Object-Oriented Framework for a Scalable Multicast Call Modelling

M.I. Smirnov
GMD FOKUS
Hardenbergplatz 2, 10623 Berlin, Germany

The proposed approach for the multicast Call modelling framework considers logical entities participating in the Call process. A new mediator for the multicast Call establishment is introduced - the multicast agent (MCA), which is a software object moving across the internetwork and delivering to all members of the group all functionalities needed for reliable and flexible participation in a multicast session. The Call is totally distributed, is highly scalable due to the multicast tree decomposition into unicasts and fancasts. Performance aspects and discussed. MCA implementation is proposed within CORBA 2.0

1. Introduction

Current progress in the multimedia multipoint-to-multipoint collaboration support from the internetwork could be seen both as the challenge for the application designers and the testbed for the design alternatives. New applications supporting collaboration are on their way to customers in various domains such as distributed production management, scientific collaboration, medicine, distributed computer aided design, and so forth. However, the complete understanding of the multicasting technology especially in the case of multimedia collaboration still is not achieved. Hopefully the system model of that case can contribute to the desired understanding providing a sort of a framework for successful implementations. The framework model was designed to help new applications development and to provide for a broader spectrum of related research tasks. As in any object-oriented design one of the key success factors is object classes definition suitable for design and implementation within a particular domain and design targets. This paper introduces such classes for the multimedia multicasting with some reasoning behind the choice.

The paper is organized as follows. After observing a certain number of similar approaches (Section 2) and making a general conclusion about the need for the multicast specific framework for the call modelling (Section 3), the paper defines some very general case of multicast collaboration (Section 4). Section 5 introduces Multicast Agents - major mediator for the call establishment - from structural and functional viewpoints. The multicast tree development scenario and the definition of framework components could be found in section 6. Section 7 presents some performance aspects of the proposed approach. Section 8 deals with possible implementation of the Multicast Agent within the CORBA environment. The paper ends with a short conclusion.

2. Related Work

Traditional Call Establishment model emphasizes needed actions (functions); it is point-to-point oriented; it is dependent of existing network technology; it has data transfer orientation for the data transfer phase. On contrary to this, new smart connections require more complex behaviour, need to be highly scalable to cope with the existing internetwork dimensions and to be able to deal with objects rather than with pure data.

The Call will be treated in this paper as a protocol which controls and allocates network resources required for a multipoint multimedia session; these requirements are to be confirmed by the service provider in a form of a contract. A snapshot of recent works gives an impression of development directions in the area.

The Multimedia Call Establishment [1] gives a *global view* of the Call Establishment process and deals with several objects, *medium* being among the object set, including also *relation* and *attachment*. The *relation* object is proposed to use for synchronization purposes between different information streams, while the *attachment* object is needed to relate a user to a certain medium. This approach could not be overestimated because of the potential it gives to designers. However after introducing these objects the authors deal only with service primitives.

From the signalling viewpoint, one of the trends in a call control area is separation of Call Control from a Connection Control; introducing some building blocks in the Call Control information structure and generic signalling flow mechanism based on atomic actions [2]. The authors of [2] argue also in favour of greater functional specialization among different sets of signalling entities. When defining mentioned above building blocks authors present suddenly a set of objects, such as *Calls, Parties, Services, Service Groups, Attachments, Access Channel Endpoints* and *Mapping Elements*. All these elementary call objects could be Confirmed, Local or Virtual. When this set of elementary objects is introduced they are to be "combined and organized into composite structures that logically represent a service"[2]. It should be mentioned that on contrary to the global view approach the authors represent objects only within the context of a *local view*.

The control domain for the Call Modelling in a broadband Intelligent Network [3] is partitioned into 3 domains - Service Control (modelled by the IN service logic); Connection Control (being essential for the B-ISDN) and Session Control domain, where the Call Model is considered to have 2 parts: Session Control State Model (SCSM) and the Connection Coordination Model (CCM). The SCSM provides control functions for one session (equal to one call) which actually can invoke more than one connection. The SCSM model consists of 11 Points In Call; where PIC is a distinguished state in the state transition diagram indicating the flow of operations. The CCM describes the connection topology and types of connections in some abstract way in order to be network independent. This description is based actually on 4 objects: Leg - unidirectional communication path with address, bandwidth, QoS, direction, etc. as attributes; Connection Point - interconnection in a switching matrix between two Legs; Virtual Subscriber - agent for all Call resources to initiate or terminate; Port - a source or delivery point address. The Call is modelled with the use of these objects and their attributes.

In [4], which is actually based on the same concepts and testbed as [2], the object oriented Call model is represented with the number of 12 objects grouped into 3 generic classes, namely Local Object (8 elementary objects), Shared Object (2 elementary objects) and Virtual Object (2 elementary objects). Notably that the number of messages is rather small (Request, Accept, Ack and Reject).

The most relevant work deals with the concrete multicast protocol description for the so called Protocol Independent Multicast - PIM [5]. One can try also to derive the object model out of this paper.

Another multicast implementation, Reliable Process Group Communication - RPGC, deals with the group view maintained at each host (somewhat similar to a global view) and with a sort of multicast address hierarchy enabling group dynamics within a certain domain [6]. All group views are treated consistently due to the implementation of a “two-phase commit” scheme.

In [7] the author suggests the new metaphor of the network providing a sort of a universal service for the multimedia objects (Open Bearer Service); which services are also able to provide the multicast service.

In [8] authors of Conference Control Channel Protocol describe a scalable base for building conference control applications and list a number of urgently needed primitives of the multicast.

Sometimes, it is hard to define a correct set of call primitives used in different approaches; let us agree that Call Establishment capabilities will refer to generalized primitives in each model.

It is evident from this short survey that there is no approach available for object definition in general for this kind of model. Table 1 summarizes all observed approaches in purely quantitative manner. As a complexity measure we use a product of two numbers - the number of objects and the number of primitives. The average complexity of observed approaches is 34. Probably, any object-oriented approach for the call establishment in order to be manageable will have its complexity near this value. However, reasonable complexity is not the only requirement. This complexity has to fit properly a certain framework enabling modelling and piloting.

