©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 26

CAFSM: A Communicating Adaptive Finite State Machine for Personalized Multimedia Streaming

Susan Elias
Dept. of Computer Science
Sri Venkateswara Coll of Engg
Sriperumbudur, Chennai
Tamil Nadu, India

Jasmin C.B
Dept. of Computer Science
Sri Venkateswara Coll of Engg
Sriperumbudur, Chennai
Tamil Nadu, India

Lisa Mathew
Dept. of Mathematics
Sri Venkateswara Coll of Engg
Sriperumbudur, Chennai
Tamil Nadu, India

Easwarakumar.K.S
Dept. of Computer Science
College of Engineering
Anna University, Chennai
Tamil Nadu, India

Richard Chbeir
Dept. of Computer Science
LE2I - Bourgogne University
Dijon, France

ABSTRACT
In this paper we present the design of a multimedia presentation system which permits the dynamic adaptation of the content. The Communicating Adaptive Finite State Machine (CAFSM) presented in this paper, has been used to describe the multimedia streaming and presentation system proposed here. This system is driven by a set of messages that are used for communication and co-ordination among the various component machines which form the system

Keywords
Communicating, Adaptive, Finite State Machines, Multimedia Presentation, e-Learning

1. INTRODUCTION AND MOTIVATION
The *Dynamic Extended Finite State Machine (DEFSM)* was proposed in [9, 10] to model a multimedia synchronization and execution system. This presentation control model allows dynamic transitions in order to deal efficiently with user interactions thereby reducing the number of transitions required. Our attempt to modify the *DEFSM* model in order to present content that can be dynamically adapted, led to the development of the *Communicating Adaptive Finite State Machine* (*CAFSM*) presented here. The limitation of the DEFSM model is that each synchronization point in the presentation layout is modeled as a state in the *DEFSM*. Thus, the *DEFSM*s representing fast changing multimedia presentations will have several states and transitions. The number of states and transitions in our model are independent of the number of synchronization points in the presentation, thereby reducing the memory requirement of the system especially for fast changing multimedia presentations. Synchronization points are identified as points in the presentation layout where one or more media objects begin or end their presentation.

Communicating Finite State Machines (CFSM) were introduced in [2]. In this model Finite State Machines were used to model processes and queues to represent the communication channels between them. However the communication performed by the *CFSM* was carried out using a broadcast model while our proposed *CAFSM* model can also multicast data. In our proposed system the various components communicate with each other by exchanging a finite set of messages which have been specified here. The motivation behind this work was the need to develop a distributed multimedia presentation system that would ensure the synchronized playout, while simultaneously handling adaptation and user interactions efficiently. Other existing synchronisation models and their features are discussed in [1, 15, 16].

2. APPLICATION OF THE PROPOSED RESEARCH WORK
This paper presents the design of a presentation tool that can support the dynamic adaptation of the contents in distributed multimedia applications. In this subsection a domain specific application of the adaptive multimedia presentation tool is presented.

Adaptive E-Learning System:

E-learning systems do exists and are successfully being utilized by several organizations and universities but the challenge now is to personalize the content for each user automatically. The motivation behind personalising content was the need to provide a learning environment to suit each person's learning capabilities. Adaptive e-Learning systems are currently being designed and several existing e-learning tools are being incorporated with this feature [4, 13, 5]. But most of them use predefined or adaptive tests [14] to decide the learning path, making them static adaptation models. Understanding learning models [3, 7] and incorporating them into e-learning systems is the biggest challenge and several research works have been focusing on this aspect. The integration of neuro-fuzzy logic [11] and eye tracking approaches [8, 12], to decide on the learning paths are currently being researched in order to dynamically adapt the content without the learners knowledge. *'Advance Personalised Learning'* is also one of the 14 Grand Engineering Challenges for the 21st century put forth by the *National Academy of Engineers* [17].

In case of complex multimedia presentations where the data reside in distributed servers, the response time of the presentation tool is also important during adaptation. Presentation tools need to be redesigned to handle adaptation. An innovative model called *Communicating Adaptive Finite State Machines CAFSM*, introduced in this paper in the following section, is used to present the design of the proposed presentation tool. Quality of Experience (*QoE*) parameters are also being defined and used to evaluate the performance of Adaptive e-learning Systems being designed. SCORM the standard used in the existing e-learning systems has also been extended [18] to support Adaptive e-learning.

3. COMMUNICATING ADAPTIVE FINITE STATE MACHINES: CAFSM

This section presents the proposed synchronization mechanism for distributed multimedia presentations. In order to model the synchroniser we define a modification of the finite state machine and refer to it as *Communicating Adaptive Finite State Machine (CAFSM)*. The proposed model uses three variations of the *CAFSM* referred to as

i) Presentation FSM (*PFSM*)
ii) Media Schedule Managers (MSM) and
iii) Layout generator (LG).

