12-1-1993

Transmission Scheduling of Multichannel System for Communication of Pre-orchestrated Multimedia Information

Miae Woo  
*Purdue University School of Electrical Engineering*

Arif Ghafoor  
*Purdue University School of Electrical Engineering*

Follow this and additional works at: [http://docs.lib.purdue.edu/ecetr](http://docs.lib.purdue.edu/ecetr)

[http://docs.lib.purdue.edu/ecetr/256](http://docs.lib.purdue.edu/ecetr/256)

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
TRANSMISSION SCHEDULING OF MULTICHANNEL SYSTEM FOR COMMUNICATION OF PRE-ORCHESTRATED MULTIMEDIA INFORMATION

Miae Woo
Arif Ghafoor

TR-EE 93-49
DECEMBER 1993

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285
Transmission Scheduling of Multichannel System for Communication of Pre-orchestrated Multimedia Information

Miae Woo, Arif Ghafoor,
Distributed Multimedia System Lab
School of Electrical Engineering, Purdue University
West Lafayette, Indiana 47907
Tel. (317) 494-0638
FAX (317) 494-6440
E-mail: ghafoor@ecn.purdue.edu

Abstract

Communication of multimedia information over broadband networks dictates stringent QOS requirements. In this report, we propose a communication model for pre-orchestrated multimedia information. The proposed model, called XOCPN, specifies the performance parameters to handle multimedia media effectively as well as provides resynchronization action to compensate anomalies due to random delays introduced by the network. The model is used to determine the optimal number of channels needed to transmit the multimedia information. Subsequently, we consider a channel-deficient system having homogeneous channels (i.e. channels with identical characteristics) and prove that scheduling transmission of multimedia data in this system is an NP-hard problem. Accordingly, we propose a heuristic algorithm with complexity \(O(n \log nm)\), where \(n\) represents the number of data units to be communicated over \(m\) channels. Subsequently, we consider a more general case of channel-deficient system with heterogeneous channel. Also for this system, a heuristic algorithm with complexity \(O(n \log nm + nk)\) is proposed, where \(k\) is the number of different characteristics among available channels.

Keywords: Multimedia Communication, Pre-orchestrated Multimedia Information, Transmission Scheduling, Channel-deficient System
1 Introduction

The growing demand for distributed multimedia applications has posed a new set of challenges in networking, such as managing of network's resources for guaranteeing Quality Of Service (QOS) [1, 2, 5, 19], designing real-time multimedia synchronization protocols for broadband wide area networks [7, 14, 15, 16], managing multi-party sessions to support collaborative environments, etc. Many such applications [9, 11] can be found in medicine, education, business, manufacturing, computer aided design and computer aided engineering, weather, entertainment, etc.

Most of these applications use some pre-orchestrated, stored multimedia data and precipitate the necessity for management of heterogeneous data with vastly different storage, communication, and presentation requirements. Specially with respect to communication, these data possess drastically different performance and reliability characteristics. For example, audio and video are isochronous in nature, thus requiring a service supporting real-time delivery and fine-grain synchronization. On the other hand, conventional data such as text or graphics possess less stringent end-to-end delivery requirements yet require superior reliability in transmission. However, if conventional data has synthetic timing relationships with other data, as is the case of pre-orchestrated multimedia, it should be delivered on time without error. These requirements can be met by designing a set of synchronization protocols which provide flexible transport mechanisms in an efficient way.

Among the known transport mechanisms used for broadband integrated networks, ATM technology appears to be the most promising one because of its capability to support bursty traffic [1]. In an ATM network, the ATM Adaptation Layer (AAL) provides a capability to support four different classes of service that depend on performance requirements of data. One such requirement is synchronizing both the source and the destination sites while maintaining a minimum bit rate over a selected connection mode [4].

Although ATM can effectively manage communication of broadband data, for a successful support of pre-orchestrated multimedia application in a distributed environment,
maintaining inter-stream synchronization is vital and poses many engineering challenges for an ATM-bitsed network. Inter-stream synchronization mechanisms can be used to satisfy temporal constraints among related data while they are transmitted over Multiple Virtual Channels (MVCs). At the time of transmission, the source must transmit data on MVCs in such a way that the delivery of data at the destination site is "timely" despite the presence of random network delays on MVCs. Therefore, a transmitter schedule is needed for any pre-orchestrated multimedia information. This schedule needs to be generated and make available to the transmitter and the receiver. Therefore, the generation of such a schedule should be a low complexity algorithm.

In order to handle this channel scheduling problem, we have recently proposed in [13] a communication model, known as extended Object Composition Petri-Net (XOCPN), for the pre-orchestrated multimedia information. An XOCPN uses a basic unit of communication data, called Synchronous Interval Unit (SIU). Based on XOCPN, a channel scheduling scheme is proposed in [13] for an ATM network with the assumption that the number of MVCs available are sufficiently large to satisfy the real time delivery requirements for the pre-orchestrated multimedia information [13]. However, this method has a drawback since it results in a wastage of network resources by establishing more than the necessary number of connections. In another approach, a single channel is used for the purpose of transmission. However, this scheme underutilizes network bandwidth and requires extensive buffering at the destination in order to ensure the desired QOS.

In this report, we address two aspects of the channel scheduling problem. First, we describe a mechanism to identify the optimal number of channels needed for the transmission of any pre-orchestrated multimedia information. These channels must meet media-specific requirements, such as throughput, error-ratio, and performance parameters used in specifying AAL classes. The number of channels found is optimal in the sense that they are sufficient to carry out end-to-end communication efficiently and effectively without any degradation in performance.
We then consider the case when the required number of channels are not available. As mentioned earlier, a special case of such system is a one-channel system which has been discussed in [7]. In the one-channel case, all the objects involved in an OCPN are transmitted over a single channel, irrespective to the desired QOS. In a real system, multiple channels can be available which may have either identical characteristics or heterogeneous functionality. In this report, we first consider a homogeneous environment for which all the channels are assumed to have identical characteristics in terms of throughput, end-to-end delay performance, etc. For such a system, we show that transmission scheduling problem over MVCs is NP-hard. Subsequently, we propose a heuristic algorithm with complexity $O(n \log nm)$, where $n$ is the total number of SIUs and $m$ represents the number of available channels. Also we consider a case where available channels have heterogeneous in their characteristic. For such case, we also propose a heuristic algorithm whose complexity is $O(nk^210gn)$, where $k$ is the number of channel characteristics available. Since a channel-deficient system cannot provide the optimal number of channels, buffering would be needed for each channel. Accordingly, we also describe an algorithm to determine the size of buffer needed for each channel.

The organization of this report is as follows: In Section 2, we briefly provide an overview of an XOCPN model. Section 3 describes the method of determining optimal number of channels. In Section 4, we consider channel-deficient systems, and propose heuristic algorithms for transmission scheduling over heterogeneous MVCs as well as homogeneous MVCs. In Section 5, buffer requirement for maintaining synchronization is discussed. Section 6 presents procedures for generating XOCPN for the transmitter as well as for the receiver. Section 7 concludes the report and the appendix provides uniqueness properties of XOCPN and its correctness.
2 A Communication Model for Multimedia Data

In this section, we briefly discuss XOCPN which is a timed, marked Petri-net, and is denoted by $C_{XOCPN}$. As mentioned above, XOCPN represents a communication model for the transmission of pre-orchestrated, stored multimedia information. For a channel-deficient system, we can formally specify this model by the tuple $\{T, P, I, O, Y, C, D, R, M\}$ where:

$T = \{t_1, t_2, \ldots, t_l\}$ is a set of transitions.

$P = \{p_1, p_2, \ldots, p_n\}$ is a set of places. Each place associates with a special action function which is specified by $Y$ as described below.

$I = T \rightarrow P^\infty$ is the input function, a mapping from transition to a set of places.

$O = T \rightarrow P''$ is the output function, a mapping from transition to a set of places.

