Challenges of Distributed Interactive Media State Transmission

Pedro Ferreira[1], João Orvalho[2], Fernando Boavida[3]
Communications and Telematics Group (LCT) of CISUC – Centre for Informatics and Systems of the University of Coimbra
Polo II, 3030 COIMBRA – PORTUGAL, Tel.: +351-239-790000, Fax: +351-239-701266, E-mail: {pmferr, orvalho, boavida@dei.uc.pt}

Abstract

This paper describes a middleware platform and application programming interface (API) for distributed interactive media state transmission, the Status Transmission Framework API [1], which was originally developed for distributed collaborative virtual environment applications.

We also aim to describe some of the research we are doing on extending the API to support all kinds of distributed interactive media, and more heterogeneous platforms and technologies, such as wireless and ad hoc networks, and small resources devices (such as PDA or GSM phones).

The emergence of pervasive networked data sources, such as web services, sensors, and mobile devices, enables context-sensitive, mobile applications [2]. The Status Transmission Framework can be used as model and support framework for the development of such distributed applications.

I. INTRODUCTION

A. *Distributed interactive media*

Distributed multimedia applications which have a high degree of interactivity, such as distributed collaborative virtual environment applications and some CSCW – Computer Supported Collaborative Work – applications, use interactive media. An interactive media may change according to user interaction with the media, which is opposite to streaming media, such as audio and video, which does not change at all with user interaction.

Distributed interactive media may be subdivided into discrete and continuous distributed interactive media, according to the type of media changes that may happen. We now describe these types of interactive media.

B. *Discrete distributed interactive media*

Discrete distributed interactive media changes its state only by user interaction. So, the state changes only in some perfectly defined instants. The media does not change due to the passing of time.

The mechanisms used to guarantee consistency for discrete distributed interactive media normally implement a causal order of messages [4].

[1] Superior School of Technology of Tomar, Polytechnic Institute of Tomar

[2] College of Education, Polytechnic Institute of Coimbra

[3] Informatics Department, Faculty of Sciences and Technology of Coimbra University

C. *Continuous distributed interactive media*

Continuous distributed interactive media changes both by user interaction and by the passage of time, and so messages need to include a timestamp so that the relative timing between actions can be reproduced.

The mechanisms used to maintain consistency in discrete distributed interactive media could not be applied to the continuous case [4], because, in this case, a causal order does not suffice.

D. *Consistency control and recovery mechanisms*

Several consistency control and recovery mechanisms exist for distributed interactive media. These are divided into optimistic and pessimistic approaches.

Pessimistic approaches prevent inconsistencies using floor control and locking techniques. These are very inefficient techniques not adequate for applications with high user interactivity requirements.

Optimistic approaches allow for short-term inconsistencies to happen and then correct them. The most important optimistic techniques are dead reckoning, timewarp and state requesting [5].

II. STATE TRANSMISSION FRAMEWORK

A. *General architecture*

The Status Transmission Framework [1] – STF - is a middleware platform for the transmission of state in distributed collaborative virtual environment applications.

This platform consists in an application-programming interface – API – that provides multiple services to applications. These services include state transmission and reception, distributed latejoin, virtual environment partitioning, internal virtual time synchronization and statistics collecting services to test the API.

The services provided by the STF API – designed and implemented in Java - are represented by Java interfaces, which are implemented by a channel manager – a special Java class.

The channel manager class – currently only STFChannelManager – uses a multipoint communication channel to handle the services provided by STF to a specific application.

This architecture allows for future expansion and optimizing of STF without having to change existing application code, just by creating new channel manager classes or changing the code of existing ones. An application that wishes to use the new channel manager only has to instantiate that manager class instead of the old STFChannelManager class. New service interfaces may be added to be used by applications specially designed for the new versions of STF.

We now describe each of the services provided in the first version of STF.

B. State transmission services

Methods and properties of the STFMessager interface represent the state transmission services provided by STF. This interface has methods to transmit state update messages, represented by instances of the STFStateMessage class. Each state update message includes the state – represented by any class that implements the STFState interface -, the entity key – represented by any class that implements the STFKey -, and the message priority.

