Consistent Synchronization of Action Order with Least Noticeable Delays in Fast-Paced Multiplayer Online Games

JINGXI XU and BENJAMIN W. WAH, Chinese University of Hong Kong

When running multiplayer online games on IP networks with losses and delays, the order of actions may be changed when compared to the order run on an ideal network with no delays and losses. To maintain a proper ordering of events, traditional approaches either use rollbacks to undo certain actions or local lags to introduce additional delays. Both may be perceived by players because their changes are beyond the just-noticeable-difference (JND) threshold. In this article, we propose a novel method for ensuring a strongly consistent completion order of actions, where *strong consistency* refers to the same completion order as well as the same interval between any completion time and the corresponding ideal reference completion time under no network delay. We find that small adjustments within the JND on the duration of an action would not be perceivable, as long as the duration is comparable to the network round-trip time. We utilize this property to control the vector of durations of actions and formulate the search of the vector as a multidimensional optimization problem. By using the property that players are generally more sensitive to the most prominent delay effect (with the highest *probability of noticeability* P_{notice} or the probability of correctly noticing a change when compared to the reference), we prove that the optimal solution occurs when P_{notice} of the individual adjustments are equal. As this search can be done efficiently in polynomial time (∼5ms) with a small amount of space (∼160KB), the search can be done at runtime to determine the optimal control. Last, we evaluate our approach on the popular open-source online shooting game BZFlag.

CCS Concepts: • **Information systems** → **Massively multiplayer online games**; • **Human-centered computing** → Collaborative interaction;

Additional Key Words and Phrases: Internet, consistency, human factors, quality of experience

ACM Reference Format:
Jingxi Xu and Benjamin W. Wah. 2016. Consistent synchronization of action order with least noticeable delays in fast-paced multiplayer online games. ACM Trans. Multimedia Comput. Commun. Appl. 13, 1, Article 8 (December 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/3003727

1. INTRODUCTION

Multiplayer online games (MMOs) refer to computer games with multiple players who interact remotely in the real world over the Internet. Among them, fast-paced online games are increasingly popular with improvements in network bandwidth and reduced latency. Here, *fast-paced games* refers to games in which the reaction time required is near the limit of human reaction time (215ms on average according to an online test [Human Benchmark 2016]). We are interested in those games with action durations ranging from 300 to 700ms, particularly scenarios with precise weapons. Examples include shooting games with bullets or missiles, fighting games with fast punching or kicking, and racing games with weapons shooting enemies.

Authors' addresses: J. Xu, SHB 115, Chinese University of Hong Kong. Shatin, N.T., Hong Kong; email: jxxu@cse.cuhk.edu.hk; B. W. Wah, Room 202, University Administration Building, Chinese University of Hong Kong. Shatin, N.T., Hong Kong; email: bwah@cuhk.edu.hk.

DOI: http://dx.doi.org/10.1145/3003727

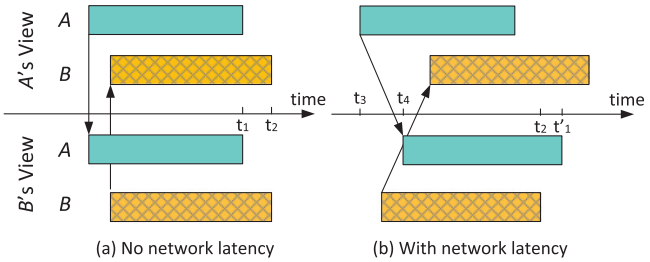


Fig. 1. Physical network latencies can delay the completion of actions and cause the reordering of completions that are different when compared to the reference order.

In these games, players interact in a common game world in virtual space, even though they may be separated in physical space in the real world. In virtual space, players perceive virtual time, which is not necessarily the same as physical time, as it can be affected by network and buffering latencies.

An *action* is a translation or a movement of a virtual object triggered by a player, which has some effects in virtual space. As an action will not have any effect before it completes, the order of actions is defined by their *completion order*.

We define the *reference order* of actions to be the order of completions in virtual space without network latency (i.e., in a virtual perfect world). Figure 1(a) illustrates the reference order in a two-player game in which two players are shooting at the same target. A shaded box shows the start, duration, and completion of an action that represents the shooting of a bullet in a player's view in virtual space. When there is no network latency, the messages of A's and B's actions are immediately sent to the other player's virtual space. In both spaces, A's and B's actions terminate at t_1, and t_2, respectively, where $t_2 > t_1$ defines the reference order. The example can be generalized to more than two players, and the orders perceived by all players in their virtual spaces are the same. In this case, virtual time is aligned to physical time.

In contrast, when there is network latency, the *actual order* of completions of actions in virtual space can be different from the reference order. Figure 1(b) shows that the messages of A's and B's actions are delayed by network latency $t_4 - t_3$ (assuming the same two-way latencies). In B's view, B's action still terminates at t_2, but A's action is now delayed by $t_4 - t_3$ and terminates at $t'_1 = t_1 + t_4 - t_3$. Note that $t'_1 > t_2$ implies an inconsistent order of completions between A and B when compared to the reference; in A's view, A's action terminates earlier than B's action, whereas in B's view, A's action terminates later. We call this phenomenon the *reordering problem* [Mauve et al. 2004].

Reordering is directly related to consistency and correctness in the continuous domain [Mauve et al. 2004]. Consistency requires the state (action order in our context) at time t to be the same in any two players' virtual spaces if both have completed all the operations supposed to be executed before t. Correctness further requires the state to be the same as the virtual perfect site (reference order in our context).

In this article, we define *strong consistency* to mean identical actual and reference orders (thus satisfying consistency and correctness defined in Mauve et al. [2004]), and that the interval between any two completion times is unchanged when compared to the reference order. The latter requirement is important for the following reasons:

(a) If the intervals of completion times of two actions are longer than those in the reference, then the delays in the corresponding actions will accumulate. A later action may therefore terminate significantly later, and its effect can be noticed.
(b) If an interval is shorter than the reference, then the deadline of the corresponding action is moved up, and there may not be sufficient time to process this action.

Our definition also applies to the traditional well-studied MMOs, where consistency of states over clients is important. Related work has been reviewed in a recent survey paper [Yahyavi and Kemme 2013]. Several techniques have been developed for maintaining consistency. Dead reckoning is useful for predicting an unreceived future state if the corresponding play pattern is relatively simple [Shi et al. 2014]. Rollbacks move virtual times backward and replay the actions in the correct order once reordering is detected. They have been widely adopted because they are easy to implement. Recent approaches have focused on smooth correction techniques that repair inconsistent states during a game. However, subjective studies have found that such corrections are noticeable and annoying. In general, important factors that affect players' detection of corrections [Savery 2014] include a player's locus of attention and the smoothness and duration of a correction.

The preceding approaches, however, can lead to noticeable inconsistencies in fast-paced games before the inconsistencies are fixed [Stuckel and Gutwin 2008]. These games have higher requirements on keeping their states consistent than MMOs because their action durations are short and comparable to the network latency. Players thus need to pay high attention to all fast-paced actions. For example, in a fighting game, a defender should keep changing the way she guards when defending the constantly changing attack patterns. Such a high attention level, as well as the short action durations, can render the inconsistency of rollback or correction noticeable [Savery 2014].

Several techniques have been developed to solve the preceding problem. The local-lag method [Mauve et al. 2004] maintains consistency by delaying an action in virtual space slightly after a player initiates the action. Although this can avoid inconsistencies when compared to rollback and smooth correction techniques, it can lead to noticeably sluggish response. Local perception filters [Smed et al. 2005], instead, modify the durations of actions to maintain consistency, but they can produce noticeable change of durations, especially in fast-paced actions. In this article, we call these *delay effects* because they cause players to feel "sluggishness" or "having undue slowness" in their games. These were also called *glitches* in some previous papers.

To demonstrate that the local-lag strategy can produce noticeable delay effects, we modified the code of an open-source online tank-battle game BZFlag [Myers et al. 2012] to implement the strategy. The game has both rapid actions and fast interactions. To better represent fast-paced games, we modified the base speed of a bullet in BZFlag from the original setting of 100 units/s to 300 units/s. Using bullets of different durations, we hired 14 students to perform subjective tests and asked each to choose which setting had slower response: one in a reference network without latency and another in a network with latency but with the local-lag strategy. If they could correctly identify the second setting, then the local-lag strategy was unable to conceal the delay effects.

Figure 2 displays the result using a just-noticeable-difference (JND) surface developed in our previous work [Xu and Wah 2015] (previously called *JND profile* and is renamed here to avoid confusion with just-noticeable distortion profiles in video coding [Jayant 1992]). It shows the *probability of noticeability* P_{notice}, or probability of subjects correctly noticing a change when compared to the reference. It was generated by testing a few representative combinations of action duration and network latency and by interpolating the P_{notice} of the remaining duration-latency pairs using a bicubic algorithm. It shows that the local-lag strategy cannot conceal the delay effects because 50% of the subjects can identify the scenario with a short 25ms one-way network latency, even when the action duration is 0.5 seconds (that covers about one third of the distance in the game map and is 20 times longer than the network latency).

Our goal in this article is to develop methods that can significantly reduce the perception of delay effects when strong consistency is maintained in fast-paced online

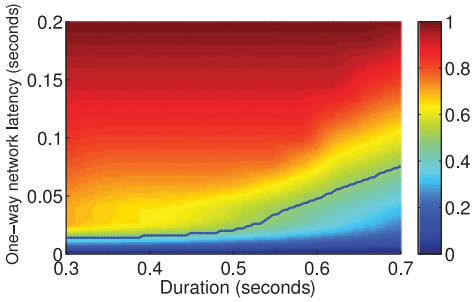


Fig. 2. P_{notice} when using the local-lag strategy to conceal the effects of network latency in BZFlag. The x-axis shows the original action duration common in all virtual spaces; the y-axis is the one-way network latency. The dark blue curve shows the contour $P_{\text{notice}} = 50\%$.

games. Our approach is to determine at runtime the best setting of control parameters that can conceal the delay effects. The possible combinations of control parameters to use are found by offline experiments evaluated by the probability of noticeability P_{notice} that can quantify players' subjective perception of delay effects (or quality of experience (QoE)). Our work is unique because it uses offline QoE results to determine the controls at runtime. In contrast, previous work has focused on studying the delay effects in online games [Teal and Rudnicky 1992; Lin et al. 2002; Chen and El Zarki 2011] as well as the QoE of general consistencies in multimedia applications [Huang et al. 2013]. However, their metrics were not used to optimize the setting of controls at runtime for concealing delay effects.

