NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Synchronization Studies in Multimedia Communications

Li Li

A THESIS

submitted to the School of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Ottawa-Carleton Institute of Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

University of Ottawa

OTTAWA, ONTARIO, K1N 6N5

ISBN 0-315-93595-2

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To my parents and my teachers

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Li Li

I further authorize the University of Ottawa to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Li Li

Acknowledgments

First, I would like to thank both my supervisors Dr.N.D.Georganas and Dr.A.Karmouch for introducing me to the exciting field of Multimedia Communications. I am most grateful to their support and encouragement during my thesis work. I enjoyed the benefits of being a member in the Multimedia Communications Research Laboratory (MCRLab) at University of Ottawa.

I would also like to thank all the past and current students of Room B-409, Colonel By of University of Ottawa, for making my stay at this university a memorable one. My special thanks are due to Dr.Man Li, Dr.Anil Gupta and Dr.Sudhakar Ganti with whom I had good technical discussions. My thanks are also due to the past and current members of Room A505, CBY MCRLab, who have kindly helped me in my research, specially to Dominique Richens, Louise Lamont and Grant Henderson.

I would like to give my thanks to all the support staff members of Electrical Engineering for their help, to Amanda Lauzon, Lucette Lepage, Michèle Roy and Suzanne St-Michel.

I am truly grateful to my beloved Robert and my family for their consistent support, without which this work would not have been possible.

Abstract

Synchronization is considered as a key issue in distributed multimedia systems. Synchronization problems exist in both multimedia presentational and conversational applications. In a multimedia presentation, two basic synchronization issues are involved. Temporal Presentation Control (TPC) and Simultaneous Real-Time Data Delivery (SRTDD). TPC deals with the ordering of the data objects in the presentation. SRTDD refers to simultaneously delivering multimedia data in different data streams belonging to the same time interval. In multimedia conversational applications, e.g., in cooperative applications, coordination control and causal ordering of the events give another synchronization problem. This thesis investigates the synchronization control in the distributed multimedia systems. All the above synchronization issues are covered. The temporal model and a synchronization algorithm for the TPC are proposed, which can support the real-time multimedia teleorchestration when multiple independent information sources are involved. The temporal model handles the relative and imprecise temporal requirements among the data objects in the "fuzzy" presentation scenario, and supports the synchronization dialogue among the independent sources. Thus the synchronization algorithm can be performed in a distributed manner. For the SRTDD control, an entire control architecture is presented which follows the layered communications model. We have introduced the multimedia session management, multimedia segment delivery, etc., to achieve the simultaneous data delivery. The Synchronization Quality of Service (SQoS) parameters are first applied for the SRTDD control over the ATM network. Meanwhile, taking the multimedia joint editing, one of the popular multimedia collaborative applications, as an example, we have developed the coordination control and the causal ordering algorithms for the joint editing session. Then in the second part of this thesis, we have studied the analytical performance modeling for distributed multimedia systems where the synchronization is considered. Stochastic Petri Nets (SPN) are applied to model the different multimedia synchronization systems. We developed a computational algorithm for a class of SPNs, i.e., the LBSPNs (Locally Balanced Stochastic Petri Nets), which have product form solutions. Applying the computational algorithm for LBSPN, the performance analysis of a joint editing system is conducted.

Contents

List of Figures

List of Tables

List of Simbols

$\vec{n}_x$: Interval vector of object X;
N: Set of all the intervals in a scenario;
N_x: Interval node for object X;
Δ_x: Vector node model of interval $\vec{n}_x$;
Δ_N: A set of nodes for the interval vectors;
$\delta_x^{(p)}, \delta_x^{(f)}$: δ node(s) of interval $\vec{n}_x$;
$\mathcal{D}(X)$: Display duratoin of object X;
$\mathcal{P}(N_x)$: Nodes preceded N_x in the time-flow graph model;
$\mathcal{S}(N_x)$: Nodes succeed N_x in the time-flow graph model;
$\mathcal{E}(\{\vec{n}_x\})$: The set of intervals that succeed interval $\vec{n}_x$;
AC_i: The i-th activity occurrence in scenario;
Λ_i^p: Parallel subset of AC_i;
Λ_i^s: Sequential subset of AC_i;
REF_i^c: Reference set of AC_i;
t_{nt}: Network delay of a voice packet;
$p_{nt}(t)$: Probability density function(p.d.f) of t_{nt};
t_{it}: Intention delay of a voice packet;
t_{dt}: Total delay of a packet;
t_{iv}: Intention delay of a video frame;
t_{sv}: The system delay of a frame, $t_{sv} = t_{nv} + t_{iv}$;
t_{dv}: Total delay of a frame;
$t_{dv,n}$: Total delay of the n-th frame;
$p_x(y)$: p.d.f.of the random variable y_x;
$P_x(y)$: Distribution function of the random variable y_x;
T_x: The constant value for the random variable of t_x or just a constant;
P_i: Place i in SPN;
t_i: Transition i in SPN;
μ_i: Marking of SPN;
μ_0: Initial marking of SPN;
$\hat{\mu}_i^j$: a Basic Subnet j Marking in LBSPN;
μ_i^j: Separate Subnet j Marking in LBSPN;

Chapter 1

Introduction

1.1 Background

Recently, as the field of computer communications has become well developed, application requirements for faster networks and more efficient devices are bringing us ultra high-speed broadband networks, powerful table top computers and multimedia equipment. This evolution of broadband telecommunication networks and multimedia technologies (video, optical storage [43] [8]) has significantly contributed to the emergence of new multimedia applications. Multimedia electronic mail, video conferencing systems and other kinds of more advanced multimedia applications, such as cooperative design, joint editing, etc., are coming into our everyday life.

In general, multimedia applications are designed to handle various types of information such as text, graphics, image, audio and video. The application can be a local application, such as local multimedia database retrieval [43]. The more attractive multimedia applications are usually remote applications, carried out in a distributed system over the networks. Real-time video conferencing [8][17], for instance, allows a group of users to interact through their workstations or terminals over the network. Users may have cameras and monitors associated with their workstations. At any time, participants can display the pictures of all the participants on their monitors. Meanwhile, the participants can exchange files, graphics, images and other types of data, while talking on the

phone. Queries to the distributed multimedia database may also be performed. One of the users may, for instance, show a picture to the others while describing it with a telepointer. The electronic work-place — an organization-wide system that integrates various information processing and communication activities, gives another example of the interesting distributed multimedia applications. The study of such systems is growing into a new multidisciplinary field: Computer-Supported Cooperative Work (CSCW) [12]. As an example, joint editing is one of the popular CSCW applications that are being developed [8][12][17]. In joint editing, a group of users will cooperate to produce a multimedia document. One of the users, for instance, may paste a picture while another is editing a text portion. Other types of data such as video, audio (Hi-Fi music), voice etc. may also be involved in the editing. Distributed multimedia applications include both presentational and conversational applications. For instance, browsing multimedia news retrieved from a multimedia database is a typical presentational application. A multimedia conferencing system provides a multimedia conversational application, where the multimedia document is taken as the information vehicle, instead of only having voice in the conversation. The conversational applications always involve the presentations of multimedia data.

Distributed multimedia applications can be categorized into two classes: real-time and store-and-forward applications. From the communications point of view, real-time applications are those that require information to be transmitted in continuous mode from one location to another (real-time delivery). An example of such application is the video conferencing system introduced above [17]. In this application, the information exchanged among the users or between the multimedia database and the users is transmitted on the network and displayed on the workstations at almost the same time. We notice that all conversational applications are real-time. On the other hand, store-and-forward applications by nature are not real-time applications. An example of such application is the multimedia mailing system where the documents are transferred from one location to another in store-and-forward mode. At the receiver side, the message is stored in the mail-box until the user is ready to read it.

The trend in developing multimedia systems is to ensure that they support real-time

distributed applications [33][28]. Real-time distributed multimedia applications require a high performance distributed environment that will meet the overall new multimedia requirements in terms of processing, integration, storage and communications. The basic element of a distributed multimedia system is multimedia data integration. In multimedia data integration, information items (data objects) of various types and origin are composed as well as transferred over the communication network, for the purpose of presentation, storage, and communication. One significant issue raised in multimedia data integration is that the temporal relationships involved in the data objects, which are required by the multimedia application itself, have to be considered. Therefore, real-time multimedia applications often require a distributed computing system to order and schedule the delivery of the data objects in real-time, in order to maintain a desired service quality, for instance, the quality of the display of the data objects in an integrated multimedia document. Meanwhile, in order to support the transmission of time-dependent multimedia data, real-time communication protocols are essential. It is a challenge for the existing communication networks (e.g., FDDI, ATM Networks), protocols and standards (e.g., OSI etc.) to be extended and adapted to support the wide range of time-constrained multimedia applications.

Basically, one of the requirements of any system supporting a distributed multimedia application is the need to provide synchronization of data elements in multimedia data integration over the network. As a matter of fact, in the effort of developing multimedia distributed application systems, media synchronization is not only a considerable problem but also one of the most important elements of the whole system. This chapter states the different facets of the synchronization problems involved in various multimedia application systems. In Chapter 2 we will review the synchronization control schemes that have appeared in the literature.

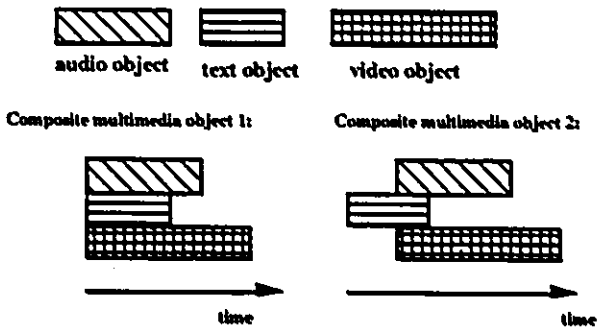


Figure 1.1: Data object and composite multimedia object

1.2 Multimedia Synchronization Requirements

Multimedia synchronization requirements arise from the multimedia applications. As mentioned above, we have presentational and conversational applications, which are carried out in the distributed multimedia system in real-time. In both applications, the multimedia presentation is required. In a distributed system, multimedia presentation involves the integration of the data in various media.

In a multimedia system, an information item in mono-media is referred as a data object [29]. For example, one page of a text file and a segment of audio can both be data objects. In multimedia data integration, several data objects of different media are combined to form a new multimedia object, which is a composite data object. In Fig. 1.1, three data objects and their composite multimedia data objects are shown. The data object is taken as the basic data unit in the multimedia system. In multimedia data integration, the temporal relationship among the data objects, which is required by the composite multimedia object, has to be handled. In Fig. 1.1, for instance, two different composite objects result from the different temporal relationships implemented among the three objects during the integration. This temporal relationship is called a scenario. For instance, in multimedia document retrieval, a multimedia file, which is a composite multimedia object, will be retrieved from the database. Fig. 1.2 presents the scenario of this composite object which consists of several data objects. When the multimedia file

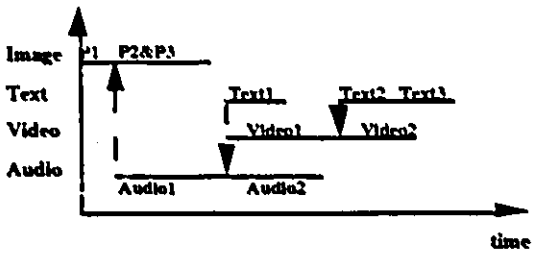


Figure 1.2: An example of a multimedia scenario

is being represented, e.g., displayed on the screen of the viewer, the data objects should be presented in the temporal order which is required by the scenario. Such requirements can be the sequential or simultaneous representation of the data objects contained. For example, from Fig. 1.2, we can see that the image object $P1$ is supposed to be displayed first. Then, while the image object $P2$ follows $P1$, the audio object *Audio*1 should be displayed simultaneously with $P2$ and then $P3$. During the presentation, the guarantee of the temporal ordering of the data objects is essential to the quality of the application.

In order to handle the temporal relationship of the data objects in presentation, synchronization of the different objects is required. The temporal relationship needs to be specified and maintained during the presentation. The Temporal Presentation Control (TPC) and the Simultaneous Real-Time Data Delivery (SRTDD) are designed for this purpose.

Temporal Presentation Control (TPC) is concerned with the temporal ordering of the data objects in the multimedia presentation. For instance, the set of multimedia data objects which are involved in the presentation scenario of Fig. 1.2 and delivered by the distributed database servers should be ordered properly for the presentation. In Shepherd et al. [40], TPC was named "event driven synchronization" and declared as one of the basic synchronization controls in multimedia. In [29], this kind of synchronization is referred as "synthetic synchronization". As addressed in [42], synthetic synchronization should ensure that at presentation time, the various pieces of information (segments) must be properly ordered and synchronized in time. Since this type of synchronization actually arises

directly from the temporal presentation requirements, it is named here TPC (Temporal Presentation Control). TPC is regarded as one of the basic synchronization issues involved in the distributed multimedia system. As an application, or more precisely a "scenario driven" synchronization control, TPC is directly related to the temporal requirements in the data presentation. TPC deals with the ordering of data objects in a distributed multimedia system, to conduct every phase of the multimedia presentation in conformity with the involved temporal relationships which are required by the presentation scenario.

When the multimedia data are transferred on the network in a distributed multimedia system, because of the random network delays, synchronization control is required at the transport level to maintain the temporal relationship among the data delivered. Not only the ordering of the data objects ought to be preserved, but also the coupling among the data streams should be constructed according to the presentation requirements. Especially, simultaneous playback of two or even more data streams needs to be implemented. For instance, in the 'lip-sync" example, the lip movements should "match" the play-out of the speech. In a teleconference window, the position of the tele-pointer needs to match the voice description to keep the application valid. While the sequential ordering of the data delivered can always be kept by the connection-oriented transport system, this simultaneous temporal relationship needs a special consideration in real-time multimedia data delivery. Thus, the communication networks supporting real-time distributed multimedia applications differ from the traditional data communication networks in terms of the real-time synchronization control in the multimedia data delivery. That is, the network has to be able to manage the Simultaneous Real-Time Data Delivery (SRTDD). Shepherd et al. [40], named this type of synchronization control as "continuous synchronization", while Little and Ghafoor [29] defined the "live synchronization" for it. Simultaneous Real-Time Data Delivery (SRTDD), also a basic issue of the multimedia synchronizations, refers to the simultaneous delivery of multimedia data in different data streams belonging to the same time interval.

The studies reported for the multimedia synchronization control are mostly focused on maintaining the validity as well as the quality of the multimedia presentation. However,

another synchronization issue has also been brought into attention [38][47]. This type of the synchronization problem, namely, the coordination and the event causal ordering in a cooperative multimedia application system, is encountered in the conversational applications. For instance, in joint editing, the multimedia data objects contained in a document are shared by a group of users over the network. Coordination is required among the distributed users for concurrent accessing of the shared objects. Meanwhile, the user's manipulation of the multimedia document in the shared window should be consistently "seen" by all the participants on their workstations, irrespective of their locations in the system and the random network conditions. If participant B, for instance, makes some modification after B has observed the changes made by another participant A, the editions supplied from A and B should be presented in the correct and same order to all the participants' on the network. Thus, causal ordering of the events is required in a multimedia cooperative system. Notice the concept of "ordering" here is different from that of the data objects in the TPC for the multimedia presentation.

Event ordering problem in distributed computing systems, including the logic clocks, the partial and total ordering of events, have been well studied [19][4]. However, the multimedia cooperative system has introduced new requirements concerning the causal ordering in the distributed system. The concurrency degree needed in a cooperative system varies in different applications. For instance, in joint editing, mutual exclusive accessing of the shared document is not favored [17][12]. The application needs to allow the concurrent document manipulation from different participants [17]. At the same time, "conflicting", where two or more participants are changing the same portion of the document concurrently and thus generating different versions at the different participants' sites, should be prevented. For example, after every manipulation conducted by a participant in the system, the result ought to be delivered and displayed to every other participant before it can be changed again. Thus, the basic concepts and methods developed in distributed computing have to be adapted and applied, to cope with the problem of coordination and causal ordering in the new multimedia cooperative applications.

1.3 Overview, Structure of The Thesis and Contributions Made

In this thesis, we begin by investigating the Temporal Presentation Control (TPC) scheme for the multimedia presentation control. We establish a new temporal model for the specification of the "fuzzy" multimedia scenarios in the distributed system where only the relative temporal knowledge is available. Based on this model, a distributed scheduling algorithm is devised for TPC which can conduct a synchronization dialogue among multiple independent information sources (e.g., distributed database servers) to form a high quality remote multimedia presentation in real-time. Some interesting applications such as remote multimedia document creation, multimedia telcorchestra with independent sources, etc., can thus been supported [35][26]. Therefore, our first contribution in this thesis is the distributed TPC scheme designed to support the multimedia telcorchestration with independent sources.

In order to present the temporal relationship among the data objects at the receiver, TPC requires a real-time transport system with synchronization control over the multimedia data streams. In this thesis, we propose an entire synchronization control scheme for real-time multimedia delivery, which is embedded in a layered communication architecture. This is our second contribution, the Simultaneous Real-Time Data Delivery (SRTDD) scheme for multimedia transmission. In this scheme, the Synchronization Quality of Service (SQoS) parameters are first applied to ensure the presentation quality required by the application. Meanwhile, the recovery algorithm in our SRTDD system can diminish the synchronization errors during the network fluctuations.

The third contribution made in this thesis are the distributed algorithms for the coordination and causal ordering control in CSCW applications (e.g., joint editing). In this part of our work, Lamport's clock system [19] was adapted and applied to support the multimedia collaboration in a shared space over the network. Without tracking the document history, a consistent version of the shared multimedia document is guaranteed at all the participants' workstations during the concurrent editings. A WISIWYS (What

I See Is What You See) system is thus achieved.

The second part of this thesis is devoted to the performance modeling of the distributed multimedia application systems where synchronization is considered. This is our fourth contribution. We apply the Generalized Stochastic Petri Nets (GSPN) models for the performance evaluation of the remote multimedia document creation system and the real-time data delivery system. A class of product form GSPNs, the Locally Balanced Stochastic Petri Nets (LBSPN), which were studied by M. Li and Georganas [51], are applied to model the multimedia cooperative system. We present a computational algorithm developed for the LBSPN models which is our fifth contribution. The computational algorithm takes the approach of the convolution of the subnets. Finally, we employ this computational algorithm, which was programmed in C language on the UNIX OS to solve the LBSPN models for the CSCW application of joint editing.

This thesis is then, divided into two parts. In the first part, we present the multimedia synchronization scheme that we have developed. The second part deals with issues of performance modeling and evaluation for the system. In part I, Chapter 2 reviews the synchronization control methods that have been reported in the literature. In Chapter 3, the temporal models for the multimedia scenarios and the TPC algorithms are presented. Chapter 4 addresses the SRTDD system designed for the multimedia data delivery. The synchronization models and algorithms for the multimedia causal ordering in multimedia cooperative applications are discussed in Chapter 5.

In part II, Chapter 6 gives a review on the performance modeling techniques and examines their modeling power for multimedia systems, where the synchronization is involved. In Chapter 7, we conduct performance modeling for some of the multimedia synchronization systems, using the GSPN models. The LBSPN models and the computational algorithm are presented in Chapter 8. Then the computational algorithm is employed to solve the LBSPN models of the joint editing system. Finally, in Chapter 9, we conclude by summarizing the contributions of this thesis and suggesting further research topics.

1.4 List of Publications Arising from This Research

a) In Refereed Journals

1. L. Li, A. Karmouch and N. D. Georganas, "Real-Time Synchronization Control in Multimedia Distributed Systems" *ACM Computer Communication Review*, Vol.22, No.3, July, 1992, pp79-86.

2. L. Li, A. Karmouch and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 — Temporal modeling of Collaborative Multimedia Scenarios", *ACM Journal of Multimedia Systems*, Feb., 1994.

3. L. Li, A. Karmouch and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 2 — Synchronization Algorithms", *ACM Journal of Multimedia Systems*, Feb., 1994.

b) In Refereed Conferencing Proceedings

1. L. Li, A. Karmouch and N. D. Georganas, "Performance modeling of Distributed Data Integration in Real-time Multimedia Systems", *Proc. ICCC (International Council for Computer Communications) Multimedia Communications'93* , Banff, Canada, April, 1993.

2. L. Li, A. Karmouch and N. D. Georganas, "Synchronization in Real-Time Multimedia Data Delivery," *Proc. IEEE ICC'92*, Chicago, U.S., June, 1992.

3. L. Li, A. Karmouch and N. D. Georganas, "Real Time Synchronization Control in Multimedia Distributed Systems" *Proc. IEEE Multimedia'92*, Monterey, U.S.A. April, 1992.

4. L. Li, L. Lamount, A. Karmouch and N. D. Georganas,"A Distributed Synchronization Control Scheme in a Group-oriented Multimedia Conferencing System". *Proc. 2nd International Conferencing on Broadband Island'93*, Athens, Greece, June, 1993.

5. L. Li, A. Karmouch and N. D. Georganas, "Multimedia Segment Delivery Scheme and Its Performance for Real-time Synchronization Control", *Proc. IEEE ICC'94*. New Orleans, U.S., May, 1994.

Part I

Synchronization Control in Multimedia Communications

Chapter 2

Review

As mentioned in Chapter 1, multimedia synchronization is needed to ensure both the temporal presentation and the event causal ordering in the distributed multimedia system [41][32][42]. In this chapter, we first review the reported synchronization control techniques for multimedia presentations. Then we also cover the reported coordination control and causal ordering schemes for multimedia cooperative systems.

2.1 Multimedia Temporal Presentation Control

We first give an application example of remote document creation to explain the basic ideas of Temporal Presentation Control (TPC), which deals with the temporal ordering of multimedia objects in the presentation. In the application of remote document creation, a user creates a multimedia document using the data objects stored in a distributed system. Data objects are acquired from the information sources. The information sources can be the distributed multimedia database servers or some other users who possess the particular data. The data objects required are to be structured to form a new multimedia document according to the scenario designed by the user. From the distributed sources, the newly created multimedia document may then be retrieved and delivered over the network for display on the viewer's workstations. For instance, a designer or a group of designers can create new models and send them to a group of viewers for either a discussion or a final

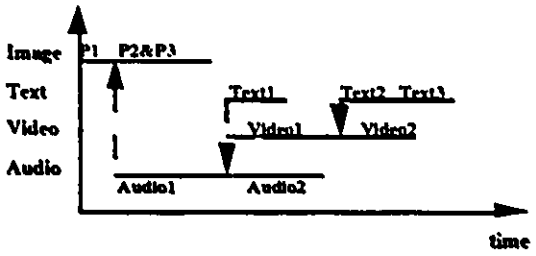


Figure 2.1: Example of a multimedia scenario

decision [12][17].

Consider the scenario in Fig. 2.1 as that of the document edited by a user. In order to create the document, the user sends the scenario to the distributed multimedia information sources to retrieve the relevant data objects. When the data objects are combined to form the multimedia document, the temporal relationships required in the scenario should be represented. Therefore, in order to form the multimedia temporal presentation, the synchronization control includes two basic parts: the definition of the temporal relationships among audio, video and other data objects, and the delivery of the data in accordance with these relationships. Thus, in this section of the TPC (Temporal Presentation Control) review, we start with the specification methods for the temporal relationships among the multimedia objects.

2.1.1 Temporal Relationships of Data Objects in Presentation

In order to handle the temporal relationship of the data objects in presentation, the specification as well as the modelling of multimedia composition with respect to intermedia timing is required first. Some relevant work has already been reported in [34][29].

In [29], Little proposed a technique for the formal specification and modeling of the temporal relationships involved in the multimedia integration. The proposed model is based on logical temporal intervals and Timed Petri Nets. In the following, we briefly describe this method [29].

Temporal relationships between the media may be implied, as in the simultaneous

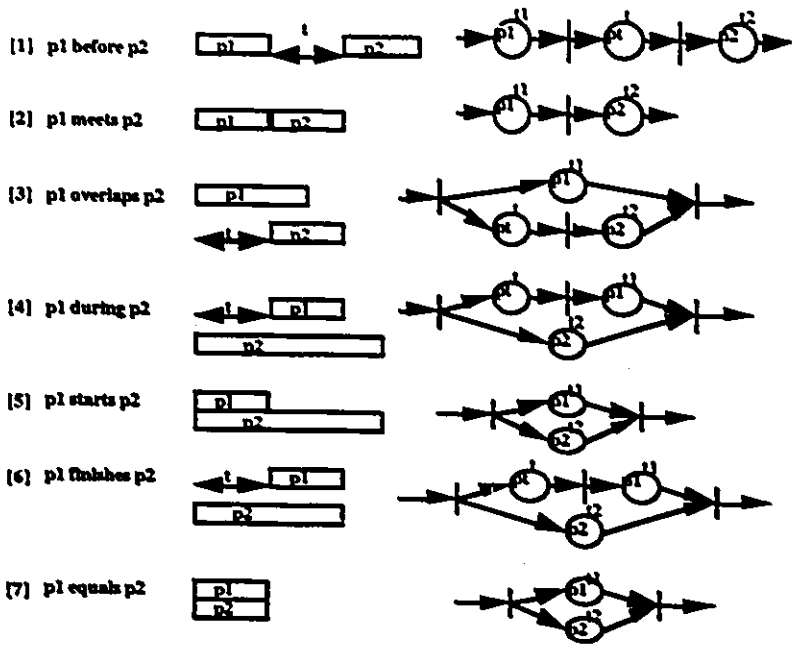


Figure 2.2: Basic temporal relationships and their OCPNs

acquisition of voice and video, or may be explicitly formulated, as in the case of a multimedia document which possesses voice annotated text. In either situation, the characteristics of and relationships among media have to be established.

Little believes that given any two atomic processes specified by their temporal intervals, there exists a Petri Net representation for their relationship in time [29]. This Petri Net presentation was named Object Composition Petri Net (OCPN). The temporal interval here is characterized as a nonzero duration of time in any set of units, for example, "a week" or " 100 ms". The Petri Nets for seven basic temporal relations were also given, as shown in Fig. 2.2. In Fig. 2.3, an example is illustrated, using the OCPN to describe the temporal ordering in a presentation. The OCPN model captures the basic temporal relations that exist between every two intervals. Meanwhile, the interval based OCPN models can well support a TPC scheduling algorithm [28]. In [28], the OCPN model was applied by Little to design an algorithm that indicates the retrieval time of data objects from the database in a manner which preserves the temporal requirements of the initial specification [29][28]. Then, if the specification represented by the OCPN is made accord-

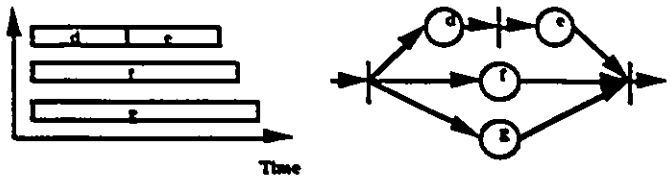


Figure 2.3: An example of the OCPN model

ing to the temporal relationships of the data objects required in the presentation, these data objects should be displayed in conformity with the temporal requirements.

As defined in [29][28], in an OCPN model, every place must be associated with a parameter of the time duration ($t1$, $t2$ or t in Fig. 2.2). These parameters are used in the TPC scheduling algorithm. Among the seven basic temporal relationships presented by OCPNs (see Fig. 2.2), the relationship of "$p1$ during $p2$" is distinguished from the relationship of "$p1$ finishes $p2$" only by the different values of the durations associated with the place pt (different values of the t in the Fig. 2.2). Similarly is the relationship "$p1$ starts $p2$" from the relationship "$p1$ equals $p2$". Therefore, in the OCPN models, all the temporal interval lengths have to be known *a priori*, to ensure that the OCPN represents the temporal relationships correctly. Without the precise value of the time duration parameter for every place, the OCPN model could turn into a misinterpretation of a given timeline scenario.

In many distributed applications, the parameters required to form the OCPN cannot be supplied by the user or by the application itself. For instance, in the above example of multimedia document creation, when the user is creating a document that contains two video stories, the user may desire an audio segment to accompany the second video story. However, the user does not have to know how long the first video story will last. Then the user cannot specify the time interval of the first video story in the scenario. In this case, it is impossible to establish an OCPN model which has the required time parameter associated with every place directly from the application's scenario.

In [42], this problem was addressed as "relative synchronization operations" and

"absolute synchronization operations". For example, "present segment A at time T" is an absolute synchronization operation. On the other hand, "present segment A simultaneously with (or after) segment B" is a relative synchronization operation. In other words, time parameters are specified in the requirement of an absolute synchronization operation, while the relative synchronization operation is described only by certain reference without explicit time parameters in terms of the starting time, ending time or durations. Obviously, all the relative synchronization operations have to be converted internally by the system into the absolute synchronization operations in order that the relative synchronization operations can be performed. In [31], this converting was referred as a "synchronization dialogue" for the evaluation of the temporal specifications of the objects. However, OCPN models do not support the synchronization dialogue but are the results of the dialogue, because they cannot describe the relative synchronization requirements. In order to support the synchronization dialogue, the specification model has to be able to describe the relative synchronization operations.

In [34][45], another type of synchronization model was developed for the presentation of multimedia objects. Instead of "interval", "synchronization points" were used in these models as the primitive to describe the presentation sequence. Compared to OCPN, the "point based" models represent relatively rich temporal relationships that can be involved in multimedia presentation [34][45], including some relative synchronization requirements. The point based models do not require precise duration information. However, the model introduced in [34] did not take the interval durations into account, neither did it deal with the issues of deadline inference and object scheduling. The model in [45][46] focused on the user's interface and the file editing system, and was not designed for the synchronization control in a distributed environment.

2.1.2 Temporal Presentation Control Algorithms

In order to implement a real-time Temporal Presentation Control (TPC), scheduling of multimedia data delivery has been proposed [28][41]. In [28], an algorithm was established on the OCPN model. This algorithm comprises two parts, named "Serialized-Net Algo-

rithm" and "Resource-Decomposition Algorithm". The serialized-net algorithm generates a playout schedule as the set of deadlines for playout of data elements specified for each OCPN. After the serialization is completed, a playout schedule can be decomposed into subschedules for each information source.

As mentioned before, OCPN requires all the temporal interval lengths to be known *a priori*. In the serialized-net algorithm the time schedules for the data objects are scheduled through the parameter t associated with every place in the OCPN. Therefore, the scheduling algorithms proposed on OCPN models do not include the synchronization dialogue mentioned above.

The scheduling algorithms were performed in a centralized manner [31], whether at a destination workstation or at an intermediate node. The scheduling centre has to know whether an information source (e.g., a database server) is congested at a given time or whether the data object required is locked by another application (e.g., for concurrency control in joint editing [17]). Meanwhile, from the system's point of view, the centre can become the bottleneck of the whole system. Therefore, in a distributed multimedia application system, a centralized scheduling algorithm is neither sufficient nor robust.

2.2 Synchronization and Simultaneous Data Delivery

2.2.1 Temporal Relationship among Multimedia Data Streams

In real-time distributed multimedia system, when the data objects are delivered onto the communication network, the temporal relationships have to be maintained during the transmission in order to realize a real-time data presentation at the receiver side. Therefore, synchronization is also required in multimedia transmission.

In order for the communication system to manage the synchronization control in multimedia transmission, the temporal relationships involved in the data transferred also have to be specified. However this case is not the same as that at the data object level.

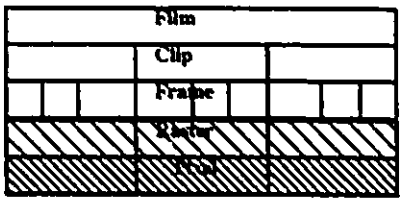


Figure 2.4: Granularity of Media: LDUs

Instead, what we have here are data streams transferred on the network. There are continuous media (CM) streams such as voice data stream or video data stream, as well as discrete media (DM) streams such as computer I/O data [42]. The concept of Logical Data Unit (LDU) was introduced to the CM streams in [42] as the "granularity" of a media stream in the synchronization control. Thus, CM streams are viewed as a sequence of LDUs. Each kind of medium has an associated LDU. For instance, the LDU of a video stream can either be a single frame or a well-defined sequence of frames (Fig. 2.4). The temporal relationships among the LDUs of different media can thus be described.

At the transport level, the way to describe the temporal relationship among the CM and DM streams has been investigated in [39][42]. One of the ideas proposed is based on transferring the multimedia data streams in parallel [39], i.e. to have a separate logical virtual connection for each medium. Parallel transmission on multiple virtual connections, which apply different means for data in different media depending on their various characteristics, would meet the real-time application requirements. Meanwhile, this scheme makes it possible to map the temporal relationships required by the application more directly onto the data streams in the multimedia transport system. Thus, the temporal relationships among the data objects will be represented as the temporal relationships involved in the multiple data streams which are transferred on multiple virtual connections. For instance, in Fig. 2.5(a), the text object $t1$ is required to be delivered at the same time with that of the audio object $a2$. Then, in Fig. 2.5(b), the same simultaneous relationship is maintained on the two connections.

In [42], the only temporal relationships of CM and DM streams studied is the "tight

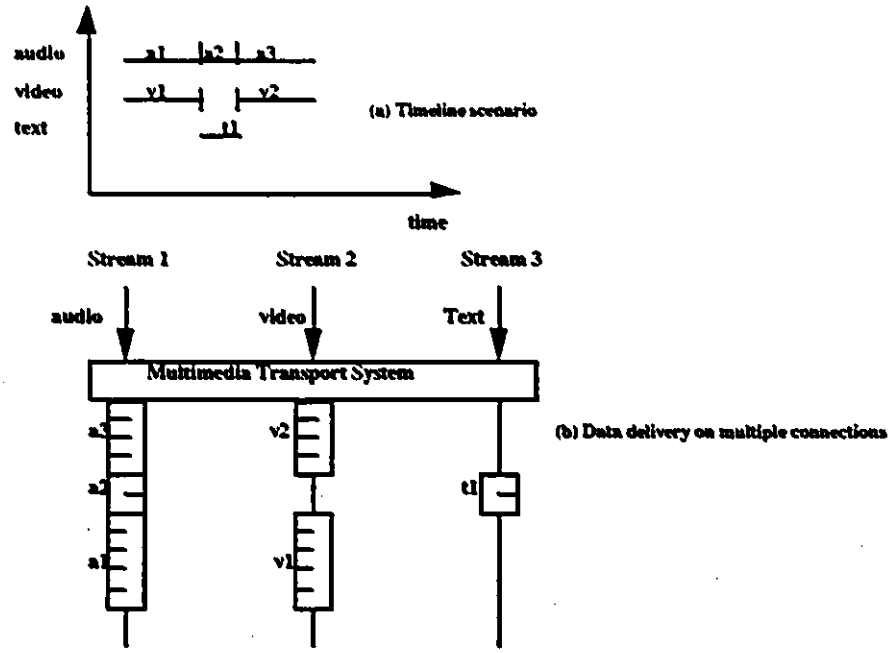


Figure 2 5: Mapping the temporal relationships onto multiple data streams

synchronization", i.e. the tight matching of different data streams. This is the "simultaneous" relationship among media streams. As we addressed in Chapter 1, "simultaneous" relationships among data streams on different channels is the most significant aspect that differentiates the multimedia data transport system from the traditional mono-media data transport system. If the data transport system can guarantee the simultaneous data delivery on multiple channels, that is, if the reference of the matching points (e.g. flags, marks etc.) in multiple data streams transferred on different channels can be preserved, all the various temporal orderings of data on different channels can be implemented. For instance, in Fig. 2.6, the data streams on channel $c1$ and $c2$ are required to be simultaneously delivered in order to be displayed at the same time at the receiver. The data stream on channel $c3$, on the other hand, is required to be delivered right after the data delivery in channel $c1$ and $c2$ is finished. If, the matching of data on channel $c1$ and $c2$ at the reference (marking point) A, and the matching of channel $c3$ data with channel $c2$ data at reference B can be achieved, the temporal relationships involved in the three data

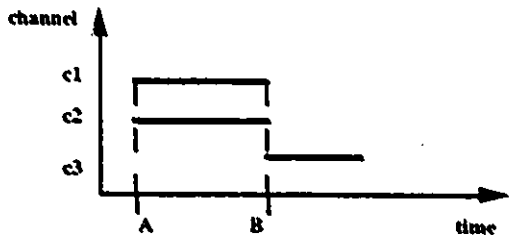


Figure 2.6: Tight synchronization

streams will be preserved. Meanwhile, the ordering of the data delivery on every channel should be maintained as it is in connection-oriented data networks.

2.2.2 Simultaneous Real-Time Data Delivery

Basically, one of the main reasons that gives rise to the synchronization errors (i.e., the mismatching) in the multimedia data delivery is the random delay experienced by the data streams on the networks. Since multimedia data streams of different types are characterized very differently on the networks and have various network and delivery requirements, the random delays introduced by the networks to the multiple data streams are difficult to predict. An example is shown in Fig. 2.7 to illustrate the basic ideas of simultaneous real-time data delivery. In Fig. 2.7, three data streams are delivered to the network and transferred in parallel in different virtual channels. Because of the random network delays experienced, the data within the same time interval SE (marked S_1E_1, S_2E_2 & S_1E_1 in three streams) may arrive at the receiver at different time ($S_1'E_1'$, $S_2'E_2'$ & $S_3'E_3'$). If one of the data streams is a user's voice and another is the video picture of his face, the voice and the movements of the user's lips may not match anymore when displayed at the receiver. Thus, Simultaneous Real-Time Data Delivery (SRTDD) control is required in the network protocols in order to maintain the service quality.

Besides the requirement of "matching", in every single medium data stream, the continuity of the data streams is required as well. Because of the random network delay, the continuity of the data stream will be interrupted. As shown in Fig. 2.8, the random

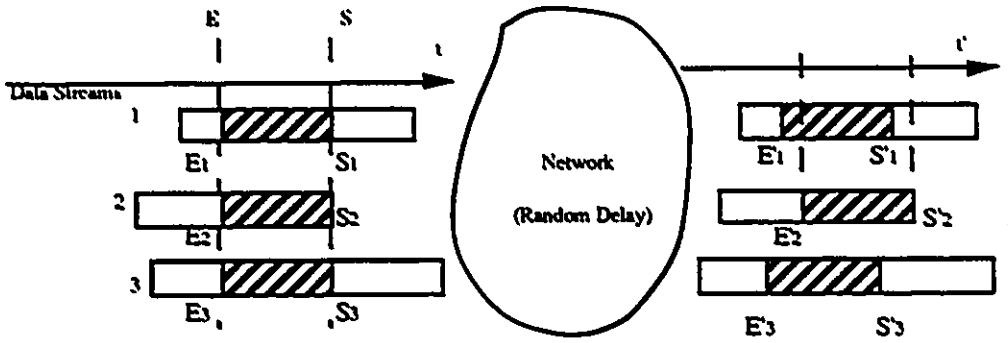


Figure 2.7: Synchronization loss: mismatching

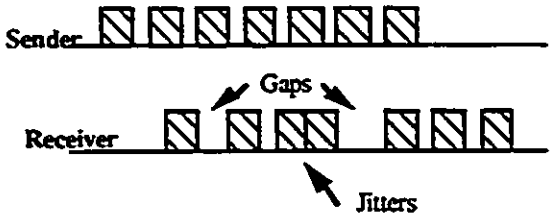


Figure 2.8: Synchronization loss: discontinuity

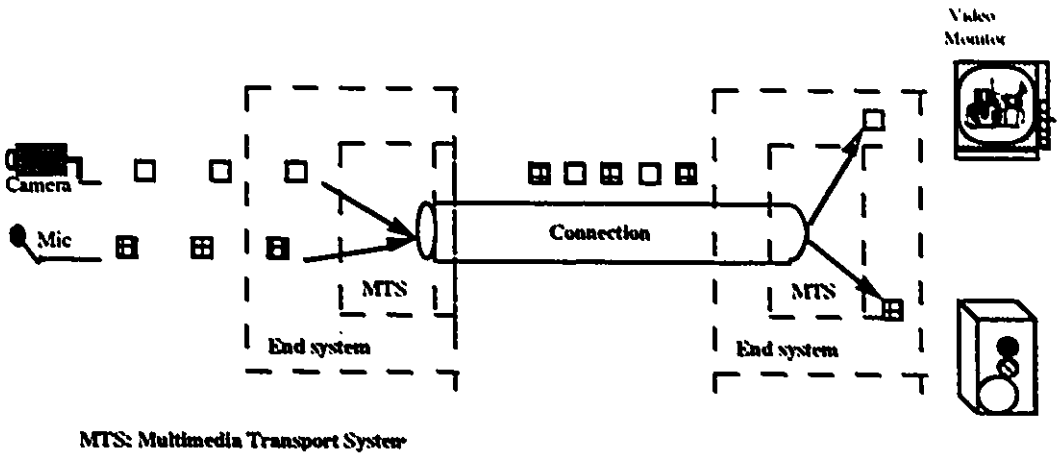


Figure 2.9: Transmitting multimedia data on a single connection

network delay introduces gaps and jitters in the data stream during transmission. These gaps and jitters destroy the display quality at the receiver. This continuity problem exists in all kinds of packet switching networks.

Some work has already been reported for solving the problem of SRTDD [30][41] [40] [20]. Simultaneous real-time data delivery is essentially a problem of "rate matching" among multimedia data streams. A very straight-forward approach is to avoid this matching problem. Recall that the synchronization unit of multimedia data streams transferred on the network was referred as LDU, as addressed before. Then, the LDUs of different data streams can be interleaved into one composite LDU stream, as illustrated in Fig. 2.9 [42]. These combined LDUs are time-stamped at the source and separated at the sink and presented according to the time stamp. As the multimedia data which are supposed to match with each other are interleaved and transferred on a single connection, the LDUs that need to be simultaneously delivered for presentation may arrive at the receiver at almost the same time. In this case certain recovery still has to be applied at the receiver to filter out gaps and smooth jitters.

However, when multimedia data are transferred on broadband ISDN, such as ATM (Asychronous Transfer Mode) networks, this approach faces difficulties in implementation. In ATM, a connection is assigned to a certain class of traffic, for example, a voice

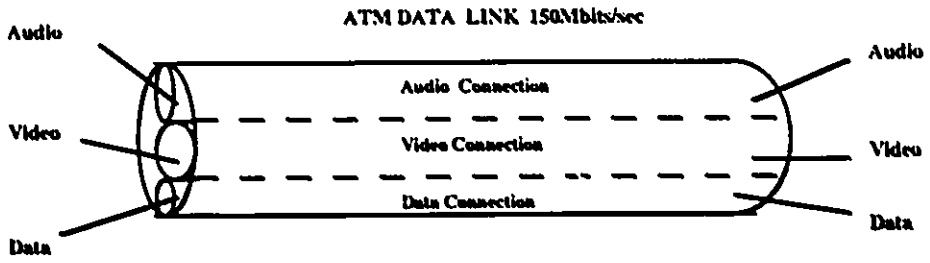


Figure 2.10: Multimedia data on ATM networks

connection or a video connection etc., as shown in Fig. 2.10. For efficient utilization of the bandwidth, as well as for flow and traffic control on the ATM network, connections are managed according to the traffic characteristics, such as the average peak rate, the burst length of the source, etc. Meanwhile, the QoS (Quality of Service) parameters required by every connection, such as the tolerable cell transfer delay, cell loss and the smoothing requirement have to be supported by the underlying ATM network. Different data streams have very different characteristics. For instance, voice data are very sensitive to the response time but can tolerate certain data transmission errors. On the other hand, video data are very sensitive to transmission errors and also require a very high throughput. Data of different media by nature cannot possibly be described by one set of traffic parameters and located on one ATM connection. In any distributed multimedia application, data streams can originate from sources at different sites. For example, they may be delivered from distributed servers. Then, it will not be possible to put all the data streams on one connection.

When multiple data streams are transferred on different connections, the matching among the data streams (i.e., among the different virtual connections) becomes a problem that has to be solved. Shepherd et al. [39] proposed to extend the OSI standard of transport layer to accommodate multimedia traffic. The main idea of theirs was the introduction of a special channel used for synchronization control. This is named the synchronization channel. The synchronization commands and parameters, as well as the references to the matching points in the data streams are transmitted on this channel. In

the real-time case, the synchronization channel makes it possible to perform sophisticated synchronization control with involved data stream relationships. Moreover, when the synchronization channel is employed, user equipment, for example video devices, may be attached easily because the data streams are not modified [39]. However, the use of a synchronization channel has the drawback of requiring more communication resources. How to manage the bandwidth of this channel is not specified in the work of [39].

Nicolaou [33] presented a scheme for implementing synchronization among the related streams by inserting synchronization points in each individual stream. In [28], schedules are computed for data packets before the transmission. Multimedia data traffic will be running according to the schedule, to reach the destination in time for the presentation. As far as SRTDD is concerned, matching schedules can be planned for the data streams, so that if the data belonging to each stream can arrive at the destination in conformity to their schedules, they will be "matching" each other during their display. In [10], during the multimedia data transfer, every node on the transmission path in the network is involved in the synchronization control by regulating the random delay which has been introduced to an arriving packet by the previous hop. Therefore, if an "ideal" real-time channel can be guaranteed, at the destination, an almost fixed end-to-end delay (except for the random delay experienced on the last hop) can be obtained for the packet received on each connection. However, in B-ISDN networks (e.g., ATM network), all the intermediate nodes are fast cell switches, which carry thousands of connections. Therefore, it would be difficult for this scheme to be implemented since an extra layer is required at every switching node for the synchronization control, producing too much workload on the switch. The flow synchronization control protocol addressed in [11] is based on a similar idea of implementing an end-to-end synchronization delay. Instead of regulating the delay at every hop, random delay experienced by each packet will be regulated (equalized) at the end. In both delay regulating schemes, a globally accurate clock is required. At the same time, the actual delay experienced by every data packet cannot exceed the pre-set end-to-end or hop delay for the network connection.

Instead of the sender's predicting the possible packet arrival time at the receiver,

Rangan et al., have proposed another SRTDD scheme where the receiver derives the possible packet generating time at the sender. That is the multimedia mixer [35]. Without any global clock, the multimedia mixer performs a complete algorithm to synchronize data packets received from the different sources. Assuming the maximum and minimum delays that can be experienced by the data packet on the network, the mixer can derive the possible generating time of a received packet, according to the receiving time. The data packets can be mixed into a fusion set if they are generated at the same or almost the same time. The mixer also requires no control information transferred. This control scheme is easy to be implemented and can be used in the conferencing system. Relatively "good" network connections and clocks are required so that the time derivation and mixing can be valid [35]. Otherwise, incorrect fusions can be generated and the mixer cannot find out the mistake by itself. This is only because no control message is exchanged between the sender and the receiver.

Although schedules and estimates can be made for the multimedia data to be transferred, random network disturbance and synchronization errors are inevitable. Then, only scheduling the traffic is not sufficient to maintain a simultaneous data delivery. Therefore, for SRTDD, at the receiver, certain compensation is still necessary, when the synchronization error occurs. In [36], a feedback technique was presented which can inform the sender about the synchronization errors (i.e., "mismatching") in the play-back. Then the sender will adjust the data transmission of different data streams, e.g., speeding up the slow stream by skipping some data. This control scheme does not require any global clock. Meanwhile, it does not require any control message to be transferred between the sender and the receiver. Therefore, it is very suitable for the simple terminals such as a video monitor in a video on demand service.

2.3 Coordinations and Causal Ordering in Multimedia Cooperations

In multimedia cooperative applications, coordination and causal ordering are required. Coordination refers to the concurrency control among the participants in a group. Causal ordering, on the other hand, performs the ordering of the events in the cooperation, as we discussed in Chapter 1.

The concurrency degree needed in a system varies in different applications. So does the corresponding control scheme. One extreme case offers the maximum concurrency degree to the application users. For instance, in a shared working space (e.g., multimedia document joint-editing), a shared window is displayed at each participant's workstation. If the system provides no concurrency control for simultaneous actions, the participants will be allowed to constantly edit in the shared space, as well as to view the editing actions of others. In contrast to this "freedom", a minimum concurrency degree may be enforced by a strict control based on the notion of a "floor control" in some of the applications, e.g., a floor controlled teleconferencing system. In this case, only one participant who holds the "floor" (token) is allowed to "speak" (or transmit data) at any time. Between the two extremes, a varying degree of coordination may be necessary or desirable for present and future applications [47].

From the communication's view-point, token control provides ordering of the senders during their data transmissions. If more than one tokens are allowed in the system, at each receiver, the sequential order must be distinguished among the data received for display, when the order matters in the application. The same version of the document can thus be maintained at all the participants' sites (see Chapter 1). Ideally, the causal ordering of the events should be the same as the ordering of the occurring times of the events. Notice the event ordering here is different from the object ordering in a multimedia presentation.

The event ordering problem in distributed computing systems, including the logic clocks, the partial and total ordering of events, have been well studied [19][4]. Related with the event ordering, mutual exclusion in accessing the shared resource was taken as a

typical issue of synchronization control. Different algorithms were developed [4]. In recent works on multimedia synchronization control [38][47], e.g., multimedia event ordering in conferencing systems, the basic concepts and methods developed in distributed computing have been adapted and applied for the multimedia cooperative applications.

In [47], both the token based concurrency control and the real-time defined $\Delta-$causal ordering algorithm were proposed. In the control scheme, the number of tokens in a system can be varied for different applications, or during the different phases of one application. In the $\Delta-$causal control, when a sender is sending a message, it will embed the so-called "context" in the message. The context is actually the message-id's of the most recently received message(s) at this site. For instance, assume message m_i is received at time t_1 at the workstation N. If message m_j is sent by N at time $t_2 < t_1 + \Delta\tau$, N will embed the message-id of m_i as the "context" in m_j. Then each receiver will order the message display according to the available messages-id's in the received message, as long as the real-time constraint on the data is not violated. That is, when m_j is received, receivers will manage to display m_i before m_j if m_i does not arrive too late [47]. In developing cooperative applications, the $\Delta-$causal control is very favorable for real-time voice conferencing. However, as addressed in [47], the context of the message-id's recorded in a message sent can be incomplete due to the delay of certain messages. Only a partial ordering can be achieved at the destinations.

In [38], a contention control was provided on top of the proposed communication structure, for distributed conferencing applications. A high level abstraction of an event cluster was provided that allows a user to view related activities as one composite activity. A total ordering on a set of incoming events was enforced by one of the application participants designated as coordinator. Thus, it is a centralized control method.

2.4 Summary

In this chapter, we have described the synchronization control schemes and algorithms for both multimedia presentational and conversational applications. All the three issues of

the multimedia synchronization control are covered, including the Temporal Presentation Control, the Simultaneous Real-Time Data Delivery and the coordination and causal ordering for multimedia cooperation.

In the Temporal Presentation Control (TPC), OCPN models were proposed by Little et Ghafoor [29]. The scheduling algorithms were as well designed. However, we have found that the presentation application in a distributed system where multiple information sources are involved gives rise to a new requirement. That is the synchronization dialogue among these independent sources for the evaluation of the presentation scenario, in order to convert the relative and uncertain temporal requirement into the absolute and precise synchronization control.

Many schemes have been proposed for the multimedia synchronization control on the networks. Among these schemes, multimedia mixer [35] and feedback techniques [36] are very practical ones. Making a complete algorithm, the mixer itself is not a multimedia transport protocol. Neither is the feedback technique. It will be very interesting and useful if the synchronization algorithms can be embedded into the transport protocols and an entire multimedia communication architecture can thus be formed.

In the Multimedia Communications Research Laboratory (MCRLab) at the University of Ottawa, a real-time multimedia conferencing system has been developed [17]. On this conferencing platform, a CSCW application, i.e., multimedia joint editing has been designed and implemented. Algorithms are thus required for the coordination control of the manipulation on the shared document, and for the causal ordering of the different document versions. The existing control methods need to be examined and suitable algorithms have to be implemented.

From Chapter 3, we will present our work concerning the above three aspects of multimedia synchronization control. We will propose control schemes, protocols and algorithms developed to fill the gaps in this new and exciting field.

Chapter 3

Temporal Presentation Control for Multimedia Teleorchestra with Independent Sources

Real-time multimedia TPC (Temporal Presentation Control) requires the scheduling of the multimedia data according to the designed presentation scenario. From the previous chapter, we know that the scenario may provide only relative temporal requirements, although TPC scheduling needs the precise temporal information. As suggested in [31] [42], the relative synchronization requirement should be converted, through a synchronization dialogue, into an absolute synchronization operation, i.e., scheduling. This "synchronization dialogue" requires the modeling of the uncertain and relative temporal relationships involved in the presentation scenario.

In our work, this challenging issue is studied by investigating the teleorchestra application. The remote document creation application mentioned in Chapter 2 is just an example of multimedia teleorchestration. In the teleorchestra system, independent multimedia information sources are involved. One of the important aspects of real-time teleorchestration, e.g., in the remote document creation, is that a user can create a document without having to retrieve the objects from the sources to his/her workstation. All the user has to do is to make a scenario. Then in the distributed system, the required data

objects will be delivered from the multiple information sources directly to the viewers' workstations. These objects will be presented according to the designed scenario. Hence, the real-time distributed multimedia data integration is performed in the teleorchestra system. It is worth noting that a user can create different presentation scenarios using the same data objects. These remarkable features make the teleorchestration very useful in collaborative and conversational applications, for instance, in "brainstorming" sessions [12].

