Adaptive Δ-Causality Control with Adaptive Dead-Reckoning in Networked Games

Yutaka Ishibashi, Yousuke Hashimoto, Tomohito Ikedo, and Shinji Sugawara
Department of Computer Science and Engineering
Graduate School of Engineering
Nagoya Institute of Technology
Nagoya 466–8555, Japan
ishibasi@nitech.ac.jp, hashiyou@mcl.elcom.nitech.ac.jp, shinji@nitech.ac.jp

ABSTRACT

This paper proposes an adaptive Δ-causality control scheme with adaptive dead-reckoning to preserve the consistency among players and the causality for networked games. The proposed scheme carries out adaptive Δ-causality control and adaptive dead-reckoning together. By simulation, we make a performance comparison among nine schemes including the scheme for a networked racing game. We also investigate the influence of the maximum value of Δ of the scheme on the interactivity by subjective assessment. As a result, we illustrate that the scheme is superior to the other schemes in terms of the consistency among players. Also, we show that if the maximum value of Δ is less than about 100 ms, the influence of the value on the interactivity is small.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Communications Applications; K.8.0 [Personal Computing]: General—Games

General Terms

Algorithms, Performance, Human Factors, Experimentation, Measurement

Keywords

Networked racing game, Consistency, Causality control, Simulation, Subjective assessment

1. INTRODUCTION

A number of papers about networked games such as networked racing games and networked shooting games have been published [1]-[3]. However, the consistency and the causality may be disturbed owing to the network latency in the Internet. To solve the problem, we need to carry out causality control, prediction control, and so on.

In [3], the authors make a performance comparison among the Δ-causality scheme, the dead-reckoning scheme, and the Δ-causality scheme with dead-reckoning for a networked racing game. As a result, they illustrate that the consistency among players and the causality can be maintained by the Δ-causality scheme when the network load is light. However, when the network load is heavy, the Δ-causality scheme with dead-reckoning is superior to the other schemes. The scheme carries out the Δ-causality control and dead-reckoning together. Thus, there is a possibility that the combination use of control improves the performance. Therefore, we need to investigate the combination use of other types of control. On the other hand, in [4], the authors demonstrate that the consistency and the causality are improved by the adaptive Δ-causality control. In [5], where the authors do not deal with networked games but collaborative work, they show that the efficiency of the work is improved by the adaptive dead-reckoning, which can dynamically change the amount of traffic according to the network load. If the adaptive dead-reckoning is applied to networked games, we expect that the consistency can be improved.

Therefore, the combination use of adaptive Δ-causality control and adaptive dead-reckoning can achieve the further amelioration of the consistency among players. This is a key idea of this paper.

In this paper, based on the above key idea, we propose an adaptive Δ-causality control scheme with adaptive dead-reckoning, in which the adaptive Δ-causality control is exerted together with adaptive dead-reckoning, to preserve the consistency among players and the causality in networked games. By simulation, we make a performance comparison among nine schemes including the proposed scheme for a networked racing game. We also investigate the influence of the maximum value of Δ of the scheme on the interactivity by subjective assessment.

The rest of this paper is organized as follows. Section 2 proposes the adaptive Δ-causality control scheme with adaptive dead-reckoning. A performance comparison is made in Section 3, and the influence of the maximum value of Δ on the interactivity is examined in Section 4. Section 5 concludes the paper.

2. ADAPTIVE Δ-CAUSALITY CONTROL WITH ADAPTIVE DEAD-RECKONING

Here we propose an adaptive scheme which carries out the adaptive Δ-causality control and adaptive dead-reckoning together. In this paper, each terminal inputs the positional information about the player's car of a networked racing game (or the player's fighter of a networked shooting game) at regular intervals (every 33 ms in our experimental system) and sends the information with its timestamp, which denotes the generation time of the information, as a computer data media unit (MU) to the other players. An MU is the information unit for causality and dead-reckoning.