The *PFSM* represents the entire presentation at the server ($PFSM_{Server}$) and also at the client ($PFSM_{Client}$). Media Schedule Managers (*MSM*) are used to co-ordinate the transfer/streaming of discrete/continuous media objects, and are present at both the server MSM_{server} and at the client MSM_{client}. There is one *MSM* for each continuous media object (such as video, audio and animation) present at the server as well as the client and a single *MSM* each at the server and at the client to handle the transfer of all discrete media objects (such as text, images). The function of *LG* is, to extract from the temporal layout of the presentation the list of media objects beginning and ending at each synchronization point, and communicate them to the $PFSM_{client}$ on demand. The communication between the various components of the synchronization model is depicted in Figure 1. To ensure synchronisation during playout, messages are used to control the entire data flow required for the presentation. The sequence of messages passed between the finite state machines and their state changes, as well as the details of the individual components of the proposed *CAFSM* model will be explained in the following subsections.

3.1 Presentation Finite State Machines: PFSM

Formally, a presentation finite state machine *PFSM* can be defined as a 4-tuple (S, Σ, L, δ), where S is a finite set of states $\{S_1, S_2, S_3\}$, $\sum$ is a set of the form (m, M) where m indicates a message and M indicates the *CAFSM* involved, L is a set of participating media objects, and $\delta : S \times 2^{\Sigma} \times L^2 \rightarrow S \times 2^{\Sigma} \times L^2$ is a transition function. The set of messages used by all the *CAFSM*s is given in Table 1. The transitions are in the form $\delta(S_i, (m_j, M_k), L', L'') = (S_l, (m_p, M_q), L^*, L^{**})$, where S_i and S_l are states of the *PFSM* under consideration, (m_j, M_k) and (m_p, M_q) are the messages received and sent respectively, along with the source / destination finite state machines, L' L'', L^* and L^{**} are lists of multimedia objects. Here, it is assumed that the *PFSMs* can communicate with each other as well as with the local *MSMs* only. In case there is a need to send or receive the same message m to / from a list L of local *MSMs*, the notation *MSM(L)* is used in place of M_k or M_q. All interactions from the user are treated as messages initiated by the $PFSM_{client}$. The rest of the section deals with the behaviour of *PFSMs* i.e. the transitions associated with each of the *PFSMs*.

Table 1: List of messages

S.No	Message	Title
1	CR	Connection Request
2	CI	Connection Indication
3	CC	Connection Confirmation
4	BT	Begin Transmission
5	BP	Begin Presentation
6	FP	Finished Presentation
7	FT	Finished Transmission
8	OK	OK
9	SL_0	Send List L_0
10	SL_1	Send List L_1
11	SL_2	Send List L_2
12	RL_0	Receive List L_0
13	RL_1	Receive List L_1
14	RL_2	Receive List L_2
15	NL	New List
16	DR	Disconnection Request
17	DI	Disconnection Indication
18	DC	Disconnection Confirmation
19	PS	Pause
20	RS	Resume
21	FR	Fast-forward/Rewind
22	DS	Discard
23	AI	Authoring-based Interaction
24	RD	Receive Data
25	SP	Start Presentation

Client PFSM

The *PFSM* at the client side represents the status of the ongoing presentation at any point of time. It is modeled using 3 states S_1, S_2, S_3 and 18 transitions as detailed below. Figure 1 shows the transition diagram for the $PFSM_{client}$.

T1: When the request for the presentation is initiated by the user, the system is in state S_1. This is represented by the *Start Presentation (SP)* message from the $PFSM_{client}$ to itself. The $PFSM_{client}$ initiates the connection establishment further by sending *CR* messages to *LG* and remains in state S_1 until connection has been established successfully.

T2: In state S_1, when the Connection Indication (*CI*) message is received from *LG*, a request for the list L_0 is sent to *LG*.

T3: In state S_1, when the Receive List RL_0 message followed by the list L_0 of all participating media objects is received from *LG*, the $PFSM_{client}$ further initiates connection establishment (*CR* message) with all the *MSM* in L_0 and with the $PFSM_{Server}$.

T4: In state S_1 when the Send List (SL_0) is received from the $PFSM_{Server}$ requesting for the list L_0, it is forwarded to the $PFSM_{server}$ preceded by the RL_0 message.

