REHE INFORMATIK
9/99

Consistency in Continuous Distributed Interactive Media

Martin Mauve
Universität Mannheim
Praktische Informatik IV
L15, 16
D-68131 Mannheim

Consistency in Continuous Distributed Interactive Media

Martin Mauve
mauve@informatik.uni-mannheim.de
Praktische Informatik IV, University of Mannheim, Germany

ABSTRACT

In this paper we investigate how consistency can be ensured for continuous distributed interactive media, i.e. distributed media which change their state in reaction to user initiated operations as well as because of the passing of time. Existing approaches to reach consistency in discrete distributed interactive media are briefly outlined and it is shown that these fail in the continuous domain. In order to allow a thorough discussion of the problem, a formal definition of the term consistency in the continuous domain is given. Based on this definition we show that an important trade off relationship exists between the responsiveness of the medium and the appearance of short term inconsistencies. Currently this trade off is not taken into consideration for consistency in the continuous domain, thereby severely limiting the consistency related fidelity for a large number of applications. We show that for those applications the fidelity can be significantly raised by voluntarily decreasing the responsiveness of the medium. This concept is called local lag and it enables the distribution of continuous interactive media which are more vulnerable to short term inconsistencies than e.g. battlefield simulations. We prove that the concept of local lag is valid by describing how local lag was successfully used to ensure consistency in a 3D telecooperation application.

Keywords: Consistency, Continuous Distributed Interactive Media

1. INTRODUCTION

During the past decade the problem of consistency in synchronous CSCW applications (e.g. shared text editors) has been the focus of attention for many research groups. Most work in this area has been done on the consistency of *discrete distributed interactive media* [11], i.e. distributed media which change their state only in response to user initiated operations. In the last three to four years, however, a broad variety of applications has evolved which employ *continuous distributed interactive media*, e.g. networked computer-games, multi user virtual reality, distributed simulations for training and education, as well as CSCW applications for jointly working with active objects. In contrast to discrete media, a continuous medium can change its state not only in response to user initiated operations, but also because of the passing of time. For this class of media the issue of consistency is still unexplored to a large extend. Only in the area of distributed virtual environments (DVEs) [9] [10][3] have significant efforts been made to tackle the problem of consistency in continuous distributed interactive media.

As we shall show in this paper, the approaches for discrete media are not usable in the continuous domain. Moreover, the algorithms used for DVEs focus on optimizing responsiveness at the cost of frequent short term inconsistencies. While this might be reasonable for e.g. a battlefield scenario, it provides only mediocre results for other continuous media. It is therefore necessary to investigate the issue of consistency in the continuous domain in a more general way than this has been done before.

In order to get a broader insight into consistency for the continuous domain, it is necessary to establish a common view on what consistency comprises for continuous media. Also, restrictions and trade off relationships need to be identified. Only after this has been done is it possible to evaluate the approach currently used by most DVEs and propose improvements. An essential part of this work is therefore dedicated to the formal specification of a consistency criterion for the continuous domain.

With this definition in place, we are then able to identify and to evaluate the trade off between responsiveness and short term inconsistencies. This leads to the concept of local lag - deliberately decreasing responsiveness to lower the number of short term inconsistencies. As a proof that local lag can improve the consistency related fidelity of continuous distributed interactive media, we have successfully applied the concept to an application which is able to share arbitrary active VRML (Virtual

Reality Modeling Language) objects between a group of users. Throughout this work we use the terms continuous domain and continuous media as a shortcut for continuous distributed interactive media, while the terms discrete domain and discrete media are used as synonyms for discrete distributed interactive media.

In Section Two of this paper we discuss the consistency model for discrete media and show why approaches which implement this consistency model fail in the continuous domain. The third section is dedicated to the formal definition of the term consistency for continuous media. It also examines the trade off between responsiveness and short term inconsistencies. With theses tools in place, the commonly used dead reckoning approach to achieve consistency in DVEs is evaluated in Section Four. A concept to exploit the trade off between responsiveness and short term inconsistencies is proposed in the fifth section - it is called local lag. Section Six briefly outlines a 3D telecooperation application which makes use of local lag to achieve consistency. A summary and an outlook to future work is given in Section Seven.