Our study is based on five assumptions that are general enough to cover many fast-paced interactive online games. However, they are not essential for designing strategies in slow-paced games, as these games have sufficient slacks for performing rollbacks and other smooth corrections without being noticed by players. The assumptions are as follows:

(a) We assume that network latencies in the near future are similar to those of the recent past (typically in the last few seconds). This assumption allows a priori setting of the size of delay buffers. Many previous studies [Stuckel and Gutwin 2008; Smed et al. 2005; Hariri et al. 2011] rely on this implicit assumption, which has also been verified in our previous work [Xu and Wah 2013b].
(b) We develop our methods with respect to one weapon. The methods developed can be generalized well to games with multiple weapons because changing weapons is a relatively slow action, and there is sufficient time to notify other players when a player changes her weapon.
(c) We assume that weapons are precise, that each player can attack one target at a time, and that multiple attackers can attack the same or multiple targets at the same time. We do not consider imprecise weapons (e.g., a boom or a field magic) that attack multiple targets at the same time. In this case, players do not care about the order of the attacks and the consistency of their completion times.
(d) An action is realized only when it completes, which is common for precise weapons. For instance, the effect of shooting a bullet is realized when the bullet arrives at the target. Without this, it is not possible to conceal delay effects by changing the timing or duration of an action.
(e) An attack action (e.g., punching or kicking in fighting games or shooting a bullet in shooting games) is much faster than the movements of avatars of players in virtual space. This means that the duration of an action does not vary much with respect to the movements of avatars. The assumption is reasonable, as the reaction times

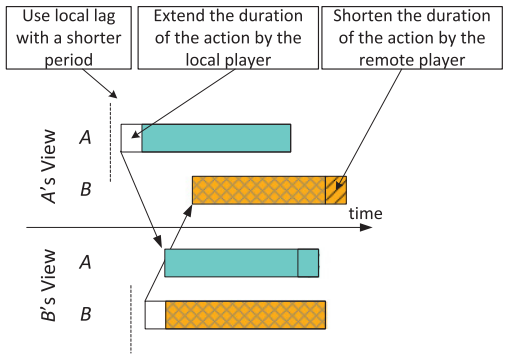


Fig. 3. Illustration of our proposed approach for solving the reordering problem.

in fast-paced games are near the limit of human reaction times, and such a short duration can only relate to an attack action rather than the movement of an avatar. Example action durations studied are 300 to 700ms, although slower-paced games with longer action durations can also benefit from our algorithms.

Problem statement. Based on these five assumptions, we study the reordering of action completions in fast-paced MMOs running in a network with latency. We study it under two cases: one in which the target's response is predictable and another in which it is not. In the first case, an attacker does not have to wait for the target's response before proceeding to her next action. On the other hand, in the second case, the outcome to an action is unknown until the target's response has been received (defined as the *blank period* in Section 2). This means that an attacker has to wait for the target's response before proceeding to her next action. Under each of these two cases, we study two related subproblems.

In the first subproblem, we develop an analytic method for achieving strong consistency at runtime.

Figure 3 illustrates the approach that corrects an inconsistent order by combining the local-lag strategy (by delaying the start of a player's action) and the local-perception filters (by extending the duration of her action and by shortening the duration of the other player's action). Because all of these are small adjustments, the delay effect due to each will be less noticeable than that caused by each adjustment when applied in isolation.

In the second subproblem, we develop a polynomial-time algorithm for finding the optimal multidimensional P_{notice} of delay effects caused by multiple controls that are perceived as a whole. Our approach decomposes the evaluation of the multidimensional P_{notice} into multiple simpler subproblems of evaluating P_{notice} of each control. We prove that when the target's response is predictable, optimality occurs when P_{notice} of all of its component controls are equal. This reduces the complexity of finding the optimal P_{notice} from exponential to linear. If the response is unpredictable, the optimal P_{notice} can still be found in polynomial time.

Our solution extends our previous work [Xu and Wah 2013a, 2013b] for solving the blank-period problem described earlier when the target's response is unknown ahead of time. Our previous approach reschedules the actions so that an attacker can have consistent information from a defender, without taking into account the reordering of action completions. Our current approach allows strong consistency to be enforced, both for targets with predictable and unpredictable responses.

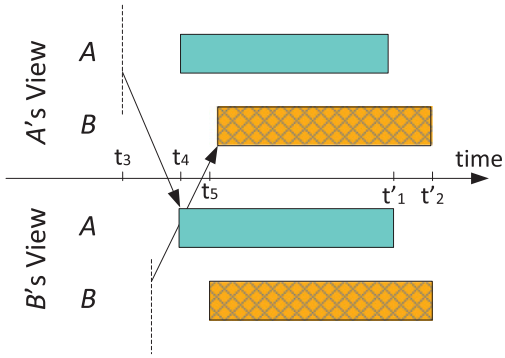


Fig. 4. In delaying an action by the longest network latency, local-lag algorithms solve the reordering problem by compensating the virtual delay caused by the network latency. However, the delay effect may be perceived by players, as a player has to wait for $t_4 - t_3$ before her action is carried out.

This article is divided into five sections. Section 2 summarizes the shortcomings of related studies. Sections 3 and 4 present our solutions for solving the reordering problem when a target's response is predictable and unpredictable. For simplicity, we use a shooting game as a running example but evaluate the algorithms using the online game BZFlag [Myers et al. 2012]. Finally, conclusions are drawn in Section 5.

2. PREVIOUS WORKS

2.1. Previous Approaches for Solving the Reordering Problem

There are four classes of approaches for solving the reordering problem.

Rollback methods [Mauve et al. 2004] solve the reordering problem by canceling the current actions leading to an incorrect outcome, reverting the affected states to those before reordering happens, and proceeding from then on in an order consistent with the reference. These methods have been found to cause significant delay effects in fast-paced games. When an existing outcome is canceled, a player may see its status changed from a winner to a loser or vice versa [Xu and Wah 2013a].

Smooth correction methods [Savery and Graham 2014] repair those inconsistent states after they have occurred by changing the speeds of later actions. They emphasize on local rather than global consistency, leading to smooth animations of actions in MMOs. However, in fast-paced interactive games, high players' attention to actions, as well as their short durations, may result in noticeable delay effects. Another work [Li et al. 2011] combined dead-reckoning and local-perception methods to maintain continuous synchronization. The resulting schedule of actions does not consider perceptual effects because it lacks a metric for measuring perception in its optimization.

Local-lag methods maintain consistency by delaying the launch of an action [Teal and Rudnicky 1992; Mauve et al. 2004; Stuckel and Gutwin 2008]. A typical algorithm involves two steps. It first predicts the longest physical network latency between any two clients with the network information collected in the last several seconds. Next, in each player's view, it delays all actions by an amount equal to the latency collected when compared to the reference. If the message of an action is received before this latency, it waits until the latency (called *synchronization delay*) is completed before starting the action. As all messages from the other sides should have been received at that moment, all actions can now be executed in the reference order.

Figure 4 illustrates the local-lag method. The longest network latency $(t_4 - t_3)$ is first estimated. In A's view, A's action is delayed by $t_4 - t_3$ before it is transmitted to B. B's action in A's view is then forced to be delayed by $\delta_{B,B} = t_4 - t_3$ to t_5 to ensure all

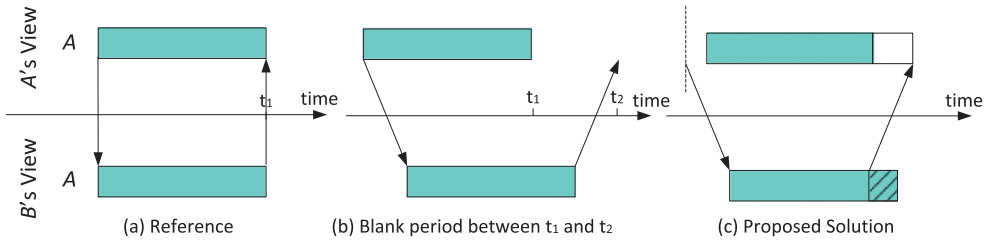


Fig. 5. Illustration of the blank-period problem in a fighting game and its proposed solution [Xu and Wah 2013a].

actions in A that should start before this action to have enough time to inform B. The completion times of A's and B's actions are t_1' and t_2' ($t_1' < t_2'$), respectively, which are the same as the reference order in Figure 1(a).

This strategy can lead to noticeable delay effects, as the late start of an action after initiated can be perceived as a sluggish response. Previous work has shown that players are very sensitive to such delays when playing as a first-person avatar in shooting games [Beigbeder et al. 2004; Stuckel and Gutwin 2008; Raaen and Grønli 2014].

System-level methods try to address the problem at the system level. One high-level approach used a distributed framework [Chan et al. 2007; Zhang et al. 2008], but it could not solve the reordering problem in a short period. Some [Chan et al. 2007] focused on efficiently distributing event messages but left the handling of synchronization to the application level. Others [Zhang et al. 2008] studied synchronization on low-level consistency, such as the positions of players. The distributed architecture Colyseus [Bharambe et al. 2006] maintained consistency in a large-scale shooting game Quake. It focused on action delays but did not solve the reordering problem, as it adopted "an optimistic consistency model with no bounds on order or numerical error in order to limit staleness as much as possible." Another work [Marshall et al. 2010] considered the consistency of distributed interactive applications but focused on optimizing network bandwidth. Efforts had been made to address the critical casual order of events [Zhou et al. 2007] but did not consider the human perception of delay effects.

2.2. Previous Approaches for Solving the Blank-Period Problem

The *blank-period problem* occurs when the response of a target is unpredictable at the time the action from an attacker is completed. We illustrate it using one attacker and one defender. There is no reordering here, as there is only one action from the attacker.

In the reference case with no network latency, an attack action terminates when the target's response is received. Figure 5(a) shows that A's action terminates at t_1 in both attacker A's and target B's views. In contrast, when there is latency, the action in the attacker's view terminates before the target's (unknown) response is received. Without knowing this response, there is a blank period in which the response cannot be displayed in the attacker's view. Figure 5(b) shows that the action in A's view terminates at t_1, whereas B's response is available at $t_2 = t_1 + RTT$. To ensure a correct outcome in A's view, the action in A's view should not terminate until B's response is received.

Consider the example when A shoots B in BZFlag and B will move to dodge the shot. When there is network delay, A does not know whether B has been shot before receiving B's message. Hence, A will need to wait for a blank period before seeing B's result, even after seeing the shot hitting B in her own view. This is different from her playing experience in offline games without network delay.

Traditional methods for solving the reordering problem can also be adopted to solve the blank-period problem. The local-lag [Mauve et al. 2004] and the local

perception-filter [Smed et al. 2005] methods can reschedule actions to let them complete after the target's response has been received. However, as mentioned in Section 2.1, these methods may lead to significant delay effects that are noticeable by players.

To this end, our previous work [Xu and Wah 2013a] proposed to delay and extend the attacker's action, and to shorten the target's action (Figure 5(c)). In this way, the blank period is covered by three smaller adjustments that are less noticeable than a single large adjustment of extending the attacker's action. However, the method does not address the strong consistency of action completions. However, it inspires us to study in this work the use of multiple smaller adjustments for maintaining strong consistency without being noticed.

2.3. Measuring Delay Effects Using Probability of Noticeability

JND is a well-known concept in psychophysics for measuring perceptual effects. Traditionally, JND rests on perceptual attention, although it is arguable whether some representative aspects of attention are directly related to consciousness. Further, JND was developed using some oversimplifications in its classical models, including Weber's law and Steven's power law. In short, JND by itself is inadequate for representing subjective QoE.

Instead of using JND and its classical theory, here we evaluate perceptual differences by the probability of subjects noticing a change in pairwise comparisons of two alternatives whose difference is captured by JND.

Specifically, we employ a subjective-test method called *constant stimuli* for measuring JND. We show pairs of original and changed settings in a random order to subjects and ask them to identify the changed setting. With a given setting R as *reference* and a *modification* M that results in a changed setting $R + M$, we define *awareness* $p(R, M)$ [Xu and Wah 2015] to measure the probability a subject can identify the changed setting when we present the two settings in a random order. Rather than arguing that JND is a hard threshold that differentiates between noticing or not noticing a change, we describe the result of subjective tests in a probabilistic form. In this way, we directly measure the effects of subjective QoE instead of deriving QoE by JND.

