International Journal of Multimedia and Ubiquitous Engineering
Vol. 4, No. 2, April, 2009

A Formal Approach for an Environment-aware Verification of the Consistency of a Multimedia Presentation

Abdelkrim Abdelli
LSI Laboratory- Computer Science Department- USTHB university
Algiers - Algeria
Abdelli@lsi-usthb.dz

Abstract

We propose in this paper a formal methodology that makes it possible to check whether a multimedia presentation could behave consistently when it is processed in a resource-limited environment. The approach uses the TSPN model to specify the resource constraints of the environment as well as the time and the synchronization requirements of the presentation. A graph is then built from this specification to check over its consistency. Hence it becomes possible to manage more efficiently the resource availability, and to adapt the content of the presentation so that the latter can be performed consistently under the given environment constraints.

Keywords: *Consistency verification, Multimedia authoring, Petri nets, Reachability graph.*

1. Introduction

Distributed multimedia applications become nowadays a useful medium to provide a rapid and a pervasive access to friendly-authored multimedia presentations. They are applicable in various domains such as education, advertising, entertainment and communication. To describe such applications, many standards have been defined and the most dominant reference on the Internet is the SMIL (Synchronized Multimedia Integration Language) language [6]. The objects requested in such a document are referenced throughout its code and their data are located widespread over a wide area network. Moreover, these applications often have complex requirements on quality of service and temporal order among media streams. Hence, the design and the implementation of these requirements are inherently complex and present an extraordinary design and programming challenge. For this effect, formal methods have been widely used to overcome these issues [4][5][9][12][14]. These methodologies assist the author in building a consistent multimedia code according to user-requirements by checking whether the latter are conflicting or not. However, these techniques still remains incomplete and not efficient as they do not take into account the needs of these presentations in terms of resources; actually they consider that the environment under which the presentation will be processed provides enough resources to deliver its content in time. However, as the environment resources [1] may be subject to limitations at times, these situations can affect both quality and the consistency of the multimedia presentations. In concrete terms, as the requested data is hosted on different servers located widespread over heterogeneous networks, it takes a certain time to recover it. This delay called the latency (which lies on the bandwidth fluctuations and the workloads of the remote servers), can be too high to make the code requirements be not met. Besides, the memory space available in

[1] We consider the availability of two main resources: bandwidth connection, and buffering memory space.

the player device can be short to save all the data to buffer, thus causing the stalling of the presentation.

We present in this paper a formal methodology that makes it possible to check over multimedia presentation consistency while considering the environment constraints under which the presentation would be processed. With this intention, in addition to the time and synchronization requirements, for which the presentation must conform with, the approach takes into account the resource constraints of the environment. For this effect, the methodology uses the *TSPN (Time stream Petri net)* model [13] for the specification of all the requirements of the multimedia presentation. Then by using the algorithm defined in [1], it derives thereof the reacheability graph that depicts all the possible runs of the presentation. The obtained graph is then used to check whether the presentation can behave consistently under the assumed environment constraints, or not. Hence, if the resources are short to satisfy all the needs, then the author can decide, for instance, to resize the media-files so that the resource conflicts are removed. Furthermore, by using the graph, the author can adapt the content of the presentation in order to manage efficiently the resource availabilities, and to improve the overall quality of the presentation.

The rest of this paper is organized as follows: Section 2 lays down the problematic and the methodology to follow. Section 3 shows how to check over the consistency of a multimedia presentation.

2. Methodology

Our methodology is dedicated to work within an authoring tool. It allows assisting the author in building a consistent code that can run under any limited environment. So far, the existing approaches allow only to check over the intrinsic consistency of a multimedia presentation, namely whether the user requirements are conflicting or not. However, as a multimedia document can be processed in various environments, therefore its presentation may behave differently. For example, the bandwidth is much higher in a LAN than for a xDSL link. On the other hand, the capacity of the buffer is very limited in pocket devices like PDA or cellular phones, whereas it is unlimited in a PC or a Laptop. Consequently, an intrinsically consistent presentation which may not run correctly under a limited environment might perform coherently and timely under an unrestricted one. To deal with that, the trend is towards adapting the code according to user profiles [10][11][7][8][2], so that to provide for each environment a customized code that performs consistently.

