Adaptive QoS Control Mechanism in Flexible Videoconference System

Sungdoke Lee[1], Dongsoo Han[1], and Sanggil Kang[2]

[1] School of Engineering, Information and Communication University
P. O. Box 77, Yuseong, Daejeon, 305-600 Korea
{sdlee, dshan}@icu.ac.kr
[2] Computer Science & Engineering
Inha University
253 Younghyun-dong, Nam-gu, Incheon, 402-751 Korea
sgkang@inha.ac.kr

Abstract. In this paper, we propose an adaptive QoS (Quality of Service) control mechanism and its corresponding architecture using multi-agent framework to improve flexibility of a videoconference system (FVCS). The proposed mechanism realizes more flexible by changing their QoS control strategies dynamically than a conventional videoconference system. We modelled the prototype of our modified FVCS and implemented the proposed mechanism by varying QoS parameters. The prototype system shows its capability of flexible problem solving against QoS degradation, along with other possible problems within the given time limitation.

1 Introduction

To use videoconference system (VCS) on heterogeneous computers and network environments, users have to consider the status of system resources, situations of other site over network, and working condition of videoconference processes on machines in order to maintain a comfortable system operation [1]-[4]. Usually, these tasks of Quality of Service (QoS) burden novice users. To reduce the users' burden, Flexible Videoconference System (FVCS) [5]-[7] has been designed by adding some flexible features to traditional VCS. FVCS can change its functions and performances autonomously in accordance with changes of user requirements or system/network environments. In the research area of adaptive QoS control based on the network environments, an FVCS has been developed in order to control its outgoing data rate for considering congestion condition of the network. However, most FVCSs control the QoS in a static fashion, which makes their problem solving capability monolithic. Because of that, flexible behavior considering importance or emergence of given problems during conference is difficult to achieve.

In this paper, we propose an adaptive QoS control mechanism to overcome the limitation explained above. The adaptive QoS control mechanism 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

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part I, LNAI 4251, pp. 260–267, 2006.

process, and user requirements. The QoS control is done by a new architecture of knowledge processing called M-INTER to switching problem solving strategy. In the experimental section, we develop our prototype FVCS and show its capability of solving the problems of QoS decreasing, along with other possible problems within the given time limit.

The remainder of this paper is organized as follows: Section 2 briefly explains basic concept of FVCS. Section 3 then presents adaptive QoS control mechanism and its architecture. Section 4 shows experiment result and its analysis using the prototype system. In Section 5, we conclude our paper.

2 Flexible Videoconference System

To lighten users' burdens of VCS, FVCS [5]-[7] is attained by embedding the following functionality to the existing VCS, 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. The function reduces users' burden 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. The 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).

Due to the static QoS control, most existing FVCSs have several problems as follows: (P1) Acceptable range of environmental changes is small: Most FVCSs are designed and customized to use on a specific network environment, i.e. the Internet environment. For an unexpected environment, QoS control ability is extremely limited. (P2) Load balancing capability is very limited: Since each user terminal plays roles of both a sender and a receiver for videoconferencing, heavy load on CPU and I/O of each terminal node can be imposed, which causes limited capability to deal with various types of changes. (P3) Meta level requirements of QoS are not considered: To achieve an effective and delicate QoS control, meta level monitoring capability for detecting classes of problem and progress status of problem resolving process is needed.

3 Strategy-Centric Adaptive QoS Control

3.1 Adaptive QoS Control with M-INTER Model

To overcome the problems of traditional QoS control mechanisms described in the previous section, especially (P3), we embed an adaptive QoS control mechanism to VCM agents in FVCS and realizes the service tuning function (F2). The adaptive QoS control mechanism is designed along with the following policies: (1) Introduce the knowledge representation scheme of meta level knowledge

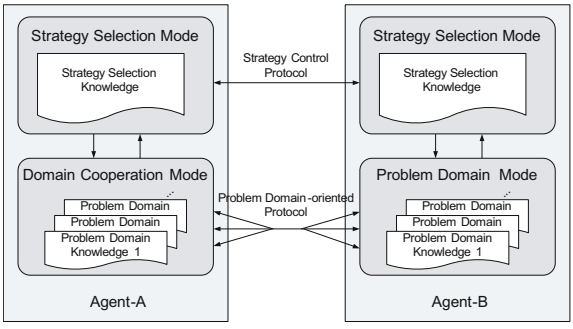

Fig. 1. Conceptual scheme

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. (4) Realize the proposed mechanism as software modules to promote re-usability and maintenance during agent design and implementation.