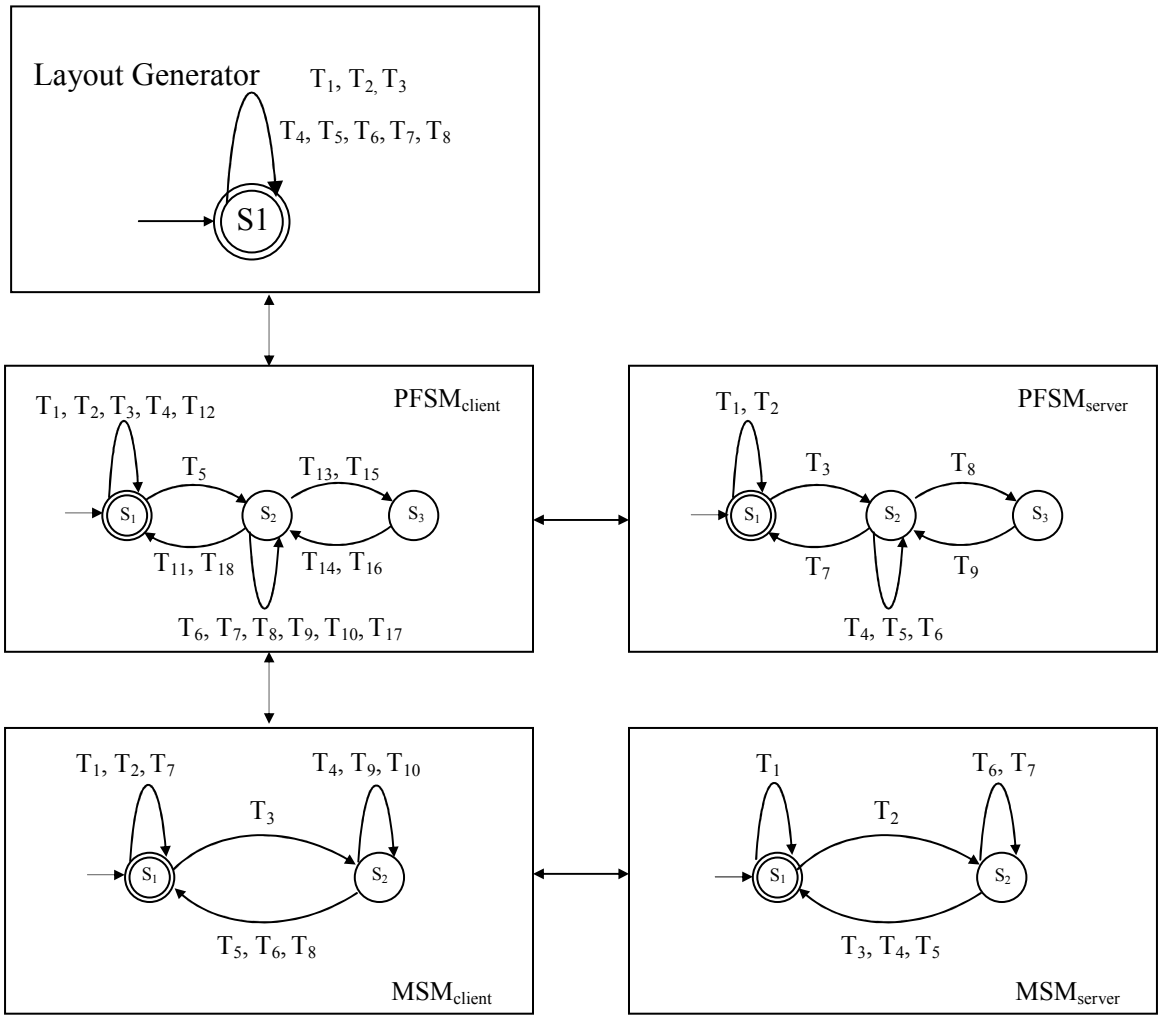

Figure 1: CAFSM Model

T5: In state S_1, when the *Connection Confirmation* (*CC*) message is received from all the *MSMs* in L_0 and the *CI* message is received from the $PFSM_{server}$, the $PFSM_{client}$ changes state to S_2 and sends the Send List (SL_1 and SL_2) messages to *LG*, requesting for lists L_1 and L_2.

T6: In state S_2, a message RL_1 from *LG*, indicating that the list L_1 is being sent, triggers this transition. The lists are received and a *Begin Transmission* (*BT*) message is sent to the $PFSM_{server}$. The list L_2 could possibly be empty at this stage (for the first synchronization point L_2 will definitely be empty as no media object ends there).

T7: This transition is triggered in state S_2, by a message RL_2 from *LG*, indicating that the list L_2 is being sent. At this stage the list L_1 could be possibly empty (for the last synchronization point L_1 will definitely be empty as no media object begins there).

T8: When in state S_2 and a SL_1 message from the $PFSM_{server}$ requesting for the list L_1 is received, the available list is forwarded.

T9: As mentioned earlier, *OK* messages are sent by each *MSM* as they receive sufficient data in their buffer during streaming. When in state S_2 and all the *OK* messages are received from *MSMs* of media objects that belong to the list L_1, the *Begin Presentation* (*BP*) message is sent to all the media objects that need to

commence. After sending the *BP* messages, the List L_1 is emptied and the request for the new list is sent.