2. CONSISTENCY IN DISCRETE DISTRIBUTED INTERACTIVE MEDIA

A discrete medium is a medium which changes its state solely because of (user initiated) operations. Generally speaking, consistency for discrete distributed interactive media is about finding a 'correct' sequence of all operations issued by the participants of a session and making sure that, at all participating sites, the state of the medium looks as if all operations had been issued in the order of that sequence. To be more precise we use the terms causal ordering relation and dependent operation from Sun/Ellis [11], which are derived from Lamports work on virtual clocks [5]:

> "*Definition 1: Causal ordering Relation "→". Given two operations O_a and O_b generated at sites i and j, then $O_a \rightarrow O_b$, iff: (1) i=j and the generation of O_a happened before the generation of O_b, or (2) i≠j and the execution of O_a at site j happened before the generation of O_b, or (3) there exists an operation O_x, such that $O_a \rightarrow O_x$ and $O_x \rightarrow O_b$.*
>
> *Definition 2: Dependent and independent operations. Given any two operations O_a and O_b. (1) O_b is dependent on O_a iff $O_a \rightarrow O_b$. (2) O_a and O_b are independent (or concurrent), expressed as $O_a || O_b$, iff neither $O_a \rightarrow O_b$, nor $O_b \rightarrow O_a$.*"

In order to illustrate these terms consider the session depicted in Figure 1: in this session the three operations O_1, O_2 and O_3 take place. The causal ordering relation for this example is $O_2 \rightarrow O_3$, since O_2 is executed at site 2 before O_3 is generated. An example for the semantics of O_2 could be that the user at site 2 marks a certain part of a text in a group text editor. In reaction to this, the user at site 3 chooses to delete a different section as operation O_3 (maybe because the section marked by the user at site 2 contained all the information of the deleted section). From the ordering relation it can be derived that O_3 depends on O_2, while O_1 and O_2 are independent of each other.

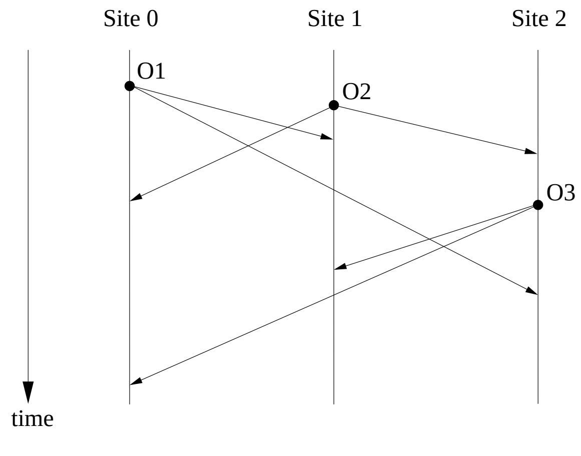

Figure 1: Example of a session with a discrete distributed interactive medium

Based on the definition of the causal ordering relation, Ellis and Gibbs give the following consistency correctness criteria in the context of the GROVE (GRoup Outline Viewing Editor) system [1]:

(1) **Convergence property**. After all operations have been executed the state of the discrete interactive medium is identical at all sites.

(2) **Precedence property**. For any two operations O_a and O_b, if $O_a \rightarrow O_b$, then at each site O_a will be executed before O_b.

The first criterion ensures that all users will have the same state of the interactive medium after all operations are resolved, it is the base consistency criterion. A violation of this criterion would result in different states of the distributed medium and therefore render it useless.

The second criterion makes sure that dependent operations are executed in order. If (2) where not required, the user at site 0 might see the result of dependent action O_3 before the result of the action O_2 is visible. This would be confusing for the user, since the causality of the operations is not preserved.

In order to ensure that the consistency correctness criteria are not violated there exists a number of different approaches, ranging from a strict single user floor control policy to enforce the same order of events at all sites, to sophisticated algorithms such as operational transformation [1] which basically 'repair' ordering problems as they occur. Since at least the convergence property is also desirable for distributed continuous interactive media, the question arises, whether the approaches which ensure this criterion for discrete media could be reused in the continuous domain. Unfortunately the answer to this question is: 'no'.

The reason why the approaches for discrete media fail when they are applied to continuous media is that consistency in continuous distributed interactive media is not only about finding a correct sequence of operations and making sure that at each site the result of all operations looks as if the operations had been executed in that sequence. In addition it requires that the result of all operations looks as if the operations had been executed at the *correct point in time*. The algorithms for establishing consistency in the discrete domain can therefore be regarded as inappropriate for the continuous domain.

To illustrate this problem let us examine a very simple example. Given is a session involving a discrete distributed interactive medium. This session is attended by two users U_a and U_b. Now let U_a perform an operation on the medium. This operation will be executed first on U_a's copy of the interactive medium and some time later (because of the network transmission delay) on U_b's copy of the medium. Since for a discrete interactive medium a single operation can not result in inconsistencies no consistency algorithms from the discrete domain will take any special actions in this example.

Now let us transfer this example to the continuous domain: in a distributed simulation, attended by two spacially separated users U_a and U_b, a train is approaching a switch. The switch can be operated by any user participating in the session. Imagine, that just before the train arrives at the junction, U_a operates the switch. In the copy of U_a's simulation the operation takes place immediately. However, the information about U_a operating the switch will arrive at the copy U_b at a later point in time. Applying the operation at this point in time to U_b's copy could lead to an inconsistent state because the train might have already passed the switch in U_b's copy of the simulation. As explained above, methods for ensuring consistency in discrete media will take no actions to correct this problem. This reveals the core reason why the mechanisms for consistency used in the discrete domain are not sufficient for continuous media, namely because they neglect the problem of executing operations at the correct point in time.

