Consistency and Fairness among Players in Networked Racing Games: Influence of Network Delays

Takahiro Yasui and Yutaka Ishibashi
Department of Computer Science and Engineering, Graduate School of Engineering,
Nagoya Institute of Technology, Nagoya 466–8555, Japan
taka@mcl.elcom.nitech.ac.jp, ishibasi@nitech.ac.jp

Abstract

This paper investigates the influence of network delays and delay jitter on the consistency and fairness among players in networked racing games. To maintain the consistency and fairness, we deal with the Δ-causality control as causality control and dead-reckoning as prediction control. By experiment, we make a performance comparison among the Δ-causality scheme, the dead-reckoning scheme, a scheme which carries out the Δ-causality control and dead-reckoning together, and a scheme which performs neither of the two types of control. As a result, we show that the combination use of the two types of control can keep the consistency and fairness in good condition.

Key words: Networked racing game, Consistency, Fairness, Causality control, Dead-reckoning, Experiment

1. Introduction

Broadband access to the Internet is enabling networked real-time games such as networked racing games and networked shooting games. Such games have severe constraints on the delay from input of information to its output. Also, the consistency of the state among players and the causality of input events are important. These features largely influence the outcome of a race (i.e., victory or defeat) in the games.

However, owing to the network delay and its jitter, the causality and consistency among players may be disturbed. As the difference in network delay among players becomes larger, the positions of racing cars displayed at one player become more largely different from those at another player. This brings the unfairness among the players. To solve these problems, we need to carry out causality control [1]–[4] and prediction control [4], [5], and so on.

In [1], the authors demonstrate that the causality can be maintained by the Δ-causality control. However, since a number of packets are discarded when the network load is heavy, the consistency and fairness among players are disturbed. Pantel and Wolf illustrate the effectiveness of dead-reckoning [6], which is one of prediction control schemes, by applying dead-reckoning to several kinds of games [5]. In [4], Diot and Gautier propose the bucket synchronization, which carries out causality control and prediction control together. However, they do not make a performance comparison between the bucket synchronization and other schemes. Also, they do not clarify how the joint use of the two types of control is superior to the individual use of each type of control. Furthermore, to the best of the authors' knowledge, there is no paper which clarifies the influence of the network delay and its jitter on the consistency and fairness quantitatively.

This paper deals with the Δ-causality control as causality control and dead-reckoning as prediction control. By experiment, we make a performance comparison among the Δ-causality scheme, the dead-reckoning scheme, a scheme which carries out the Δ-causality control and dead-reckoning together, and a scheme which performs neither of the two types of control. We also investigate the influence of the network delay and its jitter on the consistency and fairness in the four schemes.

The rest of this paper is organized as follows. Section 2 discusses the consistency and fairness in networked racing games and describes their performance measures. Section 3 explains the four schemes. The method of the experiment is explained in Section 4, and experimental results are presented in Section 5. Section 6 concludes the paper.

2. Consistency and fairness

We suppose in this paper that two players (players 1 and 2) play a networked racing game. We also assume that the game is implemented based on the peer-to-peer model. In what follows, we first handle the case in which network delays between the two players are small. Next, we deal with the case in which the network delays are large or largely different from each other.

In Fig. 1, we show displayed images of the racing cars of players 1 and 2 in the former case. The image on the left-hand side in the figure is displayed at player 1, and that on the right-hand side is at player 2. For simplicity, we assume in the figure that the racing course is straight. Players 1 and 2 drive cars 1 and 2, respectively. At each player, the player's car and the other player's car are displayed. The information about the position of player 1 (2)'s car arrives late at player 2 (1) owing to network delays. Therefore, if we update the position of the car 1 (2) at player 2 (1) without any control, the position of car 1 (2) at player 1 (2) is different from that of car 1 (2) at player 2 (1). This means that the state at player 1 is not consistent with that at player 2.