In teleorchestration, among the data objects to be presented, relative and uncertain temporal requirements may be involved. "Fuzzy" presentation scenarios are thus generated. In this chapter, we first describe a temporal model that can handle these "fuzzy" scenarios which contain imprecise synchronization constraints, such as unknown object presentation durations and relative event occuring times. A synchronization model for multimedia teleorchestra is thus established, including the synchronization dialogue carried out among the independent sources. Based on this model, a TPC scheduling algorithm and control scheme are then designed to synchronize the independent sources for the remote presentation.

3.1 Temporal Relationships in Multimedia Integrations

We observe the following characteristics of the multimedia temporal presentation in teleorchestration. First, because of the distributed environment as well as various media included, the initiator of an orchestration (e.g., the user or the application) cannot specify all the information about the objects required. While the display duration of a static image can be described as a playout time plus a holding time of the picture on the screen, the presentation duration of a video object which resides at a remote source is usually unknown to a user. Explicit temporal information of the data objects is thus unlikely available in the scenarios. Despite the lack of temporal information, very "rich" and yet very "relative" temporal relationships may be required among the data objects to be pre-

sented. In a collaborative application, e.g., in joint editing [16][45], "fuzzy" scenarios are often involved. When n multimedia objects are displayed at the same time, for example, the user may require all the $n-1$ objects to be displayed "during"[1] the presentation of the object which has the longest duration (e.g., the theme of the presentation), although the user may have no knowledge about the display duration of these objects. The synchronization requirements may involve both the presentation durations and some particular points, such as the starting, the ending times (points) of an object presentation, as well as any points in between (e.g., at the end of the second page, the audio segment starts to play out).

Secondly, data object scheduling is needed to provide the temporal control in the real-time object delivery and display [30]. However, scheduling the data objects needs precise information, e.g., the required occurring times of the display events. Therefore the "relative synchronization requirement" in the scenario has to be converted into the "absolute synchronization operation"[42] among the distributed sources.

In summary, the following important characteristics are relevant to our work.

(1) The available temporal knowledge about the objects involved in the presentation is strictly relative. Precise time values such as durations and occurring times may not be known. Meanwhile, in order to support the application, the representation of scenarios should allow uncertainty of information.

(2) Temporal modeling of the integration should support the synchronization dialogue among the independent sources, and the real-time data scheduling.

In Chapter 2, we mentioned the OCPN models [29]. Since the temporal interval is taken as the primitive, OCPN models rely on the duration information in their representation. The basic OCPN models for the relation of "during", "finishes" and "overlapping" can only be distinguished by the different duration values of the involved intervals [29]. Meanwhile, when multiple paths exist between two transitions, the OCPN model by nature can only represent the maximum presentation duration among the paths. The point based models proposed in [34][45], on the other hand, do not take the TPC scheduling into account.

We present here a mathematical model for the multimedia scenario in a distributed environment. We also take the notion of *temporal interval* as primitive. Nevertheless, our model can represent all the basic temporal relations that may exist between every two temporal intervals [1], and the nested relationships among the multiple intervals, without the knowledge of their durations. At the same time, various synchronization points are handled. When the acquisition of the duration information generates the changes in the temporal relationships represented in the scenario, the consistency and correctness of the model are preserved. Thus, "fuzzy" scenarios can be managed. Meanwhile, from the relative information available, the model is well designed for the inference of absolute time parameters that are relevant to the synchronization operation. A synchronization algorithm based on this model is also presented.

3.2 Temporal Intervals in Multimedia Scenarios

First, we give an example to show the temporal relationships among the data objects in a fuzzy orchestration scenario. In this example, several distributed data objects are required to be presented according to the scenario depicted in Fig. 3.1. Picture $P1$ should first start to display. When $P1$ is finished, picture $P2$ will start at the same time with an audio object $A1$. When audio $A1$ is finished, audio $A2$, video $V1$ and text $T1$ will be presented. Between $A2$ and $V1$, $A2$ is to be displayed "during" $V1$, if the presentation of object $V1$ lasts longer. If the display of $A2$ lasts longer than that of $V1$, $V1$ should be displayed during the presentation of $A2$. In Fig. 3.1(a) and Fig. 3.1(b), these two cases are illustrated by the timelines. In either case, text $T1$ is required to start with audio $A2$ simultaneously. Then after the display of $T1$, $V1$ and $A2$, $T2$ and $A3$ will be presented. $A3$ is required to finish at the same time as $T2$. The scenario determines the temporal presentation of the involved data objects. In the scenario, the temporal relationships can be specified by the constraints on the starting, ending time (points) of the display, or any other time point during the display, or on the entire display interval of an object.

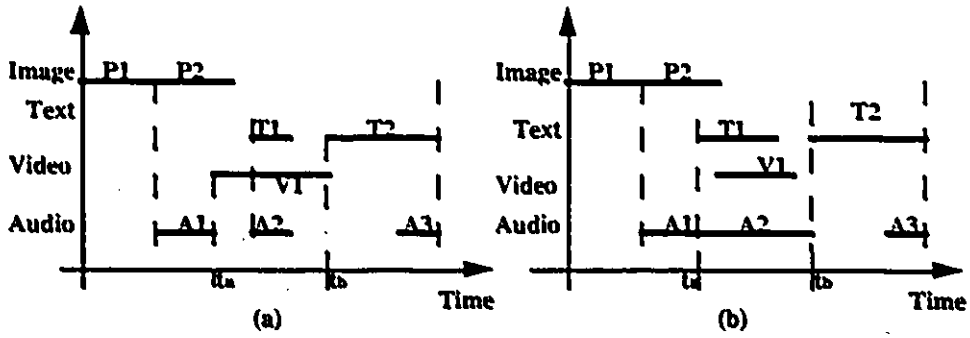


Figure 3.1: Example of temporal requirements in a multimedia scenario

3.2.1 Temporal Logic for Multimedia Intervals

Time can be referred in two ways: as a point or as an interval. Multimedia objects always have durations in their presentation. An isolated time point is not meaningful in describing the temporal requirement unless it is related to certain object, which is delivered and presented in a certain interval. Therefore, the notion of *temporal interval* is taken as the primitive in our model.

Temporal interval is a non-zero duration of time, e.g., a temporal interval of "3 minutes". This is contrasted with a time instant which is a "zero-width" time point, e.g. "3:00pm" or "the starting time point of the video object". A formal definition of the temporal interval can be found in [13][29].

In multimedia temporal presentation, one multimedia data object associates with one temporal interval. The terms of "object X" and "interval X" are thus both used equivalently. On the other hand, one interval may contain several objects in different media. For example, in Fig. 3.1, during the interval $[t_a, t_b]$, video $V1$, audio $A2$ and text $T1$, i.e., "video interval $V1$", "audio interval $A2$" and "text interval $T1$" are involved. Therefore, one temporal interval may involve several concurrent multimedia intervals.

A simple logic of temporal interval was introduced in [13][1]. Now we adapt this logic for modeling the multimedia scenario.

As addressed in [1], there are thirteen relations (i.e., seven relations and their inverses) that can hold between any two temporal intervals. Using the representation of [1]

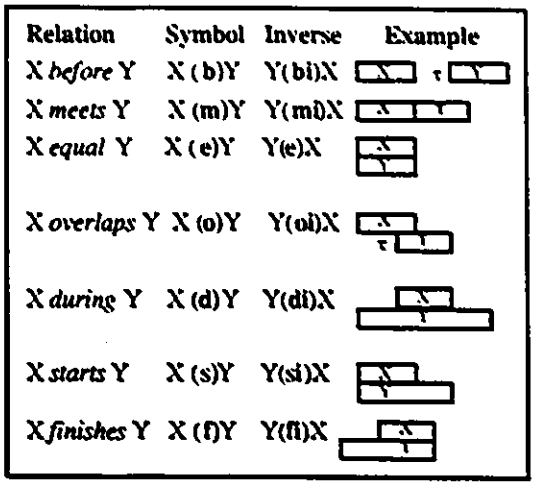


Figure 3.2: Basic relations involved in two temporal intervals

and [29], they are illustrated in Fig. 3.2.

Take $R = \{\mathbf{b}, \mathbf{e}, \mathbf{m}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}, \mathbf{bi}, \mathbf{mi}, \mathbf{oi}, \mathbf{di}, \mathbf{si}, \mathbf{fi}\}$, including all the thirteen possible relations between any two intervals. Given two intervals X and Y, $\mathcal{T}(X, Y)$ returns the basic relation specified between them. That is $\mathcal{T}(X, Y) = r$, when $X\,(r)\,Y$, $r \in R$ (see Fig. 3.2).

1. Parallel relations

A subset of $R_f \subset R$ is defined as

$$R_f = R_{1f} \cup R_{2f} = \{\mathbf{e}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}\} \cup \{\mathbf{oi}, \mathbf{di}, \mathbf{si}, \mathbf{fi}\} = \{\mathbf{e}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}, \mathbf{oi}, \mathbf{di}, \mathbf{si}, \mathbf{fi}\}$$

In order to specify any of the relations $r \in R_f$, the knowledge of the duration is needed. Given two intervals X, Y and $r =$ 'o', for instance, $X\,(o)\,Y$ is specified and τ is given (see Fig. 3.2). If the duration of X is longer than expected, the relation of $r =$ 'o' can be changed to $r =$ 'fi' or $r =$ 'di'.

The other relations $r \in R_f$ present the same feature. Among $r \in R_f$, when the durations are unknown, the starting and ending time points of the involved temporal intervals may embody both the specified and unspecified relations to the other intervals. In order to capture the semantics in the temporal requirement, two functions f_s and f_e

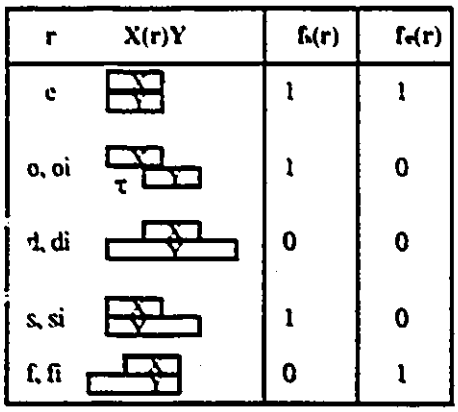


Figure 3.3: Functions of $f_s(r)$ and $f_e(r)$

are defined to describe the margin information of the intervals involved in relation $r \in R_f$.

$$f_s, f_e : R_f \longrightarrow B$$

where B is the boolean set $B = \{0, 1\}$.

If $f_s(r) = 0$, $r \in R_f$, then r gives no specified temporal requirement at the starting times of the two intervals involved. On the other hand, $f_s(r) = 1$ constrains the starting times of the two intervals. The $f_e(r)$ is interpreted similarly for the ending time of the intervals. In Fig. 3.3, $f_s(r)$ and $f_e(r)$ are defined for $\forall r \in R_f$.

Through $f_s(r)$ and $f_e(r)$, the semantics of the relation $r \in R_f$ at the starting and the ending points of the involved intervals are defined. When $r =$'d' and $r =$ 'di' ("during"), we have $f_e(r) = f_s(r) = 0$. This relation can be used to model the case when the exact matching times ("points") of the two objects are not strictly required in the application. For instance, when object X is just a description that loosely accompanies the presentation of object Y, $X\,(d)\,Y$ may be applied. Then if the duration of interval X is shorter than that of Y, the presentation of X can be placed in the middle of that of Y (or vice versa). On the other hand, the relation of $r =$ 'o' ("overlapping") can be applied to specify a desired matching time (i.e., the matching point) between two intervals involved. The matching point is represented by an interval τ, which may only be relatively referred in the scenario. For instance, the τ can be required as "at the end of the second page of text

Duration Requirement	Specification	Symbol Adopted or New Symbol	f_s, f_e functions	Involved Relation of $r \in R_f$
Maximum dura. Fig. 3.4)(a)	$X(d)Y(d)Z$	d	$f_s(d) = 0$ $f_e(d) = 0$	r = 'd' implied between every two intervals
Minimum dura. Fig. 3.4)(b)	$X(e)Y(e)Z$	e	$f_s(e) = 1$ $f_e(e) = 1$	r = 'e' implied between every two intervals
A chosen dura. Fig. 3.4)(c)	$ch(X,Y,Z) = Y$ $Y(d)X, Z(s)Y$	ch	$f_s(ch) = 1$ $f_e(ch) = 1$	$r \in R_f$ specified between the intervals

Table 3.1: Duration Specifications

T^m.

Now, except for r = 'o' and 'oi', the other parallel relations $r \in R_f$ are not distinguished from their inverse in the scenario specifications. The reason is that the only difference between these relations and their inverses is the duration information. The f_s and f_e functions will be employed to deal with the unknown duration information. For instance, when $X(d)Y$ is specified, $X(di)Y$ will also be handled, benefitting from the identical f_s and f_e functions.

In the application scenario, the temporal relationship can be required in terms of the temporal durations despite the lack of duration information of the involved intervals. Assume the interval during which several concurrent objects are presented is $[t_s, t_e]$. Then the duration of $[t_s, t_e]$ can be required as one of the cases shown in Table 3.1. As presented in the table, the f_s and f_e functions provide the means to specify these duration requirements. The presentation semantics are defined as:

$X(d)Y(d)Z$: All the other objects are displayed during the presentation of the object that has the longest presentation duration.

$X(e)Y(e)Z$: The display of all the objects starts simultaneously. The presentation of all the other objects will be cut off when one (the first) of the objects is finished.

$ch(X,Y,Z) = Y$: The presentation duration of all the involved objects is required to equal that of the one chosen. Display the objects according to the relations $r \in R_f$

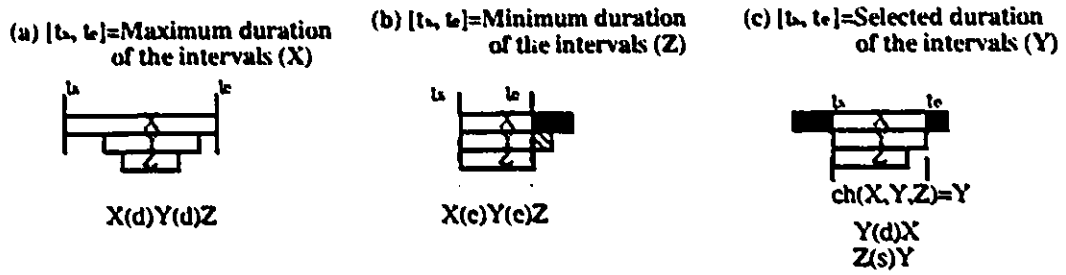


Figure 3.4: Specification on Durations

specified between every two of them. The presentation of some objects may be cut off to fit the specified duration.

Now, we have three duration specifications $r \in R_m = \{d, e, ch\}$, and $f_s, f_e : R_f \cup R_m \longrightarrow B$. While the thirteen basic relations $r \in R$ are only specified between every two temporal intervals, duration requirement $r \in R_m$ can be applied to $[t_s, t_e]$ involving concurrent multiple intervals. The duration specifications are useful in collaborative multimedia presentation. For instance, when the display of the other objects wanted to be cut off to match the object of the theme, the $[t_s, t_e]$ can be specified as the display duration of the theme object.

2. Sequential relations

There are two basic sequential relations $r =$ 'm' and $r =$ 'b' between two temporal intervals. However, before the durations of the relevant intervals are known, relations $r =$ 'm' and $r =$ 'b' may not be determined. We also observe that $X(b)Y$ can be decomposed into $X(m)\tau, \tau(m)Y$ (Fig. 3.2). Meanwhile in the application scenarios, multiple intervals can be involved in the sequential relation requirement. Thus, the sequential specifications are provided as:

$\{A\}B$: Interval(s) in B will start after all the intervals in A are finished.

$<A> B$: Interval(s) in B will start when one of the intervals (the first) in A is finished.

In $\{A\}B$, the interval which is the last to be finished in A, say $\vec{n}_x$, will "meet" the interval that is the first to start in B, say $\vec{n}_y$. Thus we have $\vec{n}_x(m)\vec{n}_y$. In the case of $<A> B$, the

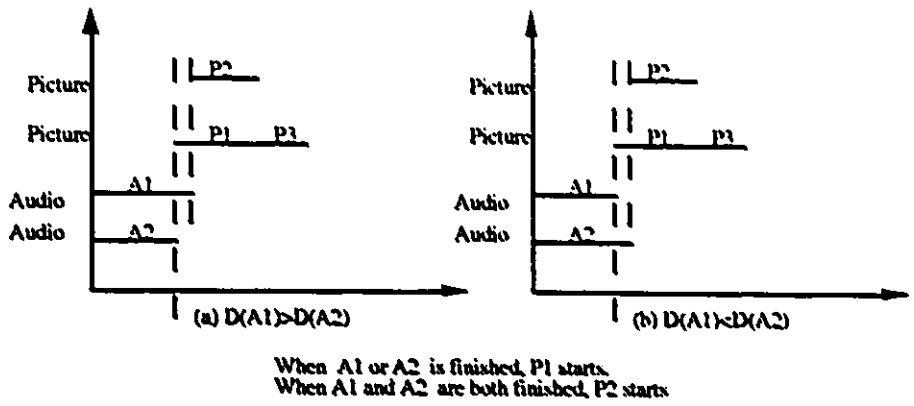


Figure 3.5: Multimedia scenario: example 2

Specification for the scenario in Fig. 3.1.	Specification for the scenario in Fig. 3.5
$Start\{P1\}\{P2(s)A1\}End1;$	$Start <A1(s)A2><P1><P3> End1;$
$A1\{T1(s)A2, V1(d)A2\}\{T2(f)A3\}End2$	$\{A1(s)A2\}\{P2\}End2$

Table 3.2: Specification Examples

interval that is the first to be finished in A, say $\vec{n}_x$, will "meet" $\vec{n}_y$, i.e., $\vec{n}_x(m)\vec{n}_y$. The intervals contained in A may be related with each other through $r \in R_f \cup R_m$. They may also have no relations among them at all. On the other hand, if more than one intervals are contained in B, they are assumed to be related with each other through some relation(s) of $r \in R_f \cup R_m$. When A and B both contain only one interval, $\{A\}B = <A> B = A(m)B$.

Applying the sequential and the parallel relations introduced, a multimedia scenario can be formally specified. In the appendix 1, we give the BNF style description for the scenario specifications. In Table 3.2, the specifications of two example scenarios are illustrated.

Thus, the temporal logic, including both the sequential and parallel relations, as well as the duration requirements, are established for the multimedia intervals. The restrictions on the temporal relationships required in a scenario will be discussed later.

interval $\vec{n} \in \{X, Y_1, ..., Y_n\}$	$\mathcal{T}(X,Y)$ in $\vec{n}.F_s$ & $\vec{n}.F_e$
$\vec{n}_x$	ch
$\vec{n}_{y_k}$ s.t. $\vec{n}_x(r)\vec{n}_{y_k}, r \in R_f$	r
other $\vec{n}_{y_k}$'s	All the $r \in R_f$ exists between $\vec{n}_{y_k}$ and the intervals in $\{Y_1, ..., Y_n\}$

Table 3.3: Definitions for r = 'ch'

3.2.2 modeling Temporal Intervals

Based on the functions f_s and f_e, the new model for the representation of the multimedia temporal interval is defined.

Definition 3.1 *An interval vector*

$$\vec{n}_x = (S_r, O_{id}, F_s, F_e)$$

is used to represent a temporal interval X in the multimedia scenario. $\vec{n}_x.S_r$ denotes the source that is actually the "owner" of interval X. The name of the corresponding object of the interval is indicated in $\vec{n}_x.O_{id}$.

If $\vec{n}_x$ is not involved in any $r \in R_f \cup R_m$, $\vec{n}_x.F_s = \vec{n}_x.F_e = \phi$ is defined. If $\vec{n}_x$ is only involved in $r \in R_f$:

$$\vec{n}_x.F_s = \{\vec{n}_y : f_s(\mathcal{T}(X,Y)) = 0\} \text{ and } \vec{n}_x.F_e = \{\vec{n}_y : f_e(\mathcal{T}(X,Y)) = 0\} \quad (3.1)$$

In the case that $\vec{n}_x$ is involved in r = 'd', 'e'$\in R_m$, $\vec{n}_x.F_s$ and $\vec{n}_x.F_e$ are conformed to the definition given in Formula 3.1.

The relation r = 'ch'$\in R_m$, as addressed in the previous subsection, includes various relation of $r \in R_f$ specified between every two involved intervals. Therefore, given $X = ch(X, Y_1, ..., Y_n)$, $X(r_1)Y_i$, $Y_i(r_2)Y_j$, ..., etc., the $\mathcal{T}(X,Y)$ in Formula 3.1 is defined in Table 3.3. That is, if Y_i is involved in relation r with X, $X = ch(X, Y_1, Y_2, \cdots, Y_n)$, only the relation r counts in $\vec{n}_{y_i}.F_s$ and $\vec{n}_{y_i}.F_e$, despite that Y_i may be related with other $Y_j \in \{Y_1, ..., Y_n\}$.

Examples: Given five objects X, Y, Z, V, U, the temporal relationships among them are: $X = ch(X, Y, Z, V, U)$, $Y(d)X$, $Z(s)X$, $V(f)X$ and $U(s)V$. Then we have:

interval $\vec{n}$	X	Y	Z	V	U
$\vec{n}.F_s$	ϕ	$\{X\}$	ϕ	$\{X\}$	ϕ
$\vec{n}.F_e$	ϕ	$\{X\}$	$\{X\}$	ϕ	$\{V\}$

For the scenario given in Fig. 3.1, we can obtain:

interval $\vec{n}$	P_1	P_2	A_1	A_2	T_1	V_1	T_2	A_3
$\vec{n}.F_s$	ϕ	ϕ	ϕ	$\{V_1\}$	ϕ	$\{A_2\}$	$\{A_3\}$	$\{T_2\}$
$\vec{n}.F_e$	ϕ	$\{A_1\}$	$\{P_2\}$	$\{V_1, T_1\}$	$\{A_2\}$	$\{A_2\}$	ϕ	ϕ

The scenario shown in Fig. 3.5 has $\vec{n}_{A_1}.F_e = \{A_2\}$, and $\vec{n}_{A_2}.F_e = \{A_1\}$. All the other $\vec{n}.F_s$ and $\vec{n}.F_e$ equal ϕ. A symmetry can be observed in $\vec{n}_x.F_s$ and $\vec{n}_y.F_s$ ($\vec{n}_x.F_e$ and $\vec{n}_y.F_e$), when $\vec{n}_x$ and $\vec{n}_y$ are related through relation $r \in R_f \cup R_m - \{e, ch\}$. That is, if $\vec{n}_x \in \vec{n}_y.F_s$ (or $\vec{n}_x \in \vec{n}_y.F_e$), $\vec{n}_y \in \vec{n}_x.F_s$ (or $\vec{n}_y \in \vec{n}_x.F_e$) also holds.

We use $\vec{N}$ to denote the set of all the intervals involved in a scenario. $\mathcal{D}(\vec{n}_x)$ ($\forall \vec{n}_x \in \vec{N}$) is the duration of the corresponding interval. The final value of $\mathcal{D}(\vec{n}_x)$ may not be known until the relevant sources join the data integration for the presentation.

An interval vector can be an object vector or an intermission vector. An intermission vector $\vec{n}_\tau$ represents an interval which is not mapped to any multimedia object, i.e., it is an intermission. In this case, $\vec{n}_\tau.O_{id}$ indicates τ is not for any object but an intermission. The duration $\mathcal{D}(\vec{n}_\tau)$ can be specified by referring to other objects, e.g., "at the end of the second page of text T".

Intermission τ can be used in the relations of "before" and "overlapping". In the relation $r =$ 'o' (or $r =$ 'oi'), τ does not appear in the specification (e.g., $\vec{n}_x(o)\vec{n}_y$, $\vec{n}_y(oi)\vec{n}_x$). The intermission vector $\vec{n}_\tau$ is thus called the virtual intermission vector here. In $\vec{n}_x(o)\vec{n}_y$, $\vec{n}_\tau.S_\tau = \vec{n}_y.S_\tau$ is assumed, since $\vec{n}_\tau(m)\vec{n}_y$ holds. On the other hand, the intermission vector $\vec{n}_\tau$ is specified for the relation of "before", i.e., $<\vec{n}_x><\vec{n}_\tau><\vec{n}_y>$ for $\vec{n}_x(b)\vec{n}_y$. In the application scenario, we usually have $<\vec{n}_x><\vec{n}_\tau> B$, where B contains more than one intervals. When it is uncertain which interval $\vec{n} \in B$ will evince $\vec{n}_\tau(m)\vec{n}$, $\vec{n}_\tau.S_\tau$ will include all the $\vec{n}.S_\tau$, $\vec{n} \in B$. Therefore, an isolated intermission that meets or before no object is prohibited. Intermission τ always has $\vec{n}_\tau.F_s = \vec{n}_\tau.F_e = \phi$.

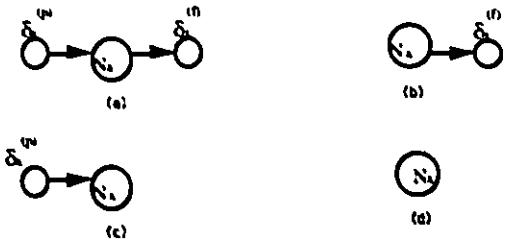


Figure 3.6: Node of a temporal vector

3.3 Time-flow Graph for Multimedia Scenarios

3.3.1 Time-Flow Graph Model

The model of interval vectors and the involved temporal information are maintained in a Time-Flow Graph (TFG).

Definition 3.2 *A* $TFG = \{\Delta_N, N_t, E_d\}$ *is a time-flow graph for a scenario which has an interval set* $\vec{N}$. Δ_N *is a set of nodes for the interval vectors.* N_t *is a set of transit nodes and* E_d *is a set of directed edges.*

For an interval vector $\vec{n}_x \in \vec{N}$, its model $\Delta_x \in \Delta_N$ is composed of an interval node N_x, representing the interval of $\vec{n}_x$, and δ node(s), representing $\vec{n}_x.F_s$ and $\vec{n}_x.F_e$. Δ_x can be one of the following forms:

$$(\delta_x^{(p)}, N_x, \delta_x^{(f)}), (\delta_x^{(p)}, N_x), (N_x, \delta_x^{(f)}), (N_x)$$

If $\vec{n}_x.F_s \neq \phi$ & $\vec{n}_x.F_e \neq \phi$, Δ_x is $(\delta_x^{(p)}, N_x, \delta_x^{(f)})$, as shown in Fig. 3.6(a). The model of $\vec{n}_x.F_s = \phi$ & $\vec{n}_x.F_e \neq \phi$ is $(N_x, \delta_x^{(f)})$, shown in Fig. 3.6(b). Similarly, $(\delta_x^{(p)}, N_x)$ is the model for $\vec{n}_x.F_s \neq \phi$ & $\vec{n}_x.F_e = \phi$ (Fig. 3.6(c)). At last, if $\vec{n}_x.F_s = \vec{n}_x.F_e = \phi$, (N_x) is given in Fig. 3.6(d). In the models, the δ nodes ($\delta_x^{(p)}$ and $\delta_x^{(f)}$) do not exist independently and only belong to the node N_x. An intermission node has no δ nodes.

There are two kinds of transit nodes in N_t, the $sq-node$ and the $tri-node$. The transit nodes are employed for the sequential specifications of $\{A\}B$ ($< A > B$), when either no interval, or multiple intervals are involved in A (see Fig. 3.8). The $sq-node$

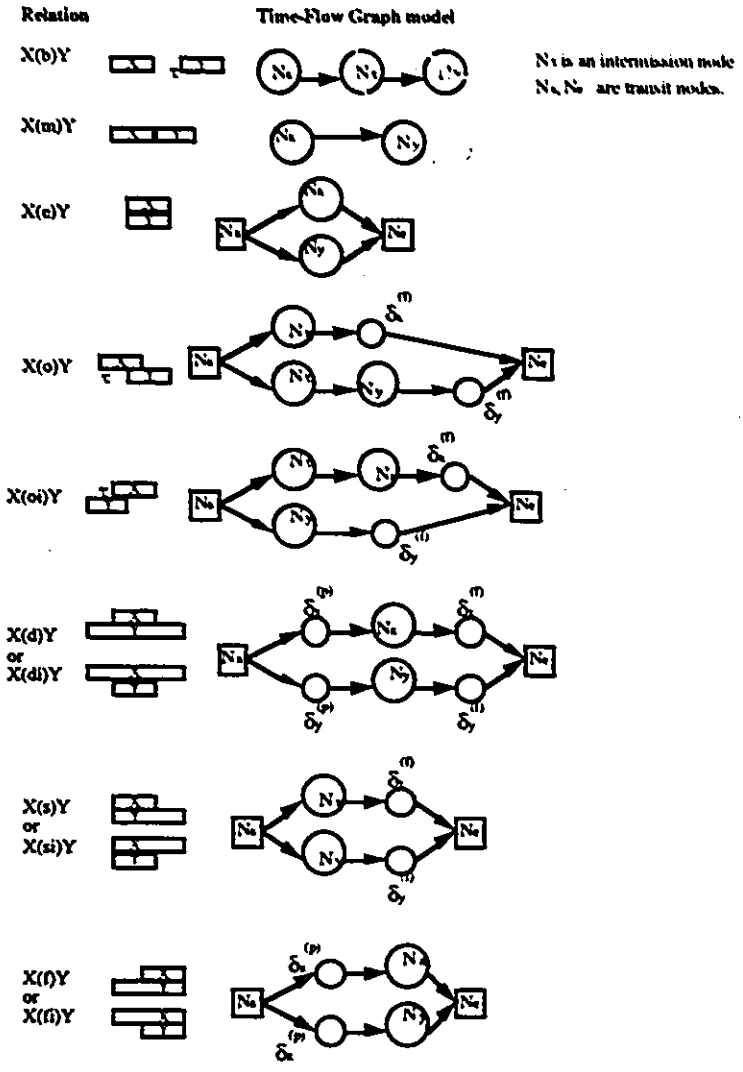


Figure 3.7: Time-flow graph for basic relations $r \in R$

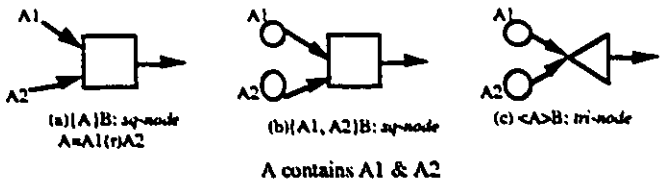


Figure 3.8: Time-flow graph for sequential specifications

and $tri - node$ represent $\{A\}B$ and $< A > B$, respectively. An edge $e_d \in E_d$ is used to connect the nodes in the model of a Δ_x (see Fig. 3.6), as well as the nodes Δ_x in TFG models. The direction of e_d represents the sequential ordering of the presentation in time, which is explicit in the scenario. In the TFG, a flow path from node Δ_i to Δ_j is a set of nodes Δ_i, Δ_{i+1},...,Δ_{j-1}, Δ_j and edges (directed) such that there is an edge between successive pairs of nodes (Δ_i and Δ_{i+1}, Δ_{i+1} and Δ_{i+2}, ..., Δ_{j-1} and Δ_j). Since Δ_i, Δ_{i+1},...,Δ_{j-1}, Δ_j are all intervals with non-zero durations, there exists no path Δ_i, Δ_{i+1},...,Δ_{j-1}, Δ_j such that $\Delta_i = \Delta_j$. That is, a TFG (Δ_N, N_t, E_d) contains no cycle.

In Fig. 3.7, Fig. 3.8 and Fig. 3.9 the relations discussed in section 3.1 are represented with TFG models. Two intervals involved in $r \in R_f \cup R_m$ need a common starting node (e.g., the transit node N_s in Fig. 3.7) as well as a common ending node (e.g., the node N_e in Fig. 3.7). In the relations of $r =$ 'o' and $r =$ 'oi' (Fig. 3.7), virtual intermission τ is represented by the node N_τ. Since it is a virtual intermission, N_τ is included in Δ_y in $\vec{n}_x(o)\vec{n}_y$ (or Δ_x in $\vec{n}_y(oi)\vec{n}_x$). Assume that two intervals A_1 and A_2 in A, $< A > B$, have no relation $r \in R_f \cup R_m$ between them. Then A_1 and A_2 are connected directly to the $sq - node$ (Fig. 3.8(b)). There is no confusion of the relation between A_1 and A_2 since they have no common starting node. The interval vectors in Fig. 3.9(c) were already discussed in one of the examples given in section 3.2. The relationships among the five intervals are as follows.

Between Y and X: If $\mathcal{D}(Y) \leq \mathcal{D}(X)$, $Y(d)X$ or $Y(e)X$; If $\mathcal{D}(Y) > \mathcal{D}(X)$, the display of Y will be constrained to equal $\mathcal{D}(X)$. The display of Y can be cut at both ends.

Between X and Z: The display of Z will start at the same time as that of X. It will be cut off when the presentation of X is finished.

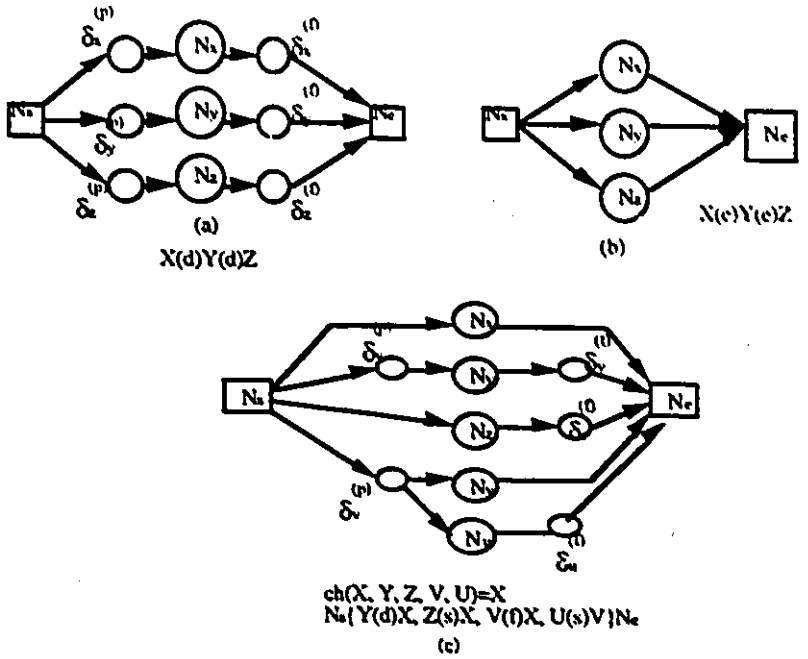


Figure 3.9: Time-flow graph for duration specifications

Between V and X: The display of V will "finish" X. If $\mathcal{D}(V) > \mathcal{D}(X)$, the head of the presentation of V will be cut to make $D(V) = \mathcal{D}(X)$.

Between V and U: Object U will start at the same time as object V. The display of U will be cut off when the presentation of X is finished.

3.3.2 Examples and Discussions

In Fig. 3.10, another example which involves multiple concurrent intervals is given. In Fig. 3.11, both the scenario examples in Fig. 3.1 and Fig. 3.5 are represented in TFG models. When the basic models are cascaded and paralleled to represent an entire scenario, the common starting node may be any kind of nodes, e.g., vector node ($\Lambda 1$ in Fig. 3.11(a)) or δ node ($\delta_y^{(p)}$ in Fig. 3.10).

Now we present two more scenario examples to demonstrate the mixed application of relation $r \in R_f$ and $r \in R_m$.

Scenario Example 3: During the presentation of a text object T, an interval

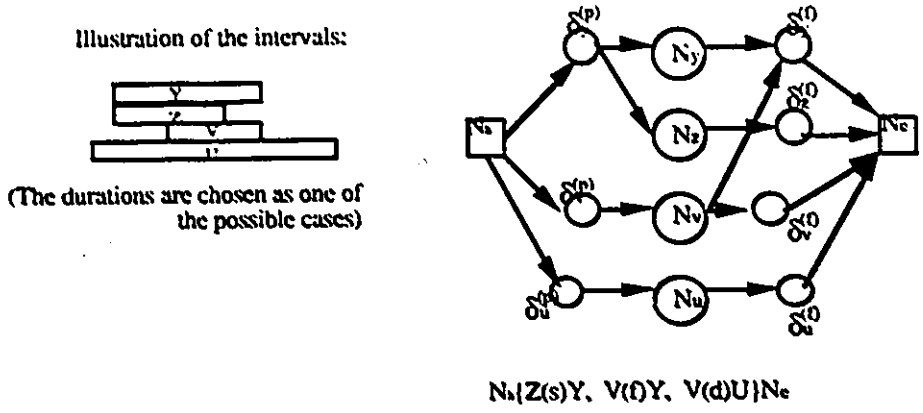
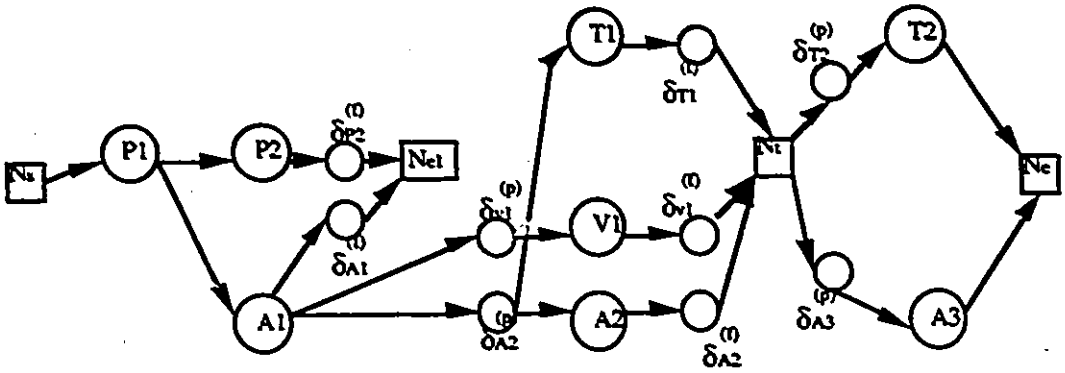
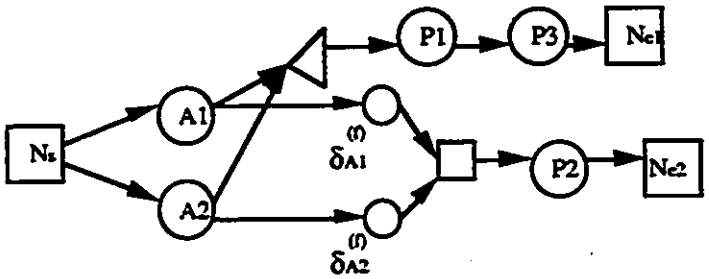


Figure 3.10: An example of time-flow graph



(a) TFG for the scenario in Fig. 3.1



(b) TFG for the scenario in Fig. 3.5

Figure 3.11: Examples of TFG models

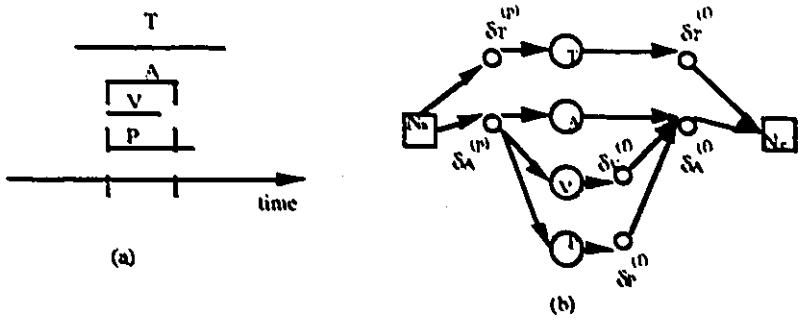


Figure 3.12: Mixed applications of $r \in R_f$ and $r =$ 'ch'

involves a video object V, an audio object A and images P is inserted, as illustrated in Fig. 3.12(a). The rendition of the inserted interval is cropped according to the play-out time of audio A. For this scenario, we have the following specification:

$$N_s\{T(d)A, A = ch(V, A, P), A(s)V, A(s)P\}N_e$$

The corresponding TFG model is depicted in Fig. 3.12(b).

The $\vec{n}.F_s$ and $\vec{n}.F_e$ are obtained as the following.

interval $\vec{n}$	T	A	V	P
$\vec{n}.F_s$	$\{A\}$	$\{T\}$	ϕ	ϕ
$\vec{n}.F_e$	$\{A\}$	$\{T\}$	$\{A\}$	$\{A\}$

In the example, relations $r_1 \in R_f$ and $r_2 =$ 'ch' coexist. The relation of $r_1 \in R_f$ is only specified between interval T and A, $A = ch(V, A, P)$. Thus, the intervals A, V, P involved in $r_2 =$ 'ch' is taken as a templet, whose presentation duration is determined by the "representative" interval A.

Scenario Example 4: During the presentation of text T, an intermission involves a video object V, an audio object A and images P is inserted. The rendition of the three inserted objects is cropped according to the minimum presentation duration (see Fig. 3.13(a)). When one of the three is finished, the other two are cut off.

In this example, intervals A, P and V are still taken as a templet. Meanwhile, one of the intervals is arbitrarily chosen as the "representative" of the templet. For instance, assume the interval V is chosen. Then the relation $r_1 =$ 'd'$\in R_f$ is specified between

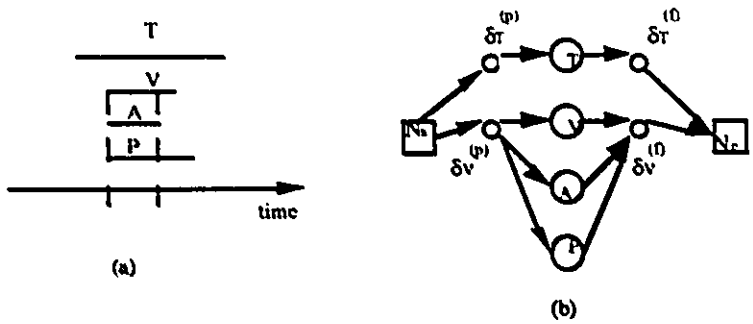


Figure 3.13: Mixed applications of $r \in R_f$ and $r =$ 'e'

interval V and T. We have the following specification for this scenario.

$$N_s\{T(d)(A(e)V(e)P)\}N_e.$$

The TFG model (Fig. 3.13(b)), $\vec{n}.F_s$ and $\vec{n}.F_e$ are as well obtained .

interval $\vec{n}$	T	A	V	P
$\vec{n}.F_s$	$\{V\}$	ϕ	$\{T\}$	ϕ
$\vec{n}.F_e$	$\{V\}$	ϕ	$\{T\}$	ϕ

When one of the other intervals (i.e., A or P) is selected instead of V, different TFG models will be resulted. Nevertheless, they all represent the same presentation sequence of the objects. The mixed application of relations $r_1 \in R_f$ and $r_2 =$ 'd'$\in R_m$ is rarely encountered in the application scenarios. This mixed application can also be handled by "templet", provided that the "representative" interval is explicitly specified.

Given a node N_x in a time-flow graph, $\mathcal{P}(N_x)$ contains the nodes preceded in the graph. The nodes that succeed N_x are in $\mathcal{S}(N_x)$. For instance, in Fig. 3.11(a), $\mathcal{P}(A1) = \{P1\}$, and $\mathcal{S}(A1) = \{\delta_{V1}^{(p)}, \delta_{A2}^{(p)}, \delta_{A1}^{(f)}\}$. $\mathcal{P}(\Delta_x)$ ($\mathcal{S}(\Delta_x)$) are defined similarly for Δ_x. That is, in Fig. 3.11(a), $\mathcal{P}(\Delta_{A2}) = \{A1\}$ and $\mathcal{S}(\Delta_{A1}) = \{N_{e1}\}$.

In TFG models, if $\vec{n}_x\,(r)\,\vec{n}_y$, $r \in R_f \cup R_m$, $(\mathcal{P}(\Delta_x) \cup \mathcal{P}(N_x)) \cap (\mathcal{P}(\Delta_y) \cup \mathcal{P}(N_y)) \neq \phi$ and $(\mathcal{S}(\Delta_x) \cup \mathcal{S}(N_x)) \cap (\mathcal{S}(\Delta_y) \cup \mathcal{S}(N_y)) \neq \phi$ hold, because the intervals have common starting and ending nodes. In Fig. 3.10, for instance, as $V(f)Y$ is specified, we have

$$(\mathcal{P}(N_y) \cup \mathcal{P}(\Delta_y)) \cap (\mathcal{P}(N_v) \cup \mathcal{P}(\Delta_v)) = \{\delta_y^{(p)}, N_s\} \cap \{\delta_v^{(p)}, N_s\} = \{N_s\};$$

and $(\mathcal{S}(N_y) \cup \mathcal{S}(\Delta_y)) \cap (\mathcal{S}(N_z) \cup \mathcal{S}(\Delta_z)) = \{\delta_y^{(f)}, N_e\} \cap \{\delta_z^{(f)}, \delta_y^{(f)}, N_e\} \quad \{\delta_y^{(f)}, N_e\}$.

In the models, a partial sequential ordering among the intervals in $\vec{N}$ is reflected directly from the scenario specifications. Given the specification $\{A\}B$ or $< A > B$, assume that $\vec{N}_A$ and $\vec{N}_B$ are the sets of intervals contained in A and B, respectively. Then, $\vec{N}_B = \mathcal{E}(\vec{N}_A)$ is defined. At the same time, we also denote $\vec{N}_A \quad \mathcal{E}^{-1}(\vec{N}_B)$. Thus, $\mathcal{E}(\{\vec{n}_x\})$ ($\mathcal{E}(\{N_x\})$ equivalently) is the set of intervals that succeed interval $\vec{n}_x$. In a TFG, if between nodes N_x and N_y, there is a flow path which contains only δ nodes, transit nodes and/or virtual intermission nodes, then we have $\vec{n}_y \in \mathcal{E}(\{\vec{n}_x\})$. If $\vec{n}_y \in \mathcal{E}(\{\vec{n}_x\})$, $\vec{n}_x \in \mathcal{E}^{-1}(\{\vec{n}_y\})$ is also true. In a scenario, at least one vector $\vec{n}_x$ has $\mathcal{E}^{-1}(\{\vec{n}_x\}) = \phi$. Notice that $\mathcal{E}$ is defined on the power set of $\vec{N}$. In Fig. 3.11(a) for example, $\mathcal{E}(\{V1\}) = \{T2, A3\}$, and $\mathcal{E}^{-1}(\{T2, A3\}) = \mathcal{E}^{-1}(\{T2\}) = \{T1, V1, A2\}$.

3.3.3 Properties of the TFG Models

Although an explicit relation can be specified between every two intervals or even among a group of them, there is usually no guarantee that the relation involved in or inferred from the multiple intervals will be consistent [1]. In the application, without the precise temporal knowledge about the involved intervals, temporal conflicts may occur in the scenario specifications. In order to avoid the conflicts of the temporal relationships, the following requirements are imposed on the multimedia scenarios.

R1: When $\vec{n}_x$ is involved in $r \in R_f$, in the specification, only one interval $\vec{n}_y$ is referred and only one r is required as $\vec{n}_y\,(r)\,\vec{n}_x$, where $\vec{n}_y$ is already specified in the scenario.

R1 prevents the possible conflicts caused by the specifications of the parallel relations, such as $\vec{n}_x(r_1)\vec{n}_y$, $\vec{n}_y(r_2)\vec{n}_x$ and $\vec{n}_x(r_3)\vec{n}_x$. For an interval $\vec{n}$, only the specified relation is recorded in its $\vec{n}.F_s$ and $\vec{n}.F_e$. The object display schedule will also be derived according to this specified relation [26]. Meanwhile, given the specification $\{A\}B$ and $<A> B$, R1 is the way in which the intervals contained in set B are related.

R2: If $|\mathcal{E}^{-1}(\{\vec{n}_x\})| > 1$, then before reach $\vec{n}_x$, , $\forall \vec{n}_y \in \mathcal{E}^{-1}(\{\vec{n}_x\})$ pass through a

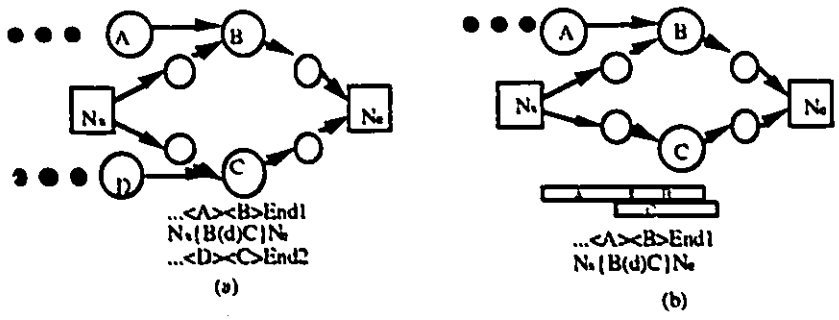


Figure 3.14: Temporal Conflicts in TFG models

common transit node N_{tx}. That is, $\forall \vec{n}_y \in \mathcal{E}^{-1}(\{\vec{n}_x\})$, $N_{tx} \in \mathcal{S}(\Delta_y) \cup \mathcal{S}(N_y)$.

R2 ensures that $\vec{n}_x$ can only have one sequentially determined position (i.e., the starting time of $\vec{n}_x$). In the TFG, this position is marked by the transit node, when $\vec{n}_x$ is preceded by multiple intervals.

R3: If $\vec{n}_x(r)\vec{n}_y$, $r \in R_f \cup R_m$, then either $\mathcal{E}^{-1}(\{\vec{n}_x\}) = \mathcal{E}^{-1}(\{\vec{n}_y\})$ holds or one of them is ϕ.

R3 prevents the conflicts between the sequential and the parallel relations. For instance, in Fig. 3.14(a), we have $\mathcal{E}^{-1}(\{C\}) = \{D\} \neq \mathcal{E}^{-1}(\{B\}) = \{A\}$. Then $< A >< B >$, $B(d)C$ and $< D >< C >$ may not be able to hold at the same time. On the other hand, the scenario in Fig. 3.14(b) presents no conflicts between $< A >< B >$ and $B(d)C$. Take $\tilde{\mathcal{P}}(\vec{n}_x) = (\mathcal{P}(N_x) \cup \mathcal{P}(\Delta_x)) - \{\delta$ nodes and virtual intermission nodes$\}$. If $\vec{n}_x(r)\vec{n}_y$, $r \in R_f \cup R_m$ and $|\tilde{\mathcal{P}}(\vec{n}_x) \cup \tilde{\mathcal{P}}(\vec{n}_y)| \leq 1$, then $\mathcal{E}^{-1}(\{\vec{n}_x\}) = \mathcal{E}^{-1}(\{\vec{n}_y\})$ is held. When $|\tilde{\mathcal{P}}(\vec{n}_x) \cup \tilde{\mathcal{P}}(\vec{n}_y)| > 1$, without R3, the conflicts can be involved.

The relation $r \in R_f \cup R_m - \{e, ch\}$ is recorded in $\vec{n}.F_s$ and/or $\vec{n}.F_e$ of every involved vector $\vec{n}$. According to the definition of $r =$ 'ch', between every two involved intervals, the relation $r \in R_f$ may only be recorded in one of them. On the other hand, relation $r =$ 'e' is not reflected in the $\vec{n}.F_s$ (or $\vec{n}.F_e$) of any involved $\vec{n}$. In the following, we formally prove that TFG models represent all the relations defined among the intervals.

Theorem 3.1 *If $\vec{n}_x \in \vec{N}$ is involved in any relation $r \in R_m \cup R_f$, then the relation r is represented by the TFG model.*

Proof: Form the set $\vec{N}_s = \{\vec{n}_x, \vec{n}_{y_1}, \vec{n}_{y_2}, \cdots, \vec{n}_{y_s}\} \subset \vec{N}$, s.t. $(\mathcal{P}(\Delta_x) \cup \mathcal{P}(N_x)) \cap (\mathcal{P}(\Delta_{y_1}) \cup \mathcal{P}(N_{y_1})) \cap \cdots \cap (\mathcal{P}(\Delta_{y_s}) \cup \mathcal{P}(N_{y_s})) \neq \phi$ and $(\mathcal{S}(\Delta_x) \cup \mathcal{S}(N_x)) \cap (\mathcal{S}(\Delta_{y_1}) \cup \mathcal{S}(N_{y_1})) \cap \cdots \cap (\mathcal{S}(\Delta_{y_s}) \cup \mathcal{S}(N_{y_s})) \neq \phi$. Given a $\vec{n}_x$, there may be several such $\vec{N}_s$'s. For every $\vec{N}_s$, we have the following two possible cases:

1) If $\exists \vec{n} \in \vec{N}_s$ s.t., $(\vec{n}.F_s \cup \vec{n}.F_e) \cap \vec{N}_s = \phi$, relation $r =$ 'e' or $r =$ 'ch' exists among the vectors $\vec{n} \in \vec{N}_s$ (e.g., Fig. 3.12 and Fig. 3.13);

2) If case 1) does not stand, $\vec{n}_x$ is only involved in $r \in R_f \cup R_m - \{e, ch\}$ with vectors $\vec{n}_{y_i} \in \vec{N}_s$.