3. CONSISTENCY IN CONTINUOUS DISTRIBUTED INTERACTIVE MEDIA

A continuous interactive medium is a medium which changes its state in response to (user initiated) operations as well as because of the passage of time. This definition implies that the medium has access to a physical clock which can be used to measure the progress of time. In the scope of this work we assume that the physical clocks of all participants are reasonably synchronized - with a deviation of less than about 50ms (achievable using e.g. NTP [8] or GPS clocks).

In continuous distributed interactive media an user initiated operation needs to be executed at a specific point in time, denoted by its timestamp. Similar to consistency in the discrete domain, consistency for continuous media is about finding a correct order of all operations. However, in addition it must be guaranteed that, at all sites, the state of the interactive medium after these operations have been executed is the same as if the operations had been executed in the correct order *at the time identified by their timestamps*.

In order to be able to discuss this in a more formal way we define a partial ordering on the user initiated operations as follows:

Definition 3: Partial physical time ordering relation "<". Given two operations O_a and O_b with timestamps T_a and T_b, then O_a is said to happen before O_b, expressed as $O_a<O_b$, iff $T_a<T_b$.

Definition 4: Any two operations O_a and O_b with timestamps T_a and T_b are said to be simultaneous, expressed as $O_a \approx O_b$, iff $T_a=T_b$.

Note, that this definition is based on physical clocks as opposed to logical clock based methods for discrete media. The partial physical time ordering relation can be extended to become a complete physical time ordering relation by using an arbitrary tiebreaker for simultaneous operations. An example for such a tiebreaker could be the IP address to break ties for simultaneous operations from two different participants in combination with a counter to break ties of the same participant.

Definition 5: Complete physical time ordering relation “<<“. Given two operations O_a and O_b with Timestamps T_a and T_b and tiebreakers B_a and B_b, then $O_a<<O_b$, iff (1) $T_a < T_b$, or (2) $T_a = T_b$ and $B_a<B_b$.

Now we are able to define the consistency criterion for continuous distributed interactive media as follows:

(3) **Consistency criterion**. A continuous interactive medium is consistent, iff after all operations have been executed at all sites, the state of the medium at all sites is identical to the state which would have been reached by executing all operations in the order given by the complete physical time ordering relation at the physical time denoted by the timestamps of the operations.

The consistency criterion includes the semantics of the precedence property from the discrete domain, provided that the following reasonable assumption holds true: it is impossible for a user to react to an operation O_a with timestamp T_a by issuing an operation O_b with a timestamp $T_b <= T_a$.

The consistency criterion for the continuous domain is illustrated in Figure 2. The physical clocks of all three sites are slightly out of synchronization, the dashed lines identify the points in time with the same reading of the physical clock at each site.The small filled circles indicate the time when an action is executed. In the example the following ordering of operations applies: O1<O2, O1<O2 and O2≈O3 as well as O1<<O2, O1<<O3 and O2<<O3 (if the site number is used as tiebreaker). Figure 2 shows a somewhat unrealistic situation where all operations are always known at all sites at the time denoted by their timestamp. This situation, however, is important, since the meaning of the consistency criterion is that at some time after the last site has received all operations (identified as x in the example) the medium will look as if the operations had been executed as depicted in Figure 2.

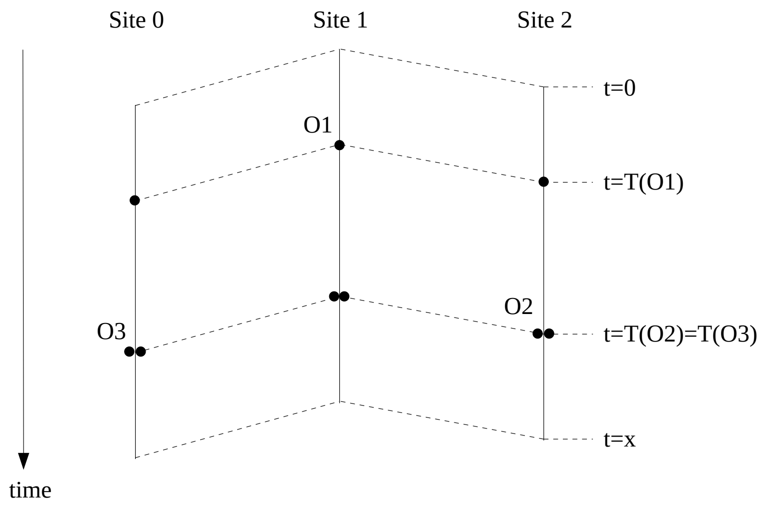

Figure 2: A consistent continuous distributed interactive medium

By guaranteeing that the state of the medium will be identical at all sites after all operations have been resolved, the consistency criterion is the most important prerequisite to make sure that a continuous distributed interactive medium is actually usable. However, even if the consistency criterion is ensured by a proper algorithm, it is still possible that consistency related, transient artifacts occur. In addition to the basic consistency criterion we therefore define two additional fidelity criteria:

(4) **Avoidance of short term inconsistencies**. If an operation with a timestamp T is not executed at a site i at the physical time T then this operation is said to have caused a short term inconsistency of the interactive medium at site i. Ideally, short term inconsistencies should be non-existent.