T10: As the *Finish Presentation* (*FP*) messages are received in state S_2 from all the *MSMs* corresponding to the list L_2, they are eliminated from the list. When L_2 becomes empty, the request for the new list is forwarded to *LG*.

T11: At the end of the presentation, when there are no more synchronization points, the LG sends a *Disconnection Indication* (*DI*) in response to a request from the $PFSM_{client}$ for the lists of objects corresponding to the next synchronization point. At this instance, L_1 and L_2 are empty and the $PFSM_{client}$ changes state to S_1 from state S_2, and continues with the disconnection process by sending disconnection request (*DR*) messages to the $PFSM_{server}$ and $MSM(L_0)$ at the client.

T12: In state S_1, the $PFSM_{client}$ waits until it receives the Disconnection Indication (*DI*) from the $PFSM_{server}$ and *Disconnection Confirmations* (*DC*) from $MSM(L_0)$ and it remains in S_1 which is also considered as the final state of the system.

T13: In case the $PFSM_{client}$ receives a *fast-forward/rewind FR* message from the user while it is in state S_2, it sends a *Discard Message (DS)* to the $PFSM_{server}$ and to the *MSMs* of all participating media objects at the client. It also calculates the new index value that represents the point at which the presentation should resume and sends it to *LG* with a request for a new list that includes the media objects from the point requested by the user and then changes to state S_3. The lists L_1 and L_2 are discarded.

T14: In state S_3, the $PFSM_{client}$ receives the *FP* messages from the *MSMs* of all media objects at the client, indicating that they have successfully discarded all the previous data. It also receives a RL_1 message from the *LG* which then sends a new list L_1. Subsequently, the $PFSM_{client}$ sends a *BT* message to the $PFSM_{server}$ and changes the state to S_2.

T15: While in S_2, the $PFSM_{client}$ could receive an interaction in the form of *Pause (PS*) Presentation. The $PFSM_{client}$ merely forwards this message to the *MSMs* of all the media objects involved in the presentation and goes to the state S_3.

T16: While in S_3, the $PFSM_{client}$ could receive an interaction in the form of resume *RS* message which is normally preceded by a *PS* message. In this case, the $PFSM_{client}$ also forwards this message to the *MSM* of all the media objects involved in the presentation and goes to the state S_2.

T17: While in S_2, the $PFSM_{client}$ could receive an *Adaptation-based-Interaction (AI)* message and that could result in a change in the presentation layout. In this case, the $PFSM_{client}$ forwards the data relating to this interaction to the *LG* along with a request for a *New List* (*NL*) message and sends *DS* messages to the $PFSM_{server}$ and all its *MSMs*.

T18: In state S_2, when a request for disconnection from the user (represented here by a *Disconnection Request (DR)* message, in which the $PFSM_{client}$ sends to itself in order to further initiate the disconnection process) is received, it sends *DR* messages to the $PFSM_{server}$ and to the $MSM(L_0)$, and goes back to state S_1.

Server PFSM

The *PFSM* at the server side represents at any point of time the status of the ongoing transmissions. It is modelled using 3 states say S_1, S_2 and S_3, and 9 transitions as detailed below. Figure 1 shows the transition diagram for the $PFSM_{server}$. The transitions in the $PFSM_{server}$ require only the lists L' and L^* here contains only list L_1.

T1: The *Connection Request (CR)* initiated by the $PFSM_{client}$ triggers this transition in state S_1. At this stage, the list L_1 is empty. It responds with a Send List SL_0 message to the $PFSM_{client}$.

T2: The $PFSM_{client}$ sends a *Receive List* (RL_0) message followed by the list L_0. On receiving this message the $PFSM_{server}$ sends a (*CI*) message to the $PFSM_{client}$, and continues to remain in state S_1.

T3: When the $PFSM_{server}$ is in state S_1 and it receives *Connection Indication (CI*) messages from all the *MSMs*, it changes the state to S_2.

T4: While in state S_2, the $PFSM_{server}$ receives *Begin Transmission (BT)* message from the $PFSM_{client}$. It remains in the same state and makes a request to the $PFSM_{client}$ for the list L_1.

T5: In state S_2, a *Receive List* (RL_1) message from the $PFSM_{client}$ indicates that the list L_1 is being sent. A *BT* message is sent to each *MSM* in the received list.

T6: While in state S_2, as the *FT* messages are received from the *MSMs* corresponding to list L_1, they are eliminated from the list and a request for the next list SL_1 is transmitted.

T7: When in state S_2, on receiving the *Disconnection Request (DR)* from $PFSM_{client}$ and *Disconnection Indication (DI)* from $MSM(L_0)$ at the server, it sends a *DI* in turn to $PFSM_{client}$ and changes the state to S_1.

T8: The $PFSM_{server}$, on receiving the *DS* message, moves to state S_3 and discards the list L_1.

