# Time Management for Virtual Worlds based on Constrained Communication Model

Umar Farooq and John Glauert
School of Computing Sciences,
University of East Anglia,
Norwich NR4 7TJ,
United Kingdom.
(u.farooq, j.glauert)@uea.ac.uk

***Abstract*—Time Management (TM) is an integral part of the parallel and distributed systems that maintains the temporal order of events in a system. In this paper, we present a decentralised TM approach using a constrained communication model based on the inherent properties of virtual worlds. The proposed method uses a flat communication model and a region synchronises itself with a set of regions that share boundaries with it. It is evaluated with the help of a simple simulation model and compared with non-synchronised and decentralised scenarios. The simulation results show that it maintains local causality constraint and reduces communication of messages over the network. It is potentially more scalable and minimises longer delays and complexity compared with hierarchical strategies with multiple levels.**

## I. Introduction

The horizon of Parallel and Distributed Simulations (PADSs) has recently been extended for a number of different applications. However, Virtual Environments (VEs) among these applications have gained much attention from the end users. The reasons include the provision of impressive online content, a significant increase in speed, and a significant decrease in prices of high speed computers and network resources. Most recently, people are showing a great interest in social collaborative spaces called virtual worlds. The users have freedom in content development according to their desires on virtual land that is purchased using inland virtual currency [1][2][3]. Since, a PADS system is partitioned and executed with the help of a set of dedicated computers, it is a challenging task to maintain temporal order of events. This issue is called Time Management (TM) (alternatively synchronisation or consistency) and it is an integral part of a PADS system.

Parallel and distributed systems are fault tolerant, scalable and provide better interactive user experience but their performance is degraded when conservative approaches are used for time synchronisation [4]. The existing algorithms for PADS systems have shown great success for their target applications but they have major performance issues when used for virtual worlds. The target application of this research is a world like Second Life [2][3] that imitates the nature of social activities of the physical world. It gives more freedom than physical world and has an inherent property that a user or an activity is only influenced by the users or activities surrounding them. Based on this property, we put restrictions on communication and data distribution, and a system (serving a region) synchronises itself with the regions (defined as adjacent regions) that share physical boundaries with it. This work proposes an efficient decentralised TM approach based on the inherent properties of a virtual world. It is based on a conservative event driven approach and maintains a consistent view of dynamic hierarchical models based on our split strategies [5]. Control is fully decentralised and a server takes purely local decisions in consultation with adjacent servers. It maintains traditional constraints and guarantees that all events are processed in their temporal order. It greatly reduces communication overhead, complexity, and delays by avoiding a number of intermediate points compared with traditional mechanisms based on conservative approaches.

This paper is organized as follows. Section II gives the background of this research. The proposed decentralised TM scheme is presented in section III. Section IV provides the simulation results and illustrations. Conclusions and future work are presented in section V.

## II. Background and Motivation

According to Fujimoto [6], the basic goal of TM is to get exactly the same results as a sequential computer while executing a system with a set of servers. Each system maintains a list of events (both internal and external) and in each iteration removes the smallest timestamped event from the list and processes it in correct temporal order (the local causality constraint). Early synchronisation attempts proposed by Bryant [7], and Chandy et al. [8] are prone to deadlock. To resolve this issue, the concept of null messaging is used to advance the simulation clock of a process. It uses a value known as the Lookahead constraint that determines a safe range for event processing. The Lookahead value has a dramatic performance effect on the TM algorithm [9]. According to Pan et al. [10], asynchronous TM algorithms with small Lookahead values have a "time creep" problem. The main drawback of the null message algorithm is that it generates an excessive number of null messages thus introducing longer

delays. A typical synchronous algorithm maintains a value called the Lower Bound on Time Stamp (LBTS) for each process based on current time, Lookahead value and timestamp of the earliest event to solve this issue [4]. However, the time advance in a synchronous algorithms might be blocked by a process sending information with a low frequency. A number of traditional techniques such as message counters and flush queues are used to cope with transient messages [4][11].