Short term inconsistencies occur when an operation issued at site j arrives at site i after the time denoted in the timestamp of the operation. Because we assume that there exists an algorithm to ensure the convergence property, a late arrival means that the state of the interactive medium at site i needs to be repaired. In the train example this would mean moving the train from the wrong position on one branch of the tracks to the right position on the other branch of the tracks.

Short term inconsistencies basically have two implications: (1) they usually result in visual artifacts (the train “jumps” from one position to a different one) and (2) a user might take actions which are based on an inconsistent state of the medium. An example for the latter is the following situation: Assume that in the train example U_b pulls the breaks of the train, after it has passed the switch, but before the operation from U_a has arrived. In this case U_b intends to stop the train because it is going in the wrong direction. After U_b has issued this operation the operation from U_a arrives (late). In order to satisfy the convergence property, the train will jump to the correct position. The net result of both operations is that the train will stop while it was headed in the right direction, which is a clear violation of the intention of the users. This effect is similar to the intention-violation problem in discrete media, as described in [1].

(5) **Low response time**. The response time of an interactive medium is the time between the physical time a user issues an operation and the timestamp of that operation. Ideally the response time should be zero.

If the response time exceeds a certain threshold, the users will notice that there exists a delay between issuing operations and the time when the operations are executed. This would result in an ‘unnatural’ behavior of the medium. Response time and responsiveness are referring to the same attribute of a medium.

While it would be desirable to be able to guarantee that no consistency related artifacts occur in a continuous distributed interactive medium, this is not possible. The reason for this is that the optimization of the response time and the avoidance of short term inconsistencies are conflicting goals. Part (a) of Figure 3 shows an example where the responsiveness is optimal (i.e. response time = 0). In this case the operations issued by the users take effect immediately. The points in time when O_1 and O_2 should take effect are indicated by the dotted line. As can be seen, only the user issuing the operation does not experience short term inconsistencies. Any other user will have a short term inconsistency for that operation because of the transmission delay for the remote operation. Part (b) of Figure 3 shows the same situation optimized for the short term inconsistency criterion. Here short term inconsistencies are avoided at the cost of increasing the response time for each operation to the maximum transmission delay between any two participants.

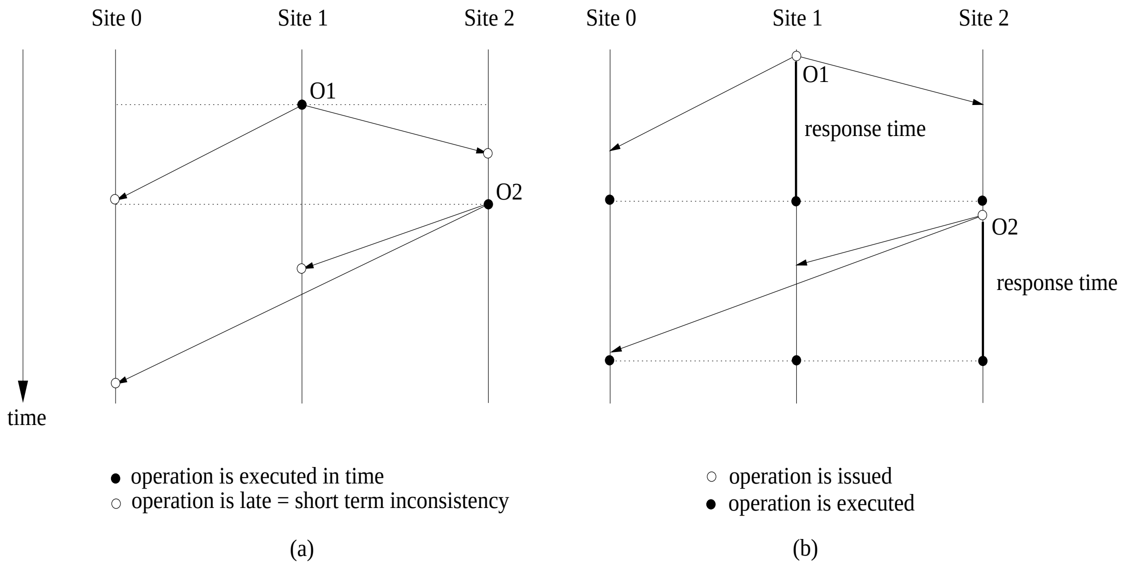

Figure 3: Short term inconsistency vs. response time

An observant reader might have noticed two peculiarities: (1) what if the time deviation between sender and receiver happens to compensate for the transmission delay and (2) in an environment where network packets might get lost, how exactly can we calculate the maximum transmission delay for an operation?

