

A Framework for Group Integrity Management in Multimedia Multicasting

Andreas Meissner¹, Lars Wolf^{1,2}, Wolfgang Schönfeld^{1,3}, Ralf Steinmetz^{1,3}
{Andreas.Meissner, Wolfgang.Schoenfeld, Ralf.Steinmetz}@gmd.de, Lars.Wolf@uni-karlsruhe.de

¹ GMD - German National Research Center for Information Technology, Institute IPSI,
Dolivostrasse 15, 64293 Darmstadt, Germany

² University of Karlsruhe, Zirkel 2, 76128 Karlsruhe, Germany

³ Darmstadt Technical University, KOM, Merckstr. 25, 64283 Darmstadt, Germany

Abstract

Multicast research has so far been focused on routing and network-level group management. Conditions on the composition of multicast groups have however been kept simple, with little efforts to specify requirements in terms of membership, member roles and group organization. Integrity conditions on multicast groups have been largely neglected despite the fact that it is often desirable to express who should be admissible as a member, with what role and privileges, what relation a group should have to other groups, etc. Furthermore, the traditional multicasting model has been flat, with no finer granularity than a group, and without inter-group relationships. In this paper, we address both issues that we see as shortcomings of current models. We introduce a framework that allows us to sub-divide multicast groups into subgroups, e.g. for low and high quality versions of a media stream, and, on the other hand, to form and manage meta groups from groups, thus integrating "multi" media groups. On all three levels, our framework provides for specification of various integrity conditions as part of a comprehensive policy framework, including integrity on state and state transition as well as action and transition policies for group management.

1 Introduction

Multicasting has received tremendous research attention in recent years. With the advent of high-bandwidth multimedia services for the mass market, it is becoming increasingly evident that establishing a large number of unicast sessions in parallel will waste bandwidth and impose unacceptable stress on network resources. Multicasting is seen as a solution to this problem, and thus much work has been done on multicast routing as well as group management. Existing multicast group management approaches are however often limited

to simple user-initiated join and leave operations and allow only decisions based on network-level conditions. Application interaction is then required in order to consider further criteria [18]. More comprehensive group management systems, on the other hand, do not adequately consider specific *multimedia* requirements. The work done in the context of *video conferencing* systems [2] [3] does provide more semantically rich criteria, such as floor control mechanisms, but is not easily generalized to other multicasting applications.

We observe that no comprehensive framework has yet been presented that supports a rich set of group integrity conditions for multimedia multicasting. Our work, carried out in the context of the *GCAP project* [6], aims at providing such a framework. In order to support multimedia multicasting, we first introduce three hierarchy levels: Separate multicast *groups* are established for each media type, such as one group for video and another group for audio. These groups are themselves subdivided into *subgroups* for actual data transfer. This allows us to conceptually integrate different coding formats (e.g. audio streams of different quality levels) into one group. Assuming that the data transmitted in several subgroups of a group is often semantically identical, we suggest a relation "*can be generated out of*" between subgroups, so *transformers* can be introduced that feed one subgroup with re-coded data obtained from another subgroup. Having introduced *subgroups* and *groups*, we define the highest (i.e. third) hierarchy level, *meta groups*, formed by inter-related "multi" media groups.

A framework for specifying *integrity conditions* on all three levels constitutes the core of our work. By integrity conditions, we mean requirements imposed on subgroups, groups and meta groups describing valid *states* and *state transitions* with regard to membership set, member roles, organization and topology, e.g. a minimum number of members in a subgroup or the requirement that users join the control group whenever they join any multimedia

group. Furthermore, we suggest integrity conditions on *traffic*. In addition to these conditions, our set of policies includes *action policies*, specifying how a group manager may re-establish integrity in case it has been found violated, and *transition policies*, stating how to handle potentially conflicting requests by members, e.g. from members who want to start acting as a sender.

Paper Outline: After this introduction and an overview of related work in section 2, we introduce the monomedia part of our integrity framework in section 3, including the concepts of subgroups and transformers. Section 4 extends this approach to the multimedia case with meta groups. Section 5 summarizes and mentions future work.

