# Modeling and Analysis of Collaborative Virtual Environments by using Extended Fuzzy-Timing Petri Nets

Y. Zhou, T. Murata, and T. DeFanti

Department of Electrical Engineering and Computer Science
University of Illinois at Chicago
Chicago, Illinois 60607-7053 USA
{yzhou1, murata, tom}@eecs.uic.edu

**Abstract.** Virtual Reality (VR) systems (the CAVE$^{TM}$[1]) generate images in real-time on the basis of the viewer's view in the virtual world, so that the viewer percepts a real-life, three-dimensional view of a given scene. The concurrency and real-time features in virtual environments systems make them difficult to design, implement and test. Collaborative Virtual Environment (CVEs) makes it more complicated by adding network consideration into VRs. CVEs demand high Quality-of-Service (QoS) requirements on the network to maintain natural and real-time interactions among users. By using formal methods to model CVEs and analyze their real-time behavior, we can evaluate the network effects on CVEs and the performance of CVEs. To model temporal uncertainties in CVEs, we propose an extension of Fuzzy-Timing Petri Nets (EFTN) in this paper. We give our EFTN models for the CAVE and the NICE (Narrative Immersive Constructionist/Collaborative Environments) and we analyze the network effects on the NICE and the dynamic performance of the NICE.

## 1 Introduction

Virtual Reality (VR) can be defined as interactive computer graphics that provide viewer-centered perspective, large field of view and stereo. The CAVE$^{TM}$[1] (Cave Automatic Virtual Environment) ([2], [3], [4]) is a virtual reality environment designed and implemented at the Electronic Visualization Laboratory at the University of Illinois at Chicago. The CAVE, as shown in Fig. 1, is a surround screen, surround sound, projection based virtual reality environment system. The actual environment is a 10x10x10 foot cube, where images are rear-projected in stereo on 3 walls (front wall, left wall, and right wall), and down-projected onto the floor. (The floor can be called as floor wall. So there are totally 4 walls.) The 4 walls display computer generated images in real-time on the basis of the viewer's view in the virtual world, so that the viewer percepts a real-life, three-dimensional view of a given scene. The viewer wears head tracker and holds a wand (the CAVE input device). So the viewer's head and hand position can be detected and the viewer can grab objects with the wand. The three-dimensional view is generated in real-time to match the viewer's perspective in the virtual world.

[1] CAVE$^{TM}$ is a registered trademark of the Regents of the University of Illinois.

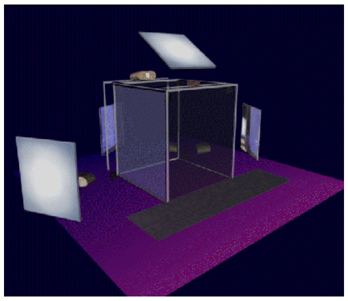

**Fig. 1.** The CAVE

Because of the concurrency and real-time features in virtual environments systems, it is difficult to design, implement and test VRs. Collaborative Virtual Environment (CVE) makes it more complicated by adding network consideration into VRs. It allows people in remote virtual environments to learn from each other, work together on designing systems, or perform a complex group task together over networks.

Narrative Immersive Constructionist/Collaborative Environments (NICEs) ([2], [3]), developed by Electronic Visualization Laboratory at the University of Illinois at Chicago, are collaborative learning environments where children can do gardening and learning cooperatively. In the NICE, children located in distributed virtual environments (e.g., CAVEs), can take care of a virtual garden together in the center of a virtual island. The children, represented by avatars, collaboratively plant, grow, and pick vegetables and flowers. They make sure that the plants have sufficient water, sunlight, and space to grow, and they keep hungry animals away from sneaking in the garden and eat the plants. Collaboration in NICE refers to communication and shared experience between children who are in distributed locations and tending the same virtual garden.

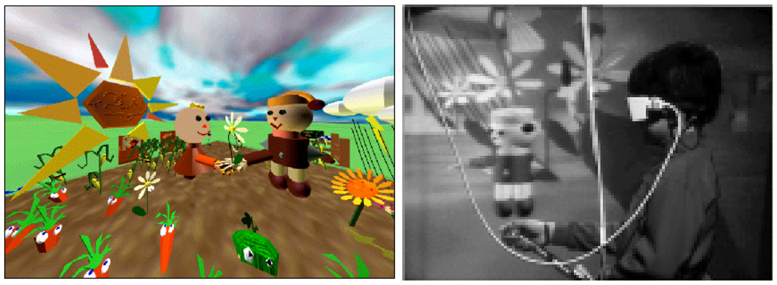

**Fig. 2.** (a) Jim (the avatar) is handing a flower to Eddie (the other avatar); (b) A child is interacting with an avatar in the CAVE.