Table 1:

Model	Objects	Primitives	Complexity
Multimedia Call Establishment [1]	3	8	24
Call Control Information Model [2]	8	9	72
Call in Broadband IN [10]	4	5	20
Object Oriented Call Model [3]	12	4	48
Open Bearer Service [7]	4	4	16
Protocol Independent Multicast [5]	4	6	24
Reliable Process Group Communication [6]	4	10	40
Conference Control Channel Protocol [8]	2	12	24

What the multicast designer would like to have is, according to the author’s opinion, some framework which will provide the designer with typical objects (i.e. most general, suitable almost for any cases); their interrelations, some “method template” for every object, and implementation scenario. An attempt to provide such a framework for a multicast call establishment modelling was undertaken in this paper based on a very few considerations essential for a multicast support of collaborative work.

3. Design Approach

The standard approach for any network technology modelling for the sake of its evaluation according to one of the fundamental works in the area [9] is as follows. The model of the network technology, say, the protocol is to be represented with a set of its [distributed] functions and entities which are to be placed within the model of the network environment.

In this traditional model the communication environment usually is modelled by the limits of possible variations of major factors influencing the protocol. Normally the designer should be fairly happy if [s]he can prove via this model that the protocol (mechanism) under development operates satisfactorily under all given environment conditions.

Normally in a point-to-point case we have 2 communicating entities (objects) with one set of protocols in between facilitating any of required communication modes. This general model is also applicable for the Call Establishment. However in the [multi]point-to-multipoint case we have more complicated picture of communication. Introducing logical channels between the sender and each of the receivers of the multicast flow is probably a bad idea, especially for the Call model because it hides essential features of the multicasting. These features are: partial sharing of the multicast tree between receivers, negotiating custom quality of service (QoS) requirements for heterogeneous receiver applications, the receiver group dynamic properties. The latter means that we can't separate in time the Call establishment phase from the information transfer phase within the multicast. Multicast receivers are to be allowed to join and leave the session at any time. However, we can separate these two phases logically. Moreover, we can also take into consideration the possibility to advertise the multicast session for potential receivers during the Call Establishment phase.

The proposed approach for the Call modelling framework tries to separate further logical entities participating in the Call process as much as possible and specify their interfaces as well as object' methods behind them in such a way that real design could benefit from this separation in terms of model flexibility, completeness, feasibility and, mainly, manageability. The rationale behind this separation was perfectly introduced in [2]; it was stated there that such services as

- communication among heterogeneous terminal equipment;
- selectable quality of service;
- support of multiple connection calls;
- support of multiple party connections;
- individual control of mapping and presentation of information on each user's access,

"cannot be modified effectively by a monolithic call object or single state machine. It would be necessary to enumerate a very long list of service and capability variations and the resulting finite state machine would be so enormous that it would be unmanageable" [2, p.302]. Proposed logical separation of objects is essential for multicast. It distinguishes source - flow, network - flow, and receiver - flow interfaces.

4. The Multicast Case Description

The Internet philosophy it to provide the so called best-effort service. In order to prove the vitality of any networking approach it seems reasonable to show how this approach works within the Internet environment. If the approach shows all desired properties, such as robustness, scalability, reliability, etc. in this environment then one can expect even better behaviour in less hard conditions, which could be improved by additional management and control policies, resource allocation assurance scheme, etc.

In terms of communication mode parameters any multicast session could be [10] planned or spontaneous, managed or unmanaged, with controlled or uncontrolled access; with different number of senders. In terms of network environment it could also utilize the same multicast tree for all possible senders or use different multicast trees, make use of the same applications and/or QoS requirements throughout the receiver group or different applications/QoS requirements, etc.

Let us describe some generic multicasting case, having in mind application collaboration demanding multicasting support rather than pure conference. The multicast session starts as a planned event according to some schedule with one sender and a certain number of receivers who agreed to participate in advance (*dedicated* receivers - DR). That means that this single sender (initial sender) will issue some request with the list of dedicated receivers in it. The request message has certain parameters requiring certain services from the network, i.e. from the network service provider (SP). The SP has to confirm/reject requirements via separate message to all/some of the multicast group members (i.e. to initial sender or to all dedicated receivers). Negotiation mechanism on service parameters with the SP is not considered here. When confirmed in a form of some *contract* the multicast Call Establishment request has to follow the *multicast tree* (MCT) for the particular session, dedicated receivers being the leaves of this tree. We assume that MCT is pre-calculated logical structure spanning all DR in most efficient way [13].

More formally a contract could be presented as follows:

```
multicast_session_contract(management_domain,
   initiator, QoS_requirements,
   [number_of_simultaneous_senders],
   [session_duration], MCT, [modality],
   [MCT_development_policy]).
```

The MCT could be seen as the main routing information for the call establishment phase and as an association of network routers participating in the call process at the moment. Each router, obtaining a contract, has an ability to send the *join request* message for more destinations than it was required in the dedicated receivers list or to more hosts adjacent to it, thus *advertising* the future session. Note, that the number and set of advertised routers could be restricted by a *modality* parameter of a contract together with the *development policy* [15].Normally, every dedicated receiver upon receiving the *request* issues its own *acknowledgement* message to the sender of its *join request* message. The initial sender can start sending multicast packets when the number of acknowledgements received from DR is sufficient. This decision making is out of the scope of the paper.

This paper proposes a highly scalable approach:

- only one member of the multicast group negotiates initially with the SP, namely the so called initial source;
- receivers send their *acknowledgements* only to the nearest *splitting point* of the multicast flow;
- all active senders share the same multicast tree (that's why the SP has to limit probably the maximal number of active sources);
- all QoS negotiations with the receivers could be provided at the splitting points.

