マルチメディア通信と分散処理　70−17
（１９９５．　５．　２６）

マルチメディア環境における最適かつ効果的なユーザ要求の実現

Goutam Chakraborty　唐橋拓史　白鳥則郎

東北大学電気通信研究所 / 情報科学研究科

〒980 仙台市青葉区片平 2-1-1

Phone:022-212-1439 Fax: 022-263-9848

E-mail:{gbc,garapa,norio}@shiratori.riec.tohoku.ac.jp

あらまし　情報ネットワークの高速化とともに, マルチメディアアプリケーションを支援する環境が必要とされている. 現在, マルチメディアサービスに適したネットワークプロトコルに関して多くの研究が行なわれている. 従来はアプリケーションが要求する資源は要求された分だけ割り当てていた. しかし, マルチメディアアプリケーションでは大規模なデータをリアルタイムに処理しなければならないため, 従来の方法ではユーザの期待通りのサービスを提供することは困難である. そこで, 必要に応じて動的なリソースの再配置を行ない, 場合によってはユーザ要求を再評価することで, より多くのマルチメディアアプリケーションを同時に実行させることが可能であると予想される. 本稿では, この問題を解決する手段として, メタ_アプリケーション層と呼ばれる知的なソフトウェア層を提案している. この層は各アプリケーションのリソースの利用配分を知的に最適化する.

Tailoring User's Request Optimally and Efficiently in a Multimedia Environment

G.Chakraborty T.Karahashi N.Shiratori

Research Institute of Electrical Communication
/ Graduate School of Information Sciences, Tohoku University
2-1-1 Katahira, Aoba Ku, Sendai, Japan 980
Phone: +81-22-212-1439 Fax: +81-22-263-9848
E-mail:{gbc,garapa,norio}@shiratori.riec.tohoku.ac.jp

Abstract With the increasing speed of computation and communication, new generation computers and networks are aimed to support multimedia(MM) applications in addition to their existing responsibilities. Much research efforts are presently involved in developing network protocols suitable for MM services through LAN and even Internet. The present approaches are to reserve resources a priori , depending on the application's requirements. Due to large size and time sensitiveness of data, it is usually a hard task to deliver the services to the expectations of the users. But the basic intention of the user, for the MM services, usually have a wide scope for tolerance and flexibility. If that flexibility is correctly learned and exploited, it is possible to basically satisfy the user even with marginal resources. More MM applications could be simultaneously accommodated by dynamically reallocating the resources, reevaluating users intrinsic intentions as necessary. In this paper we are proposing an additional intelligent software layer, we call it meta_application layer, to expedite this. This layer will intelligently ensure the optimum utilization of the resources.

I. Introduction

Users of network services, especially while using multimedia applications like video-conferencing, usually face disappointment due to delay, sluggishness and many other problems [2]. Still it is a fun to see pictures with mails, or listen voice over Internet. People are working hard to realize these [3]. Heavy applications over network always run under resource constraints. General feeling that with high speed computers and communication networks becoming more common, these resource constraints will not be there in future. But this is not true. More media intensive applications will come, and due to growing popularity more people will communicate jamming the network. The constraint will continue whatever speed the computer or communication system may reach. In this paper we are looking for an overall solution to this problem.

When starting an application, the user bears some intentions in mind. Application usually run at a level different from the User's intention. On the other hand, most of the time the applications run at a level of performance which is acceptable or at least tolerable by the user. Also, almost always the user's specifications are either incomplete or not very strict.

Let us consider an interactive application, the ftp service. The user's implicit requirements are: (1) the file should be transferred without any error (a strict requirement), and (2) the delay should be as small as possible (a very filexible requirement). For the delay constraint, neither the user puts some upper bound to the delay, nor he wish to hinder the file transfer by putting strict upper bound for the delay. If long delay is unavoidable due to network congestion, the user is usually ready to accept the situation.

The situation may be quite different in many other applications, where small bit errors are tolerable, but the delay constraint is stringent. Especially in case of multimedia applications there are many orthogonal as well as interdependent dimensions of requirements. The different medias have their own relative importances. In some applications audio may be more important than video, whereas it may be the reverse for other applications. Errors are tolerated usually, but not necessarily. For motion video the frame rate may vary over a wide range still satisfying the user. About many specifications the user himself is not very sure.

