Multicast Communication for Distributed Virtual Environments based on Active Networks

Flavio Paiva Junqueira — flavio@gta.ufrj.br
Renata Cruz Teixeira — renata@gta.ufrj.br
Otto Carlos M. B. Duarte * — otto@gta.ufrj.br

Grupo de Teleinformática e Automação - GTA
COPPE/EE - Programa de Engenharia Elétrica
Universidade Federal do Rio de Janeiro
P. O. Box 68504 - 21945-970 - Rio de Janeiro - Brazil

Abstract – This paper presents a communication model based on active networking concepts for virtual reality systems. This model provides a dynamic form to users join and leave multicast groups. The main features of the proposed model, required to suport large virtual worlds, are: fast construction of distribution tree, scalability, and reliability on demand.

Keywords: active networking, capsule, virtual reality, multicast.

1. Introduction

Virtual Reality (VR) offers different perspectives for human-computer interaction, allowing users to intuitively interact with a dynamic and immersive computer-generated environment. VR distributed systems provide the interaction of geographically dispersed users. Due to the communication and the VR system restrictions, the majority of the current distributed virtual environments (DVEs) does not support a large number of participants.

Research into large-scale virtual worlds has been growing significantly. The NPSNET[1], developed by the Naval Postgraduate School, is a simulation system for military training based on the distributed interactive simulation (DIS) protocol and the logical partition of the world into classes. The communication for the Distributed Interactive Virtual Environment (DIVE)[2] uses a reliable multicast protocol with negative acknowledgement (NACK). The Model, Architecture and System for Spatial Interaction in Virtual Environments (MASSIVE)[3] system uses a spatial model for data partitioning among clients. The SPLINE (Scalable Platform for Large Interactive Networked Environment) splits the world into locales, that can be processed separately.

These systems and other ones are based on multicast protocols to increase the number of participants. However, dynamism needed in DVEs, because of frequent exchange among groups, is not provided by the current multicast services. Another important issue is the large range of VR applications and their variety of requirements. DVEs architectures are normally domain specific. Therefore, a platform to support all these application types must be capable of adapting to a variety of requirements in order to admit a larger number of applications [4].

Active Networks provide exactly an easy-deployment and flexible network service. These networks are active since the network nodes perform computations on and modify the packet contents. It means that a given application carries out specific computations in the network nodes on its data flows in order to guarantee its right execution [5, 6].

In this paper, active networks are the basis for the development of a multicast communication model. This model aims at allowing the dynamism and the flexibility needed by VR applications, providing a network service concerning these issues.

This paper is organized as follows. In section 2, we discuss the multicast service. Section 3 provides active network concepts and definitions. We present the proposal of the multicast communication model well suited for DVEs and the active protocol for this model in sections 4 and 5. Finally, in section 6 the paper is concluded.

*This work received financial support from: UFRJ, FUJB, PROTEM-CC, CAPES, CNPq e COFECUB

2. Distributed Virtual Environments Requirements

A DVE is composed by objects where the information that describes an object is the object state. A distributed VR system is responsible for notifying the state of one object to all other participants in the virtual world. This permanent updating is important to maintain world consistency. All participants must receive updates from all the objects of the virtual world and it produces an enormous data traffic on the network [4].

However, usually, an user does not interact with all virtual world objects simultaneously. Therefore, it is possible to define the limits of a participant scope and reduce the number of exchanged update messages. This technique is known as data filtering.

An interesting way to implement data filtering is using multicast groups [4, 2, 1], since it is possible to directly map the filtering groups into multicast groups. It is important to stress that even when the number of participants in the world increases, inside each group the number remains the same. This property is essential for the environment scalability.

Nevertheless, the partition of the objects in multicast/filtering groups does not remain fix. Objects can move through the virtual world crossing the limits of the group area and changing its scope of interaction. While participants move, they may join or leave multicast groups dynamically.