The call establishment thus continues during the information transfer phase of the session: acknowledgements are still coming to the initial sender and to splitting points; advertised receivers start joining the session; some of them can decide to leave the session; others can decide to request their own flow specifications different from the default flow specification. We also assume that any of the dedicated receivers can become a sender to the same session. This option could be enabled by a special *floor request* message sent to the group. The only difference of the floor request from the initial sender request is that it doesn't need to be confirmed by the service provider. For some period of time we assume that there could be several senders for the same multicast session sending in parallel.

For the sake of the design simplicity it is reasonable to decompose the whole call process into three phases: 1. Session initialization; 2. Continuous Session; 3. Session End. The last phase could be introduced as follows. If the session is planned for a certain period of time, then it has obviously to have some agreement on session termination which is assumed to be fulfilled by all current senders stopping their transmission and informing the SP about the session end. Another possibility could arise when no one receiver is listening to the session. However, the latter case has to introduce the group control entity within the multicast.

5. Multicast Agents

5. 1 Structural Motivation

The most general idea of the approach comes out of the decomposition which in turn follows the general definition of the multicasting system [11] (MCS):

$$MCS = (MCAgents, MCConnectivity, MCProtocols, MCDescriptionTools) .$$

The above mentioned decomposition affects two sides of the definition - the MC Connectivity, that is the Multicast Tree spanning all members of the multicast session; and the MC Agents which are defined in this paper as some objects capable to move across the network, clone, intercommunicate and perform a set of specified actions.

Topologically, any multicast tree could be seen as a composition of several serial interconnections (unicasts) and fanouts of nodes. Therefore, any communication supporting and providing multicast could be composed out of several unicasts and simple multicasts. The multicast could be called simple when the corresponding multicast tree is a fanout, for that reason a new term could be introduced - *a fancast,* i.e. inevitable multicast which could not be reduced. From these reasonings comes the idea of a scalable multicast. The multicast tree decomposition is presented in Figure 2, where simple composition components are put within smaller icons. This decomposition is complete, i.e. no more fanouts or unicasts could be found. However, this is not a single possible decomposition. Different structures which can arise during the call establishment will be addressed later. Out of this example it is evident that due to inevitable overlapping of unicasts and fanouts the functionalities needed to support unicasts and simple multicasts are to be placed within their intersection points, while these points are actually the *splitting points* of the multicast flow. A splitting point could be defined as a network router where packets of the multicast flow have to be reproduced in at least 1 copy. All over the paper we assume that receiving hosts connected to routers are not shown in the figures.

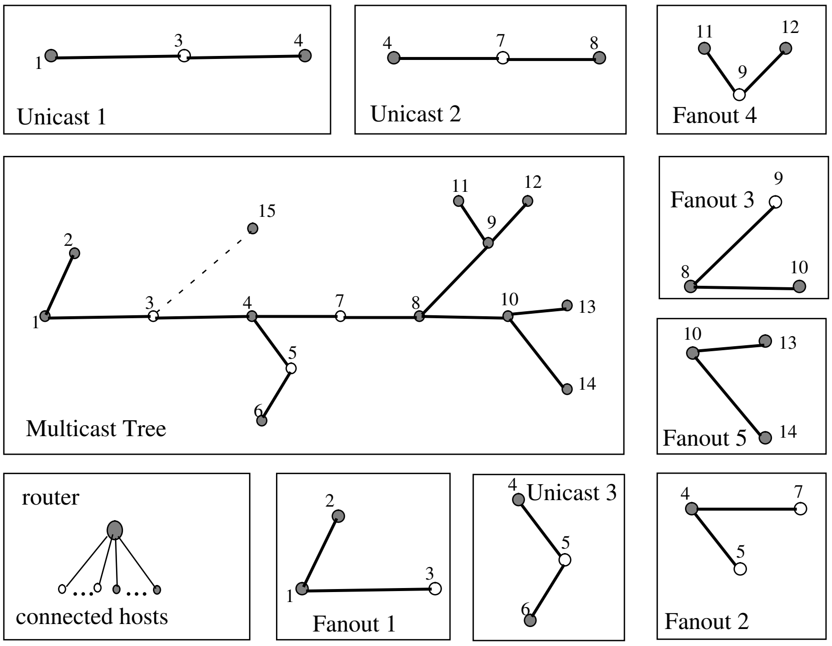

Fig. 1 A Sample Multicast Tree Decomposition

The desired scalability feature could be achieved via limited visibility out of each splitting point: we say that from each splitting point only a set of nearest reachable splitting points and receivers could be observed. This improves scalability and manageability of these partial multicasts. However, the need to provide the self-distribution of the multicast functionality over the multicast tree is the cost for these improvements.

5.2 Scalability

In this section the scalability feature for the multicast session will be introduced in some details. First we consider the mostly primitive case of the simplest fancast and a model for essential communication between involved entities; then we show how this scheme could be made scalable. In the following sections we'll try to show how more complicated interactions between components could be modelled.

The simple multicast case will consist of one sender and two receivers (Fig. 3). The only thing the sender has to do in this case is to copy packets to two destinations. The multicast flow direction is shown by solid arrows (Fig. 3 (a)); the call establishment flows of messages and corresponding components are shown in dashed lines (Fig. 3 (b)). We introduce here 3 generic functional components, namely:

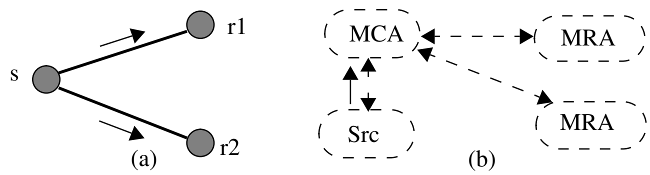

Fig. 2 The fancast development

1. *Src* - the source of the multicast flow (it could be an application or a composition of applications, each sending its own flow to a different media such as video, audio, joint editing/drawing application space, etc.);

2. *Multicast Agent (MCA)* - a generic functionality within the multicast flow source network node which communicates with the service provider and actually sends multicast packets to the receivers;

3. *Multicast Receiving Agent (MRA)* - a functionality within the receiver router; it accepts multicast packets and negotiates if needed with the MCA and the receiver host applications; also participates in the call establishment process.