The fundamental concept used by STF is that the virtual world is divided in entities, which have changing states. A state change of any entity always causes a state update message to be sent. Since the concrete state information and entity key defining information are very dependent on specific applications, STF has two interfaces to represent them. An entity is represented by classes implementing the STFKey interface. A state is represented by classes implementing the STFState interface.

State update messages are internally marked with timestamps, and organized into state interaction streams, as suggested by Georganas research [6], and represented in Figure 1. This is done in order to facilitate bandwith - optimizing strategies through the use of QoS – Quality of Service - related classifications.

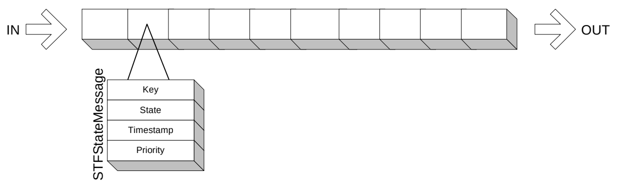

Figure 1 - State interaction streams

State transmission is optimized according to the state classification. This optimization is done on a per interaction stream basis. The STF API, through the STFState interface, allows states to be classified by:

- The state redundancy: A state may be redundant or essential. An essential state is always transmitted reliably. A redundant state is transmitted if not overwritten by a more updated state.
- The state volatility: A state may be volatile or not volatile. A volatile state is a state that can be completely lost.
- The state independency: A state may be independent or cumulative. An independent state completely defines a state, independently of any previously transmitted state. A cumulative state is a state update, dependent of previous state transmissions.

Some of the possible combinations of state classifications do not make sense and are consequentially illegal and the STF behaviour in these cases is undefined. For example, it is illegal to have a state that is simultaneously essential and volatile.

C. State reception services

Received state update messages are delivered to the application through methods of the STFApplication interface. This interface must be implemented by the application main class and contains methods to receive state update messages. These messages may be regular state update messages, unordered messages and latejoin process messages.

When a regular state update message is received STF buffers the message until it is delivered and processed by the application. The state update message may be discarded by STF using the same classification characteristics and similar mechanisms than those used to transmit the message.

A reception lag time may be enabled and configured to force regular state update messages to remain in the buffers during a minimal time period. This enhances the STF API ability to reorder received messages and in this way prevent the need for a timewarp in the application.

D. Latejoin services

A distributed latejoin service is provided by STF to allow nodes to join an ongoing session or rejoin the same session after being disconnected due to network failures.

This is a completely distributed mechanism implemented using replicated optimizing state update message caches, and a specially designed distributed latejoin protocol.

The STFLateJoin interface has all methods to enable and disable, configure and run latejoin processes. Latejoin messages are delivered to the application by methods of the STFApplication interface.

E. Partitioning services

A simple virtual environment partitioning service is provided by STF through methods of the STFPartitioner interface. This interface allows the application to enable and disable virtual environment partitioning and to define the used partition. A partition is a set of STFKey objects, representing a set of specific virtual environment entities.

F. Virtual time synchronization services

The STF API uses an internal virtual clock synchronization protocol that makes the use of external synchronization services not needed. The use of a virtual clock, as suggested by Augusto Ciuffoletti [7], is necessary because Java does not allow us to change the system clock.

The protocol used is completely distributed and independent of any external time source excluding the currently connected nodes of the session.

G. *Statistics collecting services*

A statistics collecting mechanisms intended to ease testing of the STF platform is also provided. The methods needed to control and use this mechanism are defined in the STFStatistics interface. The STFStatisticsData class represents the statistical data, for a specific stream or for the total of all the streams.

H. *Tests*

Several different tests were made to the STF platform and its services. Transmission and reception tests were published as part of a paper presented in IDMS 2001 [8], whilst latejoin and virtual time synchronization tests were published as part of a master thesis [1].

The most important conclusions taken from these tests were the correct functioning of the STF API and all its services, with guaranteed low message delays, and especially the correct adaptation to changing bandwidth requirements.

We present here some of the test results obtained for state transmission and reception [8]. The graph in Figure 2 shows the STF API adapting to bandwidth limits by discarding not needed state messages (volatile and redundant).