The NICE uses a central server to simulate the garden and maintain consistency across all the participating virtual environments. Each virtual environment (VE) transmits the local entity information (the position and action of the child in current VE) through the network to remote sites. Meanwhile, the central server sends the world information (the information about the garden) to each site so that all sites can share the same information.  It is very important to draw remote entities in real-time in each VE so that the user will not notice any difference between the local and remote entities in the environment.

CVEs demand high Quality-of-Service (QoS) requirements on the network to maintain natural and real-time interactions among users. QoS refers to the requirements on network latency and jitter (the variability in network latency.)  By using formal methods to model CVEs and analyze their real-time behavior, we can evaluate the network effects on CVEs and the performance of CVEs. Petri Nets have rigorous analysis capability and have been shown useful for assuring the reliability and correctness of concurrent systems. In order to model and analyze real-time systems, various timed extensions of Petri Nets have been proposed. However, many real-time systems have temporal uncertainty. For example, the time duration of rendering an image for a wall in CAVE varies on the complexity of the geometric objects in the image, and the network delays in CVEs vary in a large range. To deal with temporal uncertainties in real-time systems, Murata [7] proposed Fuzzy-Timing High-Level Petri Nets (FTHNs) to model time explicitly in terms of fuzzy set theory. FTHNs model temporal uncertainties in real-time systems, and provides possibility distributions of events. So FTHNs can capture all temporal uncertainties in CVEs and they would be suitable models for CVEs.

This paper is organized as follows: Section 2 reviews Fuzzy-Timing Petri Nets and proposes an extension of Fuzzy-Timing Petri Nets (EFTN); Section 3 gives our EFTN models for the CAVE; Section 4 analyzes the dynamic behavior of our EFTN model of the CAVE; Section 5 discusses our EFTN models for the NICE; And section 6 concludes the paper and gives our future research plan.

## 2 Fuzzy-Timing Petri Nets and Extended Fuzzy-Timing Petri Nets

The main features of Fuzzy-Timing High-Level Petri Nets (FTHNs) are the following four fuzzy set theoretic functions of time called fuzzy timestamp, fuzzy enabling time, fuzzy occurrence time and fuzzy delay. A fuzzy timestamp $\pi(\tau)$ is associated with each token and each place, and $\pi(\tau)$ is a fuzzy time function or possibility distribution giving the numerical estimate of the possibility that a particular token arrives at time $\tau$ in a particular place. In FTHNs, arcs (t, p) from transitions t to places p are associated with fuzzy delays $d_{tp}(\tau)$. For simplicity, trapezoidal possibility distributions specified by the 4-tuple $(\pi_1, \pi_2, \pi_3, \pi_4)$, are used to represent fuzzy time functions.

The formal definition of FTHNs and the method to compute and update fuzzy enabling time and fuzzy occurrence time when a transition firing occurs, are given in [7]. FTHNs provide additional information on partial ordered events in terms of their degrees of possibilities, instead of transforming them into a total ordering. The computations involved in FTHNs are basically repeated additions and comparisons of real numbers and are necessary only for certain finite firing sequences, and need not generate the entire state space. Thus these computations can be done very fast and thus FTHNs are suited for estimating the performance of time-critical systems.

A Fuzzy-timing Petri Net (FTN) [11] model is an unfolded version of the fuzzy-timing high-level Petri Net (FTHN).

We extend FTN by integrating FTN with Merlin's Time Petri Net [5]. We define an Extended Fuzzy-Timing Petri Net (EFTN) model as a 6-tuple (P, T, A, D, FT, CT), where: (P, T, A, D, FT) is a Fuzzy Timing Petri Net, with the default value of $d_{tp}(\tau)$ being (0,0,0,0); CT: $T \to Q^+ \times (Q^+ \cup \infty)$ is a mapping from the transition set T to firing intervals: i.e., each transition is associated with a firing interval $[\alpha, \beta]$, where the default interval is [0, 0] (a transition fires as soon as it is enabled). If a transition t is enabled at time instant $\tau$, t may not fire before time instant $\tau+\alpha$, and t must fire before or at time instant $\tau+\beta$. A transition firing itself is an atomic event and takes zero time. (CT is taken from Merlin's Time Petri Net [5].)