However, to achieve this goal, we need to supply the author with a formal methodology that can settle on whether the adapted code could comply with the constraints of the environment to which it is dedicated. For this effect, we propose in the sequel a formal approach that allows the modeling and the verification of the requirements of a multimedia presentation that should run under given environment constraints; these are featured by the following parameters:

- *Mem*: denotes (in KBytes), the amount of memory space available to store the data to be streamed by using a *RTSP*-like protocol.
- *BW*: denotes (in KBytes/sec), the connection speed available at the player side.
- *Buffer*: denotes (in KBytes), the memory space allocated by the player to receive the data relative to each media to be streamed.

Furthermore, we assume that the available bandwidth and memory are dedicated only for the process of the multimedia presentation, and are not shared with another running application. What is more, it should be noticed that a downloaded file is saved into a hard disk cache, whereas a stream is saved temporarily into memory. However, as pocket devices are not endowed with hard disks, the *HTTP*-like protocol is no longer used to service these devices. Actually, video and audio are delivered as streams and saved temporarily into buffers of limited sizes. Thereafter, hard disk space constraints are not considered as an issue since we admit that there is enough space therein to save all the files to download.

The next section shows how the modeling of the resource requirements is worked out. Notice that the modeling of time as well as synchronization constraints with *TSPN* has been already addressed. Therefore, for further explanations concerning these issues, one should refer to the related literature [13] [14].

2.1. The modeling process

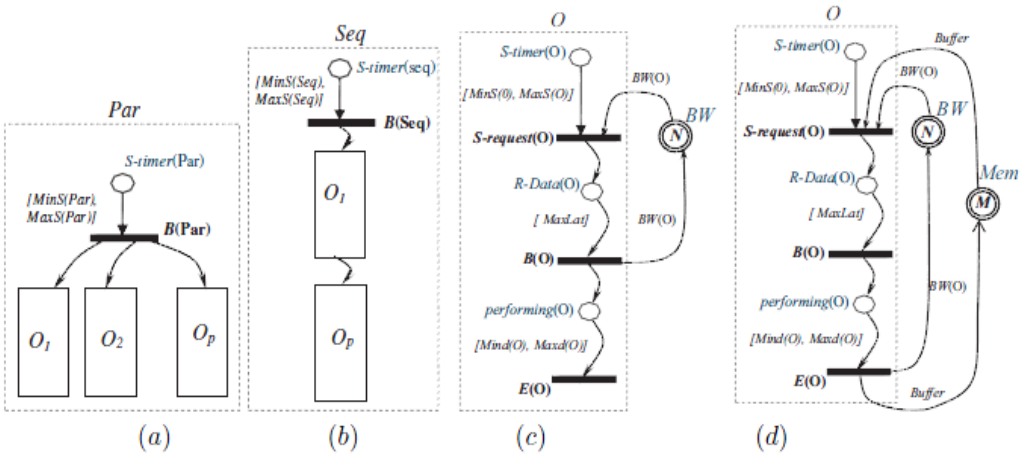

Figure 1. The modeling process with *TSPN*.

A multimedia presentation can be seen as the fitting of different basic media (video, audio, text, image or animation), that are processed according to a certain scheme (in parallel or in sequential), while being synchronized, time constrained, sharing common resources and with potential user interactions. The specification of the body of such a presentation is built hierarchically according to its structuring. The time and resources requirements of each basic media are specified as a *TSPN* unit. Then, the different units are composed in sequential or in parallel to obtain a block. Then the blocks are composed to form the main blocks until completing the body of the specification. Finally, some transitions are merged to characterize the inter media synchronization, and some features are added to characterize the beginning and the end of the multimedia presentation. For instance, a basic characterization of a sequential execution of a set of media $O_1,..,O_p$ can be performed as shown in Fig.1.b. On the other hand, if we need to model the parallel execution of the media $O_1,..,O_p$, we should follow the construction pictured in Fig.1.a. The transition *B(Par)* (respectively, *B(Seq)*), denotes once fired the occurrence of the starting event of the parallel block Par (respectively, the sequential block *Seq*). The time intervals associated with the ingoing arcs of previous

transitions denote the delay that should be observed before the effective starting of each block.