The answer to the first question is that the time deviation might indeed compensate for the network delay between a given sender and receiver pair. This makes it possible for the sender to have a response time equal to zero while the receiver does not experience short term inconsistencies. However, further reflection reveals that this works only in a broadcast environment, with a single sender which never changes. We consider this restriction too limiting for almost all applications involving continuous distributed interactive media.

The second observation arises from the problem that, in an environment where packet loss occurs, it is impossible to define an upper bound on the delay which an operation needs to arrive at a remote site. After all, the same packet might get lost over and over again as it is being retransmitted by the sender. This leads us to the conclusion that, while a reduction of short term inconsistencies is desirable, a guaranteed prevention is not possible.

With the definition of the convergence property in place and the trade off between the two fidelity criterions discussed we will now investigate a frequently used algorithm for achieving consistency in continuous distributed interactive media.

4. DEAD RECKONING

An approach which is commonly used to guarantee that the consistency criterion is satisfied in distributed virtual environments (e.g. multi-user virtual reality and battlefield simulations) is a combination of state prediction and state transmission. For this approach the application “knows” how the objects in the virtual environment should behave over time. Examples are a plane which will fly straight with constant speed or a projectile which will take a course dictated by the physical law of gravity. The ability to predict the behavior of objects is known as dead reckoning.

Each object for which dead reckoning is performed has a single controlling application, e. g. for a plane this will be the application of the pilot. The controlling application is responsible to inform peer applications when the state of the object deviates by more than a certain threshold from the predicted state. In the case of a deviation, the controlling application transmits the complete state of the affected object to all peers. Upon receiving this information, an applications discards the outdated state and uses the new state to perform dead reckoning for this object. In order to be able to use this approach over unreliable transport services, the controller of an object additionally transmits the state of the object in regular intervals as so called heart-beat messages.

User initiated operations are immediately applied to the local state of the affected object. The object will therefore be put into a state which differs significantly from the predicted state, requiring the controlling application to transmit the new state to its peer applications. Because of this behavior the dead reckoning approach optimizes the response time criterion while each action will result in a short term inconsistency at each remote site.

While this approach has a number of important advantages, such as robustness and scalability, the disadvantages of this approach limit its applicability. In detail these disadvantages are:

1. It has a maximum number of short term inconsistencies. This implies that visual artifacts and actions based on an inconsistent state are common.
2. It requires that the state of each object be transmitted for each operation. This could lead to a massive need of bandwidth, especially when the state of the object exceeds a couple of bytes.
3. Only one controller for each object is possible. This prevents collaborative actions such as two users moving a single object.

An example where this approach is used, is the well known battlefield simulation protocol called DIS (Distributed Interactive Simulations) [10] and general purpose DVEs such as DIVE [3]. In a typical battlefield simulation environment the state to the relevant objects is small (position and velocity) and relatively easy to predict, while each object will have only one controller (e.g. the pilot of a plane, the driver of a car, etc.). The visible artifacts can be reduced by using intelligent algorithms for repairing the state instead of immediately adopting the new state. For example, after a plane has changed its direction a remote application can calculate a alternate flight path which will eventually bring the plane back into the correct state (instead of jumping immediately to the correct position). Because of these attributes the state prediction and transmission approach is adequate for battlefield simulations and some distributed virtual environments.

However, this approach becomes less appropriate to use if an interactive medium has one or more of the following restrictions:

- the state of objects is complex,
- visual artifacts cannot be concealed,
- true collaboration is important, or
- actions based on an inconsistent state are critical.

For several applications it might therefore be better to have a closer look at the trade off between responsiveness and short term inconsistencies before simply maximizing responsiveness. In the following section we will discuss how to exploit the trade off to improve the consistency related fidelity of continuous distributed interactive media.

5. LOCAL LAG

The concept of local lag is simple: instead of immediately executing an operation issued by a local user, the operation is delayed for a certain amount of time before it is executed. To use the terms from section 3, this approach requires that the value of an operation’s timestamp is greater than the point in time when the operation is issued by a user. The delay which is introduced for the local user in this way is called local lag. As depicted in Figure 3 (b), if the local lag is sufficiently high, then it can reduce the number of short term inconsistencies.

5.1. Determining a Value for Local Lag

Clearly, it would not be desirable to move from one extreme where the response time is zero but short inconsistencies are very frequent to the opposite extreme, where almost no short term inconsistencies occur but the response time is unacceptably high. Therefore it is important to consider both consistency related fidelity criterions. For a given continuous distributed interactive medium we propose to do this in three steps: (1) determine a minimum for the local lag which is needed to prevent a significant amount of short term inconsistencies, then (2) determine the highest acceptable response time, and finally (3) choose a value for the local lag.

Step 1. Determining a minimal value for local lag.