Let us make this embryonic MCT grow a little bit (Fig. 4) and see what additional functionalities are to be introduced for the new case. When the intermediate node (*n*) is introduced between the sender and receivers we see that in relation to receivers this

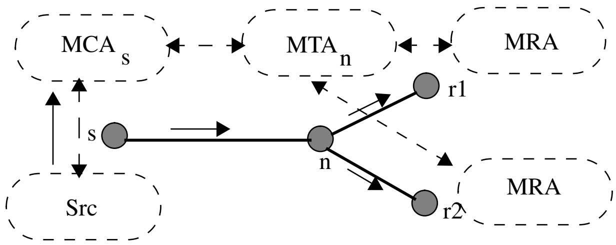

Fig. 3 Grown up multicast case

node plays exactly the same role as the sender *s* did in Fig. 3 except the acceptance of the multicast flow from Src. It looks like one MCA has been split into two agents - the initial MCA and the network agent (let us call this new object MTA - the *Multicast Transmission Agent*). It is useful to note that the transition from Fig. 3 to Fig. 4 could be made with the insertion of an additional node to the left of the initial source and with artificial movement of the multicast flow source to this new node. The functionality of MCA in Fig. 3 has been split into two parts with ordinary point-to-point connection (unicast) between them.

From this object presentation it could be seen that there are two different interfaces for the initial and the intermediate senders of the multicast flow; optionally the initial MCA can have also the MRA interface as well. If every receiver will exchange messages (for patching of losses of packets, for QoS negotiation, for receiver group dynamics control, for floor and application control, etc.) with the initial source throughout the whole MCT then this scheme will not scale well. Evidently, sooner or later the source node will become a bottleneck. In the proposed approach an exchange of messages between agents exists only during the call setup/disconnect phases. However, this exchange between receiver applications and MRA can exist at any time.

In this paper we are especially aware of the case of the so called sparse group multicast [12], when the number of receivers is very large comparatively to the number of possible senders; receivers could also be distributed across several domains. Current multicast systems are composed with applications and network support, sender and receiver applications communicating via the network. The network support consist of special address scheme, host extensions and routing support. Many authors argue in favour of more and more support for future multicast multimedia applications form the network. However, networks become more and more high speed, i.e. tend to provide less functionality during the data transfer. Hence, one of the requirements for a successful multicast implementation - to keep it simple for the network, e.g. following the Internet best-effort philosophy. The second reasoning comes from the fact that one can not change the placement of a multicast flow (source) for obvious reasons (human participants, cameras, data bases, etc.), however the *starting point* for the multicast tree could be placed *anywhere* within the network, provided that this anywhere place has a good connection to the source. Out of these two considerations comes the proposed scenario for a scalable multicast call establishment, which was named the *multicast tree development*.

6. The Multicast Tree Development

6.1 Scenario

To obtain an object model we'll use the term *agent* for some instance inside the sender host having all interfaces described above. The multicast session starts with the agreement between the network service provider and the multicast agent, the list of dedicated participants being one of the parameters of this agreement. We assume that somehow is calculated the MCT [13] which spans the initial source and all the dedicated receivers. The initial source (i.e. MCA) sends its *request* and after *confirmation* from the SP starts the process of the MCT development. Shortly, the process could be seen as follows: the MCA clones itself and places its embryonic objects in every node of the MCT in a step by step mode. Each embryonic multicast agent initially has only one (unicast) interface to communicate with the parent MCA (i.e with that MCA instance which had created it). During the MCT development each copy agent activates needed number of copies of needed interfaces. From the receiver viewpoint there is no difference to communicate with the remote MCA (placed near the multicast flow source) or with the nearest MCA. We assume that from the functional viewpoint each copy of the MCA provides the same service for the receiver[s].

This scenario will have the following script for the MCT shown in Fig. 1:

- Initial source node (MCA entity in node 1 or MCA(1) for short) establishes the contract with the SP; obtains confirmed contract;
- MCA(1) clones itself and sends copies to nodes 2 and 3;
- MCA(2) recognizes from the contract that it has only receivers attached to its current node (2) and invokes needed number of MRA interfaces;

- MCA(3) recognizes that it has no receivers attached, optionally invokes the *adv* interface and sends advertising messages to its hosts, but, first retransmits its copy to node 4;
- MCA(4) recognizes that it has different types of attached nodes and clones itself in two copies, then sends them to nodes 5 and 7; and so forth.

Note that the MCA copy could be placed in nodes 3, 5, 7 with the only *unicast* interface invoked; however we suppose that this ordinary unicast could be supported by normal network service. In some future, when a new receiver 15 (say, attached to node 3) will try to join the multicast session it will need to join with the nearest shaded node. If the MCA functionality would be provided at all MCT nodes, including those which had no initially attached dedicated receivers, then the MCA(2) obtaining the *join* request could simply invoke needed MRA interface. Note that within the proposed scenario only the source node makes the contract negotiation with the service provider.

What we have as the result of the MCT development process is a tree spanning all confirmed participants of the group; each being provided with the needed multicast functionality. In terms of a multicast flow distribution this tree could be seen as a hierarchy; however from the multicast call establishment viewpoint this tree is a distributed association of peer agents. These agents share the multicast flow being sent to a group by the source.

The embryonic MCA is shown in Fig.4. Within the Object-oriented programming ter-

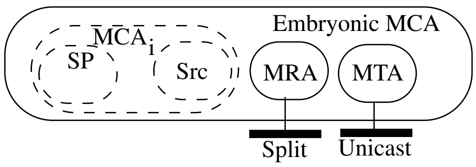

Fig.4 Embryonic MCA

minology [14] the Embryonic MCA could be seen as a complex object with SP, MTA. MRA, Src being its methods, however we'll call them also objects with their own methods (only *split* and *unicast* are shown here). The methods are invoked by messages according to the multicast call establishment scenario.