In the proposed scheme, the adaptive Δ-causality control is exerted for conservation of causality. Under the adaptive Δ-causality control, when each terminal receives an MU, the terminal saves the MU in the terminal's buffer until a time limit and then outputs it. The time limit is equal to the generation time of the MU plus Δ seconds ($\Delta \geq 0$). If the MU is received after the time limit, it is discarded. However, owing to discarding MUs, the positions of cars may be output incorrectly.

We explain the dead-reckoning technique, which is used to improve the consistency, before the explanation of the adaptive dead-reckoning. In the technique, at each terminal (or player), the current position of each car is predicted by the latest two received (transmitted) MUs. Then, we compare the predicted position with the actual position. If the difference between the predicted position and the actual position (i.e., the prediction error) is larger than a threshold value $T_{\rm dr}(> 0)$, the information about the actual position is transmitted as an MU. Otherwise, any MU is not transmitted. When an MU is received, we correct the position over several times in order to correct the position gradually until the difference becomes less than $T_{\rm dr}$. In this paper, for simplicity, we correct the position at a time.

Under the dead-reckoning, if an MU is received within the time limit (i.e., its generation time plus Δ seconds) of the MU at each terminal, the MU is saved in the terminal's buffer until the time limit; then, we predict and correct the position. Otherwise, the MU can be used only to predict the position of car at the next output time.

In the proposed scheme, the threshold $T_{\rm dr}$ of the adaptive dead-reckoning is dynamically changed according to the network load. The network load is estimated from the value of Δ in this paper. The reason is that the value of Δ is dynamically changed according to the network load under the adaptive Δ-causality control. The value of Δ is changed so as to satisfy the following relation: $\Delta_{\rm L} \leq \Delta \leq \Delta_{\rm H}$, where $\Delta_{\rm L}(> 0)$ and $\Delta_{\rm H}(> 0)$ are the minimum and maximum values of Δ, respectively. The value of Δ is increased by $\Delta_{\rm I}(> 0)$ when the number of MUs which are received continuously after their time limits reaches $N_{\rm a}(\geq 1)$. On the other hand, when $N_{\rm b}(\geq 1)$ MUs arrive continuously before their time limits, the value of Δ is decreased by $\Delta_{\rm D}(> 0)$. The initial value of Δ is set to $\Delta_{\rm L}$. Thus, the threshold $T_{\rm dr}$ of the adaptive dead-reckoning is changed according to the value of Δ. We set the threshold $T_{\rm dr}$ to larger values as the value of Δ increases. This is because the value of Δ is changed according to the network load under the adaptive Δ-causality control.

3. PERFORMANCE COMPARISON

In this section, we make a performance comparison of nine schemes including the proposed scheme by simulation when three players play a networked racing game. ns-2 [6], which is a network simulator, is used for simulation.

3.1 Network configuration

The configuration of a network which is used for simulation is shown in Figure 1. The network is constructed based on the Tiers model [7], [8]. The Tiers model is a network model which consists of three classes (i.e., WAN, MAN, and LAN) and is similar to a real network. In Figure 1, each of MANs 1 and 2 is connected to a WAN; furthermore, the MAN is connected to four LANs. There are two routers in the WAN. The number of routers constituting each MAN is 2. Every LAN excluding LAN 5 consists of two terminals with one router. LAN 5 contains one router and one terminal.

The transfer speed of each link in the WAN, each MAN, and each LAN is set to 5 Mbps, 10 Mbps, and 100 Mbps, respectively. The transfer speed of a WAN-MAN link is 5 Mbps, and that of a MAN-LAN link is 10 Mbps. Moreover, the propagation delay of a link in the WAN is set to 10 ms, and that of a link in each MAN and that of each link in each LAN are set to 5 ms and 1 ms, respectively. The propagation delay of a WAN-MAN link is set to 5 ms, and that of a MAN-LAN link is set to 2 ms.

3.2 Method of simulation

In Figure 1, we suppose that three players (players 1, 2, and 3) play a networked racing game. Player 1 uses game terminal 1 of LAN 1, player 2 uses game terminal 2 of LAN 4, and player 3 uses game terminal 3 of LAN 8. MUs are sent and received among the game terminals. Moreover, sender m ($\rm S_m$; $1 \leq m \leq 6$) sends IP datagrams (load data) of 1500 bytes each to receiver m ($\rm R_m$) at exponentially distributed intervals. MUs and load data are transmitted by UDP as a transport protocol.