2 Related Work

Group communication systems (GCS) and their properties are formally analyzed and defined in [1] and, in a survey of GCS specifications, in [17], using the concept of I/O automata [8] and discussing *safety* and *liveness* properties. [16] presents a distributed generic membership algorithm for structured, cooperative groups and discusses how data exchange may be synchronized with the evolution of group membership over time. A cooperative group is formed by members, called *agents*, organized in a *cooperation graph* expressing their relation. It is decided at the application layer what rules or predicates a graph has to fulfill in order to be considered valid. Hence, *integrity* refers to synchronizing data exchange and group membership changes as well as to choosing valid subgraphs of the cooperation graph. [16] does not provide a framework for expressing such integrity conditions.

Group management has been studied not only in the multicast research domain [10] [5], but also in the context of distributed systems and computer supported cooperative work (CSCW).

For *distributed systems*, such as distributed operating or file systems and distributed databases, the focus is traditionally on groups of processes that communicate at the operating system level (Amoeba [7], V Kernel [4]). The primary objectives related to group integrity are fault tolerance, consistency of message delivery among group members, and provision of measures to ensure that a common view on group membership and data is maintained among group members.

CSCW systems are concerned with organizing work groups and data flow between members of such a work group. There are systems for *asynchronous* collaboration (i.e., the members do not have to process group data simultaneously) and for *synchronous* collaboration, such as video conferences with simultaneous presence and data processing by all participants [12] [2]. Floor control and other group management functions in CSCW systems are generally performed at the application layer, group members being human users. Group integrity thus refers to

conditions on the composition of the set of human participants and their individual rights and roles.

For multicasting, [9] suggests a group communication framework and introduces the concepts of *group association* and *active group*: members of a group need to *register* in order to be bound to the group address. Registered members form the *registered group*. For the purpose of data transfer, a *group association* is established between a subset of group members. There are *active* and *passive* participants in a group association, with active participants forming the *active group*. An active group may be *static* or *dynamic* (membership changes allowed during the association's lifetime), *indeterminate* or *determinate* (membership known by participants), and *open* or *closed* (only registered members are permitted). *Active group integrity (AGI)* conditions are imposed on the active group membership of an association, such as a minimum number of participants or a set of mandatory active participants. A *multicast conversation (MC)* within an association is an instance of communication, e.g. with one sender (*master*) and several receivers. *Association topology integrity (ATI)* conditions include both AGI conditions on individual conversations (now referred to as *MC-AGI*) and conditions on the whole group association, such as a maximum number of concurrent conversations within an association. In our view, the framework in [9] is a valuable contribution setting the path for multicast group integrity research, but it neither addresses the multimedia context, nor does it specify how integrity conditions going beyond member identities, member counts and topology might be expressed or even enforced.

Group management in *XTP 4.0* [18] is based on receiver lists and allows, by keeping track of all receivers, reliable multicasting. Thus, if full reliability is required for all group members, a failing member destroys the group's integrity. XTP assumes that sufficient knowledge for a proper reaction, as for any decisions on valid group composition, is available only at the application layer and therefore makes no effort to provide such functionality itself. In particular, there is no framework for group integrity in XTP. [14] employs the AGI concept of [9] on top of XTP and distinguishes between group *establishment conditions* and *communication integrity conditions*.

A comprehensive overview of current multicasting research is provided in [15] and, with focus on multimedia, in [13].

3 Groups and Integrity in Monomedia Multicasting

By *monomedia multicasting*, we mean multicasting of one type of media, such as audio, from any number of senders to any number of receivers who are members of a multicast group, i.e. we do *not* require the simplifying assumption of a "1:n" topology. In this work, we are

dealing with *closed* groups, i.e. only members may participate in the communication. Groups are *determined*, thus all members are known, and *dynamic*, so membership may change during the groups' lifetime. We denote groups by G_1, G_2, \dots . Groups are controlled by a central *group manager*, responsible for managing group properties and members, their roles and rights [10], as well as controlling and enforcing group *integrity* conditions according to a set of *policies*. The manager of group G_i is denoted $M(G_i)$.

Data sent by a member is called a data *stream*. Note that, despite this term, we do not limit the applicability of the framework in this paper to continuous media transmissions.