6.2 The Multicast Call Establishment Components

The Multicast Call establishment scenario development is not the purpose of this paper, however we would like to show what components are essential with the use of the MCT Development concept and the Multicast Agents object presentation. Actually we consider the MCT Development scenario as a framework for the Call Establishment Model design.

Following the above mentioned MCTD scenario and [1, 5, 6, 8] let's start with the source node.

The *Src* method of the complex MCA object is an object with the following methods:

- attach/add/detach/remove media;
- define medium-to-medium relations (for multi-media multicasting);
- include/exclude applications;

- floor control;
- give current multicast state (optional).

The *SP* method of the complex MCA object is an object with the following methods (parameters):

- enable/disable multicast session (contract);
- quality of service parameters (DR list; Media list).

The *MRA* method of the complex MCA object is an object with the following methods:

- filtering (Media list; QoS requirements list);
- include/exclude applications;
- attach/add/detach/remove media;
- patching (Medium, badly received packets);
- advertising (optional);
- floor control;
- participation control;
- give current multicast state (optional).

The *MTA* method of the complex MCA object is an object with the following methods:

- clone (MCA);
- send copy (MCA);
- invoke (MCA_method);
- prune (MCA_instance);
- split, i.e send copy (multicast packet).

We assume that when the SP method is activated it starts negotiation with the service provider for a multicast session with required dedicated receivers set and required QoS parameters for a certain media involved. This contract is assumed to be assured by the service provider all over the MCT. If different receiver hosts will require changes to the contract defaults then they have to negotiate them with their MRA.

Actually the MTA entity realizes a protocol for distributed multicast management; this particular method has to realize essential multicast functionality to achieve scalability of the sparse multicast groups.

The complete set of objects introduced to support the object-oriented framework for a scalable multicast is presented in Figure 5.

This set is called a framework because it is not elaborated yet in the sense of object messages, also some methods are optional and others could arise during the multicast implementation. These building blocks can support the whole lifecycle of the multicast session, including Call Establishment and Call Modification.

The Call Establishment phase needs to invoke the Src and SP objects to establish primary contract with the service provider and MTA object to clone, transmit and self-install the MCA complex object within the nodes of the MCT. Note that we suppose to clone not an abstract MCA complex object but its instance customized according to a pre-negotiated set of session parameters (dedicated receivers list; media list; default QoS parameters; maximum number of active senders, etc.). The Call Establishment phase continues with invoking of MRA objects and their local negotiations with local receivers, session advertising, etc.

The Call Modification phase can start even during the Call Establishment. For example some of the dedicated receivers could become unreachable; crossing the domain boundary may cause changes in the default session parameters, one of possible active senders may ask for a floor even before the start of the initial flow, etc. We assume that this could not cause any problem because any MCA instance after the self-installation is ready to accept any of the messages which could take place within the entire multicast session. The shortcoming of this approach is that we have no opportunity to change all global parameters of the multicast session (DR list, Media List and default QoS for the reserved Media). The only possibility to change these parameters is to renegotiate with the service provider via the SP object. This object could be invoked only by any of the dedicated receivers (each of which have a potential to become a sender). The only possibility to change the DR list is to apply to the initial source.

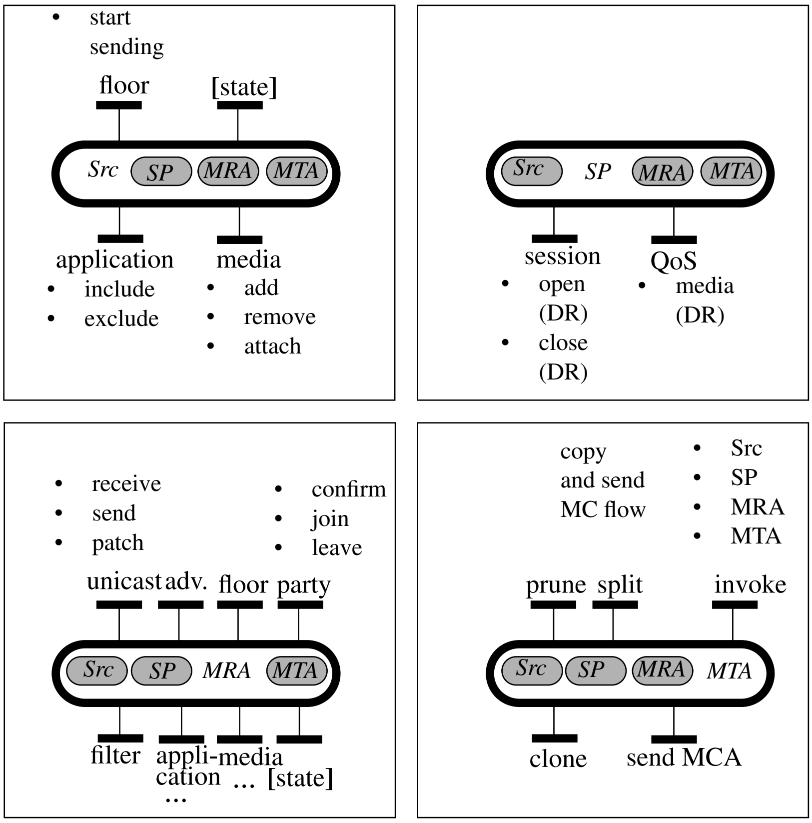

Fig.5 Object Framework Components

Note, that if the number of dedicated receivers is too large for a service provider to permit them to send in parallel, then the number of active sources is to be introduced. However this number is to be somewhat less than the actual possible number of concurrent sources because we assume no information exchange between the instances of the MCA. Consequently the only way for them to know the number of actual active sources is to listen to the multicast. However, due to a delay introduced by the network several sources could start sending simultaneously based on their local permission. To prevent this situation we assume simply that the MCA instance nearest to the last source has to stop sending when recognizes that actual number of active sources is greater than default maximum.