Before starting the simulation, in order to generate the

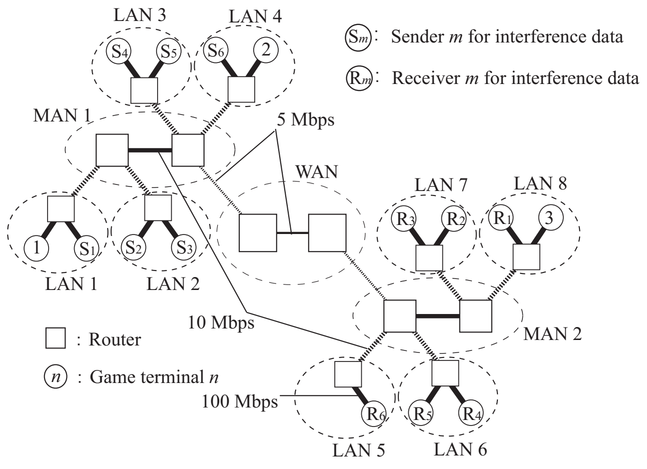


Figure 1: Configuration of network.

computer data traffic of the same amount in the simulation, we stored the positions of three cars in files every 33 ms in our experiment; we made ten files. Each of the three players drove a car along a racing course (see Figure 2) for 30 seconds ten times in the case of no network latency. In the simulation, car i (i = 1, 2, and 3) is moved according to the stored files at game terminal i. Game terminal i transmits MUs each of which includes the position of car i to the other game terminals. When the game terminal receives an MU, it updates the position of each car by using the positional information included in the MU.

In this paper, we handle nine schemes for a performance comparison. The schemes are the proposed scheme, the adaptive Δ-causality scheme with the prediction technique and dead-reckoning (referred to as Prediction/DR+Adaptive Δ) [9], the adaptive Δ-causality scheme with the prediction technique (Prediction+Adaptive Δ) [9], the adaptive Δ-causality scheme with dead-reckoning (DR+Adaptive Δ) [9], the adaptive Δ-causality scheme (Adaptive Δ) [3], the Δ-causality scheme with dead-reckoning (DR+Δ) [3], the Δ-causality scheme (Δ) [3], dead-reckoning (DR) [3], and a scheme which does not carry out causality control or prediction control (the no-control scheme: NC) [3].

In the simulation, the values of Δ in Δ-Causality and DR+Δ, which use the Δ-causality control, are set to 100 ms [2]. In the proposed scheme, Prediction/DR+Adaptive Δ, Prediction+Adaptive Δ, DR+Adaptive Δ, and Adaptive Δ, which use the adaptive Δ-causality control, parameters of the adaptive Δ-causality control are set as follows: Δ_L = 50 ms, Δ_H =100 ms, Δ_I = 10 ms, Δ_D = 5 ms, N_a = 1, and N_b = 10. The threshold values T_{dr} in Prediction/DR+Adaptive Δ, DR+Adaptive Δ, DR+Δ, and DR, which use the dead-reckoning technique, are set to 0.1, where the depth of each car is assumed to be 1. The threshold value T_{dr} in the proposed scheme, which uses the adaptive dead-reckoning technique, is set as follows:

$$\begin{aligned} T_{dr} &= 0 \ (50 \text{ ms} \le \Delta < 75 \text{ ms}), \\ T_{dr} &= 0.02 \ (75 \text{ ms} \le \Delta < 100 \text{ ms}), \\ T_{dr} &= 0.1 \ (\Delta = 100 \text{ ms}). \end{aligned}$$

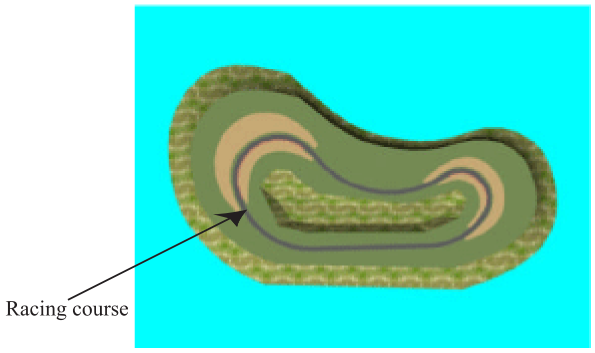


Figure 2: Image of racing course.