Note that awareness p by itself is larger than the probability of subjects who can actually notice a change, as there are subjects who give the correct answer by random guesses. Without random guesses, there are $P_{\text{notice}} = 2p - 1$ (defined as the *probability of noticeability*) of those subjects who can actually notice the change. Assuming the responses of subjects to be independent and identically distributed, $P_{\text{notice}}N$ of the N subjects can actually notice the change, whereas $(1-P_{\text{notice}})N$ answer by random guesses. Hence, $P_{\text{notice}}N \times 1 + (1-P_{\text{notice}})N \times 0.5 = pN \Rightarrow P_{\text{notice}} = 2p - 1$ [Xu and Wah 2015]. When the modified setting is the same as the original ($M = 0$), 0% of the subjects can notice the change and everyone responds by random guesses; hence, $p = 50\%$ ($\Rightarrow P_{\text{notice}} = 0$). In psychophysics, $p = 75\%$ ($\Rightarrow P_{\text{notice}} = 0.5$) is generally used as a level of noticeable change. In short, p and P_{notice} are equivalent representations of noticeability.

In our previous work, we developed a graphical method called *JND surface* [Xu and Wah 2015] that amalgamates all combinations of reference R, modification M, and awareness p into a 3D graph $p(R, M)$. To avoid confusion, here we represent the 3D graph using the probability of noticeability $P_{\text{notice}}(R, M)$. Figure 2 illustrates such a surface that shows the fraction of subjects who can actually identify the change after using the local-lag strategy in solving the reordering problem. Its x-axis R shows the duration of an attack action, and its y-axis M shows the one-way latency from an attacker to a target (assumed to be identical in both directions) that needs to be concealed by the local-lag strategy; the color indicates the corresponding $P_{\text{notice}}(R, M)$.

This surface allows a succinct presentation of the performance of the local-lag strategy across all combinations of action duration and network latency.

Figure 2 further shows that a larger M is more likely to be noticed under a smaller R. This *monotonicity* property [Xu and Wah 2015] in terms of P_{notice} is stated as follows.

AXIOM 1. *(a) $P_{notice}(R, M)$ is monotonically nonincreasing with respect to reference R, as a given modification M is less noticeable under a larger R. (b) $P_{notice}(R, M)$ is monotonically nondecreasing with respect to M, as a larger M is more noticeable under a given R. (c) For multidimensional modifications, $P_{notice\text{-}f}(ref, m_1, m_2, \ldots, m_n)$ is monotonically nondecreasing when all $m_i, i = 1, \ldots, m$, are nondecreasing.*

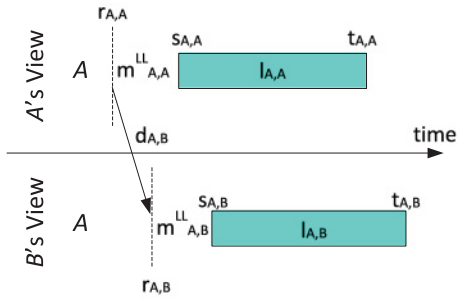
To generate a JND surface, in earlier work [Xu and Wah 2013a] we proposed a brute-force method for sampling P_{notice} of many (R, M) pairs and to interpolate P_{notice} for all intermediate combinations. In our past experiments, 16 subjective tests had to be carried out by each subject to generate a surface with adequate accuracy. An effective greedy approach was later developed [Xu and Wah 2015, 2016] to use the monotonicity property to heuristically sample a much smaller number of combinations (around 7) and to achieve the same level of accuracy. Another concept called the *just-noticeable distortion profile* developed in video coding [Jayant 1992; Chou and Li 1995; Yang et al. 2005; Ma et al. 2011] has a similar name but is different from the JND surface used here. A JND profile is defined by closed-form models that can be used for optimizing control assignments at runtime. Such models were acquired through previous analysis on structural characteristics of signals that are specific to video coding. They are hard to get in general multimedia applications. A different study on the quality of perception [Gulliver and Ghinea 2006] explored the human perception on distributed multimedia quality but focused on video streaming and cannot be adopted for optimizing online games.

2.4. Multi-Dimensional JNDs

The generation of a JND surface is more challenging when there are multiple controls to be adjusted, as each control will be a dimension in the surface. For instance, in Figure 3, there are three places in the actions to be controlled, resulting in a 5D surface and an exponential increase in complexity with respect to the number of controls.

When multiple controls are applied together, their effects are perceived by humans as a whole. Some previous studies proposed to combine their effects on JND by taking the maximum JND. They argued that humans tended to notice the most significant change when there were multiple changes [Chou and Li 1995]. This is not always true in practice, as P_{notice} of a large change in one control may not be larger than P_{notice} of a smaller change in another control. Note that the chance of perceiving a change is based on P_{notice}, not on the magnitude of the change. Some other methods used square root to calculate the combined JND [von Helmholtz 1891]. An integration was also proposed when an explicit JND function was given [Dzhafarov and Colonius 1999].

All previous methods considered the combined changes in determining their overall effect. Our analysis, however, shows that P_{notice} is a true indication on whether a subject can (probabilistically and actually) detect a change. Hence, a better approach to determine the overall effect is to utilize its P_{notice}. By assuming that subjects will notice the change with the maximum P_{notice} when there are multiple changes, we propose to decompose the evaluation of a multidimensional P_{notice} into the evaluation of individual P_{notice}'s, each corresponding to one control assignment. For the case when the target's response is predictable, we prove that the minimum multidimensional P_{notice} occurs when the individual component P_{notice}'s are equal.



Symbol	Definition
$r_{A,B}$	Time when A's action is ready in B's virtual space
$m^{\text{strategy}}_{A,B}$	Modification of time in A's action when shown in B's virtual space using the strategy defined
$s_{A,B}$	Actual starting time of A's action and shown in B's virtual space
$l_{A,B}$	Duration of A's action when shown in B's virtual space
$t_{A,B}$	Completion time of A's action when shown in B's virtual space
$d_{A,B}$	Physical network latency from A to B
LL	Local-lag strategy
LPF1	Strategy that extends the duration of action
LPF2	Strategy that shortens the duration of action
ATK	Attacker
DEF	Defender

Fig. 6. Symbols defined on the action performed by player A in A's and B's virtual spaces, respectively.

3. SOLVING THE REORDERING PROBLEM FOR TARGETS WITH PREDICTABLE RESPONSE

In this and the next sections we solve the reordering problem in fast-paced multiplayer games. By extending the local player's action while shortening the corresponding remote player's action, we show that delay effects can be made less aware while strong consistency can be maintained. We divide our discussion into two scenarios. In this section, we assume that the target's response is *predictable*, which means that the attacker does not have to wait for the target's response before starting her next action. In Section 4, we assume that the target's action is unpredictable. To maintain strong consistency without rollback or correction, the attacker needs to wait for the target's response before initiating her next action.

Figure 6 summarizes the symbols defined on actions. Let $s_{A,B}$, $r_{A,B}$, and $t_{A,B}$ respectively be the starting time, time it is ready, and completion time of A's action in B's virtual space. In B's virtual space, $r_{A,B} = r_{A,A} + d_{A,B}$, where $d_{A,B}$ is the network latency between A and B. Note that $s_{A,B}$ and $r_{A,B}$ may not be the same.

We aim to design control strategies to improve $P_{\text{notice}}(R, M)$. We identify the strategy used to control M by its superscript and omit it when obvious. For example, $P_{\text{notice}}(R, M^{\mathbf{LL}})$ refers to the case when $M^{\mathbf{LL}}$ is controlled by the local-lag strategy.

Note that P_{notice} represents the probability of noticeability of a strategy used to control M and is common for all players using this strategy. A player using the strategy will have her P_{notice} represented by a JND surface, which is obtained by instantiating $P_{\text{notice}}(R, M)$ to $P_{\text{notice}}(ref, m)$, where ref and m correspond to the reference and modification used by the player. We further use the second superscript to distinguish the role of a player in whose view the modification is made (by default, the role is attacker **ATK**). The subscripts A, B are similar to those used in actions. For example, $P_{\text{notice}}(ref^{\mathbf{ATK}}_{A,B}, m^{\mathbf{LPF1,ATK}}_{A,B})$ stands for P_{notice} when the action duration of attacker A in B's view has changed from $ref^{\mathbf{ATK}}$ to $ref^{\mathbf{ATK}} + m^{\mathbf{LPF1,ATK}}$. We omit the subscripts when the player and the view are obvious.

Since the defenders studied in this section have predictable responses, all actions are by default initiated by attackers, and it is not necessary to indicate their role. In Section 4, the role of a player is to be explicitly indicated (whether she is an attacker or a defender) when a target's action is not predictable.

3.1. Maintaining Strong Consistency by the Local-Lag Strategy

In the reordering problem illustrated in Figure 1, network latency causes the actual order to be inconsistent with the reference order. We prove in this section that the local-lag method [Mauve et al. 2004] can always maintain strong consistency. Let $m^{\mathbf{LL}}_{i,j}$

ALGORITHM 1: Proposed Strategy for Solving the Reordering Problem

Require: Offline-measured JND surfaces instantiated from the strategies based on local lag $P_{\text{notice}}(R, M^{\mathbf{LL}})$ and local perception filters $P_{\text{notice}}(R, M^{\mathbf{LPF1}})$ and $P_{\text{notice}}(R, M^{\mathbf{LPF2}})$;
Ensure: The extent of modification due to local lag ($m^{\mathbf{LL}}$) and local perception filters ($m^{\mathbf{LPF1}}$ and $m^{\mathbf{LPF2}}$);
1: Estimate the duration of actions using (7);
2: Estimate network latency;
3: **if** action is carried out by a local player **then**
4: Search for the optimal $m^{\mathbf{LL}}$ and $m^{\mathbf{LPF1}}$ using (20) and (21);
5: **else**
6: Search for the optimal $m^{\mathbf{LPF2}}$ using (20) and (21);
7: **end if**
8: Modify the action using the $m^{\mathbf{LL}}$, $m^{\mathbf{LPF1}}$ and $m^{\mathbf{LPF2}}$ found.

be the local lag before an action is started. We have

$$s_{i,j} = r_{i,j} + m^{\mathbf{LL}}_{i,j}. \tag{1}$$

We first define the term *synchronization delay* before proving the theorem.

Definition 1. The synchronization delay $\Delta t_{i,j}$ of i's action in j's virtual space is the delay between the time when the action is initiated by i in her view to the time when this action is started in j's view:

$$\Delta t_{i,j} \;=\; s_{i,j} - r_{i,i} = r_{i,j} + m^{\mathbf{LL}}_{i,j} - r_{i,i} = r_{i,i} + d_{i,j} + m^{\mathbf{LL}}_{i,j} - r_{i,i} = d_{i,j} + m^{\mathbf{LL}}_{i,j}. \tag{2}$$

THEOREM 3.1. *The local-lag strategy can maintain a strongly consistent order with respect to the reference when the synchronization delay of i's action in j's virtual space is* $D_{\max} = \max_{x,y} d_{x,y}$*, the maximum one-way latency between any two clients. In other words, strong consistency is maintained by the local-lag strategy when*

$$m^{\mathbf{LL}}_{i,j} = \Delta t_{i,j} - d_{i,j} = D_{\max} - d_{i,j}. \tag{3}$$

PROOF. According to Definition 1, the maximum $\Delta t_{i,j}$ is governed by $D_{\max}$. In other words, in the worst case, the local-lag strategy will need to conceal $\Delta t_{i,j} = D_{\max}$. To ensure strong consistency in all views, we have i's action terminate in j's virtual space at

$$t_{i,j} \;=\; t_{i,i} + \Delta t_{i,j} \;=\; t_{i,i} + D_{\max}. \tag{4}$$

When $\Delta t_{i,j} = D_{\max}$, the delay between the completions of p's and q's actions is

$$t_{p,j} - t_{q,j} = t_{p,p} + \Delta t_{p,j} - (t_{q,q} + \Delta t_{q,j}) = t_{p,p} - t_{q,q}. \tag{5}$$

This proves that the delay is strongly consistent with that of the reference. □

The long synchronization delay specified in Theorem 3.1 may lead to noticeable delay effects. To address this issue, we next propose a better strategy that can maintain strong consistency while incurring less noticeable delay effects.