The High Level Architecture (HLA) [12] is the current simulation standard and it was basically developed as a common interoperability architecture to integrate different classes of simulations. It is generalised and builds upon the results from DIS (Distributed Interactive Simulation) [13] and similar approaches such as Aggregate Level Simulation Protocol (ALSP) [6], and SIMNET (SIMulator NETworking) [14]. It serves well for the applications such as analysis, training, and test and evaluation. TM for an HLA federation is realised jointly by a Run Time Infrastructure (RTI) and participating federates. RTI is responsible for messages delivery and each federate makes an explicit time advance request to it. RTI grants permission if it can guarantee that no messages would be received in a federate's past. HLA has shown great success for military applications, but it does not provide load balancing and is poorly scalable. It is too complex, difficult to learn, adopt, and use [12]. The basic HLA standard does not support multi-level federations but implements all federates at one level as a single federation. It includes no means for inter-federation communication and conversion of a complex hierarchical structure into a flat one introduces several issues regarding data exchange, security, and re-usability [15]. Kim et al. [16], argue that these issues can be easily resolved if modular and hierarchical modelling methods are adopted.

To resolve these issues a number of attempts are made to provide interoperability between federations of a federation community [17][18][15], where a federation community is a group of federations and RTIs working together for a common goal. Myjak et al. [18] have proposed the following four approaches for inter-federation communication: Federation Gateway, Proxy/Bridge Federate, RTI Broker, and RTI-to-RTI Protocol. Cai et al. [19] have proposed a hybrid approach by integrating properties of both gateway and proxy mechanisms. It filters confidential information and improves security and interoperability. According to Cramp et al. [20], Federate Proxy (FP) is the simplest architecture but it might not be an ideal data filter for geographically distributed simulation as it is a single local process. Further, it handles a single federation with a flat structure. To minimise traversal of potentially large distances, Magee et al. [21] presented the concept of Distributed Federate Proxy (DFP). Cramp et al. [22] further extended their architecture for handling hierarchical federation communities. They proposed to distribute a FP into a number of Distributed FP Components (DFPCs) where each is assigned and processed local to a federation. These components are linked together with the help of tree nodes called SimNodes thus forming a hierarchical federation community. However, according to Cramp et al. [20], these architectures impose additional LBTS constraints on TM services for system components and time advance decisions are made by the ultimate root SimNode thus introducing unwanted longer delays.

Kim et al. [23] argue that the hierarchical methods described above are temporary solutions requiring additional interfaces that are not part of the RTI specification. It is therefore difficult to achieve interoperability among RTI systems developed by different vendors. According to them, it improves the overall performance of an RTI, if hierarchical federations are supported by an RTI itself. They have proposed a hierarchical extension of HLA to handle complex hierarchical models. The hierarchical structure is implemented with the help of two types of processes called a Federation Execution (FedEx) process and a federate process. A FedEx process acts as a federate in upper level federation and is defined as a representative federate. It has two possible implementations: FedEx Processes, and fully distributed federations. The number of FedEx processes in first might be a major problem when a hierarchy goes deeper. Since no limits are set on depth of a hierarchy, it might significantly increase delays because of dependencies between FedEx processes. The decentralised control in the second approach is difficult to manage [23][16].

The existing centralised and distributed TM methods perform well with smaller and medium scale simulation environments, but they have key performance issues when used for large scale virtual worlds. The hierarchical mechanisms are complex and introduce larger delays because of dependencies among their components for achieving a consistent environment. Similarly, decentralised peer-to-peer schemes incur an excessive number of messages over the network. We believe that, more scalable worlds can be developed by using a fully decentralised control with a constrained communication model. Different activities far apart from each other could be carried out without blockage and communication overhead might be greatly reduced. According to our knowledge, there is no systematic TM approach dealing with virtual worlds of this nature. To cope with these issues and achieve better performance, we propose a decentralised TM approach to potentially handle complex hierarchical models [5], using a flat structure in this paper. This work put more emphasis on actual simulations for local causality constraint and comparison of proposed method with non-synchronised and decentralised approaches.