3.3 Simulation results

We show the inconsistency rate of positional relations as a function of the average load in Figure 3. We also show the mean square error of position of player's car versus the average load in Figures 4 through 7. The inconsistency rate of positional relations is defined as the ratio of the number of disagreements among the three positional relations to the total number of comparisons. This measure is important since the positional relations are closely related to the outcome of a race (i.e., victory or defeat). The mean square error of position denotes the average square of the positional error of one player's car between two game terminals. The average load is defined as the average number of interference data bits transmitted in a second at each sender; the average loads at all the senders are the same. In Figures 4 through 7, we assume that the depth of each car is 1. We further plot the 95 % confidence intervals of the performance measures in Figures 3 through 7.

In Figure 3, the inconsistency rate of positional relations of Δ is not plotted when the average load is less than 0.65 Mbps, and the inconsistency rates of the proposed scheme, Prediction/DR+ Adaptive Δ, and Prediction+ Adaptive Δ are not plotted when the average load is less than 0.60 Mbps. The reason is that the inconsistency rates of the schemes were zero in this area. In Figures 4 through 7, we show the mean square error of position of car 1 between game terminals 1 and 2, that of car 1 between game terminals 2 and 3, that of car 2 between game terminals 2 and 3, and that of car 3 between game terminals 1 and 2, respectively. The mean square errors of Δ, Adaptive Δ, and Prediction+Adaptive Δ are not plotted in Figure 4 since the errors were zero. For the same reason, the mean square errors of all the schemes excluding NC and DR are not plotted in Figure 7. The mean square errors of car 2 between game terminals 1 and 2, car 3 between game terminals 1 and 3, and car 3 between game terminals 2 and 3 were almost the same as those in Figure 4. Furthermore, the mean square error of car 1 between game terminals 1 and 3, and that of car 2 between game terminals 1 and 3 were almost the same as those in Figures 5 and 6, respectively.

In Figure 3, we see that the inconsistency rate of positional relations of the proposed scheme is the second smallest when the average load is less than around 0.65 Mbps. The inconsistency rate of the proposed scheme is the same as those of Prediction/DR+Adaptive Δ, DR+Adaptive Δ, and DR+Δ when the average load is heavier than about 0.80 Mbps, and the rate is smaller than those of the other schemes. When the average load is between around 0.65 Mbps and 0.80 Mbps, the proposed scheme is the best.

In Figure 4, the mean square errors of Prediction/DR+Adaptive Δ and the proposed scheme tend to increase as the average load becomes heavier. The mean square errors of Prediction/DR+Adaptive Δ and the proposed scheme are the same as those of DR+Δ and DR+Adaptive Δ when the average load is heavier than about 0.80 Mbps.

In Figure 5, we see that the mean square error of the proposed scheme is the smallest or the second smallest. The mean square error of the proposed scheme is almost the same as those of Prediction/DR+Adaptive Δ, DR+Adaptive Δ, and DR+Δ when the average load is larger than about 0.80 Mbps. This reason is that all the MUs are received within their time limits.

From Figure 6, we find that the mean square error of the proposed scheme is the second smallest when the average load is less than around 0.60 Mbps; in this area, note that the error of Δ is zero. When the average load is heavier than about 0.65 Mbps, the proposed scheme is the best.

In Figure 7, we observe that the mean square errors of all the schemes are independent of the average load. This is because MUs input at terminal 3 are transmitted in the opposite direction to the load data.

From the above observations, we can say that the proposed scheme is superior to the other schemes in terms of the consistency among players. The reason is that the prediction error is reduced by changing the threshold value T_{dr} according to the network load in the proposed scheme.