3.2. Proposed Strategy for Maintaining Strong Consistency

We first show a necessary and sufficient condition for maintaining strong consistency. This condition can be calculated at runtime based on information estimated from the recent past. Using this condition, we show that the duration of actions can be adjusted to maintain strong consistency.

Algorithm 1 presents our proposed strategy, which has been illustrated in Figure 3.

3.2.1. Necessary and Sufficient Condition for Maintaining Strong Consistency. To ensure that our proposed strategy can maintain a strongly consistent order with respect to the reference, we first evaluate the completion time of the action initiated by i to j:

$$t_{i,j} \;=\; s_{i,j} + l_{i,j} = r_{i,i} + d_{i,j} + m_{i,j}^{\mathbf{LL}} + l_{i,j}. \tag{6}$$

Note that j does not know $r_{i,i}$ and $t_{i,i}$ $(i \neq j)$ before receiving the message of that action. However, $l_{i,j}$ is predictable, as the duration of an action can be estimated by the speed of the shot and the virtual distance between i and j in the recent past and does not vary much with respect to the movements of players (Assumption (e) in Section 1). Let $dist_{i,j}(t-1)$ and $dist_{i,j}(t)$ respectively be the virtual distances between i and j in the previous and the current time windows:

$$l_{i,j} = dist_{i,j}(t)/v_{i,j} \approx dist_{i,j}(t-1)/v_{i,j}. \tag{7}$$

The necessary and sufficient condition for maintaining strong consistency is as follows.

THEOREM 3.2. *Strong consistency in targets with predictable responses can be maintained by the local-lag strategy if and only if*

$$d_{p,j} + m_{p,j}^{\mathbf{LL}} + l_{p,j} - d_{q,j} - m_{q,j}^{\mathbf{LL}} - l_{q,j} \;=\; m_{p,p}^{\mathbf{LL}} + l_{p,p} - m_{q,q}^{\mathbf{LL}} - l_{q,q}. \tag{8}$$

PROOF. According to (5), to maintain strong consistency for players p and q,

$$t_{p,j} - t_{q,j} = t_{p,p} - t_{q,q}. \tag{9}$$

By expanding both sides of (9), we have

$$t_{p,j} - t_{q,j} \;=\; r_{p,p} + d_{p,j} + m_{p,j}^{\mathbf{LL}} + l_{p,j} - \left(r_{q,q} + d_{q,j} + m_{q,j}^{\mathbf{LL}} + l_{q,j}\right), \tag{10}$$

$$t_{p,p} - t_{q,q} \;=\; r_{p,p} + m_{p,p}^{\mathbf{LL}} + l_{p,p} - \left(r_{q,q} + m_{q,q}^{\mathbf{LL}} + l_{q,q}\right). \tag{11}$$

The theorem is proved by substituting (10) and (11) into (9). □

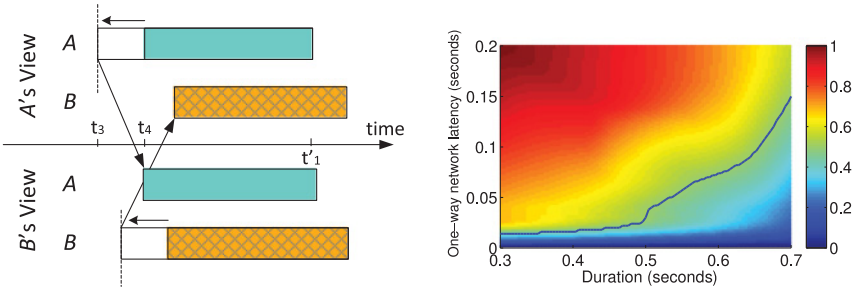
By substituting (3) into (8), strong consistency can be enforced by the local-lag strategy before the message of i's action is received by j. This is possible because $d_{i,j}$ is predictable (Assumption (a)) and (8) does not have any unknown terms involving r.

An important observation of (8) is that both $m_{i,j}^{\mathbf{LL}}$ and $l_{i,j}$ can be changed without violating strong consistency, as long as (8) is satisfied. In the rest of this section, we present methods to do several small changes on the durations of actions to make the overall effect less noticeable. We prove the correctness of the proposed strategy in two steps. First, we prove that the reordering problem can be solved by extending or by shortening the duration of action(s). This provides the basis for proving the correctness of the combined strategy in the second step. We further show that P_{notice} of the combined strategy can be significantly reduced.

3.2.2. Maintaining Strong Consistency by Extending the Durations of Actions. Figure 7(a) illustrates the strategy. Starting from the local-lag setting in Figure 4, instead of delaying the start of a local action, we extend the action in each player's view to cover the empty period before it starts while keeping the completion time unchanged.

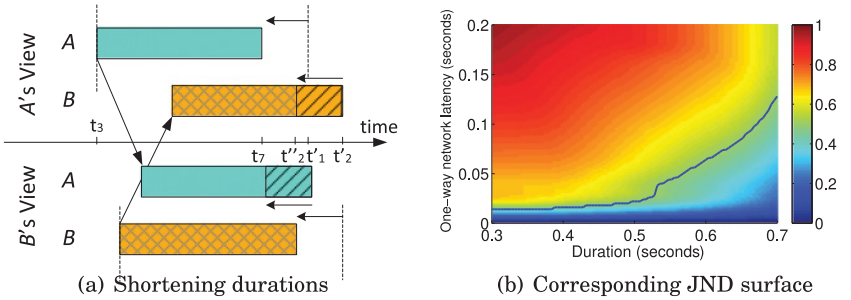
Figure 7(b) shows the JND surface after conducting subjective tests to measure subjects' sensitivity on detecting delay effects.

When using this strategy, the optimal extension is exactly the one-way latency from the player who started the action to the receiver. A longer period will lead to a larger change and make more players notice the delay effects, whereas a shorter extension is infeasible due to the definition of strong consistency.



(a) Extending durations (b) Corresponding JND surface

Fig. 7. Strong consistency can be maintained by extending the duration of actions. (a) Starting from the state in Figure 4 where actions are adjusted by the local-lag strategy, we extend the duration of the local actions (shown as unshaded boxes, each with an arrow). In A's view, A's action now starts at t_3 like that in the reference but still ends at t'_1. We readjust the duration similarly in B's view. Strong consistency is maintained because the completion times are not changed from the local-lag strategy. (b) P_{notice} due to extending the durations of actions in BZFlag is lower than that of the local-lag method (see Figure 2, whose axes are defined in the same way). The curve shows the contour $P_{\text{notice}} = 50\%$.



(a) Shortening durations (b) Corresponding JND surface

Fig. 8. Strong consistency can be maintained by shortening the durations of actions. (a) Starting from the state in Figure 4 whose actions are controlled by the local-lag strategy, we shorten the durations of remote actions (shown as shaded boxes). In A's view, B's action now completes at t''_2 instead of t'_2, without changing its starting time. Because the duration of B's action can be estimated in A's view (Assumption (e)), A can also start its action earlier and allows it to complete at t_7 as in the reference. Hence, the order is still strongly consistent. (b) P_{notice} due to shortening the durations of actions in BZFlag is lower than that of the local-lag method (see Figure 2, whose axes are defined in the same way). The curve shows the $P_{\text{notice}} = 50\%$ contour.

Due to space limitation, we state without proof the following corollary on the correctness of maintaining strong consistency. (See Figure 7(a) on the idea of the proof.)

COROLLARY 3.3. *Starting from the state in which actions are adjusted by the local-lag strategy, extending the starting times of local actions will not change the strong consistency of completions with respect to the reference.*

3.2.3. Maintaining Strong Consistency by Shortening the Durations of Actions. Figure 8(a) illustrates the strategy. Starting from the state in Figure 4 where actions are adjusted by the local-lag strategy, we start the local action in each player's view earlier but terminate the other player's action earlier while keeping its starting time unchanged.

Figure 8(b) shows the JND surface after conducting subjective tests to measure subjects' sensitivity on detecting delay effects. Similar to before, the optimal period to shorten the durations of actions is exactly the one-way network latency from the player who started the action to the receiver.

Due to space limitation, we state without proof the following corollary on the correctness of maintaining strong consistency. (See Figure 8(a) on the idea of the proof.)

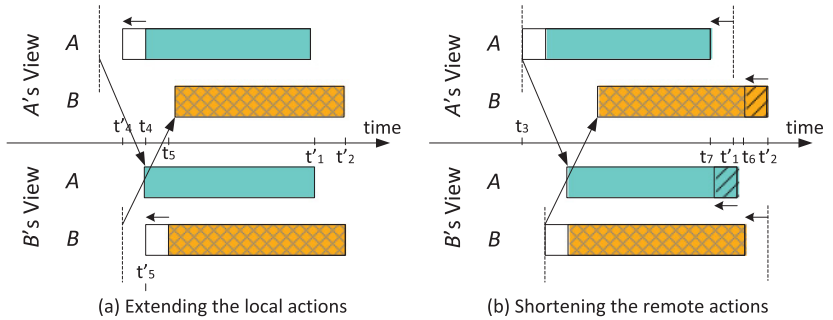


Fig. 9. In every virtual space, we extend the local actions and start them earlier while shortening the remote actions to reduce their synchronization delay. (a) Starting from the state in Figure 4, we extend the durations of local actions (shown as unshaded boxes). In A's view, A's action now starts at t'_4. When this is small compared to the original duration $t'_1 - t_4$, it is less likely to be noticed. Similarly, B's local action is extended. (b) Starting from the state in (a), we shorten the remote actions. In A's view, we move B's completion time from t'_2 to t_6. Because the duration of B's action can be estimated in A's view, A can also start her action earlier by the same amount (from t'_1 to t_7). Hence, the order is still strongly consistent. Similarly, in B's view, we can do the same modification.

COROLLARY 3.4. *Starting from the state in which actions are adjusted by the local-lag strategy, shortening the remote action and starting the local action earlier will not change the strong consistency of completions with respect to the reference.*

3.2.4. Maintaining Strong Consistency in the Combined Strategy. The three strategies (local-lag, extending and shortening an action) can be combined to solve the reordering problem. Figure 9 shows the corresponding adjustments. Based on the state in which actions are adjusted by the local-lag strategy, we can extend a player's action or start it earlier while shortening the other player's action. By comparing this figure to Figure 1(a), we find identical orders of action completions and intervals between action completions in each view, showing the correctness of the proposed approach.

