マルチメディア通信と分散処理 70−15
（1995. 5. 26）

やわらかいネットワークに関する実験的一考察

杉浦茂樹[1] Martin MOSER[1] Goutam CHAKRABORTY[1]
菅原研次[2] 白鳥則郎[1]

[1] 東北大学 電気通信研究所／情報科学研究科
[2] 千葉工業大学 情報工学科

アプリケーション、とりわけコンピュータネットワークのためのアプリケーションはやわらかさ — 知性、恒常性、発展性 — が要求されるが、これまでは充分には提供されていなかった。
本稿では、まずはじめに「やわらかいネットワーク」を概念的に定義する。次にエージェントに基づくやわらかいネットワークの実験システム (TV 会議システム) について述べる。このシステムは利用可能な資源の変更が発生したときにも、ユーザを煩わせることなく、その状況で最大限のサービスを提供する機能を有する。

An Experimental Consideration of Flexible Network

Shigeki SUGIURA[1] Martin MOSER[1] Goutam CHAKRABORTY[1]
Kenji SUGAWARA[2] Norio SHIRATORI[1]

[1]Research Institute of Electrical Communication,
Graduate School of Information Sciences, Tohoku University
[2]Department of Computer Science, Chiba Institute of Technology

Till now applications, especially these using computer network, have neither the intelligence to cope with the different and varied end users' requirements, nor the power of homeostasis so that any internal disaster is absorbed by the network so that it automatically goes to another stable equilibrium state. The first problem limits ordinary users to access the functions offered by the system, while the second leads applications to degrade disgracefully.
In this paper we first propose the idea of Flexible Networking. We then describe the experimental system using the architecture based on agents. It has a mechanism not to bother the user if changes in the assets occur, but to offer maximum possible functionalities under the given circumstances.

1 Introduction

In today's systems, failures or new changes appear as a big hurdle. Whenever such a change occurs, it takes a long time for the system and the users to adapt to that. The common goal of everyone involved in the field of research regarding largescale information network is to create an environment, where users would be able to interact with the network in a seamless manner to exploit its full potential, as well as different components of the network should be able to communicate seamlessly. Moreover, any change or disaster of the networks hould not be observable from outside, or should be absorbed gracefully by the system. In other words the network will behave flexibly maintaining the QoS offered to the users. We go one step ahead, to build next generation network infrastructure. There are already some reports, including ours, giving an outline of the image of flexible networks as a key to solve future networking problems[1, 2]. We introduce here the broader concept of flexible information network.

In this paper we propose an approach to realize the Flexible Networking. We then describe a TV conferencing system we are implementing, s an example to demonstrate the concepts using agent[3]. It will not bother the user even if there are changes in the assets. It continues to offer the maximum possible functionalities under the given circumstances.

2 Flexible Networking: an outline

There is not yet a common view of what a Flexible Network should be. In this paper we propose an approach that allows an user to define a service and its *quality of service frame*.

In the following subsections we give a brief outline of the theoretical background of the Flexible Network.

2.1 Configuration and degradation

An application provides a service that can be seen as consisting of a set of elementary services, which either generate data for participants of the session, or present the data generated by them.

It seems desirable to enpower users to specify the the following four features of the service they desire:

Generation quality Depending on the current situation, the quality requirements towards data generated locally may vary.

Presentation quality The participants of a session may have requirements towards the data coming in from a presenter that differ from the generation quality the presenter chose.

Priority Priorities allow the users to specify which elementary services and which features of the elementary services are most important and should therefore be kept from degrading as long as possible.

Limitations Degradation should respect lower bounds of services.

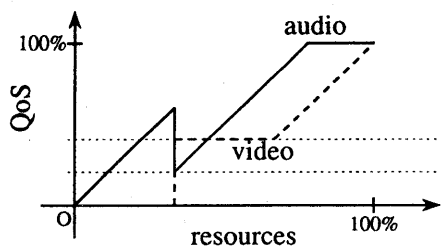

Figure 1: Services degrade according to priorities.