4. INFLUENCE OF Δ_{H}

When the maximum value of Δ (i.e., Δ_{H}) of the proposed scheme increases, the interactivity may be damaged. Therefore, we have examined the influence of Δ_{H} on the interactivity by subjective assessment.

4.1 Experimental system

We show the configuration of the experimental system in Figure 8. The experimental system consists of two game

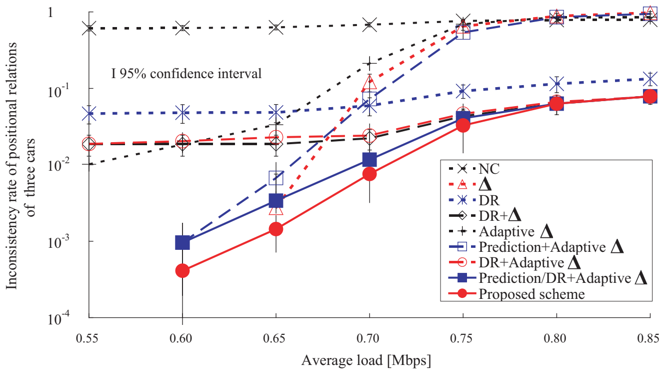


Figure 3: Inconsistency rate of positional relations of three cars.

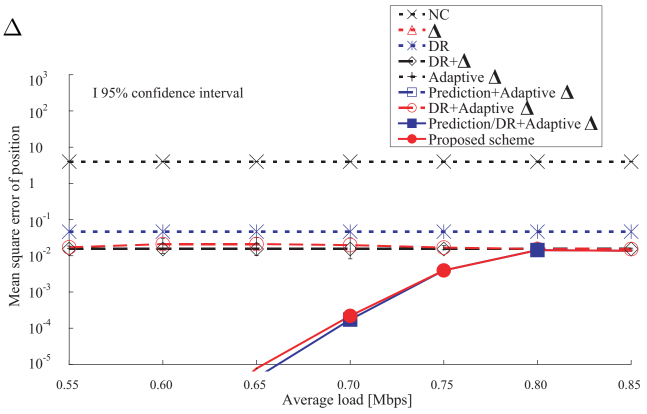


Figure 4: Mean square error of position of car 1 between game terminals 1 and 2.

terminals (CPU: Pentium4 2.4 GHz, OS: WindowsXP Home Edition, RAM: 512 Mbytes, Graphic board: GeForce4 MX 420) and a network emulator (NIST Net [10]). The two game terminals are connected to NIST Net via Ethernet cables (100BASE-T). NIST Net generates an additional delay for each MU according to the Pareto-normal distribution [10]. Each user operates a car along a racing course in Figure 2 (see Figure 9) with a wheel type input device (Microsoft SIDEWINDER FORCE FEEDBACK WHEEL).

4.2 Method of experiment

We have performed subjective assessment to examine the influence on the interactivity by changing the value of Δ_{H} in the proposed scheme. The value of Δ_{H} is selected from among 50 ms, 75 ms, 100 ms, 125 ms, 150 ms, 175 ms, and 200 ms. The standard deviation of the additional delay from terminal 1 to terminal 2 and that from terminal 2 to terminal 1 are set to 10 ms. The average additional delay from terminal 1 to terminal 2 and that from terminal 2 to terminal 1 are chosen from among 10 ms, 100 ms, and 200 ms.

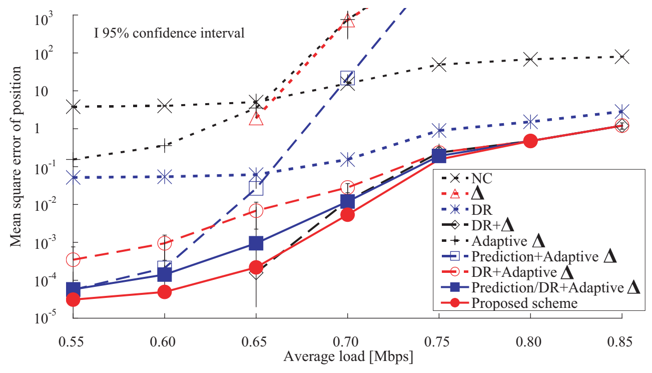


Figure 5: Mean square error of position of car 1 between game terminals 2 and 3.