3.1 Subgroups

In our framework, a group consists of zero or more *subgroups*, all controlled by the same group manager. If a group G_i has n subgroups, they are denoted $G_i^0, G_i^1, \dots, G_i^{n-1}$ and by definition $M(G_i) \equiv M(G_i^j)$ for $j=0..n-1$. The concept of subgroups allows us to conceptually integrate into one group several formats used for transmission of one media type. As an example, in figure 1, for multicasting audio, there might be a high quality subgroup (☆☆☆), preferred for receivers who have high bandwidth available, a medium quality (☆☆) and a low quality (☆) subgroup, preferred for low-bandwidth receivers. Unlike some *layered multicasting* schemes, we do not assume that a user has to listen to several subgroups in order to construct a meaningful stream. As with the concept of receiver oriented congestion control [15], a member of a high-bandwidth subgroup may switch over to a lower-bandwidth subgroup if he experiences excessive loss.

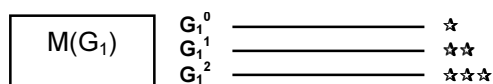


Figure 1 - Subgroups

We say that a subgroup G_i^j *mirrors* a subgroup G_i^k if the data transmitted in G_i^j is, apart from format differences, semantically the same as the G_i^k data. (The exact definition of “mirroring” has to be made individually for each pair of formats; e.g. it has to be decided if a sequence of still images is considered semantically “the same” as a 25 fps video of the same scene.) A subgroup G_i^j *super-mirrors* a subgroup G_i^k if G_i^j mirrors G_i^k and is allowed to include *additional* data.

Transfer of user data traffic always occurs in a subgroup, while group management control messages are conceptually exchanged in the group. Subgroups can be created and destroyed by the group manager on request by authorized group members according to group policy.

3.2 Users, Members, and Roles

We use the term *user* for a communication entity that is a *member* of zero or more subgroups; after a user U_k has successfully joined G_i^j , U_k is called a member of G_i^j and G_i . (If U_k joins G_i^j , he automatically joins G_i .) Each user has a component called *user controller* that is ready to interact with a group manager and locally enforces any instructions received by the manager of any group the user is a member of. A member U_k may have different *roles*: If U_k sends data, he is called a data *source*, if U_k receives data, he is a *sink* (figure 2). Note that U_k may be a source in one or more subgroups and a sink in one or more (possibly other) subgroups at the same time. If U_k is a sink (source) in any G_i^j , he is called a sink (source) in G_i . U_k may be a *key member* of a group (or subgroup) if he is in some respect considered important, as we will see later when we discuss integrity conditions (e.g. a multicast lecture might not start before a VIP has arrived/joined). Key members have no special privileges towards the group manager, while *senior members* do: They may, for example, request that another member be removed from the group or subgroup, or they may change a policy.¹

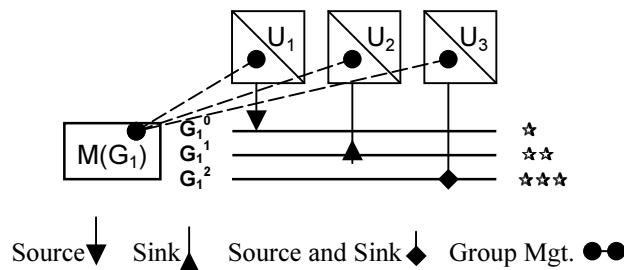


Figure 2 - Users, Members, and Roles

Members can change their roles during the lifetime of the group if permitted or instructed to do so by the group manager. For example, the manager can revoke a member the source role. During the join process, users authenticate themselves and provide the manager with their parameters by way of their user controller. According to these parameters, along with further information received from a trusted user directory service, and to the policies set forth for the group and the relevant subgroups, the manager admits or rejects the join after negotiation. Users unable to authenticate themselves are rejected or, according to (sub-) group policy, admitted only on a limited-rights basis. For example, through data encryption, unauthenticated users may be excluded from sensible data exchange.

We use the following notation:

U : set of users $\{U_1, \dots, U_m\}$; m is the total number of users
 $P(G_i^j)$: Set of members of subgroup G_i^j ; $K(G_i^j)$, $S(G_i^j)$, $\text{sink}(G_i^j)$, $\text{source}(G_i^j)$ are the set of key members, senior

¹ It is actually more flexible to assign *attributes* to members specifying their individual properties, right and duties. We will however use *key* and *senior* member as generic terms in this paper.

members, sinks, and sources, respectively. $P(G_i)$ etc. accordingly for groups.

Note that, with n subgroups in G_i : $P(G_i) = \cup_{j=0..n-1} P(G_i^j)$.