We focus now in how to characterize the requirements of each basic media. For this effect, two main places are used: The place *BW* (whose initial marking is set to the value *N*), which denotes the bandwidth availability; The place *Mem* (whose initial marking is set to *M*), which models the memory space availability. These two places are shared by all the units of the specification, and model the common resources for which the different media are in conflict. So, let *O* be a media to perform within a multimedia presentation. We assume that the time delay to observe before starting to process *O* is delimited by the interval *[MinS(O), MaxS(O)]*. We suppose also that the duration of the presentation of *O* can last a certain time within the range *[MinD(O), MaxD(O)]*. To characterize the resource requirements of a media, we need to determine whether its service uses a file-oriented protocol like *HTTP*, or a stream oriented protocol like *RTSP*. Notice that *HTTP* differs from *RTSP*, as it requires downloading the entire file before starting to play the requested portion, whereas *RTSP* delivers the media almost in real time as an encoded and a compressed stream. Further, we need to determine the size of the data to download from the hosting server. This amount of data, noted *size(O)*, is given by the size of the original file in case of a *HTTP*-like protocol or by the size of the data stream to download in case of a *RTSP*-like protocol. Moreover, as the player needs a certain time to download the data, it allows therefore an elastic time when enforcing the time constraints of the media [3]. However, the majority of the players assume that this elasticity should not surpass a given threshold, denoted hereafter by *Maxlat*. In concrete terms, if the media *O* is requested to start at time δ, then the player tolerates that the effective beginning of its presentation can be delayed to not before $\delta+Maxlat$. If the last threshold is overtaken without recovering all the data required to perform the presentation, then the player reports a time constraint violation.

The characterization of the requirements of a media *O* can be specified by the *TSPN's* depicted in Fig 1(c,d). Actually, Fig 1.c shows the specification obtained when considering a *HTTP*-like delivery, whereas Fig 1.d represents the modeling in case of a *RTSP*-like one. The place *S-timer(O)* denotes once marked that the timer (which counts down the remaining time before starting *O*), is activated. The firing of the transition *S-request* denotes here that the request has been serviced and the data is being received (The place *R-data* is marked). This is achieved if there is enough bandwidth available such that the necessary data can be recovered on time ($Bw(O) \leq N$). In case of a *HTTP* like protocol, the value of *Bw(O)* is determined, such that the original file can be downloaded entirely before the effective beginning of the presentation of the media; namely its download should not last more than *Maxlat* seconds. Besides, as the request services consume an additional bandwidth for network overhead, error correction, resending lost data, and so on, we need therefore to consider an additional bandwidth for these needs, which can be approximatively assessed to 30 % of the original assigned value. Hence we have: $Bw(O) := ((size(O) \times 1{,}30)/(Maxlat))$.

On the other hand, for a *RTSP*-like delivery, the bandwidth to allocate is determined according to the client connection speed and the nature of the media, as stipulated by Real Networks recommendations (see Tab 1). Note that before starting to service a *RTSP* request, the player needs also to allocate a space of memory to save the data to buffer, amount of which is configured at player level by the user. Once the necessary data is received, the effective starting of the media *O* (reported by the firing of the transition *B(O)*), is launched, and the performance lasts until its effective end will be reported (firing the transition *E(O)*).

Notice that in case of a *RTSP*-like delivery, the allocated resources (bandwidth and buffer), are released once that the performance of the media is ended; since the data is continuously streamed until this time. As concerns *HTTP* requests, the allocated bandwidth is released before the effective starting of the presentation.