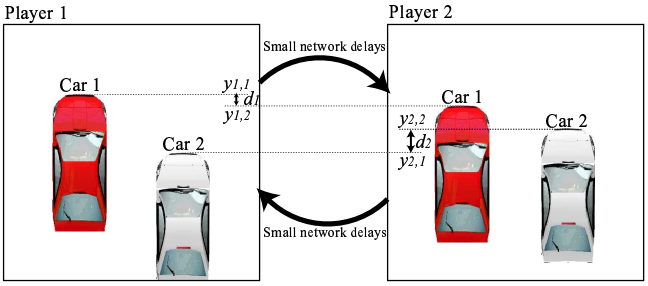
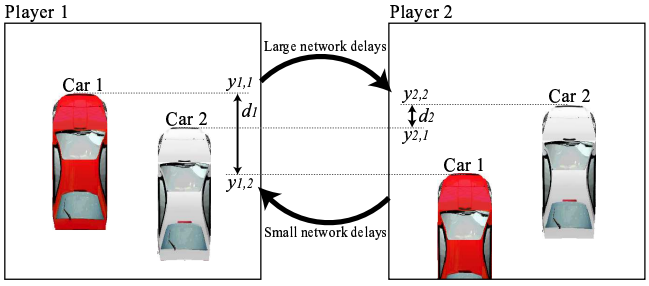
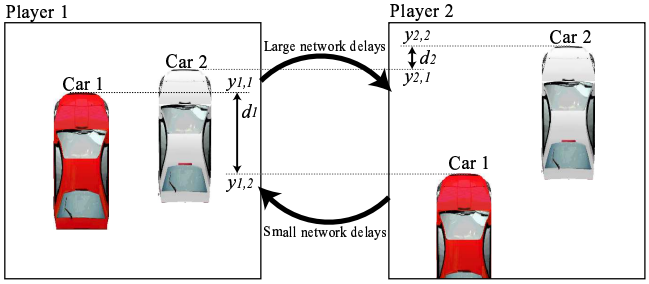


Fig. 1. Displayed images at players 1 and 2 in the case where network delays are small.



(a) Case in which the positional relations of the two cars between players 1 and 2 are not consistent



(b) Case in which the positional relations of the two cars between players 1 and 2 are consistent

Fig. 2. Displayed images at players 1 and 2 in the case where network delays are largely different from each other.

In order to discuss the consistency quantitatively, we introduce the positional error of a car, which is defined as the difference between the position of car 1 (2) at player 1 and that at player 2. Let us denote the positions of cars 1 and 2 at player 1 by $y_{1,1}$ and $y_{2,1}$, respectively, at a given time and those at player 2 by $y_{1,2}$ and $y_{2,2}$, respectively. Then, the positional error d_1 of car 1 is given by $d_1 = y_{1,1} - y_{1,2}$, and that of car 2 is $d_2 = y_{2,2} - y_{2,1}$ (see Fig. 1)[†] . For simplicity, we here assume that $d_1 \geq 0$ and $d_2 \geq 0$. If $d_1 = d_2 = 0$, the consistency is maintained. Otherwise, the consistency is not strictly kept. However, even when $d_1 \neq 0$ or $d_2 \neq 0$, the outcome of the race (i.e., victory or defeat) at player 1 is the same as that at player 2 if which car is ahead of the other car is the same between the two players [††] . In Fig. 1, since $d_1 \neq 0$ and $d_2 \neq 0$, the consistency is disturbed; however, which car is ahead of the other car is the same between the two players; this is because the values of d_1 and d_2 are small.

As a performance measure for the consistency, this paper introduces the inconsistency rate of the positional relations of the two cars (that is, which car is ahead of the other car) between the two players. We obtain the positional information about the two cars at regular intervals and compare the positional relation between the two cars at player 1 with that at player 2. The inconsistency rate is defined as the ratio of the number of disagreements between the two positional relations to the total number of comparisons. This measure is very important since the positional relations are closely related to the outcome of a race (i.e., victory or defeat).

Then, we handle the case in which the network delays are large or largely different from each other as shown in Fig. 2. Figure 2 (a) illustrates the case in which the positional relations of the two cars between players 1 and 2 are not the same. In Fig. 2 (b), the positional relations of the two cars between players 1 and 2 are the same; that is, car 2 is ahead of car 1 at the two players. We set the values of d_1 and d_2 in Fig. 2 (a) to the same as those in Fig. 2 (b), respectively ($d_1 > d_2$ in Fig. 2). In Fig. 2 (b), since car 2 is largely ahead of car 1 at player 2, car 2 is slightly ahead of car 1 at player 1. If car 2 is ahead of car 1 at player 2 by more than $d_1 + d_2$, the positional relations can be the same (see Fig. 2). Also, at player 1, the positional relations can be the same if car 1 is ahead of car 2 by more than $d_1 + d_2$. That is, even if the consistency of the state is violated, the consistency of the positional relations can be preserved; this depends on the positional errors of the two cars (i.e., d_1 and d_2). As the values of d_1 and d_2 increase, it becomes more difficult to keep the same positional relations.