Now, in EFTN, the *fuzzy enabling time* $e_t(\tau)$ of transition t is still computed by $e_t(\tau) = \mathit{latest}\{\pi_i(\tau), i = 1, 2, ..., n\}$. When there are m transitions enabled with their *fuzzy enabling times*, $e_i(\tau)$, $i = 1, 2, ..., t, ..., m$, and $CT(t_i) = p_i[\alpha_i, \beta_i]$, we compute the *fuzzy occurrence time* $o_t(\tau)$ of transition t whose *fuzzy enabling time* $e_t(\tau)$, as follows: $o_t(\tau) = \mathit{min}\{e_t(\tau) \oplus p_t(\alpha_t, \alpha_t, \beta_t, \beta_t), \mathit{earliest}\{e_i(\tau) \oplus p_i (\alpha_i, \alpha_i, \beta_i, \beta_i), i = 1, 2, ..., t, ..., m\}\}$. We can see that, $o_{t1}(\tau) < o_{t2}(\tau)$, when $e_{t1}(\tau) = e_{t2}(\tau)$ but $[\alpha_1, \beta_1] < [\alpha_2, \beta_2]$. ($\oplus$ is the extended addition defined as $\pi_{tp}(\tau) = o_t(\tau) \oplus d_{tp}(\tau) = h_1(o_1,o_2,o_3,o_4) \oplus h_2(d_1,d_2,d_3,d_4) = \mathit{min}\ (h_1, h_2)\ (o_1 + d_1, o_2 + d_2, o_3 + d_3, o_4 + d_4)$) The main utility of the above firing intervals of transitions is to give a firing possibility and priority among transitions in conflict. This is very useful for modeling real-time systems.

## 3 EFTN models for the CAVE

The CAVE has three main subsystems: (Fig. 4 gives an EFTN model for the CAVE.)

- Tracker subsystem: which obtains data about the position of the viewer's head and hand. Since the viewer wears a head tracker and holds a wand where sensors are located, his position is detected by the tracker operating at 96 HZ sampling frequency. The tracking sample is obtained every 10.4 ms when the monitor signal arises [6].
- Main subsystem: which creates images to be displayed on the walls of the CAVE. There are four graphic pipelines working concurrently. Each of them is used to render the image on one wall. The CAVE implementation uses double buffering between the main subsystem and the display subsystem. While the main subsystem is writing into one buffer, the display subsystem reads from the other buffer. The buffer swapping is synchronized by a monitor signal at 46 HZ frequency. Once images for all 4 walls have been rendered, buffer swapping takes place at the next leading edge of the monitor signal if the display subsystem is also ready to swap buffer.
- Image display subsystem: which draws the images on the four walls. When the drawings of 4 images are all finished, the display subsystem is ready to swap buffer.

## 4 The Analysis of EFTN models for the CAVE

– Reduction Rules for EFTN

In order to analyze EFTN models, we introduce two reduction rules [1] for EFTNs in this section. Our reduction rules can reduce the size of EFTN models and preserve safeness, deadlock and timing properties of EFTN. We only illustrate the two rules applied to our EFTN models for the CAVE in Fig. 5. We plan to give all of our reduction rules and the formal proofs in our future paper.

– Behavior of EFTN models for the CAVE: After we apply our reduction rules to the EFTN model, the reduced EFTN model is shown as Fig. 6.

Transition Swap_and_draw and Swap_Signal_passed are in conflict. The possibility of Swap_and_draw fires instead of Swap_Signal_passed is

possibility($\pi_{\text{Complete\_render\_4walls}}(\tau) < e_{\text{Swap—Signal\_passed}}(\tau) \oplus (\varepsilon 4, \varepsilon 4, \varepsilon 4, \varepsilon 4)$)
= possibility((5,10,18,28) < (21,21,21,21))
= shaded_area/area_trapizoidal(5,10,18,28) = 0.842.

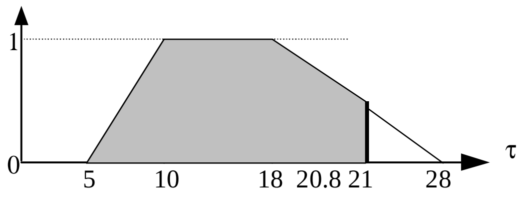

**Fig. 3.** Possibility of Swap_and_draw fires instead of Swap_Signal_passed

If Swap_Signal_passed fires, the display subsystem won't begin to draw any image until the next monitor signal comes. This Possibility decides that the delay that the user's movement being reflected on the walls is around 84 ms or 104 ms. possibility(delay $\approx$ 84ms) = 0.842.

We use Design/CPN [14] to simulate our EFTN model for the CAVE. By recording the timestamp that a token i generated by firing Hand_Wand_Input and the timestamp that the wall drawing is completed by using data represented by token i, we can get the delay. In our simulation, the delay is around 84 ms for 5739 times in a total of 6843 delay times we recorded. The simulation result is consistent with our possibility analysis.