In order for local lag to be useful it needs to significantly reduce the number of short term inconsistencies for all participants. Moreover, short term inconsistencies should be reduced for each sender/receiver pair. We therefore propose to use the maximum average network delay between any two participants as the minimum amount of local lag. Typical values (assuming an uncongested network) for the network induced delays are: less than 1ms for a LAN, 20ms for an european country, 40 ms for a continent, and 100 ms for a world wide session. Choosing the maximal average network-layer delay as minimal value for local lag implies that short term inconsistencies will occur only if packets get lost or the jitter becomes significant because of network congestions. Even though we consider this to be a reasonable lower bound for local lag, there might exist applications which could make use of even smaller local lag. The choice of this minimum is therefore application dependent.

If the clocks of the participants are not completely in synchronization, then the maximum offset between any two clocks need to be added to the result. This is necessary because the sender of an operation might be behind in time relative to the recipient of the operation. If this time deviation would not be accounted for, the time deviation plus the network induced transmission delay might be larger than the local lag, which would result in a short term inconsistency. This problem is shown in Figure 4, where site 0 is ahead in time of site 1. Even though the operation O1 from the user at site 1 has a local lag greater than the maximum network delay, a short term inconsistency occurs at site 0 because of the deviating clocks. Typical values for time deviation of computers using NTP or SNTP are 20-50 ms if the NTP server is reached via WAN and less than 10ms if the NTP server is part of the same LAN as the NTP client.

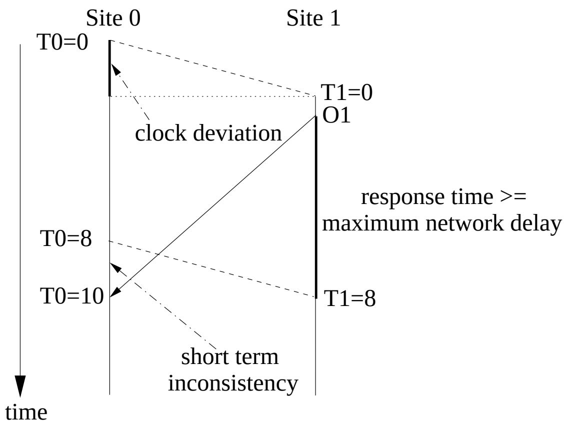

Figure 4: Problem with deviating clocks

Step 2. Determining the highest acceptable response time

The maximum local lag is dependent on how much response time a user can tolerate for a given interactive medium. In order to determine this value, it is necessary to conduct perceptional psychological experiments. Ideally the experiments will deliver two values: the first is the amount of local lag for which a user can not notice the delay. The second value is the amount of local lag which can be tolerated by the user, i.e. the delay can be noticed but is not disturbing. It seems to be reasonable to use the second value as the maximum amount of local lag for an interactive medium.

We are currently in the process of evaluating different operations for interactive media in order to determine these values. First experiments suggest that local lag of up to 50ms is not noticeable by the user independent of the operation and the medium. The value where local lag gets disturbing seems to depend heavily on the operation, ranging from 300-400ms for simple click operations to 100-150ms for drag operations.

Step 3. Choosing a value for the local lag.

The previous two steps can result in two main cases: either the minimum for local lag is smaller than or equal to the highest acceptable response time, or it is not. In the first case a value between the minimum for local lag and the highest acceptable response time can be chosen. This choice should be based on additional criterions such as the question whether additional time is required to receive forward error correction packets.

If the highest acceptable response time is lower than the minimal useful value for local lag, then a true trade off situation occurs. Given the values mentioned above this should be relatively rare, though it might happen for very demanding media and applications. In this case it is necessary to lower the fidelity of both criterions in a way which provides the best consistency based overall fidelity to the user. Most likely this will require another set of perceptional psychological experiments to find the amount of local lag which is considered to provide the highest fidelity to the user.

5.2. Repairing Short Term Inconsistencies

If the amount of local lag is chosen well, it eliminates a significant amount of inconsistencies. However, it does not completely prevent all inconsistencies, since operations might still arrive late or not at all (e.g. if the network is congested or a transport packet gets lost). It is therefore necessary to have a mechanism which will repair the state in these cases. In the following we will summarize three possible approaches to repair inconsistent states.

State prediction and Transmission.

An obvious approach is to combine local lag with the mechanism for state prediction and transmission. Instead of immediately executing a local operation on an object, the operation is delayed using the value for local lag: the operation gets a timestamp equal to the current time plus the value of the local lag. Immediately as the user issues the operation however, the new state for the object, at the time denoted by the timestamp of the operation, is calculated. This state includes the operation and bears the same timestamp as the operation. The calculated state is immediately transmitted to all participating sites.

When the physical time identified by the timestamp is reached, the site where the operation was issued will execute the operation. The remote sites which have received the new state for the object by then will start using the new state for dead reckoning without encountering any short term inconsistencies. Only those sites which receive the state late (because of an unexpected transmission delay or because they missed a packet completely) will experience short term inconsistencies while the state of the object is being repaired.