With this picture about service requirement, let us now turn towards the present researches in network architecture to facilitate MM communication. The isochronous continuous media like audio and video has the time dimension associated with the data stream. From the source i.e. the generation point till it reaches the presentation i.e. the sink point, the data undergoes through different interfaces using different resources like CPU, fle system, the communication network. Statistical delay, noise, jitter etc. are added to the data on their way. MM communication researches are aimed to schedule the different resources such that the data presented at the end point is still meaningful, and as per the quality specified by the user's requirement.

MM communication are typically characterized as soft real time systems demanding usually only a statistical guarantee from the resource schedulers . Also, as retransmitted packet is as good as lost, and lost or erroneous data is not a disaster, lightweight protocol without acknowledgement or retransmission is used. Missed deadlines are as well not terminal conditions. All these soft features of MM communication are usually being taken care of with the existing schedulers like meta-scheduling or orchestration layer.

But these scheduling or management schemes are hard in two ways,

1. They translate the user request to QoS requirement without any considerations about the intrinsic flexibility of it.

2. Once the schedule is complete for a session, the resources are reserved for the whole session. Non-realtime interactive or background processes are accommodated when some resources are available beyond some threshold value after scheduling the MM tasks. When a new request comes in, the scheduler tries to accommodate it without disturbing existing sessions.

Considering the following facts, we propose modifications of this scheduling scheme. User's intentions are expressible as a set of some flexible and some strict requirements. There usually is a scope of variations in actual execution of the application, still satisfying the user. At the same time it is not necessary that a session should continue at the same level of QoS throughout its life. Degradation in quality to accommodate new user, or improving of the quality due to lowering of traffic, within the range of user's flexible intention would improve the throughput of the resources in one hand, and extend MM service more widely on the other hand.

To help creating the infrastructure for the above proposal, we explore the following two fields of researches. Lots of researches are involved in dynamically creating the most efficient protocol tree for the given service, instead of the layered protocol suite, for increased efficiency. From a large set of simple protocols, a complex but efficient protocol tree is created. Underlying network architecture could also be dynamically modeled for the best overall performance and efficiently using the available resources. Thus lightweight communication protocol could be implemented whenever that suffices the communication needs. Similarly, when a poor quality picture is satisfactory to the user, a fast and high compression protocol in the presentation layer may be used to save the channel band width as well as CPU cycles.

In addition to the above, researches on how to schedule the system resources to satisfy the QoS needs of MM applications would also serve as a basis. These two could serve as the basis or lower level structure for our architecture. On top of these two, we add an intelligent interface to generate the optimum QoS parameters for an application, taking account of the softness of user's request as well as the dynamically changing status of the resources. With every new session request or change from a running session, there would be rescheduling including the existing sessions.

The rest of this paper is organized as follows. Section 2 reviews some of the proposed architectures and features for the network to accomplish multimedia communication. In section 3 we present some important factors to be considered for multimedia communication that are lacking in the present network architecture. Finally in section 4 we introduce our proposed protocol, the meta_application layer. The structure and working of the proposed expert system is explained in the two subsections of section 4. Section 5 discusses some concluding remarks.

II. Network Architectures for multimedia communication

Applications running on computers can primarily be classified into time insensitive and time sensitive (interactive, real time) applications. Also most of the present day applications need some communication support, as they either use a distributed file system, a LAN or may be the whole Internet. As more and more applications are loaded to the system and the network, the resources become more and more constrained. The conventional operating system and the conventional Internet protocol supporting point to point best effort service is inadequate for this constrained situation. With increase in number of applications, the resource share per application goes down and running of some applications become meaningless. One consideration is very important: Some applications are sensitive to QoS, and they need at least some basic level of QoS for sensible performance. There are other important aspects like multicast associated with multimedia communication [6]. But this is not the main concern of this paper.

New architectures and service models are coming up to accommodate these new application requirements of QoS. There has been a widespread agreement that to accommodate the new multimedia applications the following capabilities are essential.