Fig. 1 represents the concept of adaptive QoS control mechanism. In the 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. In Strategy Selection 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, etc. With the conditions, agents select the most adequate strategy by using Strategy Selection Knowledge. Moreover, each agent has to negotiate to select the best strategy and exchange information of each cooperation status and problem domain knowledge of them. Hence, we defined a strategy control protocol between agents in order to promote the meta level cooperation. In Domain Cooperation Mode, the problem domain specific cooperation is performed between agents. A problem domain means a class of problems 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 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.

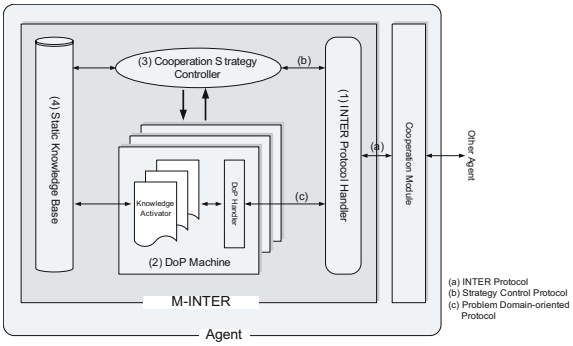

Fig. 2. M-INTER architecture

3.2 Meta-Interaction Architecture

To accomplish the strategy-centric adaptive QoS control, we propose Meta-Interaction (M-INTER) Architecture which consists of a new architecture of knowledge in VCM agents and a new protocol between agents as seen in Fig. 2.

From Fig. 2, INTER Protocol Handler is a simple message handling module to cope with inter-agent communication messages driven by INTER Protocol. The primary protocol used for cooperating between agents is represented in communication primitives as in Table 1. In the table, "S" stands for a sender of a message, while "R" stands for a recipient of a message. 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 [6].

Problem Domain-oriented Protocol Machine (DoP Machine) is 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 Fig. 1. The 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 be 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. Cooperation Strategy Controller is a strategy control module activated in Strategy Selection Mode. This module is charged with selecting DoP Machine and Knowledge Activator and negotiating with other agents using Strategy Control Protocol as seen in 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 the messages some kinds of information about required

Table 1. Performatives Used in the INTER Protocol

Performative	Performative
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

Table 2. Performatives used in the strategy control protocol

Performative	Performative
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

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 the 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. Static Knowledge Base is a container of expert knowledge that is used by Cooperation Strategy Controller and Knowledge Activators.

By applying the architecture described above, it can change cooperation strategies flexibly switching DoP machines and Knowledge Activators. The mechanism can improve dynamic range against the changes on environments and user requirements, and it can also realize meta level requirements of QoS.

3.3 Applying M-INTER Model to FVCS

To apply M-INTER model to FVCS, we define four types of DoP Machines for QoS control of video process in FVCS as follows: (1) Basic Protocol Machine: a simple protocol machine to control QoS of video in both sites. By the protocol, VCM agents direct videoconference processes repeatedly to increase/decrease the values of QoS parameters in a fixed range until resource conditions are recovered. (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 meta state on limitations of degradation of QoS parameters, namely compromise level. VCM agents perform negotiation to find the compromise point each other

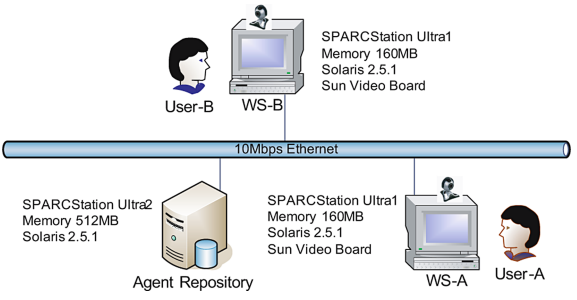

Fig. 3. Experiment environment

and to change their compromise level dynamically. (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 the 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.