Table 1.Maximal connection rates for streaming media.

Connection rate	Video	Voice only	Voice and music	Music mono	Music stereo
28.8 Kbps Modem	20 Kbps	6.5 Kbps	6.5 Kbps	8 Kbps	8 Kbps
56 Kbps Modem	34 Kbps		8.5 Kbps		
64 Kbps ISDN	45 Kbps	8.5 Kbps		11 Kbps	11 Kbps
112 Kbps	80 Kbps	16 Kbps	20 Kbps	16 Kbps	20 Kbps
Corporate LAN	150 Kbps	32 Kbps	32 Kbps	32 Kbps	32 Kbps
256 Kbps DSL cable	225 Kbps		44 Kbps	44 Kbps	44 Kbps
384 Kbps DSL cable	350 Kbps	64 Kbps	64 Kbps	64 Kbps	64 Kbps
512Kbps DSL cable	450 Kbps				
786Kbps DS cable	700 Kbps				

2.2. Example

For a better understanding of the approach, let us consider the following multimedia presentation which consists in a streaming video *Vid* that should start *3s* later than a sequence of an image *Img* followed by another streaming video Vid_2. This presentation ends once that Vid_2 stops. Notice that the performance of *Vid* respectively, *Img* and Vid_2 should last between *6s* and *8s*, respectively *10s*, and between *8s* and *10s*. We assume that this presentation is likely to be performed under the following environment: *BW:=1000 KB/s, Mem:=800 KB, Buffer:=500 KB*, and that the time elasticity of the player should not surpass the value: *Maxlat=5s*. What is more, we have:

- *size(Img):=384 KB,* hence we determine *B(W)=100 KBps.*
- The bandwidth rates used to stream respectively *Vid* and Vid_2 are *Bw(Vid):=800KBps*, $Bw(Vid_2)$*:=600 KBps*.

The *TSPN* of Fig 2.a models the requirements of the previous multimedia presentation. So, each media involved in the presentation is modeled as a *TSPN* unit. Then, the different units are connected in parallel. Finally, the transitions *B(Doc)* and *E(Doc)* are added to characterize the start and the end of the presentation. Notice that the Master rule is associated with the transition *E(Doc)* to denote that the process identified by the master arc coming from the place End_1 is that which governs its firing.

3. Consistency verification

The formal verification of a multimedia presentation uses the reachability graph derived from its *TSPN* specification. This graph is obtained by applying the construction defined in [1] and encompasses all the possible temporal runs of the presentation. Therefore, to check whether the specification is consistent or not, we need to verify whether the graph satisfies the

consistency property or not. As defined in [5], the graph enjoys the consistency property if: "*For every path in the graph, the event characterizing the beginning of the multimedia presentation is necessarily followed by the event characterizing its end*".