- Flow Specification: Before some application flood the network with its voluminous data, the source should specify to the network the traffic characteristics of the data it will deliver. Also it would specify the service requirements of the application. Once the network agrees to service this flow specifications, the application could be launched.
- Resource reservation: Once the application hands over the flow specification to the network, network should check that the corresponding required resources, like bandwidth and buffer, are available or not. If available it should reserve those resources for that application, and confirm the same to the application. These resources are reserved till the end of the session to ensure undegraded service during the session.
- Admission Control: Not all flow specifications requested by potential applications could be accommodated. Admission control is the algorithm which determines, which flow specification to admit and which one to deny.

Researchers are working on these different aspects. We argue that this approach, though efficient as performance is concerned, puts lots of restrictions on the applications. We are going to discuss the lacuna of the above approach in details in the next section.

III. Flexible architecture and its requirements

In the last section we have discussed about the present trend of the researchers to modify the network architecture to successfully accommodate the multimedia applications. We, on the other hand want to emphasize that, we are forgetting many important features while proposing the above network modifications. We list some of those important factors as follows:

- It is felt that there is a big scope of flexibility between the intention of the user, and to the level to which the application should run. As proposed in the flow specification, the traffic and service characteristics should be specified in the beginning to the network, and to ensure uninterrupted service the network would reserve those resources till the end of the session. But this is not to the best of all the users. Instead of reserving big chunks of the resources on a first-come-first-serve basis, if the allocation could dynamically be changed, it could bring better justice to greater number of users. We should keep in mind that, though to service the user's requests some system and network resources are demanded, they are in general very flexible in nature.

That flexibility should be exploited, as the load on the network varies.

- Usually many of these new multimedia applications involve more than one user. The service needs resources at all the end users locations. Grabbing the moving picture at a high rate, and transmitting it to others would be a meaningless wastage of computation power and communication resources, if at the other end due to lack of necessary resources those data could not be processed. To avoid such a situation the participating ends should negotiate to fix an optimum level of the application.
- In the last section, by resource reservation and admission control, the level of a particular application was fixed. But it may so happen, that during the progress of the session, the user need to change or extend his requirement. For example, a session started with only audio may ask for still picture transmission. Or due to decrease in load, the network could be able to deliver better service to an existing session. Thus, it is required that the network architecture should have dynamic interactions with the users' intentions, whenever there is a change in the whole system, so that the application could always run at a level which is optimum as far as the system resources and users' intentions are concerned.

We propose in this paper the inclusion of a layer of protocol, above the application layer, to map user's intention to proper applications and protocols of the network. This mapping or construction mechanism will affect all the subsequent layers of the protocol right from the application layer. The service level of the application, the compression technique in presentation layer, even the communication mode, reliable or unreliable, all would be intelligently and dynamically set. We call this the meta_application layer. The general outline of the working of this layer is described in the following section.

IV. Meta-application layer

Before describing the working of our proposed meta-application layer, we will introduce some simple definitions for clear expressions in the subsequent parts.

Definition 4.1 Let Σ denotes the set of system resources. This includes hardwares (CPU and file system) and softwares (including protocol functions) resources. These resources are mainly involved in executing the four layers, from application to transport, of the 7-layer OSI protocol. Thus

$$\Sigma = \{\sigma_1, \sigma_2, .., \sigma_s, ..\}$$

Here, σ_ss denote the different components of Σ.

Each component σ_s has two parts $< name\_ \sigma_s, value\_ \sigma_s >$. The *name* part identifies the component, and the *value*, represented by a real or boolean number shows the actual level of the resource, or denote whether a resource is available or not.

The information about the static as well as dynamic part of the Σ is available with the meta_application layer.

Definition 4.2 Similar to system resources, Ω denotes the set of network resources. This includes resources involved in the execution of the three layers of protocols from network to physical, of the 7-layer OSI protocol. This demarcation is not very important though, and the reasons are indicated in the conclusion. Exactly as above, Ω have components denoted as

$$\Omega = \{\omega_1, \omega_2, .., \omega_o, ...\}$$

As earlier, each ω_i has two parts, the name part and the value part as $< name\_ \omega_i, value\_ \omega_i >$. $< name\_ \omega_i >$ identifies the component, and $< value\_ \omega_i >$ denotes the availability of that component, denoted by a real number or boolean.