In this paper, we assume that a user knows how to find and contact the manager for the (sub-)group he wants to join. Note that, in our concept, groups and subgroups are *not* identical to the set of members (i.e. $G_i \neq P(G_i)$), as we will see later, when we further characterize them by a set of policies.

3.3 Transformers

If group policy states that all subgroups mirror each other, we identify two alternatives for feeding data into such a group's subgroups:

First, any user who is a source for one subgroup must also act as a source for all other subgroups. (This imposes restrictions on what subgroups are possible and on who can be a source.)

Second, a source is mandated to feed a subgroup with a data format out of which all other subgroups' data formats can be computed. (If it cannot do that, the group manager denies it the source role.) The job of transforming one format into another is accomplished by special members called *transformers*. A transformer that generates the data format of G_i^j out of the format of G_i^k is denoted $T(G_i^k, G_i^j)$. Note that if a source for G_i^k acts itself as $T(G_i^k, G_i^j)$ for all $j \neq k$, we are in the first alternative. $T(G_i^k, G_i^j)$ may be located anywhere in the data tree for G_i^k ; it is a sink for G_i^k and a source for G_i^j . There may have to be several $T(G_i^k, G_i^j)$ in G_i if G_i^j is allowed to be partitioned and the transformer only serves a limited (geographical) region (figure 3). In that way, it is not necessary to multicast in parallel the data for all subgroups $G_i^{j \neq k}$ over long distances if there are clusters of sinks for G_i^j only in some regions. $M(G_i)$ is responsible to admit members only if the subgroup they want to join (as sink) can be made available in their region, e.g. by activating a proper transformer. Transformers may be identical to active intermediate systems in the data tree and join subgroups for the sole purpose of performing the transformation work. Transformers are thus a generalization of *transcoders*.

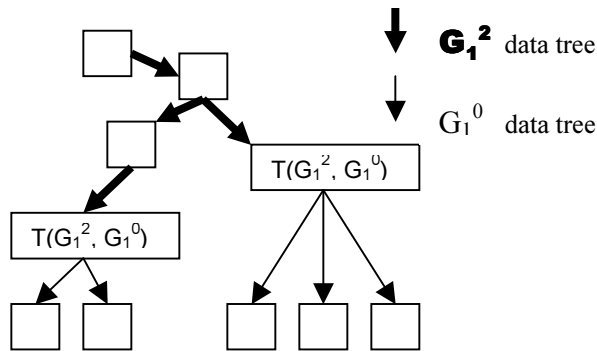


Figure 3 - Transformers, Partitioned Subgroup

While subgroups are not conceptually required to be in any way *ordered* within a group, transformers allow us to introduce a *coding hierarchy*. If we assume, for example, that the data for the media type in a group can be scaled along one or more dimensions, such as sampling rate and error robustness in the audio context, we introduce a relation "*can be generated out of*" among the subgroups. We write $G_i^j \rightarrow G_i^k$ if it is possible to transform the data format in G_i^k into the data format in G_i^j . Note that $G_i^j \rightarrow G_i^k$ does not imply that a transformer $T(G_i^k, G_i^j)$ is actually operational or even available in G_i . We will come back to such dependencies when we discuss group integrity.

Transformers may also *aggregate* the data streams from different sources in G_i^k into *one* data stream in G_i^j and thereby act as *mixers* (e.g. audio mixers or video overlay generators). We denote a mixing transformer as $T^m(G_i^k, G_i^j)$, as opposed to $T(G_i^k, G_i^j)$ which means that multiple input streams in G_i^k are transformed into *multiple* output streams in G_i^j [11]. For a mixer $T^m(G_i^k, G_i^j)$, we do not require $j \neq k$. Furthermore, it is possible to extend the concept of mixers to the form $T^m(G_i^*, G_i^j)$, where G_i^* is a set of subgroups of G_i , thus enabling a mixer to feed a "comprehensive" subgroup G_i^j from the data in other subgroups.

Up to this point, we have not considered transformers $T(G_a^k, G_b^j)$ with $a \neq b$. As we will see later, this restriction allows us to assume that the impact of *traffic* integrity failures in G_a^k will be contained, at most, to G_a . However, in some cases, there might be a relation $G_b^j \rightarrow G_a^k$, $a \neq b$. (We will see an example in section 4.2.) If there are no $(i,l) \neq (a,k)$ with $G_b^j \rightarrow G_i^l$, or if no other transformers capable of feeding G_b^j exist, a traffic integrity violation in G_a^k will transitively cause a violation in another subgroup G_b^j , and therefore another group G_b .