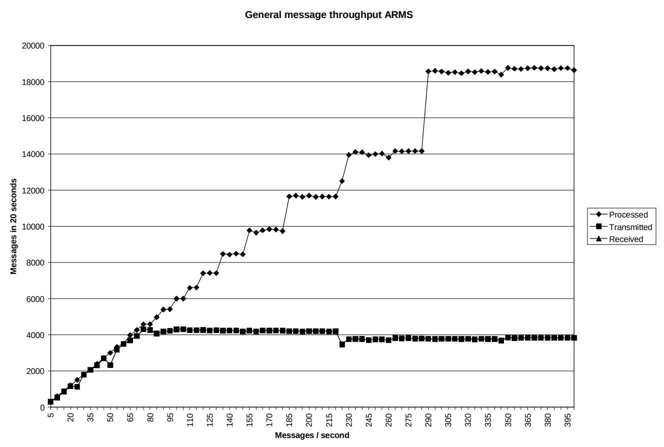

Figure 2 - STF Message processing tests

III. Future Challenges

There are currently many new challenges for network applications, including distributed state transmission middleware, and research is under way on many technological and architectural topics. We now present some of the most important areas of research and its corresponding challenges for distributed interactive media applications, and most specifically for distributed state transmission middleware.

A. *Wireless and ad hoc networks*

Currently, wireless networks and mobile phone networks are on the way of convergence.

Wireless networks appeal to an increasingly larger number of users, both because of their increasing speed and bandwidth and their decreasing price, especially in terms of end-user devices.

Now available in bandwidths until 54Mbps [9], and commonly used at 11Mbps, WiFi networks promise to be the leading force in this convergence scenario, together with UMTS - Universal Mobile Telecommunications System - and the future generation mobile phone standards, probably completely implemented using ipv6 [10]. There are currently some plans by US mobile phone companies, which aim to join UMTS and WiFi on a common mobile phone network. This may enable interactive multimedia application builders to free themselves of the limits imposed by the UMTS standard for mobile phone networks.

There is a parallel convergence of the different terminal hardware devices on a very Internet - similar architecture, but freed from physical (i.e., wire) limits.

The applications for these kinds of technologies are infinite, limited only by man’s imagination and current market needs.

Ad hoc wireless networks also present some challenges for new applications to address. How to allow the use of all the application functionality without a central coordinating entity is an obvious use for peer to peer network technology.

Telematics is the next wave in mobile computing [11], but needs to address issues such as security and privacy of data, scalability and device - independence. For the later, Java [3] is the platform of choice, especially J2ME – Java 2 Micro Edition.

B. *Pervasive computing*

The pervasive computing field aims to develop devices and services for true personal computing. The ideal situation is for the user to just forget that it’s using small computers. This is only possible with a very high degree of integration of these devices with the user everyday life.

The emergence of pervasive networked data sources, such as web services, sensors, and mobile devices, enables context - sensitive, mobile applications [2]. The Status Transmission Framework could be used as model and support framework for the development of such distributed applications. However, some problems must be addressed, such as small processing power, small memory and little or none persistent storage capacity. Other problem is data composition, caused by the large number of different data sources possible [12].

C. *Embedded systems*

There is also an opportunity for embedded systems design to include support for distributed interactive media state transmission.

Most available equipment that is really mobile has normally very little resources, in terms of memory, processing power and persistent storage capacity.

Embedded systems must address all the feature requests of an increasingly more ubiquitous networked world.

A distributed state transmission middleware platform would also be useful for applications where embedded systems are a required component. Smart sensors, for example, can be applied in medical implants connected to each other and a base station via a wireless network [13]. This kind of applications has a lot of potential for medical research, and future medical treatments. It also introduces a lot of addressable problems, such as network and processing requirements, low power operation, material constraints, reliability, robustness and fault tolerance, scalability, security and interference issues.

D. Multimedia integration

Tighter multimedia integration is desirable in the future in order to make the state transmission framework API more transparent and easy to use.