$Y : P \rightarrow \{\text{connection-setup, connection-release, SIU-playout, SIU-transmit, interstream-synchronize}\}$ is a mapping from the set of places to a set of actions that are performed during the communication of multimedia information. The connection-setup and the connection-release places are used for managing end-to-end virtual circuit connection. The SIU-playout place is used for receiving incoming data and presenting them on the media output device. The SIU-transmit place initiates the transfer of data at the source site. The interstream-synchronize place is used for applying synchronization policies to related streams from MVCs [13].

$C : P \rightarrow \{c_1, c_2, \ldots, c_l\}$ is a mapping from the set of places to a set of channels. Basically, this mapping describes association among actions and the channels being scheduled.

$D : P \rightarrow \{(d_1, d_2)\}$, where $d_1, d_2 \in \mathbb{R}^+$. $d_1$ represents the delay before taking an action and $d_2$ represents the execution duration of the action.

$R : P \rightarrow \{r_1, r_2, \ldots, r_k\}$ is a mapping from the set of places to a set of types of data being transmitted.

$M : P \rightarrow \mathbb{N}, \mathbb{N} = \{0, 1, 2, \ldots\}$ is a mapping from the set of places to the nonnegative integers. These integers represent the number of markings in each place.

This new XOCPN specification is an extension to the original one given in [13]. In this
specification, the mapping function $C$ has been introduced to facilitate unique assignment of places in the net to the channels available in a channel-deficient system. Furthermore, two distinct mapping functions, $I$ and $O$, are used to label input and output arcs, respectively. These two functions are obtained by splitting the set of arcs $A$ in the original definition into a set of output arcs $\{(p_i, t_j) | (p_i, t_j) \in A\}$ and input arcs $\{(t_j, p_i) | (t_j, p_i) \in A\}$ [12]. Such distinction is needed in order to verify the proposed model. Correctness and verification of this model is discussed in the Appendix.

For the purpose of communication, XOCPN-based schedules are needed both at the source and the destination sites, since the transmission and the receiving schedules are different from each other. The schedule at the transmitter site, which is called transmitter-XOCPN, is mainly responsible for the provision of synchronous playback of multimedia object by arranging transmissions of SIUs to meet the specific playout deadlines. On the other hand, the receiver-XOCPN, used by the destination, deals with synchronous playback of data by compensating random delays in the network. The generation of these two XOCPNs is discussed in Sections 3 and 4.

3 Optimal Number of Channels for Communication of Multimedia Data

Since OCPN is a synchronization model for the playout of multimedia information, it cannot be used directly for deriving the transmitter-XOCPN for data transmission by simply dividing objects into appropriate number of SIUs. This is illustrated in Fig. 1 which shows the difference between the timing constraints captured in an OCPN (Fig. 1 (a) and (b)) and transmission timing (Fig. 1 (c)) which represents end-to-end transmission time over a network. In this figure, for the purpose of simplicity, we assume that each object consists of only one SIU. The objects within the multimedia information can be played out synchronously at the receiver only when they are transmitted at appropriate times by the transmitter. Therefore, the transmission start times of these objects are needed to generate
the transmitter-XOCPN.

The simplest way for transmitting multimedia information is to use a separate channel for each object as described in [13]. In this case, we only need to insert the connection-setup place before the first SIU of an object and to insert the connection-release place after the last SIU of the object while generating the transmitter-XOCPN. However this approach may not be realizable, especially when multimedia information consists of a large number of objects. Also this approach can introduce some redundant call setup overhead if objects with the same QOS requirements can share a common channel.

The first consideration for determining the optimal number of channels is that SIUs with the same media type should be assigned to the same channel, provided the duration of their channel occupancy does not overlap. This implies that the minimum number of channels for transmitting SIUs is decided by the maximum concurrency of SIUs within the XOCPN that have the same QOS criteria, provided no buffering is available at the receiver site.

There are number of considerations regarding determining the number of channels needed for an XOCPN. First, the playout deadline for each SIU is needed, which can be determined by finding the playout deadline of an object to which it belongs. The playout deadline $\pi^i$ for object $i$ from an OCPN can be generated by using the serialize-net algorithm given in [7]. The same algorithm can be used at SIU level.

Since different media types require different QOS, the knowledge of the maximum concurrency for each media type within the XOCPN is needed in order to reserve network resources with specified QOS. Therefore, we only need to compare timing information of SIUs which belong to the same media type. For this purpose, after determining the playout deadlines of the objects, the objects are clustered according to their media types, as defined by the function $E_l$ in the OCPN. This clustering can be achieved by the resource-decomposition algorithm given in [7], where cluster of media type $r_i$ is denoted by $\Pi(r_i)$.

After generating playout deadlines and clusters, an object can be divided into several SIUs. This step requires the information about the duration of each SIU, which in turn
Figure 1: Timing comparison for playout and transmission (a) An OCPN (b) Playout time in time chart (c) Transmission time in time chart
depends on the media type of the object. All the media type can be categorized into two groups: $R_I$ and $R_A$, depending upon whether this is an isochronous or anisochronous data, respectively. Here, we choose a common SIU duration for all the objects whose media types are elements of $R_I$. Isochronous objects can be segmented into SIUs with duration $1/30$ seconds which is an unit duration for a video frame. This approach makes it easier to apply further synchronization actions, like inter-stream pacing [13]. For anisochronous data, duration of an object is equal to the duration of its SIU, i.e., anisochronous object consists of only one SIU.

Using SIU duration of each object, we can find the number of SIUs for each object as follows:

$$\text{# of SIUs} = \left\lfloor \frac{\text{duration of the object}}{\text{SIU duration of the object}} \right\rfloor \quad (1)$$

Let $SIU^r_i$ be the $i$th SIU which belongs to isochronous object $r$, and let $\tau$ be the duration of SIU belonging to object $r$. Then the playout deadline for $SIU^r_i$, $\pi^r_i$, is given as:

$$\pi^r_i = \pi^r + \tau \cdot (i - 1)$$

The durations of all the SIUs of the object $r$ are same, $\tau$, and they inherit resource type of object $r$.

Next, we calculate the duration of channel occupation of an $SIU^r_i$ which can be represented by the tuple $[\gamma^r_i, \pi^r_i]$, where $\gamma^r_i$ represents the transmitting time of $SIU^r_i$. In order to find $[\gamma^r_i, \pi^r_i]$, we need to derive timing information of each SIU by considering network behavior as well as the performance requirements specified by the associated QOS. Generally, the duration of channel occupation of an SIU can be represented by the end-to-end delay. To simplify the analysis, we assume that the underlying network is perfectly reliable in the sense that no error control is needed and some kind of facility is presented to carry out end-to-end acknowledgement and retransmission. This assumption is reasonable, as the typical bit error rate of fiber-based high speed networks is $10^{-9}$ or less [3].
Generally, an end-to-end delay consists of three parts: transmission delay $\epsilon(t)$, jitter delay $\epsilon(j)$, and propagation delay $\epsilon(p)$. The propagation delay is generally proportional to the distance between the source and the destination. Thus it is constant for all SIUs. On the other hand, the transmission delay and the jitter delay are proportional to the size of the SIU. Thus the total end-to-end delay of $\text{SIU}^r_i$, $\epsilon^r_i$, can be represented as follows:

$$\epsilon^r_i = \epsilon(p) + \epsilon(t)_i^r + \epsilon(j)_i^r$$

This equation can be used to determine $\gamma^r_i$. In order to meet the playout deadline at the receiver site, an object should be transmitted ahead of the playout deadline. In other words,

$$\gamma^r_i = \pi^r_i - \epsilon^r_i.$$ 

Based on this analysis, we can find the minimum number of channels for transferring SIUs with the same QOS by counting the number of concurrent SIUs. Two SIUs are concurrent if their durations of channel occupation overlap. Here, we assume that there is a total of $n$ SIUs. In the cluster of media type $r_s$, there are $n(r_s)$ SIUs such that

$$\sum_{s=1}^{k} n(r_s) = n.$$ 

The maximum-concurrency algorithm given below determines the amount of concurrency for media type $r_s$. In the algorithm, we use $\text{SIU}(i)$ for denoting an SIU that has the i-th earliest channel occupation time among all the SIUs in the cluster for $r_s$.