3.4 Subgroup Policies

Subgroup policies describe rules according to which valid subgroup states and state transitions can be determined, and actions the group manager takes to ensure subgroup integrity. In our (discrete) model, a subgroup advances from one state to the next. A state is here described by membership set, member roles, and topology. For example, leaving out topology for the moment, state i might be described as "Sources: U_1 . Sinks: U_2, U_3 ", and state $i+1$ as "Sources: U_2 . Sinks: U_3, U_4 ". We impose integrity conditions on *states* and on *state transitions*. For a subgroup G_j^k , a sample integrity condition on state is

condition S1: $| \text{source}(G_j^k) | = 1$ and $| \text{sink}(G_j^k) | \geq 2$,
i.e. at all times, there must be exactly one source and at least two sinks in G_j^k .

If we denote the set of members of G_j^k during state i as $P(G_j^k)$, an integrity condition on state transitions might be:

condition S2: $| P(G_j^k) \cap P(G_j^k)^{i+1} | > 1$

in order to express some limit on member fluctuation.

In the above example, condition S1 is met for both state i and state $i+1$, and the transition from state i to state $i+1$ fulfills condition S2.

It is the responsibility of the group manager to control and maintain integrity within each subgroup. For this purpose, it accesses policies relevant to the subgroup from a (dynamic) *policy repository*. Before advancing from one state to the next, the group manager checks if all integrity conditions on state and all integrity conditions on state transitions would be met. If so, it advances the subgroup to the next state. (When exactly such a transition is done, i.e. when the group manager proceeds from collecting requests to evaluating them, has to be specified [16]; we are not discussing this here.) In the above example, $M(G_j^k)$ would have received, during state i , the following requests:

from U_1 : leave subgroup G_j^k

from U_2 : switch role from sink to source in G_j^k

from U_4 : join subgroup G_j^k as sink

Having checked conditions S1 and S2, it determined that, combined, these requests would not violate the conditions, and therefore it committed them.

In addition to *membership and role related integrity conditions*, as introduced so far, there are *topology related integrity conditions* on subgroup states. Such conditions may limit the geographical scope of the subgroup.

Extending the above example, if U_4 had for some reason failed to complete the requested join, state $i+1$ would violate condition S1 due to the number of sinks. In such a case, the group manager has to know how to act in order to re-establish integrity. Therefore, group policies additionally include *action policies* according to which the group manager reacts on observed integrity failures. A simple example for such a policy is to perform a "roll-back" to the previous state. We could also require the subgroup to be suspended until the join (as sink) of U_4 or another user is completed.

In our concept, subgroup integrity can be violated by unexpected events, such as members failing to complete a join or failing to perform the role (e.g. source) they are supposed to perform. The group manager detects these events and, according to subgroup action policies, takes action to re-establish integrity by advancing to a new state in which integrity conditions are met. It is thus well possible to have such "emergency" state transitions.

Another type of subgroup policies are integrity conditions on *traffic*. While integrity conditions on state and state transitions, as discussed so far, are *discrete*, traffic integrity conditions are *continuous*, i.e. they have to be evaluated continuously and not only on the occasion of state transitions or events in the sense introduced in the previous paragraph. Integrity conditions on traffic specify what requirements the data traffic in the subgroup has to meet. This includes adherence to the data format specified for the subgroup (no source may produce an illegal audio stream, for example), conditions on quality of service or

acceptable error rate, and conditions on security, such as mandatory encryption by all sources. If the group manager determines a violation of traffic integrity conditions, it takes action according to action policies. Such action policies might state that the group manager has to revoke the source role to a user who continuously produces illegal data. If an excessive error rate is observed, an action policy might require that the number of sinks in the subgroup be reduced. The group manager may outsource the job of monitoring traffic according to traffic integrity conditions to trusted members, referred to as *agents*, or act only if notified by a quorum of members about a problem.

Note that we generally assume that user controllers always successfully enforce group manager orders locally, that no unauthorized operations are performed, and that non-members (e.g. users forcefully removed from the subgroup) cannot disturb subgroup communication.