According to (3),the maximum synchronization delay to be concealed by the local-lag strategy alone is $m^{\mathbf{LL}}_{i,i} = D_{\max} = \max_{x,y} d_{x,y}$. After adjusting the durations of actions, the new local-lag delay in i's virtual space becomes significantly shorter:

$$m^{\mathbf{LL}}_{i,i} = D_{\max} - m^{\mathbf{LPF1}}_{i,i} - m^{\mathbf{LPF2}}_{j,i}. \tag{12}$$

This means that the combined approach leads to a smaller change in each of $m^{\mathbf{LL}}_{i,j}$, $m^{\mathbf{LPF1}}_{p,p}$, $m^{\mathbf{LPF2}}_{q,q}$, resulting less noticeable delay effects due to each change.

The correctness of the combined strategy is shown in the following theorem, which can be proved by combining the proofs of Corollaries 3.3 and 3.4.

THEOREM 3.5. *The order of completions in the combined strategy is strongly consistent with respect to the reference if and only if (12) is satisfied.*

Similarly to the last two strategies, the optimal setting in the combined strategy should exactly fill $D_{\max}$. Any deviation will lead to undesirable delay effects.

3.3. Optimizing the Combined Strategy

As shown in (12), the combined strategy entails the control of the durations and the delays of actions while satisfying strong consistency. In this section, we develop methods for controlling these changes to make the overall delay effect the least noticeable.

We like to find $P^{\mathbf{COMB}}_{\text{notice}}(ref, m)$, the JND surface of the combined strategy. As $m = (m^{\mathbf{LL}}, m^{\mathbf{LPF1}}, m^{\mathbf{LPF2}})$, the 5D JND surface will be expensive to measure by subjective

tests. Further, the controls in m may have dependent effects. Since we do not know their dependence, we define $P^{\mathbf{COMB}}_{\text{notice}}(ref, m)$ using function f without a closed form:

$$P^{\mathbf{COMB}}_{\text{notice-f}}(ref, m) = f(P'_{\text{notice}}(ref, m^{\mathbf{LL}}), P'_{\text{notice}}(ref, m^{\mathbf{LPF1}}), P'_{\text{notice}}(ref, m^{\mathbf{LPF2}})), \tag{13}$$

$$\text{where} \begin{cases} P'_{\text{notice}}(ref, m^{\mathbf{LL}}) = P_{\text{notice}}(ref, m^{\mathbf{LL}} \mid m^{\mathbf{LPF1}} = 0, m^{\mathbf{LPF2}} = 0) \\ P'_{\text{notice}}(ref, m^{\mathbf{LPF1}}) = P_{\text{notice}}(ref, m^{\mathbf{LPF1}} \mid m^{\mathbf{LL}} = 0, m^{\mathbf{LPF2}} = 0) \\ P'_{\text{notice}}(ref, m^{\mathbf{LPF2}}) = P_{\text{notice}}(ref, m^{\mathbf{LPF2}} \mid m^{\mathbf{LL}} = 0, m^{\mathbf{LPF1}} = 0). \end{cases} \tag{14}$$

The following two lemmas can be derived directly from Axiom 1.

LEMMA 3.6.

$$\begin{aligned} &Let \quad P_{max} = \max\{P'_{notice}(ref, m^{\mathbf{LL}}), P'_{notice}(ref, m^{\mathbf{LPF1}}), P'_{notice}(ref, m^{\mathbf{LPF2}})\}, \\ &then \quad \begin{cases} \hat{m}^{\mathbf{LL}} \geq m^{\mathbf{LL}} & for\ P'_{notice}(ref, \hat{m}^{\mathbf{LL}}) = P_{max} \\ \hat{m}^{\mathbf{LPF1}} \geq m^{\mathbf{LPF1}} & for\ P'_{notice}(ref, \hat{m}^{\mathbf{LPF1}}) = P_{max} \\ \hat{m}^{\mathbf{LPF2}} \geq m^{\mathbf{LPF2}} & for\ P'_{notice}(ref, \hat{m}^{\mathbf{LPF2}}) = P_{max}. \end{cases} \end{aligned} \tag{15}$$

LEMMA 3.7. *Using the preceding definitions, we have*

$$P^{\mathbf{COMB}}_{notice\text{-}f}(ref, m^{\mathbf{LL}}, m^{\mathbf{LPF1}}, m^{\mathbf{LPF2}}) \leq P^{\mathbf{COMB}}_{notice\text{-}f}(ref, \hat{m}^{\mathbf{LL}}, \hat{m}^{\mathbf{LPF1}}, \hat{m}^{\mathbf{LPF2}}). \tag{16}$$

The following assumption (to be validated later by subjective tests) is based on the observation that in fast-paced games, subjects will only notice the dominant delay effect but not those due to individual controls when compared to the reference.

ASSUMPTION 1. *Given $\hat{m} = (\hat{m}^{\mathbf{LL}}, \hat{m}^{\mathbf{LPF1}}, \hat{m}^{\mathbf{LPF2}})$, then $P^{\mathbf{COMB}}_{notice\text{-}f}(ref, \hat{m})$ is equal to the maximum of the three individual noticeabilities when they are equal*:

$$P^{\mathbf{COMB}}_{\text{notice-f}}(ref, \hat{m}) = \max\{P'_{\text{notice}}(ref, \hat{m}^{\mathbf{LL}}), P'_{\text{notice}}(ref, \hat{m}^{\mathbf{LPF1}}), P'_{\text{notice}}(ref, \hat{m}^{\mathbf{LPF2}})\}. \tag{17}$$

COROLLARY 3.8. $P^{\mathbf{COMB}}_{notice\text{-}f}(ref, \hat{m}) = P_{max}$.

The proof is straightforward by applying Assumption 1 and Lemma 3.6.

COROLLARY 3.9.

$$P^{\mathbf{COMB}}_{notice\text{-}f}(ref, m) \leq \max\{P'_{notice}(ref, m^{\mathbf{LL}}), P'_{notice}(ref, m^{\mathbf{LPF1}}), P'_{notice}(ref, m^{\mathbf{LPF2}})\} \tag{18}$$

PROOF. This can be proved by combining Lemma 3.7 and Corollary 3.8. □

Our goal in the optimization is to minimize $P^{\mathbf{COMB}}_{\text{notice}}(ref, m)$ under given ref:

$$\mathcal{P} = \min\ P^{\mathbf{COMB}}_{\text{notice}}(ref, m) \text{ subject to } m = m^{\mathbf{LL}} + m^{\mathbf{LPF1}} + m^{\mathbf{LPF2}} = D_{\max}. \tag{19}$$

Without knowing the closed form of function f in (13), the best we can do is minimize the upper bound of f. According to Corollary 3.9, we have

$$\overline{\mathcal{P}} = \min\ \max\{P'_{\text{notice}}(ref, m^{\mathbf{LL}}), P'_{\text{notice}}(ref, m^{\mathbf{LPF1}}), P'_{\text{notice}}(ref, m^{\mathbf{LPF2}})\}, \tag{20}$$

$$\text{subject to } m = m^{\mathbf{LL}} + m^{\mathbf{LPF1}} + m^{\mathbf{LPF2}} = D_{\max}. \tag{21}$$

The following theorem proves the optimal solution to (20) and (21).

THEOREM 3.10. *The optimal solution to (20) and (21) is $(\overline{m}^{\mathbf{LL}}, \overline{m}^{\mathbf{LPF1}}, \overline{m}^{\mathbf{LPF2}})$, where*

$$P'_{notice}(ref, \overline{m}^{\mathbf{LL}}) = P'_{notice}(ref, \overline{m}^{\mathbf{LPF1}}) = P'_{notice}(ref, \overline{m}^{\mathbf{LPF2}}). \tag{22}$$

PROOF. The proof is by contradiction. If (22) is false, then without loss of generality, we assume that $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF1}})$ and $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF2}})$.

Based on Axiom 1(b), we know that $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}})$, $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF1}})$, and $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF2}})$ respectively are monotonically nondecreasing with increasing $\overline{m}^{\mathbf{LL}}$, $\overline{m}^{\mathbf{LPF1}}$, and $\overline{m}^{\mathbf{LPF2}}$. To get the optimal solution, $\overline{m}^{\mathbf{LL}}$ should be reduced as much as possible. However, as $\overline{m}^{\mathbf{LL}}$ is reduced, $\overline{m}^{\mathbf{LPF1}}$ and $\overline{m}^{\mathbf{LPF2}}$ will be increased due to (21). These lead to larger $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF1}})$ and $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF2}})$.

As we assume that $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF1}})$ and $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF2}})$, the optimal solution of (20) and (21) is when $\overline{\mathcal{P}} = P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) = P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF1}}) + \delta_1 = P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LPF2}}) + \delta_2$. However, we can always find $\delta_1 > \delta'_1 > 0$ and $\delta_2 > \delta'_2 > 0$ such that $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{\overline{m}}^{\mathbf{LL}}) = P'_{\text{notice}}(ref, \overline{\overline{m}}^{\mathbf{LPF1}}) + \delta'_1 = P'_{\text{notice}}(ref, \overline{\overline{m}}^{\mathbf{LPF2}}) + \delta'_2$, which results in $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}}) > P'_{\text{notice}}(ref, \overline{\overline{m}}^{\mathbf{LL}})$. Therefore, $P'_{\text{notice}}(ref, \overline{m}^{\mathbf{LL}})$ is not optimal. Contradiction! □

Theorem 3.10 allows us to find the optimal solution to the upper bound of $P^{\mathbf{COMB}}_{\text{notice-f}}$ at any (ref, m) when given the three JND surfaces $P'_{\text{notice}}(ref, m^{\mathbf{LL}})$, $P'_{\text{notice}}(ref, m^{\mathbf{LPF1}})$, and $P'_{\text{notice}}(ref, m^{\mathbf{LPF2}})$. We search over the three surfaces to find m that satisfies (21) and (22). In each surface, for a given ref, we first build a bidirectional graph that connects each m to the corresponding P'_{notice} (discretized to k_3 levels). The complexity is $O(k_1 k_2)$, where k_1 and k_2 are the discretization levels of the JND surface along the ref and m axes, respectively. Then we enumerate the k_3 levels of P'_{notice}, starting from the highest value, and find the first P'_{notice} that satisfies (21). At the same time, we use the graph to find the corresponding m. The complexity is $O(k_3)$.

In short, the complexity is linear with respect to k_3, the level of discretization in P'_{notice}. In our implementation, $k_3 = 101$ is sufficiently high, which corresponds to a 4ms interval in the x-axis and 2ms in the y-axis. The search can be done within 1ms by a desktop computer with an Intel Core 2 Duo E8400 3GHz CPU. As we use at most four JND surfaces, each with around 40KB, all surfaces can be stored in main memory. Hence, the search can be done in real time using the offline collected JND surfaces.

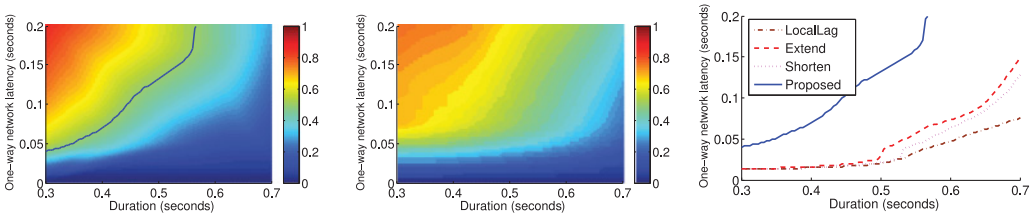
3.4. Experimental Evaluations on BZFlag

We present experimental results on evaluating the combined strategy on BZFlag [Myers et al. 2012], based on the three JND surfaces found by subjective tests. We measure the combined surface using the optimal setting in (22) for each ref. As the combined surface still follows Axiom 1, we conduct subjective tests to sample some critical points and approximate the surface using our previous algorithm [Xu and Wah 2015].