T9: When in state S_3, on receiving *FT* messages from the *MSMs* of all media objects involved in the presentation, it changes the state to S_2.

3.2 Media Schedule Managers : MSM

Formally, a *MSM* can be defined as a 3-tuple (S, Σ, δ), where S is a finite set of states $\{S_1, S_2\}$, Σ is a set of messages and $\delta : S \times 2^{\Sigma} \rightarrow S \times 2^{\Sigma}$ is a transition function that takes as input the state of the machine and the messages received and changes its state. In the subsequent paragraphs, the transitions associated with client and server *MSMs* are described.

Client MSM

T1: In state S_1, when the MSM_{client} receives a request for connection from its *PFSM*, it forwards the request to its peered *MSM* at the server.

T2: In state S_1, on receiving a *Connection Indication* (*CI*) from the MSM_{server}, the MSM_{client} sends a message confirming the connection to its *PFSM*. This completes connection establishment.

T3: In state S_1, on receiving sufficient data (which follows the *RD* message) from their peered *MSMs*, the MSM_{client} sends an *OK* message to the *PFSM* indicating that it is ready for the presentation, and changes its state to S_2.

T4: While in state S_2, the MSM_{client} receives a *BP* message from the *PFSM* and it commences its presentation.

T5: While in state S_2, after completion of the play out, it generates a message *FP* to itself which triggers a change of state to S1 and sends a *Finish Presentation (FP*) message to the $PFSM_{client}$.

T6: While in state S_2, the MSM_{client} may receive a *Disconnection Request (DR)*. This will change its state to S_1 and it will forward the request to its peered *MSM* at the server.

T7: At state S_1, the peered *MSM* replies with a *Disconnection Indication (DI*), and the MSM_{client} in turn sends a *Disconnection Confirmation (DC*) message to the $PFSM_{client}$.

T8: In state S_2, when the MSM_{client} receives the *Discard message (DS)* from the $PFSM_{client}$, it changes the state to S_1 and sends *Discard messages (DS*) to its peered MSM_{server}. It also sends a *FP* message to the $PFSM_{client}$ indicating that it has successfully discarded the current presentation data.

T9: In state S_2, when the MSM_{client} receives the *Pause (PS)* message from the $PFSM_{client}$, it forwards the message to its peered MSM_{server} at the server. It also sends a *FP* message to the *PFSM* indicating that it has successfully paused the presentation of the media object it represents.

T10: In state S_2, when the $PFSM_{client}$ receives a *Resume (RS)* message, it continues with the presentation of the media object.

Server MSM

T1: In state S_1, when the MSM_{server} receives the *CR* message from the MSM_{client}, it sends a *CI* message back to its peered MSM_{client} and also to the $PFSM_{server}$.

T2: In state S_1, on receiving a *BT* message from $PFSM_{server}$, the MSM_{server} changes its state to S_2 and starts transmitting the data to its peered MSM_{client}.

T3: In state S_2, at the end of transmission, it triggers itself with a *Finish Transmission (FT)* message and forwards the *FT* message, to the $PFSM_{server}$ and changes the state to S_1.

T4: When in state S_2 , if a message requesting disconnection *DR* is received, then the MSM_{client} sends a *Disconnection Indication (DI)* to the MSM_{client} and to the $PFSM_{server}$ and changes the state to S_1.

T5: In state S_2, when the MSM_{server} receives a *Discard message (DS*) from its MSM_{client}, it sends a *FT* message to the $PFSM_{server}$ and changes state to S_1.

T6: In state S_2, when the MSM_{server} receives a *Pause message (PS)* from the peer MSM_{client}, it sends a *Finish Transmission (FT)* message to the $PFSM_{server}$ and remains in S_2

T7: In state S_2, when the MSM_{server} receives a *Resume message (RS)* from its peer MSM_{client}, it continues with its transmission.

3.3 Layout Generator : LG

Formally, the layout generator *LG* can be defined as a 3-tuple $(\{S_1\}, \Sigma, \delta)$, where S_1 is its only state, Σ is a set of messages, and $\delta : S \times 2^{\Sigma} \rightarrow S \times 2^{\Sigma}$ is a transition function which takes as input the messages received to trigger new messages and perform appropriate actions. The transitions are shown in Figure 1.

T1: When the *LG* receives the *CR* message from the $PFSM_{client}$, it sends a *CI* message back to the $PFSM_{client}$.

T2: When the *LG* receives a SL_0 message requesting for the lists L_0, the receive lists RL_0 message is sent followed by the data (i.e. L_0).

T3: When the *LG* receives a SL_1 message requesting for the lists L_1, the receive lists RL_1 message is sent followed by the data (i.e. L_1).

T4: When the *LG* receives a SL_2 message requesting for the lists L_2, the receive lists RL_2 message is sent followed by the data (i.e. L_2).