7. Performance aspects of the MCT Development

As long as the proposed agent technology for the MCT development is totally distributed we have to pay special attention to some potentially dangerous events during the development. Main features which will be addressed here are: scalability; manageability and global state. The global state problem exists because no one MCA has a complete knowledge of the global state, however MCA at all splitting points notify all receivers joined within their fanout in a bottom-up manner. Finally, the decision making on total MCT development quality is made by initial MCA. All reasonings will be made regardless the network structure; we require for the multicast agents only the knowledge of initial MCT. First, we assume that exchange of information between MCA entities exists only within each fanout.

Let us start with the formal description of the MCT under development which will facilitate some taxonomy of possible changes during the call. The multicast tree structure could be presented as

$$MCT = \langle L, \{S_l\}, \{R_s\} \rangle ,$$

where L is a number of levels in the MCT, i.e. the largest receiver-hop distance from the initiator to the tree leaf; $\{S_l\}$ is a set of sets of splitting points at each level $l, l \in \overline{1, L}$; $\{R_s\}$ is a set of sets of receivers which are to be connected to particular splitting point s, where $s \in S_l, l \in \overline{1, L}$.

That is we can define two mapping functions $f(S_l, R_S) \rightarrow N$, $\varphi(S_l, S_{l+1}) \rightarrow N$, where N is a set of integers. Assuming that MCT is a tree structure we can use simply $f(S_l)$ and $\varphi(S_l)$ to denote the number of receivers and the number of splitting points to be attached to the given splitting point. Where it will be possible we omit the second index for a splitting point, however a complete depiction should be $S_{l,t}$, where $l \in \overline{1, L}$, $t \in \overline{1, \varphi(S_{l-1,i})}$, i is a parent splitting point index.

The MCT under development is assumed to be a pre-calculated Steiner tree structure which may have or have not other network nodes (s-nodes) in it except dedicated routers [13]. During the development of this MCT by multicast agents a number of events may happen which will change pre-calculated structure, effecting different components of the MCT. Table 2 gives a summary of possible events during the development and shows how these events influence the notification process. Most of events could be caused by current network situation and the fact that splitting point routers are able to advertise future session for new receivers, which actually were not considered when the MCT was calculated.[1]

Splitting of a splitting point is considered as any of two possible cases: add new splitting point and remove [partly] receivers from the old one; or add two new splitting points and delete an old splitting point. When s-nodes are considered within the MCT (due to obvious reasons for sparse multicast groups), the operations of add/delete s-node are to be considered at the internetwork unicast level. Other cases (when s-node becomes a receiver or even a new splitting point) could be found in the Table 2. Note that only one event (Delete receiver) could be notified only at the very top of the MCT (when all of the nearest fanout acknowledgements will arrive). All the rest of these structural changes for initial MCT could be successfully resolved in a distributed manner. Note also that some cases may change the number of MCT levels (Degrade/Delete splitting point, and Delete receiver). And, finally, it should be mentioned that notification points presented in Table 2 will be valid if all migrations will take place at the

Table 2:

Changes to	Case	Definition	Notified by
R	Migrate receiver within one level	$S_{l,i} \to S_{l,j}$	S_{l-1}
	Migrate receiver between levels	$S_{l,i} \to S_{h,j}$	S_l if $l < h$
			S_h if $l > h$
	Add new receiver	$f(S_l) = f(S_l) + 1$	S_l
	Delete receiver(s)	$f(S_l) = f(S_l) - 1$	S_0
S	Migrate splitting point	$\{S_l\} \to \{S_h\}$	S_l if $l < h$
			S_h if $l > h$
	Add new splitting point	$\varphi(S_l) = \varphi(S_l) + 1$	S_{l-1}
	Delete splitting point	$\varphi(S_l) = \varphi(S_l) - 1$	S_{l-1}
	Degrade splitting point	$\varphi(S_l) = 1$	S_{l-1}

same branch of the MCT. In opposite case almost all of the events could be resolved only at the very root of the tree. However, we assume that the MCT (and its unpruned branches being the most probable candidates for the advertising) covers the network

1. However, according to the heuristic proposed in [13], the MCT development algorithm may try to make effective use of those sub-optimal MCT branches which were not pruned.

topology in a way somewhat similar to the existing multicast connectivity in the Internet [16]. That is why we assume that probability of moving any receiver from one branch to another branch of the MCT is negligibly small..

For the sake of the MCTD manageability it is reasonable to set some timer within the initial MCA, estimating the waiting time for the whole MCT development. The same type of timer should be set in every copy MCA, placed within splitting points to estimate the development time of their fanouts. Let us denote as t_d - the average one hop delay for the MCA messages; t_i - the maximal MCA installation time; τ - the timeout to be set at the splitting point as estimated time for developing its own fanout only, and $\tau = 2 \cdot t_d + t_i$. Let also *dN(h)* denote the number of positive acknowledgements (*acks*) from the fanout under development according to the MCT. However, at some level *h* of the MCT we may achieve the following number of positive *acks*:

$$dN(h) = \sum_{i \in \{S_{h-2}\}} [f(S_{h-1,i}) + \varphi(S_{h-1,i})] + dN(h+1) + aN(h) \ ,$$

where *aN(h)* stands for unknown number of *acks* from routers being advertised to join the session, and *dN(h+1)* - for expected *acks* from the rest of the MCT branch. The addenda of the above sum should refer to three *different rates* of the MCT development as viewed from the current MCA. The first addendum refers to the greatest rate, while the second - to the slowest development. The rate of *acks* for advertised routers is unknown and depends on their number, structure and contract parameters. Even when the multicast session is planned beforehand there is no hope that all routers will have exact knowledge of the MCT being part of the contract. That is why among these new receivers could be those dedicated to join the session via another splitting point.

Figure 6 shows a typical picture of the MCT development; dashed line depicts a distributed development process which takes place in reality, however the knowledge about the progress of development is shared among unknown number of MCA; bold line depicts *acks* receiving process for the initial MCA if it goes as planned. Thick grey line shows another possible arrival process, when splitting point migrates from receiver 3 to receiver 2.