In Figure 1, the desired behavior for a service consisting of a video and an audio connection is presented: The audio channel has been given a higher priority than the video channel, as the spoken words usually carries more information than the display of the partner's face.

Please note that this scheme keeps both audio and video service operational with much less computational assets available.

2.2 The model

For the following discussion we model a participant in a distributed application. The model consists of four elements:

User requirements The user requirements provided by the participant can be given by a four tuple $\langle r_g, r_p, <, L\rangle$, defining the four items identified in the last section: the generation requirements r_g, the presentation requirements r_p, the priorities of the elementary services and their features given as a partial order $<$, and a set L of limitations of the elementary services. (Please note that the first two requirements implicitly define the set of elementary services to be provided.)

Work station Each participant holds a work station with computational assets a, which can be described in terms of CPU cycles, memory, bandwidth etc.

Peripherals Some hardware connected to the work station generates a stream of data a_g, e.g. a video picture of the participants face, a recording of his or her voice etc. The data stream a_g, which can be described in terms of frames per second, resolution, chrominance, and others, is consumed by some of the elementary services to generate a stream of data g that is transmitted to the other participants via the network.

Network A collection a_p of data streams similar to the data stream g that is sent over the network to the other participants is received over the network from the other participants. It is presented to the participant by some of the elementary services.

More formally speaking, the available computational assets, both of the work station and the network, can be described by a point in a linear vector space V_D with dimensions {CPU cycles, memory, IO bandwith, available data, ... }. (The "D" stands for "demands", for reasons that

will become apparent soon.) In analogy, the participant's requirements r_g and r_p, as well as the data assets a_g and a_p, can be expressed as points in a linear vector space V_R with dimensions {frames/sec., pixel resolution, audio quality, ...}. The dimensions of V_R are ordered by the partial order $<$. However, each dimension may have two ranks in the order, one for the generation and one for the presentation requirements. (A single partial order instead of two partial orders has been chosen such that presentation can be preferred to generation, for example.) The set L of limitations, eventually, define points on the axes of V_R, giving the lower bounds for r_g and r_p. For simplicity of graphical presentation, in the following figures only two-dimensional projections of V_R and V_D are given.

The generation requirements r_g define what kind of output data to produce and transmit to the other participants. At the other participants' sites the data is presented according to the presentation requirements r_{pi} of the individual participant. However, the presentation requirements r_{pi} may vary, and also may be different from the data generation requirements r_g. It is thus necessary to settle on some commonly agreed data generation requirements, which can be achieved by a policy π shared by all participants:

$$\begin{array}{rlll} \pi: & V_R^n & \rightarrow & V_R \\ & \langle r_g, r_{p2}, \ldots, r_{pn} \rangle & \mapsto & r_{gp} \end{array}$$

Possible policies are e.g. the elementwise maximum, minimum, or average. (More sophisticated strategies that the common policy π can easily be imagined. However, they are not the objective of this paper.)

Further, the available data assets may not match the data assets required to provide the service as requested. This may be due to a recent change in the requirements, or a different format of incoming or generated data. The requirements that can be satisfied by the service with the data assets available can be obtained by computing the elementwise minimum of r_{gp} and a_g, and r_p and a_p:

$$\begin{array}{rlll} m_g: & V_R \times V_R & \rightarrow & V_R \\ & \langle r_{pi}, a_g \rangle & \mapsto & r'_g = \min(r_{gp}, a_g) \end{array}$$

$$\begin{array}{rlll} m_p: & V_R \times V_R & \rightarrow & V_R \\ & \langle r_p, a_p \rangle & \mapsto & r'_p = \min(r_p, a_p) \end{array}$$

We now hold the requirements r'_g and r'_p of the service that was agreed on by the participants, and that is supplyable with the available data assets. We can now consider the computational assets necessary to provide this service. The service can be decomposed into elementary services, i.e. the vectors for the requirements r'_g and r'_p can be decomposed into an orthogonal vector sum. (Orthogonality is required, as using the same elementary service twice will not yield increase quality of service. Further, the elements r'_{gi} of the vector sum need no be parallel to the axes of V_R, as an elementary service may satisfy requirements in more than one dimension.) The decomposition of r'_g is not necessarily unique, however, as one elementary service may present video and audio, while there may be two individual services that only deal with either audio and video.