**maximum-concurrency** algorithm:

- Sort $\text{SIU}^r_i$'s in nondecreasing order of $\gamma^r_i$, for all $\overline{\text{SIU}}^r_i \in r_r$.

max_concurrency($r_s$) := 0 /* initialize maximum concurrency for media type $r_s$ */

for $i := 1$ to $n(r_s)$ do

    Put a token in $\text{SIU}(i)$
    concurrency := 1

    for $j := 1$ to $i - 1$ do

        /*...*/
if SIU\(_{(j)}\) has token then
  if \(\pi_{(j)} > \gamma_{(j)}\) then /* SIU\(_{(i)}\) and SIU\(_{(j)}\) are concurrent */
    concurrency := concurrency + 1
  else
    Remove token from SIU\(_{(j)}\)
  end
end

max_concurrency(r\(_{s}\)) := max(max_concurrency(r\(_{s}\)), concurrency)

The maximum concurrency for media type \(r\(_{s}\)\) is denoted by max_concurrency\((r\(_{s}\))\) in the algorithm. For each media type, the system needs to provide as many channels as given by this factor for its type since the media type dictates the throughput and end-to-end performance characteristics for a channel. The total number of required channels can be found by summing up all the max_concurrency\((r\(_{i}\))\), for all \(r\(_{i}\) \in R\). This sum is the optimal number of channels needed.

4 Transmission Scheduling for a Channel-deficient System

For a channel-deficient system, some SIUs need to be rearranged since concurrent SIUs cannot be transferred simultaneously. Such a situation requires extra buffering space and time at the receiver site in order to meet the presentation specification. In this case, the transmitter schedule should try to minimize resource usage at the receiver site.

In such situation, it is desired to start playout SIUs at the receiver as early as possible while maintaining temporal relations among SIUs. Let \(D_{i}\) be the delivery deadline of the \(i\)th SIU when sufficient number of channels are available. Thus \(D_{i}\) represents the ideal delivery time of the \(i\)th SIU from the transmitter's point of view. Let \(A_{i}\) be the arrival time of the \(i\)th SIU at the receiver according to a finalized schedule of transmission. We define tardiness of the \(i\)th SIU, \(T_{i}\), as

\[
T_{i} = \max(0, A_{i} - D_{i}).
\]
$T_i$ represents latency in delivery of the $i$th SIU with respect to the ideal delivery time, and can be used for finding an optimal schedule for transmission time as described in the following proposition.

**Proposition 1** The earliest playout start time is equal to $T_{max} + D_{min}$, where $T_{max} = \max_{1 \leq j \leq n} \{T_i\}$ and $D_{min} = \min_{1 \leq j \leq n} \{D_i\}$.

**Proof:**

Let the playout start time be $D_{min} + T_{arb}$. Suppose $T_{arb} < T_{max}$. If $T_i > T_{arb}$ for the $i$th SIU, it cannot arrive the destination at $D_i + T_{arb}$. In this case, there will be a gap of $T_i - T_{arb}$ before the playout of that SIU, which violates the presentation temporal constraints imposed on the SIU. By choosing $T_{arb} = T_{max}$, we can assure synchronous playback without introducing a gap between two consecutive SIUs.

Our objective is to minimize $T_{max} + D_{min}$ to get a minimum playout latency. Since $D_{min}$ is given by the channel throughput and the jitter characteristic of the network, minimizing $T_{max}$ results in the optimal schedule of transmission.

### 4.1 Scheduling of Transmission with a Limited Number of Homogeneous Channels;

First, we address the problem with restricted a number of channels having identical characteristics. In other words, the channels available over the network are homogeneous in the sense that they have the same capacity and provide the same QOS. In this case, all SIUs in an XOCPN need to be communicated over identical channels, irrespective of the QOS associated with various SIUs.

Since we are dealing with homogeneous channels, the end-to-end delay needs to be recalculated for SIUs having channel requirements which are different from those of available ones. Assuming that $\gamma_i$ is recalculated value based on the available channels, let $\beta_i$ be the
transmission start time. Then,

\[ \beta_i = \gamma_i - \gamma_{\text{min}} \]  
\[ D_i = \pi_i - \gamma_{\text{min}} \]

(2) (3)

In that case, instead of \([\gamma_i, \pi_i]\), we can use the tuple \([\beta_i, D_i]\) which represents newly computed interval for the duration of channel occupation of the \(i\)th SIU in a channel-deficient system.

For the assignment of SIUs to the channels, the objective, minimization of \(T_{\text{max}}\), can be achieved by keeping the precedence relation among the arrived SIUs be consistent with the precedence relation of the delivery deadlines. Let's say \(\text{SIU}_i \prec \text{SIU}_j\) if the delivery deadline of \(\text{SIU}_i\) is earlier than that of \(\text{SIU}_j\) \((D_i < D_j)\). If \(\text{SIU}_i \prec \text{SIU}_j\), then the arrival time of \(\text{SIU}_i\) should be earlier than that of \(\text{SIU}_j\) \((A_i < A_j)\) to achieve less tardiness. However this constraint does not necessarily mean that the transmission time of \(\text{SIU}_i\) has to be scheduled earlier than the transmission time of \(\text{SIU}_j\) when multiple channels are available. In terms of precedence preservation, concurrent SIUs can be exceptions if network resources are limited for delivering such SIUs simultaneously. For exploring channels and SIUs relations with precedence constraint, it is effective to consider \(-D_i\)'s, \(1 \leq i \leq n\), as a criteria of ordering SIUs.

Now we formulate transmission scheduling problem for multichannels as follows:

\[ (P1) \text{ Given a set } S \text{ of } n \text{ SIUs, a partial order on } S, \text{ and a number of channels } m, \text{ does there exist a function } f \text{ from } S \text{ to time } t \text{ such that} \]

(i) if \(\text{SIU}_i \prec \text{SIU}_j\) then \(-f(\text{SIU}_j) < -f(\text{SIU}_i)\), and

(ii) for any time \(t\), there are at most \(m\) values of SIU for which \(-f(\text{SIU}) \leq -t \leq -f(\text{SIU}) + c(\text{SIU})\)?

In \((P1)\), function \(f\) denote mapping SIJ to its arrival time, i.e. \(-f(\text{SIU}_i) = -A_i\), and \(c(\text{SIU})\) denotes end-to-end delay of SIJ. Transmission scheduling problem for multichannels \((P1)\) is very similar to multiprocessor task scheduling problems, whose practical instances
have been found to be NP-complete in [17, 18]. In [18], following problem is introduced and showed to be NP-complete.

(P2) Given a set $S$ of $n$ jobs, a relation $\prec$ on $S$, a time limit $t$, and a sequence of integers $c_0, c_1, \ldots, c_{t-1}$, where $\sum_{i=0}^{t-1} c_i = n$, does there exist a function $f$ from $S$ to $\{0, 1, \ldots, t - 1\}$ such that

(i) $f^{-1}(i)$ has exactly $c_i$ members, and

(ii) if $J \prec J'$, then $f(J) \prec f(J')$?

We now show that transmission scheduling problem over MVCs is NP-hard by showing that (P2) is polynomial-time reducible to (P1).