T5: If the user abruptly ends the presentation, then the *LG* receives a *Disconnection Request* from the $PFSM_{client}$ and sends a *Disconnection Indication* back to the $PFSM_{client}$.

T6: On the other hand, if the $PFSM_{client}$ requests for the next set of lists while there are no more synchronization points, then the *LG* sends a *DR* messages to itself which triggers further disconnection by sending *Disconnection Indication* message to the $PFSM_{client}$

T7: Here, the request from the *PFSM* for a new list is made following a fast-forward/rewind request made by the user. The *PFSM* also computes the index value to indicate the location where the presentation should commence, and sends this as the index value *i*. The *LG* sends a new list replacing the existing list L_1.

T8: The $PFSM_{client}$ requests for the next set of lists in the form of the message new list *NL*, followed by data. This represents an (adaptation-based) interaction with

the presentation which could lead to changes in the spatio-temporal relations and may cause inconsistency to occur. Thus, the consistency checking algorithm [6] is employed by the *LG* to produce a consistent set of constraints with the *LG* using the modified layout to generate the next list L_1 as data to the $PFSM_{client}$.

4. ANALYSIS OF THE PROPOSED MODEL

The salient feature of the proposed model is the design of the set of transitions that guide the runtime module to ensure a synchronised play-out of the presentation. The set of transitions, presented along with each of the *CAFSMs* in the previous section, also define the sequence of the messages that are used to control the flow of the presentation. The proposed model is deterministic i.e there is only one path that can be traversed for a particular event. For multimedia presentations that have rapidly changing objects, there would be several synchronisation points leading to several states and transitions in the existing *DEFSM* model, while the proposed *CAFSMs* have a fixed number of states and transitions irrespective of the number of synchronisation points. The response time, in case of interactions with the presentation as well as navigations like skip, would then have a worst-case complexity of *O(n)*, where *n* represents the number of transitions to search from in the existing *DEFSM* models. But in the proposed *CAFSM* model, there are only 3 states and 18 transitions (at the most) to search from at any point of time, leading to a system with an efficient response time to interactions. The proposed *CAFSM* model has however been checked using the model checker *SPIN* and the simulation results obtained have correlated with the theoretical results.

5. CONCLUSIONS

This paper presents the design of a multimedia synchronisation mechanism using a formal approach. The proposed *CAFSM* model with the help of its message passing scheme ensures a synchronised play out of the presentation and is also efficient in its time and space complexity in comparison with the existing formal approaches that define similar synchronization mechanisms. The proposed model can be further enhanced to handle adaptive presentations, buffer management and efficient retrieval policies for bandwidth restricted mobile multimedia communication.

6. ACKNOWLEDGEMENT

We would like to acknowledge and thank the Defence Research Development Organisation (DRDO), New Delhi, India for granting us Extramural Research Funds for carrying out this research work. This work is part of the research project titled "*The design and development of a multimedia presentation system that streams MPEG-21 compatible media-on-demand*".

7. REFERENCES

[1] E. Bertino and E. Ferrari. Temporal synchronization models for multimedia data. *IEEE Trans. on Knowl. and Data Eng.*, 10(4):612-631, 1998.

[2] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323-342, 1983.

[3] J. Canavan. Personalised e-learning through learning style aware adaptive systems: MS thesis, 2004.

[4] H. Chorfi and M. Jemni. Perso: Towards an adaptive e-learning system. *Journal of Interactive Learning Research*, 15(4):433-447, 2004.

[5] O. Conlan and V. Wade. Evaluation of APeLS - an adaptive eLearning service based on the multi-model, metadata-driven approach. *Adaptive Hypermedia and Adaptive Web-Based Systems*, 3137:291-295, 2004.

[6] S. Elias, K. S. Easwarakumar, and R. Chbeir. Dynamic consistency checking for temporal and spatial relations in multimedia presentations. In *Proceedings of The 21st ACM Symposium on Applied Computing*, pages 1380-84, Dijon, France, April 2006.

[7] T. Flor. Experiences with adaptive user and learning models in elearning systems for higher education. *Journal of Universal Computer Science*, 10(1):58-72, 2004.

[8] C. Gütl, M. Pivec, C. Trummer, V. M. García-Barrios, F. Modritscher, J. Pripfl, and M. Umgeher. Adele (adaptive e-learning with eye-tracking): Theoretical background, system architecture and application scenarios. *European Journal of open, Distance and E-learning(EURODL)*, 2004.

[9] C.-M. Huang and C. Wang. Synchronization for interactive multimedia presentations. *IEEE Multimedia*, pages 44-61, Oct. 1998.

[10] C. M. Huang, C. Wang, and C. H. Lin. Interactive multimedia synchronisation in the distributed environment using the formal approach. *IEE Proceedings - Software*, 147(4):131-146, 2000.