In order to support multicast groups, the network layer must provide a service based on a routing algorithm. Although there are differences among the several existing algorithms, all of them need a kind of distribution tree [7]. The mechanism used by participants to join or leave a distribution tree must be efficient, so that to provide a dynamic service.

Besides dynamism, distributed VR systems need a reliable communication service. Furthermore, a DVE must be consistent. For this purpose, virtual world information must be distributed in a reliable way. But, not all kinds of information need reliability as, for example, audio and video data. Thus, a platform for VR applications must provide both the reliable and non-reliable service.

When a multicast application demands reliability, an approach using negative acknowledgments (NACKs) is better than one using positive acknowledgments (ACKs). It is due to the fact that sending ACKs for each receiver and for each packet received produces a network overload and needs to synchronize the server to this transmission of acknowledgments [7].

Therefore, an important DVE requirement is the multicast communication service. It must be efficient in addressing group changes, allowing the dynamism needed, and reliable, maintaining world consistency.

3. Active Networks - The Integrated Model

In an active network, routers or switches perform customized computations on the messages flowing through them [5]. Thereby, mechanisms used to inject these computations within the network are different for each one of the current proposed models. The discrete model separates injection of programs from processing of messages. It means that a network administrator will choose and download programs rather than individual users. The integrated model makes the opposite approach. Common packets, carrying only user data, are replaced by capsules, that carry small programs which encapsulate data.

The capsule approach was proposed by Tennenhouse *et al* [5, 6]. In this approach, execution of capsule programs (also known as code fragments) corresponds to customized computations needed by applications on its data flow, in the network. A flow is a sequence of capsules that follow the same path and require the same processing. Thus, execution of these programs is only possible if the computational model adopted by all active nodes is the same [8]. In this context, we define the computational model as being the instruction set and the available resources in the active nodes for the code fragments.

When a capsule reaches an active node, its content goes to a transient environment, where its program is carried out. The end of execution results in the finalization of resources used by the transient environment, so that it can be used to execute other capsules. To write a capsule procedure, the programming language provides control instructions and some way to access external methods during execution in the nodes. These methods are used to make basic computations on capsule contents and to make decisions related to capsule destination like: fission to be transmitted in more than one link; fusion to reduce the bandwidth usage; *rendez-vous* with other capsules to synchronize; and discard when the capsule is no more needed. The external access made by capsule programs can be made in a number of ways. It can be made by foundation components, which are methods implemented to access available globally resources, for example, routing table and link state table queries. The non-transient storage is also another external resource available, where capsules can read and update soft-state left by other capsules of the same flow. The last kind of external resource available is called extensions, where capsules can leave classes and methods to be used by other capsules.

Based on the integrated model concept, the same team developed the ANTS (Active Node Transfer System) [6]. The service offered by this system is flexible, since applications can introduce new protocols into the network by the definition of procedures to be executed by capsules in the network nodes. The ANTS replaces routers and end-hosts by active nodes and common packets by capsules, which contain users data and a reference to a procedure. A mechanism of code distribution exists to assure a dynamic and automatic download of the procedures to the nodes where they are needed.

In the ANTS, an active protocol defines the processing to occur across the network as a whole, to provide the service needed by the application. An active protocol is a collection of related code groups and code groups are a collection of related capsule types. It is important to remark that the code group is transferred by the download mechanism as a unit. So, a good definition of code groups is needed to enhance the code distribution.

As seen before, the capsule procedure execution occurs in a restricted environment that limits access to shared resources. To implement the capsule procedures, a programming language must be defined and a set of primitive instructions available in network nodes. The ANTS uses the Java language, because the Java compilation produces intermediate code. This kind of code has a better performance than a source code programming language, and is more portable, if compared to object code.

Primitive instructions to program the capsule procedures are grouped by categories. Among the current defined categories in ANTS, three are mentioned. The control operations category allows capsules to transmit, discard, suspend, and wake up themselves. Primitives to access the non-transient storage exist to provide access to the soft-state. To allocate resources, like bandwidth and processing time, there are environment access primitives.