**Theorem 1** Scheduling of transmission time of SIUs over the $m$ homogeneous channels is an NP-hard problem.

**Proof:**

For the mapping of an instance of (P2) to an instance of (P1), we view a job in (P2) as an SIU in (P1), and execution time of a job in (P2) as end-to-end delay of SIU, $\epsilon(\text{SIU})$, in (P1). Given an instance of (P2), by simply choosing an unit time for all $\epsilon(\text{SIU})$'s, it is easy to see that (P2) is just a special case in terms of execution time. For the condition (ii) in (P1), we can just apply same argument given in [18], which is restated here. Let's introduce new jobs $I_{ij}$ for $0 \leq i < t$ and $0 \leq j \leq n - c_i$. Let the old jobs be related by $\prec$ as before and let $I_{ij} \prec I_{i+1,j}$ for $0 \leq i < t - 1$ and arbitrary $j$ and $m$. If we choose $n + 1$ channels and time limit $t$, we have an instance of (P1). Since in any solution, exactly $n + 1 - c_i$ of new jobs must be executed at the $i$th time unit, the instance of (P1) will have a solution if and only if the instance of (P2) does.

Clearly the time needed to construct the instance of (P1) is at most quadratic in the length of the representation for the instances of (P2). Thus (P2) has been polynomially transformed to (P1). \(\square\)
Because of the high complexity of the problem, we propose a greedy heuristic for the assignment of SIUs for a channel-deficient system. The proposed algorithm searches for an unoccupied channel during the interval $[\beta_i, D_i]$, whenever it needs to assign the $i$th SIU to one of the channels. If it cannot find a channel within this interval, it chooses a channel which gives the least latency $T_i$ for the $i$th SIU. For this purpose, we consider delivery deadline $D_i$ as a criteria of ordering SIUs. In order to preserve precedence relations among SIUs, they are sorted in a nondecreasing order of delivery deadlines and are assigned in that order. For assignment, we choose a backward assignment scheme. In this scheme, the SIU with the latest delivery deadline is scheduled first on a free channel, which corresponds to the problem formulation done in (P1). Let's assume a channel $c_i$ is idle up to $a_i$. Formally, the heuristic algorithm, called the SIU-homogeneous-channel assignment algorithm, is described below. In this algorithm, $S_i$ represents scheduled transmission time for the $i$th SIU, and $m$ represents the total number of available channels. As the result of running of this algorithm, $m$ schedule lists are obtained, each of list holds a set of scheduled SIUs for each channel.

**SIU-homogeneous-channel assignment algorithm:**

Sort $n$ SIUs in a nondecreasing order of $D_i$.

for $i := 1$ to $m$ do

\[ a_i := \infty \text{ /* initialize channel idle time for all channels */} \]

Initialize schedule list for channel $c_i$ to be NULL.

end

Construct a priority queue $PQ$ with $a_i$'s.

for $i := 0$ to $n - 1$ do

\[ a_i := \text{delete-max}(PQ) \text{ /* } a_i \text{ is the max. idle time */} \]

Identify channel $c_i$ for $a_i$.

Add the $(n - i)$th SIU to the schedule list of channel $c_i$.

if $a_i \geq D_{n-i}$ then /* channel $c_i$ is idle at delivery deadline of the $(n - i)$th SIU */

\[ A_{n-i} := D_{n-i} \text{ /* arrival time = delivery deadline */} \]
\[ S_{n-i} := \beta_{n-i} \text{ /* scheduled transmission time = transmission starting time */} \]
\[ a_i := S_{n-i} + \epsilon(p) \]

15
else /* there is no idle channel */
    \[ A_{n-i} := a_r \] /* arrival time = max. channel idle time */
    \[ S_{n-i} := \alpha_c - (D_{n-i} - \beta_{n-i}) \] /* transmission time should be scheduled earlier than 
    because of overlap with the other SIU */
    \[ c \] := \[ S_{n-i} + \epsilon(y) \]
end
insert(PQ, \[ A_r \])

The complexity for sorting \( n \) SIUs is \( O(n \log n) \). Construction of priority queue can be done in \( \theta(1) \) time in this case, since the value of \( \alpha_i \)'s are same initially. The complexity for generating a schedule list for all channels is \( O(n \log m) \). Thus, the complexity of this algorithm is \( O(n \log nm) \) for \( m \geq 2 \). If only one channel is available, the \textit{SIU-homogeneous-channel assignment algorithm} becomes the \textit{earliest deadline} first scheduling policy which is the \textit{optimal} solution for that case. Also its complexity reduces to \( O(n \log n) \) which is incurred for sorting \( n \) SIUs.

As a result of the \textit{SIU-homogeneous-channel assignment algorithm}, \( m \) different schedules are produced for \( m \) channels. Let \( n(c_i) \) represent the number of SIUs assigned to the channel \( c_i \). Then, clearly

\[
\sum_{i=1}^{m} n(c_i) = n.
\]

Within a channel, SIUs maintain their precedence relations with timing information, such as arrival time and scheduled transmission time, which are derived from the \textit{SIU-homogeneous-channel assignment algorithm}. However, it can be noticed that there are no temporal relations among SIUs assigned to the different channels.

### 4.2 Scheduling of Transmission over a Limited Number of Heterogeneous Channels

Here, we consider another case for a channel-deficient system, where the available channels are limited and their QoSs are not same. Let \( \Pi(r_i) \) be a set of SIUs whose media type is \( r_i \).

\[
\Pi(r_i) = \{ \text{SIU}_j^i \mid R(\text{SIU}_j^i) = r_i, \ 1 \leq j \leq n(r_i) \}
\]
Also let \( \Gamma_i \) denote a set of the available channels whose QoS matches to the QoS requirement by media type \( r_i \):

\[
\Gamma_i = \{ c_j^i \mid QOS(c_j^i) = QOS(r_i), \ 1 \leq j \leq n(\Gamma_i) \},
\]

where \( n(\Gamma_i) \) represents the number of channels in \( \Gamma_i \). In a channel-deficient system with the total \( m \) channels,

\[
0 \leq n(\Gamma_i) \leq \text{max}\_\text{concurrency}(r_i)
\]

and

\[
\sum_{i=1}^{k} n(\Gamma_i) = m < \sum_{i=1}^{k} \text{max}\_\text{concurrency}(r_i).
\]

If \( n(\Gamma_i) \neq 0 \), for all \( 1 \leq i \leq k \), and \( n(\Gamma_i) < \text{max}\_\text{concurrency}(r_i) \) for some \( i \)'s, this is the case where the channels are available for all media types but they are not enough for supporting transmission of concurrent objects. In this case, SIUs in \( \Pi(\Gamma_i) \) can be transferred via the channels in \( \Gamma_i \) with support of the required QoS but with possible latency in delivery. For each \( \Pi(\Gamma_i) \), transmission schedule can be obtained by running the SIU-homogeneous-channel assignment algorithm given in Section 4.1 on the channels in \( \Gamma_i \) and SIUs in \( \Pi(\Gamma_i) \).

However, if there are some empty \( \Gamma_i \)'s, i.e. \( n(\Gamma_i) = 0 \), decision has to be made what is the best way to transfer SIUs in \( \Pi(\Gamma_i) \). In such a case, we need to recluster SIUs into the available channel types. One natural way for recluster is grouping \( \Pi(\Gamma_i) \)'s based on whether media type of \( r_i \) is isochronous or anisochronous. In other words, if \( r_i \in R_I \) and \( \Gamma_i = \emptyset \), then move SIUs in \( \Pi(\Gamma_i) \) into \( \Pi(\Gamma_j) \) where \( r_j \in R_I \) and \( \Gamma_j \neq \emptyset \). Similarly, if \( r_i \in R_A \) and \( \Gamma_i = \emptyset \), then regroup SIUs in \( \Pi(\Gamma_i) \) into \( \Pi(\Gamma_j) \) where \( r_j \in R_A \) and \( \Gamma_j \neq \emptyset \). For instance, if the channels for audio media are not available and the channels for video media are available, then object type of audio may use the channels for video media. Same argument can be applied to the objects type of image and text.