In addition, large differences between d_1 and d_2 lead to the unfairness between the two players. This is because as the positional error becomes larger, the car of a player is displayed more largely late at the other player.

To discuss the fairness in further detail, we assume that the position of car 1 at player 1 is equal to that of car 2 at player 2; that is, $y_{1,1} = y_{2,2}$ (see Fig. 3). This means that the skills of the two players are equal to each other. As described earlier, when the difference in the position between the two cars at player 1 or 2 is larger than $d_1 + d_2$, the positional relations between the two players can be the same. Thus, if player 1 can advance the position of car 1 from the current position by d_1, the positional relations can be the

[†] When the course is curved, we obtain the positional error by approximating the curve with multiple straight lines which have a constant length.

[††] It should be noted that generally, prediction control schemes produce positional errors.

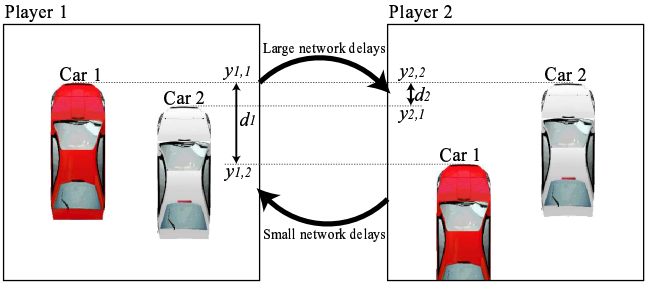


Fig. 3. Displayed images at players 1 and 2 in the case where player 1 has the same skill as player 2 (i.e., $y_{1,1} = y_{2,2}$).

same between the two players. As in player 1, if player 2 can advance the position of car 2 by d_2, the positional relations can also be the same. In Fig. 3, since $d_1 > d_2$, player 1 needs to advance the position of car 1 largely than player 2 in order to have the same positional relations between the two players. Therefore, we say in this paper that player 1 is disadvantageous in terms of the fairness.

To discuss the fairness between the two players quantitatively, we employ the difference $d_1 - d_2$ in the positional errors between the two cars. As in Fig. 3, when $d_1 - d_2 > 0$, player 1 is unfavorable in terms of the fairness. When $d_1 - d_2 = 0$, players 1 and 2 are even. When $d_1 - d_2 < 0$, player 2 is disadvantageous.

3. Schemes for performance comparison

In this section, we handle four schemes for a performance comparison; the Δ-causality scheme (referred to as Causality in this paper), the dead-reckoning scheme (DR), and a scheme which carries out the Δ-causality control and dead-reckoning together (DR+Causality), and a scheme which performs neither of the two types of control (the no-control scheme; NC), In the four schemes, each player obtains the positional information about the player's car at regular intervals (every 33 ms in our experimental system) and sends the information with its timestamp, which is the generation time of the information, as a computer data media unit (MU) [1], [10] to the other player.

In DR at each terminal (or player)[†] , the current position of the other player's car is predicted by the past received (transmitted) MUs. For simplicity, we use the first-order prediction [5]. In the first-order prediction, we calculate the predicted position by using the information about the position included in the last received (transmitted) MU and about the velocity calculated with the positional information included in 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 $T_{\rm dr}$, the information about the actual position is transmitted as an MU. Otherwise, the information is not transmitted. In the convergence technique, 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 make the convergence at a time.

NC outputs MUs on receiving the MUs at each terminal. When we employ NC or DR, an MU which is captured at each terminal is output at its generation time immediately.

In Causality, 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. If the MU is received after the time limit, it is discarded. In this paper, we assume that globally synchronized clocks are employed; that is, the clock ticks at the two terminals have the same advancement, and the current local times are also the same[††] .