The combination of local lag with the state prediction and transmission approach retains two limitations from the original approach without local lag: only one controller per object is possible and the state of an object needs to be small (since the state is still transmitted for each operation). In the following approach we will allow the state of objects to be complex.

Requesting States

In order to allow objects to have a complex state of a size significantly larger than the 200 - 300 bytes used for states in the DIS protocol, it is important that the state of an object is only transmitted if it is really needed to repair a short term inconsistency. This aim can be reached by explicitly requesting the state in the case of a short term inconsistency, rather than transmitting the state for each operation.

An algorithm realizing this approach needs to solve two problems: (1) it must be possible to identify those participants with a correct state of the object and (2) it must be guaranteed that at least one participant can determine the correct state of the object. These problems can be solved by allowing only a single controller at a time for each object. Since only one user is able to issue operations for the object, that user will always have the correct state and it should be the task of the application of that user to reply to state requests for the object. The controller of an object can change, as long as it is guaranteed that all operations (and the repair of short term inconsistencies induced by these operations) of the old controller have been executed before a different participant becomes the new controller of the object.

The main limitation of this approach is that no true collaborative operations are possible, i.e. two users are not able to interact with the same object at the same time (unless some more sophisticated algorithm is used to determine who will reply to a state request). Also, repairing an inconsistent state now requires an additional round trip for the state request/reply. Therefore the requesting states approach is useful primarily for applications where simultaneous access to a single object is not required and the likeliness of an inconsistent state is rather low (high value for local lag).

Time Warp

This method is an adoption of the time warp algorithm used for optimistic parallel discrete event simulation [2] [4]. In time warp the state of each object does not need to be transmitted, even when a short term inexistence occurs. Instead local information is used to repair the state. Additionally time warp allows an arbitrary number of users to interact with the same object at the same time.

The fundamental idea of time warp is that each participant saves the state of the interactive medium at certain times. All operations up to a certain point in time in the past are kept in a log. When a short term inconsistency occurs, then the interactive medium is rolled back to the last saved state of the interactive medium before the operation should have taken place. Then the operation which caused the short term inconsistency is inserted into the log. After that the medium is played in fast forward mode, executing the operations from the log at appropriate times until the current time for the medium is reached and operation is resumed at normal pace. Only the end result of this operation should be visible to the user.

Choosing the points when to save the state is dependent of the application and the medium. If the state is saved to frequently, it might effect the visual quality of the presentation and use a large amount of memory. If the state is saved not frequently enough, a lot of operations have to be performed when a short term inconsistency occurs, extending the time the inconsistency is visible to the user.

The main drawback for this approach is that it requires a sophisticated application which is able to handle the process of time warping the state of a medium. In addition, if the state is complex, then the time warp approach might consume a large amount of memory for saving the states as well as placing a heavy load on the computer at the time a time warp has to be executed.

As a summary it can be said that choosing the amount of local lag as well as choosing a repair mechanism for short term inconsistency is heavily dependent on the application and the medium itself. In the following section we will show how these decisions have been made for a 3D telecooperation application.

6. USING LOCAL LAG IN A 3D TELECOOPERATION APPLICATION

We have tested the concept of local lag in the context of the 3D telecooperation application TeCo3D - a shared workspace for dynamic and interactive 3D models [6]. TeCo3D was developed to allow users to share collaboration-unaware VRML (Virtual Reality Modeling Language) models, i.e. models which have not been specifically developed to be used by more than one user at a time. With this functionality it is possible to include arbitrary VRML content, as generated by standard CAD or animation software, into teleconferencing sessions.

TeCo3D was developed by reusing the Java3D VRML loader [12] as 3D presentation and execution engine and employs a completely replicated distribution architecture with reliable multicast as means of communication. When a user imports a local VRML object, the VRML code is parsed and the parts which are responsible for user interactions are replaced with custom components turning the collaboration-unaware object into a collaboration-aware model. User initiated operations are captured by the custom components and are transmitted to all peer instance in the session, where they are injected into the local model. In order to provide remote users with the initial state of the imported objects and to repair short term inconsistencies we have enhanced the VRML loader by a method to get and set the state of arbitrary VRML objects [7].

The algorithm which provides consistency for TeCo3D had to take into consideration that the state of VRML objects can be complex, ranging from 200 bytes for a simple moving sphere to 50 000+ bytes for an interactive cartoon. Extracting and setting the state of objects is costly, both in terms of visual artifacts (the animation freezes for 100-200 ms) and computational overhead. It was therefore of critical importance to minimize the number of short term inconsistencies. The minimal useful amount of local lag (step 1) was therefore determined as 150 ms = 100 ms (world-wide conferences) + 50 ms (maximum time deviation).