Now we discuss each case.

(1) $r \in R_f \cup R_m - \{e, ch\} = \{d, f, s, o, oi\}$

If any $\vec{n}_y \in \vec{N}_s$ has a certain relation with $\vec{n}_x$, $(\{\vec{n}_y\} \cap \vec{n}_x.F_s) \cup (\{\vec{n}_y\} \cap \vec{n}_x.F_e) \neq \phi$ should hold. Moreover, if $\vec{n}_y \in \vec{n}_x.F_s$ & $\vec{n}_y \in \vec{n}_x.F_e$, the relation between $\vec{n}_x$ and $\vec{n}_y$ is $\vec{n}_x(d)\vec{n}_y$; if $\vec{n}_y \in \vec{n}_x.F_s$ & $\vec{n}_y \notin \vec{n}_x.F_e$, the relation is $\vec{n}_x(f)\vec{n}_y$; if $\vec{n}_y \notin \vec{n}_x.F_s$ & $\vec{n}_y \in \vec{n}_x.F_e$, the relation is

$$\begin{cases} \vec{n}_x(s)\vec{n}_y & \mathcal{P}(N_x) \cap \mathcal{P}(N_y) \neq \phi; \\ \vec{n}_x(o)\vec{n}_y & \mathcal{P}(N_x) \cap \mathcal{P}(N_y) = \phi \ \& \ \mathcal{P}(N_y) \text{ is a virtual intermission node;} \\ \vec{n}_y(o)\vec{n}_x & \mathcal{P}(N_x) \cap \mathcal{P}(N_y) = \phi \ \& \ \mathcal{P}(N_x) \text{ is a virtual intermission node.} \end{cases}$$

Therefore, looking into $\vec{n}_x.F_s$ and $\vec{n}_x.F_e$, the temporal relation between $\vec{n}_x$ and any other vectors in $\vec{N}_s$ can be determined.

(2) $r =$ 'ch' or $r =$ 'e' is involved in $\vec{N}_s$.

case A: $(\vec{n}_x.F_s \cup \vec{n}_x.F_e) \cap \vec{N}_s = \phi$, then for $\forall \vec{n}_z \in \vec{N}_s$ s.t. $(\vec{n}_z.F_s \cup \vec{n}_z.F_e) \cap \vec{N}_s = \phi$, $\vec{n}_x(e)\vec{n}_z$ holds. For $\vec{n}_y \in N_s$ s.t. $(\vec{n}_y.F_s \cap \{\vec{n}_x\}) \cup (\vec{n}_y.F_e \cap \{\vec{n}_x\}) \neq \phi$, we have the following results. If $\vec{n}_x \in \vec{n}_y.F_s$ & $\vec{n}_x \in \vec{n}_y.F_e$, the relation between $\vec{n}_x$ and $\vec{n}_y$ is $\vec{n}_y(d)\vec{n}_x$; if $\vec{n}_x \in \vec{n}_y.F_s$ & $\vec{n}_x \notin \vec{n}_y.F_e$, the relation is $\vec{n}_y(f)\vec{n}_x$; if $\vec{n}_x \notin \vec{n}_y.F_s$ & $\vec{n}_x \in \vec{n}_y.F_e$: $r =$'s', $r =$ 'o' or $r =$ 'oi' can be identified, as we did above.

case B: $(\vec{n}_x.F_s \cup \vec{n}_x.F_e) \cap \vec{N}_s \neq \phi$, assume it is $\vec{n}_y \in \vec{N}_s$ s.t. $(\vec{n}_y.F_s \cup \vec{n}_y.F_e) \cap \vec{N}_s = \phi$.

— If $(\vec{n}_x.F_s \cap \{\vec{n}_y\}) \cup (\vec{n}_x.F_e \cap \{\vec{n}_y\}) \neq \phi$, then by comparing $\vec{n}_y$, $\vec{n}_x.F_s$ and $\vec{n}_x.F_e$, the relation between $\vec{n}_x$ and $\vec{n}_y$ can be identified.

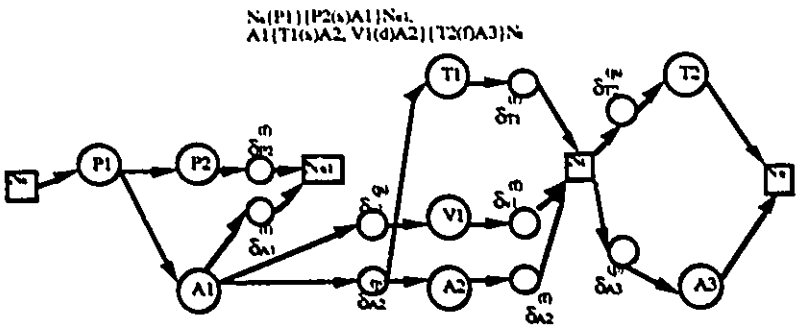


Figure 3.15: The TFG model of the scenario example

Meanwhile, if $\exists \vec{n} \in \vec{N}_s$ s.t. $(\{\vec{n}_x\} \cap \vec{n}.F_s) \cup (\{\vec{n}_x\} \cap \vec{n}.F_e) \neq \phi$, the relation between $\vec{n}_x$ and $\vec{n}$ can be determined similarly, by comparing $\vec{n}_x$, $\vec{n}.F_s$ and $\vec{n}.F_e$.

— If $(\vec{n}_x.F_s \cap \{\vec{n}_y\}) \cup (\vec{n}_x.F_e \cap \{\vec{n}_y\}) = \phi$, then for $\vec{n} \in (\vec{n}_x.F_s \cup \vec{n}_x.F_e)$, the relation between $\vec{n}_x$ and $\vec{n}$ can be identified.

Examine the other $\vec{N}_s$'s of $\vec{n}_x$ in the same way, all the relations of $r \in R_n \cup R_f$ that $\vec{n}_x$ is involved in can be identified. —Q.E.D.

Now all the relations of $r \in R_m \cup R_f$ that $\vec{n}_x$ is involved in are represented by the TFG model. Since the transit nodes explicitly represent the sequential requirements, all the relations defined are represented by the TFG.

3.4 Temporal Structure of Time-Flow Graph

In last section we presented the temporal model, i.e., the TFG (Time-Flow Graph) model for collaborative multimedia presentation scenarios. For example, Fig. 3.15 shows the TFG and the specifications of the scenario given in Fig. 3.1. In this scenario, picture $P1$ will start first. $P2$ and audio $A1$ start simultaneously. Then $A2$ is displayed during video $V1$ or vice versa (see Fig. 3.1(b)), depending on their display durations. In either case, $T1$ has to start at the same time with $A2$. Finally, $T2$ and $A3$ finish together. The scheduling algorithm to be presented are based on the TFG models.

3.4.1 Activity Occurrence and Interval Reference

Although various temporal relationships can be involved in a scenario, the major issue of scheduling is to select the starting time for the presentation of each object. The starting time of an object, which is related to the other objects, may require synchronization operation among the sources. Thus, the starting time that involves multiple objects should be captured from the TFG model.

Definition 3.3 *In a scenario, the interval set $\vec{N}$ is decomposed into a set of ACs (Activity oCcurrence). An AC_i is constructed by two disjoint sets $\Lambda_i^p \neq \phi$ and Λ_i^s, i.e., $AC_i = \Lambda_i^p \cup \Lambda_i^s$. Since $|\Lambda_i^p| \geq 1$, given $\vec{n}_i \in \Lambda_i^p$, we define*

$$\Lambda_i^p = \{\vec{n}_i\} \cup \{\vec{n}_x : \vec{n}_x(r)\vec{n}_y, \vec{n}_y \in \Lambda_i^p, r \in R_f \cup R_m\} \text{ and,}$$

$$\Lambda_i^s = \{\vec{n}_z : \nexists \vec{n}_v \in \vec{N} \text{ s.t. } \vec{n}_z(r)\vec{n}_v, r \in R_f \cup R_m \\ \& \ \mathcal{E}^{-1}(\{\vec{n}_z\}) = \{\vec{n}_u\} \text{ where } \vec{n}_u \in \Lambda_i^p \cup \Lambda_i^s \ \& \ \vec{n}_u.S_r = \vec{n}_z.S_r\}$$

Recall that $\mathcal{E}^{-1}(\{\vec{n}_x\})$ denotes the set of intervals by which $\vec{n}_x$ is preceded in a TFG. In the scenario shown in Fig. 3.16, for instance, $AC_1 = \Lambda_1^p = \{A1, A2\}$, $\Lambda_2^p = \{P1\}$ and $\Lambda_2^s = \{P3\}$, $AC_3 = \Lambda_3^p = \{P2\}$, assume the pictures are from the same source.

From the definition, $\forall \vec{n}_x \in \vec{N}$ are divided into the ACs. An AC distinguishes a starting time when either different sources are involved, or a new group of concurrent intervals and their temporal relations are required. In AC_i, the Λ_i^p contains the concurrent intervals and the Λ_i^s includes the sequential ones.

Proposition 3.1 *An interval $\vec{n}_x \in \vec{N}$ can only be in one AC.*

Proof: Suppose $\vec{n}_x \in AC_i$ & $\vec{n}_x \in AC_j$, where $AC_i \neq AC_j$.

From the definition, it is impossible that $\vec{n}_x \in \Lambda_i^p$ & $\vec{n}_x \in \Lambda_j^s$, neither $\vec{n}_x \in \Lambda_i^s$ & $\vec{n}_x \in \Lambda_j^p$. If $\vec{n}_x \in \Lambda_i^p$ & $\vec{n}_x \in \Lambda_j^p$ assumed, then we have $\Lambda_i^p = \Lambda_j^p \Longrightarrow \Lambda_i^s = \Lambda_j^s$, i.e., $AC_i = AC_j$.

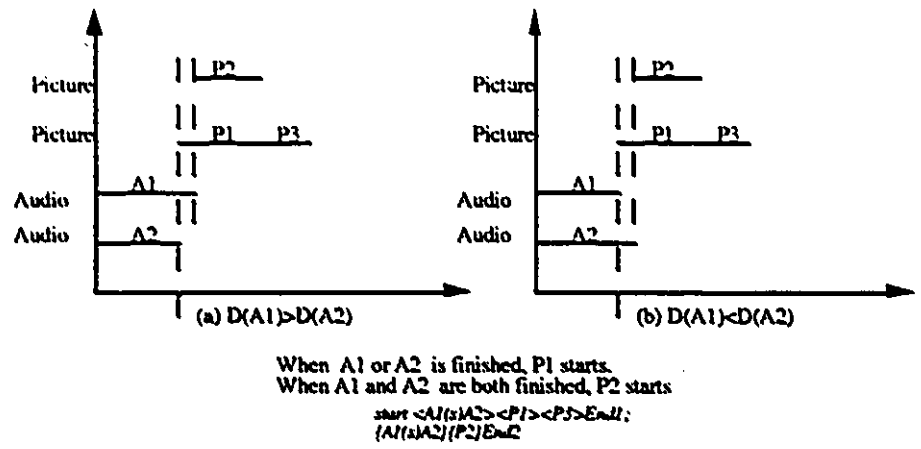


Figure 3.16: Scenario example 2

Finally, if $\vec{n}_x \in \Lambda_i^s$ & $\vec{n}_x \in \Lambda_j^s$, then for $\vec{n}_u \in \mathcal{E}^{-1}(\{\vec{n}_x\})$, $\vec{n}_u \in \Lambda_i^p \cup \Lambda_i^s$ & $\vec{n}_u \in \Lambda_j^p \cup \Lambda_j^s$. From the above, it is only possible that $\vec{n}_u \in \Lambda_i^s$ & $\vec{n}_u \in \Lambda_j^s$. Look into $\mathcal{E}^{-1}(\{\vec{n}_u\})$ and then $\mathcal{E}^{-1}(\mathcal{E}^{-1}(\{\vec{n}_u\}))$, ..., etc. Eventually, there is a $\vec{n} \in \Lambda_i^s$ & $\vec{n} \in \Lambda_j^s$, such that $\mathcal{E}^{-1}(\{\vec{n}\}) = \{\vec{n}_s\}$, and only one of the following three cases can hold for $\vec{n}_s$. That is, either $\vec{n}_s \in \Lambda_i^p$ & $\vec{n}_s \in \Lambda_j^p$, or $\vec{n}_s \in \Lambda_i^p$ & $\vec{n}_s \in \Lambda_j^s$, or $\vec{n}_s \in \Lambda_i^s$ & $\vec{n}_s \in \Lambda_j^p$. However, none of these cases is possible.

Therefore, $\vec{n}_x$ can only be in one AC. — Q.E.D.

The AC that $\vec{n}_x$ is contained in is denoted by $A(N_x)$ or $A(\vec{n}_x)$, equivalently. For example, if $\vec{n}_x$ is in AC_i, then $A(\vec{n}_x) = i$. In a given scenario, which has n ACs all together, the set of ACs, i.e., $\mathcal{L} = \{AC_0, AC_1, \cdots, AC_n\}$ forms a partition of $\vec{N}$. That is, $\bigcup_{i=0}^{n} AC_i = \vec{N}$, $AC_i \cap AC_j = \phi$, for all $AC_i, AC_j \in \mathcal{L}$, $i \neq j$. Take AC_0 as the first AC in a scenario, then $\forall \vec{n} \in \Lambda_0^p$, $\mathcal{E}^{-1}(\{\vec{n}\}) = \phi$.

For every AC, a reference is defined in order to infer the sequential position of the AC in the TFG. Usually, without the knowledge of $\mathcal{D}(\vec{n}_x), \forall \vec{n}_x \in \vec{N}$, it is difficult to identify a precise sequential relationship, i.e., whether $<\vec{n}_y><\vec{n}_x>$ holds, between two intervals $\vec{n}_x$ and $\vec{n}_y$. It can only be inferred that in the TFG model, $\vec{n}_y \in \mathcal{E}^{-1}(\{\vec{n}_x\})$ may "meet" $\vec{n}_x$. Thus, according to the sequential ordering reflected by $\mathcal{E}^{-1}(\{\vec{n}_x\})$, a subset

$REF_i^c \subset \vec{N}$ is formed for $\vec{n}_x$:

$$REF_i^c \subseteq \{\vec{n}_y : \vec{n}_y \in \mathcal{E}^{-1}(\{\vec{n}_x\}), \vec{n}_y \notin AC_i, \vec{n}_x \in AC_i\}.$$

Based on REF_i^c, the following definition for the "reference vector" is given.

Definition 3.4 *The REFerence vector REF_i of AC_i is defined as*

$$REF_i^c \supseteq REF_i = \{\vec{n}_y : <\vec{n}_y><\vec{n}_x>, \vec{n}_y \in REF_i^c, \vec{n}_x \in AC_i \ \& \ \vec{n}_x \in \mathcal{E}(\{\vec{n}_y\})\},$$

and $REF_0^c = REF_0 = \phi$.

Proposition 3.2 *The number of intervals in REF_i can be limited to $|REF_i| = 1$.*

Proof: First, according to the requirement R3 in section 3.3.3 [25], if $\vec{n}_x(r)\vec{n}_z$, then either $\mathcal{E}^{-1}(\{\vec{n}_x\}) = \mathcal{E}^{-1}(\{\vec{n}_z\})$ holds or one of them is ϕ. Therefore for a given AC, say AC_i, there can be only one REF_i^c. As required in R2 in the previous section [25], $\forall \vec{n} \in \mathcal{E}^{-1}(\{\vec{n}_x\})$ have to pass through one transit node N_{tx}. The transit node N_{tx} is used to represent the sequential specification $\{A\}B$ or $<A>B$. According to the definitions of the sequential relations given in section 3.3.3, in either case, a unique starting time of intervals contained in B is defined. That is, if $\{A\}B$, then $<\vec{n}_y><\vec{n}_x>$. Here $\vec{n}_y$ is the last to finish its display among the intervals contained in A and $\vec{n}_x$ is the first to start its display among the intervals contained in B. If $<A>B$, we also have $<\vec{n}_y><\vec{n}_x>$ where $\vec{n}_y$, in this case, is the first to be finished in A. Since there is only one starting time of the $\vec{n}_x$ which is the first to start in B, more intervals in REF_i will not change this temporal position. We can thus have $|REF_i| = 1$ — Q.E.D.

3.4.2 Aggregated Time-Flow Graph (ATFG) for Scenarios

From the Theorem 3.1 given in last section, all the relations that an interval $\vec{n}_x$ is involved are represented. Thus, for the convenience of description, we assume algorithms $\vec{T}_i(\vec{N}, \vec{n}_x)$ and $\vec{T}_r(\vec{N}, \vec{n}_x, \vec{N}_s)$. $\vec{T}_i(\vec{N}, \vec{n}_x)$ returns all the intervals that are related to $\vec{n}_x$ in the scenario $\vec{N}$. The algorithm $\vec{T}_r(\vec{N}, \vec{n}_x, \vec{N}_s)$ produces the relation(s) between $\vec{n}_x$ and the vectors in

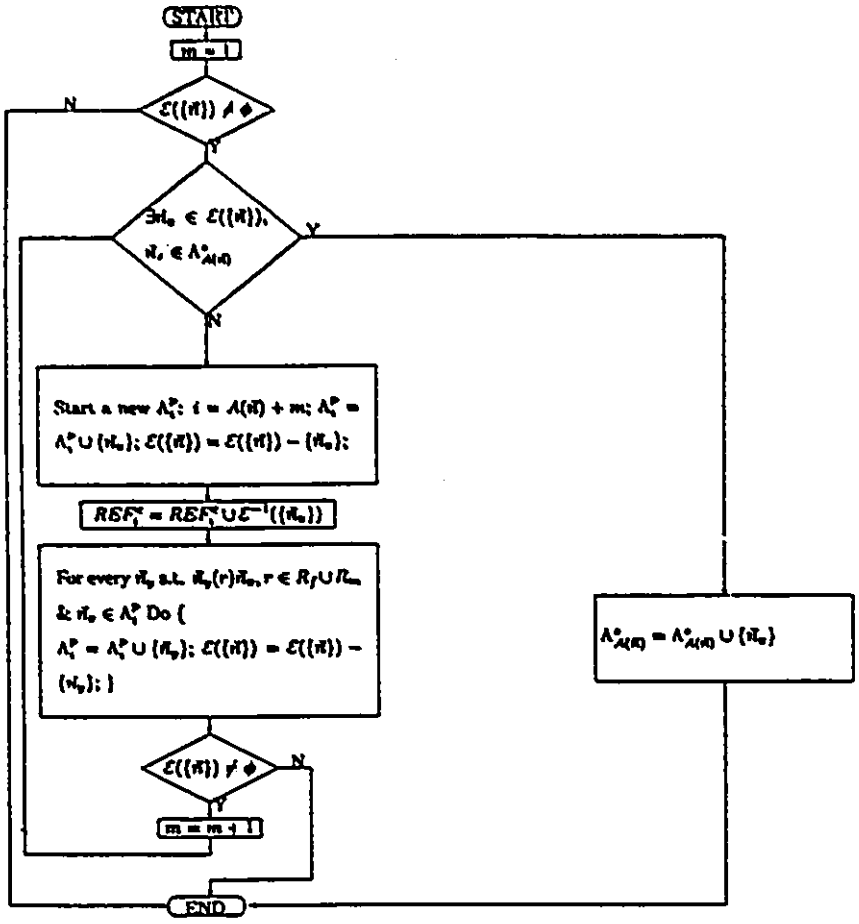


Figure 3.17: Generating AC for $\mathcal{E}(\{\vec{n}\})$

subset $\vec{N}_s \subseteq \vec{N}$, specified in the scenario $\vec{N}$. For instance, given the scenario shown in Fig. 3.15, $\tilde{T}_i(\vec{N}, A2) = \{T1, V1\}$, and $\tilde{T}_r(\vec{N}, A2, \{V1\}) = \{d\}$

At node n in the TFG of a scenario, Fig. 3.17 shows how to obtain the ACs for nodes $n_i \in \mathcal{E}(\{\vec{n}\})$. Applying the recursion technique, the algorithm **Establish**($\vec{n}$) generates all the ACs and their corresponding REF^c for a scenario, with N_s as the initial value of $\vec{n}$. The N_s is the starting transit node in the TFG model.

Establish($\vec{n}$)

Begin

$:n = 0$;

If the node is N_s Begin

$REF_0 = \phi$;

While ($\mathcal{E}(\{N_s\}) \neq \phi$)

Begin

While($\Lambda^p_m \neq \phi$) $m = m + 1$;

Take $\vec{n}_i \in \mathcal{E}(\{N_s\})$

While ($\Lambda^p_m = \phi$) or ($\exists \vec{n}_k \in \Lambda^p_m$ & $\exists \vec{n}_i \in \tilde{T}_i(\vec{N}, \vec{n}_k)$ & $\vec{n}_i \notin \Lambda^p_m$)

Begin

$A(\vec{n}_i) = m$;

$\Lambda^p_m = \Lambda^p_m \cup \{\vec{n}_i\}$;

$\mathcal{E}(\{N_s\}) = \mathcal{E}(\{N_s\}) - \{\vec{n}_i\}$;

Establish($\vec{n}_i$);

End /* While */

End /* While */

End /*If */

Else While ($\mathcal{E}(\{\vec{n}\}) \neq \phi$)

Begin

If $|\mathcal{E}(\{\vec{n}\})| > 1$ or $\exists \vec{n}_x \in \mathcal{E}(\{\vec{n}\})$ & $\vec{n}_x.S_r \neq \vec{n}.S_r$

or $|\mathcal{E}^{-1}(\{\vec{n}_x\})| > 1$ or $\vec{n}_x$ is involved in $r \in R_f \cup R_m$ — (1)

Begin

While($\Lambda^p_{A(\vec{n})+m} \neq \phi$) $m = m + 1$;

Take $\vec{n}_i \in \mathcal{E}(\{\vec{n}\})$;

While ($\Lambda^p_{A(\vec{n})+m} = \phi$) or ($\exists \vec{n}_k \in \Lambda^p_{A(\vec{n})+m}$ & $\exists \vec{n}_i \in \tilde{\mathcal{T}}_i(\vec{N}, \vec{n}_k)$
& $\vec{n}_i \notin \Lambda^p_{A(\vec{n})+m}$)

Begin — (2)

$A(\vec{n}_i) = A(\vec{n}) + m$;

$REF^c_{A(\vec{n}_i)} = \mathcal{E}^{-1}(\{\vec{n}_i\}) \cup REF^c_{A(\vec{n}_i)}$;

$\Lambda^p_{A(\vec{n}_i)} = \Lambda^p_{A(\vec{n}_i)} \cup \{\vec{n}_i\}$;

For every $\vec{n}_j \in \mathcal{E}^{-1}(\{\vec{n}_i\})$:

$\mathcal{E}(\{\vec{n}_j\}) = \mathcal{E}(\{\vec{n}_j\}) - \{\vec{n}_i\}$; — (3)

Establish($\vec{n}_i$);

End /* While * /

End /* If */

Else Begin

Take $\vec{n}_i \in \mathcal{E}(\{\vec{n}\})$;

$\Lambda^s_{A(\vec{n})} = \Lambda^s_{A(\vec{n})} \cup \{\vec{n}_i\}$; — (4)

Establish($\vec{n}_i$);

End /* Else */

End /* While */

End

In the algorithm at line (1), a new AC is generated. In the loop starting from line (2), Λ^p is constructed. The line (3) ensures that a vector $\vec{n}_x$ would be contained in only one *AC*. At line (4), the $\Lambda^s_{A(\vec{n})}$ is built.

For the scenario illustrated in Fig. 3.15, the *AC*s and their REF^cs generated from executing the algorithm **Establish($\vec{n}$)** are given below:

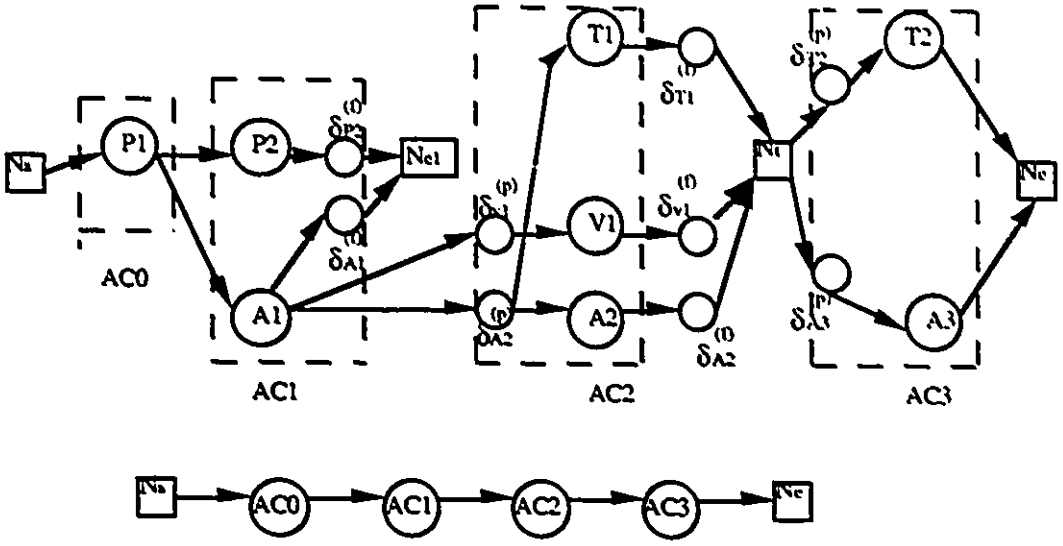


Figure 3.18: The aggregated time-flow graph: example 1

i	Λ_i^P	REF_i^c
0	$\{P1\}$	ϕ
1	$\{P2, A1\}$	$\{P1\}$
2	$\{V1, A2, T1\}$	$\{A1\}$
3	$\{T2, A3\}$	$\{T1, V1, A2\}$

In a scenario, after all the ACs are obtained, a subset of nodes which contains $\forall \vec{n} \in AC_i$ and their connecting edges can be replaced by one composite AC node. Given AC_i and AC_j, if $\exists \vec{n}_x \in REF_j^c$ & $\vec{n}_x \in AC_i$, then an edge is directed from AC_i to AC_j. Thus, by decomposing the vector nodes into ACs, an Aggregated Time-Flow Graph ($ATFG$) can be formed. In Fig. 3.18, the $ATFG$ of the scenario given in Fig. 3.1 (TFG in Fig. 3.15) is presented. If there is an edge from AC_i to AC_j, it is also denoted as $AC_j \in \mathcal{E}(\{AC_i\})$, $AC_i \in \mathcal{E}^{-1}(\{AC_j\})$. In Fig. 3.19, the $ATFG$ of the scenario shown in Fig. 3.16 is given. In this example, we have $m > 1$ in computing the ACs. Notice the difference between the $ATFGs$ in Fig. 3.18 and Fig. 3.19.

Proposition 3.3 *Given a scenario (Δ_N, N_t, E_d), and the partition of $\vec{N}$ as $\mathcal{L} = \{AC_0, AC_1, \cdots, AC_n\}$, if $|REF_i| = 1$, then the ATFG of the scenario is a tree.*

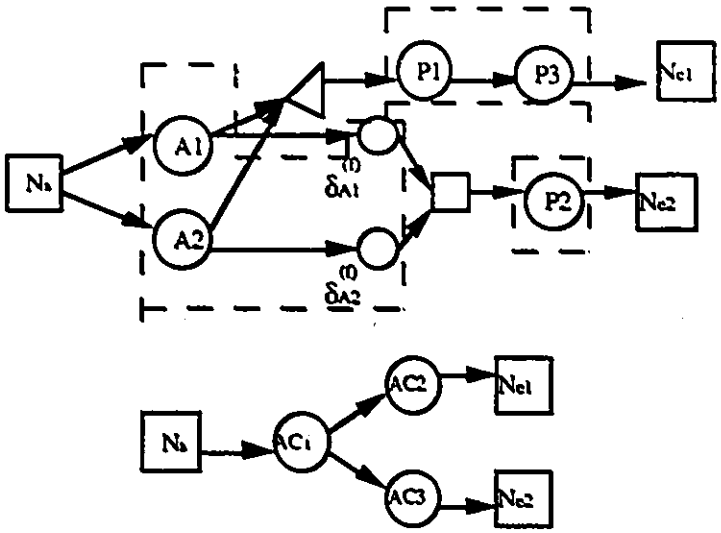


Figure 3.19: The aggregated time-flow graph: example 2

Proof: Because there is no cycle in a TFG (section 3.2), there cannot be any cycle in the $ATFG$.

Since $REF_0 = \phi$, we have $\mathcal{E}(\{AC_0\}) = \phi$. AC_0 is taken as the root node of the tree.

For the other $AC_i \in \mathcal{L}$, since $|REF_i| = 1$, assume $REF_i = \{\vec{n}\}$. As $\vec{n}$ can be in one and only one AC, $\mathcal{E}^{-1}(\{AC_i\}) = \{AC_{A(\vec{n})}\}$ is obtained. That is, $AC_{A(\vec{n})}$ and only $AC_{A(\vec{n})}$ is connected to AC_i by a directed edge. Therefore, the $ATFG$ is a tree. — Q.E.D.

3.5 TPC Scheduling Algorithm

3.5.1 Real-Time Multimedia Integration

In real-time multimedia remote presentation, e.g., the application of teleorchestra, activities such as multimedia object delivery and display are supposed to be performed continuously, in real-time. The solution for this real-time data integration is that all the sources which are supplying the objects for the orchestration participate in the temporal integration, by scheduling their object delivery [28]. Thus synchronization among the independent sources is required. The TPC scheduling algorithm proposed here is designed

for this purpose.

We can observe that the integration of the temporal relationships may have to be implemented in a distributed manner. A centralized scheduling center may be easy to design. However, the center can be the bottleneck of the system. Besides, whether a database server is congested at a given time or the data object required is locked by some other applications (e.g., for concurrency control in joint editing [17]), also have to be considered in a network connection situation. Recently, collaborative environment designs (e.g., DiCE [6]) often aim to a hub-free (fully distributed) architecture. Thus, the scheduling algorithm may have to be executed in a distributed manner, taking the autonomy of the sources into account.

The algorithm designed here can be performed in a fully distributed manner. With respect to the autonomy of the involved sources, we intend to support the synchronization dialogue, including the temporal inference and schedule derivation in a distributed manner over the network. Based on the Time-Flow Graph (TFG) models proposed for the scenarios used in the teleorchestra, it is assumed that much of our temporal knowledge is relative and distributed in the system. In this case, we provide a synchronization model so that the independent sources can negotiate the remote temporal presentation of the multimedia data. User's interactions on the presentation (e.g., pause, fast-forwarding and fast-rewinding) are accommodated as well.

3.5.2 The Algorithm

After the execution of the algorithm, the following three conditions should be satisfied.

C1: After the execution of the algorithm, the display schedules for all the objects are obtained.

C2: The server should be scheduled in accordance with the sequential specifications in the scenario. Therefore, no jitters nor gaps [41] would occur in the source's schedules. In Fig. 3.20(a), the occurrence of jitters in the schedule forces the presentation of the object *audio*2 to start before the previous object (*audio*1) can be finished. The play-out of *audio*1 is hence garbled, against the requirement in the scenario. On the other hand,

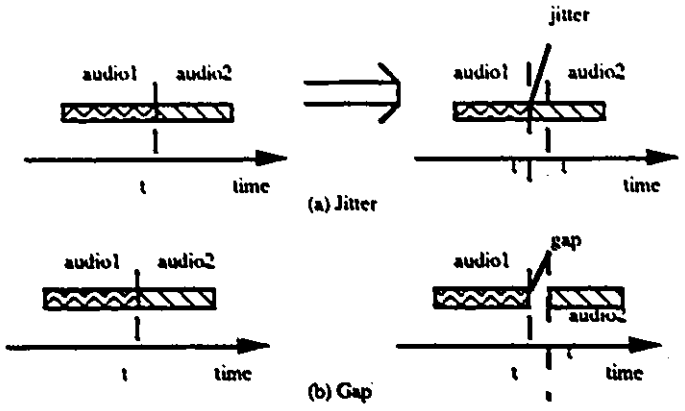


Figure 3.20: Gaps and jitters in the schedules

in Fig. 3.20(b), the gap in the schedule introduces an extra intermission (gap) in the presentation which is not specified in the scenario.

C3: The parallel relation among the intervals should be scheduled.

Condition 1 states that all the data objects required in the scenario should be scheduled for display. The second condition prevents jitters and gaps from the schedules of the object display. C3 indicates that the "matching" of the multimedia objects in the scenario, which the involved sources are responsible for, should be maintained. The scheduling consists of three major steps, as depicted in Fig. 3.21. It is actually a synchronization dialogue among the multiple sources. In the following, we describe each step.

1. Local scheduling

First, the algorithm **Establish**($\vec{n}$) is executed. The algorithm can be performed at any site which holds the scenario generated from the application. Then the results will be sent to all the sources involved in the integration. If the scenario is broadcast, **Establish**($\vec{n}$) can be executed at every source distributedly.

Then every source performs local scheduling. This is the step 2. The schedules for the objects that belong to S_i are produced. At the end of step 2, source S_i will have the following results:

t_m^{ij} : the schedule of $\vec{n}_j$ given by source S_i, i.e., $\vec{n}_j.S_r = S_i$, and $\vec{n}_j \in AC_m$;

$t_k^i = t_m^{ij} + D_m^j$: the finishing time of $\vec{n}_j$ given by source S_i that "owns" $\vec{n}_j \in REF_k^c$,

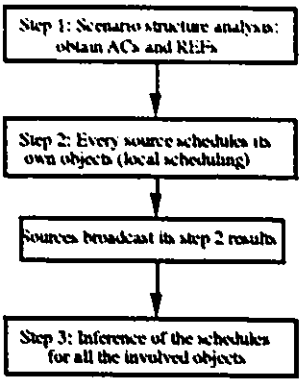


Figure 3.21: Scheduling among the sources

$\vec{n}_j \in AC_m$. D_m^j is the display duration of object j in AC_m. The display duration includes the play-out time $\mathcal{D}(j)$ of the object, and/or the holding time required for a static media (e.g., text, image). The case in which a certain object has no predictable display duration, e.g., a program to be discussed, will be addressed later. Here D_m^j is assumed for every $\vec{n}_j$.

For instance, given $\vec{n}_j \in AC_m$, $\vec{n}_i.S_r = S_i$, the source S_i can make the local schedule as:

If S_i congested or $\vec{n}_j.O_{id}$ is locked

t_m^{ij} =the earliest possible time;

else $t_m^{ij} = 0$; /* S_i schedules its object contained in Λ_m^p */

While ($\exists N_l \in \mathcal{S}(N_j)$ & $\vec{n}_l \in \Lambda_m^s$) /* Sequential objects in Λ_m^s */

Begin

$t_m^{il} = t_m^{ij} + D_m^j$;

If object $\vec{n}_l$ has to be delayed until $t_m^{il} + \Delta\tau$

For $\forall \vec{n}_j \in \Lambda_m^p \cup \Lambda_m^s$ & $\vec{n}_j.S_r = S_i : t_m^{ij} = t_m^{ij} + \Delta\tau$;

$\vec{n}_j = \vec{n}_l$;

End /* while */

After step 2, source S_i broadcasts a message containing all its t_m^{ij}, D_m^j and $t_k^i = t_m^{ij} + D_m^j$ (if $\vec{n}_j \in REF_k^c$), for $\forall AC_m \in \mathcal{L}$. Meanwhile, if $\mathcal{D}(\vec{n}_\tau)$ of an intermission τ is specified by referring to a certain object X, e.g., "at the end of the second page of object X", the

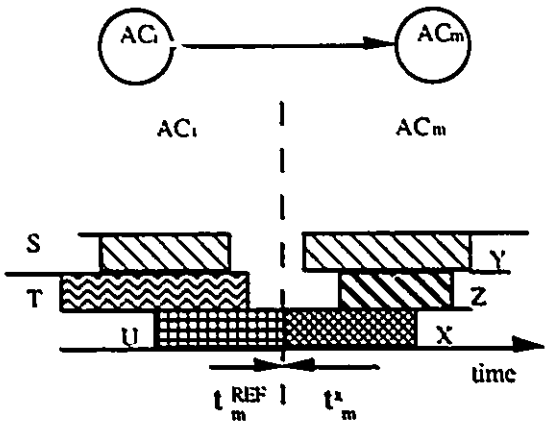


Figure 3.22: Schedule inference

owner of object X also provides the value of $\mathcal{D}(\vec{n}_\tau)$. Given M sources in the integration, altogether $M(M-1)$ messages are generated in the system. After source S_i receives the messages from all the other sources, it starts step 3. Now, in the TFG models of $\vec{n}_x(o)\vec{n}_y$ ($\vec{n}_y(oi)\vec{n}_x$ equivalently), the virtual intermission $\vec{n}_\tau$ is composed into node N_y such that $D(\vec{n}_y) = D(\vec{n}_\tau) + D(\vec{n}_y)$. Therefore, the relation $r =$ 'o' and $r =$ 'oi' will be differentiated from $r =$'s' at the source locally, through the interval τ.

2. Inference of the interval sequence

In step 3, before the schedule inference, a subset $\vec{N}_f^m$ of intervals is constructed for every $AC_m \in \mathcal{L}$, as shown below. The set $\vec{N}_f^m$ contains all the objects $\vec{n}_x \in AC_m$ s.t. $\vec{n}_u(m)\vec{n}_x$ where $\vec{n}_u \in REF_m$. Given $AC_m = \mathcal{E}(\{AC_i\})$, Fig. 3.22 manifests the schedule inference. In Fig. 3.22, t_m^{REF} is the finishing time of interval $\vec{n}_u \in AC_i$, and $\vec{n}_u \in REF_m$. For the intervals in $\vec{N}_f^m$ (e.g., $\vec{n}_x$ in Fig. 3.22), t_m^{REF} is then also their starting time that is determined sequentially by the preceding presentation intervals. The schedules for other intervals $\vec{n}_i \in AC_m$ & $\vec{n}_i \notin \vec{N}_f^m$ (e.g., $\vec{n}_y$, $\vec{n}_z$ in Fig. 3.22) can be derived in accordance with their relations to the interval $\vec{n}_x \in \vec{N}_f^m$. Now, we construct the $\vec{N}_f^i$ for a given AC_i. First, vectors $\vec{n} \in \Lambda_i^p$ are selected to form subset $\vec{S}_i$, s.t. $\vec{S}_i = \{\vec{n} \in \Lambda_i^p \ \& \ \mathcal{E}^{-1}(\{\vec{n}\}) \neq \phi\}$. Then the templet which involves relation $r =$ 'e' or $r =$ 'ch' is searched:

If $\exists \vec{N}_s = \{\vec{n}_1, \vec{n}_2, \cdots, \vec{n}_k\} \subset \vec{S}_i$ where

$(\mathcal{P}(\Delta_1) \cup \mathcal{P}(N_1)) \cap (\mathcal{P}(\Delta_2) \cup \mathcal{P}(N_2)) \cap \cdots \cap (\mathcal{P}(\Delta_k) \cup \mathcal{P}(N_k)) \neq \phi$
and $(\mathcal{S}(\Delta_1) \cup \mathcal{S}(N_1)) \cap (\mathcal{S}(\Delta_2) \cup \mathcal{S}(N_2)) \cap \cdots \cap (\mathcal{S}(\Delta_k) \cup \mathcal{S}(N_k)) \neq \phi$
and $\exists \vec{n}_i \in \vec{N}_s$ s.t. $(\vec{n}_i.F_s \cup \vec{n}_i.F_e) \cap \vec{N}_s = \phi$

Begin

$D_m = min\{D_m^i : \forall \vec{n}_i \in \vec{N}_s$ s.t. $(\vec{n}_i.F_s \cup \vec{n}_i.F_e) \cap \vec{N}_s = \phi\}$

For $\forall \vec{n}_i \in \vec{N}_s$ s.t. $(\vec{n}_i.F_s \cup \vec{n}_i.F_e) \cap \vec{N}_s = \phi$

$D_m^i = D_m$;

$\vec{S}_i = \vec{S}_i - \vec{N}_s$;

Select $\vec{n}_j \in \vec{N}_s$ s.t. $(\vec{n}_j.F_s \cup \vec{n}_j.F_e) \cap \vec{S}_i \neq \phi$;

$\vec{S}_i = \vec{S}_i \cup \{\vec{n}_j\}$;

End

Thus, the "representative" (i.e., $\vec{n}_j$) of the templet in a scenario is included in the subset $\vec{S}_i$. From the $\vec{S}_i$, $\vec{N}_f^i$ is constructed.

At every AC_i: $\vec{N}_f^i = \vec{N}_f^{i0} = \phi$;

If $\exists \vec{n}_x \in \vec{S}_i$ & $\exists$ non-δ *node* $\in \mathcal{P}(N_x)$

$\vec{N}_f^i = \vec{N}_f^i \cup \{\vec{n}_x\}$, for all such $\vec{n}_x$;

Else Begin

/* Only the relations of $r =$ 'd' and $r =$ 'f' need to be considered */

For all $\vec{n}_y \in \vec{S}_i$ & $\{d\} \cap \tilde{\mathcal{I}}_r(\vec{N}, \vec{n}_y, \vec{S}_i) \neq \phi$

For $\vec{n}_y$ s.t. $D_i^y = max\{D_i^y, D_i^z : \{d\} \cap \tilde{\mathcal{I}}_r(\vec{N}, \vec{n}_y, \{\vec{n}_z\}) \neq \phi,$
where $\vec{n}_z \in \vec{S}_i\}$

Begin

$\vec{N}_f^{i0} = \vec{N}_f^{i0} \cup \{\vec{n}_y\}$;

If $\exists \vec{n}_z (d) \vec{n}_y$ & $\{f\} \cap \tilde{\mathcal{I}}_r(\vec{N}, \vec{n}_z, \vec{S}_i) \neq \phi$

$\hat{D}_i^y = \frac{D_i^y + D_i^z}{2}$; — (1)

Else $\hat{D}_i^y = D_i^y$;

End /* For */

For all $\vec{n}_x \in \vec{S}_i$ & $\{f\} \cap \tilde{\mathcal{I}}_r(\vec{N}, \vec{n}_x, \vec{S}_i) \neq \phi$

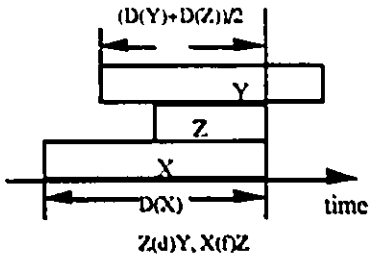


Figure 3.23: Derive the time parameter

Begin

$\hat{D}_i^x = D_i^x$;

If $\vec{n}_x \notin \vec{N}_f^{i0}$: $\vec{N}_f^{i0} = \vec{N}_f^{i0} \cup \{\vec{n}_x\}$;

End /* For */

For $\vec{n}_j \in \vec{N}_f^{i0}$ such that $\hat{D}_i^j = max\{\hat{D}_i^x : \forall \vec{n}_x \in \vec{N}_f^{i0}\}$

$\vec{N}_f^i = \vec{N}_f^i \cup \{\vec{n}_j\}$;

If $\vec{n}_j \in \vec{N}_s$, $\vec{N}_f^i = \vec{N}_f^i \cup \vec{N}_s - \{\vec{n} : \vec{n} \in \vec{N}_s \ \& \ \vec{n}.F_s \cap \vec{N}_s \neq \phi\}$;

/* Collect the "templet" $\vec{N}_s$* /

End /* Else */

The calculation at line (1) is illustrated in Fig. 3.23. If $\vec{n}_\tau \in REF_i^c$ is an intermission node and it has more than one "owners", after $\vec{N}_f^i$ obtained, we take $\vec{n} \in \vec{N}_f^i$ and $\vec{n}_\tau.S_\tau = \vec{n}.S_r$. REF_m need to be derived from REF_m^c. As addressed before, after step 2, $t_m^n = t_{A(\vec{n}_j)}^{nj} + D_{A(\vec{n}_j)}^j$ will be received from the source S_n who "owns" vector $\vec{n}_j \in REF_m^c$. Define $EQU_m^n = t_m^k = t_{A(\vec{n}_j)}^j + D_{A(\vec{n}_j)}^j$, $n = 1, 2, \cdots, |REF_m^c|$, for every $\vec{n}_j \in REF_m^c$, $\vec{n}_j.S_r = S_k$. Here $t_{A(\vec{n}_j)}^j$ is the final schedule obtained for $\vec{n}_j \in AC_{A(\vec{n}_j)}$, and $D_{A(\vec{n}_j)}^j$ is the presentation duration. Then comparing all EQU^is, the REF equation for AC_m, i.e., $EQU_m = t_m^{REF} = t_{A(\vec{n}_k)}^k + D_{A(\vec{n}_k)}^k$, $\vec{n}_k \in REF_m$, is obtained. In the comparison, the $tri-node$ and $sq-node$ determine the vector to be selected for REF_m. That is, if AC_m is started by a $tri-node$ (e.g., $AC_2 = \{P1, P3\}$ in Fig. 3.18), $EQU_m = min\{EQU_m^n, n = 1, 2, \cdots, |REF_m^c|\}$. Otherwise, we have $EQU_m = max\{EQU_m^n, n = 1, 2, \cdots, |REF_m^c|\}$.

3. Inference of the schedules for all the data objects in the scenario

Now, the inference of the final schedules for the distributed objects can be conducted. The schedules from the different sources will be synchronized to obtain the final schedule t_m^x of interval $\vec{n}_x \in AC_m$. As described above (Fig. 3.22), first the sequential inference is performed to obtain t_m^{REF}, i.e., t_m^1 for $\forall \vec{n}_i \in \vec{N}_f^m$. Then the schedules for all the other vectors in AC_m can be derived in accordance with their relations to the vector $\vec{n}_i \in \vec{N}_f^m$. Meanwhile, besides the sequential inference, the schedule for every object has also been chosen by its "owner" source in step 2. Therefore, a "maximum selection" is performed to decide the final schedules for the objects. This "maximum selection" prevents the jitters from occurring at a source (i.e., to ensure that the source will be ready to start at the time scheduled).

Use t_m^x to denote the final schedule for interval $\vec{n}_x \in AC_m$. Given schedules t_i^k for $\forall \vec{n}_k \in AC_i$, $\forall \vec{n} \in AC_m \in \mathcal{E}(\{AC_i\})$ can be scheduled as the following.

First, obtain REF_m by comparing the finishing times of the objects in REF_m^*.

1) $EQU_m = t_m^{REF}$ = the finishing time of $\vec{n}_u$, $\vec{n}_u \in REF_m$.

2) Obtain schedules for $\forall \vec{n}_y \in \vec{N}_f^m$: $t_m^{REF} = max\{t_m^{REF}, t_m^{ky} : \forall \vec{n}_y \in \vec{N}_f^m\}$; then for $\forall \vec{n}_y \in \vec{N}_f^m$, $t_m^y = t_m^{REF}$; $\Delta\tau_m = t_m^{REF} - EQU_m$. Here t_m^{ky} is the schedule made by the "owner" source S_k of $\vec{n}_y$.

3) For $\forall \vec{n}_x \in \Lambda_m^p$ & $\vec{n}_x \notin \vec{N}_f^m$, if $\exists \vec{n}_y \in \vec{N}_f^m$ & $\vec{n}_x(r)\vec{n}_y$, $r \in R_f \cup R_m$:

$t_m^x =$ the starting time of $\vec{n}_x$ derived from t_m^y;

4) If $t_m^x < t_m^{rx}$ /* t_m^{rx} is the schedule from the "owner" source of $\vec{n}_x$ */

Begin

$\Delta\tau = t_m^{rx} - t_m^x$;

For $\forall \vec{n}_y \in AC_m$ and $\vec{n}_y$ is scheduled: $t_m^y = t_m^y + \Delta\tau$;

$\Delta\tau_m = \Delta\tau_m + \Delta\tau$; /* Gap between AC_i and AC_m */

$t_m^x = t_m^{rx}$;

$\vec{N}_f^m = \vec{N}_f^m \cup \{\vec{n}_x\}$;

End

For $\forall \vec{n}_z \in \Lambda_m^s$, $\vec{n}_z \in \mathcal{E}(\{\vec{n}_x\})$ & $\vec{n}_x \in \Lambda_m^p$, schedule of $\vec{n}_z$ are obtained by following t_m^x and the display duration locally at the owner source.

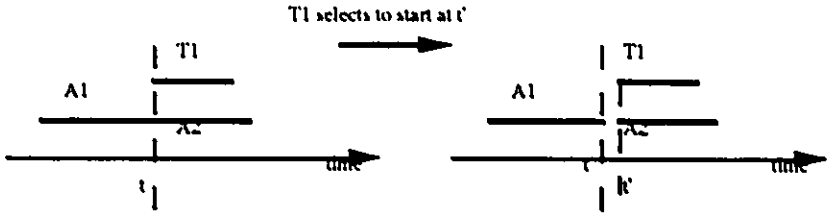


Figure 3.24: Gap occurrence in the scheduling

At step 1), the $ATFG$ has already been turned into a tree, provided that $|REF_m|$ is limited to 1. The step 3) and 4) are repeated until $\forall \vec{n}_x \in AC_m$ are scheduled. In step 2) and 4), the "maximum selection" is performed to choose the proper schedules for the objects. However, after the maximum selection, gaps can occur in the schedules. For instance, in Fig. 3.24, if the source which supplies text $T1$ can only start at time t' but t, by the maximum selection, the schedule for both audio object $A2$ and text $T1$ will be delayed to time t'. Then there will be a gap τ in the display between $A1$ and $A2$. Gap filtering is thus necessary:

$$\text{If } \Delta\tau_m > 0,\ t_i^x = t_i^x + \Delta\tau_m;\ \text{for } \forall \vec{n}_x \text{ s.t. } \vec{n}_x \in AC_i \ \&\ AC_i \notin \mathcal{L}_m,$$

where $\mathcal{L}_m$ is the sub-tree whose root node is AC_m.

Applying the recursion technique, the entire algorithm **Infer**(m) is presented in the Appendix 2. With an initial value $m = 0$, **Infer**(m) can generate the synchronized schedules for all the objects contained in a scenario.

3.5.3 Discussions

It is easy to verify that the three conditions mentioned before can be satisfied. The algorithm is conducted by scheduling the ACs. Because every vector is in one and only one AC (Proposition 3.1), all the objects in the scenario are scheduled. In the algorithm, the maximum selection prevents the jitters. The gap filtering procedure eliminates the gaps in the schedules. Meanwhile, all the relations of $r \in R_f \cup R_m$ are contained in the

ACs and are thus maintained among the intervals, after the inference.

When the $ATFG$ is linear (e.g., the $ATFG$ in Fig. 3.18), the gap filtering procedure becomes:

If $\Delta\tau_m > 0$, $t_i^x = t_i^x + \Delta\tau_m; \forall i < m$.

Thus the recursion of the gap filtering (the search on the tree) need not be performed. Therefore if an $ATFG$ has a linear structure, the computational complexity of executing **Infer**(m) is reduced. On the contrary, the scenario which has an $ATFG$ model as shown in Fig. 3.19 may cause more computation in the scheduling.

In the algorithm designed, the object X which is related to other object through a parallel relation $r \in R_f \cup R_m$ is taken as the synchronization unit. Thus, in the object scheduling, the lower bound of the synchronization unit is constrained by the parallel relations in which the objects are involved. That is, our algorithm manages the synchronization among the data objects, which is directly required by the specified temporal relationships.

3.6 Example and Practical Issues

3.6.1 A Scheduling Example

The scenario shown in Fig. 3.1 is taken as the example scenario. Three sources S_1, S_2 and S_3 are assumed in the integration. The objects and their owner sources are given as:

S_1 : $\{P1, P2, T1, T2\}$; S_2 : $\{A1, A2, A3\}$; S_3 : $\{V1\}$

The ACs and REF^cs from the algorithm **Establish**($\vec{n}$) are given here again for convenience.

$AC_0 = \{P1\}$, $REF_0^c = \phi$; $AC_1 = \{P2, A1\}$; $REF_1^c = \{P1\}$;

$AC_2 = \{V1, A2, T1\}$, $REF_2^c = \{A1\}$; $AC_3 = \{T2, A3\}$; $REF_3^c = \{T1, V1, A2\}$.

First, the source S_1 is taken as the example to demonstrate the local scheduling. The algorithm **Schedule**(m) is executed with initial value of $m = 0$.