Now suppose that there exists a combination of elementary services that satisfies presentation and generation requirements r'_g and r'_p, respectively. An elementary service may provide several realizations for the same requirements, each realization consuming a different amount of computational assets. (There may be no realizations available for certain requirements, however. We will come back to this problem in the next section.) The individual realizations can be regarded as basis transforms from the user requirement vector space V_R to the asset demand vector space V_D. (For simplicity of presentation only realizations for generating realizations are given in the Figure.) If a set of m generating and a set of n presenting realizations can be found, such that the overall computational demands do not exceed the existing computational assets a, the service supplyable with the available data assets can be provided. In other words: The m generating elementary services r'_{gi} and the n presenting elementary services r'_{pi} have to meet the following constraints:

$$\begin{aligned} \sum_{i=1}^{m} r'_{gi} &= r'_g \\ \sum_{i=1}^{n} r'_{pi} &= r'_p \\ d &:= \sum_i d_{gi} + \sum_i d_{pi} \\ &= \sum_{i=1}^{m} t(r'_{gi}) + \sum_{i=1}^{n} t(r'_{pi}) \leq a \end{aligned}$$

2.3 Flexibility

Due to high requirements or lacking computational assets, it is not always possible to find a combination of realizations that satisfies the

generation and presentation requirements r'_g and r'_p, but also demands no more than the available assets a. Furthermore, although the elementary services exist, there may be no realizations available for all possible requirements. In both cases, the requirements have to be shifted to new requirements r''_g and r''_p, which specify a service that can be realized with the available computational assets.

Shifting requirements from r'_g and r'_p to r''_g and r''_p introduces *service-requirement* differences Δ_g and Δ_p. In order to account for the different priorities and limitations of the elementary services and their features, we use the following weighted length measure, rather than the standard length, for a vector $v = (v_1, v_2, \ldots, v_n)$:

$$\|v\| := \sqrt{\sum_{i=1}^{n} (p(v_i) \cdot \sigma(v, L))^2}$$

The function σ is a threshold function which returns 0, if there exists a v_j that is lower than the limit L_j for the j-th dimension, or v_i else. This accounts for the fact that a realization of an elementary service is useless, if any of the requirements the elementary service has to satisfy is below the lower bound.

Furthermore, the weights $p(v_i)$ are the priorities of the requirements and are defined inductively using the partial order $<$. The weight of a dimension intuitively corresponds to the level of priority of the dimension.

$$p(v_i) := \begin{cases} 1 \\ \quad \text{(if } v_i \text{ is one of the smallest elements of } <) \\ \max\limits_{\forall v_j < v_i} (p(v_j)) + 1 \\ \quad \text{(else)} \end{cases}$$

We can now define Δ_g and Δ_p, which supply a measure for how much the requested service and the provided service differ:

$$\Delta_g := \frac{\|r''_g - r'_g\|}{\|r'_g\|}$$

$$\Delta_p := \frac{\|r''_p - r'_p\|}{\|r'_p\|}$$

Due to the different weights of the requirements, the differences Δ_g and Δ_p vary more strongly when the requirements r''_g and r''_p are shifted in the direction of a highly priorized dimension, than when shifted by the same amount in the direction of a lower priorized dimension. Shifting the requirements below the lower limit in one dimension forces the requirements to zero in all dimensions of the corresponding elementary service, which causes the according service-requirement difference to increase abruptly.

We can now state our notion of flexibility as a minimization problem:

Definition (Flexibility)
An application is called *flexible*, if it minimizes the service-requirement differences Δ_g and Δ_p.

3 The TV conferencing experimental system

3.1 Objective of the experiment

In this chapter we describe a TV conferencing system, which we are implementing as an example to demonstrate the concepts introduced in the previous section. The points of flexibility we are dealing with in this paper are the following two points:

(a) Minimization of the changes in the operation depending on changes in the computational assets.