4 Experiments and Evaluation

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 [8] as an agent-based computing infrastructure. The proposed architecture of M-INTER model is written in Tcl/Tk programming language [9] extending the agent's knowledge architecture provided by original ADIPS Framework.

The FVCS based on M-INTER architecture has been implemented and experimented under the environment shown in Fig. 3. 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 Fig. 3, User-A represents his requirement 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 limit was 120 seconds, respectively.

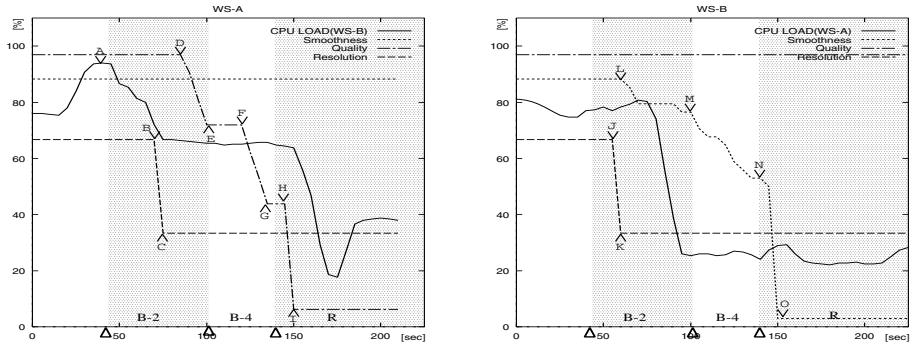

Fig. 4. Behavior of FVCS against CPU variation (time limit 120s): (a) Change of QoS at User-A, (b) Change of QoS at User-B

Fig. 4 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. In the graph, symbol triangle indicates the switching time of DoP machines or Knowledge Activators (KAs). 'B-1(the KA 1 of Basic Protocol Machine)', 'R(the Reactive Protocol Machines)' etc. represent the types of DoP machines and the KAs used in the time slot. Basically, when the CPU resources are found insufficient (maybe CPU load of WS-A or WS-B increases), the agents aim to maintain the stability of the system by considering the user's requirements. In this case, it might reduce the QoS and consequently release some CPU resources. 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. In the area of 'B-2'of Fig. 4 (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 Fig. 4 (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.

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 architecture 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 architecture and carried out experiments by applying it to FVCS. The experimental results clearly show that the flexibility is improved.

For the future work of our system, we need to develop an algorithm for improving the efficiency of the Cooperative Strategy Control.

References

1. MaCanne, S., Jacobson, V.: Vic: a flexible framework for packet video. ACM Multimedia, Nov. (1995) 511-522
2. Turletti, T., Huitema, C.: Videoconferencing on the Internet. IEEE/ACM Trans. on Networking, Vol. 4, No. 3. (1996) 340-351
3. Bolliger, J., Gross, T.: A Framework-Based Approach to the Development of Network-Aware Applications. IEEE Trans. on Software Engineering, Vol. 24, No. 5. (1998)
4. Jacobson, V., McCanne, S.: Visual Audio Tool. Lawrence Berkeley Laboratory. ftp://ftp.ee.lbl.gov/conferencing/vat
5. Suganuma, T., Kinoshita, T., Sugawara, K., Shiratori, N.: Flexible Videoconference System based on ADIPS Framework. Proc. of the 3rd Int. Conf. and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (1998) 83-100
6. Lee, S. D., Karahashi, T., Suganuma, T., Kinoshita, T., Shiratori, N.: Construction and Evaluation of Agent Domain Knowledge for Flexible Videoconference System. Trans. IEICEJ, Vol. J83-B, No. 2. (2000) 195-206
7. Shiratori, N., Sugawara, K., Kinoshita, T., Chakraborty, G.: Flexible Networks: Basic Concepts and Architecture. IEICE Trans. Commun., Vol. E77-B, No. 11. (1994) 1287-1294
8. Kinoshita, T., Sugawara, K.: ADIPS Framework for Flexible Distributed Systems. Lecture Notes in Computer Science, Vol. 1599. Springer-Verlag, Berlin Heidelberg New York (1998) 18-32
9. Ousterhout, J. K.: Tcl and the Tk Toolkit. Addison-Wesley (1994)