マルチメディア通信と分散処理 70−16
（１９９５．５．２６）

An Algorithm for Flexible Networking

M. Moser, K. Sugawara†, N. Shiratori

Tohoku University
RIEC, Department of Computer Science
Shiratori Laboratory
2-1-1 Katahira, Aoba-ku
Sendai 980, Japan

†**Chiba Institute of Technology**
Department of Computer Science
Sugawara Laboratory
2-17-1 Tsudanuma
Narashino 275, Japan

Abstract

Many of today's applications, especially applications using a computer network, cannot deal with changes in the user requirements or available computational assets. Due to the first problem, users are limited to the functionality offered by the system as is, rather than being able to combine the elements of a service as they desire. The second problem leads applications to degrade disgracefully if the availability of assets changes while the application is running.

In this paper we first analyze both problems and their interconnection, then give a unifying theoretical framework to deal with changes in the user requirements and computational assets. Based on this framework we define a flexibility criterion for applications. We then propose an agent framework, the Flexible Networking Kernel, that allows users to configure a service out of a set of elementary services, and an algorithm that, in the face of changing user requirements, supplies the best service possible with the currently available computational assets.

やわらかいネットワーキングのためのアルゴリズム

M. モーザー, K. 菅原†, N. 白鳥

東北大学
電気通信研究所／情報科学研究科、白鳥研究室
〒９８０仙台市青葉区、片平２－１－１

†千葉工業大学
情報工学科、菅原研究室
〒２７５習志野、津田沼２－１７－１

概要

現在の多くのアプリケーション、特にコンピュータネットワークを用いたアプリケーションは、ユーザ要求の変化や利用可能なコンピュータ資源の変動を扱うことはできない。第一の問題より、ユーザは自分の望むように基本的サービスを変更することはできず、システムの提供する機能をそのまま使うしかない。また第二の問題のため、アプリケーションの実行中に利用できる資源が変化すればその質はひどく低下する。

本論文では、まずこの二つの問題と両者間の関係を分析し、ついでユーザ要求やコンピュータ資源の変化を扱う統一的な理論的枠組を与える。この枠組に基づきアプリケーションのやわらかさの基準を定義する。次にユーザが基本的なサービスからサービスをつくることができるようなエージェントによる枠組、すなわち Flexible Networking Kernel と、ユーザ要求の変化に対してその時点で利用可能なコンピュータ資源を用いてできるだけ最良のサービスを提供するアルゴリズムを提案する。

1 Introduction

With the spread of powerful work stations and network connections, demand for networked real-time services is increasing. Real-time services, however, require special software support to guarantee the availability of computational assets, but although those mechanisms have been developed, transition to the new solutions seems slow and difficult due to the size and inertia of the user community.

Soft real-time services are real-time services that allow for some decrease in the quality of service, and thus can be provided without computational asset guarantees. However, lack of computational assets may cause the quality of service to decrease, and in existing systems the decrease is rather ungraceful. In this paper we develop a model of a networked service that relates the users' requirements towards the service with the amount of computational assets required to supply the service. We then give an algorithm that provides the best service possible with the computational assets available by gracefully reducing the user requirements and selecting the most suitable implementation for the service requested by the user.

2 A Model

In the following we develop a model for a multi-user service held over a computer network. There are p participants, each equipped with a regular work station, and the same software is available at all p work stations. We assume there is no real-time support, neither by the operating system nor the networking software, and the work station is not dedicated, i.e. the service is sharing the computational assets of the work station with other processes.

2.1 Participant Model

Every participant **P** can be described by the following tuple:

$$\mathbf{P} = \langle a, D_1, \ldots, D_p, r_1, \ldots, r_p, L, P \rangle$$

The computational assets of a user's work station can be measured in orthogonal terms such as CPU cycles, core memory, transmission bandwidth, transmission delay, etc., which span a n dimensional vector space V_A. The computational assets a currently available mark a point in this vector space:

$$a \in V_A$$

The variables D_i represent data streams arriving at the work station, with D_1 being produced by the local peripherals, and $D_2, \ldots, D_p$ being the data streams arriving from the other participants. These data streams D_i consist of s *elementary data streams* D_i^j:

$$D_i = \bigoplus_{j=1}^{s} D_i^j$$

The data stream D_1 generated by the local periphery should be transformed to the data stream D_0, which consists of the elementary data streams D_0^j, and transmitted to the other participants as described by the requirements r_1, while the data streams $D_2, \ldots, D_p$ coming in from the other participants should be presented to the user as described by the requirements $r_2, \ldots, r_p$.