However, since various QOS are available, the more effective way for transferring SIUs in \( \Pi(\Gamma_i) \) where \( \Gamma_i = \emptyset \) is by using the channels whose QoS parameters match most closely to
the unavailable ones. For $\Gamma_i$'s, it is possible to set a performance level. Performance level can be ranked based on the guaranteed QOS by the channels in $\Gamma_i$'s. QOS can be divided by two categories: QOS for isochronous data, and QOS for anisochronous data. These categories are identifiable by the maximum jitter delay. Since stringent end-to-end delivery is crucial for isochronous data, QOS requirement for media type $r_j$ is said to be superior to the QOS requirement for media type $r_i$, if $r_j$ requires less maximum jitter delay than $r_i$. Within such a category, QOS requirement for media type $r_j$ said to be superior to the QOS requirement for media type $r_i$, if $r_j$ requires higher throughput than $r_i$. Accordingly, $QOS(r_j)$ is referred to be superior to $QOS(r_i)$, which is denoted by $QOS(r_j) \supset QOS(r_i)$, if

1. maximum jitter delay requirement of media type $r_j$ is less than that of $r_i$

2. the throughput requirement of $r_j$ is higher than the throughput requirement of $r_i$ if maximum jitter delay requirements of media, type $r_i$ and $r_j$ are same.

For example, we can say $QOS(r_{video}) \supset QOS(r_{audio}) \supset QOS(r_{image}) \supset QOS(r_{text})$. Since the channels in $\Gamma_j$ provide same QOS, above discussion can be extended to $\Gamma_j$'s. Thus we define $\Gamma_j$ is superior to $\Gamma_i$ if $QOS(e_i^l) \supset QOS(e_j^l)$, where $e_i^l \in \Gamma_i$ for $l = i, j$, and $k = r, s$.

Using the superiority of $\Gamma_i$'s, we can effectively recluster $\Pi(r_i)$'s. In order to satisfy end-to-end delivery requirement, it is desirable for isochronous data to transfer via the channels for isochronous data. But anisochronous data can use the channels either for isochronous or for anisochronous. Thus when $\Gamma_i$ is empty, $\Pi(r_i)$ can be merged to $\Pi(r_j)$ if $QOS(r_i) \supset QOS(r_j)$.

Let $\Pi(\Gamma_i)$ be a set of SIUs whose QOS requirements can be effectively met by the channels belong to $\Gamma_i$. For each $\Pi(\Gamma_i)$, we again run SIU-homogeneous-channel assignment algorithm for scheduling.

Previously discussed methods may give large latency if some channels are relatively overloaded to the other channels. This can happen when throughput of the channels in $\Gamma_i$ is low, or too many SIUs are assigned to $\Pi(\Gamma_i)$.
To avoid such situation, some strategies are needed for load balancing among the available channels. For example, SIUs type of text can use a channel for object type of video if video channel is underloaded.

One consideration for load balancing among the channels is to make transmission duration for all the channels be approximately same. The exact transmission duration for each channel can be obtained only when the finalized schedule is available. However, iterative modification of transmission schedule for a channel poses heavy computational load on the system, which is definitely undesirable. In order to lessen computational complexity, we estimate transmission duration by the best performance case. Best performance of a channel can be achieved when a channel is fully utilized without having any idle time. Thus, by considering the best performance case, we underestimate the transmission duration.

Let \(|c_j^i|\) be a throughput of a channel \(c_j^i\). For the best performance case, average transmission duration for all the available channels is

\[
t_{avg} = \sum_{i=1}^{k} \frac{|SIU_i|}{\sum_{j=1}^{m(i)} |c_j^i|} \cdot \left( H_i + I_i \right),
\]

where \(H_i\) and \(I_i\) are the size of header (protocol overhead) and the size of information field, respectively, of packet for media type \(r_i\). Also, \(|SIU^i|\) represents the size of SIU\(_i\). Similarly, transmission duration for \(\Gamma_i\) for the best performance case is

\[
t_{\Gamma_i} = \sum_{\text{SIU}_j \in \Pi(\Gamma_i)} \frac{|SIU_j|}{\sum_{j=1}^{m(i)} |c_j^i|} \cdot \left( H_i + I_i \right)
\]

If \(t_{\Gamma_i} < t_{avg}\), then the channels in \(\Gamma_i\) are underloaded, and they can carry more data up to

\[
C_{\text{avail}}^i = (t_{avg} - t_{\Gamma_i}) \cdot \left( \sum_{j=1}^{m(i)} |c_j^i| \right) \cdot \frac{I_i}{H_i + I_i}.
\]

If \(t_{\Gamma_i} > t_{avg}\), then the channels in \(\Gamma_i\) are overloaded, and SIUs in \(\Pi(\Gamma_i)\) may be transmitted via the channels in \(\Gamma_j\)'s by amount \(-C_{\text{avail}}^i\) if \(C_{\text{avail}}^j > 0\) and \(\Gamma_j\) is superior to \(\Gamma_i\).

SIU-heterogeneous-channel algorithm is given below, which is used for scheduling SIUs for \(k'\) different types of \(m\) channels. \(k'\) is always less than or equal to the number of the media type, \(k\).
**SIU-heterogeneous-channel** algorithm:

for $i := 1$ to $k'$ do
  Sort SIUs in $\Pi(\Gamma_i)$ in a nonincreasing order of $D_i$'s.
  for $j := 1$ to $n(\Gamma_i)$ do
    $\alpha^i_j := \infty$
    Initialize schedule list for the channel $c^i_j$ to be NULL.
  end
  Construct a priority queue $PQ(i)$ with $\alpha^i_j$'s.
end
while $\Pi(\Gamma_i)$'s are not empty do
  Select SIU $^j_r$ with maximum delivery deadline $D^r_r$.
  $\alpha^r := \max(PQ(r))$
  found := false
  if $(t^r_r > t_{avg})$ and $(C^r_{\text{avail}} < 0)$ then
    for all $\Gamma_u, (\Gamma_u \supseteq \Gamma_r)$ do
      if $(C^r_{\text{avg}} > 0)$ and $(\max(PQ(u))) > \alpha^r$ then
        found := true
        $\alpha^u := \text{delete\_max}(PQ(u))$
        Assign SIU $^j_r$ to the channel $c^u_r$, where $\alpha^u = \alpha^r$
        $\Pi(\Gamma_r) := \Pi(\Gamma_r) - \{\text{SIU}^r_r\}$ /* remove SIU $^j_r$ from SIU list $\Pi(\Gamma_r)$ */
        $A^r_r := \min(\alpha^u, D^u_r)$
        Calculate end-to-end delay of SIU $^r_r$, $\epsilon(\text{SIU}^r_r)$, on the channel $c^u_r$
        $S^r_r := A^r_r - \epsilon(\text{SIU}^r_r)$
        $\alpha^r := S^r_r + \epsilon(p)$
        insert(PQ(u), $\alpha^u$)
        $C^u_{\text{avail}} := C^u_{\text{avg}} - |\text{SIU}^r_r|$
        $C^u_{\text{avg}} := C^u_{\text{avg}} + |\text{SIU}^r_r|$
      end
    end
    if (found) then
      break
    end
  end
  if (!found) then /* $(t^r_r < t_{avg})$ or $(C^r_{\text{avg}} > 0)$ or there is no other channel available */
    Assign SIU $^j_r$ to the channel $c^r_r$, where $\alpha^r := \alpha^r$
    $\Pi(\Gamma_r) := \Pi(\Gamma_r) - \{\text{SIU}^r_r\}$
    delete\_max(PQ(r))
    $A^r_r := \min(\alpha^r, D^r_r)$
    Calculate end-to-end delay of SIU $^r_r$, $\epsilon(\text{SIU}^r_r)$, on the channel $c^r_r$
    $S^r_r := A^r_r - \epsilon(\text{SIU}^r_r)$
    $\alpha^r := S^r_r + \epsilon(p)$
  end
end
Sorting of SIUs in each $\Pi(T_i)$ has $O(n \log n)$ in complexity. If $\text{SIU}_j^r$ is assigned to a channel in $\Gamma_u, r \neq u$, then the corresponding procedure requires $O(k + \log m)$. If $\text{SIU}_j^r$ is assigned to a channel in $\Gamma_r$, then the corresponding procedure requires $O(\log m)$. Thus, assignment of all the sorted SIUs to appropriate channels has $O(nk + n \log m)$ in complexity. As a result, the complexity of $\text{SIU-heterogeneous-channel assignment}$ algorithm is $O(n \log nm + nk)$.