Assume the schedules chosen by S_1 for its objects are

$$(t_0^{1P1},\ t_1^{1P2},\ t_2^{1T1},\ t_3^{1T2}) = (20,\ 40,\ 45,\ 50)$$

The other two sources perform the same local scheduling procedure. Then the messages from each source will be sent to the others. The message from source S_1 is the above schedule results, plus the $t_1^1 = t_0^{1P1} + D_0^{1P1} = 20 + 10 = 30$ and $t_3^1 = t_2^{1T1} + D_2^{1T1} = 45 + 10 = 55$. Meanwhile, the presentation duration of the objects owned by source S_1 is also sent. Assume that S_1 has $D(P1) = D(P2) = D(T1) = D(T2) = 10$.

Now, we assume that the message from S_2 contains:

$$(1)(t_1^{2A1},\ t_2^{2A2},\ t_3^{2A3}) = (0,\ 0,\ 30);\ D(A1) = D(A2) = 10,\ D(A3) = 5;$$

$$(2)t_2^2 = t_1^{2A1} + D_1^{2A1} = 0 + 10 = 10;\ t_3^2 = t_2^{2A2} + D_2^{2A2} = 0 + 10 = 10.$$

The message from S_3 includes: $(1)(t_2^{3V1}) = (58);\ D(V1) = 15;\ (2)t_3^3 = t_2^{3V1} + D_2^{3V1} = 58 + 15 = 73$.

After $S_i, i = 1, 2, 3$ receives the messages from the other two sources, $\vec{N}_f^i$ can be obtained for every AC_i as the following:

$$\vec{N}_f^0 = \{P1\};\ \vec{N}_f^1 = \{P2, A1\}.$$

Since $V1(d)A2$ & $D(V1) > D(A2)$, $\vec{N}_f^2 = \{V1\}$; At AC_3, since $T2(f)A3$ & $D(T2) > D(A3)$, we have $\vec{N}_f^3 = \{T2\}$.

Now, every S_i can execute the algorithm **Infer**(m) (see Appendix 2) with the same set of parameters to obtain the same schedules for the objects.

As **Infer**(m) is being computed, we can have $t_0^{P1} = 20$, $t_1^{REF} = 20 + 10 = 30 = EQU_1$, $t_1^{REF} = max\{30, 0, 40\} = 40$, and a gap $\Delta\tau_1 = t_1^{REF} - EQU_1 = 10$ between AC_0 and AC_1. Since $P2(s)A1$, $t_1^{P2} = t_1^{A1} = 40$. Then at AC_2, $t_2^{REF} = t_1^{A1} + D_1^{A1} = 40 + 10 = 50 = EQU_2$. The schedules for the objects in AC_2 can now be derived from the EQU_2.

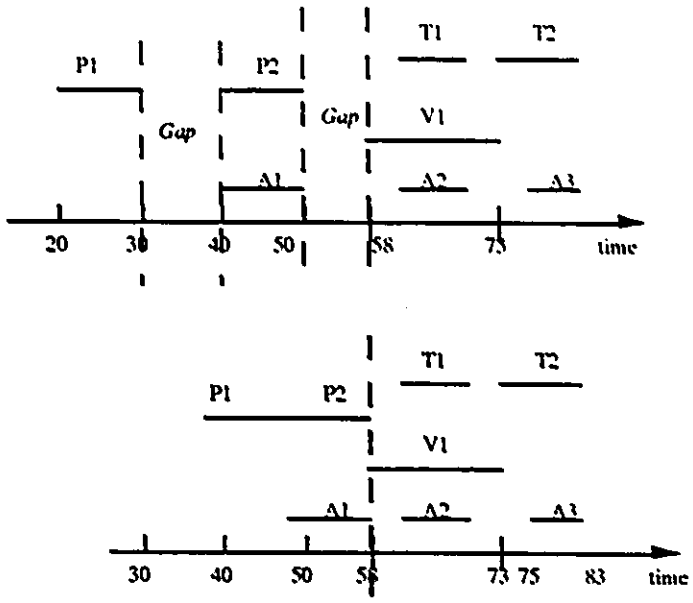


Figure 3.25: Example: Gap occurrence in the scheduling

Since $V1 \in \tilde{N}_f^2$, $t_2^{V1} = max\{50, 58\} = 58$. Then a gap $\Delta\tau_2 = 58 - t_2^{REF} = 8$ exists between AC_1 and AC_2. Meanwhile, we can obtain:

$A_2(d)V_1$, so $t_2^{A2} = max\{0, 58 + (15 - 10)/2\} = 60.5$;

$T1(s)A2$, so $t_2^{T1} = max\{45, 60.5\} = 60.5$.

At AC_3, similarly, the schedules for T_2 and A_3 are obtained as

$t_3^{T2} = max\{55, 73\} = 73$; $t_3^{A3} = max\{30, 73 + 10 - 5\} = 78$.

Because of the gap $\Delta\tau_2 = 8$, gap filtering procedure Gap(8,1) (see Appendix 2) is performed. As a result, $t_1^{P2} = t_1^{A1} = 40 + 8 = 48$. Now $\Delta\tau_1 = 40 + 8 - 30 = 18$, $t_0^{P1} = 20 + 18 = 38$. The final schedules for all the objects are obtained as

$$(t_0^{P1}, t_1^{P2}, t_1^{A1}, t_2^{A2}, t_2^{V1}, t_2^{T1}, t_3^{T2}, t_3^{A3}) = (38, 48, 48, 60.5, 58, 60.5, 73, 78)$$

as shown in Fig. 3.25. The execution of the algorithm **Infer**(m) can be done by one source selected or by three sources in a distributed manner. In the former case, the resulting

schedules are sent to the corresponding sources. If the algorithm is computed at every source, same results will be obtained at all the sources.

3.6.2 Interactions in The Collaborative Multimedia Presentation

In the multimedia presentation, the viewer may pause the data display sequence to view a certain portion. Fast-forwarding (jumping) and fast-rewinding (jumping back) of the presentation may also be required. In these applications where the interactions are involved, the sources need to perform a "re-synchronization" as follows, to restart the sequential display after the interruption.

The new display requirement, including the starting time t_{dis} and the starting position in the presentation sequence, is sent back to the sources. Assume that the new display is required to start right after the object X has already been displayed for a time of $D_{bias}(X)$. Then $\Delta\tau = t_{dis} - D_{bias}(X) - t_m^x$ is obtained by the sources, where t_m^x is the original schedule derived for object X. Thus, for $\forall \vec{n}_y \in \vec{N}$ in the following display, $t_{new}^y = t_m^y + \Delta\tau \geq t_{dis}$ can be obtained. The synchronization among all the sources are then maintained.

With the interaction of "pause", the TFG models handle the object in the scenario whose presentation duration cannot be known until the presentation is finished, e.g., a program to be discussed in conferencing. The object of the program can be assigned an estimated duration value in the designed scenario for the sake of the algorithm. When the program is displayed, all the sources come into a "pause", until the new starting time t_{dis} is received. Assume that object X is to be presented right after the program. $t_{dis} \geq t_m^x$ is required, where t_m^x is the schedule of X originally obtained from the scheduling algorithm. Then, the sequential display will be continued, as addressed above.

Similarly, live dubbing of the presentation can be implemented, although the presentation duration of the live data is often unknown until it is finished. In all the applications where live data are involved, TFG models accommodate the live data object through its

corresponding stored data object, which has been specified in the scenario and scheduled. The user, for instance, can view a document at his/her workstation while commenting it lively. Thus, the user will not be bothered (scheduled) as to when should start to talk. "Pause" and "fast-rewinding" (back-jumping) can both be used when the stored object is being displayed, to let the user finish the dubbing. Then, the t_{dis} is sent and the presentation can be resumed.

3.7 Summary

In our synchronization model proposed, we introduced a method to convert the relative and imprecise temporal relationships among the multiple presentation intervals into a precise synchronization control for the distributed multimedia sources. This is achieved by describing and preserving the temporal semantics at the starting and ending times of the concurrent intervals. Then the exact sequential, as well as the parallel relationships among the intervals can be finally derived and followed. By modeling the "fuzzy" scenarios which are generated directly from the distributed applications, synchronization of the independent sources can be handled in a hub-free manner, with respect to the autonomy of the sources in a group-oriented environment. The application of multimedia teleorchestra can be supported in real-time.

The synchronization unit in our algorithm is constrained by the parallel relations specified in the scenario. All the temporal relations that can be required in the teleorchestration are reflected and maintained during the scheduling. Meanwhile, as multiple destinations are usually involved in the application, the scheduling of the objects is considered in a multiple connection situation. User's interactions and live data objects are also handled in the synchronization model. Our above contributions were published in [25][26].

Chapter 4

Simultaneous Real-Time Multimedia Data Delivery

In order to provide the multimedia presentation to the end users over the network, we have designed a real-time multimedia communication architecture where a Simultaneous Real-Time Data Delivery (SRTDD) scheme is embedded. In this chapter, we present this SRTDD scheme.

The SRTDD control comprises the multimedia session management, the multimedia Activity Constraint Control and the multimedia Segment Delivery. Instead of the temporal relationship models used earlier in the TPC algorithm, the Synchronization Quality of Service (SQoS) parameters are employed to control the synchronization of the real-time multimedia data streams.

4.1 System Architecture

4.1.1 Layered System Model

We repeat here the synchronization problems that need to be resolved in the SRTDD control. Firstly, in a single medium data stream, the continuity of the data may be interrupted by the random network delay. Secondly, among multimedia data streams processed and transferred in parallel, mismatching (i.e., synchronization errors) may hap-

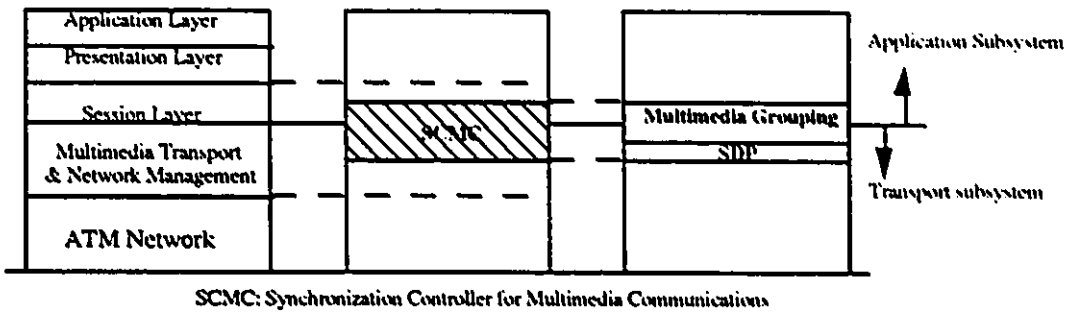


Figure 4.1: Layered architecture

pen. Then the time relationships required by the application can no longer be satisfied. The synchronization errors may be caused by the random network delays, as well as by the different data processing speeds (e.g., decoding speeds) and clock errors.

Our solution for the SRTDD in multimedia communications is based on a layered system model (see Fig. 4.1). In Fig. 4.1, the multimedia system is divided into two subsystems, the Application Sub-System (ASS) and the Transport Sub-System (TSS) [20] . The ASS consists of three layers which correspond to specific layers of the OSI Reference Model: the application layer, the presentation layer and the session layer. The TSS manages the transmission of the multimedia data. Since an ATM network is utilized for the multimedia data transfer, the three lower layers of the TSS conform to the ATM layers. The Multimedia Transport and Network Management layer (MTNM), which is located above the ATM layers in the MDCS model, prepares the multimedia data generated by the ASS for transfer on the ATM network.

It is known that the existing OSI model is unable to handle the relationships among multiple data connections for multimedia communications [39]. Therefore the Synchronization Controller for Multimedia Communications (SCMC) has been introduced to the system to manage such relationships in the SRTDD control. Part of the SCMC is located in the session layer and the rest is in the MTNM layer (see Fig. 4.1).

In order for the session layer to organize the multimedia data and to represent their simultaneous time relationships imposed by the application, the SCMC performs

the so called "grouping"(section 4.2.2) on the application data. Then the "activities" defined by the application are delivered by the SCMC as synchronization control units. If the application demands "tight" coupling, then at MTNM layer, a fine synchronization control unit, i.e. the "segment" is choked by the SCMC within every activity in the session. Delivery protocols and control schemes applied on the units by the MTNM layer filter the gaps, smooth the jitters and couple the data streams.

The MTNM layer then prepares the multimedia data for transmission on the ATM network. For each multimedia session, the MTNM layer determines the transmission requirements, the QoS parameters (e.g., parameters of the average data speed, peak rate, packet transfer delay and loss, data stream smoothing, etc.,) (will be addressed later) in conformity with the application requirements and then requests for the network connections. The desired QoS parameters will be later translated into parameters of cell transfer delay, cell loss and smoothing, etc., at the ATM level[44].

4.1.2 Multimedia Data Transfer on ATM Networks

Distributed multimedia applications require network support with high and variable bit rates for variable data types. For this reason we have chosen an ATM network for the MDCS which supports multi-connections through high capacity data links. The multimedia data are separated into multiple streams according to the variable transmission requirements of the medium they belong to. Each stream is transmitted on a suitable virtual connection.

The bandwidth of the communication connection required by data of a particular medium is determined by its source rate (sample rate and sample size). For instance, the digitized voice has a sample rate of 8 kHz and a sample size of 8 bits. Therefore the connection required is: $8kHz \times 8bits = 64kbits/sec$. If the sample rate of digitized video is $10MHz$, its sample size is $24bits$ and the number of frames generated per second is $30frames$, the data rate generated by sampling is then $(10MHz/30) \times 24 \times 30 = 240Mbits/sec = 30Mbytes/sec$. Assume the compression ratio of video coding equals 8. Then, the required connection speed is $30Mbits/sec$. In the case of high quality digitized

audio, the sample rate is at least 44 kHz and the sample size is 16 bits. The connection required, in these circumstances, has to support $44kHz \times 16bits \approx 704kbits/sec$. Such virtual connections can be carried by a 150 Mbits/sec ATM data link.

It is known that multimedia application sources burst (e.g., video and voice) with variable data rates. ATM networks are designed to support the sum of the average connection speeds as well as the acceptable peak rate [44]. In this way, an efficient utilization of network resource can be achieved [44].

4.2 Multimedia Session Management

4.2.1 Synchronization Quality of Service Parameters

Before introducing the notion of control schemes in SCMC, we first notice that the different synchronization constraints are required in different multimedia presentational applications.

"Tight coupling" is required by some applications. A typical example of strict time matching is the "lip-sync" problem in video conferencing. The movements of the speakers' lips have to match exactly their voice. On the other hand, sometimes, tight coupling is not strictly required. In the case of a music service application, for instance, the Hi-Fi music and a background video are sent to a remote user for the purpose of entertainment. The display of the video showing, e.g. views from nature, does not require any strict matching with the music. In this case, a "loose coupling" is sufficient. In video conferencing, for example, voice data may couple the video with a tight "lip-syn" while short text annotations loosely couple the voice by displaying the summary of the speech.

The "tight" and "loose" coupling should be described quantitatively for synchronization control. This is a very important issue in the multimedia synchronization studies. From the experiments conducted recently by Steinmetz in IBM Heidelberg [42], the synchronization errors that can be tolerated by human perception vary in different application scenarios. For instance, 80 *msec* mismatching in lip-sync of voice and video

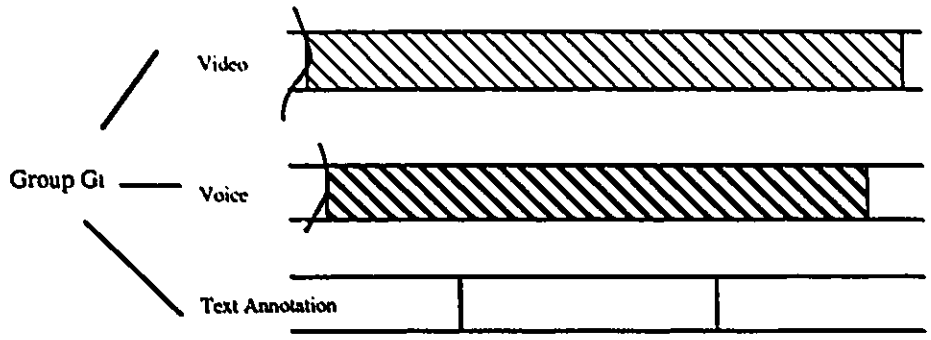


Figure 4.2: The multimedia group G_1

can be perceived as disturbing by the application users. On the other hand, if the text annotation is played out 200 *msec* earlier than the voice, the presentation can still be well accepted. We call these application defined parameters the "skew tolerance parameters". Skew tolerance parameters determine the synchronization goal that should be achieved by the SRTDD control. The issue of specifying the different coupling types and the corresponding skew tolerance parameters, and also the derivation of the parameters between the indirectly related data streams have been thoroughly studied by Steinmetz [42]. According to his study, Steinmetz has proposed that the skew tolerance parameters are taken as the Synchronization QoS (SQoS) parameters and included in the QoS requirements for the multimedia transmission network.

4.2.2 Grouping of Multimedia Data Streams

As multiple data streams are included in a multimedia application, a "group" is used to separate the multiple data streams into groups according to their relationships in a phrase of the application. Grouping primitives are offered by the Synchronization Controller of Multimedia Communications (SCMC) so that the application can declare the simultaneous temporal relationships among the multimedia data streams being generated. Data streams in the same group are related directly or indirectly in the application scenario. In Fig. 4.2, for instance, in group G_1, the voice couples the video with a tight "lip-sync" while the short text annotations loosely couple the voice by displaying the summary of

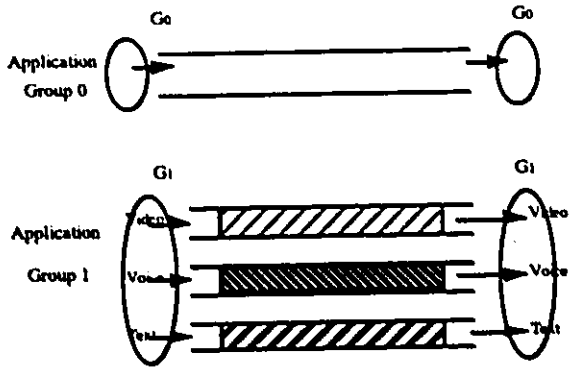


Figure 4.3: Grouping multimedia data streams

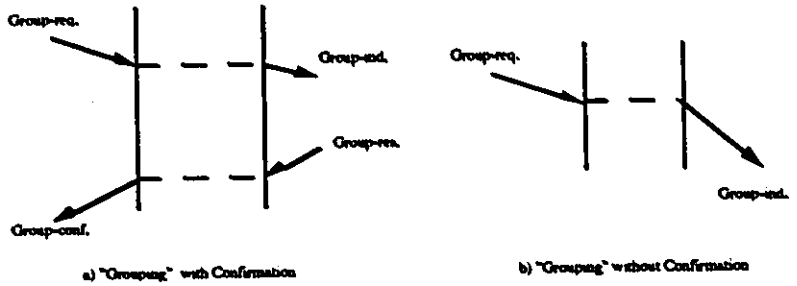


Figure 4.4: Grouping primitives

the speech. Therefore, among the three data streams in group G_1, the voice is directly related to both the video and the text, while the text data stream is directly related to the voice stream but indirectly to the video. Data streams in different groups are not temporally related. Thus there is no synchronization requirement between the streams in different groups. Different groups can be declared for the data that belong to different applications. As shown in Fig. 4.3, the data stream in group G_0 is independent from the application in group G_1. There is no real-time synchronization requirement between the processing and transmission of the data in G_0 and G_1. Thus the receiver can handle the data stream in G_0 separately (e.g., holding one window for it). Four primitives are offered by the SCMC to the application (see Fig. 4.4). The primitives can be used to add in new groups (e.g. user joining in another sub-conferencing [43]) as well as change the old ones (e.g., user trying to close one of the media that was in use). They are also used

to declare the change in the coupling requirements between the data streams. Therefore, the status of "grouping" varies with the change of the media in use as well as the new synchronization constraints on the media. Both of them indicate the change in the application by presenting the new relationships of the data streams. In the real application, the new relationship of data starts naturally a new application phrase (session). Thus the multimedia session management which concerns the relationships of the data in the application is carried out by the "grouping". By the exchanging of a set of "grouping" primitives, the SCMC starts the new application session. The session includes the groups of multiple data streams declared and, within each group, the coupling types required for the data streams.

The class of "coupling types" is provided by SCMC to support the different applications. The skew tolerance parameters are associated with every "coupling type". The coupling types involved in the group are required in the primitives. Then the synchronization control should maintain the required coupling by limiting the synchronization errors within the corresponding skew tolerance parameters. We notice that more than one coupling types can be involved in a group. For instance, in the example of group G_1, voice and video streams are coupled by the "lip-sync" skew tolerance parameters, while the text and the voice streams fall into another type. Thus, in our synchronization control architecture, SCMC undertakes the task of achieving the required quality according to the SQoS parameters.

In order for the TSS (Transport Sub-System) to manage the network connections for the multimedia data streams included in the application session, the second function of "grouping" is to pass the transmission requirements of the data streams as the QoS and SQoS parameters to the TSS. The TSS will then resume old connections that are still usable, release extra connections and set up new ones that are necessary for the session. Therefore the "grouping" helps to finish the network resource reservation for the transport management while organizing the data streams for the synchronization control.

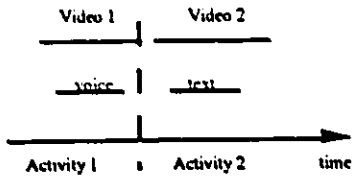


Figure 4.5: Activities

4.3 Activity Constraint Control

In every group, activity is "marked" in the data streams by the application itself, indicating the desired measure of the matching constraint at the application level. For instance, activity can be defined by the synthetic presentation scenarios as we encountered in the previous section of Temporal Presentation Control. If we take the data delivery for an entire multimedia scenario as a session, the simultaneous presentation of a set of coupled objects can be taken as an activity. For instance, two activities can be assumed in the scenario shown in Fig. 4.5. The first activity is a video clip summarized by a voice segment. The second is another video clip explained by a piece of text. In each activity, there is no tight coupling between the streams. However, in order to keep the application valid, the first activity should be finished before the data streams in the second can be started. The activity constraint control primitives are provided by SCMC to allow the application declare the activities, so that the sequential ordering of the data on multiple streams can be managed. ACC is also a session management entity. The ACC primitives are shown in Fig. 4.6.

4.4 Multimedia Segment Delivery

After the groups are declared, the SQoS parameters are passed to the multimedia Segment Delivery (SD) entity in the Transport Sub-System (TSS). The (SD) can then be performed in every group.

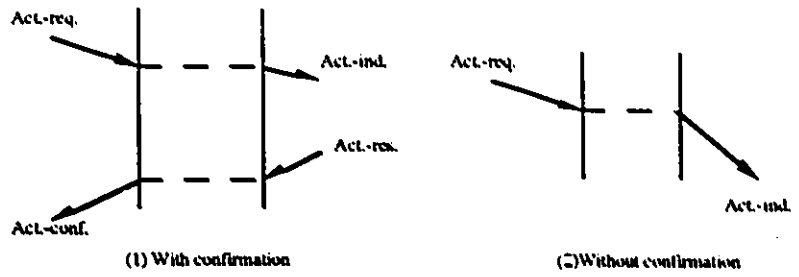


Figure 4.6: Activity primitives

4.4.1 Multimedia Segment

As well as a multimedia data delivery unit, the multimedia segment is the synchronization granularity. In the transport subsystem [20], SCMC forms the multimedia segments by dividing the data streams in a multimedia group on the time axis. If the activities are already defined on the streams in the group, one activity may contain several segments. As multiple data streams are involved, one segment contains several components. Each component corresponds to an individual stream in the group. A segment marks the required simultaneous temporal relationship: streams within a segment are supposed to be displayed simultaneously. The synchronization control is performed on the segment.

The sender workstation forms the multimedia segments and delivers them. Below, we illustrate that the desired segment length is constrained by the required SQoS.

Assume $\Delta\tau_{ij}$ is the skew tolerance parameter between the data streams i and j. Define t_i and t_j as the average delay between the display of the two adjacent packets in the stream i and j, respectively. t_i (t_j) includes the random network delay and the processing delay. Let T_i (T_j) be the time slot of one i (j) data packet. Then the segment size S (time length) is constrained by

$$\begin{cases} max\{T_i, T_j\} \le S \le \frac{T_i T_j \Delta\tau_{ij}}{T_i t_j - T_j t_i} & \text{If } (\frac{T_i}{T_j} > \frac{t_i}{t_j}) \\ max\{T_i, T_j\} \le S \le \frac{T_i T_j \Delta\tau_{ij}}{T_j t_i - T_i t_j} & \text{If } (\frac{T_i}{T_j} < \frac{t_i}{t_j}), \end{cases} \tag{4.1}$$

which is a derivation from the two constraints

$$|\frac{S}{T_i} \times t_i - \frac{S}{T_j} \times t_j| \le \Delta\tau_{ij} \;\&\; S \ge \max\{T_i, T_j\}$$

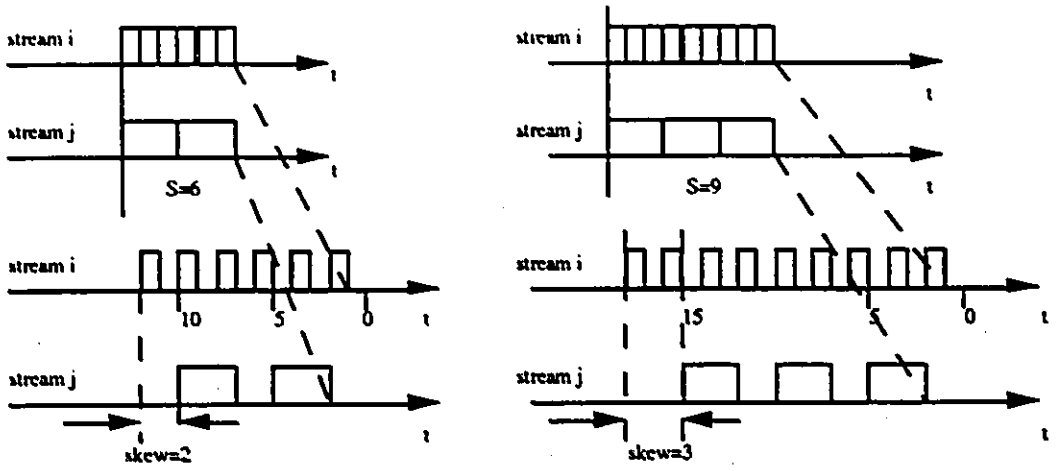


Figure 4.7: Example to show the length of S

For example, let $T_i = 1$, $T_j = 3$, $t_i = 1$ and $t_j = 2$. If the skew tolerance parameter is $\Delta\tau_{ij} = 2$, Fig. 4.7 depicts the skews incurred in the cases of $S = 6$ and $S = 9$. From this simple illustration, it can be observed that if S is too big, the skew tolerance parameter will not be satisfied. On the other hand, it is unnecessary to make the segment too small.

However, the value of the segment size S cannot be determined so simply, because t_i and t_j should both represent random variables. In order to choose the size S, certain performance modeling of the segment delivery scheme is required. This will be discussed in Part II of our thesis.

In Fig. 4.8, as an example, segments are divided in the group G_1 as $G_1\vec{S} = \{\vec{S}_i, \vec{S}_j, \vec{S}_k\}$ on the three involved data streams. $G_1\vec{S}^n = \{s_i^n, s_j^n, s_k^n\}$ is the n-th segment in the group. $G_1\vec{S}^{n-1}$ is the previous segment to $G_1\vec{S}^n$.

As different coupling types and thus different skew tolerance parameters may be involved within the data streams in a group, the segment lengths should vary with the parameters. This may result in two or even more different segmentation scales on one data stream. The examples in Fig. 4.8 and Fig. 4.9 present two segmentation schemes to handle the different scales. In Fig. 4.8, only one segmentation scale is employed in the

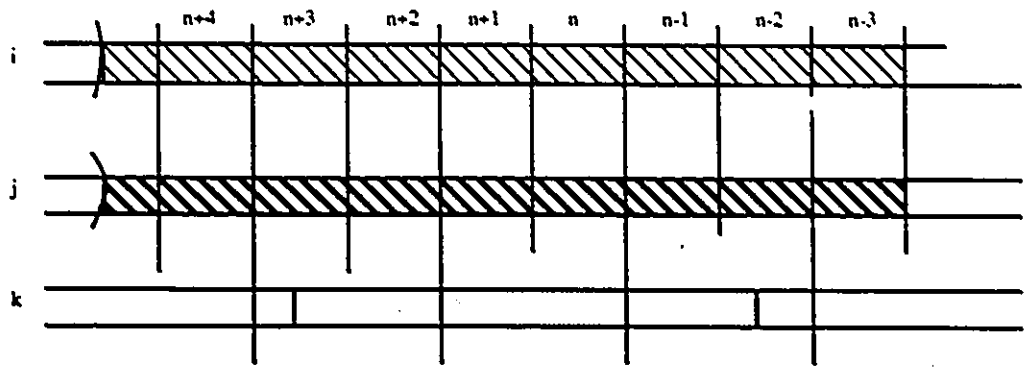


Figure 4.8: Segment on group G_1

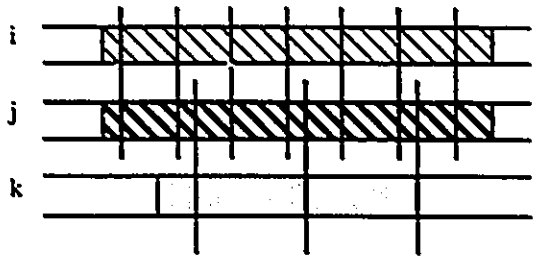


Figure 4.9: Different segmentation scales

group. The segments $\vec{S}^{n-3}$ to $\vec{S}^{n+4}$ are listed as follows.

$$\vec{S}^{n-3} = \{s_i^{n-3}, s_j^{n-3}, \phi_k\};\ \vec{S}^{n-2} = \{s_i^{n-2}, s_j^{n-2}, \phi_k\};$$

$$\vec{S}^{n-1} = \{s_i^{n-1}, s_j^{n-1}, \phi_k\};\ \vec{S}^{n} = \{s_i^{n}, s_j^{n}, s_k^{n}\};$$

$$\vec{S}^{n+1} = \{s_i^{n+1}, s_j^{n+1}, \phi_k\};\ \vec{S}^{n+2} = \{s_i^{n+2}, s_j^{n+2}, s_k^{n+2}\};$$

$$\vec{S}^{n+3} = \{s_i^{n+3}, s_j^{n+3}, \phi_k\};\ \vec{S}^{n+4} = \{s_i^{n+4}, s_j^{n+4}, \phi_k\};$$

Here, ϕ_k specifies that this segment is not applied on the data stream k. Although stream k may still be delivered during this time interval, no synchronization control needs to be performed on it. Thus only one segmentation scale is kept on the stream j. The actual segment size for data stream k is twice as that of streams i and j. In Fig. 4.9, two different segmentation scales are used on the data stream j. This scheme can assign more

accurate segment sizes to the different streams according to the required skew tolerance parameters. However, new skew may be generated during the synchronization control on a segment, as we will present in the subsection below.

4.4.2 Segment Delivery

From the above, the multimedia data streams are delivered as a sequence of segments. The sender chooses the segment size S according to the SQoS (skew tolerance parameters), and also the available network information. Primitives can be used by the receiver to negotiate as well as require the size S according to the actual skew incurred in the play-back. During the delivery, the sender also employs the available network information to predict the delay that the data will experience, during the transmission and the processing at the receiver [11]. Thus schedules can be made for the packet of every component in a segment [28]. Since different components may be transferred on different network connections (e.g., in an ATM network), data of different components may have different schedules in order to be available for the display at the receiver at the "same" time.

Although schedules can be made at the sender, random delays are always inevitable. To deal with the random disturbance experienced by the data, at the receiver, the segment delivery entity performs the synchronization recovery on each segment. The simultaneity between the data streams is restored by starting to display the components contained in the segment, ideally, at the same time. Hence, the first packets (e.g., video frames) of the component streams which arrive earlier are delayed until the first packets (frames) of the remaining streams are received. At the receiver, the "synchronization delay" is introduced to the earlier arrivals by the SCMC due to this waiting. Because of the synchronization delay, a buffer is required at the receiver for the synchronization recovery at the beginning of every segment. This is the synchronization recovery buffer. It can now be seen that if more than one segment scales are forced on one data stream (e.g., Fig. 4.9 stream j), the synchronization delay caused by waiting for a data stream (e.g., stream k) may introduce new skews between streams i and j in their segment display.

Although the first packets (frames) of all the data streams in a segment are started

out together, synchronization errors among them will gradually increase, because of the random delays introduced to the following packets and frames. The errors will not be recovered until the next segment is played out. If some data streams have not arrived even when the threshold of the synchronization delay is reached, the segment may have to be started to display due to the application's real-time requirements. Then the segment is displayed with a "start-out" mismatching at the beginning of the segment [20]. Often the applications can tolerate some "start-out" mismatching as long as the errors are controlled under the skew tolerance parameter.

In order to filter the gaps, smooth the jitters and thus decrease the mismatching at the end of a segment, an "intentional delay" is placed at the beginning of each segment. That is, in a segment, the first packet(frame) of each data stream will be delayed intentionally before being played out. This extra delay helps reconstruct the data streams. The intentional delay is already applied in voice packet switching [7] systems and video application systems. Different media require different intentional delay times for recovering from gaps and jitters [2][3].

The segment information need to be transferred from the sender to the receiver. Below, we present three different schemes for the multimedia segment delivery.

4.4.3 Different Delivery Schemes

1. In-Band Segment Delivery (IBSD)

In-Band Segment Delivery (IBSD) shares the same idea as the "markers" [33]. In IBSD, each component of a segment carries a segment header as shown in Fig. 4.10 (a). The "segment header" starts a new segment. $CDI_i = 1$ indicates the header is a valid segment marker for data stream i. On the other hand, $CDI_i = 0$ informs the receiver that this segment marker is not applied on data stream i. Thus, the receiver will not wait for stream i to match the other streams. Assume ζ_1 bits are needed in a segment header. The bandwidth used for sending each header, when the segment size is S seconds is: $\frac{\zeta_1}{S} \times N$, assuming N components (streams) are included. Since extra bandwidth is required for transmitting the header, the available bandwidth offered by the network constrains the

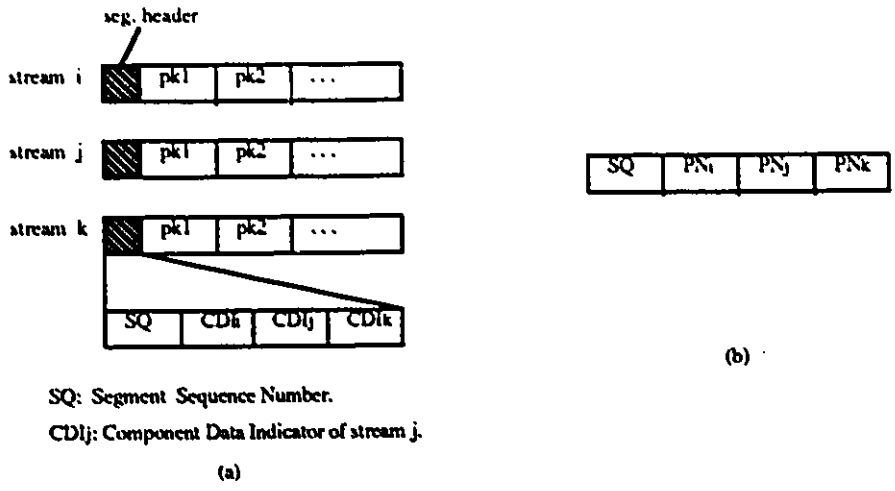


Figure 4.10: Segment headers for IBSD & OBSD

lower-bound of the segment size S. Given B as the bandwidth available for the header transmission, the value of S should be:

$$\begin{cases} max\{\frac{\zeta_1 \times N}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_i t_j - T_j t_i} & \text{If } (\frac{T_i}{T_j} > \frac{t_i}{t_j}) \\ max\{\frac{\zeta_1 \times N}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_j t_i - T_i t_j} & \text{If } (\frac{T_i}{T_j} < \frac{t_i}{t_j}) \end{cases}$$

This is directly obtained from the Formula 4.1 in the previous section.

IBSD easily transfers the segment information during the data delivery. It takes almost no risk of losing any information for reconstructing the segment at the receiver, compared to the out-of-band segment delivery scheme presented below. However, inserting the segment header requires to reformat the data streams which are originally generated from the acquisition devices. Certain difficulties may thus be encountered in connecting the user equipments to the SCMC and the multimedia transport subsystem.

2. Out-of-Band Segment Delivery (OBSD)

Out-of-Band Segment Delivery (OBSD) employs an additional connection for the synchronization control [40]. On this synchronization connection, synchronization parameters such as the segment headers are transferred. For every segment, a header of Fig. 4.10 (b) is sent to the receiver on the connection. Thus, no matter how many components are included in a segment, only one header is sent. For a component of stream i, PN_i gives

the sequence number of the first packet in the component, marking the starting of the segment on stream i. The value of PN_i may also indicate that the segment is not applied on stream i, the same as in the in-band delivery. Assume ζ_2 bits are needed in the segment header. The bandwidth used for sending each header is $\frac{\zeta_2}{S}$, when the segment size is S second. Therefore, the OBSD delivery scheme may require less bandwidth. The constraints on S, in this case, become:

$$\begin{cases} max\{\frac{\zeta_2}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_i t_j - T_j t_i} & \text{If } (\frac{T_i}{T_j} > \frac{t_i}{t_j}) \\ max\{\frac{\zeta_2}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_j t_i - T_i t_j} & \text{If } (\frac{T_i}{T_j} < \frac{t_i}{t_j}) \end{cases}$$

OBSD needs an extra synchronization connection for the transmission. The impact of network fluctuation on this connection may delay the segment header being transferred. Then the receiver will be forced to wait for the header information or even give up a certain segment reconstruction due to the delay. Hence, in the network flow management, different treatments should be applied for the synchronization connection and other data connections. This certainly complicates the management of SCMC and the network.

3. Variable Rate Segment Delivery (VRSD)

Besides the above two segment delivery schemes, a better performance can be achieved by the Variable Rate Segment Delivery (VRSD). The purpose of VRSD is to reduce the segment information sent to the receiver. We observe that the header can contain redundant information unless the application data streams are "bursty". By bursty, we mean that a stream can have two states of "on" and "off" at different times. When a stream is on at time t, data are being transferred on the stream. When a stream is off at time t, no data is being transferred on the stream at the time. In VRSD, the sender only transfers a segment header when at least one of the data streams changes its state during a segment interval S. Fig. 4.11 gives an example. With VRSD, only the headers for segment $n-1$, $n+1$, $n+3$ and $n+5$ are sent.

The receiver reconstructs the segment according to the header received, or the predecided segment length when no header is received. In this case, the same time slot of packet size for all the data streams included in a group can help to simplify the algorithm at the receiver. For instance, if the video source generates 30 frames per second, a packet

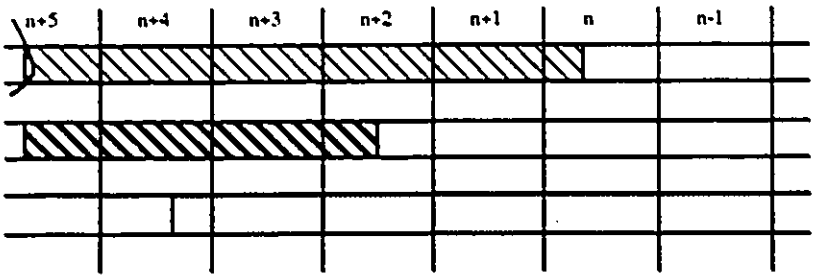


Figure 4.11: VRSD Example

for a 64kbits/sec voice connection should contain 267 bytes. Then if the segment size is, say, 6 packets, the receiver can easily collect 6 packets from every involved stream to reconstruct a segment. In fact, as long as the receiver knows the packet size of the data on each stream, the segment can always be recognized with a segment size S.

In VRSD, the more bursty the data streams, the more information needs to be transferred. Assume that an on-off data stream i can be modeled by a two state Markov chain shown in Fig. 4.12 as:

$$i(t) = \begin{cases} 1 & \text{at time } t, \text{ the state of stream } i \text{ is "on"} \\ 0 & \text{at time } t, \text{ the state of stream } i \text{ is "off"} \end{cases}$$

Then the steady-state probabilities are given by

$$P\{i(t) = 1\} = \frac{\gamma_i}{\gamma_i + \rho_i}, \; P\{i(t) = 0\} = \frac{\rho_i}{\gamma_i + \rho_i}$$

The probability that stream i is "on" for a time longer than t_0 is

$$P\{\sigma_i^1 > t_0\} = e^{-\rho_i t_0}.$$

Among the data streams in group G_1, for example, assume that two streams i and j are independent and bursty, modeled by the above Markov chains. Define the random variable

$$\mathcal{H} = \begin{cases} 1 & \text{a header is sent in a segment interval } S \\ 0 & \text{no header is sent in a segment interval } S. \end{cases}$$

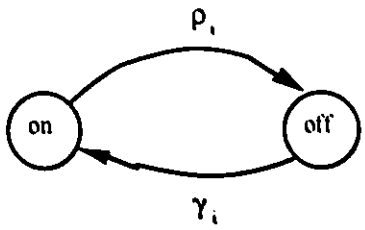


Figure 4.12: The two-state Markov chain model for an on-off stream

Then during a segment interval S, a segment header is sent with probability

$$\begin{aligned} P\{\mathcal{H}=1\} &= \frac{\gamma_i\gamma_j}{(\gamma_i+\rho_i)(\gamma_j+\rho_j)} \times (1-e^{-\rho_i S}e^{-\rho_j S}) \\ &+\frac{\gamma_i\rho_j}{(\gamma_i+\rho_i)(\gamma_j+\rho_j)} \times (1-e^{-\rho_i S}e^{-\gamma_j S}) \\ &+\frac{\rho_i\gamma_j}{(\gamma_i+\rho_i)(\gamma_j+\rho_j)} \times (1-e^{-\gamma_i S}e^{-\rho_j S}) \\ &+\frac{\rho_i\rho_j}{(\gamma_i+\rho_i)(\gamma_j+\rho_j)} \times (1-e^{-\gamma_i S}e^{-\gamma_j S}) \end{aligned} \tag{4.2}$$

The proof of Formula 4.2 is straightforward. At the beginning time , say t, of a segment interval S, there are altogether four possible states of the two streams i and j, listed as the following:

$$\{i(t)=1\cap j(t)=1\},\ \{i(t)=1\cap j(t)=0\},\ \{i(t)=0\cap j(t)=1\}\ \{i(t)=0\cap j(t)=0\}$$

. Then,

$$\begin{aligned} P\{\mathcal{H}=1\} &= P\{\sigma_i^1<S\cup\sigma_j^1<S, i(t)=1\cap j(t)=1\} \\ &+P\{\sigma_i^1<S\cup\sigma_j^0<S, i(t)=1\cap j(t)=0\} \\ &+P\{\sigma_i^0<S\cup\sigma_0^1<S, i(t)=0\cap j(t)=1\} \\ &+P\{\sigma_i^0<S\cup\sigma_j^0<S, i(t)=0\cap j(t)=0\} \end{aligned}$$

Take the first item. We have:

$$\begin{aligned} & P\{\sigma_i^1<S\cup\sigma_j^1<S, i(t)=1\cap j(t)=1\} \\ = \ & P\{\sigma_i^1<S\cup\sigma_j^1<S|i(t)=1\cap j(t)=1\}\times P\{i(t)=1\cap j(t)=1\} \\ = \ & (1-P\{\sigma_i^1>S\cap\sigma_j^1>S|i(t)=1\cap j(t)=1\})\times P\{i(t)=1\cap j(t)=1\} \end{aligned}$$

Since $i(t)$ and $j(t)$ are independent, we can get:

$$\begin{aligned}
& 1 - P\{\sigma_i^1 > S \cap \sigma_j^1 > S | i(t) = 1 \cap j(t) = 1\} \\
= \; & 1 - P\{\sigma_i^1 > S | i(t) = 1 \cap j(t) = 1\} \times P\{\sigma_j^1 > S | i(t) = 1 \cap j(t) = 1\} \\
= \; & 1 - P\{\sigma_i^1 > S | i(t) = 1\} \times P\{\sigma_j^1 > S | j(t) = 1\} \\
= \; & 1 - e^{-\rho_i S} e^{-\rho_j S}
\end{aligned}$$

Then, the following is obtained:

$$\begin{aligned}
& P\{\sigma_i^1 < S \cup \sigma_j^1 < S, i(t) = 1 \cap j(t) = 1\} \\
= \; & (1 - e^{-\rho_i S} e^{-\rho_j S}) \times P\{i(t) = 1\} \times P\{j(t) = 1\} \\
= \; & \frac{\gamma_i \gamma_j}{(\gamma_i + \rho_i)(\gamma_j + \rho_j)} \times (1 - e^{-\rho_i S} e^{-\rho_j S})
\end{aligned}$$

Similarly, it can be derived:

$$\begin{aligned}
& P\{\sigma_i^1 < S \cup \sigma_j^0 < S, i(t) = 1 \cap j(t) = 0\} \\
= \; & P\{\sigma_i^1 < S \cup \sigma_j^0 < S | i(t) = 1 \cap j(t) = 0\} \times P\{i(t) = 1 \cap j(t) = 0\} \\
= \; & (1 - P\{\sigma_i^1 > S \cap \sigma_j^0 > S | i(t) = 1 \cap j(t) = 0\}) \times P\{i(t) = 1 \cap j(t) = 0\} \\
= \; & \frac{\gamma_i \rho_j}{(\gamma_i + \rho_i)(\gamma_j + \rho_j)} \times (1 - e^{-\rho_i S} e^{-\gamma_j S})
\end{aligned}$$

In the same way, the rest of the Formula 4.2 can be derived. Thus, the average number of the headers sent in a segment interval S is

$$\vartheta = P\{\mathcal{H} = 0\} \times 0 + P\{\mathcal{H} = 1\} \times 1 = P\{\mathcal{H} = 1\}.$$

The bandwidth required in VRSD for transmitting the segment header is $\frac{\vartheta \times \zeta}{S}$, where ζ is the number of bits contained in a header. Both in-band and out-of-band segment delivery can be used to transfer the header for VRSD. As $\vartheta \leq 1$, the bandwidth required for the header in VRSD can be less than the other two schemes. The constraints on the segment size S become:

$$\begin{cases}
max\{\frac{\zeta\vartheta}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_i t_j - T_j t_i} & \text{If } (\frac{T_i}{T_j} > \frac{t_i}{t_j}) \\
max\{\frac{\zeta\vartheta}{B}, T_i, T_j\} \leq S \leq \frac{T_i T_j \Delta\tau_{ij}}{T_j t_i - T_i t_j} & \text{If } (\frac{T_i}{T_j} < \frac{t_i}{t_j})
\end{cases}$$

In VRSD, a more flexible and robust segment delivery and reconstruction method can be explored. If the sender sends a header whenever a stream changes its state, the receiver can determine the segment length by itself, according to the skew presented, as long as the receiver knows the time slot of the packet size for every stream. Variable segment lengths can be employed by the receiver, in different network situations. However, if the application data streams are too bursty, more headers may have to be sent in this VRSD variation. Assume the minimum segment size agreed between the sender and receiver is S_0. That is, the segment size S selected by the receiver always satisfies $S \geq S_0$. In this case, if N streams are assumed in the group, when

$$\min_{i=1,\ldots,N}\{(\frac{1}{\rho_i}+\frac{1}{\gamma_i})/2\} < S_0,$$

more headers may be sent than that in the other delivery methods.

4.4.4 Different Intentional Delay Scheme

The intentional delay schemes can be classified as the FD (Fixed Delay for the first arrival) protocol, the CD (Channel Delay for the first arrival) protocol, etc. [2][3]. Here, we only address the CD protocol in detail. Define t as the slot time of a packet (or a video frame). Define $t_d = t + t_n + t_i = t + t_s$, where t_n: Network delay of a packet (frame); t_i: Intention delay of the packet (frame); and $t_s = t_n + t_i$ is named as the system delay of the packet (frame).

In the CD protocol, the receiver adds a fixed intentional delay for the first packet after receiving it, i.e. for the first frame and the first packet: $t_d = t + t_n + T_i$;

Therefore the receiver waits for the arrival of the first packet(frame) and then, after a fixed intention delay T_i , the first packet(frame) of the component is ready for display. The delay time before the play-out of a segment includes the random network delay of the first packet(frame), the constant intention delay T_i and the time of waiting for the other data streams if any of them is not ready yet. The CD protocol can be used by either voice or audio.

Different media may have different intentional delay schemes. For instance, while the CD protocol is applied on the voice data, the first frame in the video data can be allowed for display when the second frame arrives, following a different intentional delay scheme.

At the receiver, there are theoretically two policies that can be applied on the following packets and frames which arrive later in a segment. The two policies are discriminated on the time axis:

E policy—It is a time expanding policy. The receiver will wait for the late packets (frames). There is no packet (frame) discarding. The E policy is more suitable for a video receiver.

I policy—It is a time preserving policy. The receiver will preserve the timing at the expense of ignoring late packets(frames). The receiver will drop the late data. The I policy can only be used for a voice receiver because it is difficult to meet the high quality requirements of the video data.

Now, we are looking into an example of multimedia segmentation and delivery. We choose voice and video as our sample media. The synchronization between real-time voice and video is the basic synchronization control in most popular applications as video conferencing, video phone, etc.

4.4.5 An Example of Multimedia Segmentation and Segment Delivery

The real-time voice and video data streams are taken as our example. It is known that voice data streams are not strictly continuous. The voice data corresponding to a person's talk-spurt are interrupted by silent intervals. People talk alternating between talk-spurts and silent intervals. The length of the talk-spurts and the silent intervals can be modeled as random variables with exponential distributions, with respective mean values 1.32*sec* and 1.82*sec* [2]. Therefore the voice source can be modeled by a two-state Markov Chain

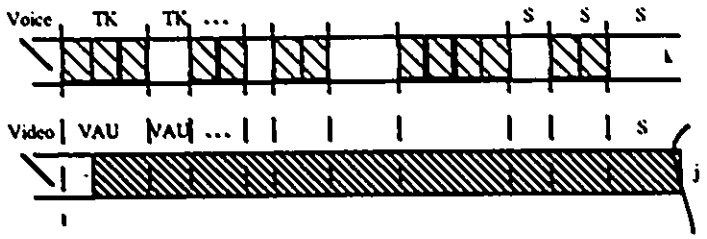


Figure 4.13: Segmentation on Real-time Voice & Video

and the distribution of the number of packets in a talk-spurt is geometric,

$$p_t(\alpha) = prob.\{(\alpha+1)\ packets\ in\ \alpha\ talk-spurt\} = (1-\rho_t)\rho_t^{\alpha}, \quad \alpha = 0, 1, 2, ...(4.3)$$

Voice data has redundant information and can tolerate certain transmission errors.

Video demands high bit rate and low error rate. It is sensitive to transmission errors. High throughput is also a key point in the application. But its response time constraint is not so critical as that of voice. Basically we should avoid losing frames in the video transfer.

We briefly address our video source modeling as follows. The trend of video coding is VBR (Variable-Bit-Rate) based. Assume the video data is generated with a fixed number of 30($usually >= 24$) frames per second. Each frame is differentially encoded by the difference of the movement it contains compared with the previous frame. A frame is made of packets. The number of packets and thus the amount of data contained in a frame is variable.

1. Segmentation on Real Time Voice and Video

Now we form the segment, i.e., the segment by one talk-spurt (or talk silence) with its describing video data, i.e. a matching Video Activity Unit (VAU) (as in Fig. 4.13). The VAU is thus defined as the amount of frames caught (generated) within the time duration of its matching talk-spurt or talk silence. When there is only one data stream and the other is empty in the application, e.g., during talk silence or no talking for a while, data in the video stream will be taken as one segment (e.g., S^n, S^{n+2}, etc., contains only video data).

There are two important features of this segmentation scheme. First, the scheme is very reasonable if we consider the voice data stream itself. If the synchronization unit cuts into a talk-spurt to form a multimedia segment, the synchronization operation within a talk-spurt will destroy the continuity of the speech. As human perception is very sensitive to the voice continuity, keeping the talk-spurt as a unit helps to maintain a good voice play-out quality.

Second important feature of this segmentation scheme is the adjustable segment length. As mentioned above, the data will be taken as one segment when all the other data streams have empty components in this segment. Since no inter-stream synchronization is needed when there is only one data stream on display, the efficiency of segmentation is improved. Because the non-empty component need not wait for the empty one for the display, the length of the empty component (e.g., talk silence) during the delivery will be adjusted according to the arrival of the non-empty component, which is interfered by the random network delay. This gives much more flexibility in the data delivery. The video data corresponding to the talk silence will be displayed continuously after the video component in the previous segment (i.e., corresponding to the talk-spurt) without any synchronization waiting gaps. As the empty voice component is always following and followed by a non-empty one, this scheme supports the continuity of video data streams. It offers a more smooth connection between the display of the video components in the two neighbouring segments with less (less than half) occurrence of gaps .