The development of the active protocol presented in this paper is based on the ANTS proposal. Capsule procedure algorithms, that appears latter in this paper, uses the main primitives categories defined in this system.

4. Communication Model

The flexibility provided for network users by the ANTS system through the supply of general use elements, like soft-state and Java programming language, allows the fitness of the network service to VR applications. In some cases it is possible to obtain improvements on bandwidth, data delay, and the traffic of messages through the network, improving application performances.

An user entering a virtual environment is inside an area that corresponds to a multicast group. Inside this group he acts as both sender and receiver. Although, the interaction scope of this user, usually, goes beyond the limits of this area. Therefore, he must join other multicast groups that compose his scope, acting as receiver for these groups.

When an user gets in the world or changes his interaction scope (moving through the world), in fact he is joining and leaving multicast groups. Every time a participant joins a group, he needs to receive information about the part of the environment representing that group and about the group senders. Thus, the first step on the proposed model is group information search. This is done by sending a group information search capsule for any group member. After collecting world and group sender data, the capsule returns to the active node which has originated it. Then, to join the multicast group, as a receiver, the user sends a subscribe capsule to each one of the group senders. This capsule follows the way in the direction of each sender until it finds the node itself or an active node that already belongs to the group distribution tree. Additionally, it marks each active node in its way with link indications, as a soft-state. Figure 1 shows the path followed by a subscribe capsule. Multicast data capsules follow the distribution tree composed by the active nodes current link indications, these capsules are responsible for carrying the object state updates.

In addition to joining groups as receivers, a participant must join a group as a sender too. It is done by issuing a sender request capsule. The goal of this capsule is to reach all group participants. After receiving a sender request capsule, group members must reply with a subscribe cap-

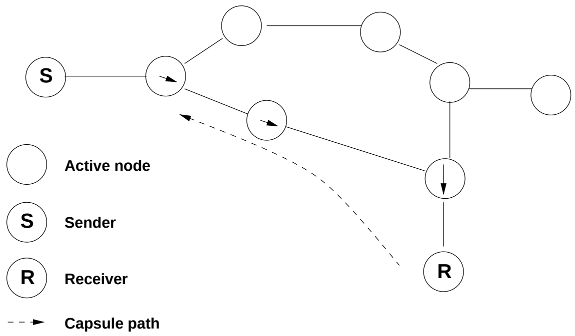

Figure 1: Subscribe capsule.

sule, joining the sender distribution tree.

There are two advantages in using this procedure:

- The speed to build the distribution tree, because since the closest group members receive the sender request capsule, they are able to start sending subscribe capsules to join the new sender group
- Fast warm-up, i.e., after receiving the first subscribe capsule, the sender starts immediately to send his data

Once the sender receives the first subscribe capsule it sends a resource reservation capsule. This capsule aims at reserving active nodes processing time and bandwidth needed to send multicast capsules.

When a virtual world participant cross the boundaries of the area representing the filtering group, it must leave all multicast groups that do not belong to his interaction scope anymore. If this user is just a receiver, he will leave the group when the soft-state link indications expire. When the participant is a sender, a warning capsule is responsible for disconnecting it.

The multicast protocol reliability is addressed by combination of NACK capsules and a data unit buffer that storages the units received by the multicast data capsules. The number of data units that the buffer admits depends directly on the data capsules mean sending time and the maximum available storage per group.

At the perception of a missing data unit sequence number, the host must send a NACK capsule with the missing numbers, for them to be searched in the network intermediate nodes. NACK capsules follow the way to the sender until it finds the required data units in the buffers. Then, data units are resent into multicast data capsules.

Thus, NACK capsules combined to the other capsules defined for this protocol uses a set of resources that the active network integrated model provides to offer a data distribution service based on scalable reliable multicast communication for distributed virtual environments. The next section presents procedure algorithms for the capsules defined.