Subgroup policies may be changed during the lifetime of the subgroup by authorized *senior* members, by way of the group manager. (E.g., during the establishment phase of a subgroup, integrity conditions could be less restrictive in order to allow the new subgroup to develop and build up its membership.) Such changes in the policy repository trigger a re-evaluation of the current state according to the new set of integrity conditions. If a violation has occurred, the group manager takes action, as described above.

Now further extending the above example, we assume that U_4 successfully completed the join as a sink, so that state $i+1$ fulfills the (only) state integrity condition S1. We define a set $K = \{U_2, U_4\}$ and introduce a new

condition S3: $P(G_j^k) \supseteq K$,

i.e. we demand that (at least) key members U_2 and U_4 have joined the subgroup. During the re-evaluation, this new condition is determined to be met during state $i+1$, so no action is required by $M(G_j^k)$. However, the question arises what to do if U_4 later wants to leave G_j^k again. We observe that introducing a new integrity condition on *state* may require a new *policy* for state *transitions* (unless some default is to be used), i.e. a procedure stating how to handle requests that would result in violation of this condition in the subsequent state. We call this type of subgroup policies *transition policies*, not to be confused with (descriptive) integrity conditions on state transitions.

Generally, in case integrity cannot be re-established (after enforcement measures according to action policies have failed), the subgroup must be either suspended or terminated [9]. Suspension means that data transfer is halted as long as the integrity conditions cannot be met. A suspended subgroup is still subject to integrity verification. Hence, the group manager can determine when the violation has been overcome (e.g., after a join request, the minimum number of members can be reached again) and thus revoke the suspension. If a subgroup is terminated and members cannot switch to another subgroup, they have to leave the group.

3.5 Group Policies

The policy framework introduced for subgroups is imposed on groups accordingly. The set of group policies conceptually includes the policies of all subgroups. In this section, however, we address group policies referring to integrity conditions and actions applied *across subgroups*, i.e. defining the inter-relation of a group's subgroups.

We distinguish between integrity conditions on group *state* and on group *state transitions*. Both are discrete and include *membership and role* related as well as *topology* and *group organization* related conditions. The former define how members may be distributed among the subgroups and what roles they may have. For example, for group G_j , with n subgroups G_j^0, \dots, G_j^{n-1} , there might be a

condition G1: $P(G_j^h) \cap P(G_j^k) = \emptyset \forall h \neq k$
with $h, k \in \{0, \dots, n-1\}$

i.e. no user may be a member of more than one subgroup at a time², and (assuming condition S1 is in place for all subgroups, guaranteeing that there are no subgroups without members):

condition G2: $|P(G_j^h)| / |P(G_j^k)| \in [0.5, 2] \forall h \neq k$
with $h, k \in \{0, \dots, n-1\}$

so that the number of members be somewhat balanced among the subgroups.

Group organization refers to requirements on what subgroups, with what properties, have to exist within the group. A group state integrity condition on both topology and group organization is:

condition G3: $\exists k \in \{0, \dots, n-1\}: G_j^k$ non-partitioned and $G_j^h \rightarrow G_j^k$ and $\exists T(G_j^k, G_j^h) \forall h \neq k, h \in \{0, \dots, n-1\}$ stating that there be at least one non-partitioned subgroup out of which all other subgroups can be generated and that a proper transformer exist. Furthermore, another group organization integrity condition might require that no two subgroups with identical data formats exist.

Integrity conditions on *state transitions* describe how groups may advance from one state to the next, again with regard to *membership and role* as well as *topology* and *group organization*. For example,

condition G4: $\nexists l, h: U_l \in {}^{i-1}P(G_j^h)$ and $U_l \in {}^iP(G_j^{k \neq h})$ and $U_l \in {}^{i+1}P(G_j^h)$

would, along with condition G1, prohibit oscillating behavior, i.e. a user switching back and forth between the same subgroups.

Even though, by definition, user data traffic is exchanged in subgroups, continuous integrity conditions on *traffic* have to be evaluated at group level. If continuous media is multicast in different formats or qualities in subgroups of a group G_j that thus mirror each other, $M(G_j)$ has to monitor if they remain in sync. Moreover, a violation of traffic integrity in one subgroup G_j^k can affect traffic integrity in another subgroup G_j^h if

G_j^h is fed by a transformer $T(G_j^k, G_j^h)$ and there is no other subgroup G_j^c for which $G_j^h \rightarrow G_j^c$ with operational transformer $T(G_j^c, G_j^h)$.