5 Estimating the Buffer Requirement

In a channel-deficient system, there is latency for the presentation of information as described in Proposition 1. Thus, the receiver requires some buffering capability in order to compensate latency and to maintain the presentation temporal relations among SIUs during the playout. Let $S_{\text{min}} = \min_{1 \leq i \leq n} \{S_i\}$, where $S_i$'s are the values calculated in the $\text{SIU-homogeneous-channel assignment}$ or the $\text{SIU-heterogeneous-channel assignment}$ algorithm. Then, the maximum tardiness that results due to channel-deficiency is given as follows:

$$T_{\text{max}} = \max_{1 \leq i \leq n} \{\max(0, A_i - S_{\text{min}} - D_i)\}$$

$T_{\text{max}}$ and the arrival time of an SIU, $A_i$, can be used to estimate the buffer requirement for channel $c_i$ at time $t$. Let $\text{SIU}^r_{(j)}$ denote the $j$th SIU in a schedule for channel $c_i$, with size $|\text{SIU}^r_{(j)}|$ and the delivery deadline $D^r_{(j)}$. Then, the buffer size is given by the following theorem.

**Theorem 2** The buffer size, $B^i(t)$, for channel $c_i$ at time $t$ is:

$$B^i(t) = \sum_{j=1}^{n(c_i)} 1_{[A^r_{(j)} - S_{\text{min}}, D^r_{(j)} + T_{\text{max}}]}(t) |\text{SIU}^r_{(j)}|$$  \hspace{1cm} (5)
Proof:

According to the schedule, SIU\textsubscript{(j)}\textsuperscript{ci} arrives at \( A_{(j)}^{ci} - S_{min} \) and is played out at \( D_{(j)}^{ci} + T_{max} \) at the receiver. Thus, if SIU\textsubscript{(j)}\textsuperscript{ci} arrives before its playout time then it must be buffered for \( D_{(j)}^{ci} + T_{max} - ( A - S_{min} ) \). At particular time \( t \), buffer is needed for all the SIUs which have already arrived but not played out yet, as is described in Eq. 5. □

Notice that \( B(t) \) increases at each \( A_{(j)}^{ci} - S_{min} \) and decreases at each \( D_{(j)}^{ci} + T_{max} \). Thus, the maximum buffer requirement for channel \( c_i \) can be estimated by evaluating \( B(t) \) at \( t = A_{(j)}^{ci} - S_{min} \), for all \( 1 \leq j \leq n(c_i) \). In the following, the maximum-buffer-estimation algorithm is given which yields the buffer requirement for channel \( c_i \). The estimated buffer size is used by the receiver so that appropriate memory be allocated before starting data transfer and be used for buffering SIUs.

\begin{verbatim}
maximum-buffer-estimation algorithm:

\( B_{max} := 0 \) /* initialize the max. buffer size needed for channel \( c_i \) */

for \( k := 1 \) to \( n(c_i) \) do

\( B := |SIU_{(k)}^{ci}| \) /* set buffer size to the size of SIU_{(k)}^{ci} */

for \( j := 1 \) to \( k - 1 \)

if \( D_{(j)}^{ci} + T_{max} > A_{(j)}^{ci} - S_{min} \) then

\( B := B + |SIU_{(j)}^{ci}| \)

end

\( B_{max} := max(B_{max}, B) \)

end
\end{verbatim}

6 XOCPN Generation

6.1 Generation of the Transmitter-XOCPN

The analysis in Section 4 can be used to generate the transmitter-XOCPN. This XOCPN has as many threads as the number of channels opened between the initial and the final places
in the XOCPN. These threads correspond to the schedule lists of SIUs which are generated by the SIU-homogeneous-channel assignment or the SIU-heterogeneous-channel assignment algorithm.

For each thread in the XOCPN, the connection-setup place is added as an output place of the initial transition for the connection establishment between the transmitter and the receiver sites with the desired QOS. Subsequently, the SIU-transmit places are: added according the sequence of SIUs within a channel. The complete duration of each SIU-transmit place is specified by \(d_1 + d_2\), where \(d_1\) is an artificial delay that is introduced in order to take into account the propagation delay and the jitter delay of the preceding SIU, so that the transmission start time for each SIU can be maintained properly. \(d_2\) represents the transmission delay for the SIU-transmit place since an SIU occupies the output port of the transmitter system only for that amount time. Consequently, \(d_2\) of the \(i\)th SIU-transmit place and \(d_1\) of the \((i + 1)\)th SIU-transmit place altogether account for the end-to-end delay of the \(i\)th SIU.

The earliest possible transmission time for the \(i\)th SIU is \(S_i - S_{\text{min}}\). Therefore, for the first SIU in each thread, a delay of \(S_{(1)} - S_{\text{min}}\) is needed before starting the transmission of that SIU. For the subsequent \(i\)th SIU, the delay of \((S_{(i)} - S_{\text{min}}) - (S_{(i-1)} - S_{\text{min}} + \epsilon_{(i-1)}(t))\) is required as depicted in Fig. 2.

Based on this discussion, the transmitter-XOCPN can be generated using the following algorithm. Here, \(\text{SIU}_{(j)}^{c_i}\) represents the \(j\)th SIU assigned to channel \(c_i\).

\[
\text{transmitter-XOCPN generation algorithm:} \\
P := \emptyset \text{ /* initialize the set of places */}
\]
\[ T := \emptyset \] /* initialize the set of transitions */

Add initial place \( p_{\text{initial}} \) to \( P \)

Add initial transition \( t_{\text{initial}} \) to \( T \)

\begin{verbatim}
for i := 1 to m do /* repeat for all channels */
    Add the connection-setup places for channel \( c_i \); as outputs of initial transition according to the number of available channels.
    Add transition with input place being the connection-setup place for channel \( c_i \); for all SIUs assigned to the channel \( c_i \); /*
    Insert the SIU-transmit place which corresponds to SIU\(_{(j)}\).
    if \( j = 1 \) then /* input place of the transition = connection-setup */
        \( d_1 := S_{(j)}^c - S_{\text{min}} \)
    else
        \( d_1 := S_{(j)}^c - (S_{(j-1)}^c + e_{(j-1)}^c(t)) \)
    end
    \( d_2 := e_{(j)}^c(t) \) /* \( d_2 \) corresponds to the transmission delay of the SIU */
    \( C(p) := c_i \)
    Insert transition having an input place as SIU\(_{(j)}\).
end

Insert the connection-release place for channel \( c_i \).
end

Insert the final transition that has all its input places as the connection-release places.

Insert the final place which is the output place of the final transition.
\end{verbatim}

6.2 Generation of the Receiver-XOCPN

XOCPN at the receiver site, known the receiver-XOCPN, is used to receive and playout the multimedia data. This XOCPN is generated along with the transmitter-XOCPN at the time the object is accessed by the user over the network. For the purpose of receiving the data, it establishes a connection according to the connection-setup place and closes the connection according to the connection-release place, passively. Our connection management approach differs from the one given in [13]. Since SIUs from the same object may not necessarily be transferred through the same channel, we cannot set up an end-to-end connection at the object level as proposed in [13]. Instead, we put the connection-setup places before starting
any **SIU-playout** action and insert the **connection-release** places after finishing all the **SIU-playout** actions. In that way, it is possible for the receiver to interact with the transmitter for reserving network resources prior to the communication of the multimedia information. In this scenario, the receiver-XOCPN does not control the channel path over which SIUs are transmitted, rather it receives arriving SIUs passively. The receiver recognizes each SIU by some embedded information regarding object identity.

The second functionality of the receiver-XOCPN is the synchronous **playout** of SIUs according to the OCPN specification. To ensure such a playout, precedence relation among SIUs and timing analysis of the transmitter-XOCPN need to be considered in the receiver-XOCPN. For example, the earliest playout start time given in Proposition 1 also provides initial delay before starting playout at the receiver. From the receiver’s point of view, initial delay for $T_{max}$ is required to remove possible gaps during the playout of SIUs.