## 5 EFTN models for the NICE

The main distributed components of the NICE consist of the garden simulation server, the WWW servers from which models are downloaded, and the NICE clients [2]. The NICE used an unreliable protocol (either multicasting or UDP) to share avatar information from magnetic trackers in remote locations, a reliable socket connection (TCP protocol) to share world (garden) state information and a reliable connection dynamically download models from WWW servers using the HTTP 1.0 protocol. The NICE server supports both the garden's simulation and broadcasting avatar information.

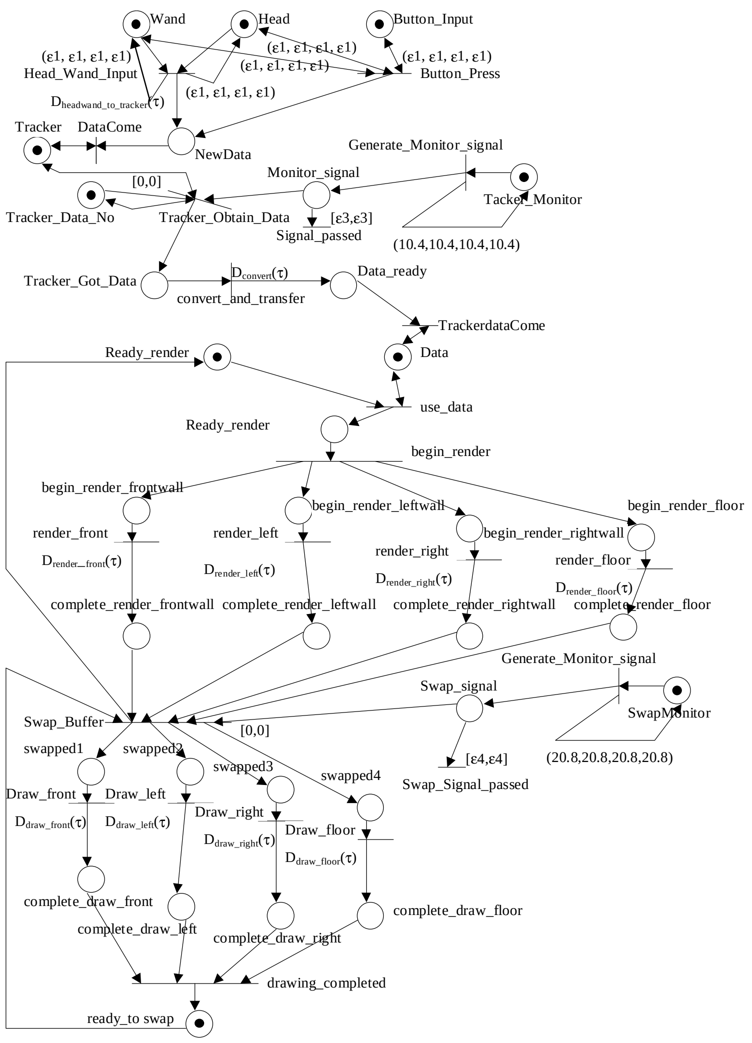

**Fig. 4.** An EFTN model for the CAVE, where the timestamps of tokens arriving in Head, Wand, Button_Input, TrackerMonitor and SwapMonitor at initial state are $\pi_{Head}(\tau) = \pi_{Wand}(\tau) = \pi_{Button\_Input}(\tau) = (0,0,0,0)$, $\pi_{TrackerMonitor}(\tau) = (10.4,10.4,10.4,10.4)$, and $\pi_{SwapMonitor}(\tau) = (20.7,20.8,20.8,20.8)$. The delay for Head and Wand data arriving at the Tracker is $D_{headwand_to_tracker}(\tau) = (50,50,50,50)$ms. ε3 = 0.2 ms. $D_{convert}(\tau) = (10,10,10,10)$ms. $D_{render\_front}(\tau)$, $D_{render\_left}(\tau)$, $D_{render\_right}(\tau)$, and $D_{render\_floor}(\tau)$ are the fuzzy delays of rendering images for front wall, left wall, right wall, and the floor. Assume $D_{render_front}(\tau) = (5,10,18,28)$ms, and $D_{render_left}(\tau) = D_{render_right}(\tau) = D_{render_floor}(\tau) = (4,6,8,10)$ms, since the image on the front wall is usually more complicated than the ones on other walls. The delay for converting and transferring tracker data to the computer is $D_{convert}(\tau) = (10,12,18,20)$ms, the delay for drawing images on each wall is $D_{draw_front}(\tau) = D_{draw_left}(\tau) = D_{draw_right}(\tau) = D_{draw_floor}(\tau) = (2,2,2,2)$ms, and ε4 = 0.2 ms (a short time period that the monitor signal lasts).