When we use DR+Causality, 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 and then used for the prediction and convergence by the same method as that of DR. If the MU is received after the time limit, it can be used for the prediction at the next output time. In Causality and DR+Causality, each MU which is generated at each terminal is also stored in the terminal's buffer until its time limit in order to be output.

4. Method of the experiment

Figure 4 shows displayed images of a networked racing game which was used in our experiment. In order to generate the computer data traffic of the same amount in each experimental run, we stored the positions of two cars in files every 33 ms. Two players who have almost the same skill[†††] drove the two cars along a racing course (see Fig. 4 (b)) for 30 seconds in the case of no network delay and no delay jitter. In the experiment, car 1 (2) is moved according to the stored file at terminal 1 (2). Terminal 1 (2) transmits MUs each of which includes the position of car 1 (2) to terminal 2 (1). When the terminal receives an MU, it updates the position of the car by using the positional information in the MU.

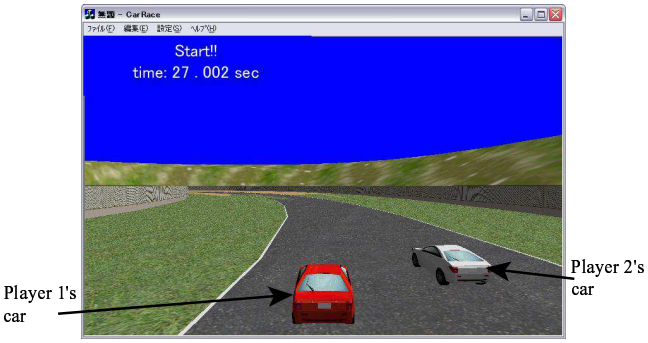
In the experiment, the values of Δ in Causality and DR+Causality are set to 100 ms [9] [‡] . Those of $T_{\rm dr}$ in DR and

[†] We use the words 'terminal' and 'player' interchangeably here.

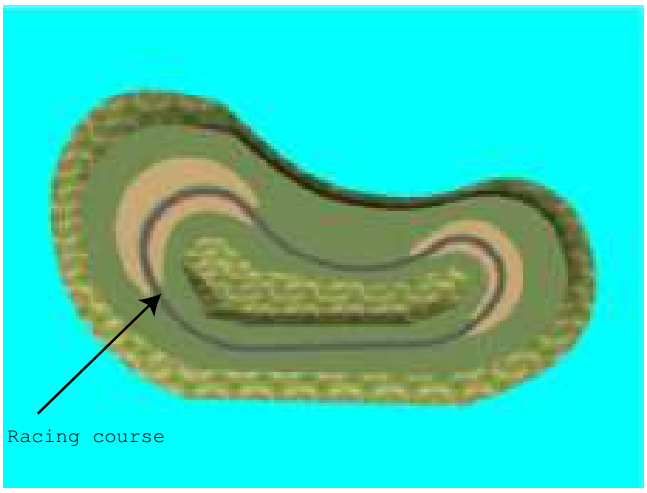
[††] Using the Network Time Protocol (NTP) [8], we can adjust the clock ticks to each other within a few milliseconds.

[†††] The reason why we use the two players with almost the same skill is that we make the positional relation of the two cars (that is, which car is ahead of the other car) switch frequently. In the experiment, the positional relation of the two cars switches ten times. If the skills of the two players are largely different from each other, the more skillful player's car is largely ahead of the other player's one. In this case, the network delay does not influence the positional relation of the two cars as in Fig. 2 (b).

[‡] In [9], it is shown that delays within around 100 ms are subjectively allowable in a networked racing game.



(a) An example image displayed at player 1



(b) An image of the racing course

Fig. 4. Displayed images of the networked racing game.

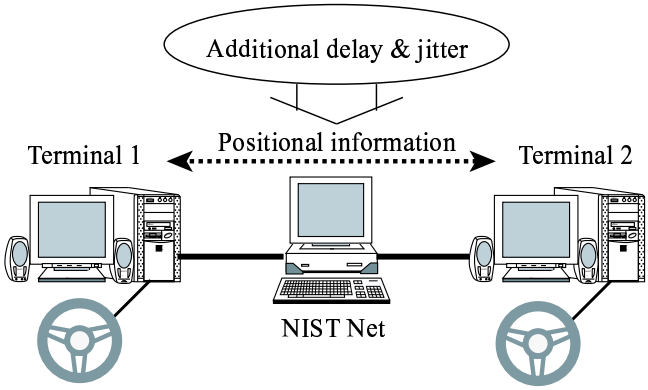


Fig. 5. Configuration of the experimental system.