## III. The Proposed Mechanism

This section presents a fully decentralised TM approach for dynamic, and potentially hierarchical, models of scalable virtual worlds with restricted communication that are described in section I. The basic aim of this work is to get a consistent state of a given world. To illustrate the proposed time advance mechanism, we use a simple world of 1*4 regional grid that is shown in fig.1. In future it will be extended for the hierarchical models presented in [5]. Fig.1 highlights two examples showing neighbouring regions with respect to a

region marked with a star. It gives us an idea about the regions that might be directly affected by the events generated by a region. Each region shows its current LBTS value, Lookahead value, the latest LBTS values of adjacent regions, and status of Local Queue. A circle highlights a federations with respect to a federate with a star. We believe that the assumption of constrained communication in conjunction with a fully decentralised approach potentially reduces complexity and delay with a decrease in the number of interacting servers. It therefore improves interactive user experience. It is important to note that the proposed TM approach uses a flat infrastructure for direct communication between the servers.

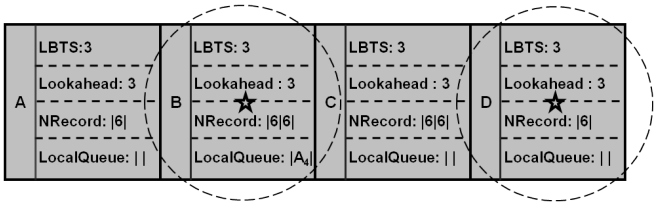


Fig. 1. Illustration of the proposed Time Management with constrained communication model.

We redefine the terms federate and federation for dynamic hierarchical models with a decentralised control and constrained communication. A *federate* in this work is a server executing a region. In general a region may have hierarchical structure, but in our examples we restrict to simple square cells. A *federation* is defined with respect to a federate and is a collection of federates that share boundaries with it. In general the federation associated with a federate must include all those regions that might generate events that could directly affect the federate. Hence, each federate participates in a number of different federations, and the functionality of the RTI is distributed among federates. The concepts of LBTS and Lookahead values are used to maintain the local causality constraint. Each federate in the proposed mechanism processes its events when they are safe in consultation with the adjacent federates. It carries out the central part of the mechanism and follows a straightforward approach that is presented in Algorithm 1. Each federate also provides its federate LBTS value to other federates in its federation and guarantees never to generate events earlier than the federate LBTS. Hence, the local LBTS is calculated as the minimum of the neighbouring federate LBTSs and the earliest queued event (if any). The Lookahead is added to the local LBTS and is then sent to its neighbours if and only if the LBTS increases. This definition has a recursive nature and, especially at system startup, a number of updates to the LBTS may occur before the local LBTS reaches the point where queued events can be processed. A push strategy is used to send federate LBTS values to adjacent federates with the aim of reducing potential overhead in communication and minimising temporary blockage. A federate ensures that timestamped messages destined for a neighbouring federate are delivered before sending its LBTS information thus guaranteeing that messages will never arrive in a federate's past. Being a conservative algorithm, it always considers a positive Lookahead value. It achieves traditional guarantees and significantly reduces intermediate processing elements (hops) and dependencies by directly communicating with neighbouring regions.

Each federate executing Algorithm 1 allows a safe range of event processing based on an LBTS value. It maintains a LocalQueue, LBTS and Lookahead values, and an array AdjacentLBTSValue that stores the latest LBTS value received from adjacent federates. A value in AdjacentLBTSValue changes dynamically when a new LBTS value is received. The main loop is executed while the simulation is running. To guarantee that the events are processed in their temporal order, a new LBTS value is calculated at the start of each iteration. Later on, an event with the earliest timestamp is processed if it is safe (when its timestamp value is less than or equal to the LBTS), and might generate more internal and/or external events in response. It schedules new events and repeats this process for the new earliest event. A simple condition is used that never allows processing of an event with timestamp greater than the current LBTS value. To simplify the consideration of transient messages for an LBTS computation, a federate is forced to send any destined messages before sending its LBTS value. The LBTS computation in the proposed method is straightforward in terms of transient messages that might require traversal of different components in traditional hierarchical systems. Time management calculations are simple and communication is localised. At some times there might be no activity in some federates. The federate must maintain its LBTS value for its neighbours but could sleep until an event or updated neighbouring LBTS value is received. When an LBTS update arrives, the federate updates the corresponding entry in AdjacentLBTSValue and wakes up the process if it is sleeping. Similarly, when a new event arrives, it is added to the LocalQueue and wakes up the process if necessary. The primary aim of our work is to ensure a consistent world with an emphasis on reducing communication delays and does not consider Quality of Service requirements at this stage.