Similar to the computational assets, both data and requirements can be described as points in a m dimensional vector space V_D with orthogonal dimensions such as frames/sec., pixel resolution, number of channels, sampling rate, etc, all of which are assumed to be discrete.

$$D_i, r_i \in V_D$$

Further, $L = \langle l, u \rangle, l, u \in V_D$ describe upper and lower limits for the requirements in each dimension, thus bounding a subsection of V_D. The priorities $P \in \mathbb{R}^m$ eventually provide priorities for the axes of V_D. Both L and P will be described in more detail in subsection 2.3 which is concerned with our notion of flexibility.

2.2 From the requirements to the implementation

The service S can be decomposed into a collection of s *elementary services* E_j such as

audio, video, a whiteboard, etc., one for each elementary data stream D_i^j of data stream D_i:

$$S = \{E_j | j \in [1, \ldots, s]\}$$

Correspondingly, the vector space V_D can be decomposed into a set of s orthogonal subvector spaces V_{Di}, one for each elementary service, with the axes of each V_{Di} being a subset of the axes of V_D:

$$V_D = \bigoplus_{i=1}^{s} V_{Di}, V_{Di} \perp V_{Dj}, i \neq j$$

Consequently, the requirements r_i decompose into orthogonal vector sums consisting of *elementary requirements*, one for each elementary service E_j:

$$r_i = \bigoplus_{j=1}^{s} R_i^j, R_i^j \in V_{Di}, R_i^j \perp R_i^k, j \neq k$$

Each elementary service implements the elementary requirements R_i^j expressible in the subvector space V_{Dj}, consuming a part D_i^j of the data stream D_i. The elementary services consuming a part D_1^j of data stream D_1 generate the part D_0^j of data stream D_0. Thus, the *requested service* S consuming the data streams $D_1, \ldots, D_p$, and generating the data stream D_0, is described by the requirements $R = \langle r_1, \ldots, r_p, L, P \rangle$.

Each elementary service E_j consists of a set of implementations I_{jk} which realize the R_i^j:

$$E_j = \{I_{jk} | j \in [1, \ldots, s], k \in \mathbb{N}\}$$

Out of each set E_j one implementation $I_{jk(j)}$ is chosen to provide the requested service. The amount $d_{jk(j)}^i$ of computational assets consumed by an implementation $I_{jk(j)}$ to realize the elementary requirements R_i^j can be written as a function $f_{jk(j)}$ of R_i^j and $I_{jk(j)}$:

$$d_{jk(j)}^i = f_{jk(j)}(R_i^j, I_{jk(j)})$$

Now, the overall computational assets d required to implement the requested service S can be computed as:

$$d(r_i) = \sum_{i=1}^{p} \sum_{j=1}^{|S|} d_{jk(j)}^i, k(j) \in [1, \ldots |E_j|]$$

The requested service can be provided while the incoming data streams D_i and the computational assets available suffice, i.e. while the following conditions hold:

$$\begin{aligned} d(r_i) &\leq a \\ D_i &= r_i, i \in [1, \ldots, p] \end{aligned}$$

We will refer to the requested service as *satiated* if the above conditions are satisfied.

2.3 Flexibility

Next to satiation, a service may S be in three other states:

hungry The incoming data does not suffice to satisfy the user requirements.

overfed More data is coming in than is needed to satisfy the user requirements.

suffering The computational assets do not suffice to process the data as requested.

When the service is hungry, some of its elementary services are hungry. The hungry elementary services could try to extrapolate the missing samples. However, no general extrapolation mechanism can be provided, as the extrapolation algorithm depends strongly on the kind of data involved. Therefore, this issue will not be explored within this paper. Rather, the implementations I_{jk} of the hungry elementary services reduce the requirements R_i^j to fit the incoming data D_i^j by means of a function δ_{jk}:

$$R'^j_i = \delta_{jk}(R_i^j, D_i^j), R'^j_i \in V_D$$

When the service is overfed, the service may either provide a higher quality of service than requested, or it may prune the incoming data to fit the amount required. As pruning can be implemented in a rather straightforward way, e.g. by dropping samples, and may happen *on the fly* while processing the data, we will, for the sake of simplicity only treat the second method in this paper. We can such assume that the implementations I_{jk} each provide a function π_{jk} yielding the desired amount D'^j_i of data:

$$D'^j_i = \pi_{jk}(R_i^j, D_i^j), D'^j_i \in V_D$$

A requested service suffers if either the user requirements are increased, or the availability of computational assets decreases to an extent that the condition $d(r_i) \leq a$ is violated. Now there are two possibilities: On the one hand there may be a different choice of implementations $I_{jk'(j)}$ for the same requirements R_i^j, which provide a satiated service, and switching to them absorbs the change. On the other hand, the requirements may just be too high to be satisfied with the computational assets a available. In the latter case the requirements r_i must be reduced to some lower requirements r_i'' that are in some way optimal. The optimal requirements r_i'' can be characterized as the one those *closest* to the original requirements r_i, for which the condition $d(r_i'') \leq a$ holds.