(b) Maximization of the service, if the available resources do not suffice to fulfill the user requirements perfectly.

The TV conferencing system is a system that allows communication via video and audio channels, and therefore requires real-time transmission and processing. Therefore, its performance is easily influenced by changes in the computational assets, as e.g. work station and network resources. Present systems do either a) not consider this kind of changes in the computational assets, or b) have the resources being directly allocated by the user.

We experiment our system as shown in Figure 2 with a TV conference connecting Tohoku University and Chiba Institute of Technology. The work stations are existing work stations, and the network used is the WIDE Internet. Therefore, we cannot ignore the changes in the computational and network assets in our experimental system. As in our experiment we introduce an architecture based on agents, the user is not bothered if changes in the assets occur. Rather, in this situation we execute a mechanism that offers the maximally possible functionality under the given circumstances.

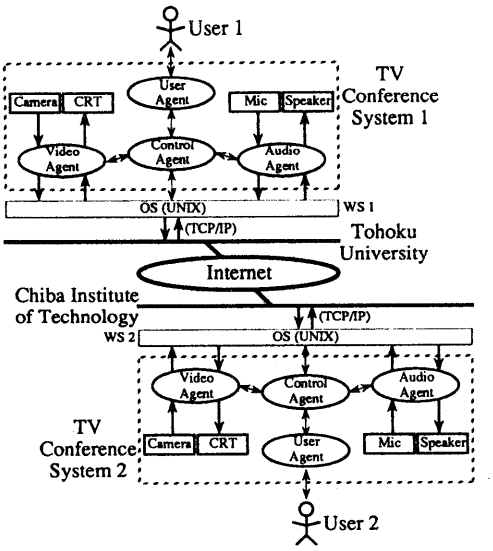

Figure 2: Outline of the Experimental System

3.1.1 Absorption of changes in the assets

Computational assets can be represented as vectors in n and m dimensional vector spaces, V_D being the one for the computational assets, V_R the one for the user requirements. We define a point $a \in V_D$ to be the computational assets that are available to the Experimental System, and the assets required by the Experiment System to implement the service as $d_g + d_p = d \in V_D$. The functionality required by the user is given by $r_g + r_p = r \in V_R$, the service actually offered by the Experimental System by $r''_g + r''_p = r'' \in V_R$.

In general, if the assets a change to $a \rightarrow a + \Delta a$, the demands d change to $d \rightarrow d + \Delta d$ the following has to hold:

$$d + \Delta d \leq a + \Delta a$$

Further, the requirements r'' are changed to:

$$r'' \rightarrow r'' + \Delta r''$$

In this case, our Experimental System changes d and r'' further to

$$\begin{cases} d + \Delta d \rightarrow d + \Delta d + \delta d \\ r'' + \Delta r'' \rightarrow r'' + \Delta r'' + \delta r'' \end{cases}$$

in order to get $r'' + \Delta r''$ as close to r as possible. However, the condition $d + \Delta d + \delta d \leq a + \Delta a$ must be obeyed.

These changes δd and $\delta r''$ are introduced automatically by the domain knowledge the system holds.

3.2 System Structure

The flexibility of our Experimental System is based on its agent architecture.

The Experimental System consists of the following four agents, as depicted in figure 2:

(a) **Control Agent** (CA)
This agent holds the domain knowledge related to the flexibility described in the previous section, and such is the part which actually exhibits flexibility.

(b) **User Agent** (UA)
This agent is the part which acquires the user requirements in a conversation with the user. The Experimental System avoids, in contrast to present systems, direct representation by numerical values. Rather, it aims at handling intelligent representations.

(c) **Video Agent** (VA)
The Video Agent is the part that offers the function to transmit video pictures. The Video Agent itself is not flexible, but it can display flexibility by cooperation with the Control Agent.

(d) **Audio Agent** (AA)
The Audio Agent offers the functionality to transmit Audio signals. Like the Video Agent, the Audio Agent posesses no flexibility itself, but can act in a flexible way when cooperating with the Control Agent.