Definition 4.3 The set of requirements from the users is denoted by

$$\Re = \{\Lambda_1, \Lambda_2, .., \Lambda_i, ...\}$$

where Λ_i denotes requirement from user i. Now this user requirement Λ_i may have several components like audio, video etc. We denote them as λs. Therefore,

$$\Lambda_i = \{\lambda_{i1}, \lambda_{i2}, ..., \lambda_{ij}..\}$$

As we have discussed, Λ_is and therefore the λ_{ij}s are usually flexibly defined i.e. the user's intentions are not very strict or even clear. To run a particular Λ_i, we have to map different λ_{ij}s to the proper σ_ss and ω_os. Or, in other words, we have to set up proper combination of σ_ss and ω_os. It should be done optimally for the present situation. If some changes occur, like arrival of a new service request or closing of a session, the mapping of Λ_i to σ_ss and ω_os is to be reconsidered for the maximum utilization of the resources to the benefit of the maximum number of users.

Definition 4.4 When applications are running, they use some of the system and network resources. For actual execution of Λ_i, the involved system resources are denoted as $\{\rho_1^i, \rho_2^i, ..., \rho_r^i, ..\}$, and the network resources as $\{\nu_1^i, \nu_2^i, ..., \nu_n^i, ..\}$. Obviously, for every instant of time, we can write

$$\sum_{i=all\ users} \rho_r^i \le \sigma_r$$

and similarly,

$$\sum_{i=all\ users} \nu_n^i \leq \omega_n$$

We propose that, with the occurrence of any change, there should be consideration for a rearrangement of these allocations. This would be done at the meta_application layer, for the optimum utilization of the resources, and satisfaction to the maximum number of users.

Definition 4.5 We now define the state $\Im$ as

$$\Im = < \Sigma, \Omega, \Re >$$

The three components of $\Im$ express some constraints. They are either from the scarcity of the resources or constraints set by the user i.e. demands for resources for the intended service. Satisfying all the constraints set up by the three terms, our problem is how to find the proper set of ρs and νs.

Suppose initially there is no user application running, and all the resources are actively waiting, and suppose we call it $\Im_o$. The state $\Im$ changes as and when new service requests come, or there is some change in the available resources. The protocol of the meta_application layer finds the corresponding optimum values for ρs and νs. Then allow the application to run at that level.

This transition of state may even be due to some user wanting to change his/her present service level. As an example, an user may like to upgrade the audio level from speech to CD quality. Less likely causes are the failure of some route or node or hardware. Whatever it be, meta_application layer protocol takes care of that to maintain a graceful service level.

As the most frequent changes are expected from $\Re$, and they are most flexible in character, in the next section we discuss them in more details with examples. Also we propose a framework for the design of this meta_application layer.

A. Expressing Flexibility in User's Requirement

We will first discuss how the user's requirements could be expressed such that their flexibility could be described.

Users requirements are in general not so strict. We begin with some examples and then express them more formally.

Let us again consider the simple case of *ftp* service. The transmission error should be nil, and the delay up to certain point, say 3 mins is completely acceptable. Let also suppose that delay beyond 15 mins is unacceptable. It can be expressed formally as,

```
(definition ftp;
    (error = 0)
    (delay : (0mins  1) (3mins  1) (15mins  0))
)
```

The second line of the *ftp* definition says that error should be 0. The third line defines the acceptable delay. Different values of the delay comes with a factor, we call it acceptability factor. Delay of $0 mins$ has acceptability factor of 1. $3 mins$ delay also have acceptability factor 1. When the delay is $15 mins$ or more, the acceptability goes to zero. This acceptability factor is used to express the fuzziness of the user's requirements. The acceptability factor could change linearly, or according to some other continuous or discontinuous function. It could always be presentable with some expressions, though it may be a little more complicated than the above.

As another example, suppose for a particular service, the user is ready to spent some particular amount of money. He says that up to $2/min is completely acceptable, and beyond $5/min is completely unacceptable. This can be expressed as,

```
(definition my_service;
    (error = ...)
    :      (cost ($0/min  1) ($2/min  1) ($5/min  0))
)
```

A little more intricated requirement, like band width for audio communication. Similar to the above, it can be expressed as,

```
(definition audio;
    (bad       (16  .5) (24  1);
     good      (32  .5) (64  1);
     excellent (64  .5) (96  1);
    )
)
```

Depending on the type of requirement, many varieties of expressions could be possible, and they could be formed in a nested fashion for compound requirements. The detail is not of importance for this present paper. A large set of those definitions would be present as a knowledge base in the meta_application layer. It could serve as default values when the user only empirically mentions his requirements, like "I need good quality sound". In fact at many occasions the user is not even conversant of how to precisely describe his intentions. The expert user, on the other hand, should be allowed to specify his requirements more precisely. It should also be possible that the user could upgrade his/her requirement during the running of the session e.g. want to play a CD record to his/her friend instead of just talking.

