Use of Atomic Action Principles to Co-ordinate the Interaction between TINA Service Managers.

Guy Reyniers, Patrick Hellemans
Alcatel CRC Antwerp
e-mail: grey@rc.bel.alcatel.be

Abstract: TINA provides an architecture to support the Telecommunication Services of the future [2,3]. This architecture consists of a number of Service Managers (e.g. Service Session Manager, User Service Manager) that co-operate to provide Telecommunication Services to the users. In certain cases, complex interactions are required and this paper suggests the use of Atomic Action (AA) transaction principles to provide the required co-operation mechanisms.

1 Introduction

The aim of the TINA architecture is to provide an architecture for efficient creation, deployment, operation and management of Telecommunication Services on a world-wide basis. It consists of a number of Service Managers that co-operate to provide Telecommunication Services to users.

An issue still under consideration within TINA is an effective mechanism for coordinating the interactions between Service Managers. In the specifications released during 1994, these interactions are modelled as unrelated synchronous method invocations. The problems with this approach are elaborated in section 2.

Within TINA, new session control mechanisms are being worked out to introduce grouping of different operation invocations. In this paper, we describe a method based on Atomic Action (AA) principles, which is based on work performed in the RACE MAGIC project [1,4] associated with enhancements to B-ISDN signalling for Multimedia Multiparty Services. Whilst developing this approach, it was found necessary to extend AA principles to include the concept of transferring mastership over an AA. This is further described in section 3 of this paper.

It is our intention to validate the approach presented in this paper in the ACTS VITAL project, where we will be developing a MultiMedia MultiParty (MMMP) service within the framework of a TINA based distributed architecture.

2 Problems with the Current Approach

The major computational objects in the TINA Service Architecture are the Service Session Manager (SSM), User Service Session Manager (USM) and Communication Session Manager (CSM). An instance of service execution (a session) is realised by one SSM and some USMs (one per involved user). The SSM supports service execution, joining of users, and negotiation among users. It manages information and resources shared by users in the service execution, while the USM takes care of only those for its user. The CSM controls the connectivity and the allocation of resources related to the communication. More information can be found in references [2,3].
In the current TINA approach, the co-operation between an SSM, USMs and CSM is defined in terms of simple operations on the Service Managers. For example, in order to add another "Party" and a "Medium" (e.g. audio, video) to the Service Session, two unrelated operations are needed. If the "creation" of the medium fails for some reason (e.g. the user has no permission to add media to the Service Session), the user himself is responsible for requesting the removal of the requested additional Party.

A drawback of this approach is that it does not handle partial failures of related operations automatically. In general, following a partial failure, it is usually essential that the state of the system is brought back to the state before the requests were issued. When something goes wrong with any of the individual operations, the 'client' has to cover each individual failure and rollback scenario.

The example of a bandwidth manager for a trunk shows that it is not always possible to undo the effects of related requests. Suppose that the user's request implies a bandwidth increase (or allocate) for one connection (VC) and a bandwidth decrease (or deallocate) for another one. Take also into account that it must be possible to undo these operations because of the failure of an operation on some Service Manager. Suppose that the bandwidth increase is not possible without the decrease because of bandwidth limits on the access link. The following scenario causes a problem for the current TINA approach: deallocate bandwidth for connection1, another request (unrelated, for another user) allocates bandwidth for connectionX, allocating bandwidth for connection2 fails because there is insufficient bandwidth left. Reallocating bandwidth for connection1 can not be done so the 'undo' is not possible. If the decrease and the increase were performed in one AA the other Service Manager could not have seized the bandwidth.

Another example is the fact that the service creation (by a Service Factory) and service requests are unrelated. Suppose that the service session creation succeeds, but a user does not accept participation in the service session (a service request by the session owner). In this case the owner must request to delete the service session. Again this problem can be avoided by combining the service creation and the initial service request. This illustrates the need for a more elaborated co-operation mechanism between not only the service related managers (SSM, USM) but also between those and other computational objects (e.g. the factories).

An additional drawback is that operations are carried out one by one, not as a group. The additional delay introduced by each remote method invocation obviously has a negative impact on the expected performance of the system.

3 The Atomic Action Mechanism

Atomic Actions are groups of operations performed by multiple agents either as a whole or not at all. Performing an AA consists of 2 phases. In the first phase, the AA tree is constructed and agents indicate that they are willing to perform the requested operations. In the second phase, agents are instructed to commit (the requested operations must be completed) or to rollback (the system must be brought back in the state before the AA started).

The proposed Atomic Action mechanism is based on the Commitment, Concurrency and Recovery (CCR) Protocol. Some other transaction mechanisms use a different terminology, but the principles are the same. An Atomic Action method should be considered as an envelope that encapsulates primitive operations.