When defining the distance measure required to specify the closest requirements, two additional features of the user requirements should be considered. The features are the two remaining elements L and P of the user requirements: Firstly, some requirements may be more important to the user than others, and thus should not be violated unless unavoidable. It is therefore preferable to reduce the quality of service first in direction of the lower priorized dimensions, as specified by the priorities P. Secondly, the user specified limitations L on the requirements should be respected. Lowering the quality of service in the i-th dimension below the lower limit l_i may render the according elementary service useless for the user. If, due to the lack of computational assets, the requirements must be reduced beyond this point it is preferable to drop the corresponding elementary service completely to avoid wasting computational assets.

The following weighted vector length accounts for both priorities and limitations:

$$\|v, L, P\| := \sqrt{\sum_{i=1}^{m} (p_i \cdot \sigma(v, L))^2}, v \in V_D$$

The priorities p_i amplify the contribution of the higher priorized dimensions, while σ is a threshold function that returns 0, if v violates any of the limitations on the dimenstions of the subvector space V_{Dj} the i-th dimension belongs to, or v_i otherwise. Thus σ accounts for the fact that the realization of an elementary service E_j is useless, if any of the requirements the elementary service has to satisfy is below its lower bound.

We can now give the conditions the reduced user requirements r_i'', which are produced by a *reduction function* ρ, should satisfy:

$$\begin{aligned} \|r_i - r_i''\| &= min., \\ d(r_i'') &\leq a; \\ r_i'' &= \rho(r_i', L, P, a), \\ r_i' &= \bigoplus_{j=1}^{s} R_i'^j, \\ R_i'^j &= \delta_{jk}(R_i^j, D_i^j) \end{aligned}$$

Provided we hold the reduced requirements r_i'' that can be implemented with the available computational assets, they can be decomposed into the elementary requirements R''^j_i. The selected implementations $I_{jk(j)}$ of the elementary services can then apply the function π_{jk} to obtain the required amount D'^j_i of data from the data streams D_i^j.

A real system may be less flexible that the ideal system, i.e. for certain elementary requirements R'^j_i there is no implementation. Consequently, the reduction function ϱ of the real system may return a result that is different from the r_i'' yielded by the reduction function ρ of the ideal system. To measure the degree φ of flexibility of a real system we introduce the degree φ of flexibility:

$$\begin{aligned} \varphi &= \frac{1}{l} \sum_{\forall r \in L} \frac{\Delta(r_i, \varrho(r, L, P, a))}{\Delta(r_i, \rho(r, L, P, a))} \\ \Delta(r, s) &= \frac{\|r - s\|}{\|r\|} \end{aligned}$$

The requirements r run through all l points in the subspace of V_D that is bounded by the limits $L = \langle l, u \rangle$. Obviously, φ is equal to one, i.e. the system is ideally flexible, if ϱ is equal to ρ within the subspace of V_D. The flexibility decreases with the number of requirements yielding different results, and also with the amount the results of ϱ and ρ differ.

2.4 An Algorithm

The function ϱ can be implemented as a variant of the well-known knapsack problem[4]: The size of the knapsack is the amount of available computational assets a. The things to be placed in the knapsack are the implementations I_{jk} that satisfy the elementary requirements R_i^j, their value is the weighted length $\|R_i^j\|$ of the requirement they satisfy, and their weight the amount d_{jk}^i computational assets they consume. In contrast to the standard knapsack problem the implementations are grouped according to the elementary service they belong to, and at most p implementations of each group, i.e. one for every incoming data stream D_i may be placed in the knapsack. Further, the same implementation may be used more than one time.

Theorem 1 *Suppose there exists an implementation I_{jk} for all reduced elementary requirements R_i^j within the subspace bounded by L. Then the above described algorithm computes the ideal reduction function ρ, i.e. $\varrho = \rho$.*

3 Related Work

Recently, there is strong interest in Japan in developing what is called a Flexible Network[15]. Although it is not yet sufficiently clear what exactly a Flexible Network should be there seems to be an emerging agreement that it should be an extension of the current network that is more user friendly and softer with respect to temporary and permanent changes in the network behavior. The starting point for the work presented in this paper was provided by Shiratori et al.[12], who identify the following three reasons for the necessity of Flexible Systems:

1. changes in the representation of user requirements,
2. temporary changes in the availability of computational assets, and
3. permanent changes in the availability of computational assets.

Further, a general definition of a Flexible System is presented in [12]. In this paper we focused on the second of the three aspects listed above, building on the body of existing work, and extending our previous proposal[8].