Let us now examine how this performance could be improved with additional messaging to the initial MCA. According to Table 2 the most critical message to be forwarded backwards to the initial MCA is the negative acknowledgement (*nack*) from a dedicated receiver; this nack could be generated after expiry of the timeout by appropriate MCA in the nearest splitting point. However, in order to assure that this receiver is unreachable, the MCA has to wait for the whole hierarchy of timeout periods spanning its branch. This long delay prevents from early detection of failed receiver within the totally distributed scheme of the MCT development. From the other hand, if the *nack* will be sent after a single timeout it may confuse the initial MCA in case when this failed receiver has joined the MCT via another MCA (receiver migration). Also if *all acks* and *nacks* will be sent to the initial MCA this will cause a high risk of MCA flooding by these messages and, again, may cause conflicts in detection of the MCT state. Here we propose another solution: every MCA reports *acks* on new receivers to initial MCA only in case if this receiver is dedicated one but joining the MCT not in planned manner.

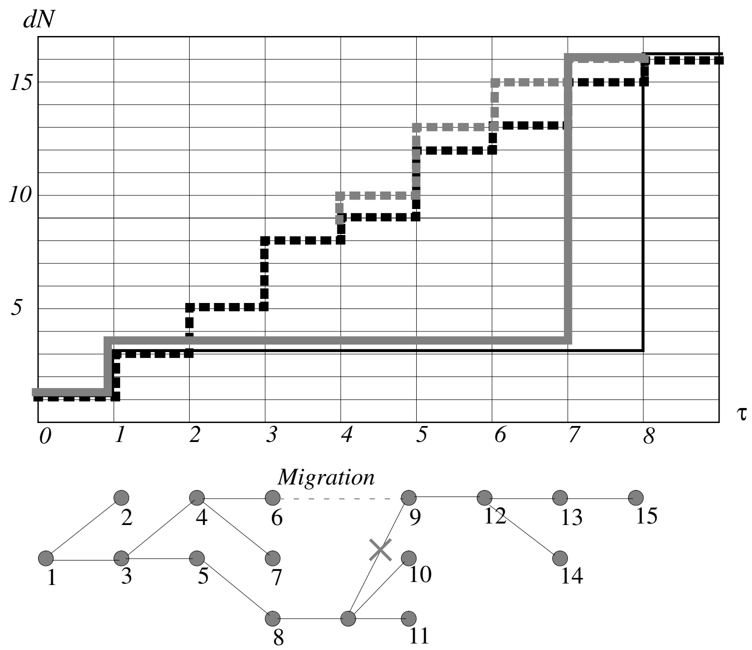

Fig. 6 MCT Development and acknowledgement process

Other performance issues include the following.

*Support for the Multidomain Networks***.** When the multicast Call traverses several domains (where domain could be represented by a single service provider and/or network management) then at the border node of a new domain the MCA has to invoke it's SP method to renegotiate all needed parameters and services (media, etc.). At the same time the whole new domain is visible to the source domain as a single receiver, that's why the border node within the sender domain has to invoke the MRA method and to communicate with the Src method of the MCA copy inside a new domain.

Sender Substitution. One of the main advantages of the proposed scheme is that any network node which belongs to the MCT possesses the embryonic MCA, that is why any node can invoke needed interface, thus enabling a new active sender or joining the current MCT by a new receiver, etc. We assume that when the SP method starts negotiation with the service provider it sends a multicast session request message with the list of dedicated receivers as it's parameter; that means for the service provider that any member of the d-nodes list can become a sender. As an option we can also use another parameter, say, maximal number of simultaneously active senders.

Quality of Service. The default quality of service with regard to each media mentioned in the contract with the service provider is to be guaranteed by the service providers in each domain. However, due to different hardware equipment, applications and types of host connection to the internetwork actual receivers may have a willingness to change the default QoS parameters. This can be done by the filter interface of the MRA object for every pair (MRA, Receiving Host).

Reliability. This feature could be achieved at the network level with the use of the same technology as proposed in [21] for a single application. However, reliable integration of applications could be achieved only with application independent reliability, i.e. within the underlying transport service including multicast.

4. Implementation Remarks

The technical availability of the migrating multicast agent could be explained within the Common Object Request Broker Architecture (CORBA) specification as shown in Figure 7. Two disjoint Object Request Brokers (ORB) can exchange any objects via the

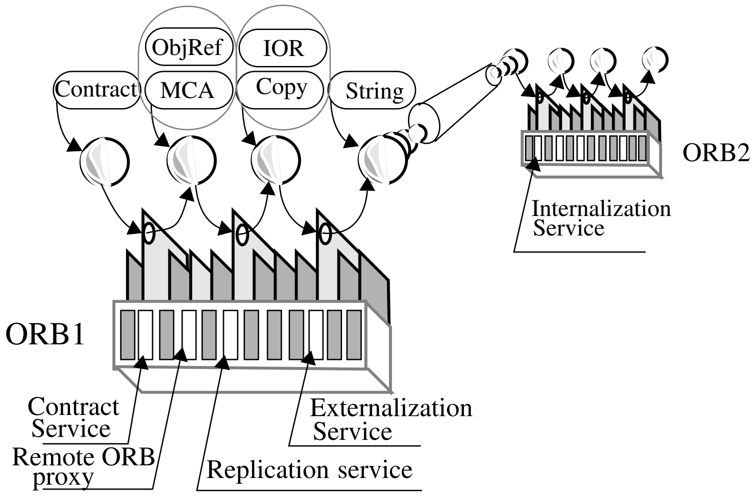

Fig. 7 MCA migration in CORBA environment

internetwork with the use of Replication and Externalization/Internalization Services of CORBA 2.0. The Externalization Service assumes that the Stream Service object holds externalized objects as files. To solve the problem of unique object references the CORBA suggests [17] that the Interoperable Object Reference (IOR) should be externalized together with the externalization of the copy object. Consequently, two proxies are to be installed within the naming services of two collaborating ORBs. This paper suggests that a special Contract Service should be considered within CORBA specification set, and, additionally, that no reference should be created for a copy MCA, except the IOR. Moreover, locking of the destination suggested in [15] actually locks the copy MCA as well, preventing from confusion with multiple objects within single ORB.