The other consideration is the inter-stream pacing functionality [13]. It is only applicable to isochronous data which share some strict timing constraints. Such data can be identified by checking whether they are inputs or outputs of the same transitions in an OCPN. Thus for $R(p_i), R(p_j) \in R_I$, if $p_i, p_j \in I(t_r)$ for any $t_r \in T$, then objects $p_i$ and $p_j$ need interstream pacing between them. Same conditions apply to $p_i$ and $p_j$ with $p_i, p_j \in O(t_r)$.

Now, we formally present the **receiver-XOCPN generation** algorithm as follows.

---

**receiver-XOCPN generation algorithm:**

$\mathbf{P} := \mathbf{O} / *$ initialize the set of places */

$\mathbf{T} := \mathbf{O} / *$ initialize the set of transitions */

Add the initial place $p_{initial}$ to $\mathbf{P}$

Add the initial transition $t_{initial}$ to $\mathbf{T}$

Insert the connection-setup places as outputs of the initial transition according to the number of available channels.

Add a transition to $\mathbf{T}$ having all its input places as the connection-setup places.

for each transition in the OCPN, except the final one, do
for all the object places that are output places of the transition do
Divide an object into $l$ SIUs, where $l$ is given in Eq. 1.
if the object needs inter-stream pacing then
    Determine the inter-stream pacing point interval $u$.
end
for $i := 1$ to $l$ do
    Insert the SIU-playout place for $i$th SIU.
    Set $d_2$ of the place equal to SIU duration according to the media type of the object.
    if the object $\in O(t_{initial})$ in $OCPN$ and $i = 1$ then
        Set $d_1$ of the place to $T_{max}$. /* introduce the initial delay before playout */
    else
        Set $d_1$ to be zero.
    end
    Insert a transition with input place representing the $i$th SIU.
    if the object needs inter-stream pacing and $i$ is multiple of $u$ then
        Insert interstream-synchronize place.
        Insert a transition with input place being the interstream-synchronize place.
    end
end
end

Insert all the connection-release places.

Insert the final transition with all its input places as the connection-release places and along its output place as $p_{final}$.

Insert the final place which is the output place of the final transition.

7 An Example

In this section, we present an example that illustrates the concepts described in the previous sections. As an example, consider the OCPN shown in Fig. 3 which represents a temporal model for a pre-orchestrated multimedia information. In this model, streams of four video data, three audio data, two images and one text data are integrated in time. We assume that the time unit used for each place is normalized to the duration of one video frame. In other words, a duration of 6 units of time represents $6/30$ seconds. To simplify the analysis, we further assume the following sizes of various data objects:

- Image: $4 \times 4$ in color bit-map.
Figure 3: An example of OCPN for the pre-orchestrated multimedia information

- Text: 80 col x 24 lines ASCII characters.
- Audio: telephone quality audio.
- Video: uncompressed medium resolution color video.

First, we need to find the playout deadlines for each object which can be generated by using the serialize-net algorithm [7]. The resulting playout deadlines are given by the schedule \( \Pi = \{0, 0, 0, 0, 6, 6, 9, 9, 12, 12, 12\} \).

Assuming an ATM network with cell size of 53 bytes, each cell can have up to 47 bytes information for video and audio data, and 44 bytes information for image and text data. The remaining bytes are used for the ATM layer header, the AAL layer header and the trailer. For the delay analysis, we assume the end-to-end propagation delay is 100 \( \mu \)sec, and the jitter delay has a statistical delay bound. Also, 100 Mbps, 10 Mbps, 77 Kbps, and 280 Mbps are the given channel throughput requirements for image, text, audio, and video respectively.

After clustering objects in the OCPN and segmenting them into SIUs, the channel occupation durations can be found for all the SIUs based on the formulations given in Section 3.1. Accordingly, the channel occupation durations can be used to find the maximum concurrency for each media type. As a result, of this analysis, we need two channels for video.
Table 1: Result of channel assignment of SIUs

<table>
<thead>
<tr>
<th>SIU</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

data, two channels for image data, one channel for audio data, and one channel for text data, respectively. This results in a total of 6 channels, which is the optimal channel number.

Suppose the system can only provide two channels of video quality, i.e. these channels have 280 Mbps throughput. This requires recalculating the transmission delay for SIUs of audio, image, and text data. Using the SIU-homogeneous-channel assignment algorithm, the mapping from SIUs to channels is generated and is summarized in Table 1.

The first row of the table indicates object number \(j\) in the OCPN shown in Fig. 3, and first column represents SIU number for an object. The entries in the table indicate the channel numbers assigned to the various SIUs.

Consequently, the transmission schedules for both the channels are given as follows:

Channel 1 = \{{(1)}_1, (2)_1, (3)_1, (4)_1, (5)_1, (6)_1, (7)_1, (8)_1, (9)_1, (10)_1, (11)_1, (12)_1, (13)_1, (14)_1, (15)_1, (16)_1\},

and

Channel 2 = \{{(1)}_2, (2)_2, (3)_2, (4)_2, (5)_2, (6)_2, (7)_2, (8)_2, (9)_2, (10)_2, (11)_2, (12)_2, (13)_2, (14)_2, (15)_2, (16)_2\},

where \({(i)}_j\) denotes the \(i\)th SIU of the object \(j\).

The resulted transmitter-XOCPN is depicted in Fig. 4. In this figure, \(R\) represents the connection-setup place, and \(D\) represents the connection-release place.

At the receiver, we assume interval for inter-stream pacing to be 3. Thus, video and audio streams have the interstream-synchronize places in every three SIU-playout places as shown in Fig. 7. In this figure, the interstream-synchronize place is represented as IPP.
Figure 4: Transmitter-XOCPN for channel-deficient system

Figure 5: Receiver-XOCPN for channel-deficient system

Using the maximum-buffer-estimation algorithm and the timing information given in the finalized schedule, the estimated size of the buffer for two channels can be found. For channel 1, it is 4,320,273 bytes, and for channel 2, it is 616,320 bytes. In this case, the total buffer requirement in the receiver site is 4,936,593 bytes, which is sum of buffer requirements for the both channels.

8 Conclusion

In this report, we have proposed a communication model, called XOCPN, for pre-orchestrated multimedia information, which is used to specify the communication requirements for handling heterogeneous data. We have proposed an approach for generating XOCPNs both for the transmitter and the receiver sites. The transmitter-XOCPN can help in identifying the optimal number of channels which are sufficient to provide concurrent transmissions of communication data units. Subsequently, we have considered a channel-deficient system which is more realistic environment. First, we consider a case that channels are homogeneous in their characteristics. For such a system, we have proved that scheduling transmission of multimedia data is an NP-hard problem, and subsequently proposed a heuristic algorithm with complexity $O(n \log nm)$, where $n$ represents the number of data units to be communicated.
over \( m \) channels. Then, we consider a channel-deficient system with heterogeneous channels. For this case, which also belongs to NP-hard class, we propose a heuristic algorithm whose complexity is \( O(nk^210^gn) \), where \( k \) is the number of channel types available. For the eventual synchronization of streams from MVCs, we also have described the model for the receiver site which is executed consistently with the transmission schedule.

**Appendix: Properties and Verification of the XOCPN Model**

Here, we present the unique properties of XOCPN and prove the Petri-net properties of the proposed XOCPN model. Following the definitions given in [12], the properties which are applicable to an XOCPN are listed below.

