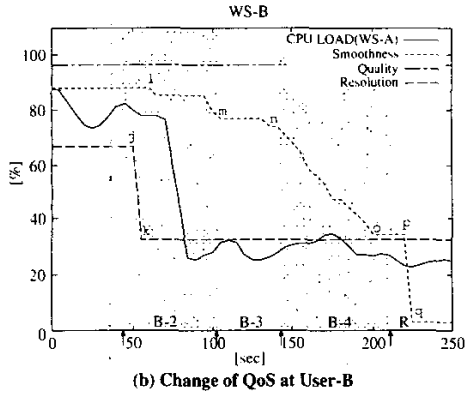
(b) Change of QoS at User-B

Figure 6. Behavior of FVCS agninst CPU Variation 1: (time limit 120s)

In the area of 'B-2'of Figure 6 (a), the resolution of video provided to User-A at WS-A was reduced at point B-C according to user priority. Secondly, the video quality was reduced at point D-E. While the resolution of video provided to User-B at WS-B was reduced at point J-K in 'B-2' shown in Figure 6 (b). In the next instance, smoothness was reduced at point L-M. In this stage, the Cooperation Strategy Controller starts activating. It calculates the remaining time and the degree of problem solution (in this case, the release of CPU resource). By considering these results it selects the KA without changing the protocol machines. As a result, the parameter value has a sharp declination between (M-N) points. When the remaining time becomes very small (at the warning stage), the Cooperation Strategy Controller starts activating again and changes its protocol from Basic to Reactive Protocol Machine, 'R'. During a Reactive Protocol session, only the highest priority QoS parameter remain unchanged, but other QoS parameters are decreased to the minimum, without considering any further conditions. Therefore, CPU resources are released (points H-I, N-O). In this given time limit (120 seconds), the agents try different strategies to satisfy the user requirements as much as possible and finally the system succeeds in releasing the CPU resources.



(a) Change of QoS at User-A



(b) Change of QoS at User-B

Figure 7. Behavior of FVCS agninst CPU Variation 2: (time limit 180s)

Figure 7 is the case when the time limit was set up for 180 seconds. Extra CPU load is injected (at point a or after 40 seconds), and the agents of FVCS began cooperative actions at this point.

In this case, as the given time (180 seconds) is longer than the previous time limit (120 seconds), from the beginning, the Cooperation Strategy Controller selects the KA 2 (B-2) of the Basic Protocol Machine. Since the system has enough time in this case, it selects the KAs B-3, B-4 and B-5 step by step without making any hurry to select any reactive protocols. Between 140 seconds to 210 seconds, the WS-B side selected the KA 4 (B-4), whilst the WS-A side selected 5 (B-5). As we see from the experimental results, the agents act flexibly to satisfy the user requirements as much as it can and finally the system succeeds in releasing the CPU resources.

5. Conclusion

In this paper, we have proposed a mechanism and it's architecture called M-INTER to accomplish adaptive QoS control. This model extends the functions of the QoS control mechanism by sophisticated cooperation among agents in FVCS. The proposed mechanism analyzes the property of the problem occurred on QoS, considers every step during a session, changes the strategies dynamically and solves the problem even more flexibly.

We have implemented the proposed mechanism and carried out experiments by applying it to FVCS. The experimental results clearly show that the flexibility is improved. The future works of this system include improvement of the efficiency of the Cooperative Strategy Controller.

REFERENCES

[1] S. MaCanne and V. Jacobson, "Vic: a flexible framework for packet video", ACM Multimedia, pp.511-522, Nov. 1995.

[2] T. Turletti and C. Huitema, "Videoconferencing on the Internet", IEEE/ACM Trans. on Networking, Vol.4, No.3, pp.340-351, 1996.

[3] V. Jacobson and S. McCanne, "Visual Audio Tool", Lawrence Berkeley Laboratory. ftp://ftp.ee.lbl.gov/conferencing/vat

[4] V. Jacobson and S. McCanne, "LBL whiteboard", Lawrence Berkeley Laboratory. ftp://ftp.ee.lbl.gov/conferencing/wb

[5] T. Suganuma, S. Fujita, K. Sugawara, T. Kinoshita, and N. Shiratori, "Flexible Videoconference System Based on Multiagent-based Architecture", Trans. IPSJapan, Vol.38, No.6, pp.1214-1224, 1997 (Japanese).

[6] T. Suganuma, T. Kinoshita, K. Sugawara, and N. Shiratori, "Flexible Videoconference System based on ADIPS Framework", Proc. of the 3rd International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology(PAAM98), pp.83-100, 1998.

[7] S. D. Lee, T. Karahashi, T. Suganuma, T. Kinoshita, and N. Shiratori, "Construction and Evaluation of Agent Domain Knowledge for Flexible Videoconference System", Trans. IEICEJ, Vol. J83-B, No. 2, pp.195-206, 2000 (Japanese).

[8] N. Shiratori, K. Sugawara, T. Kinoshita, and G. Chakraborty, "Flexible Networks: Basic Concepts and Architecture", IEICE Trans. Commun.,Vol.E77-B, No.11, pp.1287-1294, 1994.

[9] S. Fujita, H. Hara, K. Sugawara, T. Kinoshita, and N. Shiratori, "Agent-based design model of adaptive distributed systems", Applied Intelligence, Vol.9, No.1, pp.57-70, July/Aug. 1998.

[10] T. Kinoshita and K. Sugawara, "ADIPS Framework for Flexible Distributed Systems," Springer-Verlag Lecture Notes in AI, 1599, pp. 18-32, 1998.

[11] J. Bolliger and T. Gross, "A Framework-Based Approach to the Development of Network-Aware Applications", IEEE Trans. on Software Engineering, Vol.24, No.5, 1998.

[12] Ousterhout, J. K., "Tcl and the Tk Toolkit", Addison-Wesley, 1994.