The Information Request Broker (IRB) is the core of all client and server applications in the NICE. An IRB is an autonomous repository of persistent data that is accessible by a variety of networking interfaces. A key is a handle to a storage location in an IRB's database. Keys are uniquely identified across all IRBs. A local key can initiate and accept multiple linkages to and from other remote IRBs. Any modifications that are made to one key will automatically be propagated to all the other linked keys.

The garden server is an IRB with two main keys: an incoming message key and an outgoing message key. Each client has a key on the server to hold its avatar-state information. Any changes to client1 (avatar1) will be sent by the client1 to the avatar1-state key on the sever. Another client (e.g. avatar2) will get the state of client1 by subscribing to the avatar1-state key on the server. If the local avatar has any action changing the garden (e.g., plant a tree), the local VE will send a message from the local OUT Key to the server's IN key. Then the garden server updates the world state and sends the new world state information to each clients via the server's OUT keys. The garden world evolves itself as the plant grows, the weather changes, and animals appear. So the server sends each client the new world state information once it updates. Here we give the EFTN model for 2 existing NICE clients communicating with each other and with the server as shown in Fig. 7.

Fig. 7 shows the communication interfaces of the NICE clients and server. In the NICE, a local VE (e.g., CAVE) will need local avatar state information, remote avatars state information, and world state information to render images. The EFTN model for a CAVE in the NICE as a distributed component is shown in Fig. 8. After we put the communication interface and internal structure of distributed CAVEs together, we may analyze the network effects on the NICE and the dynamic performance of the NICE.

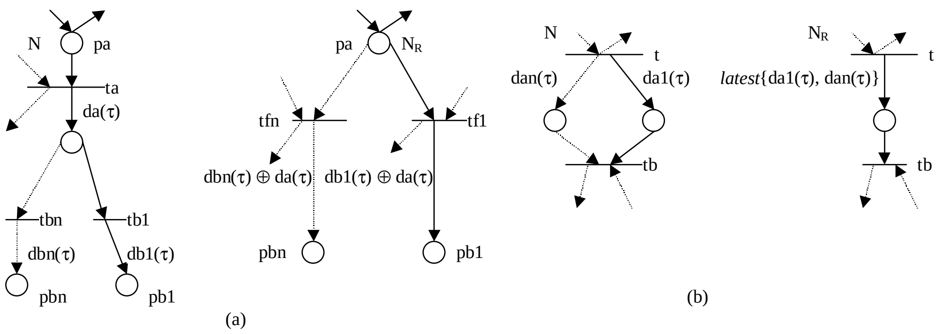

**Fig. 5.** (a) Post-fusion (post-fuse transition ta with tbs); (b) Parallel fusion of places.

Unreliable protocols (e.g. UDP) are used for the transmission of avatar/state information (remote tracker data). That is because: 1. The loss of one tracker data is usually followed shortly afterwards by newer ones, and 2. Unreliable protocols have a lower latency and utilize lower bandwidth than reliable protocols. However, UDP protocol is unordered. Using unordered remote tracker data will make the remote avatar jump back and forth on the local screen. Fig. 9 shows the time of avatar2's original movement and the time that movement is displayed on NICE client1's screen.

We can see that the remote avatar's display jumps back and forth. To avoid the jumping back behavior, the NICE currently uses a filter to accept remote avatar data in increasing order. In Fig. 8, the transition Got_new_avatar2_data works as a filter. Fig. 10 shows the display behavior using the filter. Now the jumping back phenomenon is eliminated. However, one early arriving remote avatar tracker data will make the filter discard all remote tracker data that are sent before, but received later than that early arriving data. From Fig. 10, we can see that the display of remote avatar is not very smooth. To display the remote avatar's movement more smoothly, we can use a buffer to store the incoming remote avatar state information sent after the last one used for local display. And we use the remote avatar's state information in smooth gap. The EFTN model and the simulation result of using this new strategy are to be included in [13].

Part of simulation results of our EFTN models for the NICE are shown in Figs. 11, and 12. Fig. 11 shows the distribution of the delay that remote tracker information being displayed on local screens. And Fig. 12 shows the distribution of the time that remote tracker data lags behind the local data. CVEs demand high requirement on network delay and jitter so that remotely distributed users' collaboration won't be disturbed. By using reduction, simulation, or occurrence graph on our EFTN model, the network effects on CVEs can be easily evaluated. More timing analysis (simulation and occurrence graph analysis), and our EFTN model for the TCP protocol and effects of using TCP in CVEs will be included in [13].