## IV. Evaluation and Comparison

### A. Illustrations

Consider, a federation B in fig.1 to explain the basic time advance with an emphasis on executing safe events. The current LBTS value is 3 and it wants to process an event with timestamp 4. A new LBTS value is computed with the help of adjacent federates that is smallest among a set of LBTS values received from the adjacent federates (stored in NRecord) and timestamp of the smallest event in the LocalQueue (timestamp is the subscripted value). The new LBTS value in this case is 4 which allows B to execute the event with timestamp 4. The proposed approach does not impose global dependency allowing federations far apart from each other to take independent decisions for their time advance. Similarly, federations having a common federate might be able to carry on with their processing without blocking each other. However, in certain situations a time advance might be temporarily blocked

```
Data: LocalQueue, LBTS, Lookahead, AdjacentLBTSValues
// Initialisations
// In general, the set of adjacent federates might change dynamically based on split and merge operations [5]
int n = Number of adjacent federates;
int AdjacentLBTSValue[n];
for (i = 0; i < n; i++) do
    AdjacentLBTSValue[i] = 0; // changes dynamically with the LBTS value sent by adjacent federate i
end
int LBTS = -1;// In order to force distribution of an initial LBTS value
int NewLBTS = 0;
Insert initial event(s) to LocalQueue; // used for synchronisation with other federates
// Main loop of program for safe processing
while (System is running) do
    // Update LBTS value if necessary
    NewLBTS = Min_{i=0}^{n-1}(AdjacentLBTSValue[i]);// determines minimum of LBTS values of adjacent federates
    if (LocalQueue has Events) then
        NewLBTS= Min(NewLBTS, Timestamp of earliest LocalQueue event);
    end
    if (NewLBTS > LBTS) then
        LBTS = NewLBTS;
        Send (LBTS + Lookahead) value to the adjacent federates;
    end
    // Check for an event that is safe to process
    if (LocalQueue has Events and Timestamp of earliest LocalQueue event ≤ LBTS) then
        Process Event; // Remove the event and may generate new internal and external events
        Schedule internal and external events if any; // External events are sent to adjacent federates via messages
    else
        Go to Sleep;
    end
end
```

**Algorithm 1**. The Proposed Decentralised Time Management Approach

because of adjacent federates with smaller LBTS values. Since each federate is continuously trying to advance its time, these states are resolved quickly.

### *B. Simulation Results*

To verify the effectiveness of the proposed mechanism, we have simulated the temporal order of the events for a simple scenario presented in fig.2. It also shows the flow of events for a simple application that could violate local causality where agents in regions raise flags in response to other events and an observer should only see certain combinations. Region A raises a flag. After a delay, region B copies A and, later, region C copies B. Region D observes both B and C but hence should never see a flag raised in C before B. However, if messages are delayed and time management is not enforced correctly, there is a potential threat to correct synchronisation.

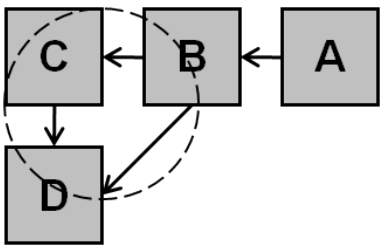


Fig. 2. The simulated world and events flow model.