5. The Active Protocol

An active protocol, as defined in the ANTS, is a collection of related code groups and each code group is a collection of related capsules. It is important to notice that the code group is the unit exchanged by active nodes. The active protocol proposed in this paper for the multicast communication model uses three code groups: data distribution, distribution management, and sender definition. Senders use data distribution and sender definition groups, and receivers use management distribution group. Figure 2 shows capsules and code groups in the protocol hierarchy.

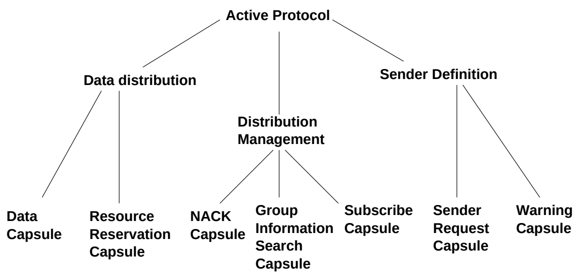

Figure 2: Multicast model protocol hierarchy.

5.1. Data Distribution Group

The data distribution group is composed by the multicast data capsule and the resource reservation capsule.

Multicast data capsules follow the distribution tree built by subscribe capsules. At each intermediate node, the procedure of the capsule sends a copy of the data capsule to each one of the links that are indicated in the group soft-state. The algorithm of the capsule procedure appears bellow.

Procedure UDados
If `there is a group soft-state`
    Then `extract`
    Else `discard the capsule`
If `data unit exists in the buffer`
    Then `discard the capsule`

```
For each link indicated in the soft-state
     Send a copy of the capsule
Put the data unit in the buffer
End-of-Procedure
```

When a sender first receives a subscribe capsule, he immediately sends a resource reservation capsule. This capsule goal is to make a reservation of processing time and bandwidth in the nodes that take part of the distribution tree, to the transmission of multicast data capsules. The resource reservation capsules must be sent periodically to confirm the usage of allocated resources by data flows. The resource reservation procedure is illustrated bellow.

```
Procedure Resource Reservation( CPU
cycles, bandwidth)
If there is group soft-state
     Then extract
     Else discard capsule
Allocate CPU cycles in the maximum to
the flow
For each link indication in the
soft-state
     Allocate bandwidth in the maximum
     to the flow
     Send a copy of the capsule
End-of-Procedure
```

5.2. Distribution Management Group

In the distribution management group there are three types of capsules: subscribe, group information search, and NACK capsules.

The subscribe capsule search the multicast group soft-state for the indication of the next links that composes the distribution tree. In case the soft-state does not exist it is created. If the link indication in the direction to the receiver interested in the multicast group already exists, the algorithm checks if the indication will expire soon, if so, the capsule must follows its way to maintain the soft-state. On the other hand, if there is not an indication, it is created and the capsule continue its way to the sender. It is important to ensure that the expiration time must be maintained by link indication and not by the complete group soft-state. The system demands this capsules to be periodically resent. This solution avoid subscribe capsule implosion, because only some of the capsules follow the path to the sender. Therefore, this is a scalable mechanism. A participant leaves a group when he stops sending subscribe capsules.

```
Procedure Subscribe
If there is group soft-state
     Then extract
     Else create
If there are link indications
     Then If time is near expiration
         Then
              Update indication
              expiration time
              Follow the way to the
              sender
         Else discard capsule
     Else
         Create link indication
         Follow the way to the sender
End-of-Procedure
```

If a participant wants to join a group, first, it must search information of the virtual world and the current group senders. Therefore, it sends a group information search capsule to any member of the group. After getting the data needed, the capsule returns to the node which has sent it.

```
Procedure Information Search
If capsule has already reached the
destination
     Then If capsule is at the source
         Then delivery data
         Else send capsule to the source
     Else If capsule is at the
     destination
         Then
              Extract group data
              Send capsule to the source
         Else Send capsule to the
         destination
End-of-Procedure
```