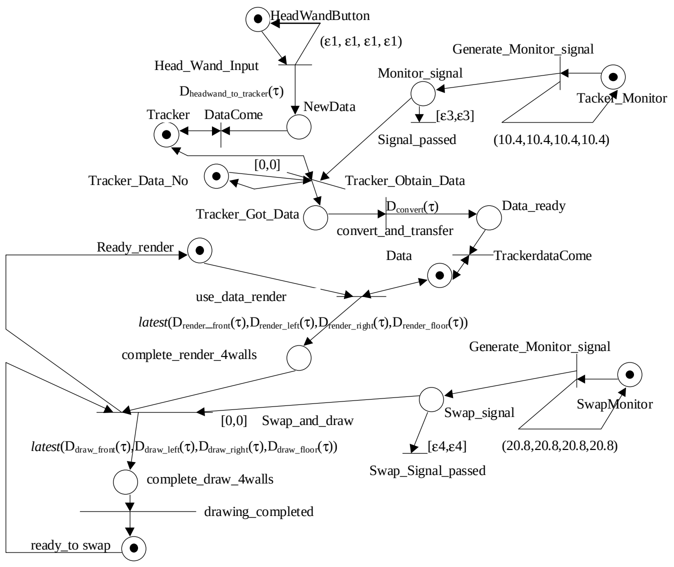

**Fig. 6.** The reduced EFTN model, where *latest*($D_{render\_front}(\tau)$,$D_{render\_left}(\tau)$ , $D_{render\_right}(\tau)$, $D_{render\_floor}(\tau)$) = *latest*((5,10,18,28), (4,6,8,10), (4,6,8,10), (4,6,8,10)) = (5,10,18,28)ms, *latest*($D_{draw\_front}(\tau)$, $D_{draw\_left}(\tau)$, $D_{draw\_right}(\tau)$, $D_{draw\_floor}(\tau)$) = (2,2,2,2)ms, and $D_{convert}(\tau)$ =(10,10,10,10)ms.

## 6 Conclusion and Future Work

Our EFTN models for the CAVE and NICE and the analysis of our EFTN models for the CAVE, indicate that EFTNs are powerful to specify and verify VRs. EFTNs can capture the temporal uncertainties in CVEs. By simulating our EFTN models, we can analyze the network effects on CVEs and the dynamic performance of CVEs.

In section 4, we show a simple example of possibility analysis of our EFTN model for the CAVE. EFTN models can give information on partial ordered events in terms of their degrees of possibilities. The possibility analysis is based on model checking on transition firing sequences [11] or occurrence graphs. The possibility analysis of our EFTN model for the NICE (and other CVEs) is to be included in our future paper.

Experiment results in [9] indicate that TCP's reliable and ordered behavior greatly increases the average network latency and jitter. Designing new transport layer protocol suitable for transmitting world state information (minimize the jitter) will be desirable. We plan to propose new protocols, model and analyze theirs performance and effects on CVEs in our future paper.