The CORBA specification was chosen for the following reasons:

- CORBA implementations provide efficient interoperability across heterogeneous software platforms;
- being a software development platform CORBA has an application viewpoint;
- CORBA deployment is rather fast in the direction for a cross-compatibility with ODP (TINA) [18]; OpenDoc, OLE [19] and CAD's Portable Common Tool Environment [20];
- adopted UNO approach [17] enables flexible distributed systems architecture.

The following implementation approaches seem to be possible:

- all nodes of the internetwork could be equipped in advance with customizable copies of the MCA to save network resources needed to retransmit and self extract the MCA;
- MCA distribution at the Call Establishment phase could be made as proposed without any further exchange of information between the entities of MCA;
- providing for MCA distribution with some information exchange between these peer entities.

Future research in this area will consider quantitative scalability evaluation via different modelling approaches, as well as functional extensions of the presented framework.

9. Conclusion

This paper aimed to demonstrate that the *agent paradigm* could be extremely successful not only in the area of Artificial Intelligence and personal communications but in for a collaborative multicast call establishment as well.

The *multicast connectivity on demand* is a big challenge for telecommunication service providers. However, existing approaches for the call establishment don't fit special requirements of the multicast session establishment, or, when modelled, appear to be enormously complicated.

Presented approach tries to benefit from the logical separation of three major players in the area: collaborative applications, producing and consuming multimedia multicast flows; the internetwork with its structure and best effort philosophy, and, finally, the mediator of successful collaboration - the multicast agents dynamic association.

If there could be any intelligence in setting up a call between the sender and the set of receivers, then the multicast agent brings this intelligence to a number of positions nearest to receivers applications. This MCA placement enables to receivers effective QoS negotiation, error recovery, group dynamics and management.

Performance aspects of the proposed approach are to be refined in more details as well as the multicast agent implementability within existing platforms.

References

1. Heijenk X.H., Niemegeers I.G. *Communication Systems Supporting Multimedia Multi-user Applications.* - IEEE Network, January/February 1994, pp.34 - 44
2. Minzer S., Bussey H., Porter R., Ratta G. *Evolutionary Trends in Call Control.* - Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 300-304
3. Maastricht van C., Schalk E. *Call modelling in a Broadband IN architecture*. - In. Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 340 - 344
4. Grebenö J., Hanberger N., Ohlman B. *Experiences from implementing an Object Oriented Call Model.* - Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 77 - 81
5. Deering S., Estrin D., Farinacci D., Jacobson V., Liu Ch.-G., Wei L. *An Architecture for Wide-Area Multicast Routing,* In: Proc. SIGCOMM 94 -8/94, pp.126 - 135
6. Murata S., Shionozaki A., Tokoro M. *A Network Architecture for Reliable*

Process Group Communication. - In Proc. 14th Int. Conf. on Distributed Computing Systems, June, 1994, Poznan, PL, ICDCS'94 (Piscataway, N.J.: IEEE), pp. 66 - 73

7. Covaci S. *Towards the Global Provision and Management of a New Open Bearer Service*, In: Proceedings of the First International Workshop on High Speed networks and Open Distributed Platforms (HSN&ODPl), June, 1995, St.Petersburg, Russia, 16 p.
8. Handley M., Wakeman I. *CCCP: Conference Control Channel Protocol. A scalable base for building conference control applications*. v.1.4. An html link was provided at http://www.research.att.com/cgi-bin/bibsearch_html?CCCP
9. Raiser M. *Performance Evaluation of Data Communication Systems.* - In Proc. IEEE v.70, No 2, February, 1982, pp.171-196
10. Szyperski C., Ventre G. *Efficient Multicasting for Interactive Multimedia Applications*, TR-93-017, International Computer Science Institute, Berkeley.
11. Smirnov M.I. *Multicast Routing: State of the Art and Research Ideas*. Presentation at GMD Fokus, STEP group, March, 1995; URL ftp://ftp.fokus.gmd.de/pub/step/papers/Smir9503:MCSofA.ps.gz
12. Deering S., Estrin D., Farinacci D., Jakobsen V. *IGMP Router Extensions for Routing to Sparse Multicast-Groups*, Internet Draft, available e.g. at http://karin.elch.lu.se/pub/stan... ft-ietf-idmr-igmp-sparce-00.txt
13. Smirnov M.I. *Efficient Multicast Routing in High Speed Networks.* - In: Proceedings of the First International Workshop on High Speed networks and Open Distributed Platforms (HSN&ODPl), June, 1995, St.Petersburg, Russia, 17 p.
14. Prabhat A. K., Gretzinger M.R. *Distributed Object-Oriented Data-Systems Design*, Prentice Hall.
15. Smirnov M.I. *Object-Oriented Framework for the Multicast Applications Integration*. - to appear in Proceedings of the DOOC'95 Workshop, Object World Conference, Frankfurt/Main, 10-11, October, 1995, 12 p.
16. Casner S. *The MBONE Frequently Asked Questions,* at: http://www.research.att.com/mbone_faq.html
17. ORB 2.0 RFP Submission "*Universal Networked Objects*", Sept.,1994, OMG TC Document 94.9.32
18. *Comparison of the OMG and ISO/CCITT Object Models. The Report of the Joint NM Forum/OMG Task force on Object Modelling*, OMG Document 94-12-30, 27 p.
19. Adler R.M., *Emerging Standards for Component Software*. - In Computer, March, 1995, vol.28, No. 3, p. 68-77
20. Start K, Patel A. *The Distribution Management of Service Software*. - Computer Standards and Interfaces, 17(1995):291-301
21. Floyd S.,Jacobson V., McCanne S., Liu Ch.-G.,Zhang L. *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*. - to appear in Proc. SIGCOMM, Cambridge, Mass., Sept.1995