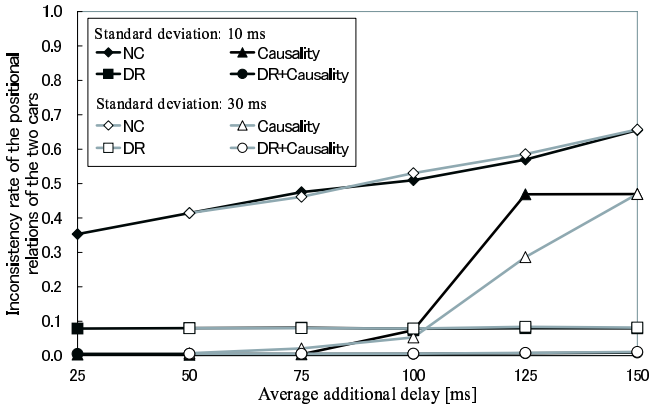


Fig. 6. Inconsistency rate of the positional relations of the two cars versus the average additional delay.

DR+ Causality are set to 0.25, where the width of each car is 1.

We show configuration of the experimental system in Fig. 5. The experimental system consists of the two terminals (CPU: Pentium4 2.4 GHz, OS: WindowsXP Home Edition, RAM: 512 Mbytes, Graphic board: GeForce4 MX 420) and a network emulator (NIST Net [7]). The two 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 [7]. In the experiment, the average and the standard deviation of the additional delay from terminal 2 to terminal 1 are set to 75 ms and 10 ms, respectively. Those from terminal 1 to terminal 2 are changed.

5. Experimental results

We show the inconsistency rate of the positional relations of the two cars as a function of the average additional delay from terminal 1 to terminal 2 in Fig. 6. We also plot the average difference in the positional errors between the two cars versus the average additional delay in Fig. 7. In the figures, we show experimental results when the standard deviation of the additional delay is 10 ms or 30 ms. We do not handle the cases in which the standard deviation of the additional delay is greater than the average additional delay in the figures.

In Fig. 6, we see that the inconsistency rate of NC becomes larger linearly as the average additional delay increases. The reason is that the positional error of car 1 or 2 becomes larger in this case. From Fig. 6, we also find that the inconsistency rate of Causality is equal to zero when the average additional delay is less than around 100 ms; thus, the consistency of positional relations of cars is retained. However, when the average additional delay exceeds around 100 ms, the inconsistency rate of Causality jumps up. This is because that the number of MUs which are discarded owing to missing their time limits increases largely since the average additional delay from terminal 1 to terminal 2 exceeds 100 ms ($= \Delta$). In the figure, we notice that the inconsistency rates of DR and DR+Causality are almost constant independently of the average additional delay. The reason is that even when an MU arrives late, we

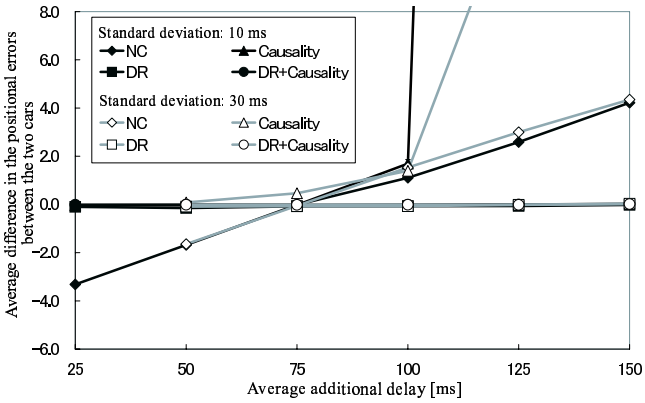


Fig. 7. Average difference in the positional errors between the two cars versus the average additional delay.

can output the predicted MU by the prediction. Furthermore, in the two schemes, the predicted position of each car was not largely different from the actual position. In Fig. 6, the inconsistency rate of DR+Causality is smaller than that of DR. In addition, when the average additional delay is larger than around 100 ms, DR+Causality has the smallest inconsistency rate among all the schemes. This is because DR+Causality can compensate for larger additional delay jitters by buffering MUs for Δ seconds. Hence, DR+Causality can predict the position of the car more precisely than DR.