First tests with increasing the response time from zero up to a value of 300 ms have shown that a delay of 100 ms is barely noticeable and 200 ms is still very acceptable for most interactive 3D objects. In addition users seem to be able to adapt to a constant delay rather fast, making a response time of up to 300 ms feasible for most types of interaction. The maximum acceptable response time (step 2) was therefore estimated to be between 200 and 300ms. With a minimum value for the local lag at 150ms and the maximum value between 200 and 300 ms we have chosen to set the local lag to 200ms, adding 50 ms to the minimum value for some additional protection against network jitter.

The next decision which had to be made was the choice of the repair mechanism. The state prediction approach was not taken into consideration because of the complexity of the state information for VRML objects. While it was tempting to use time warp, we did not want to burden our prototype with the complex mechanisms necessary for this approach. We therefore chose the requesting states method to repair short term inconsistencies. The implementation of both local lag and repair mechanism where straight forward and needed less than 500 lines of code.

First experiments with TeCo3D show that, as expected, no short term inconsistencies occur as long as all network packets arrive at their destination. When packet loss increases (and therefore retransmitted operations arrive late), short term inconsistencies start to appear. In our test sessions with loss rates between 0% and 5% these cases where sufficiently rare to not significantly affect the consistency related fidelity of the application.

In the future we will use forward error correction to protect user initiated operations from packets loss, effectively giving the transmission of operations a higher priority than the transmission of states for repairing short term inconsistencies. This should significantly increase the maximum tolerable packet loss rate.

7. CONCLUSION AND OUTLOOK

In this paper we investigated the problem of consistency in continuous distributed interactive media. It was shown that algorithms for consistency in the continuous domain differ significantly from those used for discrete media. In order to systematically approach the problem, a formal definition of the term consistency in the continuous domain was deduced from the special characteristics of the continuous media class. Based on this definition an important trade off relationship between responsiveness and the occurrence of short term inconsistencies was examined. We proposed to make use of the knowledge of this trade off relationship in order to increase the consistency related fidelity of the medium. This can be done by deliberately increasing the response time in order to decrease the number of short term inconsistencies, leading to the concept of local lag. The local lag approach for consistency in the continuous domain was proven to be usable by integrating it into a 3D telecooperation application.

Currently we focus our work on getting a deeper insight into how much local lag can be tolerated for different combinations of media and interaction types. In order to reach this goal a series of wharnehmungspsychologischer experiments is conducted at the University of Mannheim, on which we will report as soon as they have been properly evaluated.

Another area where future work is planned is a more thorough investigation of the repair mechanisms used in combination with local lag. It is especially tempting to employ the time warp approach for our 3D telecooperation application. However, before we are able to use time warp as an alternative to requesting states, we need to make sure that the complex algorithm of time warping and the overhead for regular state saves does not limit the fidelity of the application.

REFERENCES

[1] C. A. Ellis and S. J. Gibbs: “Concurrency control in groupware systems,” in: *Proceedings of the 1989 ACM SIGMOD Conference on Managements of Data*, Portland, OR, USA, 1989, pp. 399-407.

[2] R. M. Fujimoto: “Parallel Discrete Event Simulation,” *Communications of the ACM*, Volume 33, Number 10, pp. 30-53, 1990.

[3] E. Frécon and M. Stenius: “DIVE: A Scalable network architecture for distributed virutal environments,” Distributed Systems Engineering Journal, Volume 5, Number 3, pp. 91-100, 1998.

[4] D. R. Jefferson: “Virtual Time,” *ACM Transactions on Programming Languages and Systems*, Volume 7, Number 3, pp. 404-425, 1989.

[5] L. Lamport: “Time, Clocks, and the Ordering of Events in a Distributed System,” *Communications of the ACM*,

Volume 21, Number 7, pp. 558-565, 1978.

[6] M. Mauve: “TeCo3D - A 3D Telecooperation Application based on VRML and Java,” in: *Proceedings of Multimedia Computing and Networking MMCN/SPIE’99*, pp. 240-251, 1999.

[7] M. Mauve: “Transparent Access To And Encoding Of VRML State Information,” in: Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language VRML’99, pp. 29-38, 1999.

[8] D. L. Mills: “Network Time Protocol (Version 3) specification, implementation and analysis”, *DARPA Network Working Group Report RFC-1305*, University of Delaware, 1992.

[9] S. K. Singhal, “*Effective Remote Modeling in Large Scale Distributed Simulation and Visualization Environments*,” Ph.D Dissertation, Department of computerscience, Stanford University, 1996.

[10] S. Srinivasan: “Efficient Data Consistency in HLA/DIS++,” in: *Proceedings of the 1996 Winter Simulation Conference*, pp. 946-951, 1996.

[11] C. Sun and C. Ellis: “Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements,” in: *Proceedings of the ACM 1998 conference on Computer Supported Cooperative Work (CSCW’98)*, pp. 59-68, 1998.

[12] VRML Consortium: “The Java3D and VRML Working Group,” web page: www.vrml.org/WorkingGroups/vrml-java3d/.