[11] M. Kock. Computational intelligence for communication and cooperation guidance in adaptive e-learning systems. In *LWA*, volume 448 of *Technical Report*, pages 32-34. Department of Computer Science, University of Wurzburg, Germany, 2008.

[12] F. Modritscher, V. Garcia Barrios, and C. Gutl. Enhancement of scorm to support adaptive e-learning within the scope of the research project adele. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2004*, pages 2499-2505, Washington, DC, USA, 2004.

[13] M. Meccawy, P. Brusilovsky, H. Ashman, M. Yudelson, and O. Scherbinina. Integrating interactive learning content into an adaptive e-learning system: Lessons learned. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2007*, pages 6314-6319, Quebec City, Canada, 2007.

[14] H. D. Surjono. Empirical evaluation of an adaptive e-learning system and the effects of knowledge, learning styles and multimedia mode on student achievement. In *Proceedings of the UiTM International Conference on E-Learning*, pages 12-14, Shah Alam, Malaysia, December 2007.

[15] C.-C. Yang, Y.-C. Wang, and C.-W. Tein. Synchronization modeling and its application for SMIL 2.0 presentations. *The journal of Systems and Software*, 80:1142-1155, 2007.

[16] A. Zhang. Synchruler: A rule-based flexible synchronization model with model checking. *IEEE Trans. on Knowl. and Data Eng.*, 17(12):1706-1720, 2005. Member-Aygun, Ramazan Savas.

[17] www.engineeringchallenges.org

[18] Mödritscher, F., Garcia Barrios, V.M. & Gütl, C. (2004). Enhancement of SCORM to support adaptive E-Learning within the Scope of the Research Project AdeLE. In Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and

Higher Education 2004 (pp. 2499-2505). Chesapeake, VA: AACE.

APPENDIX

A. CLIENT PFSM

T1: $\delta(S_1, (SP, PFSM_{client}), \phi, \phi) = (S_1, (CR,LG), \phi, \phi)$
T2: $\delta(S_1, \{(CI,LG)\}, \phi, \phi) = (S_1, (SL_0), LG), \phi, \phi)$
T3: $\delta(S_1, \{(RL_0,LG),L_0,\phi) = (S_1, \{(CR, PFSM_{server}), (CR, MSM(L_0))\}, \phi, \phi)$
T4: $\delta(S_1, (SL_0, PFSM_{server}), \phi, \phi) = (S_1, \{(RL_0, PFSM_{server}, LG), (SL_2, LG)\},L_0, \phi)$
T5: $\delta(S_1, \{(CC,MSM(L_0)), (CI, PFSM_{Server})\}, \phi, \phi) = (S_2, \{(SL_1,LG), (SL_2, LG)\}, \phi, \phi)$
T6: $\delta(S_2, (RL_1,LG), \phi,L_2) = (S_2, (BT, PFSM_{Server}),L_1,L_2)$
T7: $\delta(S_2, (RL_2,LG),L_1, \phi) = (S_2, \phi, L_1, L_2)$
T8: $\delta(S_2, (SL_1, PFSM_{Server}), L_1, L_2) = (S_2, (RL_1, PFSM_{Server}),L_1, L_2)$
T9: $\delta(S_2, (OK,MSM(L_1),L_1,L_2) = (S_2, \{(BP,MSM(L_1), (SL_1,LG)\}, \phi,L_2)$
T10: $\delta(S_2, (FP,MSM(L_2)),L_1,L_2) = (S_2, (SL_2,LG),L_1, \phi)$
T11: $\delta(S2, (DI,LG), \phi, \phi) = (S_1, \{(DR, PFSM_{Server}), (DR,MSM(L_0)\}, \phi, \phi)$
T12: $\delta(S_1, \{(DI, PFSM_{Server}), (DC,MSM(L_0)\}, \phi, \phi) = (S_1, \phi, \phi, \phi)$
T13: $\delta(S_2, (FR, PFSM_{Client}),L_1,L_2) = (S_3, \{(DS, PFSM_{Server}), (DS,MSM(L_0), ((NL, i),LG)\}, \phi, \phi)$
T14: $\delta(S_3, \{(FP,MSM(L_0), (RL_1,LG)\}, \phi, \phi) = (S_2, (BT, PFSM_{Server}),L_1, \phi)$
T15: $\delta(S_2, (PS, PFSM_{client}),L_1,L_2) = (S_3, \{(PS,MSM(L_0)), (PS, PFSM_{Server})\}, \phi, \phi)$
T16: $\delta(S_3, (RS, PFSM_{client}),L_1,L_2) = (S_2, \{(RS, PFSM_{Server}), (RS,MSM(L_0))\}, \phi, \phi)$
T17: $\delta(S_2, (AI, PFSM_{client}),L_1,L_2) = (S_2, \{(DS, PFSM_{server}), (DS,MSM(L_0), ((NL, data),LG)\}, \phi, \phi)$
T18: $\delta(S_2, (DR, PFSM_{client}),L_1,L_2) = (S_1, \{(DR, PFSM_{server}), (DR,MSM(L_0)), (DR,LG)\}, \phi, \phi)$