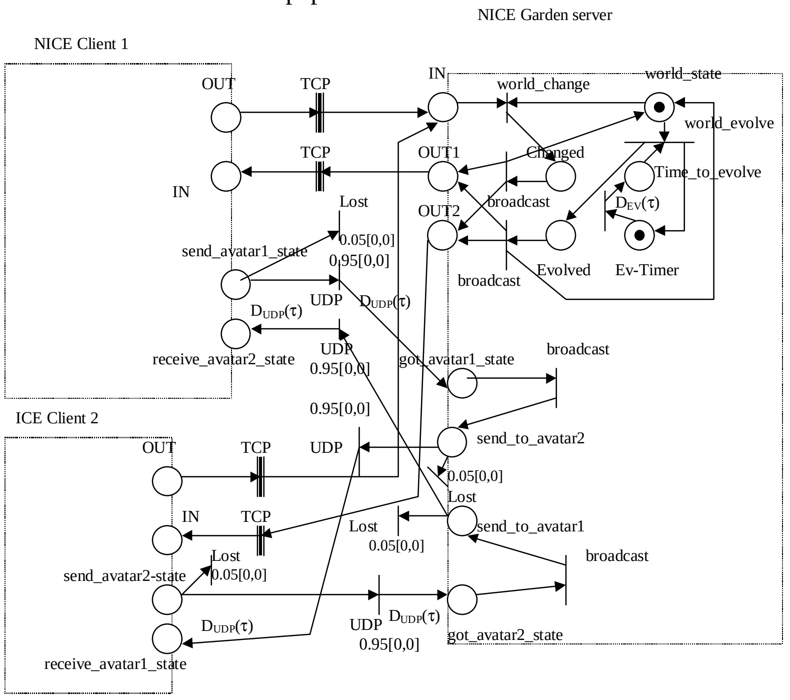

**Fig. 7.** An EFTN model for 2 existing NICE clients communicating with each other and with the server, where the delay of UDP channel is $D_{UDP}(\tau)$ = (50,100,150,200)ms, the world evolves in the interval $D_{EV}(\tau)$ = (10,60,180,300)s, and each TCP transition is an abstract of the subnet for TCP protocol ( TCP is a reliable and ordered protocol. One Protocol Data Unit (PDU)'s loss will delay all subsequent PDUs. No subsequent PDU can be delivered to the application layer until that PDU is successfully received. The subnet model for TCP protocol is to be included in the full paper [13].)