Then, we note in Fig. 6 that the inconsistency rates of NC, DR, and DR+Causality are hardly dependent on the standard deviation of the additional delay. However, when the average additional delay is 125 ms, the inconsistency rate of Causality at the standard deviation of 30 ms is smaller than that of 10 ms. The reason is that the number of MUs which arrive within their time limits increases as the standard deviation becomes larger.

In Fig. 7, we can see that the average difference in the positional errors of NC changes from negative into positive when the average additional delay exceeds 75 ms. The reason is as follows. Since the average additional delay from terminal 2 to terminal 1 is set to 75 ms, the positional error of the car 1 is smaller than that of car 2 when the average additional delay from terminal 1 to terminal 2 is smaller than 75 ms. However, the positional error of the car 1 becomes larger than that of the car 2 when the average additional delay exceeds 75 ms. We also observe in Fig. 7 that the average difference in the positional errors of Causality suddenly jumps up when the average additional delay exceeds 100 ms. This is because that as described in the case of Fig. 6, the number of discarded MUs increases largely at terminal 2 since the MUs do not arrive until their time limits. The figure also reveals that the average differences of DR and DR+Causality are almost zero. Therefore, the fairness among players is almost perfectly maintained in the two schemes.

From the above considerations, we can say that a scheme which carries out the Δ-causality control and dead-reckoning together can retain the consistency and fairness among players in better condition than the other three schemes even when the network delay becomes larger.

6. Conclusions

In this paper, we discussed the consistency and fairness among players for networked racing games. By experiment, we made a performance comparison among four schemes: The Δ-causality scheme, the dead-reckoning scheme, a scheme which carries out the Δ-causality control and dead-reckoning together, and a scheme which performs neither of the two types of control. As a result, we found that the scheme which carries out the Δ-causality control and dead-reckoning together can keep the consistency and fairness among players in good condition.

As the next step of our research, we need to investigate the influence of packet loss on the consistency and fairness. Also, we will make a performance comparison between the client-server model and the peer-to-peer model as in [10]. Furthermore, we plan to investigate the performance in the case where there exist a number of players by simulation.

Acknowledgment

This work was supported by the Grant-In-Aid for Scientific Research of Japan Society for the Promotion of Science under Grant 16560331.

References

[1] Y. Ishibashi and S. Tasaka, "A media synchronization scheme with causality control in network environments," in *Proc. IEEE LCN'99*, pp. 232-241, Oct. 1999.

[2] Y. Lin, K. Guo, and S. Paul, "Sync-MS: Synchronized messaging service for real-time multi-player distributed games," in *Proc. IEEE ICNP'02*, Nov. 2002.

[3] K. Guo, S. Mukherjee, S. Rangarajan and S. Paul, "A fair message exchange framework for distributed multi-player games," in *Proc. ACM NetGames'03*, pp. 21-33, May 2003.

[4] C. Diot and L. Gautier, "A distributed architecture for multiplayer interactive application on the Internet," *IEEE Network*, vol. 13, no. 4, pp. 6-15, July/Aug. 1999.

[5] L. Pantel and L. C. Wolf, "On the suitability of dead reckoning schemes for games," in *Proc. ACM NetGames'02*, pp. 79-84, Apr. 2002.

[6] S. Singhal and M. Zyda, "Networked virtual environments designed and implementation," *ACM Press. SIGGRAPH Series*, pp. 127-144, 1999.

[7] M. Carson and D. Santay, "NIST Net - A Linux-based network emulation tool," *ACM SIGCOMM*, vol. 33, no. 3, pp. 111-126, July 2003.

[8] D. L. Mills, "Network Time Protocol (version 3) specification, implementation and analysis," RFC-1305, Mar. 1992.

[9] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proc. ACM NetGames'02*, pp. 23-29, Apr. 2002.

[10] Y. Ishibashi and S. Tasaka, "Causality and media synchronization control for networked multimedia games: Centralized versus distributed," in *Proc. ACM NetGames'03*, pp. 34-43, May 2003.