B. SERVER PFSM

T1: $\delta(S_1, (CR, PFSM_{client}), \phi) = (S_1, (SL_0, PFSM_{Client}), \phi)$
T2: $\delta(S_1, (RL_0, PFSM_{client}),L_0) = (S_1, \phi, \phi)$
T3: $\delta(S_1, (CI, MSM(L_0)), \phi) = (S_2, (CI, PFSM_{client}), \phi)$
T4: $\delta(S_2, (BT, PFSM_{client}), \phi) = (S_2, (SL_1, PFSM_{client}), \phi)$
T5: $\delta(S_2, (RL_1, PFSM_{client}), \phi) = (S_2, (BT, MSM_{L1}),L_1)$
T6: $\delta(S_2, (FT, MSM(L_1)),L_1) = (S_2, (SL_1, PFSM_{client}), \phi)$
T7: $\delta(S_2, \{(DR, PFSM_{client}), (DI, MSM(L_0))\}, \phi) = (S_1, (DI, PFSM_{client}), \phi)$
T8: $\delta(S_2, (DS, PFSM_{client}),L_1) = (S_3, \phi, \phi)$
T9: $\delta(S_3, (FT, MSM(L_0)), \phi) = (S_2, \phi, \phi)$

C. CLIENT MSM

T1: $\delta(S_1, (CR, PFSM_{client})) = (S_1, (CR, MSM_{server}))$
T2: $\delta(S_1, (CI, MSM_{server})) = (S_1, (CC, PFSM_{client}))$
T3: $\delta(S_1, (Data, MSM_{server}) = (S_2, (OK, PFSM_{client}))$
T4: $\delta(S_2, (BP, PFSM_{client})) = (S_2, \phi)$
T5: $\delta(S_2, (FP, MSM_{client})) = (S_1, (FP, PFSM_{client}))$
T6: $\delta(S_2, (DR, PFSM_{client})) = (S_1, (DR, MSM_{server}))$
T7: $\delta(S_1, (DI, MSM_{server})) = (S_1, (DC, PFSM_{client}))$
T8: $\delta(S_2, (DS, PFSM_{client})) = (S_1, \{(DS, MSM_{server}), (FP, PFSM_{client})\})$
T9: $\delta(S_2, (PS, PFSM_{client})) = (S_2, \{(PS, MSM_{server}), (FP, PFSM_{client})\})$
T10: $\delta(S_2, (RS, PFSM_{client})) = (S_2, (RS, MSM_{server}))$

D. SERVER MSM

T1: $\delta(S_1, (CR, MSM_{client})) = (S_1, \{(CI, MSM_{client}), (CI, PFSM_{server})\})$
T2: $\delta(S_1, (BT, PFSM_{server})) = (S_2, (RD, MSM_{Client}))$
T3: $\delta(S_2, (FT, MSM_{server})) = (S_1, (FT, PFSM_{server}))$
T4: $\delta(S_2, (DR, MSM_{client})) = (S_1, \{(DI, MSM_{client}), (DI, PFSM_{server})\})$
T5: $\delta(S_2, (DS, MSM_{client})) = (S_1, (FT, PFSM_{server}))$
T6: $\delta(S_2, (PS, MSM_{client})) = (S_2, (FT, PFSM_{server}))$
T7: $\delta(S_2, (RS, MSM_{client})) = (S_2, (Data, MSM_{client}))$

E. LAYOUT GENERATOR

T1: $\delta(S_1, \{(CR, PFSM_{client})\}) = (S_1, (CI, PFSM_{client}))$
T2: $\delta(S_1, \{(SL_0, PFSM_{client}\}) = (S_1, (RL_0, PFSM_{client}))$
T3: $\delta(S_1, \{(SL_1, PFSM_{client}\}) = (S_1, (RL_1, PFSM_{client}))$
T4: $\delta(S_1, \{(SL_2, PFSM_{client}\}) = (S_1, (RL_2, PFSM_{client}))$
T5: $\delta(S_1, \{(DR, PFSM_{client})\}) = (S_1, (DI, PFSM_{client}))$
T6: $\delta(S_1, \{(DR,LG)\}) = (S_1, (DI, PFSM_{client}))$
T7: $\delta(S_1, ((NL, i), PFSM_{client})) = (S_1, (RL_1, PFSM_{client}))$
T8: $\delta(S_1, ((NL, data), PFSM)) = (S_1, (RL_1, PFSM_{client}))$
T9: $\delta(S_1, ((NL, data), PFSM)) = (S_1, (RL_1, PFSM_{client}))$