Region A is only adjacent to B and has no direct impact on the activities of C and D. The purpose of its inclusion is to show that the proposed mechanism need not consider events from arbitrary regions. The aim of this simulation is to determine if the proposed scheme maintains the local causality constraint. The proposed method considers the entire set of adjacent regions (hence D considers both B and C) and therefore achieves a consistent state. However, the simulation may fail to get a consistent state if a significant region is ignored.

A simulation run of the proposed method is presented in fig.3. It assumes an initial value of $-1$ for the LBTS values and a constant Lookahead value of 3 for each region. The initial values of NRecord (holding the LBTS values of adjacent regions) are all 0. The local queue of region A has an event with timestamp 1 (represented as $I_1$) that triggers the simulation while other queues are empty initially. The events are marked as $X_{timestamp}$, where X is the name of a region and the value of a timestamp is the sum of LBTS and Lookahead of a federate. An event generated in this simulation aims to tell the adjacent regions that it has raised its flag. The LBTS updates are sent to neighbouring regions via messages that might have different random delays. A newly generated event is placed in a corresponding Event Generated queue and sent to the adjacent regions. When these events are received by the corresponding regions, they are placed in local Queues and processed in their temporal order. A processed event at a given time is shown in bold and enclosed in square brackets. An LBTS value is calculated based on the values in NRecord that is updated each time a new LBTS for a region is calculated. Since, the messages between a pair of regions are delivered in sequence, the correct temporal order is maintained at any given time. A region updates the corresponding status as a flag set event is processed by a region. We have suppressed simulation steps for time updates in fig.3 and 4 due to space limitations.

| | Region A | | | | | Region B | | | | | Region C | | | | | Region D (Observer) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Step | Flag | LBTS | NRecord B | Local Queue | Event (s) Generated | Flag | LBTS | NRecord A,C,D | Local Queue | Event Generated | Flag | LBTS | NRecord B,D | Local Queue | Event (s) Generated | Flag | LBTS | NRecord B,C | Local Queue |
| - | 0 | -1 | 0 | $I_1$ | Empty | 0 | -1 | 0,0,0 | | Empty | 0 | -1 | 0,0 | | Empty | 0 | -1 | 0,0 | |
| 0 | 0 | 0 | 0 | $I_1$ | Empty | 0 | 0 | 0,0,0 | Empty | Empty | 0 | 0 | 0,0 | Empty | Empty | 0 | 0 | 0,0 | Empty |
| 1 | **1** | 1 | 3 | **[$I_1$]** | $A_4$ | 0 | 0 | 0,3,3 | Empty | Empty | 0 | 0 | 3,3 | Empty | Empty | 0 | 0 | 3,3 | Empty |
| 2 | **1** | 3 | 3 | Empty | Empty | 0 | 0 | 3,3,3 | Empty | Empty | 0 | 3 | 3,3 | Empty | Empty | 0 | 3 | 3,3 | Empty |
| 3 | **1** | 3 | 3 | Empty | Empty | 0 | 3 | 3,3,6 | Empty | Empty | 0 | 3 | 3,6 | Empty | Empty | 0 | 3 | 3,3 | Empty |
| 4 | **1** | 3 | 3 | Empty | Empty | 0 | 3 | 3,6,6 | $A_4$ | Empty | 0 | 3 | 3,6 | Empty | Empty | 0 | 3 | 3,6 | Empty |
| 5 | **1** | 3 | 3 | Empty | Empty | **1** | 4 | 4,6,6 | **[$A_4$]** | $B_7$ | 0 | 3 | 3,6 | Empty | Empty | 0 | 3 | 3,6 | Empty |
| 6-11 | **1** | 9 | 9 | Empty | Empty | **1** | 6 | 9,6,9 | Empty | Empty | 0 | 6 | 7,6 | Empty | Empty | 0 | 6 | 7,6 | Empty |
| 12 | **1** | 9 | 9 | Empty | Empty | **1** | 6 | 9,6,9 | Empty | Empty | **1** | 7 | 7,9 | **[$B_7$]** | $C_{10}$ | 0 | 6 | 7,6 | Empty |
| 13 | **1** | 9 | 9 | Empty | Empty | **1** | 9 | 10,9,9 | Empty | Empty | **1** | 7 | 7,9 | Empty | Empty | 0 | 7 | 7,9 | $C_{10}$ |
| 14 | **1** | 9 | 9 | Empty | Empty | **1** | 9 | 10,10,9 | Empty | Empty | **1** | 9 | 9,9 | Empty | Empty | 0 | 7 | 7,10 | $C_{10}$ |
| 15-17 | **1** | 12 | 12 | Empty | Empty | **1** | 10 | 12,12,10 | Empty | Empty | **1** | 10 | 12,10 | Empty | Empty | 0 | 7 | 7,10 | $C_{10}$ |
| 18 | **1** | 12 | 12 | Empty | Empty | **1** | 10 | 12,12,10 | Empty | Empty | **1** | 10 | 12,10 | Empty | Empty | **B** | 7 | 9,10 | **[$B_7$]** , $C_{10}$ |
| 19 | **1** | 12 | 12 | Empty | Empty | **1** | 10 | 12,12,10 | Empty | Empty | **1** | 10 | 12,10 | Empty | Empty | **C** | 10 | 12,10 | **[$C_{10}$]** |