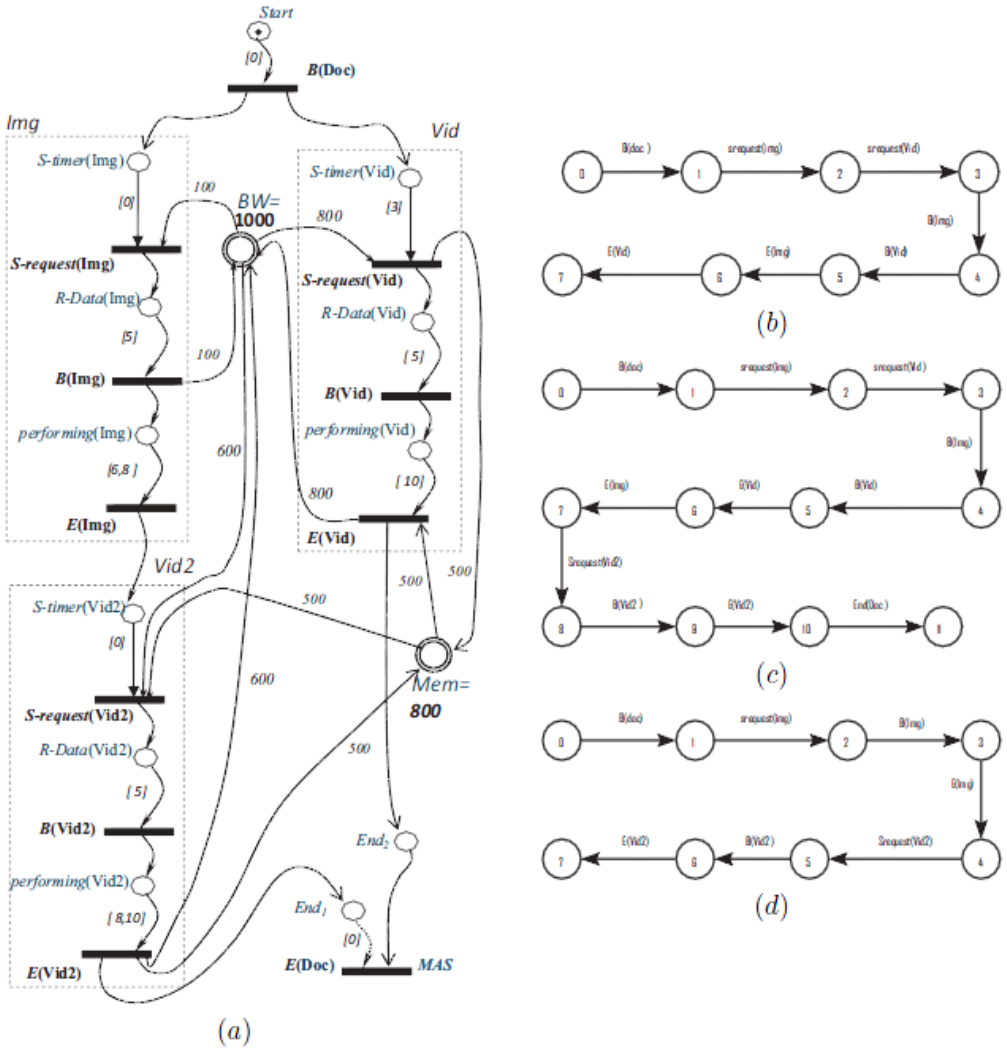

Figure 2. The specification of the multimedia presentation and its reachability graphs.

For a better understanding of this concept, let us check the consistency of the specification given in Fig 2.a. The graph derived from this specification is shown in Fig 2.b, and each path thereof describes one of the potential runs of the presentation. Hence we notice that the path (0→7) is the only inconsistent path, as its begins with the event *B(Doc)* (denoting the start of the presentation), but unfortunately does not terminate with the event *E(Doc)* (denoting the end of the presentation). Therefore, the multimedia presentation presented in Fig 2.a is *inconsistent*. This inconsistency is due to the fact that the available memory space is short to receive in the same time the data of both *Vid* and Vid_2; hence Vid_2

fails to start. This dysfunction can be removed by temporal formatting. Concretely, it needs to prolong the duration of *Img* so that the presentations of *Vid* and Vid_2 do not overlap. For instance, if the duration of *Img* lasts *14s*, then we obtain the graph shown in Fig 2.c. We notice that the latter is consistent since its unique path ($0 \rightarrow 11$) enjoys the consistency property. However, if we assume now that the time elasticity is overvalued to *2s* rather than *5s*, the presentation becomes again inconsistent (see Fig 2.d). Actually, as we have *Bw(Img):=250 KBs*, we are not able to manage the needs in terms of bandwidth for both media *Img* and *Vid* so that both can be downloaded on time. To overcome this issue, we can adapt the content of *Img* by considering a lesser resolution, thereby reducing its size and the amount of bandwidth needed for its download.

To sum up, the consistency verification is performed by exploiting the reachability graph. Therefore, if the graph does not conform to the previous consistency definition, we could cure the document by identifying the origin of the inconsistencies by locating the media involved in the resource conflict. Hence, we can remove these dysfunctions either by reducing the size of the media or by temporal formatting to prevent their appearance.