Our current research is now focusing integration with Java3D [3] and the JMF – Java Media Framework [3] - since the Status Transmission Framework API is entirely written in Java. One issue to address is how to represent the state of Java3D scene graphs, allowing for programmer introduced extensions as specified by the platform, and have the Status Transmission Framework handling all the needed network interchanges of information. Another issue to address is how to encapsulate in Java3D, extended by the Status Transmission Framework API, the state classification used to optimize network utilization.

IV. CONCLUSIONS

In this paper we described ongoing work and research on the Status Transmission Framework API, a middleware platform for state transmission in distributed interactive media applications.

Originally developed as a middleware platform for state transmission for distributed collaborative virtual environment applications [1], this platform is now being extended to support all kinds of distributed interactive media, for different target platforms. These include pervasive computing, embedded systems, wireless and ad hoc networks.

In addition to describing the architecture of STF and summarising the tests to which the prototype has been subject, the paper presented in general terms the main challenges that face new, emerging platforms, of which the Status Transmission Framework is a concrete example.

V. REFERENCES

[1] Pedro Miguel da Fonseca Marques Ferreira, "Transmissão de estados em ambientes de realidade virtual distribuídos e colaborativos", Faculty of Sciences and Technology of the University of Coimbra, Coimbra, Portugal, Master Thesis, 2001

[2] Norman H. Cohen, Hui Lei, Paul Castro, John S. Davis II, and Apratim Purakayastha, "Composing Pervasive Data Using iQL", *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002),* Callicoon, New York, 20-21 June 2002, pp. 94-104

[3] Javasoft, http://java.sun.com/

[4] Jurgen Vogel, Martin Mauve, "Consistency Control for Distributed Interactive Media", *ACM Multimedia* 2001, pp. 221-230

[5] Martin Mauve. "How to Keep a Dead Man from Shooting", in *Lecture Notes in Computer Science, Vol. 1905: Hans Scholten, Marten J. van Sinderen (editors), Interactive Distributed Multimedia Systems and Telecommunication Services*, Springer-Verlag, Berlin Heidelberg, 2000, pp. 144-157. (*Proceedings of IDMS 2000 – 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, CTIT / University of Twente, Enschede, The Netherlands, October 17-20, 2000).

[6] Shervin Shirmohammadi and Nicolas D. Georganas. "An End-to-End Communication Architecture for Collaborative Virtual Environments", *Computer Networks Journal*, Vol.35, No.2-3, Febr. 2001, pp.351-367.

[7] Augusto Ciuffoletti, "Uniform Timing of a Multi-cast service", *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, May 31 to June 04, 1999, pp. 478, Austin, Texas

[8] João Orvalho, Pedro Ferreira and Fernando Boavida, "State Transmission Mechanisms for a Collaborative Virtual Environment Middleware Platform", Springer-Verlag, Berlin Heidelberg New York, 2001, pp. 138-153, ISBN 3-540-42530-6 (*Proceedings of the 8th International Workshop on Interactive Distributed Multimedia Systems – IDMS 2001*, Lancaster, UK, September 2001)

[9] The Working Group for WLAN standards, IEEE 802.11 Wireless Local Area Networks, http://www.manta.ieee.org/groups/802/11/

[10] S. Deering, R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification* , RFC 2460, December 1998.

[11] Chatschik Bisdikian, Isaac Boamah, Paul Castro, Archan Misra, Jim Rubas, Nicolas Villoutreix, Danny Yeh, Vladimir Rasin, Henry Huang, Craig Simonds, "Intelligent Pervasive Middleware for Context - Based and Localized Telematics Services", *Proceedings of the second international workshop on Mobile commerce*, Atlanta, Georgia, USA, 2002, pp. 15-24, ISBN 1-58113-600-5

[12] Norman H. Cohen, Apratim Purakayastha, Luke Wong, Danny L. Yeh, "iQueue: A pervasive Data Composition Framework", *Proceedings of the Third International Conference on Mobile Data Management (MDM'02)*, Singapure, January 8-11, 2002, pp. 146

[13] Loren Schwiebert, Sandeep K. S. Gupta, Jennifer Weinmann, "Research challenges in wireless networks of biomedical sensors", *Proceedings of the seventh annual international conference on Mobile computing and networking*, Rome, Italy, 2001, pp. 151-165, ISBN 1-58113-422-3.