Fig. 3. Illustration of a simulation run for the proposed Time Management approach.

At the start of the simulation (see fig.3) a number of LBTS updates messages are processed before the initial event $I_1$ at region A is processed. It generates event $A_4$ for region B with timestamp 4 that is received at step 4. It updates its LBTS value and sends a message to region B to update its NRecord value. However, region B has to wait until time update messages are processed thus allowing event $A_4$ to process at step 5. Region B generates event $B_7$ for region C and D. Each region after generating an update message sends an LBTS update message which is processed by corresponding regions in temporal order. The event $B_7$ arrives at region C at step 12, but it is received at region D at step 18, even later than the response event $C_{10}$ generated by region C for D (that arrives at step 13). However, results show that the proposed mechanism does not allow D to process event $C_{10}$ until after it receives and processes $B_7$. This demonstrates that the events are processed in their temporal order.

The same specifications are used to simulate a non-synchronised approach with two different scenarios: in the first scenario, D considers only its own time information; in the second scenario, D considers C (but not B), in addition to its own time information. The simulation result for the first scenario is presented in fig.4. It is clear that at step 13, event $C_{10}$ is processed before arrival of event $B_7$, and hence a causality violation has occurred. The second scenario does the same and is therefore not included. We have also simulated a peer-to-peer environment that considers the entire set of regional LBTS values for the LBTS computation of a region that introduces longer delay in the computation of a new (effectively global) LBTS value. In this approach, region A is considered for LBTS computations by C and D even though it has has no direct impact and significantly increases exchange of messages over the network. A region has to wait for A's time advance to proceed. The peer-to-peer approach achieves the same results as our proposed method and are also not included due to space limitations.

In summary, all regions affecting a federate must be part of its federation, but no additional regions need be included. Based on the simulation results, it is clear that the proposed method maintains the local causality constraint. It considers a limited number of regions and is therefore more scalable and efficient compared with traditional centralised and distributed approaches. It potentially reduces communication overhead compared with peer-to-peer environments. In our examples with small numbers of regions, local calculations are based on a large proportion of the total regions, but in a large scale simulation environment only a very small proportion of regions would be involved in each calculation, making the proposed method flexible and scalable.