Now the time length of a video component (a VAU) is also expenentially distributed, similarly as that of its matching talk-spurt or talk silence. As only the VAU that corresponds to the talk-spurt is involved in the "matching" and "waiting" for simultaneous delivery, its length distribution is obtained as:

$$p_v(\alpha) = prob.\{(\alpha+1)\ packets\ in\ a\ (VAU)\} = (1-\rho_v)\rho_v^{\alpha}, \alpha = 0, 1, 2, ... \qquad (4.4)$$

Thus, the probabilistic characteristics of the segment size are obtained.

Any other scheme with a smaller segment size will break the talk-spurt. As a user is always very sensitive to the voice medium, interrupting a talk-spurt is irritating. Since

the play out schemes for voice based on the talk-spurt are well applied in today's packet-switching voice system, it is a very practical and also a reliable way to construct the segment on this model.

2. The Voice & Video Segment Delivery

We are now investigating typical protocols chosen for the segment delivery including the voice and video components. Our analysis will mainly deal with the random network delay.

Assume that the network delays of all packets in a talk-spurt are independent, identically distributed (i.i.d.) random variables; same are the delays of the video frames. The assumptions will be guaranteed in networks with a highly dynamic routing policy. The transmission of voice is assumed to be independent of that of the video data. As the ATM network has multi-connections, this assumption can be guaranteed when the network is not in a congestion situation. We also assume that the statistical independence between the talk-spurts is guaranteed. The assumption is also applied to the VAU frames.

Some random variables and other related variables are defined as below: t_p: Time slot of a voice packet; t_f : Time slot of a video frame; t_{nt}: Network delay of a voice packet; $p_{nt}(t)$: Probability density function(p.d.f) of t_{nt}; t_{nv}: Network delay of a frame; with its p.d.f. $p_{nt}(t)$; $\tau_{nv} = t_{nv}/t_f$; t_{it}: Intention delay of a voice packet; t_{st}: The system delay of a packet, $t_{st} = t_{nt} + t_{it}$; t_{dt}: Total delay of a packet; $t_{dt,n}$: Total delay of the n-th packet; t_{iv}: Intention delay of a video frame; t_{sv}: The system delay of a frame, $t_{sv} = t_{nv} + t_{iv}$; t_{dv}: Total delay of a frame; $t_{dv,n}$: Total delay of the n-th frame; $p_x(y)$: p.d.f.of the random variable y_x ; $P_x(y)$: Distribution function of the random variable y_x; $P_x(y) = \int_{-\infty}^{y} P_x(t)dt$; T_x: The constant value for the random variable of t_x or just a constant.

Fig. 4.14 shows the synchronizing of voice and video data streams. It manifests the data delivery process with the E policy for video and the I policy for voice.

The CD protocol with the E policy is chosen for video and the I policy for voice. Therefore the system delay of the first packet is : $t_{st} = t_{nt} + T_{it}$ and of the first frame is $t_{sv} = t_{nv} + T_{iv}$. The total delay time of the first video frame can be expressed as $t_{dv,0} = max(t_f + t_{nv} + T_{iv}, t_p + t_{nt} + T_{it}) = max(t_{nv} + T_v, t_{nt} + T_t)$; where $T_v = t_f + T_{iv}$; $T_t = t_p + T_{it}$;

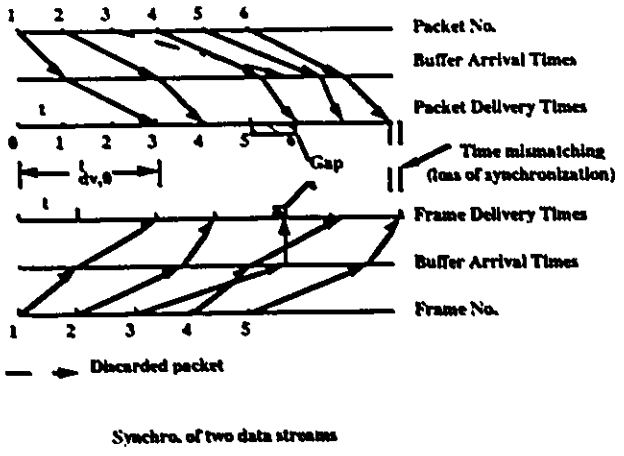


Figure 4.14: Synchronization of Two Data Streams

and for the first voice packet $t_{dt,0} = t_{dv,0}$ since they are started out at the same time.

As the p.d.f. for $(t_{nv} + T_v)$ is $\int_0^\infty p_{nv}(t-\tau)\delta(t-T_v)d\tau = p_{nv}(t-T_v)$; and similarly, the p.d.f. for $(t_{nt} + T_t)$ is $p_{nt}(t-T_t)$; So the p.d.f. of $t_{dv,0}$ is:

$$p_{dv,0}(t) = p_{nv}(t-T_v)P_{nt}(t-T_t) + p_{nt}(t-T_t)P_{nv}(t-T_v) \tag{4.5}$$

$$P_{dv,0}(t) = P_{nt}(t-T_t)P_{nv}(t-T_v) \tag{4.6}$$

$t_{dt,0}$ has the same density function as $t_{dv,0}$.

With exponential random network delay assumptions, we can obtain the numerical results of the delay before the display of a segment in SRTDD (Simultaneous Real Time Data Delivery).

Assume the network delay of a voice packet is exponentially distributed as $p_{nt}(t) = \lambda_t e^{-\lambda_t t}$, and the network delay of a video frame is: $p_{nv}(t) = \lambda_v e^{-\lambda_v t}$. Then the delay distribution of Formula 4.6 comes out:

$$P_{dv,t}(t) = (1 - e^{-\lambda_t(t-T_t)})(1 - e^{-\lambda_v(t-T_v)}). \tag{4.7}$$

If we assume one talk-spurt taking $1.32sec$, 40 packets, then the number of frames in its matching VAU is $30frames/sec \times 1.32sec = 39.6(frames)$. For simplification, we can take $t_p = t_f$. Assume the mean value of the network delay is two slots for both the voice

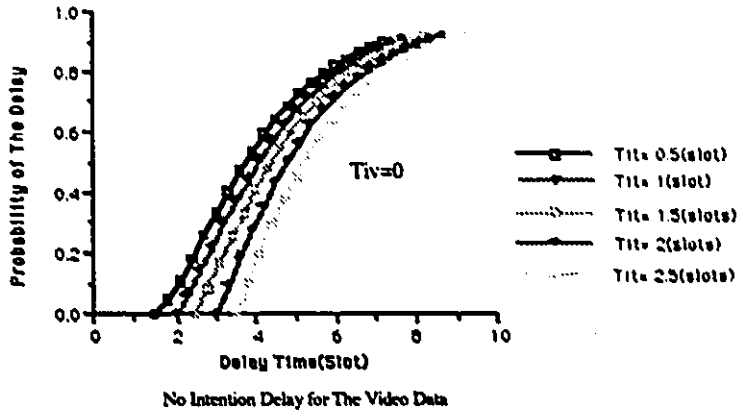


Figure 4.15: Delay Distribution at The Receiver (1)

and the video. The probability distribution of the delay time for the display of a segment is manifested in Fig. 4.15, with respect to the parameters T_v and T_t and normalized on the t_p . In Fig. 4.16 the intention delay of video is two time slot which is a practical number. Increasing the intention delay in either data stream will certainly increase the delay time before the display of the segment. Parameters should be chosen according to the delay constraints on the segment display (i.e., end-to-end response time constraint) and the necessary intention delay required by each single data stream in every individual medium. The parameters for the threshold waiting time in the SCMC can be determined on the distributions of the segment delay.

The following video frames will obey the E policy. The delay before the display of the n-th frame will be: $t_{dv,n} = max(t_f + t_{nv}, t_{dv,n-1})$; so $p_{dv,n}(t) = p_{nv}(t - t_f)P_{dv,n-1}(t) + P_{nv}(t - t_f)p_{dv,n-1}(t)$; Then

$$P_{dv,n}(t) = \int_{-\infty}^{t} p_{dv,n}(\tau)d\tau = P_{dv,n-1}(t)P_{nv}(t - t_f) = P_{nv}^{n}(t - t_f)P_{dv,0}(t) \tag{4.8}$$

Thus the pdf of n-th frame in VAU:

$$p_{dv,n}(t) = nP_{nv}^{n-1}(t - t_f)p_{nv}(t - t_f)P_{dv,0}(t) + P_{nv}^{n}(t - t_f)p_{dv,0}(t) \tag{4.9}$$

Following packets are displayed continuously, slot by slot in a determined way. Late packet arrivals will be discarded. The average number of the missing packets can be obtained as

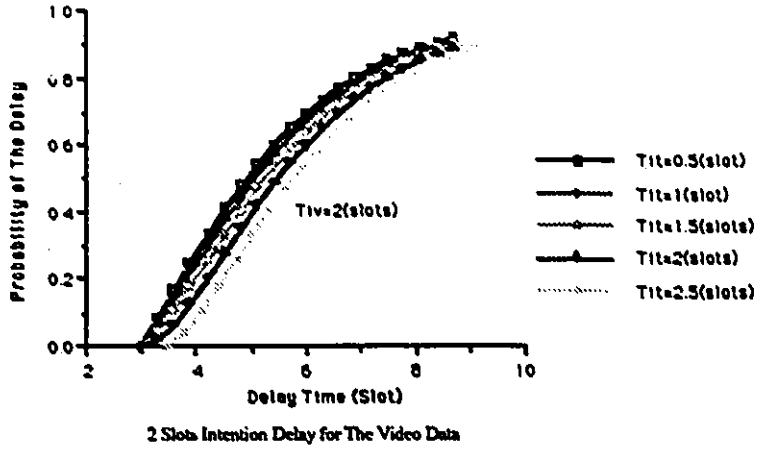


Figure 4.16: Delay Distribution at The Receiver (2)

given below:

$$\begin{aligned} Prob.(\textit{the packet is discarded}) &= p(t_p + t_{nt} > t_{dt,0}) \\ &= \int_{t_p}^{\infty} p(t_{nt} > \tau - t_p) p_{dt,0}(\tau) d\tau \\ &\equiv p_0; \end{aligned}$$

As the distribution of the number of packets in a voice component is: $p_t(n) = (1 - \rho_t)\rho_t^n, n = 0, 1, 2, ...$, then we can take the event of discarding a packet as a Bernoulli trial with probability p_0. Hence the number of the discarding packets, given n packets in a talk-spurt, is Binomial distributed as: $p(D = d|N = n) = \begin{pmatrix} n \\ d \end{pmatrix} p_0^d (1 - p_0)^{n-d}$; Therefore the average number of the discarded packets is

$$E(D) = E(E(D|N)) = p_0 \frac{1}{1 - \rho_t} \tag{4.10}$$

The numerical examples are shown in Fig. 4.17 with the same assumption as the above. We notice that the discarding of voice packets is decreased while increasing the intention delay of voice and video. This is another factor considered for choosing the intention delay time.

If another protocol for the video component is chosen as the first frame will not be played out until the arrival of the second frame, the delay of the first arrivals in the segment

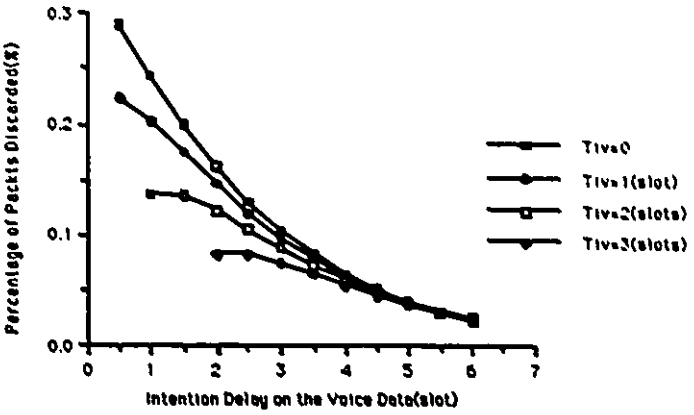


Figure 4.17: Average percentage of Packets Discarded

will then be changed to: $t_{dv,0} = max(t_f+t_{nv}, 2t_f+t_{nv'}, t_p+t_{nt}+T_{it}) = max(2t_f+t_n, t_{nt}+T_t)$; where $t_{nv'}$ is the network delay of the second frame; and $T_t = t_p + T_{it}$.

In our assumption, the network delay on every packet is i.i.d. distributed. Therefore the t_{nv} and $t_{nv'}$ are independent random variables with the same distribution. Thus the pdf of $t_{dv,0}$ is

$$\begin{aligned} p_{dv,0}(t) &= p_{nv}(t-t_f)P_{nv}(t-2t_f)P_{nt}(t-T_t) + p_{nt}(t-T_t)P_{nv}(t-2t_f)P_{nv}(t-(t_f+T_t)) \\ &\quad + p_{nv}(t-2t_f)P_{nv}(t-t_f)P_{nt}(t-T_t) \\ P_{dv,0}(t) &= P_{nv}(t-t_f)P_{nv}(t-2t_f)P_{nt}(t-T_t) \end{aligned} \tag{4.12}$$

The distribution of the $t_{dt,0}$ is the same as that of the $t_{dv,0}$.

The case of real-time voice and video is taken here as an example to demonstrate the segment delivery of SRTDD, as well as some performance results. The analytical results obtained are very limited because of the simple and ideal mathematical model used. However, the scheme is practical and the example gives us a first insight into the performance of a SRTDD system. In Part II, we will look into the issue of performance modeling for the segment delivery scheme, using better modeling techniques. Then some interesting performance measures, such as the synchronization errors with respect to different segment lengths, the buffer size needed at the receiver, etc. will be easily obtained.

4.5 Summary

In this chapter, we have presented our SRTDD control scheme for the multimedia data transmission. The scheme is applied with a layered model. The model enhances the OSI standard by the multimedia session management as well as the multimedia data delivery protocols. Thus the session and transport layers in this new model are able to handle the time relationships in the simultaneous real-time data delivery.

According to the Synchronization QoS parameters, the segment delivery scheme constrains the synchronization errors during the data delivery, to meet the application requirements. Different segment delivery schemes are proposed. We have also given a practical example of the segmentation on real-time voice and video streams. Our above contributions were published in [21][20][22].

Chapter 5

Coordination and Causal Ordering in Multimedia Cooperative Systems

In this chapter, we investigate the synchronization issues in a multimedia cooperative system. This multimedia cooperative system has been developed in the Multimedia Communications Research Laboratory (MCRLab) in the Department of Electrical Engineering, University of Ottawa [17][18]. On this CSCW (Computer Supported Cooperative Work) testbed, a multimedia collaborative application of joint editing [12] was developed. In this chapter, we describe the synchronization control algorithms designed for the multimedia joint editing application.

First, a synchronization model is established for collaborative applications. Based on the model, we devise a distributed ordering algorithm to coordinate the collaborations in the shared multimedia space. The algorithm provides concurrent access to the shared data, and at the same time preserves a consistent causal ordering of multimedia messages at all the participants' sites.

5.1 Synchronization Model for Multimedia Collaborative Applications

5.1.1 Synchronization Control in Shared Multimedia Space

In multimedia collaborations, users are working in a shared multimedia space over the communication network. In the application of joint editing, this shared space is usually occupied by a multimedia document. Participants are editing and modifying the portions of the document, where the data in various media, e.g., video, graphics, text, audio, etc., are used. When one of the participants makes a modification on a portion of the document, the new version of this portion shall be sent and displayed to all the other participants. Thus multimedia messages are sent over the network. Eventually, when the editing session is finished, a new multimedia document is obtained. Here, we are discussing the editing session where only a unique version of the document should be generated from the editing, although any participant can keep his/her different "private" versions. As we have discussed in Chapter 1 of this thesis, the synchronization model for this system should handle the concurrency control for accessing the shared space, and at the same time order the multimedia messages at all the participants' sites for display.

In the review given in Chapter 2, the $\Delta-$causal control scheme [47] was discussed. In developing cooperative applications, the $\Delta-$causal control is very favorable for real-time voice conferencing. However, in document based collaborative applications such as multimedia joint-editing or collaborative software development, a certain complement is needed. First, the $\Delta-$causal supports the applications which evince a real-time generating and comprehension of messages, e.g., voice conversation. It has no control over the situation where a received message, e.g., a new paragraph in a document, is displayed and erases the old display before the viewer can comprehend, when the two data messages are received continuously. While this is not a problem in voice conversation, loss of information can be disastrous in document collaboration. Meanwhile, as addressed in [47], the context of the message-id's recorded in a message sent can be incomplete due to the delay

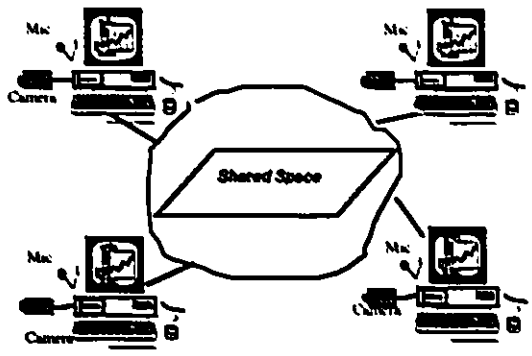


Figure 5.1: An example of a multimedia group

of certain messages. Only a partial ordering can be achieved at the destinations. This, plus a possible "fast-erase", may turn out different versions of the presentation at the receivers, which can be fatal to group collaboration. Therefore, on top of the $\Delta-$causal control which is in the transport protocol proposed, a management of multimedia message ordering is required for the collaborative applications.

We did not implement the control method introduced in [38] for cooperative applications because it is centralized. Meanwhile it requires a special transport system [38].

Now, we present the synchronization model for the multimedia collaborative applications. Then we will describe the distributed control algorithms developed for the joint editing session.

5.1.2 Synchronization Model for Multimedia Collaboration

The notion of stream is applied in the synchronization model. In the model, a stream is assigned to the data transmission of a certain medium. Multiple streams are always employed for a multimedia collaborative application. An abstraction of multimedia "*group*", which contains multiple streams, is defined. In the system, a shared document that is accessed by some distributed participants defines a *group*. The shared document is in the shared space of the group. In Fig. 5.1, an example of a multimedia group containing the participants is shown. These participants are the nodes in the space. A node that outputs

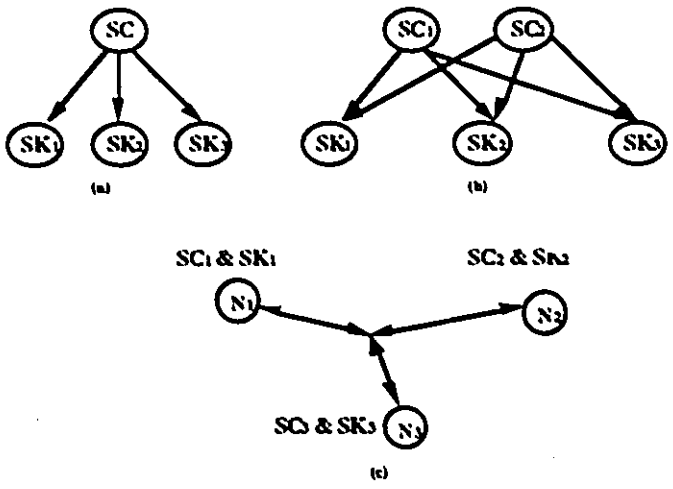


Figure 5.2: Source & sink

data to the group (e.g., editing the document) is a source of the group. A receiver node attached to the group viewing the shared document is a sink of the group. Physically, source and sink devices are media related. A source can be a mouse, a keyboard, a microphone or a camera, etc. A sink may be a display window or a set of display windows, as well as other play-out devices (e.g., a speaker). Thus, we have group-oriented multimedia collaborations. In a group-oriented collaboration, a group always has multiple sinks. There can be either one or multiple sources (Fig. 5.2). Most of the time, both a source and a sink reside at a node (Fig. 5.2(c)).

Formally, a multimedia group is defined as follows.

Definition 5.1 *A multimedia group* $P = \{N, S_c, S_k, SP\}$ *is established on a shared multimedia document, where*

$N = \{N_1, N_2, \cdots, N_m\}$ *is the set of nodes in* P, *and*

$$N_i = \begin{cases} \{sc_i\} & N_i \text{ is only a source node;} \\ \{sk_i\} & N_i \text{ is only a sink node;} \\ \{sc_i\} \cup \{sk_i\} & N_i \text{ is both a source and a sink node.} \end{cases}$$

$S_c = \{sc_1, sc_2, \cdots, sc_k\}$ *is the set of sources in* P;

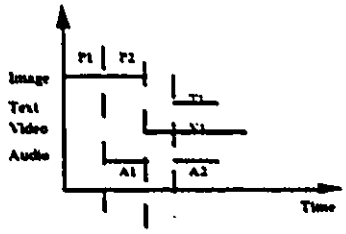


Figure 5.3: A multimedia scenario

$S_k = \{sk_1, sk_2, \cdots, sk_m\}$ *is the set of sinks in* P;

SP = {*The shared space*}

= {*The logic data space which contains the set of shared multimedia objects*}.

The shared spaces of different groups are disjoint. That is, $SP_i \cap SP_j = \phi$ holds if $i \neq j$. Although the copies of the document at different nodes in a group have different physical addresses in their workstation memories, the data share the same logic address (space) in the group. A multimedia document consists of multimedia objects which involve temporal relationships [15][5][16]. That is, a multimedia document has a temporal scenario describing its required temporal presentation sequence (see Chapter 2). Fig. 5.3 illustrates the scenario of a multimedia document. In the group, the time relationships among the data objects required in the scenario are mapped into the shared space. The data objects which are displayed during a same time interval will share the same logical space in SP. In Fig. 5.4, the shared space of the multimedia document shown in Fig. 5.3 is depicted. Given a shared space SP, a portion $pt_i \in SP$ may contain data belonging to different objects and/or different media. In a portion, the data belonging to different objects and/or different media are situated on different "logical levels". Thus, the different data can be distinguished when they share a same logical address.

In the synchronization model, the collaborative applications are represented as the collaboration in the logic data space of a group. Synchronization control is thus needed in every group. Since different groups do not share any data objects, they have different

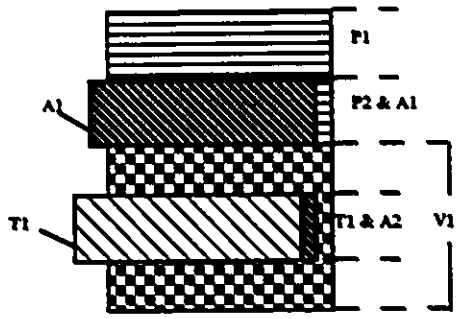


Figure 5.4: The multimedia shared space

subjects of their collaboration. Therefore, there is no synchronization requirement between different application groups. Notice this is not in contradiction with the situation where one application establishes several groups, and a user may join multiple groups at the same time. For instance, in joint-editing of a multimedia document, collaboration of different chapters can be conducted in different groups. An editor can work on two different chapters by joining the two corresponding groups. When the editors want to compile the two chapters together, they can place the two chapters into one group so that the collaboration on both chapters is coordinated.

5.2 The Synchronization Control Scheme and The Algorithm

5.2.1 Causal Dependency and Data Ordering

Before introducing the algorithm, we first address some basic definitions. In a group, if there is only one source, no concurrency control (token) is needed. Simply time stamping the messages at the source will let the sinks in the group "see" the messages at the same and correct order. When multiple sources can transfer data concurrently, causal ordering of the messages at every sink is required.

According to the causal dependency concept introduced in [19] and [47], a causal dependency relation "$\longrightarrow$" in a group is defined as:

Definition 5.2 *Given two messages m_1 and m_2 in group P. $m_1 \longrightarrow m_2$ if and only if one of the following two conditions hold:*

1. both m_1 and m_2 are sent by the same source sc_i in P and m_1 is sent before m_2.

or

2. m_2 is sent by sc_i after receiving m_1 sent by another source sc_j in P.

Now, we adopt Lamport's logic clock [19] to describe the causal dependency defined. At a node N_i, assume a logic clock C_i for group P. C_i is a function that assigns a number $C_i(a)$ to any event a (i.e., sending or receiving a message) at node N_i. $C_i(m_1^s)$ is the logic time that message m_1 is sent by N_i (i.e., sc_i), and $C_i(m_2^r)$ is the logic time that message m_2 is received at sk_i. Apparently $C_i(m_1^s) < C_j(m_1^r)$ is required. The entire system of clocks is then represented by the function C which assigns to every event a the number $C(a)$ where $C(a) = C_i(a)$ if a is an event at node N_i. Every group has its own logic clock at each node and thus its own clock system.

The condition 1 of Definition 5.2 is described by the clock as $C(m_1^s) < C(m_2^s)$ if m_2 is sent later than m_1, where $C(m_1^s) = C_i(m_1^s)$ and $C(m_2^s) = C_i(m_2^s)$. The condition 2 can be interpreted as $C(m_1^s) < C(m_1^r) < C(m_2^s)$, where $C(m_1^s) = C_j(m_1^s)$, $C(m_1^r) = C_i(m_1^r)$ and $C(m_2^s) = C_i(m_2^s)$.

The temporal ordering and the causal ordering of the messages in a group are thus related through the logical clock. To satisfy the Definition 5.2 by the clock system, N_i has to increment its C_i for group P between any two successive group P events (e.g., sending, receiving messages). Meanwhile, the following rule is also necessary

(a) If N_i sends m_i, then m_i contains a time-stamp $T_m = C_i(m_i^s)$. (b) Upon receiving m_i at N_j, N_j sets C_j greater than or equal to its present value and greater than T_m. Thus $C_j(m_i^r) = C(m_i^r) > C(m_i^s) = C_i(m_i^s)$ holds.

With this clock system, the causal ordering defined, which is a partial ordering of the messages, can be described by the time-stamps. A total ordering in the group will also be achieved [19] by using any arbitrary total ordering '$\prec$' of the nodes in the group. A relation of total ordering "$\Longrightarrow$" is thus defined as message $m_1 \Longrightarrow m_2$ if and only if either (i) $C_i(m_1^s) < C_j(m_2^s)$ or, (ii) $C_i(m_1^s) = C_j(m_2^s)$ and $N_i \prec N_j$ [19]. Then in a group,

a total ordering of the messages is formed by the relation "$\Longrightarrow$".

5.2.2 The Ordering Algorithms for Collaboration in Shared Space

Given a group P with the shared space SP, assume that users at the nodes are viewing and/or editing SP. When the user is viewing the document, the data in the SP are presented to the group sink at the user's node. To edit the shared space, the source at the node commits the new edition of a corresponding portion, say pt_i to SP by sending it to all the sinks. This new version of pt_i should be either available to (e.g., "seen" by) all the sinks or none of them. Meanwhile, when two sources are attempting to access the same portion in the SP, i.e., pt_i sent from sc_i and pt_j from sc_j are not disjoint, the two versions pt_i and pt_j should be presented to all the sinks according to the same and correct order. The ordering algorithm is designed for this purpose. The algorithm is performed distributedly at all the nodes of the group.

The assumptions given in [19] are as well taken here. Thus, for any two nodes N_i and N_j, the messages sent from sc_i to sk_j are received in the same order as they are sent. Moreover, every message is assumed eventually received.

1. Collision Detection with Global Causal Ordering

In this algorithm, a waiting queue W_i of editing requests is maintained at every node N_i for the group P. The ordering algorithm is defined by the following six rules.

1) If a source sc_i wants to commit an editing to SP, it generates message $m_i =< T_i : sc_i$ requests $pt_i >$, where T_i is the time-stamp. sc_i puts m_i on its request queue W_i and sends m_i to every other node in the group.

2) When node N_j receives the message $m_i =< T_i : sc_i$ requests $pt_i >$, it puts it in its waiting queue W_j. If there is no message $m_l =< T_l : sc_l$ requests $pt_l >$ in W_j, where $pt_i \cap pt_l \neq \phi$ & $m_l \Longrightarrow m_i$, N_j confirms the request pt_i by sending a time-stamped acknowledgment message to sc_i.

3) When the new edition of the portion pt_i is received by a sink sk_j at time t_j, the

corresponding request message $m_i =< T_i : sc_i$ requests $pt_i >$ will be removed from W_j before a deadline time of $t_j + \Delta\tau$ is exceeded. The receiving time of pt_i will be recorded if a request message $m_j =< T_j : sc_j$ requests $pt_j >$ in W_j & $pt_i \cap pt_j \neq \phi$.

4) After a node N_j deletes a request $m_i =< T_i : sc_i$ requests $pt_i >$ from its waiting queue W_j, it will send a time-stamped acknowledgment to all the nodes N_k. N_k is chosen by this rule: there is a request $m_k =< T_k : sc_k$ requests $pt_k >$ in W_j where $pt_k \cap pt_i \neq \phi$ and there is no other message $m_l =< T_l : sc_l$ requests $pt_l >$ in W_j where $pt_l \cap pt_i \neq \phi$ & $m_l \Longrightarrow m_k$.

5) sc_i can commit its editing of pt_i into SP when the following two conditions are satisfied: (1) there is a $m_i =< T_i : sc_i$ requests $pt_i >$ in W_i and there is no such request $m_l =< T_l : sc_l$ requests $pt_l >$ in W_i where $pt_l \cap pt_i \neq \phi$ & $m_l \Longrightarrow m_i$. (2) sc_i receives a confirmation message from every other node time-stamped later than T_m. $T_m = C(m^*)$ is the sending time of the latest editing on the portion pt_m committed by a certain sc_m received at sc_i, where $pt_m \cap pt_i \neq \phi$ holds.

6) After sc_i broadcasts the edition of pt_i at time t_i, the corresponding request message $m_i =< T_i : sc_i$ requests $pt_i >$ will then be deleted from W_i at or before time $t_i + \Delta\tau$. sc_i can also cancel its request before start to commit an edition by sending a cancellation and remove the request from W_i. After receiving such a cancellation message at node N_j, the corresponding request message is deleted from W_j.

2. Discussions

It can be observed that for every part in the document (i.e., the SP), a mutual exclusive access (editing) is maintained by the algorithm. This is done by the total ordering of the request messages for the portions which are not disjoint. In a group, if there is a "conflicting" request m_l generated by sc_l earlier than the request m_i, sc_i may send its request m_i without being aware of m_l, i.e., before m_l received at sc_i. However, sc_i will not be able to commit the editing because sc_i will not obtain the acknowledgment from sc_l until the editing committed by sc_l is received and "viewed" (or kept for a certain time of $\Delta\tau$) at all the sinks in the group. The rule 4) is employed to prevent a false acknowledgment. In the above example, if a sink sk_m receives m_i earlier than m_l, sk_m

will acknowledge m_i according to rule 2). Then m_l arrives at sk_m and acknowledgment for m_l is also sent (rule 2)). to let sc_l start editing. m_l is received at sc_i too. sc_i will wait for another acknowledgment from sk_m generated according to the rule 4) to make sure that the editing of sc_l is finished and "viewed" at sk_m. This is stated by rule 5) condition (2). Thus, the order is maintained in the case where the conflict occurs in the editing.

Meanwhile, in the algorithm, a portion committed to the SP will be "locked" at every sink for some time. After $\Delta\tau$, the portion will be released so that new change at and near this place may be made. This ensures that every sink can view the modification before it is changed again.

It can also be observed that the algorithm allows concurrent access (editing) in the SP if their corresponding portions are disjoint, and are not conflicting with any other unreleased portion in the SP. However, before every sink has received a certain portion and viewed it (can keep it for $\Delta\tau$ without making an extra local copy), new editing on this part of the document cannot be conducted. Thus a consistent version of the document can always be obtained at all the sinks after every change, although only a partial ordering between disjoint portions pt_i and pt_j is maintained.

Recall that a portion in SP may contain data of different objects at different logic levels. Therefore when a user requests editing of the data of one single medium (object), the other data located at the same address on different logical levels will be automatically included into the portion as well as locked during the locking time.

In the algorithm, the size of portion can play a key role in the control. The size of a request portion is usually chosen by the user. A minimum as well as a maximum size value can be set by the system. If the minimum size of a portion is big enough so that for every two portions $pt_i, pt_j \in SP$, $pt_i \cap pt_j \neq \phi$ holds, a mutual exclusive access to the whole SP is managed. That is, all the editings are totally ordered at all the nodes in the group. When a change is desired on portion pt_j because of another modification pt_i observed, the editor may wish to ensure a causal ordering of the two changes in pt_i and pt_j at all the sinks. This can be done by making the portion pt_j (pt_j may contain disjoint parts in SP) s.t. $pt_i \cap pt_j \neq \phi$. Then every sink in the group will "see" the change in pt_i

before that in pt_j.

5.2.3 Implementation of The Synchronization Control

The implementation of the synchronization control algorithm utilizes the window interface provided by the cooperative system designed. The window interface includes four different types of windows, the group window, the user window, the conference window and the private window [17].

In the joint editing, the group window at each workstation presents the shared space to the participants. The group window thus defines a multimedia group. A group window at a workstation is a sink in the group. Different objects may be contained in the group window for different media, e.g., a group window may consist of a sub-window for text display and a speaker.

The concurrency control for accessing the group window is managed by the ordering algorithm introduced above. To implement the algorithm, the user windows are employed. If a user A is granted a portion request for editing, the portion will be copied into the user window at A's workstation. The owner of this user window is then user A. That is, only A can modify the document in it. Meanwhile in the group window, the portion is shown to be locked. The conferencing window displays the information about who has locked the portion. Any user at a node N_i who is interested in the locked portion can open A's user window at node N_i locally to view the editing of A. After the editing is finished, the new version of the portion will be committed into the group window by user A.

In this implementation, the method of "group decomposing" is adopted. A user window under a certain owner establishes a new sub-group in the system. We call it a user window defined group. In this user window defined group, there is only one source residing at the owner's node. Multiple sinks may be involved. Thus, in a user window defined group, no concurrency control is needed. The SP of a user window defined group is just the reserved portion. Multiple user window defined groups can be established concurrently under one group window. When a user does not want others to view his/her editing of a portion before it is committed to the group window, the user can copy the

portion into his/her private window after the request is granted. The editing in the private window cannot be viewed by anyone else. Thus a private window belongs to no group as no SP is defined. However, if the user wants to commit the portion to the shared space, the portion has to be sent to the group window through the synchronization control algorithm. The implementation provides the flexibility to the users during collaborative applications. Users only follow very simple rules and the coordinations can be achieved. The windows well support a joint-editing interface in the real application, as long as not too many users are involved. The drawback of this interface is the limit of the number of the windows that can be opened on a workstation.

5.3 Summary

As addressed above, in multimedia conversational applications, coordination and causal ordering are required. In this chapter, we established the synchronization model for multimedia collaborative applications, a typical conversational application. Based on the model, we described the synchronization algorithm designed for the multimedia joint editing application. **Our above contributions were published in [24].**

In this Part I of our thesis, we have presented the synchronization architectures and algorithms for both the multimedia presentation as well as the conversational applications. From the overview of the literature, it can be seen that most of the reported work was done at the level of the system description, specification and architecture design, or at the level of proposal for the control protocols and algorithms. The performance of the proposed system has rarely been evaluated quantitatively. In order to implement any multimedia synchronization control, relevant performance measures for the control scheme are required. In [36], simulation was used to evaluate the performance of the proposed system. However, the analytical performance modeling techniques have rarely been addressed for multimedia synchronization systems [23]. On the other hand, the multimedia synchronization problem raises a challenge to the techniques for performance modeling and evaluation of the distributed computing as well as communication systems. It is well

known that the available analytical modeling methods have encountered various difficulties in modeling and evaluating the synchronization aspects in communications [7-1]. The performance evaluation of the multimedia data integration in a distributed system still remains an open problem. The difficulty arises from the fact that the cooperation of the servers and their temporal characteristics are not easy to model. Neither are the temporal features of the multimedia data streams delivered in real-time. It can thus be envisioned that the development of methods for the modeling of multimedia synchronization will not only take a fruitful direction, but also contribute to the general performance modeling techniques.

In the second part of this thesis, we are going to present our studies on the analytical performance modeling for multimedia synchronization systems. In Chapter 6, we will first review the analytical performance modeling methods for computer communication systems. Our purpose is to search for possible performance models for the evaluation of synchronization control in multimedia communications.

Part II

Performance Modeling of Multimedia Synchronization Control

Chapter 6

Review

Two of the best known analytical modeling methods used for computer communications performance evaluation are Queueing Networks (QNs) and Petri Nets (PNs). QNs have the incontestable advantage of combining expressive power and solution efficiency. However, QNs have serious limitations in modeling concurrency in distributed systems; they are unable to represent in a direct way the synchronization in communications. *Ad hoc* extensions and approximation solution methods have been proposed to overcome some of these shortcomings [61], but they are not general or straightforward enough.

Petri Nets (PNs) with timing properties are the second type of models used for performance analysis of concurrent systems. The representation of synchronization in PN models is fundamental to the notations. Classical PNs consist of a set of places, a number of tokens that reside at places, and a set of transitions that represent events causing tokens to move from one place to another. For performance modeling of computer systems, the classical PNs have been extended by introducing timing properties. For instance, "stochastic" Petri Nets (SPNs) are one of these extensions.

The big advantage of PNs is their modeling power. The representation of the modeled system can be done at almost any level of detail, from a very coarse to a very fine grain. Intuitively, PNs or PN extensions appear to be more suitable performance models for the representation and analysis of synchronizations in multimedia communications. Now, we first present the SPNs, one of the PNs extensions.

6.1 Introduction to Stochastic Petri Nets

The introduction of time into Classical Petri Net models allows the description of the dynamic behavior of a system, i.e. the state evolution and the duration of each action of the system. There exist several ways of associating time onto a classical PNs [53][55][56]. The most widely accepted one is the Stochastic Petri Nets (SPNs), introduced by Molloy [53] and Natkin [55].

6.1.1 Stochastic Petri Nets

SPNs are obtained by associating with each transition in a Petri Net an exponentially distributed random variable that expresses the delay from the enabling to the firing of the transition. A formal definition of SPNs is the following:

$$SPN = (P, T, A, \mu_0, R)$$

where

P: a set of places; $P = \{p_1, \ldots, p_n\}$

T: a set of transitions; $T = \{t_1, \ldots, t_m\}$

A: a set of transition input arcs A_i and transition output arcs A_o; $A_i = (P \times T)$, $A_o = (T \times P)$ and $A = (A_i \bigcup A_o)$;

A marking μ is an assignment of tokens to the places. It is usually denoted by $\mu = (m_1, \ldots, m_n)$, where m_i is the number of tokens in place p_i and

$$\mu_0 = (m_{01}, \ldots, m_{0n})$$

is the initial marking, where m_{0i} denotes the number of tokens in place p_i in the initial marking μ_0.

R is a set of possibly marking dependent firing rates associated with the transitions in SPNs.

$$R = \{r_1, \ldots, r_m\}$$

When necessary, the dependence upon a given marking μ of the firing rate of transition t_i will be denoted $r_i(\mu)$.

SPNs are isomorphic to continuous time Markov Chains [53]. The reachability set $R(\mu_0)$ of a SPN constitutes the state space of the associated Markov Chain. The transition rate from state i (marking μ_i) to state j (marking μ_j) is equal to the firing rate of the transition that is enabled by marking μ_i and whose firing generates marking μ_j. A SPN is ergodic, if it generates an ergodic continuous time Markov Chain. It is possible to show that a Stochastic Petri Net is ergodic, if μ_0, the initial marking, is reachable from any $\mu_i \in R(\mu_0)$[54].

If the Markov Chain is ergodic, the steady state probability distribution of markings is computed by solving:

$$\pi \mathbf{Q} = \mathbf{0}$$

with the additional constraint

$$\sum_i \pi_i = 1$$

where $\mathbf{Q}$ is the infinitesimal generator and π is the vector of the steady state probabilities. Other parameters of performance interest may be obtained from π.

SPN models have been applied to the analysis of the multiprocessor systems [58], communication networks [62], etc. Refer to [63] for more references.

A major difficulty in modeling a system by Stochastic Petri Nets is the explosion of the Markov Chain state space when the system is getting complicated. The modeling power of the SPNs is actually paid with the complexity in solving the models. In fact, a "simply looking" SPN model can generate a very big state space. As a result, the numerical solution of even middle-sized SPN models can often be intractable. The inefficiency in solving the SPN models has always been an open problem, especially in the performance modeling of communication systems [72]. It is known that the distributed multimedia systems involve complicated temporal cooperations among the system elements such as the database servers, the users, etc. Therefore, when modeling the multimedia synchro-

nization system using SPN models, a big state space is expected and efficient solution algorithms have to be investigated.

6.1.2 Generalized Stochastic Petri Nets

In order to cope with the state explosion problem in SPN models, Marsan et al.[58] introduced the Generalized Stochastic Petri Nets (GSPNs). Marsan observed that often it is not desirable to associate a random time with each transition, since one would rather associate times only with the events believed to have the largest impact on system performance. A typical example is the cases in which the operating sequences of a system comprises activities whose durations differ by orders of magnitude. It is then conceivable to model the brief activities only from the logical point of view, whereas time is associated with the longer lasting ones. This choice can represent the system modeled more precisely. Meanwhile by doing so, the number of states of the associated Markov Chain model is reduced, thereby reducing the solution complexity.

GSPNs are a logical extension of the SPN. In a GSPN model, the set of transitions is partitioned into a set of immediate transitions (with infinite firing rates) and a set of timed transitions. Furthermore, immediate transitions fire in zero time once they are enabled, and timed transitions fire in an exponentially distributed random time. Several transitions may be simultaneously enabled by a marking. Assume that the enabled transitions comprise the set H. If H contains only timed transitions, a transition $t_i (i \in H)$ fires with probability:

$$\frac{r_i}{\sum_{k \in H} r_k}$$

exactly as with SPNs. If H comprises one immediate transition and several timed transitions, the immediate transition will fire with probability 1. However, markings at which several immediate transitions are enabled fire with equal probability unless a probability distribution over the enabled transition selection is specified. This probability distribution is called random switching distribution and the set of enabled immediate transitions is said to form a random switch [58].

It is necessary to point out that in general the reachability set of a GSPN is a subset of the reachability set of the associated PN (or SPN) , because precedence rules introduced with immediate transitions do not allow some states to be reached. Moreover, the reachability set of the GSPN can be divided into two disjoint subsets, one of which comprises markings that enable timed transitions only, while the other comprises markings that enable immediate transitions. The state (or marking) of the former type is called "tangible" and a state (or marking) of the latter type "vanishing".

GSPNs are isomorphic to a stationary Embedded Markov Chain (EMC). Therefore, the analysis of GSPNs is based on solving the underlying EMC. The EMC of a GSPN is observed to spend a nonnegative amount of time in the tangible states, while it transits in zero time through markings enabling timed transitions. In solving the EMC of GSPNs, the vanishing states are removed from the EMC and a Reduced Embedded Markov Chain (REMC) over tangible states only can be defined. Then the transition probability matrix U' of the REMC can be obtained, which only contains the transitions between any two tangible states i and j. The probability transition matrix U' is usually smaller than the transition matrix U of the original EMC. By distinguishing tangible and vanishing states in the GSPNs' state space, the computational complexity of solving the EMC is reduced. Thus, it becomes possible for us to apply the GSPN model to relatively large systems [58].

GSPNs have improved the solution techniques as well as the modeling ability of SPNs. However, the state space of GSPNs is still too large to solve.

6.1.3 Colored Stochastic Petri Nets

Colored Stochastic Petri Nets (CSPNs) were proposed by Zenie [64]. A CSPN associates a set of colours with tokens indicating the identities of tokens. Places and transitions have also been attached a set of colors. The color attached to a token may be changed by transition firing.

The definition of a CSPN is as follows:

$$CSPN = (P, T, C, A, \mu_0, L)$$

where $L = l_{ij}, i \in T, j \in C$ is a set of color-dependent firing rates associated with each transition. P, T, A and μ_0 are the same as in SPN. For the convenience of performance modeling, CSPNs were extended to Colored Generalized SPNs (CGSPNs) by Li and Georganas [52]. The transitions in CGSPNs are separated into immediate transitions and timed transitions. CGSPNs were defined in the same way as CSPNs, where now the array L contains only the number of timed transitions in CGSPNs. Occasionally, one transition may be simultaneously enabled in several colors by a marking μ. The priority firings for different colors have also been introduced into CGSPNs for the purpose of performance analysis [52]. Different colors are thus allowed to have different priorities concerning transition firings. Therefore, if several transitions enabled by a marking μ, only the single colour which has the highest priority may fire. If the set of enabled transitions, H, consists of only timed transitions, and color j is the highest priority color among those enabled transitions, then transition $t_i (t_i \in H)$ in color j is fired with probability:

$$\frac{l_{ij}}{\sum_{k \in H'} l_{kj}}$$

and the mean time the net stays in that state is:

$$\frac{1}{\sum_{k \in H'} l_{kj}}$$

where H' comprises all the enabled transition in color j. If H comprises both immediate and timed transitions, only immediate transitions can fire, same as that in the GSPNs. A random switch is also used in CGSPNs to specify the firing probabilities for the immediate transitions when they have the same color priority.

The CGSPNs are distinguished for their ability to identify different classes. Since multimedia communication systems involve data objects and streams in different media, which naturally fall into different classes, CGSPN models offer a method to represent multimedia processing and communications. In [52], CGSPNs were applied to model a data/voice integrated system. The transport protocols for the integrated voice and data

transmissions were evaluated. The voice and data were presented by different colors in the CGSPN model and had different color priorities during the transmission on the shared high-speed communication link. It can be seen that the CGSPNs are quite promising for the performance modeling and evaluation of multimedia communication systems.

However, CGSPNs are usually solved by unfolding into GSPNs according to different colors. This unfolding procedure generates a very large state space for the resulting GSPN. Thus, in performance modeling of multimedia systems, the modeling power of CGSPN is hampered by the difficulties in solving the models.

From the above, the SPNs and their extensions evince the possible representation power for the synchronization aspects in multimedia communications. However, it can be seen that a main disadvantage of SPNs is their lack of efficient solution methods. The exact construction and solving of the generated Markov chain model suffer from exponential growth of the state space, and thus severely restrict the size of tractable models.

6.2 Solution Techniques for SPNs: The Locally Balanced Stochastic Petri Nets

As mentioned above, it is difficult to solve the underlying MC of SPNs directly because of the construction of the exponentially exploding state space of the Markov Chain, and the computation of the huge transition rate matrix. For the last two decades, people have been trying to search for efficient ways to solve SPNs. The approaches fall into two classes: exact algorithms and approximate algorithms. Product form solutions, which were originally developed in QNs, have been adopted in the SPNs as the exact computational technique for the class of separable SPNs. On the other hand, approximate algorithms are also being developed. Meanwhile, the idea of decomposing SPNs into smaller models is widely applied in the algorithms designed. In our studies, we have been focused on the exact solution algorithms. Now we present a class of SPNs which has product form solutions.

In developing solution algorithms for SPNs, Lazar and Robertazzi presented a class of Product Form SPNs[65][66]. As long as a SPN satisfies the condition given in [66], its steady state probability distribution will satisfy Local Balance equations. This class of SPNs is named Locally Balanced Stochastic Petri Nets (LBSPNs). Li and Georganas have further extended the boundary of LBSPNs and proven that the LBSPNs falling into the new boundary also have product form solutions[48][50][51]. A systematic way for identifying the LBSPN that falls into the new boundary has also been designed.

In this extended class of LBSPNs, a subnet is the basic unit to constitute a LBSPN model. M. Li has given the definition for the subnet of LBSPN [51]. However, the definition was not formally described. In Chapter 8, we will represent the definition in a formal way. In a LBSPN, two subnets never have any common transitions. Nevertheless, they may have common places. After all the subnets are identified, a test procedure can be carried out to check if the SPN model which consists of all these subnets, is a Locally Balanced Stochastic Petri Net (LBSPN) model. The test procedure was stated and proven in [51].

With the existing product form solution for the LBSPN model, instead of solving the MC of the SPN to calculate the steady state distributions, all we have to do is to solve the much smaller MC of every subnet. The details of the method will be described later in Chapter 7.

The existence of analytical form solutions allows a much more computationally efficient solution of SPN models. Instead of trying to solve the global balance equations of the entire underlying Markov Chain of a SPN model, all we need is to calculate the normalization constant in a much simpler way. However, in the works of Li et al. [48][49][50], the computational procedure for the normalization constant was not investigated. In order to obtain the normalization constant, the sum of the probability distributions of all the steady states of the entire Markov Chain (i.e. the marking of the entire net) is required. Since the associated Markov Chain for a given LBSPN may very likely have a large state space, calculating the normalization constant G direct from the distributions of all the steady states can be a tedious job.

When the size of LBSPNs is getting big, decomposition and aggregation techniques may be applied in solving the LBSPNs [49]. The "decomposition by subnet" method was proposed for LBSPNs. Thus the analysis of a LBSPN is resolved into the analysis of subnets, the solutions of which are combined to obtain exact solutions for the original system. By decomposition according to a specific subnet, an aggregated net, which is that subnet with marking dependent firing rates proportional to its original firing rate, can be constructed to represent the original system. This is actually the Norton's theorem developed for LBSPNs [49]. This theorem enables us to present the original system in a concise and yet exact way.

The field of PNs with temporal properties is much younger than the field of QNs, which explains the lack of efficient techniques for performance PN models. In order to conduct the performance analysis of multimedia communications using SPN models, an efficient solution algorithm is a must.

Chapter 7

Performance Modeling of Multimedia Synchronization Systems Using Generalized Stochastic Petri Nets

In this chapter, we will conduct the performance modeling and evaluation of the multimedia presentation applications. We will look into the simultaneous multimedia segment delivery in SRTDD system, and the remote multimedia document creation system with multiple information sources. Relevant performance results will be presented. Then, in the next chapter, we will develop the computational algorithm for Locally Balanced Stochastic Petri Nets (LBSPN) and apply it for the performance evaluation of the collaborative application of joint editing.

7.1 Performance Modeling of Simultaneous Multimedia Delivery

In this study, Generalized Stochastic Petri Nets (GSPN) Models are applied. We look into the performance measures of the multimedia presentation which are controlled by

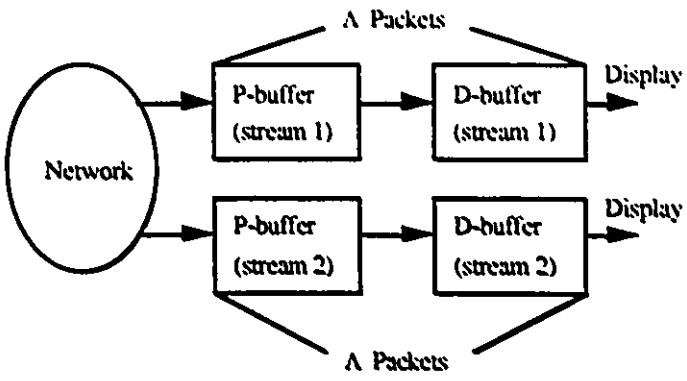


Figure 7.1: The recovery buffer at the multimedia receiver

the Segment Delivery Scheme (SDS). The synchronization errors presented in the final display, the recovery buffer size required, the system throughput, the segment length, etc. will be examined. Recall that in Chapter 4 when an analysis for real-time voice and video was conducted, we could not obtain these performance indices of interest, because of the limit of the modeling method used there.

7.1.1 Markovian Models for the Segment Display

Take a multimedia group which contains two streams. Multimedia segments are divided and delivered. At the receiver the arrival data on both streams will be processed, e.g., decoded for display. Thus, at the receiver, we have a processing buffer (P-buffer) and a display buffer (D-buffer) (Fig. 7.1). After the processing in P-buffer, the data are ready for the display and shifted into the D-buffer to be played out. The D-buffer is actually the synchronization recovery buffer mentioned before (Chapter 4). During the display, the segment header will be checked and thus the data streams will be matched. Assume the receiver provides a buffer space of Λ packets for each data stream. Then the data delivered from the network can be accepted until the buffer is full. Meanwhile, as soon as a packet is played out, the emptied buffer can be replenished by a packet arriving on the stream from the network.

Let us describe the status of the buffers by defining $K_0^i(t)$, $K_1^i(t)$, $K_2^i(t)$, $K_3^i(t)$,

i 1,2. $K_0^i(t) = \{0, 1, ..., \Lambda\}$ is the number of stream i packets in the P-buffer at time t. The $K_1^i(t)$ is defined for the data in the D-buffers

$$K_1^i(t) = \begin{cases} 1 & \text{The first packet of a segment is being displayed on stream } i \text{ at time } t \\ 0 & \text{Other packets or no packet is being displayed on stream } i \text{ at time } t \end{cases}$$

$K_2^i(t) = \{0, 1, ..., \Lambda\}$ records the total number of i-data packets in D-buffer at time t. Finally, $K_3^i(t)$ is used to describe the end of a segment display in the D-buffer:

$$K_3^i(t) = \begin{cases} 1 & \text{The last packet of a segment is being displayed on stream } i \text{ at time } t \\ 0 & \text{Other packets or no packet is being displayed on stream } i \text{ at time } t \end{cases}$$

Based on these descriptions, a multi-dimensional Markov Chain can be established for the multimedia receiver. However, it is not difficult to predict a huge state space, as well as complex transition rates that will be associated. It is impossible for us to construct and solve this Markov chain manually.