This methodology could be applied at the authoring stage to assist the author in building a consistent multimedia presentation. Besides, the reachability graph can be used at the player stage to schedule only consistent scenarios when they exist. In case where all the paths are inconsistent, the player should warn the user by pointing out the location of the resource conflicts and when they may occur. The user may be advised to increase the resource capacities of its player to be able to show the presentation. This methodology can be also used to assess the resource availability along a running scenario; this can help to compute an accurate pattern that can be used to implement efficient pre-fetching schemes.

4. Conclusion

We have proposed in this paper a formal methodology that allows the consistency verification of a multimedia presentation while considering the resource constraints of the environment under which it is dedicated to perform. For this effect, we showed how to use the *TSPN* model to specify the requirements of such presentations as well as the resource constraints of the environment to comply with. Then, by using the construction defined in [1], we are able to derive from this specification its reachability graph that depicts all the possible runs of the presentation. Hence, we discussed how to check over the consistency of the presentation and how this graph can be used to correct the conflicting situations when they occur.

References

[1] A. Abdelli and N.Badache. "Towards buiding the reachability graph of the TSPN model" In FUNDAMENTA INFORMATICAE - IOS PRESS -VOL 86-4 (2008) Pages 371-409.

[2] A. Abdelli, N. Badache: Efficient Bandwidth and Buffer Management for Multimedia Data Download. IEEE CS -FGCN (1) 2007: 424-429.

[3] B. Bachelet, P. Mahey, R. Rodrigues, L.F Soares: Elastic time computation in QoS-driven hypermedia presentations. Multimedia Syst. 12(6): 461-478 (2007).

[4] Dick C. A. Bulterman, Lynda Hardman: Structured multimedia authoring. TOMCCAP 1(1): 89-109 (2005).

[5] P.N.M. Sampaio, C.A Sontos, J. P. Courtiat. "Using formal method to verify the temporal semantics of SMIL documents". IFIP, Beijing, China, August 21 - 25 2000.P.

[6] SMIL : W3C Recommendations, SMIL2.0 Specification. URL:http://www.w3.org/TR/SMIL20.
[7] S. Benbernou, A. Makhoul, M-S Hacid, A. Mostefaoui: A Spatio-Temporal Adaptation Model for Multimedia Presentations. ISM 2005: 143-150.
[8] chevillat : V. Charvillat, R. Grigoras: "Reinforcement learning for dynamic multimedia adaptation. J. Network and Computer Applications" 30(3): 1034-1058 (2007)
[9] M. Jourdan, C. Roisin, L. Tardif: Constraint Techniques for Authoring Multimedia Documents. Constraints 6(1): 115-132 (2001).
[10] T. Lemlouma, N. Layaïda: Context-Aware Adaptation for Mobile Devices. Mobile Data Management 2004.
[11] M. J. O'Grady, G.M. P. O'Hare, C. Donaghey: Delivering adaptivity through context-awareness. J. Network and Computer Applications 30(3): 1007-1033 (2007)..
[12] Sampaio.M.P.N. Conception formelle de documents multimédia interactifs : une approche s'appuyant sur RT-LOTOS. PHD Thesis, University of Paul Sabatier, Toulouse, France, April 2003.
[13] P. Senac. Modeling logical and temporal synchronisations. In IEEE JSAC V14, n°1 Jan 1996 pp.84-103.
[14] R. Willrich, P. de Saqui-Sannes, P. Sénac, M. Diaz: Multimedia Authoring with Hierarchical Timed Stream Petri Nets and Java. Multimedia Tools Appl. 16(1): 7-27 (2002).

Authors

Dr Abdelli Abdelkrim is a lecturer at the computer science department of the university USTHB in Algiers. He is a member of the research team MOVES of the LSI laboratory. He received his master degree in 1998 and his PHD in 2007 from the same university. His areas of interests deal with formal methodologies, real time systems, and multimedia applications.