The results shown in Figure 10(a) are significantly better than those of methods using the local lag and local-perception filters (Figures 2, 7(b), and 8(b)).

To validate Assumption 1, we first show in Figure 10(a) the combined JND surface obtained by subjective tests based on the left-hand side of (17). As f is unknown, we directly measure $P^{\mathbf{COMB}}_{\text{notice-f}}(ref, m)$ using subjective tests. Next, we show in Figure 10(b) the combined JND surface derived using the setting in (22), which was obtained by minimizing the upper bound of f in (20) and (21) and taken from the right-hand side of (17). Assumption 1 is validated because the two figures are very similar, with small differences at the corners that are reasonable in subjective tests with limited subjects.

To further illustrate the improvement of the combined strategy, we compare in Figure 10(c) the network latency when $P_{\text{notice}} = 50\%$. The graph shows that the combined strategy can maintain strong consistency while concealing delay effects, even



(a) Combined surface by tests (b) Combined surface by derivation (c) Latency at $P_{\text{notice}} = 50\%$

Fig. 10. Performance of the combined strategy for solving the reordering problem on targets with predictable responses. (a) JND surface of the combined strategy found by subjective tests. The x-axis shows the duration of the action, and the y-axis shows the one-way network latency. (b) JND surface of the combined strategy found by using (17). Its similarity to (a) verifies Assumption 1. (c) One-way latency in running the game, when players can correctly notice a difference with respect to the reference at $P_{\text{notice}} = 50\%$.

with much larger network latency. Alternatively, when using the same latency as the method with extended durations, the combined strategy can use the extra latency in delay buffers to smooth network jitters and provide better loss concealment, resulting in greater stability of the game in real time.

In short, our results clearly show the merit of the combined strategy for solving the reordering problem when compared to the previous methods. The combined strategy leads to lower P_{notice} on delay effects under a given latency while providing better loss concealment at the same level of P_{notice}.

4. SOLVING THE REORDERING AND THE BLANK-PERIOD PROBLEMS TOGETHER

In this section, we consider cases in which defenders' responses are not known a priori to attackers. As illustrated earlier in Figure 5, there will be a blank period in which an attacker does not know the outcome of her action until the defender's response has been received. To avoid inconsistent outcomes or rollbacks, the attacker will need to wait for the defender's response before proceeding.

The blank-period problem described here may occur in conjunction with the reordering problem when there are multiple attackers. In this section, we present methods for concealing such delay effects. We first show the solution to the blank-period problem with one attacker and one defender. We then combine this solution with that in Section 3 for solving the reordering and the blank-period problems for multiple attackers.

4.1. Necessary and Sufficient Condition for Concealing the Blank Period

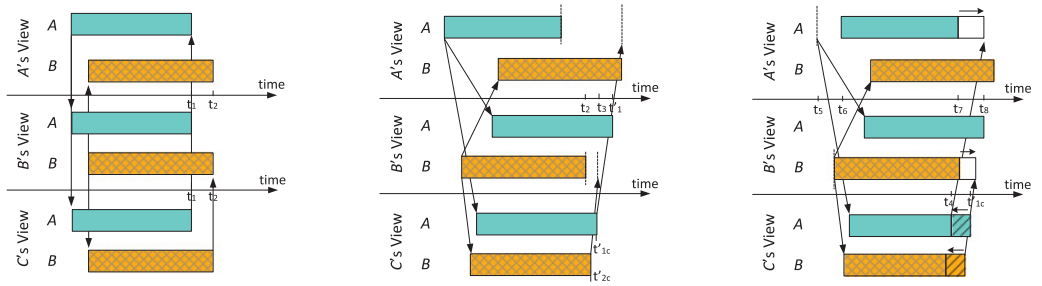
Referring to the blank-period problem in Section 2, the action in attacker i's view would terminate only after knowing defender j's action. Based on the three control strategies in Section 3.3, let $m_{i,j}^{\mathbf{LL}}$, $m_{i,j}^{\mathbf{LPF1}}$, and $m_{i,j}^{\mathbf{LPF2}}$ respectively be the extent that i's action is delayed, extended, and shortened in j's virtual space. Because a player may serve a dual role as both an attacker and defender, we add superscripts **ATK** and **DEF** to m to identify her role. The necessary and sufficient condition is stated as follows.

THEOREM 4.1. *The necessary and sufficient condition for attacker i to conceal the blank period when waiting for defender j's response is*

$$m_{i,i}^{\mathbf{LL},\mathbf{ATK}} + m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}} + m_{i,j}^{\mathbf{LPF2},\mathbf{DEF}} = d_{i,j} + d_{j,i}. \tag{23}$$

PROOF. Referring to Figure 5, i would have received j's response when

$$t_{i,i} \;=\; t_{i,j} + d_{j,i}. \tag{24}$$



(a) Reference with no latency (b) Case with network latency (c) Previous solution

Fig. 11. Illustration of the blank-period and the reordering problems under network latency when A and B shoot C, who is trying to defend against the attack. (a) Reference under no latency. (b) Case under latency. (c) Our previous work [Xu and Wah 2013a] for solving the blank-period problem may lead to reordering.

By expanding both sides, we have

$$t_{i,i} = r_{i,i} + m_{i,i}^{\mathbf{LL,ATK}} + l_{i,i} + m_{i,i}^{\mathbf{LPF1,ATK}} \text{ and } t_{i,j} + d_{j,i} = r_{i,i} + d_{i,j} + l_{i,i} - m_{i,j}^{\mathbf{LPF2,DEF}} + d_{j,i}.$$

Equation (23) follows after simplifying and rearranging the terms. □

Similar to Theorem 3.2, Equation (23) does not have any unpredictable terms involving r. Hence, it can be enforced by both players before their actions are carried out.

4.2. The Blank-Period and Reordering Problems with Multiple Attackers

The blank-period and reordering problems can happen together when there are multiple attackers instead of one attacker.

Figure 11(a) illustrates the reference case under no network latency. Here, the orders of completions are the same in all virtual spaces—that is, $t_2 > t_1$.

Figure 11(b) illustrates the case under network latency. The blank-period problem happens in B's view between t_2 and t_3 during which B does not know the result of the shot from A to C. It also happens in A's view in which the result of A's shot to C is not known until the message from C is received at A. On the other hand, the reordering problem happens in B's view in which the order of A's and B's completion times ($t'_1 > t_2$) is inconsistent with the reference ($t_1 < t_2$). This also happens in C's view ($t'_{1c} > t'_{2c}$). The example can be extended to a scenario with more than two attackers when the new attackers and their associated actions are added between B's and C's views.

In our previous work [Xu and Wah 2013a], we proposed a method for solving the blank-period problem. This is done by delaying the start of the attacker's action and by extending its duration while shortening the defender's action. These changes will allow the local action to complete only after receiving the defender's response.

Figure 11(c) illustrates the case in which our previous approach may inadvertently reorder the completions of actions when there are multiple attackers. Consider A's action. In her view, we delay the start of her action from t_5 to t_6 and extend its duration from t_7 to t_8. We also shorten the duration of her action in C's view from t'_{1c} to t_4. Similar steps can respectively be applied to B's action in B's and C's views. Reordering occurs in B's and C's views when compared to the reference in Figure 11(a). The figure can also be extended to a scenario with more than two attackers.

ALGORITHM 2: Strategy for Solving the Blank-Period and Reordering Problems

Require: Offline-measured JND surfaces based on local lag ($P'_{\text{notice}}(R, M^{\mathbf{LL}})$) and local perception filters ($P'_{\text{notice}}(R, M^{\mathbf{LPF1}})$ and $P'_{\text{notice}}(R, M^{\mathbf{LPF2}})$); JND surfaces for each player instantiated from the offline-measured JND surfaces; $N^{\mathbf{ATK}}$ attackers; $N^{\mathbf{DEF}}$ defenders;

Ensure: Local lag ($m_{i,i}^{\mathbf{LL,ATK}}$), local perception filters 1 and 2 ($m_{i,i}^{\mathbf{LPF1,ATK}}$, $m_{j,i}^{\mathbf{LPF2,ATK}}$), and local perception filter 2 ($m_{i,k}^{\mathbf{LPF2,DEF}}$, $m_{j,k}^{\mathbf{LPF2,DEF}}$);

```
Estimate the duration of actions using (7);
Estimate network latency;
for i = 1 to N^ATK do
    for j = 1 to N^ATK and j ≠ i do
        for k = 1 to N^DEF do
            In attacker i's view,
                Extend attacker i's action using (25) and (28) to find the optimal m_{i,i}^{LPF1,ATK};
                Shorten attacker j's action using (25) to find the optimal m_{j,i}^{LPF2,ATK};
            In defender k's view,
                Shorten attacker i's and j's actions by the optimal m_{i,k}^{LPF2,DEF} and m_{j,k}^{LPF2,DEF}
                (using (27) through (29)) to compensate for the difference in network latency;
            In attacker i's view,
                Delay the start of attackers i's and j's actions by the optimal m_{i,i}^{LL,ATK} (using (25)
                and (26) and (28) and (29)) to let the actions complete exactly when the
                corresponding defender's response is received.
        end for
    end for
end for
```

4.3. Proposed Strategy

In this section, we combine the strategy described earlier for solving the blank-period problem [Xu and Wah 2013a] and the strategy in Section 3.2 for solving the reordering problem to solve both problems together. Algorithm 2 shows the pseudocode. We prove its correctness in Section 4.3.1 and discuss its optimization in Section 4.3.2. Note that each player will instantiate her surfaces from the set of common offline-measured surfaces and may operate under a different operating point in the game.

4.3.1. Necessary and Sufficient Conditions for Solving the Blank-Period and Reordering Problems. The following theorem proves the correctness of Algorithm 2 under the general case when there are multiple attackers and multiple defenders.