In order to handle this complex model, Generalized Stochastic Petri Nets (GSPN) [57] models are applied. As addressed in the previous chapter, the GSPN is an abstract formal graph model useful for representing concurrency, synchronization, and communication [52][58][53]. Every GSPN model has an underlying Embedded Markov Chain (EMC). All the possible markings generated by the GSPN model construct the state space of the EMC. Here, we employ the EMC of the GSPN as the Markovian model for the analysis of SDS.

The GSPN model established for the receiver discussed above is illustrated in Fig. 7.2, with the legends for the places and transitions shown in Table 7.1. The initial tokens in place $P3$ and $P4$ represent the available buffers, including the P-buffer and D-buffer for the two streams, stream 1 and 2, respectively. Thus, instead of a simple Poisson Process, the arrival process on each stream is shaped by the buffer size. Place $P5$ and $P6$ are P-buffers (i.e., represent $K_0^1(t)$ and $K_0^2(t)$) for the streams. Transition $T3$ and $T4$ send data into the D-buffer for display. The initial tokens in place $P7$ and $P8$ stand for the segment size as the number of packets contained in each component in a segment. With the immediate transitions $t5$, $t6$, $t7$ and $t8$, different packets in a segment are distinguished. Place $P15$ and $P16$ are for the first packets in a segment on each data stream,

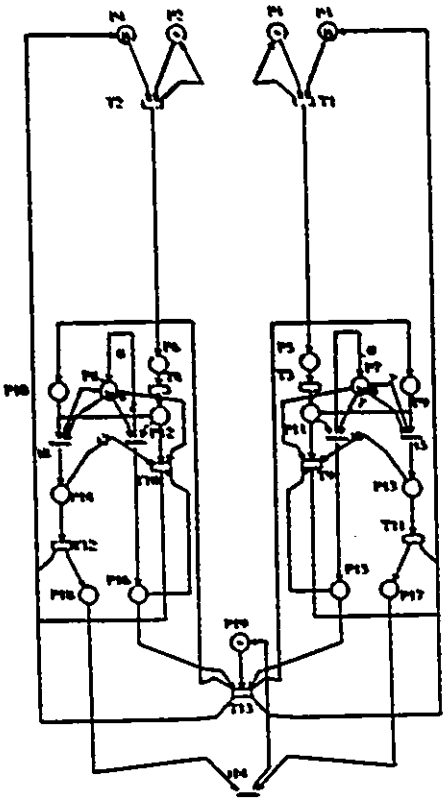


Figure 7.2: The GSPN model

P1 (P2)	network arrivals on stream 1 (2)
P3 (P4)	available buffer space (A) for stream 1 (2)
P5 (P6)	P-buffer for stream 1 (2) data
P7 (P8)	segment size
P9 (P10)	a segment display has begun
P11 (P12)	D-buffer for the intermediate packets of a segment on stream 1 (2)
P13 (P14)	D-buffer for the last packet of a segment on stream 1 (2)
P15 (P16)	D-buffer for the first packet of a segment on stream 1 (2)
P17 (P18)	The display of the last packet of a segment is finished on stream 1 (2)
P19	The display of a segment is finished
T1 (T2)	data arrival on stream 1 (2)
T3 (T4)	data processing on stream 1 (2)
t5 (t8)	last packet in a segment on stream 1 (2)
t6 (t7)	first packet in a segment on stream 1 (2)
T9 (T10)	display of the intermediate packets in a segment
T11 (T12)	display of the last packet in a segment
T13	display of the first packets in a segment
t14	the display of a segment is finished

Table 7.1: Legend for the model

case 1				case 2			
T1	45	T2	45	T1	40	T2	70
T3	45	T4	45	T3	40	T4	60
T9	45	T10	45	T9	45	T10	60
T11	45	T12	45	T11	45	T12	60
T13	45			T13	60		

Table 7.2: Assignment of the firing rate

i.e., $K_1^1(t)$ and $K_1^2(t)$. $P13$ and $P14$, on the other hand, are places for the last packets in a segment on each data stream, i.e., $K_3^1(t)$ and $K_3^2(t)$. Places $P11$ and $P12$ are D-buffers for all the other packets in a segment. Transition $T13$ restores the simultaneity of the two streams at the beginning of each segment display. Transition $t14$ is used to calculate the synchronization errors at the end of every segment. $P19$, $P9$ and $P10$ are places for ordering the delivery of different packets in a segment (Table 7.1).

The GSPN established can now be used to evaluate the performance of the system. By changing the initial token numbers in $P3$ and $P4$, the available buffer size Λ is adjusted. The segment size can be chosen by setting the number of the initial tokens in place $P7$ and $P8$.

7.1.2 Performance Results

A graphically interfaced GSPN software package GreatSPN1.5 is used for solving the models. Two cases are presented here. In the first case, the same data arrival and processing rates are assigned to the two streams. Therefore, only a random disturbance exists, resulting in synchronization errors. In the second case, skews are introduced to the two data streams by assigning the different transition rates in the model. This scenario includes the synchronization errors which result from the different network connection conditions, as well as the different data processing rates.

For the simplicity of the model, the two data streams are assumed to have the same packet size, in terms of the time slot. The D-buffer size is set as $\Lambda = 5$ packets for each

stream. The assignments of the firing rates for the timed transitions in case 1 and case 2 are given in Table 7.2. In case 2, the average delay experienced by a packet on the stream 1, including all the transitions, is about 28 *msec* longer than that on the stream 2. This number assumed is very moderate, compared to the average 100 (msec) jitters that can be incurred in a metropolitan network. After solving the underlying EMCs of the GSPN models, steady state distributions, as well as the throughput of every transition is obtained. Applying Little's Law of $L = \lambda T$, where L is the average population in a system, λ is the arrival rate and T is the average waiting time, other performance indices of interest can be calculated. For instance, the average waiting time at transition $t26$ is actually the average synchronization error at the end of a segment display. Use $\#P22$ to denote the number of tokens in place $P22$. Then $Prob.\{\#P25 > 0 \ \& \ \#P25 < seg.size \ \& \ \#P22 = 0 \ \& \ \#P30 = 0\}$ is the probability of the gap occurrence during the display of the intermediate packets in a segment.

In Table 7.3, the performance estimates of the system of case 1 are presented, with respect to different segment sizes. In the scenario of case 1, identical transition rates are assumed. However, the synchronization errors are still introduced by the random delays. It can be observed that the synchronization performance measures, such as the synchronization errors and gap probabilities are worsened with the increase of the segment size. On the other hand, a smaller segment size degrades the system throughput, and also requires more D-buffer space. Here the average D-buffer occupancy is an upper-bound of the average number of packets in the D-buffer of each stream, since it is the sum of the average number of tokens in the place $P22$, $P28$ and $P30$ ($P23$, $P35$ and $P36$). As the two streams are synchronized, they always have the same throughput.

In Fig. 7.3, the performance indices of case 1 and case 2 are compared. For case 2, the gap occurrence probabilities and the D-buffer occupancy estimates are obtained for the slower data stream 1. The synchronization errors caused by the rate skew in case 2 are considerable. When the segment size is increased to 20 packets, in case 1, the average synchronization error is approaching a limit. However, in case 2, the deterioration is not alleviated. As predicted, during the display, the system of case 2 incurs more gaps and

seg. size (pk)	sync. error (msec)	gap prob.(%)	throughput (pk/sec)	buffer occupancy (pk)
5	31.3	11.52	29.32	2.16
8	37.64	15.85	29.66	2.12
10	41.25	17.77	29.81	2.08
12	44.58	19.14	29.94	2.05
15	49.15	20.58	30.09	2.02
18	53.31	21.6	30.21	2.00
20	55.98	22.12	30.27	1.98

Table 7.3: Performance measures for the system of case 1

requires more D-buffers for synchronization recovery. In both cases, when the segment size is extended to that of 20 packets, the gap probabilities approach their maximum values. Remarkably, in the system of case 2, a bigger segment size cannot generate a higher system throughput. This is because the large synchronization errors result in longer synchronization delay for the recovery at the beginning of each segment. Thus the throughput of the system is restrained. In both cases 1 and 2, we observe that when the segment contains less than 10 packets of each stream, the performance results are more sensitive to the segment size.

With D-buffer at the receiver, an intentional delay can be employed to alleviate the gaps between the adjacent packets during the display. As described in Chapter 4, in our segment delivery system, intentional delay is placed at the beginning of every segment. That is, the first packets of all the streams in a segment will be delayed in the D-buffers before their display. Thus, the intentional delay is performed concurrently with the synchronization delay. Better synchronization performance can be achieved, as shown below.

The intentional delay time can be chosen in different ways [21][2]. For instance, when the the first packets of both data streams in the segment are delayed to wait for the arrival of the following packets [2], Fig. 7.4 presents the performance results. In the

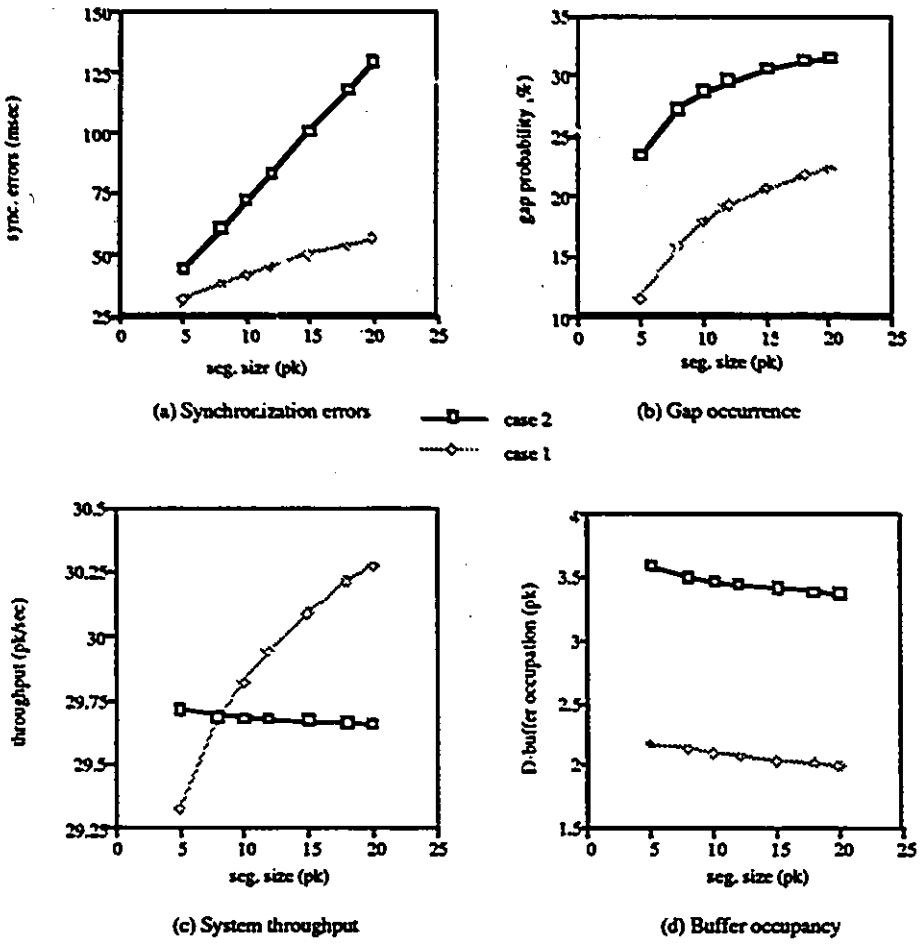


Figure 7.3: Performance comparison of case 1 and case 2

modeling, the D-buffers are increased to $A = 8$. Besides the four performance indices as used above, the delay times experienced at the beginning of every segment in the D-buffer are also given (Fig. 7.4(e)). From the comparison, it can be observed that a smaller gap occurrence can be achieved by placing an intentional delay than by reducing the segment size. However, considering the overall performance measures, segment size has a better impact than the intentional delays.

In this section, we have computed the relevant performance measures for the SDS in SRTDD control. Compared with the performance modeling for the SDS conducted in Chapter 4, the GSPN model applied here can express much more details of the system. Hence more performance indices of interest are obtained.

7.2 Performance Modeling of Multimedia data Integration

In this section, we study the performance for a remote multimedia document creation system while multiple information sources are involved. We assume, in this section, that these information sources are distributed multimedia database servers. As discussed in Chapters 2 and 3, remote multimedia document creation requires multimedia data integration in the distributed system. Thus, Temporal Presentation Control (TPC) is needed to synchronize the servers. Therefore, we have to represent the temporal synchronization of the servers in our performance models. In the study, CGSPN (Coloured Generalized Stochastic Petri Nets) models are going to be employed.

7.2.1 Cooperations of Servers in Distributed Multimedia Data Integration

As mentioned, in order to model the system, we first need to investigate the temporal behavior (cooperation) of the multimedia servers. It is known that the remote document creation, like all the other teleorchestra applications, requires the distributed multimedia

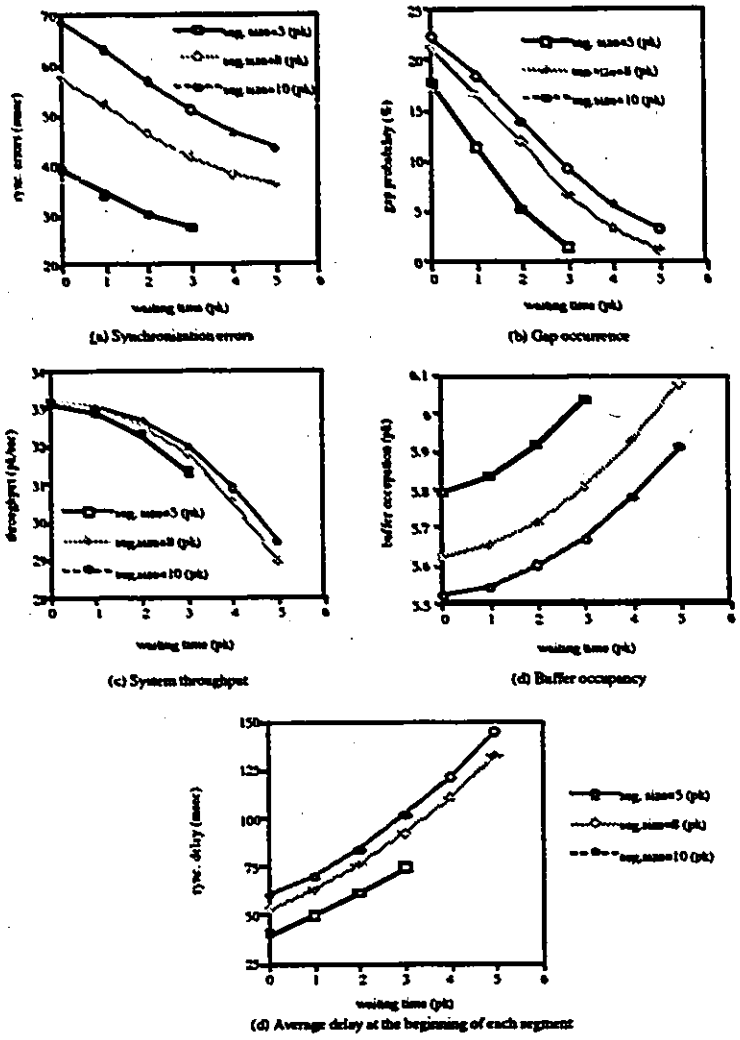


Figure 7.4: The impact of the intentional delay

data integration. Here, we first discuss the different service states and the state changes of the servers in the data integration.

1. States of The Servers in The Data Integration

The multimedia data integration is executed according to the designed scenario (Chapter 3). During the data integration for a given scenario, each server has three service states, idle, active and inactive. If the server is not involved in the data integration of the given scenario, the server is idle for this scenario. Otherwise, it is non-idle for the scenario. At a given time, a non-idle server can have two different states, active and inactive. If the server is active, the server is delivering data for the integration of the scenario. That is, the server is in a task execution at that time. If the server is inactive, then, the server does not have any task execution at that time although it might have delivered data before or will deliver data later for the data integration of the given multimedia scenario.

Apparently, if a server is idle for a given scenario, it remains idle in the whole data integration procedure of this particular scenario. That is, there won't be any state change of the server caused by the data integration of this scenario. On the other hand, it can be seen that although a server may be idle in the data integration for one scenario, it can be non-idle in the integration for another. In the performance model it is assumed that, if a server is non-idle in the integration for a given scenario, it cannot join the integration for other scenarios, until all its tasks for the data integration of this scenario are finished. In the system implemented in the MCRLab, the server generates a server process upon receiving a data integration request for a scenario. This process cannot be used by the other scenarios. The acceptance of data integration request for a new scenario is determined by the concurrency degree that can be supported by the server, as well as the number of processes that are reserved at the server for the application of the document creation, if the server is also involved in other applications. In fact, the "server" discussed in the performance model corresponds to the service process in the real system. Therefore, in order to simplify the performance model, the data integration of one scenario is taken as an atomic action for the servers (which corresponds to the server process) in the data integrations. Thus, the state of active or inactive of a given

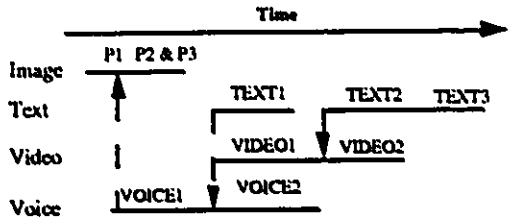


Figure 7.5: A scenario example

server represents a unique state of the server, no matter how many scenarios are being integrated in the system concurrently. On the other hand, the idle state of a server is only concerned with a given scenario. A server is available to the new data integration request only when the server is idle in all the integration jobs that are currently being executed in the system. The basic characteristics of the servers during the data integration are therefore described by the three states.

2. The Task Diagram for The Temporaliy Cooperative Servers

The servers involved in the data supplies for one scenario may start, as well as finish their tasks of data retrieval and delivery at different designated times. It can be found that in the multimedia scenario which specifies the temporal relationships amongst the data objects constructing the multimedia document, the temporal cooperation requirements of the multimedia servers that are involved in the creation of the document, are as well described. In the scenario of Fig. 7.5, it can be seen that in the creation of the document, the image server should start first. Then the voice server is supposed to join in, delivering the voice data at the same time with the image server, which is delivering the data of the images "*P2*" and "*P3*". Later on, the audio and the text servers are activated simultaneously to start their data delivery tasks for the document creation.

From a given multimedia scenario, a task diagram for the multiple servers can be constructed directly as shown in Fig. 7.6, by introducing a horizontal time axis and a vertical server axis. In the diagram, given a server, we always find that it maps to a set of time intervals on the time axis. These time intervals represent the task execution schedules required by the scenario, for the data delivery in the multimedia data integration. The

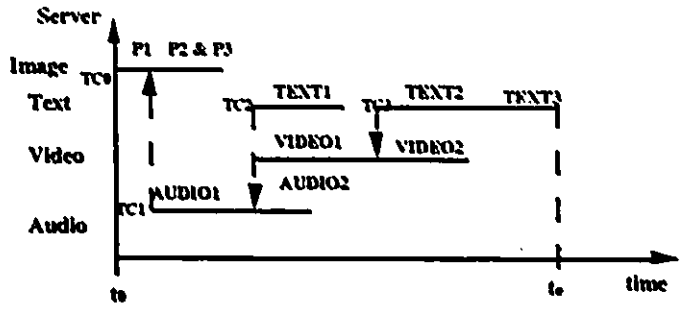


Figure 7.6: The task diagram for the scenario example

servers dealing with different storage devices have different data retrieval speeds. Then, in order to meet the task schedules required by the scenario, they may have to manage their time in order to start processing certain data objects.

The task diagram in Fig. 7.6 is made of several task execution intervals. The starting time of the task diagram t_0 corresponds to the starting time of the data delivery from the server which is supposed to start first in the scenario. Similarly, the end time of the task diagram t_e is the time when all servers finish their data delivery tasks for the scenario. One task execution interval may involve several servers, corresponding to the situation where multiple servers are supposed to deliver their data at the same time to "match" each other. Thus, the temporal cooperation of the servers required in the data integration for a scenario is represented in the corresponding task diagram. It can be seen that, if the scenario comprises many tasks, the servers involved at the beginning of the data integration for the scenario will not have much cooperation with the servers that are involved, a long time after, at the end of the integration. Because of the loose cooperation involved in the servers, a "big" scenario containing many tasks that cost long integration time, can thus be decomposed into "small" atomic scenarios for the purpose of performance modeling. Therefore, it is reasonable to take the data integration for one scenario as an atomic action in a performance model.

In the task diagram, given any time interval (t_0, t_e), there should always be a set of tasks in it, being executed by one or several servers. Empty time intervals are pro-

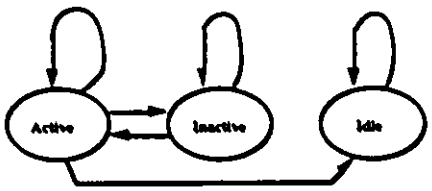


Figure 7.7: State diagram of a server during the data integration of one scenario

hibited. A scenario containing any empty intervals, in which no data is displayed on any media, is called a "interrupted scenario". An interrupted scenario can be accepted by the data integration algorithm developed for the application of the document creation [26], although it may result in a poor system performance due to the inefficiency of the servers utilization. However, in our performance model, an interrupted scenario is required to be decomposed into two scenarios by the system before the data integration. Thus, a task diagram does not contain any interruptions (empty intervals).

Now, we combine the task diagram of the scenario and the states of the servers to investigate the temporal characteristics of the cooperative servers during the data integration.

3. Temporal Characteristics of The Servers

In the task diagram of a scenario, every server that is involved has a required starting time(s) for its data delivery task(s). Some "trivial" task starting times do not need to be specified. For instance, in Fig. 7.6, the starting time of task "$P3$" at the image server is a "trivial" one, because the image server can start to deliver the data of $P3$ continuously after $P2$ is finished, without considering cooperation with other servers. On the other hand, the starting time for a task execution which involves cooperation of the servers, is important for describing the servers' cooperative behaviour.

Definition 7.1 *In a task diagram, a TC (Task oCcurrence) is the specified starting time for one or several tasks. The latter refers to the case in which several servers are required to start their data delivery tasks at the same time.*

In Fig. 7.6, the TCs are illustrated. For instance, at TC_2, three servers, the text server, the video server and the audio server are involved. In the multimedia data integration algorithm [26], at every TC, the involved servers have to synchronize in order to be "matching" with each other in the upcoming task execution interval as required by the scenario. For example, at TC_2, three servers are supposed to start their data delivery tasks at the same time (i.e., at TC_2). It can be observed that in the server-time space of a task diagram, there are no state changes of the servers during the task execution interval between two neighbouring TCs. The state change occurs only at every TC. An idle server remains idle all the time in a task diagram of the data integration for a given scenario. That is, an idle server remains idle at every TC in a task diagram, although it can be active and inactive in another task diagram at the same time. On the other hand, in a task diagram, at a TC, an active server can remain active, switch to the state of inactive or enter into an idle state if it finishes its commitment for this scenario. An inactive server can remain inactive, or become active but not idle. The reason is that if an inactive server quits the integration after an inactive time interval, it should have left before this time interval. The state diagram of one server in the data integration of one scenario is shown in Fig. 7.7. The state transitions may occur at any TC in the task diagram. Apparently, not all the servers can enter into the inactive state at a TC because this generates an interrupted scenario as mentioned earlier. Thus additional conditions are also placed on the state changes of the servers.

The temporal characteristics of all the servers at every TC during the data integration of a scenario is described in a flow graph in Fig. 7.8. Based on this flow graph, a Petri Net model can be established to represent the servers' characteristics at a TC during the data integration for one multimedia scenario. This Petri Net model, which is named as a TC switch is illustrated in Fig. 7.9, with three servers assumed. The legend is given in Fig. 7.10

As shown in Fig. 7.9, the function of block "A" in the flow graph of Fig. 7.8 is initiated by transition $t16$. When place $P24$ obtains a token, the active servers will make their choices to remain in or quit the integration for the scenario. Transitions $t26$, $t27$,

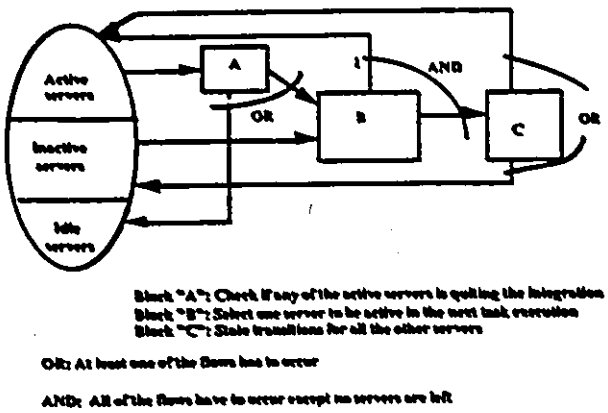


Figure 7.8: Flow graph for all the servers at a TC

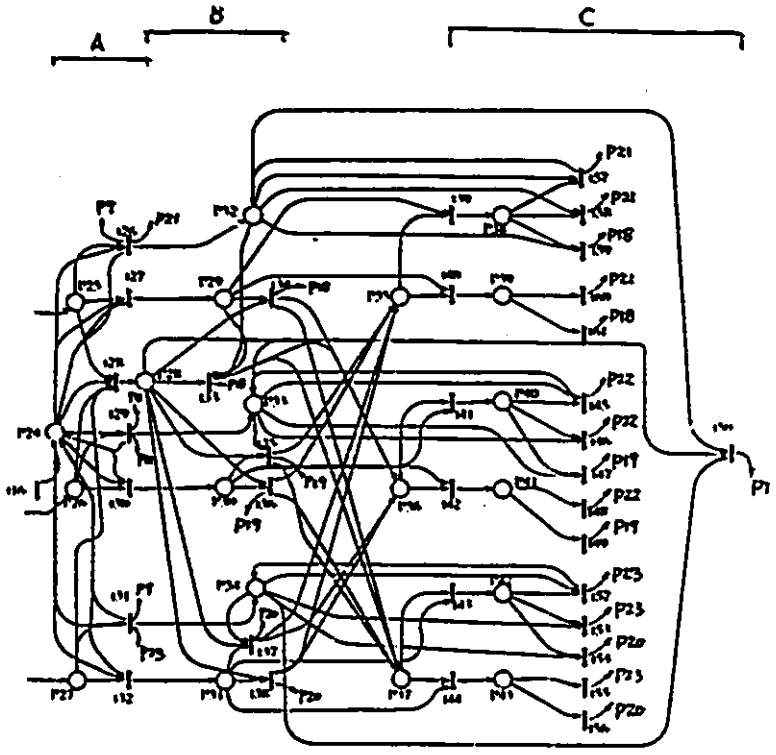


Figure 7.9: The GSPN model for TC

P24	a task execution is finished
P25	server 1 was active in last task execution
P26	server 2 was active in last task execution
P27	server 3 was active in last task execution
P28	servers enter the "block B"
P29	active server 1 will still remain in the integration of the next task execution
P30	active server 2 will still remain in the integration of the next task execution
P31	active server 3 will still remain in the integration of the next task execution
P32	server 1 is idle in the integration of the scenario
P33	server 2 is idle in the integration of the scenario
P34	server 3 is idle in the integration of the scenario
P35	place for server 1
P36	place for server 2
P37	place for servre 3
P38	server 1 which was inactive in the last task execution or idle for the scenario
P39	server 1 which was active in the last task execution
P40	server 2 which was inactive in the last task execution or idle for the scenario
P41	server 2 which was active in the last task execution
P42	server 3 which was inactive in the last task execution or idle for the scenario
P43	server 3 which was active in the last task execution
t16	the previous task execution is finished (start a TC Switch)
t26	the active server 1 finishes at this TC and quits the integration
t27	the active server 1 will remain in the data integration of the scenario
t28	finish checking the active servers which are leaving the integration
t29	the active server 2 finishes at this TC and quits the integration
t30	the active server 2 will remain in the data integration of the scenario
t31	the active server 3 finishes at this TC and quits the integration
t32	the active server 3 will remain in the data integration of the scenario
t33	inactive server 1 is selected to be active in the next task execution
t34	active server 1 is selected to be active in the next task execution
t35	inactive server 2 is selected to be active in the next task execution
t36	active server 2 is selected to be active in the next task execution
t37	inactive server 3 is selected to be active in the next task execution
t38	active server 3 is selected to be active in the next task execution
t39	server 1 was inactive in last task execution or idle for the scenario
t40	server 1 was active in last task execution
t41	server 2 was inactive in last task execution or idle for the scenario
t42	server 2 was active in last task execution
t43	server 3 was inactive in last task execution or idle for the scenario
t44	server 3 was active in last task execution
t45, t52, t57	idle server went back to idle
t46, t53, t58	inactive server remains inactive in the coming task execution interval
t47, t54, t59	inactive server turns into active in the coming task execution interval
t48, t55, t60	active server turns into inactive in the coming task execution interval
t49, t56, t61	active server remains active in the coming task execution interval
t50	the integration is finished for the scenario

Figure 7.10: The legend for the TC switch

$t29$, $t30$, $t31$ and $t32$ belong to the function block "A". The block "A" finishes by firing transition $t28$ to enter into the block "B". In block "B", from all the non-idle servers, one is selected to be active in the upcoming task execution interval. Then, all the remaining servers (including the idle ones) will enter the stage of block "C" by getting into places $P35$, $P36$ and $P37$, for server 1, server 2 and server 3, respectively. The idle servers are also included here for constructing a homogeneous net. Thus, the complexity of the net can be reduced, because the net is not distinguished by the different kinds of scenarios which ask for different servers. In function block "C", idle servers remain idle by firing transitions $t57$, $t45$, $t52$, with respect to the three servers. Inactive servers will select $t58$ or $t59$ ($t46$ and $t47$, or $t53$ and $t54$) to remain inactive or to switch to the active state. On the other hand, the active servers will choose to be inactive in the coming task interval by firing $t60$ ($t48$, $t55$), or remain active by firing $t61$ ($t49$, $t56$). The selections in the model are made with the probabilities assigned to these immediate transitions. Thus, the characteristics of the states of the servers in the data integration is modeled by this big random switch of TC. The temporal relationships and cooperation of the servers during the data integration are described by the task diagram and the TC switch.

Based on this TC switch, we apply the Coloured Generalized Stochastic Petri Net (CGSPN) models for the performance analysis of the distributed multimedia data integration through the cooperative servers.

7.2.2 Performance Modeling of Distributed Multimedia Data Integration

Below, we apply the CGSPN model to the distributed multimedia data integration. The CGSPN model will be solved by unfolding it into the uncoloured GSPN model. In our net, the colour is used to distinguish different jobs, i.e., the concurrent data integrations for the different scenarios.

1. CGSPN Models for Distributed Data Integration

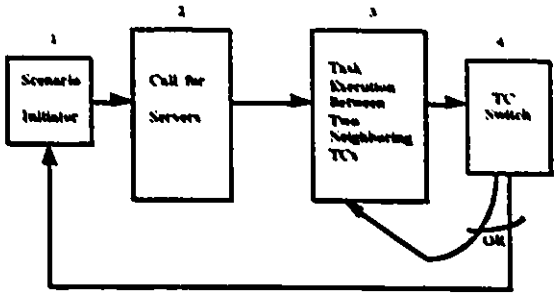
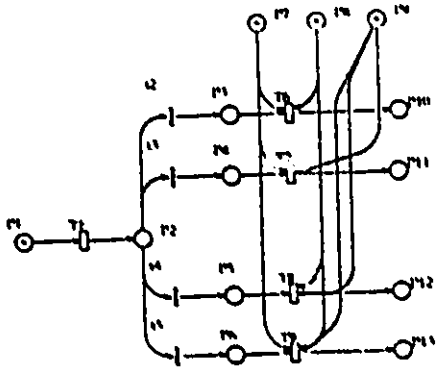


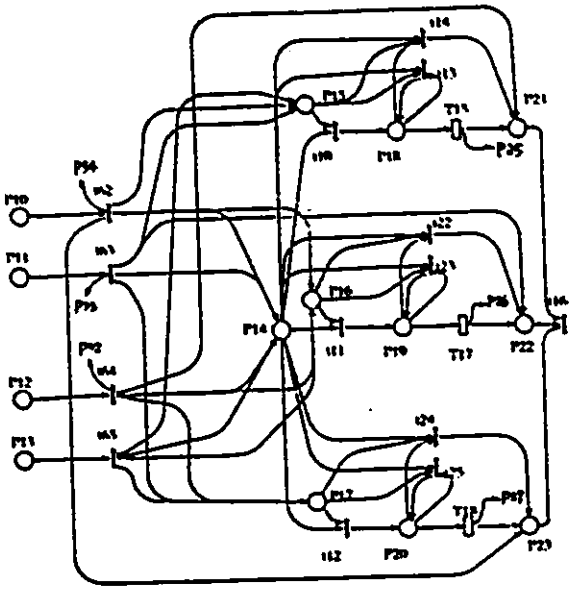
Figure 7.11: Module diagram of the GSPN model with one job token

In order to build the CGSPN model for the system, we first consider the simplest case. For the time being, only one job token is assumed in the system. That is, at any given time, only one data integration job for a scenario is allowed. With one job token, only the GSPN model is needed to represent the data integration. A diagram of the modules that are included in the GSPN model is shown in Fig. 7.11. There are three kinds of multimedia servers, server1, server2 and server3 as assumed above, e.g. an audio server, a video server and a text and graphics server. The GSPN models corresponding to every module, (except the TC switch), are shown in Fig. 7.12. The number of job tokens in place $P1$ corresponds to the number of scenarios that are allowed to be integrated concurrently in the system. The legends for the two subnets are shown in Fig. 7.13.

The subnet 1 in Fig. 7.12(a) includes the modules 1 and 2 in Fig. 7.11. The number of tokens in places $P7$, $P8$ and $P9$ represents the number of available servers, i.e., *server*1, *server*2 and *server*3. Four types of scenarios, $SCE1$, $SCE2$, $SCE3$ and $SCE4$ are considered to be generated in the system. The first type of scenario $SCE1$ requires *server*1 and *server*2, i.e., it involves the media of audio and video. The second type $SCE2$ requires *server*1 and *server*3. *Server*2 and *server*3 are involved in the integration for $SCE3$. The last kind of the integration ($SCE4$) needs all the three servers. The initiation of the scenarios of the above four types corresponds to the transitions $t2$, $t3$, $t4$ and $t5$ in Fig. 7.12(a), respectively. Then transitions $T6$, $T7$, $T8$ and $T9$ collect the required servers for the different integrations. (Places $P10$, $P11$, $P12$, $P13$ are introduced for solving the resulting CGSPN by unfolding it, when more than one job token is concerned).



(a) Sub-net 1 — Scenario initiator & calling for the servers



(b) Subnet 2 — Task execution of servers between two neighboring TCs

*P25, P26, P27, P32, P33, P34 are places in the TC Switch

Figure 7.12: Subnets in the GSPN model

P1	Job tokens
P2	scenario initiated
P3	scenario requires server 1 & 2 (*SCE1*)
P4	scenario requires server 1 & 3 (*SCE2*)
P5	scenario requires server 2 & 3 (*SCE3*)
P6	scenario requires server 1 & 2 & 3 (*SCE4*)
P7	server 1 (idle)
P8	server 2 (idle)
P9	server 3 (idle)
P10	server 1 & 2 ready for the integration of *SCE1*
P11	server 1 & 3 ready for the integration of *SCE2*
P12	server 2 & 3 ready for the integration of *SCE3*
P13	server 1 & 2 & 3 ready for integration of *SCE4*
t1	initiating scenario
t2	*SCE1* is initiated
t3	*SCE2* is initiated
t4	*SCE3* is initiated
t5	*SCE4* is initiated
T6	call for server 1 & 2
T7	call for server 1 & 3
T8	call for server 2 & 3
T9	call for server 1 & 2 & 3

(a) Legend of subnet 1

P14	start to deliver the data for the scenario
P15	server 1 is to start
P16	server 2 is to start
P17	server 3 is to start
P18	server 1 is active in the current task execution interval
P19	server 2 is active in the current task execution interval
P20	server 3 is active in the current task execution interval
P21	server 1 finishes a task execution or is inactive in this task interval
P22	server 2 finishes a task execution or is inactive in this task interval
P23	server 3 finishes a task execution or is inactive in this task interval
t62	integration job for *SCE1* is initiated
t63	integration job for *SCE2* is initiated
t64	integration job for *SCE3* is initiated
t65	integration job for *SCE4* is initiated
t10, t13	server 1 is selected to be active in the first task execution interval
t11, t23	server 2 is selected to be active in the first task execution interval
t12, t25	server 3 is selected to be active in the first task execution interval
t14	server 1 is inactive in the first task execution interval
T15	task execution of server 1
t16	all the servers finish the task execution and are synchronized to start the next task
T17	task execution of server 2
T18	task execution of server 3
t22	server 2 is inactive in the first task execution interval
t24	server 3 is inactive in the first task execution interval

(b) Legend of subnet 2

Figure 7.13: The legends of the subnets

The subnet 2 in Fig. 7.12(b) represents the task execution of the servers between two neighbouring *TCs*. Transitions $t62$, $t63$, $t64$, $t65$ set the idle and non-idle states for each server in the four types of integrations, respectively. The places $P32$, $P33$ and $P34$, which can be found in the *TC* switch of Fig. 7.9, mark the idle state of server1, server2 and server3 in the data integration for a scenario, respectively. Non-idle servers are represented by tokens in place $P15$, $P16$ and $P17$, with respect to *server*1, *server*2 and *server*3. Then the transitions $t10$, $t11$ and $t12$ will select one non-idle server to be active in the first task execution interval of the data integration for the scenario. If the server, say, *server*1 is not selected by $t10$, it can still be selected by $t13$ to become active in the first task execution interval. The reason for this is that, during a task execution interval, at least one server has to be active. However, there can be more than one active server. Transition $t14$ allows *server*1 to choose to be inactive during the first task interval. Transitions $t23$ and $t22$ ($t25$ and $t24$) offer the same choices of the active and inactive states to the non-idle *server*2 (*server*3). Transitions $T15$, $T17$ and $T18$ stand for the task execution between two neighbouring *TCs*. Finally at $t16$, the servers are supposed to recover the incurred random synchronization loss. Thus, the servers are resynchronized again to start the next task.

When the number of concurrently executed integration jobs is increased, the Coloured GSPN (CGSPN) is applied. Every job token in place $P1$ then takes its own colour. The scenario initiated, i.e., the tokens generated by the job token in $P1$ for the execution of data integration in the net, will adopt the same colour as that of their generator, i.e., the job token. Thus, the module of "Task Execution" in Fig. 7.11, which will be "multi-coloured in the CGSPN, represents the concurrent data integration of the multiple scenarios. In the resulting CGSPN model, only when the tokens in $P21$, $P22$ and $P23$ have the same colour, the transition $t16$ is enabled.

In order to solve the CGSPN model, the coloured net is unfolded into an uncoloured one. Then the resulting GSPN is solved by using a graphically interfaced GSPN solution package, GreatSPN1.5. In Fig. 7.14, a module diagram of the unfolded net is shown. In the uncoloured net, all the scenarios can share one *TC* switch to reduce the complexity

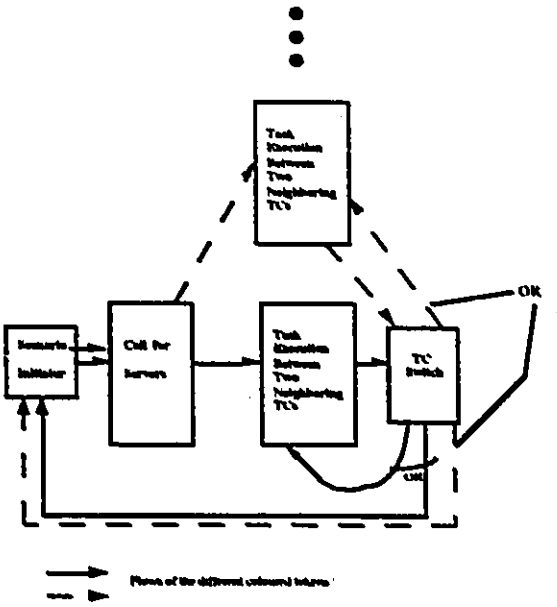


Figure 7.14: Module diagram of the unfolded CGSPN model with concurrent integration jobs

of the net model. The reason is that the TC switch is comprised of only immediate transitions. It is known that in GSPN, a random switch (immediate transitions) has no firing time [57]. Therefore, the servers which are involved in the integration of other scenarios cannot get into the TC switch, before the servers involved in the integration of one scenario leave the TC switch. This is because their "Task Execution" module cannot be finished (exponential transitions) during this zero time duration. Thus, tokens that belong to the different jobs will not be mixed in the TC switch.

2. Performance Results

The assignment of the firing rates for the timed transitions as well as the probability assignments for the random switches are given in Fig. 7.15. In this assignment, it can be interpreted that the number of the first two types of scenarios $SCE1$ and $SCE2$ created in the system is twice as that of $SCE3$ and $SCE4$. Since both $SCE1$ and $SCE2$ require the *server*1, *server*1 is relatively in high demand in the data integration. On the other

Transitions	Rate
T1	0.005
t2, t3	0.33
t4, t5	0.17
T6	0.5
T7	0.3
T8	0.4
T9	0.1
t10	0.33
t11	0.34
t12	0.33
t13, t59, t61	0.8
t14, t58, t60	0.2
T15, T16, T17	0.0167
t22, t46, t48	0.7
t23, t47, t49	0.3
t24, t53, t55	0.4
t25, t54, t56	0.6
t26	0.2
t27	0.8
t29	0.25
t30	0.75
t31	0.33
t32	0.67
t16, t28, t33, t34, t35, t36, t37, t38, t39, t40, t41, t42, t43, t44, t45, t52, t57, t62, t63, t64, t65	1

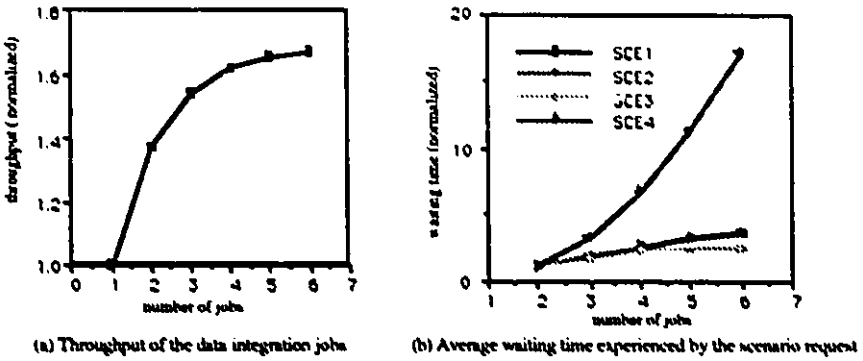
Figure 7.15: Assignment of the firing rates

hand, the firing rates assigned to transitions $t13$ and $t14$ represent the ratio of the active and inactive intervals for *server*1 during the data integration for the scenarios it involved. The rates assigned to the *server*2 (transitions $t23$ and $t22$) and *server*3 ($t25, t24$) have similar meaning.

Now, we discuss the relevant performance measures of the data integration system.

1. The throughput of the system

Firstly, only one server is assumed for each of the three kinds of servers. That is, the places $P7$, $P8$ and $P9$ contain one token each. When the number of job tokens is increased in $P1$, contention for the servers will occur. The throughput of the multimedia data integration for the scenario, with respect to different number of job tokens, is illustrated in Fig. 7.16(a). The throughput is normalized to the throughput of the system which has only one job token. Meanwhile, the average time that the job has to spend at the transitions $T6$, $T7$, $T8$ and $T9$ is shown in Fig. 7.16(b). This time represents the waiting time the job experiences, in order to wait for the required servers to be ready for the



(a) Throughput of the data integration jobs (b) Average waiting time experienced by the scenario request

Figure 7.16: The performance results (1)

data integration. The average waiting time at a transition can be calculated by applying Little's theorem, when the throughput of the transition (i.e. the mean number of firings per unit time) and the average population of its input places are obtained, by solving the underlying EMC (Embedded Markov Chain) of the GSPN model. The average waiting time presented in the Fig. 7.16(b) is normalized to the waiting times faced by the jobs in the system which has two job tokens in $P1$. This is because no waiting is caused by the contention for servers in the system which involves only one job. It can be seen, from the Fig. 7.16, that the throughput of the data integration is increased with the increase in the number of jobs allowed in the system. However, when the number of job tokens is more than 5, the improvement in the throughput is approaching its limit. At the same time, the waiting times are also increased. Especially, the waiting time of the data integration job which requires more servers will experience longer waiting time than the jobs that require few servers. Thus the type of jobs which requires more servers will fail to meet the real-time requirement imposed by the application first. Meanwhile, the jobs that involve the relatively highly demanded server (e.g., server1) also have to wait longer than the others.

In Fig. 7.17, the throughputs of two different systems are compared. In both of the two cases, the average time required by the data integration for one scenario is less than that of the original system shown in Fig. 7.16. In case 1, the average time interval of

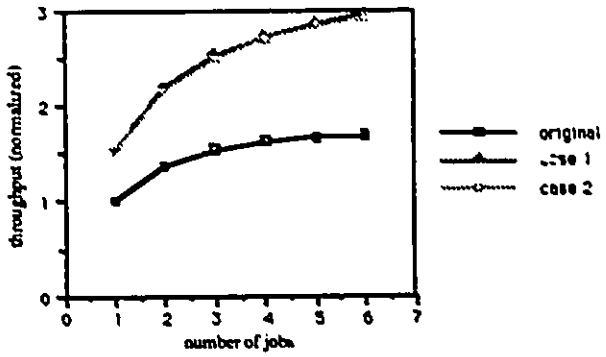


Figure 7.17: The performance results (2)

the task execution between two neighbouring TC's in the task diagram is reduced to half as that of the original system. In case 2, the task execution interval remains the same while the average number of the task executed in the data integration for each scenario is reduced to half (i.e., fewer TCs in every task diagram). Apparently, the throughputs in both of the two cases are improved, especially when more clients (job tokens) are accommodated. The throughput of these two cases will approach the maximum value after the number of job tokens exceeds 6. The system in case 1 only slightly out-performs the system in case 2. When the number of clients is increased, the system in case 1 tends to out-perform the system in case 2.

2. The effect of the different number of servers

When two job tokens are placed in $P1$, the probability that a job has to wait for the servers is investigated, with respect to different number of the servers.

Take $\#\{Pi\}$ as the population in place Pi. Then, the probabilities of jobs waiting for servers at place $P3$, $P4$,$P5$ and $P6$ are defined as the following.

$$Prob.3 = prob\{\#\{P3\} > 0 \ \& \ NOT(\#\{P7\} > 0 \ \& \ \#\{P8\} > 0)\}$$

$$Prob.4 = prob\{\#\{P4\} > 0 \ \& \ NOT(\#\{P7\} > 0 \ \& \ \#\{P9\} > 0)\}$$

$$Prob.5 = prob\{\#\{P5\} > 0 \ \& \ NOT(\#\{P8\} > 0 \ \& \ \#\{P9\} > 0)\}$$

$$Prob.6 = prob\{\#\{P4\} > 0 \ \& \ NOT(\#\{P7\} > 0 \ \& \ \#\{P8\} > 0 \ \& \ \#\{P9\} > 0)\}$$

Number of server processes S1	S2	S3	Throughput	Probability that a job has to wait for servers SCH1	SCH2	SCH3	SCH4
1	1	1	0.002045	0.184	0.180	0.072	0.110
2	1	1	0.002331	0.101	0.099	0.071	0.074
1	2	1	0.002197	0.150	0.100	0.031	0.088
1	1	2	0.002119	0.185	0.131	0.052	0.101
2	2	1	0.002568	0.000	0.000	0.031	0.012
2	1	2	0.002451	0.104	0.000	0.054	0.056
1	2	2	0.002289	0.132	0.133	0.000	0.071

Figure 7.18: Performance measures of the two job token system

In this case, a CGSPN model is unfolded into the GSPN model for the solution. Two subnets of the module of "task execution between the two neighbouring TCs" are generated in the GSPN model by the unfolding. This GSPN model is then solved. In Fig. 7.18, the waiting probabilities are illustrated, together with the throughputs. The performance reference for adding more servers into the system can be directly obtained from Fig. 7.18. For instance, it is observed that if only one server can be added into the system, then an additional *server*1 can maximally increase the throughput of the system. Meanwhile, the smallest waiting probability can be achieved for the data integration of $SCE1$, $SCE2$ and $SCE4$, except that of $SCE3$.

7.3 Summary

In this chapter, Stochastic Petri Nets (SPN) models have been applied for the performance modeling and evaluation of the multimedia synchronizations. Both the GSPN and the CGSPN models are employed.

We have established the SPN models for the distributed multimedia data integration system, as well as the simultaneous real-time data delivery scheme. The approaches we have adopted give the example of performance modeling for the multimedia application systems where the synchronization is involved. In our studies, relevant performance measures of the systems have been obtained. The performance indice can be used as the reference in the system implementation. To our knowledge, no such work had been

reported before. **Our above contributions were published in [23] [27].**

Chapter 8

A Computational Algorithm for LBSPN & Its Application in Performance Modeling of Multimedia Collaborations

In the previous chapter, GSPN and CGSPN models were applied for the performance modeling and evaluation of the multimedia synchronization systems. We have also obtained performance measures of interest for the system modeled. However, both the GSPN and CGSPN models face an exponentially increased state space. As we have addressed in Chapter 6, efficient solution methods for Stochastic Petri Nets (SPN) are required, in order for the performance analysis of multimedia systems.

In our studies, we have investigated a class of SPN models which has product form solutions [51]. That is the Locally Balanced Stochastic Petri Nets (LBSPN) models. In this chapter, we will present a computational algorithm developed for the LBSPN, which is based on the convolution of the subnets in a LBSPN model. Then, using the computational algorithm, LBSPN is employed to model the multimedia collaborative application of joint editing.

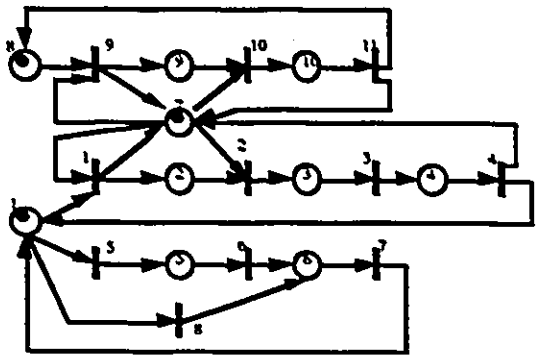
8.1 Locally Balanced Stochastic Petri Net And Product Form Solutions

Locally Balanced Stochastic Petri Nets (LBSPN) were first introduced by Lazar and Robertazzi presented [65] [66]. M. Li and Georganas have further extended the boundary of LBSPNs, proven that the LBSPNs falling into the new boundary also have product form solutions [49] [50] [51]. A systematic way for identifying the LBSPN that fall into the new boundary has also been designed [51]. However, in the work of Li et al. [48] [49] [51], the computational method for the normalization constant was not investigated. In order to obtain the normalization constant, the sum of the probability distributions of all the steady states of the entire Markov Chain (i.e., the markings of the entire net) is required. Since the associated Markov Chain for a given LBSPN may very likely have a large state space, calculating the normalization constant G directly from the distributions of all the steady states can be a tedious job. A computational algorithm is required to evaluate the normalization constant G when the LBSPNs are applied in performance modeling of any real systems.

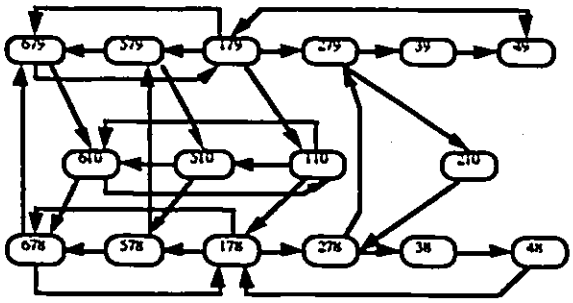
In the following, we will first represent the definition of the LBSPN, because the definition of the subnet in LBSPN given in [51] was described very informally. Then, the computational algorithm of LBSPN will be developed.