3.3 The Primitive Agents, Organization Agents, and User Agent

The Experimental System uses the following three kinds of agents:

(a) **Primitive Agent**
This agent is in the bottom layer of the agent hierarchy, and offers a primitive service. In the Experimental System, the Video and Audio Agents are Primitive Agents.

(b) **Organization Agent**
This agent is in the layer above the Primitive Agents, and offers new functions using the functions offered by the Primitive Agents. In the Experimental System, the

Control Agent is an Organization Agent offering a TV conference by using the Video and Audio Agents.

(c) **User Agent**
This agent represents the user as an agent.

3.4 The internal agent structure

The agents used in the Experimental System can be divided into a **Base process** (Bp) and a **Meta process** (Mp).

The Bp is the part that actually executes the function of the agent. For a Primitive Agent, the Bp is a single UNIX process, while for an Organization Agent, the Bp is a collection of agents. The Bp of an User Agent if the user himself and the user interface process.

The Mp adds autonomy and cooperation to the Bp. Further, a Mp posesses the following three parts:

(a) **Domain Specifier** (Ds)
It consists of domain knowledge of the object domain, and an inference mechanism that uses this knowledge.

(b) **Co-operator** (Co)
It is a mechanism that can start the Domain Specifier from other agents, and that, depending on the Domain Specifier, cooperates with other agents.

(c) **Task Processor** (Tp)
It is a mechanism that starts from the base process the Domain Specifier, and also a mechanism that is applied to the Bp by the Domain Specifier.

3.5 Example of the execution

In this section we give a concrete example of the flexibility of the experimental system for the cases that: (1) there is an excess of network bandwidth (2) there is no excess of network bandwidth

4 Conclusion

In this paper, we defined our notion of Flexibility, as given in [4], and we have presented as an example a TV conferencing system that expressed this kind of flexibility.

Concretely, we have introduced an architecture based on agents, which expresses flexibility by absorbing changes in the computational assets and functionality.

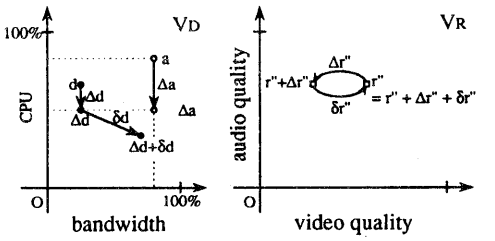

Figure 3: Example of the execution of the Experimental System (1)

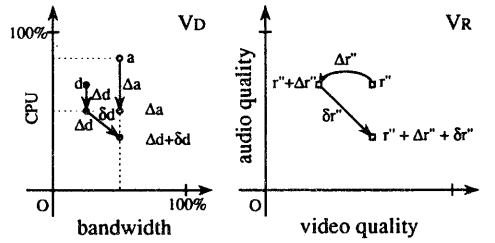

Figure 4: Example of the execution of the Experimental System (2)

References

[1] N.Shiratori, K.Sugawara, T.Kinoshita and G.Chakraborty, "Flexible Networks: Basic Concepts and Architecture", *IEICE Transactions on Communications*, Vol.E77-B, No.11, pp.1287-1294, 1994.

[2] N.Shiratori, K.Sugawara, T.Kinoshita and G.Chakraborty, "Flexible Systems: A Step Towards New Generation Networks", *Proceedings of the 9th International Conference on Information Networking*, pp.477-482, 1994.

[3] K.Sugawara, T.Kinoshita, G.Chakraborty and N.Shiratori, "Agent-Oriented Architecture for Flexible Networks", *ISADS95 (International Symposium on Autonomous Decentralized Systems)*, C2, 1995.

[4] M.Moser, S.Sugiura, S.D.Lee, K.Sugawara and N.Shiratori, "An Agent Framework for Flexible Networking", *Proceedings of the FLAIRS-95 (Florida Artificial Intelligence Research Symposium) Workshop*, Miami, 1995.