There are various mechanism that support real-time services ranging from special hardware, operating systems or real-time network protocols, see e.g. [7][9][16][1]. However, transition to these solutions requires to replace the according hard- or software, a process that nowadays is difficult and slow, if not impossible. Although the Internet had been regarded as hostile to real-time services[10], as it cannot provide any performance guarantees, a class of real-time services, so-called *play-back applications,* which can deal with loose performance guarantees, has been identified. The basic idea of play-back applications is to add some extra delay by buffering the data coming in from the network. If the transmission delay of the network increases, interruption of the play-back is avoided by using the data in the buffer. It is believed that a large class of future real-time services are play-back applications[3]. DECspin was one of the first multimedia services that was run successfully on the Internet without any real-time support. *Vat* is seen as the first adaptive application[10], which measures network delay and dynamically adapts to it[11]. However, it has been found that the performance of both services suffers when other computational expensive operations are running. According to our own experiences the same is true for the IVS system[2], which applies a sophisticated feedback mechanism to adapt to network load and to avoid resource overconsumption. We think that our scheme of a Flexible System can improve this situation significantly.

Although some systems reduce the amount of the transmitted information by adaptive coding[6], and some of the available systems allow to employ different coding and compression mechanisms[11], none of them changes the mechanisms automatically in face of lacking computational assets. Rather, changes in the transmission format, although recognized automatically by the receiver, have to

be initiated by the user. Further, the influence of other computational assets other than network throughput and delay on the quality of service are hardly ever considered. To our best knowledge this paper for the first time proposes an integrated approach to adaptive resource monitoring and distribution that exploits the time-space tradeoff, which provides a means for load shift and controlled degradation of the quality of service.

References

[1] Anderson, D.P., Herrtwich, R.G., Schaefer, C., *SRP: A resource reservation protocol for guaranteed performance communication in the Internet,* Berkeley Technical Report TR-90-006, February 1990

[2] Bolot, J.C., Turletti, T., *A rate control mechanism for packet video in the Internet,* Proceedings of the IEEE Infocom 1994

[3] Clark, D.D., Shenker, S., Zhang, L., *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism,* Proceedings of the ACM SIGCOMM '92, Baltimore, August 1992, pp. 14-26

[4] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to Algorithms,* The MIT Press, Massachusetts, London

[5] Cypser, R.J., *Communications for Cooperating Systems,* Addison-Wesley Publishing Company

[6] Huitema, C., Turletti, T., *Software codecs and work station video conferences,* INRIA Technical Report, November 1993

[7] Jeffay, K., Stone, D.L., Smith, F.D., *Transport and display mechanisms for multimedia conferencing across packet-switched networks,* Computer networks and ISDN systems, Vol. 26, No. 10, 1994

[8] Moser, M., Sugiura, S., Lee, S.D., Sugawara, K., Shiratori, N., *An Agent Framework for Flexible Networking,* Proceedings of the FLAIRS-95 Workshop, Miami, April 1995

[9] Northcutt, J.D., Clark, R.K., Maynard, D.P., Trull, J.E., *Decentralized Real-Time Scheduling,* RADC Technical Report RADC-TR-90-182, Carnegie-Mellon University, 1990

[10] Partridge, C., *Gigabit Networking,* Addision-Wesley Publishing Company, Reading, Massachusetts, 1993

[11] Sasse, M.A., Bilting, U., Schulz, C.D., Turletti, T., *Remote Seminars through Multimedia Conferencing: Experiences with the MICE project,* Proceedings of the INET '94

[12] Shiratori, N., Sugawara, K., Kinoshita, T., Chakraborty, G., *Flexible Networks: Basic Concept and Architecture,* IEICE Transactions on Communication, E77-B(11), 1994

[13] Shiratori, N., Sugawara, K., Kinoshita, T., Chakraborty, G., *Flexible Systems: A Step Towards New Generation Networks,* Proc. of the 9th Intern. Conf. on Information Networking, 477–482, 1994.

[14] Sugawara, K., Kinoshita, T., Chakraborty, G., Shiratori, N., *Agent-Oriented Architecture for Flexible Networks,* Proc. of the 2nd Intern. Symp. on Autonomous Decentralized Systems, April 25–27, 1995, Phoenix, Arizona (to appear)

[15] Tominaga, H., *Flexible Networks — Introduction,* The Journal of IEICE (Special Issue on Flexible Networks, in Japanese), Vol. 77, No. 4, 1994

[16] Zhang, L., Deering, S.E., Estrin, D., Shenker, S., Zappala, D., *RSVP: A New Resource ReSerVation Protocol,* IEEE Network Magazine, Vol. 9, No. 5, September 1993