8.1.1 Subnets & LBSPNs

Subnet is the basic unit to constitute a LBSPN. Given a SPN N_1, assume Ψ is the set of all the T-invariants of N_1. For every $\psi_i \in \Psi$, $\|\psi_i\|$ is its invariant support [63], i.e., the set of transitions corresponding to nonzero entries in the T-invariant $\psi_i \geq 0$. If, $\exists \psi_i, \psi_j \in \Psi$, and $\psi_i \cap \psi_j \neq \phi$, then we say that ψ_i and ψ_j are in one group. One group of transitions and their input/output places constitute a *subnet*. Thus, the subnets have no common transitions, although they may have common places. For instance, in Fig. 8.1a), we have $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4\}$, where $\psi_1 = \{t_9, t_{10}, t_{11}\}$, $\psi_2 = \{t_1, t_2, t_3, t_4\}$,



a) The LBSPN



b) The Markov Chain

Figure 8.1: An example of LBSPN and its Markov chain

$\psi_3 = \{t_5, t_6, t_7\}$ and $\psi_4 = \{t_7, t_8\}$. Since $\psi_3 \cap \psi_4 = \{t_7\} \neq \phi$, three subnets are obtained. They are $S_1 = \{t_9, t_{10}, t_{11}, P_7, P_8, P_9, P_{10}\}$, $S_2 = \{t_1, t_2, t_3, t_4, P_1, P_2, P_3, P_4, P_7\}$ and $S_3 = \{t_5, t_6, t_7, t_8, P_1, P_5, P_6\}$.

After all the subnets are identified, a test procedure can be employed to check if the SPN model which consists of all these subnets, is a Locally Balanced Stochastic Petri Nets (LBSPN) model. The test procedure was stated and proven in [51]. In this chapter, we only consider the models which have passed the test procedure and are thus LBSPNs.

8.1.2 Product Form Solutions

The marking of the LBSPN is denoted as μ_i. μ_0 is the initial marking of the net. Fig. 8.1b) shows the underlying Markov Chain, as well as the markings of the LBSPN in Fig. 8.1a). We need the following definitions [51] of the other kinds of markings concerned with the subnets.

1. Basic Markov Chain. A Basic Markov Chain is the Markov Chain associated with a subnet.

2. Basic Subnet Marking. A Basic Subnet Marking is a marking in the Basic Markov Chain. We use $\hat{\mu}_i^j$ to present a Basic Subnet j Marking and $\hat{\mu}_0^j$ to present the initial marking for subnet j. In Fig. 8.2a) to c), the subnets of the net in Fig. 8.1 are illustrated with their Basic Markov Chains. The Basic Subnet Markings are also presented.

3. Separate Subnet Marking. For a given marking μ_i, there is a Separate Subnet j Marking, written as μ_i^j, which is a marking consists of the places of only subnet j and the number of tokens in the place is equal to that of μ_i. When there is no token in subnet j, $\mu_i^j = 0$.

For example, given marking $\mu_i = \{6(1), 7(1), 9(1)\}$ in Fig. 8.1b), where the places P_6, P_7 and P_9 contains one token each, the corresponding Separate Subnet Markings are $\mu_i^1 = \{7(1), 9(1)\}$, $\mu_i^2 = 0$ and $\mu_i^3 = \{6(1)\}$. Note that μ_i^j and $\hat{\mu}_i^j$ may or may not be the same. In the example, we have $\mu_i^1 = \hat{\mu}_1^1 = \{7(1), 9(1)\}$ and $\mu_i^3 = \hat{\mu}_2^3 = \{6(1)\}$. However, $\mu_i^2 = 0$ does not equal any of the Basic Subnet Markings of the subnet 2.

An underline below a marking is used to indicate the same marking with the number

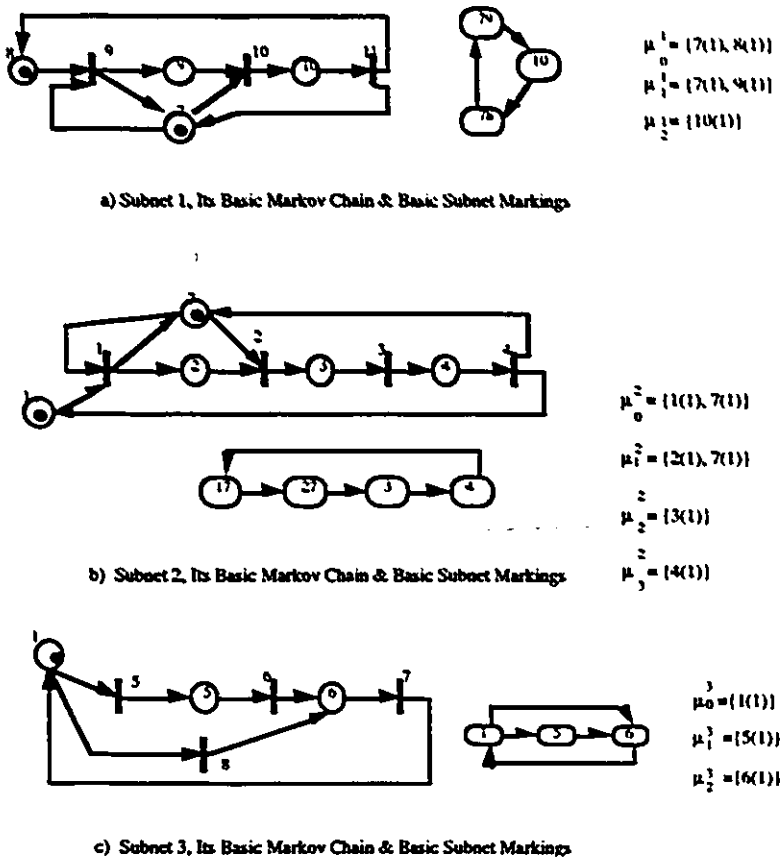


Figure 8.2: Subnets of the LBSPN example in the previous figure

of tokens in the common places being set to zero. For a Basic Markov Chain of subnet j, if $\exists \underline{\hat{\mu}_i^j} = \underline{\hat{\mu}_l^j}$, $l \neq i$, we say that $\hat{\mu}_i^j$ is not unique, i.e., there are at least two markings, where the distribution of tokens in the non-common places is the same. The difference is only the token distributions in the common places.

In order to present the product form solution in a concise way, a function $\rho()$ was defined in [51], mapping the marking μ_i (μ_i^j for subnet j) into the Basic Subnet Markings. That is, given a marking μ_i, $\rho(\mu_i^j)$ is obtained as the Basic Subnet Marking $\hat{\mu}_l^j$ of subnet j, when μ_i is mapped into the subnet j. For instance, given $\mu_i = \{6(1), 7(1), 9(1)\}$, then we have $\rho(\mu_i^1) = \{7(1), 9(1)\} = \hat{\mu}_1^1$, $\rho(\mu_i^2) = \{1(1), 7(1)\} = \hat{\mu}_0^2$ and $\rho(\mu_i^3) = \{6(1)\} = \hat{\mu}_2^3$. There is a set of rules defined [51] for this mapping function $\rho()$. The rules will not be repeated here. Basically, we have $\underline{\rho(\mu_i^j)} = \underline{\hat{\mu}_l^j}$. When $\hat{\mu}_l^j$ is not unique, the number of tokens in the common place is determined by the general rule. That is, the number of resource tokens absorbed by subnets plus the resource tokens left should be equal to the total number of the initial tokens in that place. In the case of $\underline{\mu_i^j} = 0$, we have $\rho(\mu_i^j) = \hat{\mu}_0^j$ [51].

Now, the product form solution for LBSPN, which consists of N subnets, is presented as:

$$P(\mu_i) = \frac{1}{G} \prod_{j=1}^{N} P_j(\rho(\mu_i^j)) \text{ where } G \text{ is a normalization constant:}$$

$$G = \sum_{u_i} \prod_{j=1}^{N} P_j(\rho(\mu_i^j)) \tag{8.1}$$

$P_j(\rho(\mu_i^j))$ is the probability distribution obtained from the Basic Markov Chain of subnet j.

8.2 The Computational Algorithm for LBSPN

8.2.1 Convolution of The Subnets

As mentioned in the above, the probability distributions of all the steady states, i.e., of all the markings of the LBSPN have to be obtained, if the normalization constant G is

calculated directly from the Formula 8.1. Since the function $\rho()$ has various mapping rules in different cases [51], mapping every μ_i into each subnet may not be tractable in the case of a "big" LBSPN model. In this section, we are going to approach the normalization constant G from the separated subnets. A decomposition procedure will be given to reduce a LBSPN, which allows calculating the G at the subnet level. That is, the G can be obtained through the convolution of the subnets.

1. Reducing LBSPN by Subnet

Given a LBSPN that has N subnets, we assume that $\hat{\mu}_0^N$, $\hat{\mu}_1^N$, ..., $\hat{\mu}_{K1}^N$ are the $K1$ markings of the subnet N, i.e., $\rho(\mu_i^N) \in \{\hat{\mu}_0^N, \ldots, \hat{\mu}_{K1}^N\}$.

$$
\begin{aligned}
G &= \sum_{u_i} \prod_{n=1}^{N} P_n(\rho(\mu_i^n)) = \sum_{\rho(\mu_i^N)=\hat{\mu}_0^N} \prod_{n=1}^{N} P_n(\rho(\mu_i^n)) + \sum_{\rho(\mu_i^N)\neq\hat{\mu}_0^N} \prod_{n=1}^{N} P_n(\rho(\mu_i^n)) \\
&= P_N(\hat{\mu}_0^N) \sum_{\rho(\mu_i^N)=\hat{\mu}_0^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)) + \sum_{\rho(\mu_i^N)=\hat{\mu}_1^N} \prod_{n=1}^{N} P_n(\rho(\mu_i^n)) \\
&\quad + \sum_{\rho(\mu_i^N)\neq\hat{\mu}_0^N, \rho(\mu_i^N)\neq\hat{\mu}_1^N} \prod_{n=1}^{N} P_n(\rho(\mu_i^n)) \\
&= P_N(\hat{\mu}_0^N) \sum_{\rho(\mu_i^N)=\hat{\mu}_0^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)) + P_N(\hat{\mu}_1^N) \sum_{\rho(\mu_i^N)=\hat{\mu}_1^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)) \\
&\quad + P_N(\hat{\mu}_2^N) \sum_{\rho(\mu_i^N)=\hat{\mu}_2^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)) + \ldots + P_N(\hat{\mu}_{K1}^N) \sum_{\rho(\mu_i^N)=\hat{\mu}_{K1}^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)) \\
&= P_N(\hat{\mu}_0^N) G_d^1(\hat{\mu}_0^N, N-1) + P_N(\hat{\mu}_1^N) G_d^1(\hat{\mu}_i^N, N-1) + P_N(\hat{\mu}_2^N) G_d^1(\hat{\mu}_2^N, N-1) \\
&\quad + \ldots + P_N(\hat{\mu}_{K1}^N) G_d^1(\hat{\mu}_{K1}^N, N-1) \\
&\quad \text{where } G_d^1(\hat{\mu}_j^N, N-1) = \sum_{\rho(\mu_i^N)=\hat{\mu}_j^N} \prod_{n=1}^{N-1} P_n(\rho(\mu_i^n)), \hat{\mu}_j^N \in \{\hat{\mu}_0^N, \ldots, \hat{\mu}_{K1}^N\} \qquad (8.2)
\end{aligned}
$$

Thus, G is decomposed into disjoint sets according to different Basic Subnet Markings of subnet N. It can be envisioned here that an algorithm might be formed to calculate G by decomposing the net according to the markings of subnet N. That is, the subnet N is separated from the other $N-1$ subnets and the LBSPN is thus reduced. Then instead

of G, the $G_d^1(\hat{\mu}_j^N, N-1)$s will be obtained first, provided that there is a method for us to calculate $G_d^1(\hat{\mu}_j^N, N-1)$ easily. Further, if we assume that $\hat{\mu}_0^{N-1}$, $\hat{\mu}_1^{N-1}$, ..., $\hat{\mu}_{K2}^{N-1}$ are the $K2$ markings of the subnet $N-1$, we can get, from Formula 8.2:

$$
\begin{aligned}
G &= P_N(\hat{\mu}_0^N)G_d^1(\hat{\mu}_0^N, N-1) + P_N(\hat{\mu}_1^N)G_d^1(\hat{\mu}_1^N, N-1) + P_N(\hat{\mu}_2^N)G_d^1(\hat{\mu}_2^N, N-1) \\
&\quad + \cdots + P_N(\hat{\mu}_{K1}^N)G_d^1(\hat{\mu}_{K1}^N, N-1) \\
&= P_N(\hat{\mu}_0^N)(P_{N-1}(\hat{\mu}_0^{N-1})G_d^2(\hat{\mu}_0^N\hat{\mu}_0^{N-1}, N-2) \\
&\qquad + P_{N-1}(\hat{\mu}_1^{N-1})G_d^2(\hat{\mu}_0^N\hat{\mu}_1^{N-1}, N-2) \\
&\qquad + \cdots + P_{N-1}(\hat{\mu}_{K2}^{N-1})G_d^2(\hat{\mu}_0^N\hat{\mu}_{K2}^{N-1}, N-2)) \\
&\quad + P_N(\hat{\mu}_1^N)(P_{N-1}(\hat{\mu}_0^{N-1})G_d^2(\hat{\mu}_1^N\hat{\mu}_0^{N-1}, N-2) \\
&\qquad + P_{N-1}(\hat{\mu}_1^{N-1})G_d^2(\hat{\mu}_1^N\hat{\mu}_1^{N-1}, N-2) \\
&\qquad + \cdots + P_{N-1}(\hat{\mu}_{K2}^{N-1})G_d^2(\hat{\mu}_1^N\hat{\mu}_{K2}^{N-1}, N-2)) \\
&\quad + \cdots \\
&\quad + P_N(\hat{\mu}_{K1}^N)(P_{N-1}(\hat{\mu}_0^{N-1})G_d^2(\hat{\mu}_{K1}^N\hat{\mu}_0^{N-1}, N-2) \\
&\qquad + P_{N-1}(\hat{\mu}_1^{N-1})G_d^2(\hat{\mu}_{K1}^N\hat{\mu}_1^{N-1}, N-2) \\
&\qquad + \cdots + P_{N-1}(\hat{\mu}_{K2}^{N-1})G_d^2(\hat{\mu}_{K1}^N\hat{\mu}_{K2}^{N-1}, N-2))
\end{aligned}
$$

where $G_d^2(\hat{\mu}_j^N\hat{\mu}_l^{N-1}, N-2) = \sum_{\rho(\mu_i^N)=\hat{\mu}_j^N \rho(\mu_i^N)=\hat{\mu}_j^N} \prod_{n=1}^{N-2} P_n(\rho(\mu_i^n))$

$G_d^2(\hat{\mu}_j^N\hat{\mu}_l^{N-1}, N-2)$ only includes the marking μ_i's that can give, in subnet N and $N-1$, the Basic Subnet Markings $\hat{\mu}_j^N$ and $\hat{\mu}_l^{N-1}$, if such μ_i exists. Meanwhile, only $N-2$ subnets are involved here in the mapping function $\rho()$.

Since we have

$$
\begin{aligned}
G_d^1(\hat{\mu}_j^N, N-1) &= P_{N-1}(\hat{\mu}_0^{N-1})G_d^2(\hat{\mu}_j^N\hat{\mu}_0^{N-1}, N-2) + P_{N-1}(\hat{\mu}_1^{N-1})G_d^2(\hat{\mu}_j^N\hat{\mu}_1^{N-1}, N-2) \\
&\quad + \cdots + P_{N-1}(\hat{\mu}_{K2}^{N-1})G_d^2(\hat{\mu}_j^N\hat{\mu}_{K2}^{N-1}, N-2)
\end{aligned}
$$

$G_d(\hat{\mu}_j^N, N-1)$ can thus be calculated again by reducing the system. That is, the $G_d^2(\hat{\mu}_j^N\hat{\mu}_l^{N-1}, N-2)$ will be obtained first, in respect of different Basic Subnet Markings

of $\hat{\mu}_l^{N-1}$. Then $G_d^2(\hat{\mu}_j^N \hat{\mu}_l^{N-1}, N-2)$ will be multiplied with the probabilities of the corresponding Basic Subnet Markings $\hat{\mu}_l^{N-1}$ and summed up to form the $G_d^1(\hat{\mu}_j^N, N-1)$. This is in fact the procedure in which the subnet $N-1$ is also separated from the other subnets. In the calculation of G (or G_d^k), a removed (separated) subnet is not involved in the function $\rho()$ any more. The removed subnet is replaced by the steady state distributions of its Basic Subnet Markings associated with the corresponding G_d^{k+1} of the resulting reduced system. For example, when subnet $N-1$ is removed, $P_{N-1}(\hat{\mu}_i^{N-1}) G_d^2(\hat{\mu}_j^N \hat{\mu}_i^{N-1}, N-2)$ is contained in $G_d^1(\hat{\mu}_j^N, N-1)$. In general, assuming that $\hat{\mu}_0^{N-k}$, $\hat{\mu}_1^{N-k}$, ... $\hat{\mu}_K^{N-k}$ are the K markings of the subnet $N-k$,

$$
\begin{aligned}
G_d^k(\cdots \hat{\mu}_j^{N-(k-1)}, N-k) \;=\; & P_{N-k}(\hat{\mu}_0^{N-k}) G_d^{k+1}(\cdots \hat{\mu}_j^{N-(k-1)} \hat{\mu}_0^{N-k}, N-(k+1)) \\
& + P_{N-k}(\hat{\mu}_1^{N-k}) G_d^{k+1}(\cdots \hat{\mu}_j^{N-(k-1)} \hat{\mu}_1^{N-k}, N-(k+1)) \\
& + \cdots + P_{N-k}(\hat{\mu}_K^{N-k}) G_d^{k+1}(\cdots \hat{\mu}_j^{N-(k-1)} \hat{\mu}_K^{N-k}, N-(k+1)) \\
\text{where} \quad & G_d^N(\cdots, N-N) = G_d^N(\cdots, 0) = 1
\end{aligned}
\tag{8.3}
$$

If we continue to perform this decomposition procedure through all the subnets, finally, the LBSPN will be reduced to only one subnet. However, can we find a universal method to reduce the LBSPN subnet by subnet, without considering the global marking μ_i of the LBSPN, nor the mapping function $\rho(\mu_i)$? When we remove one subnet, can we find the state (markings) of the resulting reduced systems which are associated with the different Basic Subnet Markings of the removed subnet, without the knowledge of μ_i or $\rho(\mu_i)$? If this can be achieved, only the Basic Subnet Markings are involved in the calculation of G. Thus, the evaluation of G can be performed by dealing with every separated subnet.

2. Calculating the Normalization Constant by Reducing the LBSPN

In G_d^k, the constraints on the marking μ_is that can be involved in the G_d^k are indicated. In the $G_d^2(\hat{\mu}_j^N \hat{\mu}_l^{N-1}, N-2)$, for instance, the μ_i involved in the product is required to have $\rho(\mu_i^N) = \hat{\mu}_j^N$ and $\rho(\mu_i^{N-1}) = \hat{\mu}_l^{N-1}$. The μ_i (or μ_is) which satisfies the two conditions

will be mapped into the subnet $N-2$ to obtain the Basic Subnet Marking $\hat{\mu}_k^{N-2} = \rho(\mu_i^{N-2})$. Then $G_d^3(\hat{\mu}_j^N \hat{\mu}_l^{N-1} \hat{\mu}_k^{N-2}, N-3)$ can be produced by removing the subnet $N-2$. $G_d^2(\hat{\mu}_j^N \hat{\mu}_l^{N-1}, N-2)$ will be obtained through $G_d^3(\hat{\mu}_j^N \hat{\mu}_l^{N-1} \hat{\mu}_k^{N-2}, N-3)$s. Thus, in order to calculate G_d^k, subnet $N-k$ is removed. In order to remove subnet $N-k$, the Basic Subnet Markings of subnet $N-k$ have to be obtained under certain constraints, which are placed on the Basic Subnet Markings of the subnets that have already been removed, i.e., subnets $N-(k-1)$ to N.

Now, in order to remove subnet $N-k$, the major problem is how to obtain the Basic Subnet Markings in the separated subnet $N-k$ when the Basic Subnet Markings in subnets $N-k+1$ to N are determined. Recall that every two subnets have no common transitions but common places. Meanwhile, these common places are all initial places. Therefore, given two subnets, say subnet i and j, when the Basic Subnet Marking in subnet i is fixed as $\rho(\mu_l^i)$, the Basic Subnet Markings of subnet j that can exist under this μ_l (there may be more than one μ_l that can give the same $\rho(\mu_l^i)$) are determined by how many tokens are left in the common places, shared by subnets i and j, after subnet i has reached $\rho(\mu_l^i)$. If a Basic Subnet Marking in the subnet j can be reached without consuming any token in a common place, this marking can always exist under any basic subnet markings of subnet i (e.g., the initial marking $\hat{\mu}_0^j$). This is the principle applied in generating the associated Basic Subnet Markings for a subnet in the reduced system, when a subnet is removed from the G_d^k. According to this principle, given the Basic Subnet Markings of the subnet $N-(k-1)$ required by G_d^k, the Basic Subnet Markings of subnet $N-k$, i.e., all the possible $\hat{\mu}_j^{N-k}$s, can be found out. Then, subnet $N-k$ can again be removed from G_d^k with its different basic subnet markings. This generates G_d^{k+1}, where the Basic Subnet Markings of the subnet $N-k$, i.e., $\hat{\mu}_j^{N-k}$, becomes a new constraint on the markings μ_i involved in the G_d^{k+1}. Until finally, only one subnet is left. Then we have

$$G_d^{N-1}(\hat{\mu}_{i1}^N \hat{\mu}_{i2}^{N-1} \cdots \hat{\mu}_{i(N-1)}^{N-(N-2)}, N-(N-1)) = G_d^{N-1}(\hat{\mu}_{i1}^N \hat{\mu}_{i2}^{N-1} \cdots \hat{\mu}_{i(N-1)}^2, 1) \qquad (8.4)$$

At this stage, G_d^{N-1} is only a sum of the probabilities of some basic subnet markings of subnet 1. It can be seen that in this procedure, we only need to deal with the Basic

Subnet Markings of every subnet. No $\rho()$ function is used anymore.

Given a μ_i that satisfies the constraints in Formula 8.4, i.e., $\rho(\mu_i^N) \quad \hat{\mu}_{i1}^N$ & $\rho(\mu_i^{N-1})$ $\hat{\mu}_{i2}^{N-1}$ & $\cdots$ & $\rho(\mu_i^2) = \hat{\mu}_{i(N-1)}^2$, there are two "special" cases that we would address here.

(1) The tokens left in the subnet 1 are not sufficient to fire any of the transitions and hence no basic subnet marking can be reached in subnet 1, i.e., $\rho(\mu_i^1) = 0$, then

$$G_d^{N-1}(\hat{\mu}_{i1}^N \hat{\mu}_{i2}^{N-1} \cdots \hat{\mu}_{i(N-1)}^2, 1) = G_d^{N-1}(null, 1) = P_1(\hat{\mu}_0^1);$$

where $\hat{\mu}_0^1$ is the initial marking of subnet 1 and $P_1(\hat{\mu}_0^1)$ is the probability of $\hat{\mu}_0^1$ obtained in subnet 1. This result is obtained in accordance with the mapping function $\rho()$, where $\rho(\mu_i^j) = \hat{\mu}_0^j$ when $\underline{\mu_i^j = 0}$.

(2) If $\rho(\mu_i^1) = \hat{\mu}_0^1$, then

$$G_d^{N-1}(\hat{\mu}_{i1}^N \hat{\mu}_{i2}^{N-1} \cdots \hat{\mu}_{i(N-1)}^2, 1) = G_d^{N-1}(\hat{\mu}_0^1, 1) = \sum_{\forall i} P_1(\hat{\mu}_i^1) = 1$$

Apparently, the reason here is that all the $\hat{\mu}_i^1$ can be reached by the initial marking $\hat{\mu}_0^1$.

From the above discussion, the method for calculating G by removing the subnets is stated as the following (N subnets are assumed):

Step (1): Remove a subnet, say subnet j from the current system. The current system can be system of G or G_d^k. Assume that the current system is G_d^k. It will be the same for the case of G. Then, firstly, we generate all the possible $\hat{\mu}_i^j$ of subnet j in the given marking of the current system.

Step (2): Under each $\hat{\mu}_i^j$, i.e., keeping the token distribution in subnet j as that of marking $\hat{\mu}_i^j$, remove all the places, the tokens contained in these places and the transitions that belong to subnet j, except the common places shared with other subnets. This is the resulting reduced system that corresponds to $G_d^{k+1}(\cdots \hat{\mu}_i^j, N-(k+1))$. Step (2) is repeated for each $\hat{\mu}_i^j$ generated from step (1).

Step (3) If subnet i is the next to be removed, take the system of $G_d^{k+1}(\cdots \hat{\mu}_i^j, N-(k+1))$s obtained in the step (2) as the current system and go back to step (1). It can be noticed that one system may consist of several subsystems, i.e., G_d^{k+1} system may consist of several $G_d^{k+1}(\cdots \hat{\mu}_i^j, N-(k+1))$s.

Now, according to the method discussed above for the calculation of $G_d^k(,N-k)$, we are going to demonstrate the method by calculating the normalization constant G for the example in Fig. 8.1. We start the decomposition by removing first subnet 3 then, subnet 2.

Step 1: Subnet 3 is removed.

There is no marking constraint here since the subnet 3 is the first subnet to be removed. The Basic Subnet Markings of the subnet 3 are $\hat{\mu}_0^3 = \{1(1)\}$, $\hat{\mu}_1^3 = \{5(1)\}$ and $\hat{\mu}_2^3 = \{6(1)\}$.

(1) $\hat{\mu}_0^3 = \{1(1)\} \longrightarrow G_d^1(\hat{\mu}_0^3, 2)$

After the subnet 3 is removed with its marking as $\hat{\mu}_0^3 = \{1(1)\}$, the net left is shown in Fig. 8.3a). In this case, the Basic Subnet Markings that can be reached in subnet 2 are: $\hat{\mu}_0^2 = \{1(1), 7(1)\}$, $\hat{\mu}_1^2 = \{2(1), 7(1)\}$, $\hat{\mu}_2^2 = \{3(1)\}$ and $\hat{\mu}_3^2 = \{4(1)\}$.

(2) $\hat{\mu}_1^3 = \{5(1)\} \longrightarrow G_d^1(\hat{\mu}_1^3, 2)$

The resulting net, after subnet 3 is removed with its marking as $\hat{\mu}_1^3 = \{5(1)\}$, is in Fig. 8.3b). Now, the Basic Subnet Markings that can be reached in subnet 2 is only the initial marking $\hat{\mu}_0^2 = \{1(1), 7(1)\}$, recalling that initial marking of a subnet can exist under any token distribution situation.

(3) $\hat{\mu}_2^3 = \{6(1)\} \longrightarrow G_d^1(\hat{\mu}_2^3, 2)$

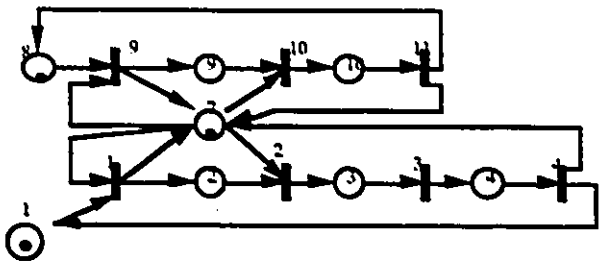
The resulting reduced net is the same as that in Fig. 8.3b). Similarly, only $\hat{\mu}_0^2 = \{1(1), 7(1)\}$ can exist in subnet 2 in this case. Hence we have $G_d^1(\hat{\mu}_2^3, 2) = G_d^1(\hat{\mu}_1^3, 2)$. Therefore, at the end of the step 1, we have

$$\begin{aligned} G &= P_3(\hat{\mu}_0^3) \times G_d^1(\hat{\mu}_0^3, 2) + P_3(\hat{\mu}_1^3) \times G_d^1(\hat{\mu}_1^3, 2) + P_3(\hat{\mu}_2^3) \times G_d^1(\hat{\mu}_2^3, 2) \\ &= P_3\{1(1)\} \times G_d^1(\hat{\mu}_0^3, 2) + \{P_3\{5(1)\} + P_3\{6(1)\}\} \times G_d^1(\hat{\mu}_1^3, 2) \end{aligned} \tag{8.5}$$

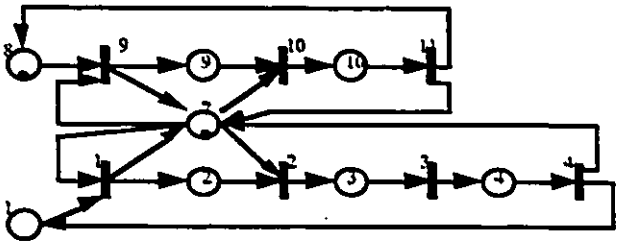
Here $P_3\{1(1)\}$ represent the probability of the Basic Subnet Marking $\hat{\mu}_0^3 = \{1(1)\}$ in subnet 3. Notations $P_3\{5(1)\}$ and $P_3\{6(1)\}$ are used in the similar way.

Step 2: Subnet 2 is removed.

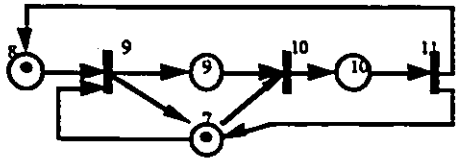
(1) Under the constraint of $G_d^1(\hat{\mu}_0^3, 2)$, i.e the current system is the subsystem $G_d^1(\hat{\mu}_0^3, 2)$ in Fig. 8.3a).



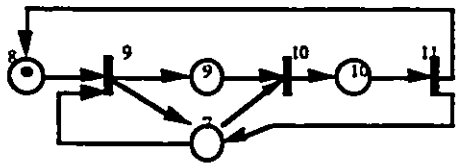
a) Subnet 3 Is Removed with Basic Subnet Marking {1(1)}



b) Subnet 3 Is Removed with Basic Subnet Marking {5(1)} or {6(1)}



c) Subnet 2 Is Removed with Basic Subnet Marking {1(1),7(1)} or {2(1),7(1)}



d) Subnet 2 Is Removed with Basic Subnet Marking {3(1)} or {4(1)}

Figure 8.3: Reducing the LBSPN to calculate C

In this case (see Fig. 8.3a)), Basic Subnet Markings $\hat{\mu}_0^2 = \{1(1), 7(1)\}$, $\hat{\mu}_1^2 = \{2(1), 7(1)\}$, $\hat{\mu}_2^2 = \{3(1)\}$ and $\hat{\mu}_3^2 = \{4(1)\}$ can all be reached in subnet 2, as mentioned above.

When subnet 2 is removed with its marking $\hat{\mu}_0^2$, the resulting net is shown in Fig. 8.3c). The Basic Subnet Markings that can be reached in subnet 1, in this case, are $\hat{\mu}_0^1 = \{7(1), 8(1)\}$, $\hat{\mu}_1^1 = \{7(1), 9(1)\}$ and $\hat{\mu}_2^1 = \{10(1)\}$. That is

$$G_d^2(\hat{\mu}_0^3\hat{\mu}_0^2, 1) = G_d^2(\hat{\mu}_0^1, 1) = 1 = P_1\{7(1), 8(1)\} + P_1\{7(1), 9(1)\} + P_1\{10(1)\}$$

When subnet 2 is removed with its marking $\hat{\mu}_1^2 = \{2(1), 7(1)\}$, the resulting net is the same as that in Fig. 8.3c). Thus, $G_d^2(\hat{\mu}_0^3\hat{\mu}_1^2, 1) = 1$.

When subnet 2 is removed with its marking $\hat{\mu}_2^2 = \{3(1)\}$, the resulting net is shown in Fig. 8.3d). No transition seems to be enabled as place P_7 contains no token. However, recall that the markings that consume no tokens from a common place can always exist. Thus, $\hat{\mu}_0^1 = \{7(1), 8(1)\}$ as well as $\hat{\mu}_1^1 = \{7(1), 9(1)\}$ can both be"reached". $\hat{\mu}_1^1 = \{7(1), 9(1)\}$ can be "reached" because the transition t_9 "consumes" no token from place P_7 by taking P_7 as its output as well as the input place. Hence, we have

$$G_d^2(\hat{\mu}_0^3\hat{\mu}_2^2, 1) = P_1\{7(1), 8(1)\} + P_1\{7(1), 9(1)\}$$

When the subnet 2 is removed with its marking $\hat{\mu}_3^2 = \{4(1)\}$, the resulting net is the same as that shown in Fig. 8.3d). We thus have again

$$G_d^2(\hat{\mu}_0^3\hat{\mu}_3^2, 1) = P_1\{7(1), 8(1)\} + P_1\{7(1), 9(1)\}$$

(2) Under the constraint of $G_d^1(\hat{\mu}_1^3, 2)$, i.e., the current system is the subsystem $G_d^1(\hat{\mu}_1^3, 2)$ in Fig. 8.3b).

In this case, we have only $\hat{\mu}_0^2 = \{1(1), 7(1)\}$ for subnet 2. Remove subnet 2 with its marking $\hat{\mu}_0^2 = \{1(1), 7(1)\}$. The resulting reduced system is same as that in Fig. 8.3c). As shown in Fig. 8.3c), all the Basic Subnet Markings of subnet 1 can be reached. Hence

$$G_d^2(\hat{\mu}_1^3\hat{\mu}_0^2, 1) = G_d^2(\hat{\mu}_0^1, 1) = 1$$

Therefore, formula 8.5 from the Step 1 becomes:

$$G = P_3\{1(1)\}G_d^1(\hat{\mu}_0^3, 2) + \{P_3\{5(1)\} + P_3\{6(1)\}\}G_d^1(\hat{\mu}_1^3, 2)$$

$$
\begin{aligned}
= \; & P_3\{1(1)\}(P_2(\hat{\mu}_0^2)G_d^2(\hat{\mu}_0^3\hat{\mu}_0^2, 1) + P_2(\hat{\mu}_1^2)G_d^2(\hat{\mu}_0^3\hat{\mu}_1^2, 1) \\
& + P_2(\hat{\mu}_2^2)G_d^2(\hat{\mu}_0^3\hat{\mu}_2^2, 1) + P_2(\hat{\mu}_3^2)G_d^2(\hat{\mu}_0^3\hat{\mu}_3^2, 1)) \\
& + (P_3\{5(1)\} + P_3\{6(1)\})\{P_2(\hat{\mu}_0^2)G_d^2(\hat{\mu}_1^3\hat{\mu}_0^2, 1)\} \\
= \; & P_3\{1(1)\}(P_2\{1(1), 7(1)\} + P_2\{2(1), 7(1)\} + P_2\{3(1)\}P_1\{7(1), 8(1)\} \\
& + P_2\{3(1)\}P_1\{7(1), 9(1)\} + P_2\{4(1)\}P_1\{7(1), 8(1)\} \\
& + P_2\{4(1)\}P_1\{7(1), 9(1)\}) + (P_3\{5(1)\} + P_3\{6(1)\})P_2\{1(1), 7(1)\}
\end{aligned}
$$

The G obtained here is exactly the same as the G calculated directly through formula 8.1.

The above decomposition procedure can be started from either of the three subnets. The removal of the subnets can be conducted in any order. Same results will be obtained.

3.Calculating The Normalization Constant by Subnet Convolution

If we turn the order of the steps in the above procedure for the calculation of G, the convolution of the subnets is performed. That is, we start from a single subnet, adding in one more subnet at a time, until the entire net is built. Every time, when a new subnet is put in, the new G_d^k is evaluated. The new subnet hence is convolved into the LBSPN. Finally, the entire LBSPN is convolved and the normalization constant G is obtained. The steps for the subnet convolution are as follows (N subnets are assumed in the LBSPN).

(1) Subnet j is added into system G_d^k. First, find out all the $\hat{\mu}_i^j$ of subnet j. Check the common places that are going to be shared by subnet j and the subnet(s) in system G_d^k.

(2) For every $\hat{\mu}_i^j$, keep the system G_d^k. Only the number of tokens in the common places of G_d^k is adjusted to be in conformity with the token distribution of marking $\hat{\mu}_i^j$. After the token adjustment, several subsystems may be generated for system G_d^k, distinguished only by their different token distributions in the common places.

(3) Check the subnets (e.g., subnet i) in system G_d^k which will share the common places with the subnet j and find out their Basic Subnet Markings that can be reached in every subsystem, after the token adjustment in step (2). Calculate $G_d^{k-1}(\cdots\hat{\mu}_i^j, N-k+1)$

according to the formula (3) (section 3.1).

(4) The places and the transitions of subnet j are added in, with the initial subnet j markings. This is the new system of G_d^{k-1}. If another subnet is the next to be convoluted, go back to step (1) with system G_d^{k-1}. If all the subnets have already been added in, sum all the G_d^{k-1} resulted from step (3) to obtain G.

Now we demonstrate this convolution procedure by calculating again, the G for the LBSPN in Fig. 8.1. We start from the subnet 1.

(1) Subnet 2 is added in.

At this time, only subnet 1 exists. There are four Basic Subnet Markings of subnet 2, $\hat{\mu}_0^2 = \{1(1), 7(1)\}$, $\hat{\mu}_1^2 = \{2(1), 7(1)\}$, $\hat{\mu}_2^2 = \{3(1)\}$ and $\hat{\mu}_3^2 = \{4(1)\}$. Under the markings $\hat{\mu}_0^2$ and $\hat{\mu}_1^2$, common place P_7 contains one token (Fig. 8.3c)). Under $\hat{\mu}_2^2$ and $\hat{\mu}_3^2$, P_7 contains no token (Fig. 8.3d)).

From Fig. 8.3c), in case of both $\hat{\mu}_0^2$ and $\hat{\mu}_1^2$, we have

$$G_d^2(\hat{\mu}_i^3\hat{\mu}_0^2, 1) = 1 = G_d^2(\hat{\mu}_i^3\hat{\mu}_1^2, 1) = P_1\{7(1), 8(1)\} + P_1\{7(1), 9(1)\} + P_1\{10(1)\}$$

From Fig. 8.3d), we have

$$G_d^2(\hat{\mu}_i^3\hat{\mu}_2^2, 1) = G_d^2(\hat{\mu}_i^3\hat{\mu}_3^2, 1) = P_1\{7(1), 8(1)\} + P_1\{7(1), 9(1)\}$$

The $\hat{\mu}_i^3$ appears in the G_d^2 here because the subnet 3 has not put in yet.

Then, subnet 2 is connected with subnet 1.

(2) Subnet 3 is added in

There are three Basic Subnet Marking of subnet 3, $\hat{\mu}_0^3 = \{1(1)\}$, $\hat{\mu}_1^3 = \{5(1)\}$ and $\hat{\mu}_2^3 = \{6(1)\}$. Common place P_1 contains 1 token under the marking $\hat{\mu}_0^3$ (Fig. 8.3a)) and contains 0 token in the other two cases (Fig. 8.3b)).

In the case of $\hat{\mu}_0^3 = \{1(1)\}$ (see Fig. 8.3a)), subnet 2 can reach all its Basic Subnet Markings. Thus

$$\begin{aligned} G_d^1(\hat{\mu}_0^3, 2) \;=\; & P_2\{1(1), 7(1)\}G_d^2(\hat{\mu}_i^3\hat{\mu}_0^2, 1) + P_2\{2(1), 7(1)\}G_d^2(\hat{\mu}_i^3\hat{\mu}_1^2, 1) \\ & + P_2\{3(1)\}G_d^2(\hat{\mu}_i^3\hat{\mu}_2^2, 1) + P_2\{4(1)\}G_d^2(\hat{\mu}_i^3\hat{\mu}_3^2, 1) \end{aligned}$$

In the case of both $\hat{\mu}_1^3 = \{5(1)\}$ and $\hat{\mu}_2^3 = \{6(1)\}$ (Fig. 8.3b)), subnet 2 can only have marking $\hat{\mu}_0^2 = \{1(1), 7(1)\}$. Hence

$$G_d^1(\hat{\mu}_1^3, 2) = G_d^1(\hat{\mu}_2^3, 2) = P_2\{1(1), 7(1)\}G_d^2(\hat{\mu}_2^3\hat{\mu}_0^2, 1) = P_2\{1(1), 7(1)\}$$

Therefore, finally we have

$$\begin{aligned}
G &= P_3\{1(1)\}G_d^1(\hat{\mu}_0^3, 2) + P_3\{5(1)\}G_d^1(\hat{\mu}_1^3, 2) + P_3\{6(1)\}G_d^1(\hat{\mu}_2^3, 2) \\
&= P_3\{1(1)\}(P_2\{1(1), 7(1)\}G_d^2(\hat{\mu}_1^3\hat{\mu}_0^2, 1) + P_2\{2(1), 7(1)\}G_d^2(\hat{\mu}_1^3\hat{\mu}_1^2, 1) \\
&\quad + P_2\{3(1)\}G_d^2(\hat{\mu}_1^3\hat{\mu}_2^2, 1) + P_2\{4(1)\}G_d^2(\hat{\mu}_1^3\hat{\mu}_3^2, 1)) \\
&\quad + (P_3\{5(1)\} + P_3\{6(1)\})P_2\{1(1), 7(1)\} \\
&= P_3\{1(1)\}(P_2\{1(1), 7(1)\} + P_2\{2(1), 7(1)\} + P_2\{3(1)\}P_1\{7(1), 8(1)\} \\
&\quad + P_2\{3(1)\}P_1\{7(1), 9(1)\} + P_2\{4(1)\}P_1\{7(1), 8(1)\} \\
&\quad + P_2\{4(1)\}P_1\{7(1), 9(1)\}) + (P_3\{5(1)\} + P_3\{6(1)\})P_2\{1(1), 7(1)\}
\end{aligned}$$

This convolution procedure can also be conducted by starting from any of the subnets and convolving the subnets in any order.

In the following, however, we are not going to follow this procedure on the subnets anymore. The convolution is going to be carried out in the table format. That is, the probability distributions of each separated subnets will be applied directly in the calculation of G. The computational algorithm for the normalization constant G can thus be very easily executed and can be implemented in software.

8.2.2 Computational Algorithm for The Evaluation of G

The computational algorithm for the evaluation of G is presented also through the example of the LBSPN in Fig. 8.1. The above subnet convolution is conducted directly in a table format. We start from any two of the subnets. For instance, the subnet 2 and 3 are taken first, as illustrated in the Table 8.1 of Fig. 8.4. The probabilities of the Basic Subnet Markings are listed in the table, as well as the number of tokens consumed by the marking from the common place(s), which is (are) shared by the two subnets. The

$P_1()$ \ $P_2()$		$P_2\{1(1)7(1)\}$	$P_2\{2(1)7(1)\}$	$P_2\{3(1)\}$	$P_2\{4(1)\}$
		0	0		
$P_1\{7(1)8(1)\}$	0	$\Pi[1(78)2(17)]$	$\Pi[1(78)2(27)]$	$\Pi[1(78)2(3)]$	$\Pi[1(78)2(4)]$
$P_1\{10(1)\}$	1	$\Pi[1(10)2(17)]$	$\Pi[1(10)2(27)]$	0	0
$P_1\{7(1)9(1)\}$	0	$\Pi[1(79)2(17)]$	$\Pi[1(79)2(27)]$	$\Pi[1(79)2(3)]$	$\Pi[1(79)2(4)]$

Table 1

$P_3()$ \		$P_3\{1(1)\}$	$P_3\{5(1)\}$	$P_3\{6(1)\}$
		0		
$\Pi[1(78)2(17)]$	0	$\Pi[1(78)2(17)3(1)]$	$\Pi[1(78)2(17)3(5)]$	$\Pi[(78)2(17)3(6)]$
$\Pi[1(78)2(27)]$	1	$\Pi[1(78)2(27)3(1)]$		
$\Pi[1(78)2(3)]$	1	$\Pi[1(78)2(3)3(1)]$		
$\Pi[1(78)2(4)]$	1	$\Pi[1(78)2(4)3(1)]$		
$\Pi[1(10)2(17)]$	0	$\Pi[1(10)2(17)3(1)]$	$\Pi[1(10)2(1)3(5)]$	$\Pi[1(10)2(17)3(6)]$
$\Pi[1(10)2(27)]$	1	$\Pi[1(10)2(27)3(1)]$		
$\Pi[1(79)2(17)]$	0	$\Pi[1(79)2(17)3(1)]$	$\Pi[1(79)2(17)3(5)]$	$\Pi[1(79)2(17)3(6)]$
$\Pi[1(79)2(27)]$	1	$\Pi[1(79)2(27)3(1)]$		
$\Pi[1(79)2(3)]$	1	$\Pi[1(79)2(3)3(1)]$		
$\Pi[1(79)2(4)]$	1	$\Pi[1(79)2(4)3(1)]$		

Table 2

Figure 8.4: Convolution Tables

common place shared by subnet 2 and 3 is place P_1. The basic subnet marking $\{1(1), 7(1)\}$ of subnet 2 consumes no token from the place P_1. On the other hand, the Basic Subnet Marking (5) of subnet 3 consumes 1 token from the place P_1. Since there is only one token in place P_1 in the initial marking of the net, therefore we consider only the entries of which the total number of tokens consumed by subnets 2 and 3 from place P_1 is less than or equal to 1. That is, at these entries in the table, the multiplication of the two probabilities is obtained. In the table, for example, $P_2\{1(1), 7(1)\} \times P_1\{7(1), 8(1)\}$ in Table 1 is denoted as $\prod[1(78)2(17)]$. After Table 1 is computed, subnet 1 is added in as shown in the Table 2 of Fig. 8.4. The same step is followed. Subnet 1 has a common place P_7 shared with subnet 2. Thus the tokens in place P_7 consumed by subnet 1 and 2 are recorded. Then the multiplications are carried out, in accordance with the number of tokens in P_7. After Table 2 is calculated, the entire LBSPN is built up. The sum of all the entries in Table 2 gives the normalization constant G. This example can also be computed according to different orders of the subnets. For instance, in Table 3 and 4 of Fig. 8.5, G is evaluated by starting with subnet 3 and subnet 1, which don't have any common place. Then subnet 2 is added in. Subnet 2 has a common place P_7 shared with subnet 1 and a place P_1 shared with subnet 3. Thus both the two places are checked. Then G is obtained as the same. In Fig. 8.6, another example of LBSPN is given with its two subnets. The Basic Subnet Marking $\{1(2), 9(1)\}$ represents that two tokens are contained in place P_1 and only one token in place P_9. Similar meaning associates the marking $\{2(2), 9(1)\}$. This LBSPN has 17 markings. Its Markov Chain is shown in Fig. 8.7. The evaluation of G can be easily finished as shown in Table 5 and 6 of Fig. 8.8. Here, $\prod[1(49)2(2_29)] = P_1\{4(1), 9(1)\} \times P_2\{2(2), 91\}$ The computational algorithm given here allows the steady state probability distributions of every separated subnet to be applied directly, to obtain the Normalization Constant G. We do not even have to look into the structural details of every subnet. This computational algorithm can be easily implemented in software.

When many subnets are involved, the algorithm will generate too many intermediate results unless certain aggregation of the intermediate products is performed. The

$P_3(\ )$ \ $P_1(\ )$		$P_1\{7(1)8(1)\}$	$P_1\{7(1)9(1)\}$	$P_1\{10(1)\}$
		0	0	0
$P_3\{1(1)\}$	0	Π[1(78)3(1)]	Π[1(79)3(1)]	Π[1(10)3(1)]
$P_3\{5(1)\}$	0	Π[1(78)3(5)]	Π[1(79)3(5)]	Π[1(10)3(5)]
$P_3\{6(1)\}$	0	Π[1(78)3(6)]	Π[1(79)3(6)]	Π[1(10)3(6)]

Table 3

$P_2(\ )$ \ P_{2_1}		$P_2\{1(1)7(1)\}$	$P_2\{2(1)7(1)\}$	$P_2\{3(1)\}$	$P_2\{4(1)\}$
		00	01	11	11
Π[1(78)3(1)]	00	Π[1(78)2(17)3(1)]	Π[1(78)2(27)3(1)]	Π[1(78)2(3)3(1)]	Π[1(78)2(4)3(1)]
Π[1(78)3(5)]	01	Π[1(78)2(17)3(5)]			
Π[1(78)3(6)]	01	Π[1(78)2(17)3(6)]			
Π[1(79)3(1)]	00	Π[1(79)2(17)3(1)]	Π[1(79)2(27)3(1)]	Π[1(79)2(3)3(1)]	Π[1(79)2(4)3(1)]
Π[1(79)3(5)]	01	Π[1(79)2(17)3(5)]			
Π[1(79)3(6)]	01	Π[1(79)2(17)3(6)]			
Π[1(10)3(1)]	10	Π[1(10)2(17)3(1)]	Π[1(10)2(27)3(1)]		
Π[1(10)3(5)]	11	Π[1(10)2(17)3(5)]			
Π[1(10)3(6)]	11	Π[1(10)2(17)3(6)]			

Table 4

Figure 8.5: Convolution tables

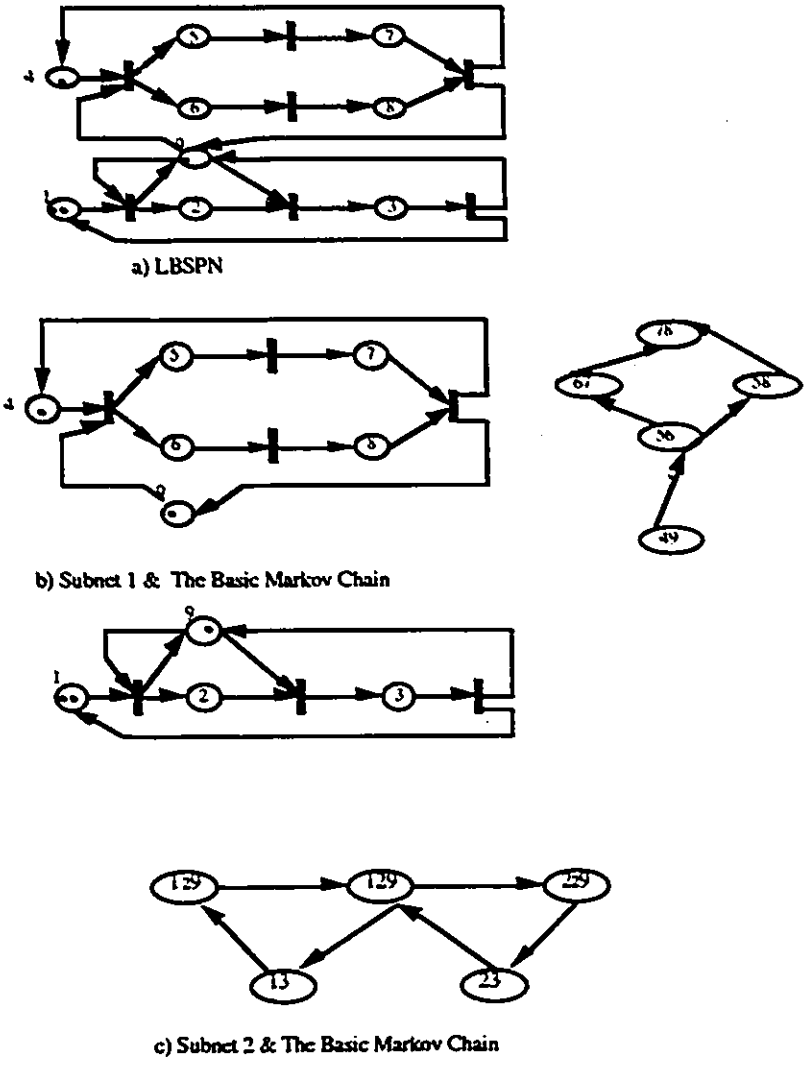


Figure 8.6: Example 2: a LBSPN, its two subnets & the basic Markov chains

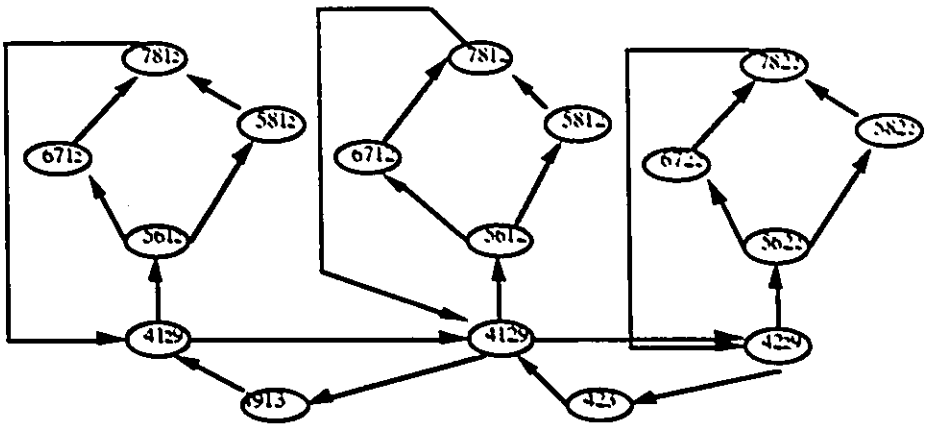


Figure 8.7: The Markov chain of the LBSPN example 2

$P_1$0 \ $P_2$0		$P_2$1(2)9(1)	$P_2$1(1)3(1)	$P_2$1(1)2(1)9(1)	$P_2$2(1)3(1)	$P_2$2(2)9(1)
		0	1	0	1	0
$P_1$4(1)9(1)	0	Π[(1(49)2(129)]	Π[(1(49)2(13)]	Π[(1(49)2(129)]	Π[(1(49)2(23)]	Π[(1(49)2(229)]
$P_1$5(1)6(1)	1	Π[(1(56)2(129)]		Π[(1(56)2(129)]		Π[(1(56)2(229)]
$P_1$6(1)7(1)	1	Π[(1(67)2(129)]		Π[(1(67)2(129)]		Π[(1(67)2(229)]
$P_1$5(1)8(1)	1	Π[(1(58)2(129)]		Π[(1(58)2(129)]		Π[(1(58)2(229)]
$P_1$7(1)8(1)	1	Π[(1(78)2(129)]		Π[(1(78)2(129)]		Π[(1(78)2(229)]

Table 5

$P_3$0		*sum1*	*sum2*
		0	1
P_3(1(1))	0	Π[3(1)*sum1*]	Π[3(1)*sum2*]
P_3(5(1))	1	Π[3(5)*sum1*]	
P_3(6(1))	1	Π[3(6)*sum1*]	

Table 6

Figure 8.8: Convolution tables of LBSPNs

aggregation of the intermediate results is conducted with respect to the different Basic Subnet Markings of the subnet that is going to be added in. For instance, in the example shown in Table 1 & 2, the products obtained from Table 1 can be aggregated before being entered into Table 2. The aggregation is performed with respect to the number of tokens in place P_1 and the Basic Subnet Markings of subnet 3. $\hat{\mu}_0^3 = \{1(1)\}$ consumes no token from place P_1. Both $\hat{\mu}_1^3 = \{5(1)\}$ and $\hat{\mu}_2^3 = \{6(1)\}$ absorbs one token from P_1. Since the initial marking μ_0 has only one token in place P_1, the products from Table 1 are divided into two groups. In the first group, the products that involve the markings which consume no token from P_1 are included. The second group includes the products that involve the markings which consume 1 token from P_1. The sum of the products in each group will be entered into Table 2. These two sums are as follows:

$$\begin{aligned} sum1 &= \prod|1(78)2(17)| + \prod|1(10)2(17)| \\ &\quad + \prod|1(79)2(17)| = P_2\{1(1), 7(1)\} \end{aligned}$$

$sum2 = \sum$ (all the other entries in Table 1)

Thus, Table 2 is simplified as shown in Table 6 of Fig. 8.8.