Since, the proposed mechanism can potentially be used for complex hierarchical models at a single level, a number of parameters might be used to compare it with traditional approaches. These parameters include dependencies, number of hops, complexity, delay, and scalability. It is worth mentioning that some of these parameters are dependent on each other. We believe that dependencies among the components and number of intermediate processing points are basic reasons for increased levels of complexity, larger delays, and poor scalability. The traditional hierarchical approaches are comparatively scalable, and easy to implement but need to keep their interfaces as simple as possible. However, with conservative approaches dependencies among different components at multiple levels (as discussed in [20]) introduce a larger delay and are therefore not easily scalable though better than centralised approaches. A message in a distributed hierarchical structure has to pass through a number of hops not only increasing complexity but also delays. Our proposed mechanism eliminates going through intermediate points by using direct communication between interacting federates. Similarly, certain federates cannot proceed further due to global dependencies among components at different levels in existing mechanisms thus degrading overall interactive experience. By using a fully

| | Region A | | | | | Region B | | | | | Region C | | | | | Region D (Observer) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NRecord | Local | Event (s) | | | NRecord | Local | Event (s) | | | NRecord | Local | Event (s) | | | NRecord | Local |
| Step | Flag | LBTS | B | Queue | Generated | Flag | LBTS | A,C,D | Queue | Generated | Flag | LBTS | B,D | Queue | Generated | Flag | LBTS | B,C | Queue |
| - | 0 | -1 | | $I_1$ | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty |
| 0 | 1 | 1 | | **[$I_1$]** | $A_4$ | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty |
| 1-3 | **1** | 1 | | Empty | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty |
| 4 | **1** | 1 | | Empty | Empty | 1 | 4 | | **[$A_4$]** | $B_7$ | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty |
| 5-11 | **1** | 1 | | Empty | Empty | 1 | 4 | | Empty | Empty | 0 | -1 | | Empty | Empty | 0 | -1 | | Empty |
| 12 | **1** | 1 | | Empty | Empty | 1 | 4 | | Empty | Empty | **1** | 7 | | **[$B_7$]** | $C_{10}$ | 0 | -1 | | Empty |
| 13 | **1** | 1 | | Empty | Empty | 1 | 4 | | Empty | Empty | **1** | 7 | | Empty | Empty | **C** | 10 | | **[$C_{10}$]** |
| 14-17 | **1** | 1 | | Empty | Empty | 1 | 4 | | Empty | Empty | **1** | 7 | | Empty | Empty | **C** | 10 | | Empty |
| 18 | **1** | 1 | | Empty | Empty | 1 | 4 | | Empty | Empty | **1** | 7 | | Empty | Empty | **B** | 7 | | **[$B_7$]** |

Fig. 4. Illustration of a simulation run for the non-synchronised approach.

decentralised approach, the proposed method allow different federations to process and advance their time without waiting for others having no impact on them.

## V. Conclusions and Future Work

This paper has presented a simple, but flexible, decentralised TM infrastructure and illustrated it with the help of a simple scenario. It is simple, scalable and allows a federation to take independent decision with distributed control among federates for direct consultation of interacting federates. It depends on the realistic assumption that events can only affect a finite and known set of adjacent regions. Simulation results support our claim that it achieves correct temporal ordering for randomly generated events. Our future work includes implementation and detailed analysis of the proposed decentralised mechanism together with our proposed JoHNUM infrastructure [5] using an open source virtual world development environment called OpenSimulator [24].

## References


[1] P. Rosedale and C. Ondrejka, "Enabling Player Created Online Worlds with Grid Computing and Streaming," *Gamasutra*, pp. 1–5, 2003.

[2] C. Ondrejka, "A PIECE OF PLACE: Modeling the Digital on the Real in Second Life," *Social Science Research Network Working Paper Series*, pp. 1–10, 2004.

[3] (2010, Apr) Second Life Homepage. [Online]. Available: http://www.secondlife.com/

[4] R.M. Fujimoto, *Parallel and Distributed Simulation Systems*. Wiley Interscience, 2000.

[5] U. Farooq and J. Glauert, "Joint Hierarchical Nodes based User Management (JoHNUM) Infrastructure for the Development of Scalable and Consistent Virtual Worlds," in *Proc. of the 13th IEEE/ACM Symposium on Distributed Simulation and Real-Time Applications (DSRT'09)*, Singapore, Oct. 2009, pp. 105–112.

[6] R.M. Fujimoto, "Parallel and Distributed Simulations Systems," in *Proc. of the 2001 Winter Simulation Conference (WSC'01)*, 2001, pp. 147–151.