For each multicast group, in a given active node, there is a buffer that storages a certain amount of data units. While carrying out the NACK procedure, the required data units are searched in this buffer. In case they are found in the intermediate active node, the retransmission request is done. On the other hand, to avoid the NACK capsules implosion, on passing by an active node, the capsule must check over the group soft-state. If any NACK capsule has already asked for the same missing numbers on this way, the current NACK capsule is discarded, otherwise, it records its pass and follows its path.

```
Procedure NACK
If there is group soft-state
     Then extract
     Else discard capsule
If data units required are in buffer
     Then
         Retransmission request
         Discard capsule
     Else If another capsule had
     required the same data units
         Then discard capsule
```

```
        Else follows the inverse way
End-of-Procedure
```

5.3. Sender Definition Group

The sender definition group is composed by warning and sender request capsules. These capsules inform group members when a sender joins or leaves the group.

Sender request capsules are used by a participant when entering a group as a sender. This capsule replicates itself and sends a copy on each link indication of the distribution tree. The goal of this capsule is to reach all group members.

```
Procedure Sender Request
Extract group information
If there is not group information
    Then discard capsule
For each link in the group soft-state
    Send capsule copy
End-of-Procedure
```

A sender issues a warning capsule to inform all the other members of the group that he is leaving the group. Thus, the receivers no longer need to send subscribe capsules to that sender.

```
Procedure Warning
If there is group soft-state
    Then extract
    Else discard capsule
For each link indication in the group
soft-state
    Send a capsule copy
End-of-Procedure
```

6. Conclusion

Distributed Virtual Environments demand a flexible multicast communication system to fit a variety of applications. Moreover, it must provide dynamism and reliability. The active networking concept seems to be a good answer to these requirements.

In this article we proposed a communication model and an active protocol, including a collection of capsules, based on the active network integrated model. This protocol proposed a dynamic form to join and to leave multicast groups, as needed by a great number of VR applications.

The main advantages of the proposed active protocol are:

- A fast distribution tree construction, needed when user multicast group changes, obtained maintaining a low data message delivery delays.
- Scalability, due to the capsule-implosion avoidance mechanisms, essential to large virtual worlds
- Reliability on demand, i.e., the protocol support reliability only when it is required

As future works, we propose an implementation of this active protocol using the available ANTS prototype and its simulation to obtain quantitative results of message delivery delays, group change delays, and also scalability.

References

[1] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham, "Exploiting Reality with Multicast Groups," *IEEE Computer Graphics and Applications*, Sept. 1995.

[2] O. Hagsand, "Interactive Multiuser VEs in the DIVE System," *IEEE Multimedia*, pp. 30–39, Spring 1996.

[3] C. Greenhalgh and S. Benford, "MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading," in *Proceedings IEEE DCS'95*, (Vancouver, Canada), 1995.

[4] R. C. Teixeira and O. C. M. B. Duarte, "Requisitos de Ambientes Virtuais Distribuídos de Larga Escala," in *I Workshop de Realidade Virtual*, (São Carlos), pp. 26–35, Nov. 1997.

[5] D. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. Wetherall, and G. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine*, pp. 80–86, Jan. 1997.

[6] J. G. David Wetherall and D. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols," in *IEEE OPENARCH'98, a ser publicado*, (São Francisco, CA), Apr. 1998.

[7] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *Journal on Selected Areas on Communications*, vol. 15, Apr. 1997.

[8] F. Junqueira, L. H. K. Costa, and O. C. M. B. Duarte, "Uma Arquitetura de Roteamento com Qualidade de Serviço baseada em Redes Ativas," in *II Seminário Franco-Brasileiro de Sistemas Informáticos Distribuídos*, (Fortaleza, Ceará), pp. 257–268, Nov. 1997.