THEOREM 4.2. *Let $D_{\max}$ be the maximum network latency between any two players. Algorithm 2 is correct and solves the reordering and the blank-period problems together for multiple attackers and defenders if and only the following conditions are satisfied for all pairs of attackers i and j and defender k:*

(*a*) *In each virtual space, the order of completion times is strongly consistent with the reference, both in attacker i's and j's views and defender k's view:*

$$m_{i,i}^{\mathbf{LL,ATK}} + m_{i,i}^{\mathbf{LPF1,ATK}} + m_{j,i}^{\mathbf{LPF2,ATK}} = D_{\max}, \tag{25}$$

$$m_{j,j}^{\mathbf{LL,ATK}} + m_{j,j}^{\mathbf{LPF1,ATK}} + m_{i,j}^{\mathbf{LPF2,ATK}} = D_{\max}, \tag{26}$$

$$m_{i,k}^{\mathbf{LPF2,DEF}} - m_{j,k}^{\mathbf{LPF2,DEF}} = d_{i,k} - d_{j,k}. \tag{27}$$

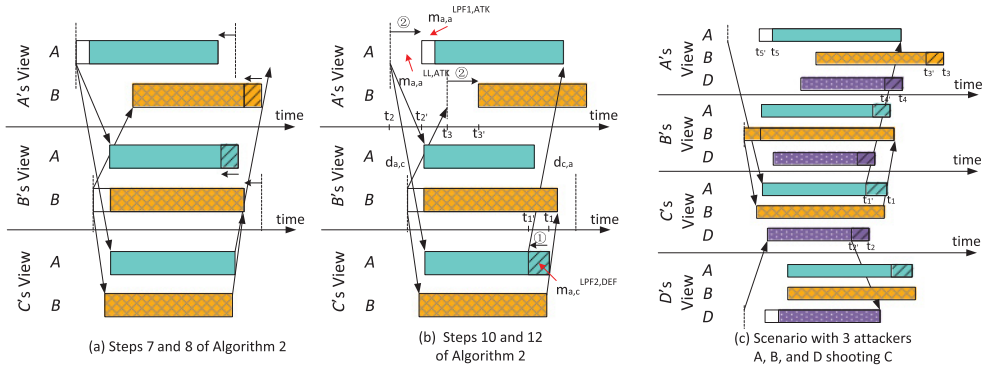


Fig. 12. Illustration of Algorithm 2 in solving the example in Figure 11 when A and B shoot C, who is trying to defend against the attack. (a) Steps 7 and 8. (b) Step 10 (① in the figure) and step 12 (② in the figure). (c) Extension to a scenario with three attackers when a new attacker D joins the game.

(*b*) *For each attacker, her local action terminates after receiving defender k's response:*

$$m_{i,i}^{\mathbf{LL,ATK}} + m_{i,i}^{\mathbf{LPF1,ATK}} + m_{i,k}^{\mathbf{LPF2,DEF}} = d_{i,k} + d_{k,i}, \tag{28}$$

$$m_{j,j}^{\mathbf{LL,ATK}} + m_{j,j}^{\mathbf{LPF1,ATK}} + m_{j,k}^{\mathbf{LPF2,DEF}} = d_{j,k} + d_{k,j}. \tag{29}$$

Proof. We prove the two parts separately.

(a) *Solving the reordering problem.* There are two steps in proving this part.

First, strong consistency of completion times in all attackers' views is ensured because (25) and (26) are the same as (12), which has been proved in Theorem 3.5.

Second, we prove the strong consistency of completion time in defender k's view. For any actions from i and j in defender k's view, we shorten them by $m_{i,k}^{\mathbf{LPF2}}$ and $m_{j,k}^{\mathbf{LPF2}}$, respectively. By (27), the completion order of these actions are enforced as follows:

$$t_{i,k} - t_{j,k} = t_{i,i} + d_{i,k} - m_{i,k}^{\mathbf{LPF2,DEF}} - \left(t_{j,j} + d_{j,k} - m_{j,k}^{\mathbf{LPF2,DEF}}\right) = t_{i,i} - t_{j,j}. \tag{30}$$

(b) *Solving the blank-period problem.* Equations (28) and (29) are exactly the necessary and sufficient condition proved in Theorem 4.1 for concealing the blank period.

The general case of multiple attackers and multiple defenders follows by combining the two parts and by considering any two attackers and any one defender. □

Figure 12(a) illustrates steps 7 and 8 of Algorithm 2 that apply (25) and (28) for Attacker A (respectively, (26) and (29) for B) to solve the reordering problem. Comparing it to Figure 9, it is clear that the modifications to the actions of both attackers are similar. It also illustrates the application of (28) and (29) to partially solve the blank-period problem.

Figure 12(b) illustrates step 10 (①) and step 12 (②) of Algorithm 2.

In step 10, we shorten the duration of A's action in defender C's view so that the difference between A's and B's adjustments in C satisfies (27). As shown in C's view, A's action that originally completes at t_1 now completes earlier at t_1'. In contrast, B's action need not be shortened in C's view because $m_{B,C}^{\mathbf{LPF2,DEF}} = 0$ already satisfies (29).

In step 12, we apply (28) and (29) to delay the start of actions in attacker A's and B's views to let their actions complete when the response from defender C is received. As shown in A's view, to make A's action complete when C's response is received, we delay its start from t_2 to t_2'. To maintain the correct order of the completions (which has been ensured by steps 7 and 8), we further delay the start of B's action in A's view

from t_3 to t_3' so that $t_3' - t_3 = t_2' - t_2 = m_{A,A}^{\mathbf{LL},\mathbf{ATK}}$. This is basically the application of (25) and (26) in solving the reordering problem, as Figure 12(b) is based on Figure 12(a). In contrast, as the latency between B and C is short, and extending B's duration by step 7 is sufficient to cover the blank period, C's response already arrives at B on time, and the start of B's action needs not be delayed in B's view.

To show that the solution is correct, we compare the reference order in Figure 11(a) to the order in Figure 12(b). It is clear that strong consistency is maintained. Further, C's responses arrive at A and B exactly when their actions complete.

Figure 12(c) shows another example of applying Algorithm 2 on three attackers A, B, and D shooting defender C. The corresponding reference order under no latency is similar to the order in Figure 11(a), but with D's action starting and completing slightly earlier than A's. For simplicity, we only show the messages between the attackers and the defender but not among the attackers.

By comparing Figures 12(c) and 12(a), we need to schedule D's action in A's and B's views, as well as A's and B's actions in D's view. These can be done by steps 7 and 8 of Algorithm 2. By comparing Figures 12(c) and 12(b), we apply step 10 to ensure that strong consistency is satisfied in C's view (Figure 11(a)). Note that D's action completes slightly earlier than A's. Finally, to solve the blank-period problem, Step 12 schedules the completion times of all attackers to the point when C's response is received.

4.3.2. Optimizing the Proposed Strategy. In this section, we present the optimization for achieving the minimum $P_{\text{notice-f}}^{\mathbf{COMB}}$ on delay effects when using Algorithm 2. The optimization has several differences with respect to that in (20) and (21). First, we consider $N^{\mathbf{ATK}}$ attackers and $N^{\mathbf{DEF}}$ defenders. Second, we address the general case in which players may have different ref. Third, we shorten the durations of all defenders' actions to solve the reordering problem in their views while delaying the starting times of all attackers' actions to solve the blank-period problem in the attackers' views.

Since each player cannot see the game in another player's view, we could optimize the combined noticeability in each view separately. However, this may result in some views having highly noticeable delay effects, whereas others have less noticeable delay effects. To maintain fairness in a multiplayer game [Xu and Wah 2013a], we minimize the noticeability in all views simultaneously.

Using notation similar to that of (14) in Section 3.3, we aim to minimize the upper bound of the combined noticeability in attacker i's and j's views.

$$P_{\text{notice-f}}^{\mathbf{COMB}}(ref, m_{i,i}^{\mathbf{ATK}}) \le \max\{P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LL},\mathbf{ATK}}),\ P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}}), P'_{\text{notice}}(ref_j, m_{j,i}^{\mathbf{LPF2},\mathbf{ATK}})\}. \tag{31}$$

In defender k's view, we employ the **LPF2** strategy with no combined noticeability:

$$P_{\text{notice}}(ref, m_{x,k}^{\mathbf{DEF}}) = P'_{\text{notice}}(ref_x, m_{x,k}^{\mathbf{LPF2},\mathbf{DEF}}),\ \text{where } x = i, j. \tag{32}$$

The overall optimization is now stated as follows:

$$\begin{aligned}\overline{\mathcal{P}} &= \min \max_{\substack{1\le i,j\le N^{\mathbf{ATK}}, i\ne j,\\ 1\le k\le N^{\mathbf{DEF}}}} \max\left\{P_{\text{notice}}^{\mathbf{COMB}}(ref, m_{i,i}^{\mathbf{ATK}}), P_{\text{notice}}(ref, m_{i,k}^{\mathbf{DEF}}), P_{\text{notice}}(ref, m_{j,k}^{\mathbf{DEF}})\right\} \\ &= \min \max_{\substack{1\le i,j\le N^{\mathbf{ATK}}, i\ne j,\\ 1\le k\le N^{\mathbf{DEF}}}} \max\left\{P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LL},\mathbf{ATK}}),\ P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}}),\right. \\ &\quad \left. P'_{\text{notice}}(ref_j, m_{j,i}^{\mathbf{LPF2},\mathbf{ATK}}),\ P'_{\text{notice}}(ref_i, m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}}),\ P'_{\text{notice}}(ref_j, m_{j,k}^{\mathbf{LPF2},\mathbf{DEF}})\right\},\end{aligned} \tag{33}$$

$$\text{subject to} \quad m_{i,i}^{\mathbf{LL},\mathbf{ATK}} + m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}} + m_{j,i}^{\mathbf{LPF2},\mathbf{ATK}} = D_{\max} \tag{34}$$

$$m_{j,j}^{\mathbf{LL},\mathbf{ATK}} + m_{j,j}^{\mathbf{LPF1},\mathbf{ATK}} + m_{i,j}^{\mathbf{LPF2},\mathbf{ATK}} = D_{\max} \tag{35}$$

$$m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}} - m_{j,k}^{\mathbf{LPF2},\mathbf{DEF}} = d_{i,k} - d_{j,k} \tag{36}$$

$$m_{i,i}^{\mathbf{LL},\mathbf{ATK}} + m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}} + m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}} = d_{i,k} + d_{k,i} \tag{37}$$

$$m_{j,j}^{\mathbf{LL},\mathbf{ATK}} + m_{j,j}^{\mathbf{LPF1},\mathbf{ATK}} + m_{j,k}^{\mathbf{LPF2},\mathbf{DEF}} = d_{j,k} + d_{k,j}, \tag{38}$$

where (34) through (38) are the same as (25) through (29) in Theorem 4.2.

Comparing the objective function in (33) with that in (20), we have one more max operator for aggregating the P'_{notice} of modifications to all attackers' actions in every view. This is used to maintain fairness (as mentioned earlier) across all players by bounding the maximum P_{notice}. We allow different ref's in (33), as actions of different durations can appear together. There are also two new terms, $P'_{\text{notice}}(ref_i, m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}})$ and $P'_{\text{notice}}(ref_j, m_{j,k}^{\mathbf{LPF2},\mathbf{DEF}})$, which represent the shortened durations of actions by attackers i and j in defender k's view. They are used to limit the delay effects in the defenders' views and to avoid some attackers' actions being too fast, making them difficult for defenders to guard against.

Comparing (34) through (38) to (21), there are new constraints for addressing the reordering and the blank-period problems.

The preceding optimization is complex to solve in a closed form. The many variables and constraints, as well as different ref's, make it hard to find an optimal solution at runtime. Fortunately, several observations can help simplify the problem.

First, from (36), once $m_{i',k}^{\mathbf{LPF2},\mathbf{DEF}}$ is determined, any $m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}}$, $1 \le i \le N^{\mathbf{ATK}}$, $i \ne i'$, can be determined uniquely.

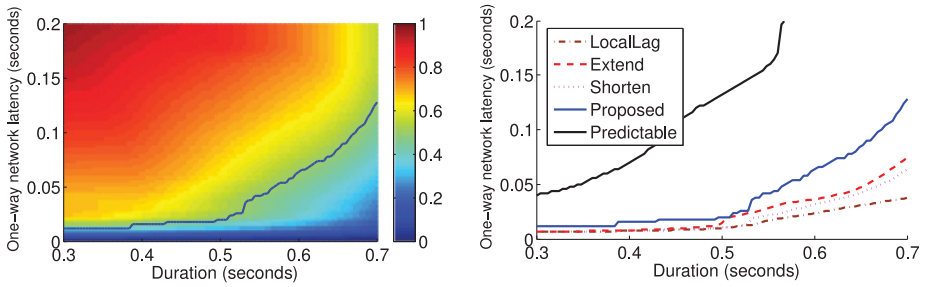
Second, by combining (34) and (37), we can directly get $m_{j,i}^{\mathbf{LPF2},\mathbf{ATK}}$ once $m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}}$ is found in the last step. This also applies when we combine (35) and (38).