- Initial place \( p_{\text{initial}} \) does not belong to any output place of any transition, i.e.
  \[
  p_{\text{initial}} \notin O(t_j), \ \forall \ t_j \in T.
  \]

- Final place \( p_{\text{final}} \) does not belong to any input place of any transition, i.e.
  \[
  p_{\text{final}} \notin I(t_j), \ \forall \ t_j \in T.
  \]

- There is no conflict in XOCPN. In other words,
  \[
  I(t_i) \cap I(t_j) = \emptyset, \ \forall \ i \neq j. \tag{6}
  \]

- Each place is an output place only for one transition. That is,
  \[
  \text{If } p_i \in O(t_j), \ \text{then } p_i \notin O(t_r), \ \forall \ t_r \in T \setminus \{t_j\}.
  \]

**Definition 1 (Transition Function)** The transition function \( \delta : N^n \times T \rightarrow N^n \) for a \( C_{\text{XOCPN}} = \{T, P, I, O, D, R, M, Y, Z\} \) with marking \( M \) and transition \( t_j \in T \) is defined if and only if

\[
M(p_i) \geq \#(p_i, I(t_j))
\]
for all $p_i \in P$. If $\delta(M, t_j)$ is defined, then $M'(M, t_j) = M'$. where

$$M'(p_i) = M(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))$$

for all $p_i \in P$.

The initial marking is denoted with $M_0$, which represents all the marking status of each place at the initial state. In XOCPN, the initial marking $M_0 = \{1,0,0,\ldots\}$. That is only the $p_{\text{initial}}$ has a token at the initial state.

The final marking is denoted with $M_f$, which represents all the marking status of each place at the final state. In XOCPN, the final marking $M_f = \{0,0,0,\ldots 1\}$. That is only the $p_{\text{final}}$ has a token at the final state.

Definition 2 (Reachability) A marking $M_n$ is said to be reachable from a marking $M_0$ if there exists a sequence of firings that transforms $M_0$ to $M_n$. The set of all possible markings reachable from $M_0$ in the net is denoted by $R(N, M_0)$ or simply $R(M_0)$.

Proposition 2 In the $C_{\text{XOCNP}}$, the final marking is reachable from the initial marking, that is $M_f \in R(M_0)$.

Proof:
Let the firing time of $t_i$ be $f(t_i)$. Also, let $t_i \in T$ be ordered so that for all $i < j$, $f(t_i) \leq f(t_j)$.

Each $t_i$ can be in one of the three states: not firable, firable, or firing. Firable state indicates that all the input places of $t_i$ have a token, but the duration of those places have not elapsed.

If $t_i$ is in firable state, then it will eventually fire. At the initial state $M_0$, $t_1$ is always firable.

Let $M_1$ be the next state of $M_0$, i.e. $M_1 = \delta(M_0, t_1)$. Then, after firing of $t_1$, $\delta(M_1, t_i)$ is defined for all $t_i$'s which satisfy

$$I(t_i) \subseteq O(t_1), \quad (7)$$

$t_i$'s in Eq. (7) are called as firable. Since places in an XOCPN have some durations, $t_i$ cannot be fired right after it becomes firable.

31
In general, at any firing time of \( t_j \), \( t_i \) should have been fired for all \( f(t_i) < f(t_j) \). Thus at the firing time \( t_j \), the places satisfying the following condition must have a token:

\[
\bigcup_{k=1}^{j} O(t_k) \setminus \bigcup_{i=1}^{j} I(t_i),
\]

where

\[
f(t_i) < f(t_j) \text{ and } f(t_i) = \max\{\{f(t_1), \ldots, f(t_j)\} \setminus \{f(t_j)\}\}.
\]

Thus \( \delta(M_i, t_j) \) is defined if

\[
I(t_j) \subseteq \bigcup_{k=1}^{j} O(t_k) \setminus \bigcup_{i=1}^{j} I(t_i)
\]

for \( i \) which satisfies Eq. (8). However, according to Eq. 6, if

\[
I(t_j) \subseteq \bigcup_{k=1}^{j} O(t_k),
\]

then \( I(t_j) \) should satisfy Eq. (9). Furthermore, for \( j \) and \( j' \) with \( f(j) = f(j') \),

\[
I(t_j) \cap O(t_{j'}) = \emptyset.
\]

Thus, for \( i = j - 1 \), Eq. 10 is satisfied, since \( f(j - 1) \leq f(j) \). In other words, if

\[
I(t_j) \subseteq \bigcup_{k=1}^{j-1} O(t_k),
\]

then \( M_j = \delta(M_{j-1}, t_j) \) can be also defined.

By using the Eq. (11) and the fact that \( M_1 \) is defined, it is possible to deduce that for all \( i, 1 < i < m \), \( M_i = \delta(M_{i-1}, t_i) \). In addition to that,

\[
\bigcup_{i=1}^{m-1} O(t_i) = P \setminus \{p_{\text{final}}\}
\]

Thus,

\[
I(t_m) \subseteq P \setminus \{p_{\text{final}}\}.
\]

As a result, \( M_f \) can be defined and it is reachable from \( M_0 \). □
Definition 3 (Boundedness) A net is said to be \( k \)-bounded or simply bounded if the number of tokens in each place does not exceed a finite number of \( k \) for any marking reachable from \( M_0 \). A net is said to be safe if it is 1-bounded.

Proposition 3 \( C_{XOCPN} \) is safe.

Proof:
Using the properties given in Eq. 7 and the fact that \( \#(p_i,O(t_j)) = 1 \), we conclude that each place can have at most one token. \( \Box \)

Definition 4 (Liveness) \( M_0 \) is said to be a live marking for a marked Petri-net if, irrespective of the marking that is reached from \( M_0 \), it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. Specifically, a transition \( t_j \) is said to be \( L1 \)-live (potentially firable) if there exists a \( M' \in R(M_0) \) such that \( t_j \) is enabled in \( M' \).

Proposition 4 All the transitions in the \( C_{XOCPN} \) is \( L1 \)-live.

Proof:
We can notice from the proof of reachability that all the transitions can be fired. Once the net has reached the final state, there is no further transition since \( p_{\text{final}} \) is not an input place for any transition. Thus all the transitions in the \( C_{XOCPN} \) can be fired exactly once. \( \Box \)

Definition 5 (Synchronous preserving transformation) We define a transformation of a net to be a synchronization preserving one if the transformed net inherit synchronization information of the original net.

Proposition 5 The receiver-\( XOCPN \) is a synchronization preserving transformation of \( OCPN \).

Proof:
The proof can be obtained by showing that \( OCPN \) can be reconstructed from the receiver-\( XOCPN \). Reconstruction is achieved by using fusion of cascaded places and fusion of cascaded transitions [10]. Fusion of cascaded transitions is used to remove connection-setup,
connection-release, and interstream-synchronize places. Also, fusion of cascaded places is used to merge SIU-playout places into object places. The procedures for the fusions are as follows:

- For the connection-setup place and connection-release place:
  All the connection-setup places are the output places of the initial transition, and the input places of one transition, say $t_r$. In this case, as we can notice from Fig. 6, the connection-setup places can be removed from XOPN by changing the set of output places of $t_1$ and the set of transitions as follows:

$$O(t_1) = \{O(t_r)\}$$

$$T \setminus \{t_r\}.$$

The same procedures are applicable to the connection-release places.

- For the interstream-synchronize place $p_j$:
  Let $p_j \in O(t_r)$ and $p_j \in I(t_s)$. Also assume $p_i \in I(t_r)$. Then we can remove $p_j$ from the receiver-XOPN by $I(t_s) \setminus \{p_j\} \cup \{p_i\}$ and $T \setminus \{t_r\}$.

- For the SIU-playout places $p_i$ and $p_j$ which belong to the same object:
  Let $p_i \in I(t_r), p_j \in O(t_r)$ and $p_j \in I(t_s)$, as is noticed in Fig. 7. $p_j$ can be merged into $p_i$ by setting duration of $p_i$ equal to the sum of durations of $p_i$ and $p_j$ and changing the set of places, transitions, and the input places of transition $t$, as follows:

$$I(t_s) \setminus \{p_j\} \cup \{p_i\}$$

$$P \setminus \{p_j\}$$
Execution of the above procedures for all the places in the receiver-XOCPN restores the original OCPN. □

References