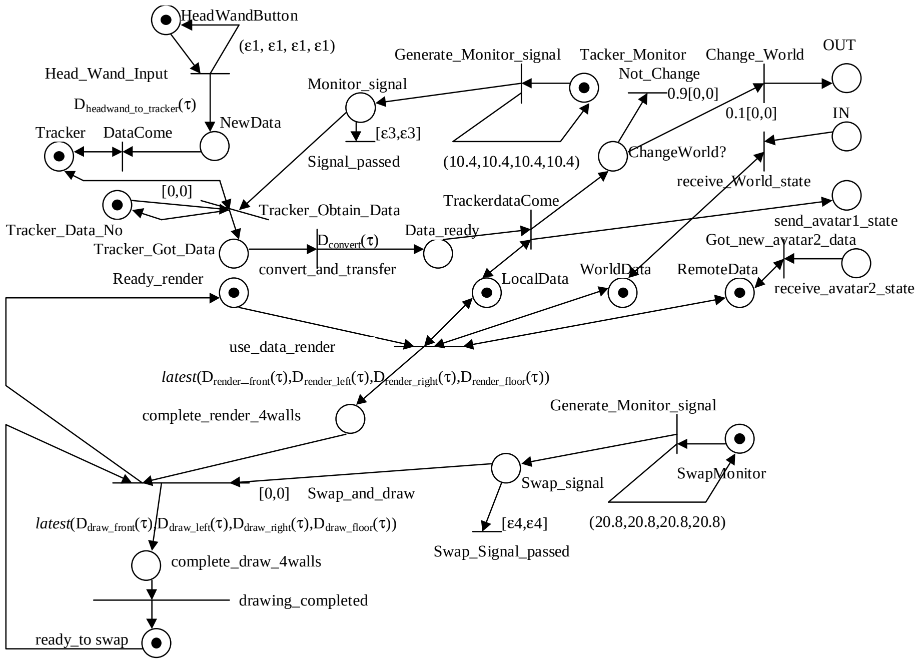

**Fig. 8.** The EFTN model for a CAVE in the NICE as a distributed component

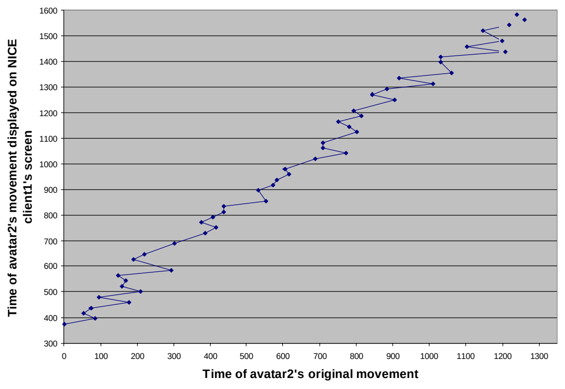

**Fig. 9.** The display behavior of remote avatar on the local screen without using the filter

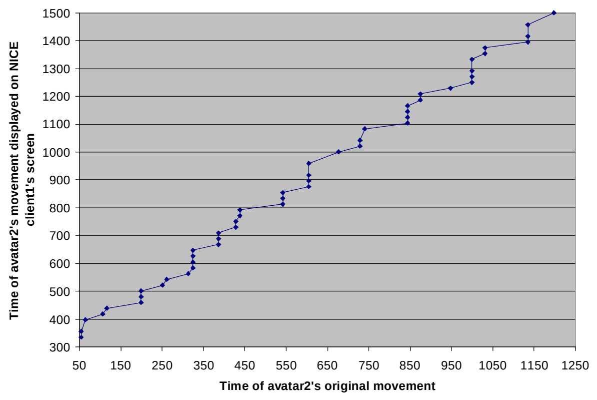

**Fig. 10.** The display behavior of remote avatar on the local screen with using the filter

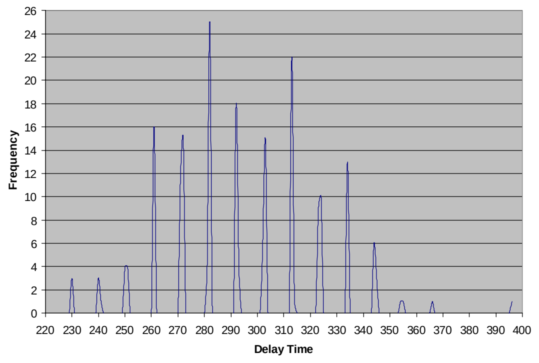

**Fig. 11.** The distribution of the delay from the time that remote user has a movement to the time that movement is displayed on the local screen.

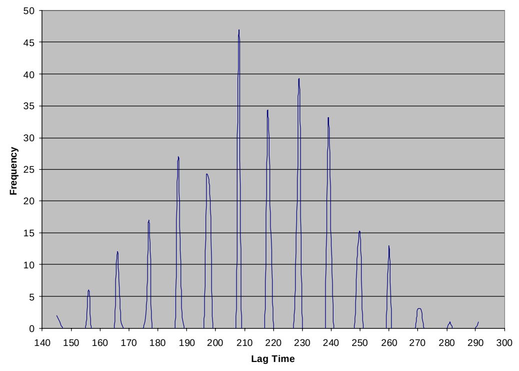

**Fig. 12.** The distribution of the time that remote avatar's display lags behind local avatar

## References

1. E. Juan,  J. P. Tsai, T. Murata, and Y. Zhou, “Reduction Methods for Real-Time Systems Using Delay Time Petri Nets,” Technical report, EECS Dept., University of Illinois, Chicago, March, 1999.
2. J. Leigh, A. Johnson, T. DeFanti, “Issues in the Design of a Flexible Distributed Architecture for Supporting Persistence and Interoperability in Collaborative Virtual Environments,” in the Proceedings of Supercomputing 97, San Jose, California, Nov 15-21, 1997.
3. J. Leigh, A. Johnson, T. DeFanti, “CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments,” in Virtual Reality: Research, Development and Applications, Vol. 2.2, December 1996 (published 1997), pp. 217-237.
4. D. Pape, “CAVE user’s guide,” Electronic Visualization Laboratory, University of Illinois at Chicago, Dec. 1996.
5. P. Merlin, “A study of the Recoverability of Computer Systems. ” Thesis, Computer Science Dept., University of California, Irvine, 1974.
6. R. Mascarenhas, D. Karumuri, U. Buy, and R. Kenyon, " Modeling and analysis of a virtual reality system with time Petri nets," Procs. 19th Int. Conf. on Software Engineering, pp. 33-42, April 1998, Kyoto, Japan.
7. T. Murata, "Temporal Uncertainty and Fuzzy-Timing High-Level Petri Nets," Invited paper at the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan,, LNCS Vol. 1091, pp. 11-28, Springer, June 1996
8. T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, Vol. 77, No 4, April, 1989, pp. 541-580.
9. K. Park, and R. Kenyon, “Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment,” Proceedings of IEEE VR `99 , Houston TX, March 13-17, 1999.
10. T. Murata, T. Suzuki and S. Shatz, "Fuzzy-Timing High-Level Petri Nets (FTHNs) for Time-Critical Systems," in J. Cardoso and H. Camargo (editors) "Fuzziness in Petri Nets" Vol. 22 in the series "Studies in Fuzziness and Soft Computing" by Springer Verlag, New York, pp. 88-114, 1999.
11. Y. Zhou and T. Murata, “Petri Net Model with Fuzzy-Timing and Fuzzy-Metric Temporal Logic,” to appear in the special issue on fuzzy Petri nets: concepts and intelligent system modeling, *International Journal of Intelligent Systems*, 1998.
12. Y. Zhou and T. Murata, “Fuzzy-Timing Petri Net Model for Distributed Multimedia Synchronization,” in the Procs. of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (SMC’98), La Jolla, Calif., Oct. 12-14, 1998.
13. Y. Zhou, T. Murata, and T. DeFanti, “Modeling and Analysis of Collaborative Virtual Environments by using Extended Fuzzy-Timing Petri Nets,” Technical report, EECS Dept., University of Illinois, Chicago, 1999.
14. K. Jensen, and Design/CPN group, “Design/CPN Online,” Department of Computer Science, University of Aarhus, Denmark . Online: http://www.daimi.au.dk/designCPN/.