Third, from (37), we know that $m_{i,i}^{\mathbf{LL},\mathbf{ATK}} + m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}}$ is determined once $m_{i,k}^{\mathbf{LPF2},\mathbf{DEF}}$ is found. As shown in the proof of Theorem 3.10, the optimal solution is attained when $P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LL},\mathbf{ATK}}) = P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}})$, which allows the optimal $m_{i,i}^{\mathbf{LL},\mathbf{ATK}}$ and $m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}}$ to be found directly. This also applies to (38).

With these observations, the optimal solution to (33) through (38) can be found by enumerating the value of a single control variable $m_{i',k}^{\mathbf{LPF2},\mathbf{DEF}}$. The computational complexity is thus $O((N^{\mathbf{ATK}})^2 N^{\mathbf{DEF}} k^2)$, where k is the discretization level of $m_{i',k}^{\mathbf{LPF2},\mathbf{DEF}}$. We use k^2 instead of k because one more loop is needed for finding $m_{i,i}^{\mathbf{LL},\mathbf{ATK}}$ and $m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}}$ that satisfy $P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LL},\mathbf{ATK}}) = P'_{\text{notice}}(ref_i, m_{i,i}^{\mathbf{LPF1},\mathbf{ATK}})$.

As the overall complexity is low, we can search for the optimal control values at runtime. This can be finished within 5ms by a computer with an Intel Core 2 Duo E8300 3GHz CPU. The size of each JND surface is 40KB, which is sufficiently small.

4.3.3. Experimental Evaluations on BZFlag. Similar to the results in Section 3.4, we present in this section the evaluation of Algorithm 2 on BZFlag [Myers et al. 2012]. To simplify the illustration, we use two attackers and one defender in the following experiments. We assume identical ref's for all players, which allows the common ref to be shown in the x-axis of a single JND surface. We further assume that the players



(a) JND surface of combined strategy

(b) Latency at $P_{\text{notice}} = 50\%$

Fig. 13. Performance of the combined strategy with two attackers and one defender having identical reference durations for solving the reordering and the blank-period problems with an unpredictable defender's actions. (a) JND surface of the combined strategy in running BZFlag. The x-axis shows the action duration, and the y-axis shows the one-way latency. (b) One-way latency when players can notice a change with respect to the reference at $P_{\text{notice}} = 50\%$. The degradations between the scenario with a predicted defender's actions (labeled "Predictable" and shown in Figure 10(a)) and the current result (labeled "Proposed") are caused by the time to handle the blank-period problem.

to have the same instantiated JND surfaces in Algorithm 1 when using the strategies in isolation based on the local-lag method and the local perception filters.

Figure 13 shows the resulting JND surface after applying Algorithm 2 as well as the tolerable one-way latencies with $P_{\text{notice}} = 50\%$. The results show that the tolerable one-way latencies are much smaller than the corresponding latencies in Figure 10(c). These degradations are also reflected in the higher P_{notice} values in Figure 13(a) when compared to those in Figure 10(a). For example, when the reference duration is 0.5s, the strategy based on extended durations allows more than 40ms tolerable one-way latency in Figure 10(c), but it only allows around 25ms here. The degradations between the results of Algorithm 1 (labeled "Predictable") and those of Algorithm 2 (labeled "Proposed") in Figure 13(a) are attributed to the additional time to handle the blank-period problem. Algorithm 2, however, leads to much better tolerable latencies when compared to the other strategies. This means that the game can run in a network with higher latency while providing comparable playing experience.

Note that Figure 13(a) is similar to Figure 8(b). This similarity can be explained by (34) through (37)—that is, $m_{i,k}^{\mathbf{LPF2,DEF}} - m_{j,i}^{\mathbf{LPF2,ATK}} = d_{i,k} + d_{k,i} - D_{\max}$. As we assume that $d_{i,k} = d_{k,i} = D_{\max}$, the optimal solution appears at $m_{j,i}^{\mathbf{LPF2,ATK}} = 0$ and $m_{i,k}^{\mathbf{LPF2,DEF}} = D_{\max}$ when we minimize the delay effect. In other words, the strategy with shortened durations at the defender's view causes the dominant delay effect in the combined strategy.

In summary, Algorithm 2 leads to less noticeable delay effects while maintaining strong consistency and concealing blank periods. Further, the controls found allow the system to operate with the same P_{notice} but with higher latency.

5. CONCLUSION

In this article, we proposed a novel method for ensuring strong consistency on the completion times of actions, while minimizing noticeable delay effects due to network latencies, in fast MMOs running on IP networks. We proposed a new approach for minimizing delay effects on user perception. The success of our approach is based on optimizing multiple controls together, each causing less noticeable delay effects than when applying the corresponding control in isolation. Finally, we evaluated our approach by conducting subjective tests using the popular open-source online shooting game BZFlag and have shown significant performance improvements over previous

strategies. Due to limitation in space, we refer interested readers to our previous work [Xu and Wah 2013a] for the application of the proposed idea in fighting games.

REFERENCES

T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. 2004. The effects of loss and latency on user performance in unreal tournament 2003. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*. 144–151.

A. R. Bharambe, J. Pang, and S. Seshan. 2006. Colyseus: A distributed architecture for online multiplayer games. In *Proceedings of the 3rd Conference on Networked Systems Design and Implementation*. 12.

L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan. 2007. Hydra: A massively-multiplayer peer-to-peer architecture for the game developer. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*. ACM, New York, NY, 37–42.

P. Chen and M. El Zarki. 2011. Perceptual view inconsistency: An objective evaluation framework for online game quality of experience (QoE). In *Proceedings of the 10th ACM SIGCOMM Workshop on Network and Systems Support for Games*. ACM, New York, NY, Article No. 2.

C.-H. Chou and Y.-C. Li. 1995. A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile. *IEEE Transactions on Circuits and Systems for Video Technology* 5, 6, 467–476.

E. N. Dzhafarov and H. Colonius. 1999. Fechnerian metrics in unidimensional and multidimensional stimulus spaces. *Psychonomic Bulletin and Review* 6, 2, 239–268.

S. R. Gulliver and G. Ghinea. 2006. Defining user perception of distributed multimedia quality. *ACM Transactions on Multimedia Computing, Communications, and Applications* 2, 4, 241–257.

N. Hariri, B. Hariri, and S. Shirmohammadi. 2011. A distributed measurement scheme for Internet latency estimation. *IEEE Transactions on Instrumentation and Measurement* 60, 5, 1594–1603.

Z. Huang, K. Nahrstedt, and R. Steinmetz. 2013. Evolution of temporal multimedia synchronization principles: A historical viewpoint. *ACM Transactions on Multimedia Computing, Communications, and Applications* 9, 1, 34.

Human Benchmark. 2016. Reaction Time Test. Retrieved November 6, 2016, from http://www.humanbenchmark.com/tests/reactiontime.

N. Jayant. 1992. Signal compression: Technology targets and research directions. *IEEE Journal on Selected Areas in Communications* 10, 5, 796–818.

F. W. B. Li, R. W. H. Lau, D. Kilis, and L. W. F. Li. 2011. Game-on-demand: An online game engine based on geometry streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* 7, 3, 19.

Y.-J. Lin, K. Guo, and S. Paul. 2002. Sync-MS: Synchronized messaging service for real-time multi-player distributed games. In *Proceedings of the 10th International Conference on Network Protocols*. IEEE, Los Alamitos, CA, 155–164.

L. Ma, K. N. Ngan, F. Zhang, and S. Li. 2011. Adaptive block-size transform based just-noticeable difference model for images/videos. *Signal Processing: Image Communication* 26, 3, 162–174.

D. Marshall, S. McLoone, and T. Ward. 2010. Optimizing consistency by maximizing bandwidth usage in distributed interactive applications. *ACM Transactions on Multimedia Computing, Communications, and Applications* 6, 4, 30.

M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. 2004. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia* 6, 1, 47–57.

J. Myers, T. Riker, F. Thilo, D. Trowbridge, S. Morrison, A. Tupone, and D. Remenak. 2012. BZFlag 2.4.2. Retrieved November 6, 2016, from http://bzflag.org/.

K. Raaen and T.-M. Grønli. 2014. Latency thresholds for usability in games: A survey. In *Proceedings of the Norsk Informatikkonferanse*. 1–12.

C. Savery. 2014. *Consistency Maintenance in Networked Games*. Ph.D. Dissertation. Queen's University, Kingston, Canada.

C. Savery and N. Graham. 2014. Reducing the negative effects of inconsistencies in networked games. In *Proceedings of the 1st ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play*. 237–246.

W. Shi, J.-P. Corriveau, and J. Agar. 2014. Dead reckoning using play patterns in a simple 2D multiplayer online game. *International Journal of Computer Games Technology* 2014, Article No. 5.

J. Smed, H. Niinisalo, and H. Hakonen. 2005. Realizing the bullet time effect in multiplayer games with local perception filters. *Computer Networks* 49, 1, 27–37.

D. Stuckel and C. Gutwin. 2008. The effects of local lag on tightly-coupled interaction in distributed groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. 447–456.

S. L. Teal and A. I. Rudnicky. 1992. A performance model of system delay and user strategy selection. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 295–305.

H. von Helmholtz. 1891. Versuch einer erweiterten Anwendung des Fechnerschen Gesetzes im farbensystem. *Z. Psychol. Physiol. Sinnesorg* 2, 1–30.

J. Xu and B. Wah. 2013a. Concealing network delays in delay-sensitive online interactive games based on just-noticeable differences. In *Proceedings of the International Conference on Multimedia and Expo*. IEEE, Los Alamitos, CA, 1–6.

J. Xu and B. W. Wah. 2013b. Exploiting just-noticeable difference of delays for improving quality of experience in video conferencing. In *Proceedings of the Multimedia Systems Conference* ACM, New York, NY, 238–248.

J. Xu and B. W. Wah. 2015. Optimizing the perceptual quality of real-time multi-metric multimedia applications using JND profiles. *IEEE Multimedia Magazine* 22, 4, 14–28.

J. Xu and B. W. Wah. 2016. Optimality of the greedy algorithm for generating just-noticeable-difference surfaces. *IEEE Transactions on Multimedia* 18, 7, 1330–1337.

A. Yahyavi and B. Kemme. 2013. Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Computing Surveys* 46, 1, 9.

X. K. Yang, W. S. Ling, Z. K. Lu, E. P. Ong, and S. S. Yao. 2005. Just noticeable distortion model and its applications in video coding. *Signal Processing: Image Communication* 20, 7, 662–680.

K. Zhang, B. Kemme, and A. Denault. 2008. Persistence in massively multiplayer online games. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*. 53–58.

S. Zhou, W. Cai, S. J. Turner, B.-S. Lee, and J. Wei. 2007. Critical causal order of events in distributed virtual environments. *ACM Transactions on Multimedia Computing, Communications, and Applications* 3, 3, Article 15.

Received July 2016; accepted September 2016