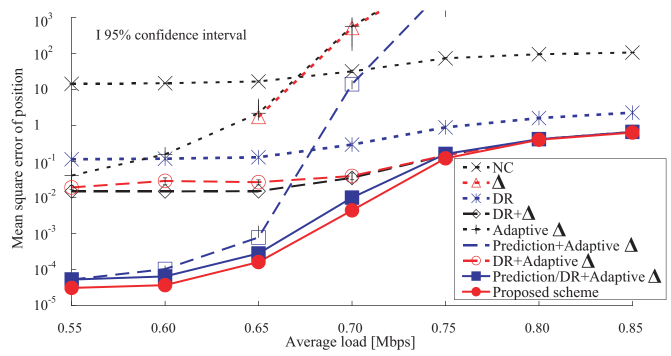


Figure 6: Mean square error of position of car 2 between game terminals 2 and 3.

The subjects were asked to base their judgments in terms of wording used to define the subjective scale (see Table 1). Each subject gave a score from 1 through 5 to each test to obtain the mean opinion score (MOS) [11]. In each test, each subject played the game for 30 seconds. The number of subjects was 20. The subjects were from 22 to 24 years old. They did not play games on a daily basis. The total time per subject was around 20 minutes.

Table 1: Five-grade impairment scale.

Score	Description
5	Imperceptible
4	Perceptible, but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

4.3 Experimental results

We show the MOS value as a function of Δ_H in Figure 10, where we also display the 95 % confidence intervals.

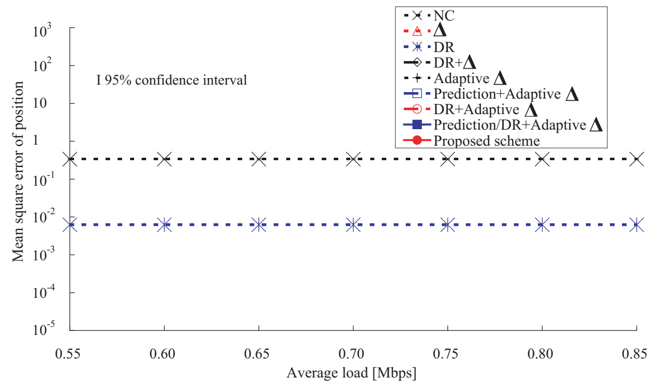
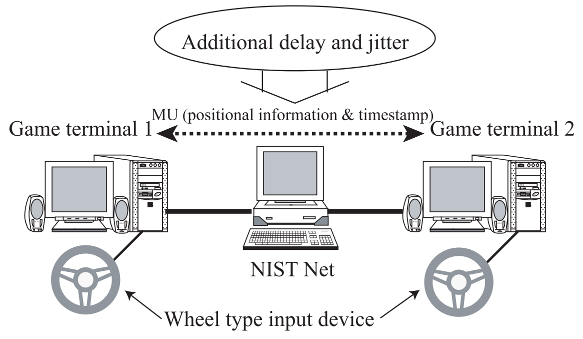


Figure 7: Mean square error of position of car 3 between game terminals 1 and 2.



Additional delay and jitter
MU (positional information & timestamp)
Game terminal 1
Game terminal 2
NIST Net
Wheel type input device

Figure 8: Configuration of experimental system.

Figure 10 reveals that the MOS value hardly depends on the value of Δ_H when the average additional delay is 10 ms. This is because when average additional delay is small, the value of Δ hardly increases. We also see in figure that the MOS value slightly decreases when the average additional delay is 100 ms and the value of Δ_H exceed about 100 ms. When the average additional delay is 200 ms and the value of Δ_H exceeds about 100 ms, the MOS value starts to decrease linearly. The reason is that the operability of car becomes more difficult as the value of Δ increases.

As a result, in the networked racing game, it is thought that the influence on the interactivity is small if the value of Δ_H is less than about 100 ms. This result is the same that in [12].