As the user requirement, the system and network resources could also be expressed, and stored as facts in the meta_application layer. They are usually more crisp than fuzzy, and more static in nature. But these facts also may go through changes, for example when there is a failure.

B. Working of Meta_Application Layer: States and transitions

As stated earlier the state $\Im$ is but a collection of facts. Some of the facts describe the system resources Σ, some describe the network resources Ω, and the rest describe the users requirements. A few empirical examples could be as follows:

$\Lambda_i := audio = good.$ %User i wants audio at good level
$LINK(X,Y) := Band\ Width = 1Gbps$ %1Gbps Link
% exist between nodes X and Y

There are several such facts in a state, and a change or transition of state means there is a change in the facts. Mostly the changes would arise from the users' requests, but not necessarily.

In every state, the facts actually describe some resource constraints and users' demands. Corresponding to every such state, there is an optimum level of distribution of resources to the different applications. When the state changes, these allocations usually need to be rearranged. Due to this state transition we may have to change the level of running of a service, but within the limits specified by the user's intentions. Thus, until a session finishes, the original facts for the users specifications should be retained. While changing the state, and therefore reallocating the resources, some rules have to be satisfied. Some of the simple rules would look like:

$$Total\ BW\ allocated\ between\ nodes\ X\ and\ Y \leq BW\ available\ between\ X\ and\ Y$$

$$Cost\ of\ different\ services\ offered\ to\ an\ User \leq affordable\ cost\ as\ declared\ by\ the\ user$$

There may be more complicated rules with $if-then$ constructs, and rules may come with some *certainty* factors (fuzzy rules). The detail of those are more implementational and beyond the scope of the present paper.

Summerizing the above discussions, it is now evident that the proposed meta_application layer would be an expert system. With every change it would try to figure out optimally the allocation of the resources to different applications. For some change it may fail to get a solution satisfying all the rules. It may reject the change (in case of some service request) or it may declare emergency (in case of failures). This uppermost intelligent layer, implemented by an expert system, would have the overall control of the network.

V. Conclusion

In this paper we have described the idea of the meta_application layer to dynamically allocate network resources for optimum utilization. It is felt that there is a long way between this idea and the actual implementation of it. But at the same time it is true that many of the important ingredients are already there contributed by earlier researchers [5] [4].

It is true that when there is a change, there may be small interruption in a running session. Frequent of such would be disastrous and should be avoided.

Though there is no real reason behind it, we have separated the system and network resources. The idea comes from the implementation considerations. For a step by step implementation, we can start with system resources first, and in the next step could include the network resources too.

References

[1] Andrew S. Tanenbaum, "Computer Networks", Prentice Hall, 1989.

[2] Ronald J. Vetter, "Videoconferencing on the Internet", IEEE Computer, Vol.28, No.1, pp.77-79, Jan 1995.

[3] Michael R. Macedonia and Donald P. Brutzman, "MBone provides Audio and Video Across the Internet", IEEE Computer, Vol. 27, No. 4, pp.30-36, April 1994.

[4] N. C. Hutchinson, and L. L. Peterson, "The *x*-Kernel: An architecture for implementing network protocols", IEEE transactions on Software Engineering, 17(1):64-76, Jan. 1991.

[5] Sean W. O'Malley, and Larry L. L. Peterson, "A Dynamic Network Architecture", ACM Transactions on Computer Systems, 10(2):110-143, May 1992.

[6] C. Pornavalai, G. Chakraborty, and N. Shiratori, "Multicast routing for multimedia Using neural ntwork", IEICE technical report, January 1995.

[7] N. Shiratori, K. Sugawara, T. Kinoshita, and G. Chakraborty, "Flexible Networks: Basic Concept and Architecture" IEICE Trans. Inf. & Syst., Nov 1994. (Japan)