[7] R.E. Bryant, *Simulation of Packet Communication Architecture Computer Systems*. Massachusetts Institute of Technology, 1977.

[8] K. M. Chandy and J. Misra, "Distributed Simulations: A Case Study in Design and Verifications of Distributed Programs," *IEEE Transactions on Software Engineering*, vol. 5, no. 5, pp. 440–452, 1978.

[9] R.M. Fujimoto, "Lookahead in Parallel Discrete Event Simulation," in *Proc. of the International Conference on Parallel Processing*, 1988, pp. 34–41.

[10] K. Pan, S.J. Turner, W. Cai and Z. Li, "A Hybrid HLA Time Management Algorithm Based on Both Conditional and Unconditional Information," *Simulation*, vol. 85, no. 9, pp. 559–573, 2008.

[11] R.M. Fujimoto, T. Mclean, K. Perumalla and I. Tacic, "Design of high performance RTI software," in *Proc. of the 4th IEEE/ACM Symposium on Distributed Simulation and Real-Time Applications (DSRT'2000)*, San Francisco, USA, 2000, pp. 89–96.

[12] J.S. Dahmann, R.M. Fujimoto and R.M. Weatherly, "The Department of Defence High Level Architecture," in *Proc. of the 1997 Winter Simulation Conference (WSC'97)*, Georgia, USA, 1997, pp. 142–149.

[13] D.A. Fullford, "Distributed Interactive Simulation: Its Past, Present and Future," in *Proc. of the 1996 Winter Simulation Conference (WSC'96)*, California, USA, 1996, pp. 179–185.

[14] D.C. Miller and J.A. Thorpe, "SIMNET: The Advent of Simulator Networking," *Proceedings of the IEEE*, vol. 83, no. 8, pp. 1114–1123, 1995.

[15] W. Cai, S.J. Turner and B.P. Gan, "Hierarchical Federations: An Architecture for Information Hiding," in *Proc. of the 15th Workshop on Parallel and Distributed Simulation*, California, USA, 2001, pp. 67–74.

[16] J.H. Kim and T.G. Kim, "Hierarchical HLA: Mapping Hierarchical Model Structure into Hierarchical Federation," in *Proc. of M&S-MTSA'06*, 2006, pp. 75–80.

[17] M.D. Mayjak, D. Clark and T. Lake, "RTI Interoperability Study Group Final Report," in *Proc. of the 1999 Fall Simulation Interoperability Workshop*, Florida, USA, 1999.

[18] M.D. Mayjak and S.T. Sharp, "Implementation of Hierarchical Federations," in *Proc. of the 1999 Fall Simulation Interoperability Workshop*, Florida, USA, 1999.

[19] W. Cai, G. Li, S.J. Turner, B.S. Lee and L. Liu, "Automatic Construction of Hierarchical Federations Architecture," in *Proc. of the 6th IEEE/ACM Symposium on Distributed Simulation and Real-Time Applications (DSRT'02)*, Texas, USA, 2002, pp. 50–57.

[20] A. Cramp, J.P. Best and M.J. Oudshoorn, "Time Management in Hierarchical Federation Communities," in *Proc. of the 2002 Fall Simulation Interoperability Workshop*, Florida, USA, 2002.

[21] G. Magee, G. Shanks and P. Haore, "Hierarchical Federations," in *Proc. of the 1999 Spring Simulation Interoperability Workshop*, Florida, USA, 1999.

[22] A. Cramp and M.J. Oudshoorn, "Employing Hierarchical Federation Communities in the Virtual Ship Architecture," in *Proc. of the 25th Australasian Computer Science Conference*, Melbourne, Australia, 2002, pp. 41–50.

[23] J.H. Kim and T.G. Kim, "Proposal of High Level Architecture Extension," in *Proc. of the 13th International Conference on Artificial Intelligence and Simulation*, Jeju Island, Korea, 2005, pp. 128–137.

[24] (2010, Apr) Open Simulator Homepage. [Online]. Available: http://opensimulator.org/