5. CONCLUSIONS

This paper proposed an adaptive Δ-causality control scheme with adaptive dead-reckoning for networked games. By simulation, we made a performance comparison among nine schemes including the proposed scheme for a networked racing game. As a result, we found that the proposed scheme is superior to the other schemes in terms of the consistency

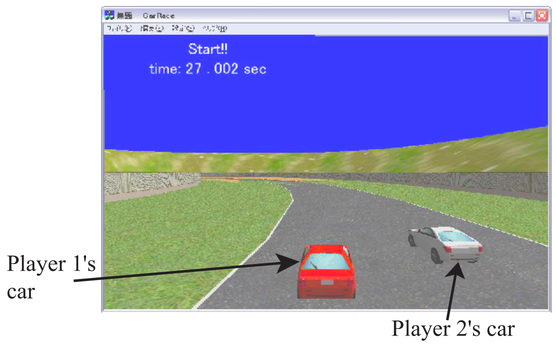


Figure 9: Displayed image at player 1.

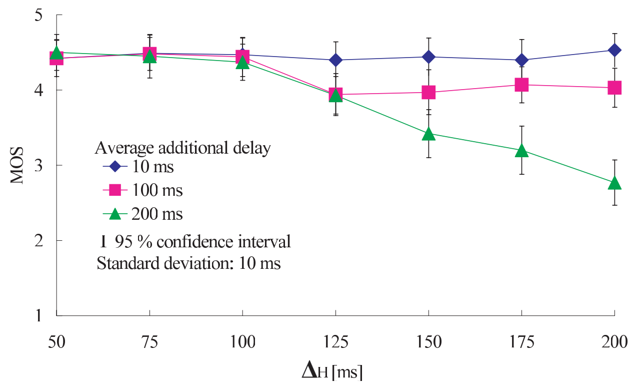


Figure 10: MOS versus $\Delta_{\rm H}$.

among players. Also, we saw that the influence on the interactivity is small if the maximum value of Δ ($\Delta_{\rm H}$) is less than about 100 ms.

As the next step of our research, we need to investigate the performance in the case where there exist a number of players and in a variety of network environments by simulation. We will also investigate the performance for other networked games such as networked shooting games in the same way as that in this paper.

6. REFERENCES

[1] C. Diot and L. Gautier. A distributed architecture for multiplayer interactive applications on the Internet. IEEE Network, 13(4):6–15, July/August 2003.

[2] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In Proc. ACM NetGames'02, pages 79–84, April 2002.

[3] T. Yasui, Y. Ishibashi, and T. Ikedo. Influence of network latency and packet loss on consistency in networked racing games. In Proc. ACM NetGames'05, October 2005.

[4] Y. Ishibashi and S. Tasaka. Causality and media synchronization control for networked multimedia games: Centralized versus distributed. In Proc. ACM NetGames'03, pages 34–43, May 2003.

[5] T. Kanbara, Y. Ishibashi, and S. Tasaka. Haptic media synchronization control with dead-reckoning in networked virtual environments. In Proc. the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'04), III, pages 158–163, July 2004.

[6] VINT Project. The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[7] M. B. Doar. A better model for generating test networks. In Conf. Rec. IEEE GLOBECOM'96, pages 86–93, November 1996.

[8] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling Internet topology. IEEE Communications Magazine, 35(6):160–163, June 1997.

[9] T. Ikedo and Y. Ishibashi. An adaptive scheme for consistency among players in networked racing games. In Proc. International Workshop on Future Mobile and Ubiquitous Information Technologies (FMUIT'06), pages 154–157, May 2006.

[10] M. Carson and D. Santay. NIST Net - A Linux-based network emulation tool. ACM SIGCOMM Computer Communication Review, 33(3):111–126, July 2003.

[11] ITU-R BT. 500-11. Method for subjective assessment of the quality of television pictures. International Telecommunication Union, June 2002.

[12] L. Pantel and L. C. Wolf. On the impact of delay on real time multiplayer games. In Proc. ACM NetGames'03, pages 23–29, April 2002.