The computational algorithm presented in this section has been programmed in C language on UNIX OS on Sun workstation 3/150. In Appendix 3, the listing of the program is presented. Given the basic markings of each subnet, the software performs the subnet convolution and generates the normalization constant G.

8.3 Performance Modeling of Multimedia Collaborations

In this section, we are going to apply LBSPN models for the performance evaluation of the muitimedia collaborative applications, e.g., the joint editing session.

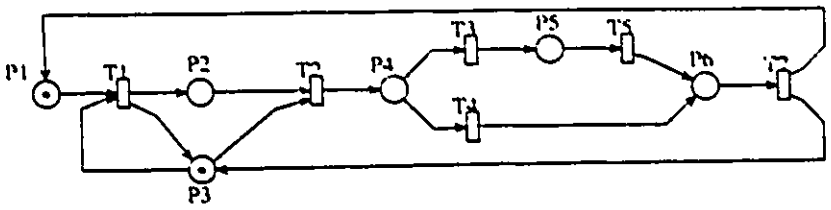
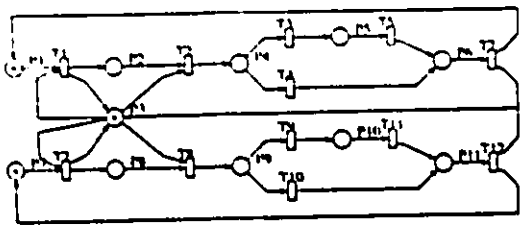


Figure 8.9: SPN model of one editor in a collaborative group

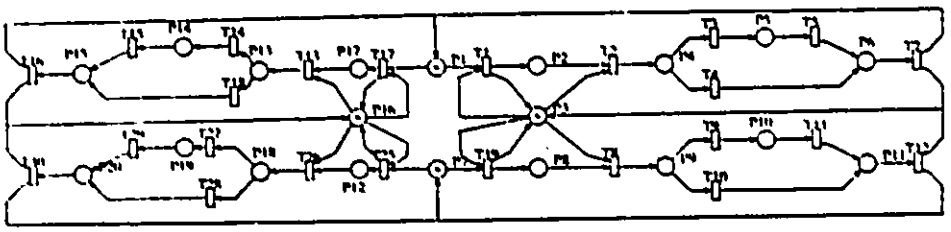
8.3.1 LBSPN models for Multimedia Joint Editing

As we have discussed in Chapter 5, in multimedia joint editing, editors are divided into the collaborative groups. For a collaborative group with multiple editors, we establish its LBSPN model.

Fig. 8.9 illustrates a SPN model which represents an editor working on a portion of the document. The token in place $P1$ represents the editor and the token in $P3$ stands for the document portion. The editor may view a certain portion and crank ideas to modify it. This is represented by the transition $T1$ in Fig. 8.9. Then the editor needs to reserve the portion, i.e., to "lock" it (see Chapter 5) through transition $T2$. After the editing (transition $T3$), the new version of the portion will be committed (sent through $T5$) into the shared space $P3$, so that all the other participants can view it. That is, the portion is available to all the participants in the collaborative group. On the other hand, collision may be incurred by an editor where the portion is reserved by another one in the group, at almost the same time. In this case, the editor is assumed to wait in $P2$ until the new version of this portion is committed from the one who has reserved it. Then the editor tries again to reserve the portion. In any case, after reserving the portion, the editor can cancel the reservation without making any modifications. This abortion of the editing is represented by transition $T4$ in th SPN model. Take the SPN model in Fig. 8.9 as a subnet, the LBSPN models can be constructed, describing multiple editors working on a multimedia document which consists of multiple portions. In Fig. 8.10 (a), two editors are shown collaborating on the same document portion. In Fig. 8.10 (b), the document comprises two portions so that the two editors can work on the document concurrently by reserving different portions. The models in Fig. 8.10 are



(a)



(b)

Figure 8.10: LBSPN model containing two editors

Class\Transitions	T1	T2	T3	T4	T5	T6
class 1	0.0083	1.00	0.0083	0.0056	0.0042	1.00
class 2	0.0056	1.00	0.0056	0.0042	0.0033	1.00

Figure 8.11: Assignment of the firing rates

both LBSPN models. Similarly, LBSPN models can be constructed by connecting the subnet to represent the collaborating group which includes more editors and document portions. When more editors and document portions are added in the model, the state space is dramatically increased. Thus, product form solution method is employed to solve the model.

8.3.2 Product Form Solutions & Performance Results

Editors may work at different speeds. This can be easily represented by the different transition rates for the subnet in Fig. 8.9. In our example, two different sets of rates are assigned to the subnet, shown in Fig. 8.11 as the working speed of class 1 and class 2. For each class of the transition rates, the subnet is solved. Thus the subnet which has only 5 steady states is solved twice, resulting in the different steady state distributions shown in Fig. 8.12. First, we study the case where only class 1 editors are working on a single document portion. When the number of the subnets is increased, Fig. 8.13 gives the G

With Class 1 Rates		*With Class 2 Rates 2*	
P{1(1),3(1)}	0.357942	P{1(1),3(1)}	0.391800
P{2(1),3(1)}	0.002971	P{2(1),3(1)}	0.002194
P{4(1)}	0.213735	P{4(1)}	0.223885
P{5(1)}	0.422381	P{5(1)}	0.379927
P{6(1)}	0.002971	P{6(1)}	0.002194

Figure 8.12: Steady State Distribution of The Subnet with Different Firing Rates

	1 editor	2 editors	3 editors	4 editors	5 editors	6 editors	7 editors	8 editors
G	1	0.591568	0.296750	0.137146	0.060311	0.025691	0.010685	0.043661

Figure 8.13: Normalization Constant G

computed through the convolution of the subnet. The results are directly obtained from the coded program. From G, the probabilities of some relevant states are calculated, given the LBSPN models which included different number of the editors. Then the throughput of every editor as well as that of the entire group is obtained, as presented in Fig. 8.14.

Next, the collaborating group with 3 concurrently accessible portions of the shared document is studied. Similarly, the number of the editors is increased. In this case, we assume that the editor works on the first portion with the rates of class 2 and the other two portions with the rates of class 1. This can be caused by either the different contents of the portion or the different media used in the portion. For instance, modifications made on a video clip takes different time from those made on a text piece in the portion. In Fig. 8.15, the normalization constant G for the LBSPN models which include different number of the editors are given. Fig. 8.16 presents the throughput of every editor and of the entire system.

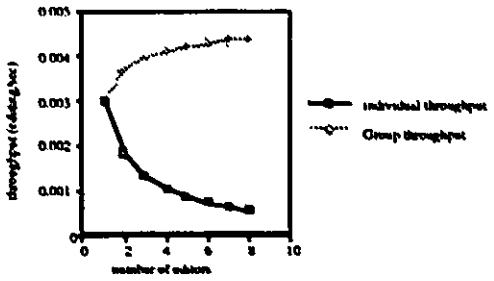
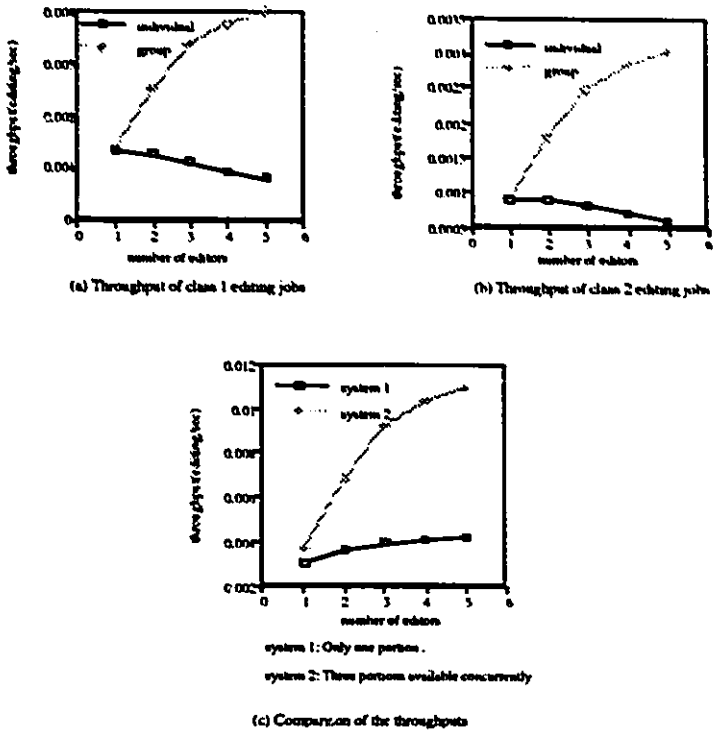


Figure 8.14: Throughput of case 1

	1 editor	2 editors	3 editors	4 editors	5 editors
G	0.3082095	0.072898369	0.012664578	0.00160754	0.000167147

Figure 8.15: Normalization Constant G



(a) Throughput of class 1 editing jobs

(b) Throughput of class 2 editing jobs

system 1: Only one person.

system 2: Three persons available concurrently

(c) Comparison of the throughputs

Figure 8.16: Throughput of case 2

8.4 Summary

In this chapter, we employed a new performance model, the LBSPN to evaluate the multimedia collaborative application of joint editing. We first have developed a computational algorithm for the LBSPN models so that the product form solution of LBSPN can be obtained efficiently. The algorithm is based on the convolution of the subnets in the LBSPN model.

In our studies, we have found that the LBSPN are useful in modeling the shared multimedia collaborative applications. We took the joint editing session as an example and obtained the performance results of the system.

Chapter 9

Conclusions

9.1 Summary of The Thesis

In this thesis, multimedia synchronization control schemes were proposed. The thesis also presents analytical performance modeling of the multimedia application systems where the synchronization is involved. Since multimedia synchronization control is a relatively new topic in the computer communication field, we first, in Chapter 1, describe the synchronization problems that are involved in various multimedia application systems. The synchronization control issues were thus divided into three catalogues, the Temporal Presentation Control (TPC), the Simultaneous Real-Time Data Delivery (SRTDD) and the multimedia causal ordering. In Chapter 2, we reviewed the synchronization control methods that were proposed in the literature.

From Chapter 3 to Chapter 5, which forms the first part of our thesis, we presented our synchronization control scheme and algorithms. Chapter 3 deals with the issue of TPC. We took the challenging problem of synchronization control in the teleorchestra system where independent information sources are involved. A temporal model has been established for the "fuzzy" presentation scenarios that appeared in the multimedia tele-orchestration applications. With only the relative and uncertain temporal information available, the model can represent the required presentation scenario and thus, support the synchronization dialogue among the multiple information sources. Based on this

temporal model, synchronization algorithms which include the synchronization dialogue were devised. Interesting multimedia applications such as the remote document creation, fast-forwarding and fast-rewinding of the multimedia document presentation can be supported.

In order to support the real-time multimedia data delivery, we proposed in Chapter 4, an entire multimedia communication architecture. In this layered model, multimedia session management was designed which can manage the temporal relationships involved in the multiple data streams. Then we described the Activity Constraint Control (ACC) and the multimedia Segment Delivery Scheme (SDS). The ACC can maintain the sequential ordering of data on different streams according to the requirement of the presentation application. Thus, the various data contents (e.g., objects on different streams) will not be mismatched during the simultaneous play-back. As the multimedia presentation requires the coupling between the data streams, we designed the SDS to achieve the required coupling, which is quantitatively defined by the skew parameters. The skew parameters are actually the Synchronization Quality of Service (SQoS) parameters. Different segment delivery schemes were proposed.

In Chapter 5, a different type of the synchronization control was investigated, in a multimedia cooperative system. We presented a synchronization model for the multimedia collaborative application of joint editing. The algorithms was designed for the synchronization control in a shared multimedia working space. The concurrent control in the system is managed. Meanwhile, the causal ordering of the different versions of the document is preserved in the system, without tracking the history of the document.

In the second part of our thesis, we have investigated the analytical performance modeling techniques for the multimedia system where the synchronization is involved. Chapter 6 reviews the existing analytical performance models and their solution techniques. Since the Stochastic Petri Nets (SPN) models can represent the synchronization aspects of a system, they were chosen as our mathematical models for the performance evaluation of the multimedia systems.

Different kinds of the multimedia application systems were taken as the examples in

the performance modeling conducted in Chapter 7. The GSPN (Generalized Stochastic Petri Nets) models were applied to model the multimedia Segment Delivery scheme. The performance indices of the segment delivery scheme were obtained, which are useful in the system design and implementation. The CGSPN (Coloured Generalized Stochastic Petri Nets) models were used to model the remote document creation system. By representing the temporal behavior of the information sources (e.g., the database servers), the CGSPN models were established for the distributed multimedia integration. The system performance was studied, when integration jobs are concurrently required from the users.

In these studies, we have found that the SPN has many advantages in modeling the synchronization aspects of multimedia communications. The SPN models are apparently more powerful than the QN (Queueing Networks) models in representing the concurrency and synchronization in the multimedia systems. However, the solution for a SPN model of a real system is threatened by the explosion of the state space. In any of the models that we have established for the multimedia systems, an increase in the number of tokens in one place in the net can often result in a huge state space (e.g., more than 100,000 markings) that hampers the solution of the model. Therefore, efficient solution algorithms for SPN models are required.

We studied the solution technique for the SPN models in Chapter 8. A class of SPN models, i.e., the Locally Balanced Stochastic Petri Nets (LBSPN) which have product form solutions were investigated. We presented a computational algorithm for the LBSPN, so that the product form solution can be utilized conveniently in the performance analysis of the real systems. The computational algorithm was based on the convolution of the subnets in the LBSPN. A software program of this convolutional algorithm was also developed on the UNIX system. Then, the LBSPN was employed to evaluate the performance of the multimedia joint editing system. When the number of the editors and also the concurrency degree allowed in the system are increased, the performance results can easily be obtained by convolving the added subnets. The performance results were presented.

9.2 Suggestion for Future Research

Multimedia synchronization has become a hot topic in the area of multimedia communications. However, the problem has not turned out to be clear; nor easy to solve. The main reason is that synchronization control has to be embedded into the application and communication systems that are being developed and evolved. With the introduction of the new multimedia data transmission scheme and system architecture, the synchronization problem and the control method have to be adapted so that they are suitable for new and different systems.

The temporal models and algorithms for the TPC control have been well studied. Nevertheless, the implementation of the system requires more efforts. Languages and programming interfaces for describing the temporal relationships of object presentation events are just now under study, development or available in first version [33] [14] [46]. However, most of them either aim at the general document structure and user's interface, or aim at the communication architecture alone. How to connect these two parts, that is, the development of a distributed document creation and presentation application environment on the communication networks, e.g., the teleorchestra system, needs to be studied. This involves the implementation of the TPC over the distributed system, which is an open problem. Meanwhile, the system has to comprise both the TPC and SRTDD control. TPC and SRTDD thus need to be connected. On the other hand, in the distributed applications, the TPC need to be combined with the concurrency control in the system. For instance, when multiple clients perform the remote document creation concurrently, the independent information sources need to initiate the synchronization dialogues properly during the source contention. Issues like this need to be solved before a public multimedia presentation or conversational application can be supported.

In the study of SRTDD, we were focused on the synchronization of the different data streams. In this case, ATM networks were assumed. We didn't look into the connection of different networks, for instance, from LAN to the ATM network. The SRTDD in the internetworking system is an interesting issue. Meanwhile, synchronization control also

needs to be studied in the case of connectionless networks.

The performance modeling of multimedia synchronization systems is an open area. Suitable performance models and solution techniques for the performance evaluation of the multimedia system are highly required.

The class of LBSPNs used in our work has efficient solution algorithms. The weakness of the approach is the restriction on the boundary of this class of SPNs. We have not discovered that LBSPN can be used for the modeling of other multimedia synchronization systems, for instance, the simultaneous data delivery. If the boundary of LBSPNs can be further extended so that this class can be more applicable in performance modeling of real systems, LBSPNs will be very promising analytical performance models.

In this thesis, we were focused on the exact solution methods for the SPN models. Nevertheless, in literature, some approximate solution methods have been proposed [72] [75], both for GSPN models and CGSPN models. In these solution techniques, decomposition methods are applied. If a complex SPN model can either be approximated by a LBSPN model, or can be decomposed "approximately" into several LBSPN models, efficient solution algorithms for the performance modeling of multimedia systems can be obtained.

Bibliography

[1] J. F. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of ACM* Vol.26, No.11, Nov. 1983, pp832-843

[2] M.K.Mehmet Ali, "Packet Voice Performance Models for Long-Haul and Local Networks, and Delays with a CSMA/CD Protocol", Ph.D dessertation, Systems Engineering Dept., Carleton University, Ottawa, Canada, April.1983

[3] G.Barberis, and D.Pazzaglia, "Analysis and Optimal Design of a Packet-Voice Receiver", *IEEE Trans. on Communications*, Vol.Com-28, No.2,Feb.1980. pp.217-227

[4] A.Goscinski, "Distributed Operating Systems: The Logical Design", Addison-Wesley Publishing Company, 1991

[5] L.Orozco-Barbosa, A.Karmouch, N.D.Georganas and M.Goldberg, "A Multimedia Inter-hospital Communications System for Medical Consultations", *IEEE JSAC*, Vol.10, No.7, September 1992, pp1145-1157

[6] M. Chen, Z.Shae et al., "A Multimedia Desktop Collaboration System", *Proc. Globecom'92*, Orlando, U.S.A., Dec.1992

[7] T.N.Chen, J.Walrand, and D.G.Messerschmitt,"Dynamic Priority Protocols for Packet Voice," *IEEE JSAC*, Vol.7, No.5, 1989, pp.632-643

[8] S.Sarin and I.Greif, "Computer-Based Real-Time Conference Systems", *Computer*, Oct. 1985, pp35-45

[9] N.A. Davies and J. R. Nicol, "Technological perspective on Multimedia Computing", *Computer Communications*, vol.15, no.6, July/Aug 1992

[10] D. Ferrari, "Delay jitter control scheme for packet-switching internetworks," *Computer Communications*, vol.15, no.6, July/Aug 1992, pp.367-373

[11] J. Escobar, D. Deutsch, and C. Partridge,"Flow Synchronization Protocol", *Proc. IEEE ICC'92*, Geneva, Switzerland, May. 1993

[12] C.A.Ellis, S.J.Gibbs, and G.I Rein, "Groupware — Some Issues and Experiences", *Communications of The ACM*, Vol.34, No.1, January 1991, pp38-58

[13] C. L. Hamblin, "Instants and Intervals," in *Proc. 1st Conf. Int. Soc. for the Study of Time*, J. T. Fraser et al., Eds. New York: Springer-Verlag

[14] C.F.Goldfarb, "HyTime: A standard for structured Hypermedia Exchange", *IEEE Computer*, vol.24, no.8, Aug. 1991, pp81-84

[15] A. Karmouch, L. Orozco-Barbosa, N. D. Georganas, and M.Goldberg, "A Multimedia Medical Communications System", *IEEE JSAC*, Vol.8, No.3, April 1990, pp325-339

[16] A. Karmouch, "Multimedia Distributed Cooperative System", *Computer Communications*, Vol.16, No.3, Sept. 1993, pp568-580

[17] L. Lamont and N. D. Georganas, "Protocols for real-time multimedia conferencing", *Proc. Can Conf. on Elec. & Comp. Engg.*, Toronto, Sept.1992

[18] L. Lamont, G. Henderson, and N.D.Georganas, "A Multimedia Real-Time Conferencing System: Architecture and Implementation", *Proc. IBM CASCON'93*, Toronto, Oct. 1993

[19] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed Systems", *Communications of the ACM*, Vol. 21, No.7, July 1978, pp.558-565

[20] L.Li, A.Karmough, and N.D.Georganas, "Real-time Synchronization Control in Multimedia Distributed Systems", *Proc. IEEE Multimedia'92*, Monterey, U.S.A. April, 1992

[21] L.Li, A.Karmough, and N.D.Georganas, "Synchronization in Real Time Multimedia Data Delivery", *Proc. IEEE ICC'92*, Chicago, U.S.A, June 1992. pp322.1

[22] L. Li, A. Karmouch and N. D. Georganas, "Real-Time Synchronization Control in Multimedia Distributed Systems" *ACM Computer Communication Review*, July, 1992

[23] L. Li, A. Karmouch and N. D. Georganas, "Performance Modelling of Distributed Data Integration in Real-time Multimedia Systems", *Proc. ICCC (International Council for Computer Communications) Multimedia Communications'93* , Banff, Canada, April, 1993

[24] L. Li, L. Lamount, A. Karmouch and N. D. Georganas,"A Distributed Synchronization Control Scheme in a Group-oriented Multimedia Conferencing System". *Proc. 2nd International Conference on Broadband Islands*, Athens, Greece, June, 1993

[25] L. Li, A. Karmouch and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 — Temporal Modeling of Collaborative Multimedia Scenarios", *ACM Journal of Multimedia Systems* (to appear)

[26] L. Li, A. Karmouch and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 2 — Synchronization Algorithms", *ACM Journal of Multimedia Systems* (to appear)

[27] L. Li, A. Karmouch and N. D. Georganas, "Multimedia Segment Delivery Scheme and Its Performance for Real-time Synchronization Control", submitted for publication, 1993

[28] T.Little and A.Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services", *IEEE JSAC*, vol.9, no.9, Dec.1991

[29] T.Little and A.Ghafoor,"Synchronization and Storage Models for Multimedai Objects", *IEEE JSAC*, Vol.8, No.3, April 1990, pp.413-427

[30] T.Little and A.Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and Communication," *IEEE Network Magazine*, Nov.1990, pp.32-49

[31] T.Little and A.Ghafoor, "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks", *IEEE Network Magazine*, Oct. 1991, pp.42-50

[32] B.Schoner, J.Ruckert, and R.Steinmetz, "Media Type Related Requirements for a Transport Service in a Distributed Multimedia System",*First interational workshop on network and operating system support for digital audio and video*, Nov.8-9, Berkeley, Calif., 1990

[33] C.Nicolaou, "An Architecture for Real-Time Multiemdia Communication Systems",*IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990

[34] P. Hoepner, "Synchronizing the Presentation of Multimedia Objects", *Computer Communications*, vol.15, no.9, 1992

[35] P. V. Ragan, H. M. Vin, and S. Ramanathan, "Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences", *IEEE/ACM Trans. on Networking*, Vol.1, No.1, Feb. 1993, pp.20-47

[36] S. Ramanathan and P. V. Ragan, "Adaptive Feedback Techniques for Synchronization Multimedia Retrieval over Integrated Networks", *IEEE/ACM Trans. on Networking*, Vol.1, No.2, Apr. 1993, pp.246-260

[37] F. Oguet, C.Schwartz, F.Kretz and M.Quere, "RAVI: A proposed Standard for the Interchange of Audio/Visal Interactive Applications", *IEEE JSAC*, Vol. 8, No.3, April 1990

[38] K. Ravindran and B. Prasad, "Communication Structures and Paradigms for Distributed Conferencing Applications", *Proc. of the ICDCS'92*, pp.598-605

[39] M. Salmony and D. Shepherd, "Extending OSI to Support Synchronization Required by Multiemdia Applicaitons",*IBM ENC Tech. Rep.* no.43.8904, April,1989

[40] D. Shepherd, D. Hutchinson, F. Garcia, and G. Coulson, "Protocol support for distributed multimedia applications," *Computer Communications*, vol.15, no.6, July/Aug 1992, pp.359-366

[41] R.Steinmetz, "Synchronization Properties in Multimedia Systems", *IEEE JSAC*, Vol.8, No.3, April 1990. pp.401-412

[42] R.Steinmetz, "Multimedia Synchronization Techniques: Exprience Based on Different System Structures", *Proc. IEEE Multimedia'92*, Monterey, U.S.A. April, 1992

[43] H.M.Vin and P.Venkat Rangan, "Design File Systems for Digital Video and Audio", *13th ACM Symposium on Operating Systems Principles, U.S.A*, Feb.1991

[44] G.M.Woodruff, and R.Kositpaiboon, "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance", *IEEE Journal on Selected Areas in Communications*, Vol.8, No.3, pp.1107-1115, April 1990 pp437-446

[45] M. Cecelia Buchanan and P. T. Zellweger, "Specifying Temporal Behavior in Hypermedia Documents", *Proc. of European Conferencing on Hypermedia'92*, Milan, Italy, Dec.1992

[46] M. Cecelia Buchanan and P. T. Zellweger, "Scheduling Multimedia Documents Using Temporal Constraints", *Proc. of The Third International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, Nov.1992

[47] R. Yavatkar, "MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications", *Proc. of the ICDCS'92*, pp.606-613

[48] M. Li and N. D. Georganas, "Exact Parametric Analysis of Stochastic Petri Nets", *IEEE Trans. on Computers* Vol.41, No.9, Sept. 1992. pp1176-1180

[49] M.Li and N. D. Georganas, "Norton's Theorem in the Analysis of a Class of Stochastic Petri Nets", submitted for publication

[50] M. Li and N. D. Georganas, "Parametric Analysis of Stochastic Petri Nets", *Fifth International Conference on Modeling and Tools for Computer Performance Evaluation*, Torino, Italy, Feb. 1991

[51] M. Li, "Applying Decomposition and Aggregation Theory to The Analysis of Stochastic Petri Nets and Queueing Networks", Ph.D dissertation, Electrical Engineering Dept., University of Ottawa, Ottawa, Canada, Dec.1991

[52] M. Li and N. D. Georganas, "Coloured generalized stochastic Petri nts for integrated system protocol performance modelling," *Computer Communications*, vol.13, no.7, Sept.1990

[53] M. K. Molloy, "Performance Analysis Using Stochastic Petri Nets", *IEEE Trans. on Computers*, Vol.31, No.9, Sept. 1982

[54] M. K. Molloy, "Fundamentals of Performance Modelling", *Macmillan Publishing Co.*, New York, 1989

[55] S. Natkin, "Les Reseaux de Petri Stochastiques et Leur Application a L'Evaluation des Systems Informatiques", These de Docteur, CNAM, Paris, France, 1980

[56] W. M. Zuberek, "Timed Petri Nets and Preliminary Performance Evaluation", *Proc. of the 7th Annual Symposium on Computer Architecture*, 1980

[57] M.A.Marsan, G. Balbo, and G.Conte, "A class of generalized stochastic Petri nets for the performance analysis of multiprocessor system," *ACM Trans. Comput. Syst.* Vol.2, no.2, May 1984

[58] M. A. Marsan, G. Balbo, G.Conte and F.Gregoretti, "Modeling Bus Contention and Memory Interference in a Multiprocessor System", *IEEE Trans. on Computers* Vol.32, No.1, 1983

[59] G. Ciardo and K. S. Trivedi, "A Decomposition Approach for Stochastic Petri Net Models," *The 4th International Workshop of Petri Nets and Performance Models*, Melbourne, Australia, Dec.1991, pp.74-83

[60] Courtois, P. J., "Decomposability: Queueing and Computer Applications," *Academic Press*, New York, 1977

[61] P. A. Jacobson and E. D. Lazowska, "Analyzing queueing networks with simultaneous resource possession", *Communications of the ACM*, 25(2):142-151, Feb.1982

[62] E. Gressier, "A Stochastic Petri Net Model for Ethernet", *International Workshop on Timed Petri Nets*, Torino, Italy, July. 1985

[63] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, vol.77, no.4, Apri. 1989

[64] Zenic, A., "Colored Stochastic Petri Nets", *International Workshop on Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp.262-271

[65] A. A. Lazar and T.G. Robertazzi, "Markovian Petri Net Protocols with Production Form Solution", *Performance Evaluation*, Vol.12, No.1, Jan.1991

[66] A. A. Lazar and T.G. Robertazzi, "Markovian Petri Net Protocols with Production Form Solution", *Proceedings of IEEE INFOCOM'87*, San Francisco, CA., 1987

[67] J. P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers", *Communications ACM*, 16, 1973, pp.325-339

[68] A. E. Conway and N. D. Georganas, "Queueing Networks - Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation", MIT Press, Boston, 1989

[69] H.H.Ammar, Y.F.Huang and R.W.Liu, "Hierarchical Models for Systems Reliability, Maintainability and Availability", *IEEE Trans. on Circuits and Systems*, Vol. CAS-34, No.6, June 1987, pp.629-638

[70] H.H.Ammar,S. M. R. Islam, "Time Scale Decomposition of A Class of Generalized Stochastic Petri Net Models", *IEEE Trans. on Software Eng.*, Vol.15, No.6, June 1989, pp.809-820

[71] Giglmayr. J., "Analysis of Stochastic Petri Nets by the Decomposition of the Transition Rate Matrix", *NTZ Archiv* (Germany), Part1: Vol.9, No.5, May 1987, pp.115-120; Part 2: Vol.9, No.6, June 1987, pp.147-152

[72] Yao. Li, and C. M. Woodside, "Performance Petri Net Analysis of Communication Protocol Software by Delay-Equivalent Aggregation," *Proc. of the Fourth International Workshop of Petri Nets and Performance Models*, Melbourne, Australia, Dec. 2-5, 1991

[73] Yao. Li, "Solution Techniques for Stochastic Petri Nets", Ph.D dissertation, Dept. Syst. & Comput. Eng., Carleton University, Ottawa, 1992

[74] D. Petriu, "Approximate Solution for Stochastic Rendezvous Networks by Markov Chain Task-Directed Aggregation," Ph.D dissertation, Dept. Syst. & Comput. Eng., Carleton University, Ottawa, 1991

[75] G. Chiola, C. Dutheillet, G. Franceschinis and S.Haddad, "On Well-formed Colored Nets and Their Symbolic Reachability Graph", *Proc. 11th International Conference on Application and Theory of Petri Nets*, Paris, France, June. 1990

[76] C. Dutheillet and S.Haddad, "Regular Stochastic Petri Nets", *Proc. 10th International Conference on Application and Theory of Petri Nets*, Bonn, Germany, June. 1989

[77] S.Haddad and J. M. Couvreur,"Towards a general and powerful computation of flows for parameterized colored nets", *Proc. 9th Europ. Workshop on Application and Theory of Petri Nets*, Venezia, Italy, June 1988

[78] C. Lin, C. Marinescu, "Stochastic High Level Petri Nets and Applications", *IEEE Trans. on Computers*, no.7, July. 1988

Appendix

.1 The BNF Style Description for The Scenario Specifications

The "braces" '⌈⌋' have been introduced into BNF to denote none or more repetitions of the enclosed material. Since '<>' is already used in the BNF description, the sequential specification '<>' is written as '()' in the following.

< scenario >::=< start >< single thread > | < start >< multiple thread >

< single thread >::=< operation > ⌈< operation >⌋ < end >

< multiple thread >:: = < single thread >; < joint >< single thread >
⌈; < joint >< single thread >⌋

< joint >::=< operation > | < variable > | < start >

< start >::=< identifier >

< end >::=< identifier >

< operation >:: = {< command list >}
|(< variable >, < variable > ⌈, < variable >⌋)

< command list >::=< command > ⌈, < command >⌋

< command >:: = < variable >< f-relation >< variable >
| < variable >< m-relation >< variable >
⌈< m-relation >< variable >⌋
| <variable>= *ch*(<variable>, <variable> ⌈, <variable>⌋)
| < variable >

< variable >:: = < interval identifier >

|(< interval identifier > (e) < interval identifier >

⌈(e) < interval identifier >⌋)

< f-relation >::= $e|s|d|f|o|oi$

< m-relation >::= $e|d$

.2 The algorithm for the schedule inference

The following variables are used in the algorithm. t_m^{rx}: the schedule made by the source S_r which is the "owner" source of object X; t_m^x: the schedule to be obtained for $\vec{n}_x \in \vec{N}$, $A(\vec{n}_x) = m$; D_m^x: The derived display duration of object X. The initial value of m is $m = 0$. Same are the other time variables.

Infer(m)

Begin

If $REF_m^c \neq \phi$ /* $m \neq 0$ */

Begin

compare the finishing time of $\vec{n}, \vec{n} \in REF_m^c$ to obtain $REF_m = \{\vec{n}_k\}$;

$EQU_m = t_m^{REF}$ = the finishing time of $\vec{n}_k = t_{A(\vec{n}_k)}^k + D_{A(\vec{n}_k)}^k$;

End /* If */

$t_m^{REF} = \max\{t_m^{REF}, t_m^{rf} : \forall \vec{n}_f \in \vec{N}_f^m\}$; /* obtain the schedule for $\forall \vec{n}_f \in \vec{N}_f^m$ */

For every $\vec{n}_i \in \vec{N}_f^m$: $t_m^i = t_m^{REF}$; /* assign the obtained schedule to $\forall \vec{n}_f \in \vec{N}_f^m$ */

While $\exists \vec{n}_x \in \Lambda_m^p$ & $\vec{n}_x \notin \vec{N}_f^m$

Begin

$t_m^x = -1$;

/* If $\tilde{\mathcal{T}}_r(\vec{N}, \vec{n}_x, \{\vec{n}_y\}) \neq \phi$, $\vec{n}_x$ is scheduled this time */

$\{e, s, o, oi\} \cap \tilde{\mathcal{T}}_r(\vec{N}, \vec{n}_x, \{\vec{n}_y\}) \neq \phi$: $t_m^x = t_m^y$;

$\{f\} \cap \tilde{\mathcal{T}}_r(\vec{N}, \vec{n}_x, \{\vec{n}_y\}) \neq \phi$: $t_m^x = t_m^y + D_m^y - D_m^x$; /

$\{d\} \cap \tilde{\mathcal{T}}_r(\vec{N}, \vec{n}_x, \{\vec{n}_y\}) \neq \phi$: $t_m^x = t_m^y + \frac{D_m^y - D_m^x}{2}$; /* $\vec{n}_x(d)\vec{n}_y$ */

If $-1 \neq t_m^x$: $\vec{N}_f^m = \vec{N}_f^m \cup \{\vec{n}_x\}$ /* $\vec{n}_x$ is scheduled */

If $-1 \neq t_m^x < t_m^{rx}$

Begin /* Compare with the schedule chosen by the "owner" source of $\vec{n}_x$ */

$t_m^{REF} = t_m^{REF} + (t_m^{rx} - t_m^x)$;

For every $\vec{n}_i \in \vec{N}_f^m$ /* Adjust the schedules */

$t_m^i = t_m^i + (t_m^{rx} - t_m^x)$;

$t_m^x = t_m^{rx}$;

End /* If */

End /* While */

$\Delta\tau_m = t_m^{REF} - EQU_m$; /* gap */

If $\exists \vec{n}_y \in \Lambda_m^p$ &$(\vec{n}_y.F_s \cup \vec{n}_y.F_e) \cap \vec{N}_s = \phi$

& for every $\vec{n} \in \vec{N}_s$, $\{e, ch\} \cap \tilde{\mathcal{T}}_r(\vec{N}, \vec{n}_y, \{\vec{n}\}) \neq \phi$, $\vec{N}_s \subseteq \Lambda_m^p$

Then for $\forall \vec{n}_x \in \vec{N}_s$ Begin /* Obtain the final display duration */

$D_m^x = min\{D_m^x, t_m^y + D_m^y - t_m^x\}$;

Adjust object $\vec{n}_x$ (may cut $\vec{n}_x$);

End

For every $AC_k = \mathcal{E}(\{AC_m\})$ & $\forall AC_i \in \mathcal{E}^{-1}(\{AC_k\})$ scheduled

Begin

Infer (k)

If $\Delta\tau_k > 0$ Gap$(\Delta\tau_k, m)$; /* Gap filtering */

$t_m^{REF} = t_m^{REF} + \Delta\tau_k$;

$\Delta\tau_m = t_m^{REF} - EQU_m$;

End

Return $(\Delta\tau_m)$; /* return the gap allocated */

End

Gap$(\Delta\tau, m)$

Begin

For $\forall \vec{n}_x \in AC_m$

$t_m^x = t_m^x + \Delta\tau$;

If $\exists AC_j \in \mathcal{E}(\{AC_m\})$ & $j \neq k$ & Infer(j) has been performed

Gap($\Delta\tau, j$);

End

.3 Program List of the Computational Algorithm for LBSPNs

```
#include<stdio.h>
#include<ctype.h>
#define ALLOCSIZE 50000
static char allocbuf[ALLOCSIZE];
static char *allocp=allocbuf;
float prob[100][100];
float result_p[1000], col_p[1000], row_p[200];
struct marking {
                int place;
                int token;
                struct marking *next;
                };
 struct marking *sub[200][200], *initial, *mark1, *mark2;
 struct place {
                int place;
                struct place *next;
              } *common[200][200], *com[200], *mcc;
int num_sub, num_sm[200]; /*number of the subnets contained and number of the ma
rkings in every subnet */
int check[100]; /* The array to record the cp'omon places shared by row and colu
mn */
 struct col_row {
                int place;
                int mk[200]; /* At most 200 markings every subnet */
                } col[200], result_m[200], row[200];
main()
{
 float pass;
 struct place *c1;
 int m,n,i,j,k,l,p,t;
 char c,*alloc(), check[2];
 result_p[0]=-1;
 n=0; m=0;
 num_sub=0; num_sm[n]=0;
 mark1=(struct marking *)alloc(sizeof(*mark1));
 sub[n][0]=mark1;
 mark2=mark1;
 do {
 printf("***** New Subnet***** \n");
 m=0;
 sub[n][0]=mark1;
 mark2=mark1;
 initiation();
 goto back;
 do {
 sub[n][m]=mark1;
 mark2=mark1;
 printf("--- Now Enter A New Marking--- \n");
 printf("Enter a new place which contains at least one token under this marking.
 \n");
 scanf("%d", &p);
 do {
        mark2->next=mark1;
        mark2=mark1;
        mark2->place =p;
        printf("How many tokens reside in the place under the marking\n");
        scanf("%d", &t);
        mark2->token=t;
        printf("Enter a new place which contains at least one token under this m
arking. If no more such places, enter -1 please\n");
        scanf("%d", &p);
        mark1=(struct marking *)alloc(sizeof(*mark1));
    } while(p!=-1);
back: mark2->next=NULL;
printf("Please enter the probability of the marking\n");
scanf("%f",&pass);
```



```
 prob[n][m]=pass;
 m++; num_sm[n]++;
 printf("Some more markings of this subnet(y/n)\n");
 scanf("%s", check);
 } while(check[0]!='n'&&check[0]!='N');
 n++; num_sub++;
 printf("Some more subnets(y/n)?\n");
 scanf("%s", check);
 } while(check[0]!='n'&&check[0]!='N');
 printf("There are %d subnets\n",num_sub);
 for(n=0; n<num_sub; n++)
{
 printf("Subnet %d has %d marking\n",n,num_sm[n]);
 mcc=com[n];
 printf("The common place in subnet %d is ---\n",n);
 do {
  printf("%d,",mcc->place);
  mcc=mcc->next;
    } while (mcc!=NULL);
 printf("\n");
 for (i=1; i<=num_sm[n]; i++)
 {
 mark2=sub[n][i-1];
 do {
 printf("place %d contains %d token", mark2->place, mark2 >token);
 mark2=mark2->next;
   } while (mark2!=NULL);
 printf("\n");
 printf("The probability of marking %d is %f\n", i,prob[n][i-1]);
}
}
printf("The initial markings -----\n");
mark1=initial;
do{ printf(" place %d contains %d token\n", mark1->place,mark1->token);
    mark1=mark1->next;
  } while (mark1!=NULL);
for(i=0; i<num_sub; i++)
 for (j=0; j<num_sub; j++)
   if (i!=j && common[i][j]!=NULL) {
    mcc=common[i][j];
    printf("the common places shared by subnet %d and %d---\n", i, j);
   do  {
    printf("%d,  ", mcc->place);
    mcc=mcc->next;
      } while (mcc!=NULL);
            }
compute();
}


char *alloc(n)
int n;
{
 if (allocp+n<=allocbuf+ALLOCSIZE) {
        allocp+=n;
        return(allocp-n);
        }
        else return(NULL);
}

initiation()
{
 struct marking *init1, *init2, *mco;
 struct place *c1,*c2, *cm1, *cm2;
 int i,j,m,n,l,k,p,t;
printf("First, please enter the initial marking of the subnet\n")
```

```
printf("Enter a place which contains at least one token under the initial markin
g\n");
scanf("%d",&p);
do {
 mark2->next=mark;
 mark2=mark;
 mark2->place=p;
 printf("How many tokens reside in this place under the initial marking?\n");
 scanf("%d", &t);
 mark2->token=t;
 if(initial==NULL) {
        init=initial;
        while(init->place!=p && init->next!=NULL)
                init=init->next;
                }
if(initial==NULL || init->place!=p)
{ init2=(struct marking *)malloc(sizeof(*init2));
  if(initial==NULL) initial=init2;
        else init->next=init2;
        init=init2;
        init->place=p;
        init->token=t;
}  /*  A new initial place */
else {
        c1=(struct place *)malloc(sizeof(*c1)); /* It's an old place */
        n=num_sub; /* So it's an common place */
        if(com[n]==NULL) com[n]=c1;
        else {
                c2=com[n];
                while (c2->next!=NULL) c2=c2->next;
                c2->next=c1;
             }
        c2=c1;
        c2->place=p;
        for(i=0;i<num_sub;i++)  /* Check the subnet i that was entered */
        {
        mco=sub[i][0];  /* Only the initial place is to be checked */
        while(mco->place!=p && mco->next!=NULL)
                mco=mco->next;
        if(mco->place==p)
         {
                if(com[i]!=NULL) {
                c2=com[i];
                while(c2->place!=p && c2->next!=NULL)
                c2=c2->next;
                        }
          if(com[i]==NULL || c2->place!=p) /* there is no place in com[i] */
                {
                c1=(struct place *)malloc(sizeof(*c1));
                if(com[i]==NULL)  com[i]=c1;
                else c2->next=c1;
                c2=c1;
                c2->place=p;
                }   /*Place p is entered into com[i]*/
        for(j=0; j<2; j++)
        if (common[i][n]!=NULL) {
                c2=common[i][n];
                while(c2->place!=p && c2->next!=NULL)
                c2=c2->next;
                        }
         if(common[i][n]==NULL || c2->place!=p)
                {
                c1=(struct place *)malloc(sizeof(*c1));
                if(common[i][n]==NULL)  common[i][n]=c1;
                else c2->next=c1;
                c2=c1;
```



```
                c2->place=p;
                }  /*Place p is entered into com[i] */
        i=i;
        j=n;
        n=i;
   }
 i=n;
}
}
  /* Go to the next subnet */
  }  /* It's an old initial place */
printf("Enter a new place which contains at least one token under the initial ma
rking. If no more such places, enter -1, please \n");
scanf("%d", &p);
mark=(struct marking *)malloc(sizeof(*mark));
} while(p!=-1);
}


compute()
{
int count, consum_t, num_mk, i, j, l, k, m, n, rslt_m, p,q,s1,s;
float g;
struct marking *init;
int recd_row[50], recd_col[50];  /*record the position of the common places that
ed by the row and column*/
rslt_m=num_sm[0];
printf("Now we are in the Compute module\n");
for(count=0; count<num_sub; )
{
printf("The first subnet\n");
num_mk=rslt_m;
if (result_p[0][0]==-1) {
                        tran_row[count];
                        count++;
                        tran_row[count];
                        count++;
                }
        else    {
                        tran_col[num_mk];  /*Get the number of the markings in t
he column */
                        tran_row[count];
                        count++;
                }
i=0;
printf("This is our col\n");
while(col[i].place!=-1)
 {
   printf("Common place %d---\n", col[i].place);
   for (j=0; j<num_mk; j++)
    printf("Marking %d consumes %d token for the common place with prob of %f. \
n", j+1, col[i].mk[j], col_p[j]);
   i++;
}
i=0;
printf("This is our row\n");
   for (j=0; j<num_sm[count-1]; j++)
    printf("Marking %d has prob %f\n", j+1, row_p[j]);
find_chk(count);
i=j=k=m=n=0;
for(i=0; i<num_sm[0]; i++) /*Check the heads of the rows*/
{
 result_p[m++].place=row[i].place; /* enter the place into the head of the resul
t */
 while(row[i].place!=check[j].mk[k] && check[j].mk[k]!=-1)
 if(row[i].place==check[j].mk[k] )  /* It's a common place that is in the row
```

```
    {
        recd_row[k]=i;
        while(col[m].place!=check[j]) m++;
        recd_col[k++]=m;
    }
 j=0; m=0;
} /* All the places in row have been entered */
recd_row[k]=-1; recd_col[k]=-1; /* The end of the array */
printf("Let's see what are in the recd---\n");
k=0;
while(recd_row[k]!=-1)
{
 printf("recd_row[%d]=%d, ", k,recd_row[k]);
 printf("recd_col[%d]=%d\n ", k,recd_col[k]);
 k++;
}
k=0;
for(i=0; (col[i].place!=-1); i++)
  {
   while(recd_col[k]!=-1 && recd_col[k]!=i) k++;
   if(recd_col[k]==-1) {
         printf("k=%d, i=%d, recd_col[k]=%d\n",k,i,recd_col[k]);
         result_m[n++].place=col[i].place; /* It's not a common place, enter it
row */
                       }
   k=0;
  }
result_m[n].place=-1; /* The head of the result is built */
for(pt=0; pt<n; pt++)
printf("result[%d] is %d\n ", pt, result_m[pt].place);
k=i=0; /* Compute the result */
printf("num_sm[count-1]=%d\n", num_sm[count-1]);
for (i=0; i<num_sm[count-1]; i++) /*Go for all the markings in the row */
 for (j=0; j<num_mk; j++) /* All the markings in the column */
   {
   printf("Row j=%d\n", j);
   printf("Row i=%d\n", i);
   s=0;
   while(result_m[s].place!=-1)
     result_m[s++].mk[j]=-1;
    m=recd_row[k];
    n=recd_col[k++];
    if(m==-1) {
        result_p[i]=col_p[j]*row_p[i];
        goto 12;
              }
    else      {
        consum_t=row[m].mk[i] * col[n].mk[j];
        init4=initial;
        while (init4->place!=row[m].place)
                init4=init4->next;
        m=recd_row[k];
        while(m!=-1 && consum_t<=init4->token)
         {
          n=recd_col[k++];
          consum_t=row[m].mk[i] * col[n].mk[j];
          init4=initial;
          while(init4->place!=row[m].place)
                init4=init4->next;
          m=recd_row[k];
         }
        k=0;
        if(consum_t<=init4->token)
        {
         result_p[i]=col_p[j]*row_p[i];
         printf("result_p[%d]=%f\n",i,result_p[i]);
```



```
          m=recd_row[k];
          n=recd_col[k++];
          while (m!=-1)
          {
           consum_t=row[m].mk[i] * col[n].mk[j];
           p=0;
           while(result_m[p].place != row[m].place)
                 p++;
           result_m[p].mk[j]=consum_t;
           m=recd_row[k];
           n=recd_col[k++];
          }
          i=j=0;
          while(row[p].place!=-1)
          {
                 s=0;
                 while(result_m[s].place!=row[p].place)
                 s++;
                 if(result_m[s].mk[j]==-1)
                        result_m[s].mk[j]=row[p].mk[i];
                 p++;
          }
          q=0;
          while(col[q].place!=-1)
          {
                 s=0;
                 while(result_m[s].place!=col[q].place)
                 s++;
                 if(result_m[s].mk[j]==-1)
                        result_m[s].mk[j]=col[q].mk[j];
            q++;
          }
          i++;
          }
                 }
k=0;
   }
reit_m=i;
}
g=0.0;
s=0;
while(result_m[s++].place!=-1)
{
for(j=0; j<i; j++)
printf("Marking %d eats %d token from place %d\n", j+1,result_m[s-1].mk[j],resu
lt_m[s-1].place);
}
for(j=0; j<i; j++)
g=g+result_p[j];
printf("The Normalization Constant G is %f\n",g);
}

tran_row(sub_count)  /* Translate a subnet into a row */
int sub_count;
{
 struct marking *m, *init;
 struct place *c;
 int i, j, k, l, m, n, p, t;
 m=0;
printf("&& tran_row&&\n");
 if(com[sub_count]!=NULL) {
                c=com[sub_count];
                do  {
                        row[m].place=c->place;
                        m++;
                        c=c->next;
```

```
                    } while (ct!=NULL);
                        }  /*build up the head */
row[m].place--;
for(j=0;j<num_sm[sub_count];j++)
{
       if(row[0].place==-1)  row[0].pmk[j]=0;
       else {
         for (i=0;i<m;i++)
         {
          init[i]=init[a];
          while(init[i]->place!=row[i].place && init[i]->next!=NULL)
               init[i]=init[i]->next;
          if(init[i]->place !=row[i].place)
               { printf("Something is wrong here \n");
                 exit(1);
               }
          else   t=init[i]->token;
       row[i].pmk[j]=t; /* The token consumed from the common place */
 mark1=sub[sub_count][j]; /* j-th marking */
 do {
       p=mark1->place;
          if(row[i].place==p)
       row[i].pmk[j]=t-mark1->token;
 mark1=mark1->next;
     } while (mark1!=NULL); /*check the common place token consuptions at the j
-th marking */
           }
     }
row_p[j]=prob[sub_count][j]; /*The probability of the j-th marking */
}  /* The end of *for* */
if (sub_count ==0) {
               for (k=0;k<m; k++)
               col[k]=row[k];
               col[k].place--;
               for(j=0; j<num_sm[0];j++)
                 col_p[j]=row_p[j];
                }
printf("This is subnet %d\n", sub_count);
for (i=0; i<m; i++)
{
  printf("Common place %d---\n", row[i].place);
  for (j=0; j<num_sm[sub_count]; j++)
   printf("Marking %d consumes %d token from the common place,with prob %f\n",
j+1, row[i].pmk[j],row_p[j]);
}
}


/* Translate the result into column */

tran_col(rltm_num)  /* rltm_num records the number of markings contained in the
results */
int rltm_num;
{
 int i,j;
 for (i=0; i<result_m[i].place+1; i++)
 col[i]=result_m[i];
 col[i].place=-1;
 for (j=0; j<rltm_num; j++)
 col_p[j]=result_p[j];
}


find_chk(sub_count)  /* Fix the check array */
int sub_count;
{
```



```
int i,j,m,n;
struct place *cm;
i=sub_count-1;
j=i-1;
printf("In check array--j=%d\n",j);
n=0;
for (m=0; m<=j; m++)  /* Search the subnet 1 to j */
if(common[i][m]!=NULL) {
       cm=common[i][m];
       do {
       check[n++]=cm->place;
       cm=cm->next;
          } while (cm!=NULL);
                    }
check[n]=-1;
printf("This is the check array\n");
n=0;
while(check[n]!=-1) {
                printf("check[%d]=%d,   ", n, check[n]);
                n++;
                    }
}
```