Action policies on group level specify how the group manager has to re-establish group level integrity. With above condition G2, if failure of a number of members has resulted in unbalanced subgroup membership numbers, $M(G_j)$ may, for example, react by forcefully switching remaining members between subgroups. If, however, conflicting request are received by $M(G_j)$ from members who intend to switch subgroups, the manager would decide according to group-level *transition policies*.

Group policies can only be changed by users who have senior member status on group level. Such members may also be allowed to create or delete subgroups.

Generally, if the integrity of any subgroup is determined to be violated, the same is true for the group. The group manager can re-establish integrity for the group by re-establishing integrity for the affected subgroup, or by suspending or terminating it. If the subgroup's integrity cannot be re-established and the non-suspended presence of the subgroup is required by group organization integrity conditions, the entire group has to be suspended or terminated. Otherwise, suspended subgroups with violated integrity do not constitute a group integrity violation.

4 Meta Groups and Integrity in Multimedia Multicasting

For multimedia multicasting, we extend the scope to two or more semantically related types of media transmitted at the same time.

4.1 Meta Groups

In the multimedia case, we are dealing with two or more multicast groups, each with the properties introduced above. (There may be more than one group per media type: For instance, during a video conference an object is discussed, and the video showing that object is transmitted in a separate group, i.e. not in the same group as the video showing the participants.) All these groups together form a *meta group*. In particular, each group has its own group manager. In order to coordinate and control them, a *meta group manager* is established, denoted $MM(G)$ if, referring to the example in figure 4, G is the meta group formed by G_1, \dots, G_4 .

A meta group manager is kept up-to-date by the group managers regarding the membership and roles as well as organizational and topological properties of their groups. It can mandate them to request permission for any group management operations they intend to perform on their groups, and it can order them to perform operations.

² We have to relax this condition if transformers are to be deployed

4.2 Meta Group Policies

The meta group manager controls meta (i.e. inter-group) integrity by applying meta group integrity conditions set forth in a policy set with *state* and *state transition* integrity conditions, *traffic* integrity conditions, *action policies* and *transition policies* on meta group level. For meta group integrity considerations, we demand that it be only relevant to know what *group* a user is a member of, but not what *subgroup*. Hence, organizational integrity conditions have to be in place that ensure meta group integrity without referring to subgroup membership. (Note, though, that, as we will see in the following example, *group managers* may well restrict subgroup membership in order to implement them.) These policies describe requirements that the groups within the meta group have to meet.

For example, we consider a video conference with four groups:

- G_1 for video, with subgroups G_1^0 for high quality and G_1^1 for low quality
 - G_2 for participants' voice audio, with subgroups G_2^0 for format A and G_2^1 for format B
 - G_3 for sign language video generated out of audio by a transformer, with just one subgroup G_3^0
 - G_4 for control, with just one subgroup G_4^0
- Together, they form meta group G.

The meta group manager MM(G) may enforce the following meta group level *state* integrity conditions:

condition M1: $P(G_4) \supseteq P(G_1) \cup P(G_2) \cup P(G_3)$

i.e. any member of G_1 , G_2 or G_3 must be a member of G_4 , too, and

condition M2: $\text{source}(G_1) \subseteq \text{source}(G_2)$

i.e. any source in G_1 must be a source in G_2 , too (so participants can hear the person they see), and

condition M3: $\text{sink}(G_2) \cap \text{sink}(G_3) = \emptyset$

i.e. no user may be a sink in both G_2 and G_3 at the same

time³, and, as an organizational integrity condition:

condition M4: $\exists j \in \{0, 1\}: G_3^0 \rightarrow G_2^j$ and $\exists T(G_2^j, G_3^0)$

i.e. we need a subgroup in G_2 out of which G_3^0 can be generated by an existing transformer.

These conditions M1-M4 are met in the state depicted in figure 4 with users U_1, U_2, U_3 and transformer $T(G_2^1, G_3^0)$. (Additional "dotted" users are, of course, required for a meaningful video conference scenario, but left out in the figure in order to keep it readable.)

The following is a sample integrity condition on *state transition*:

condition M5: $\forall k$ with $U_k \in {}^iP(G_1)$ forced to leave

$$\Rightarrow U_k \notin {}^{i+1}P(G_2) \cup {}^{i+1}P(G_3) \cup {}^{i+1}P(G_4)$$

i.e. any user who is forced to leave G_1 must leave $G_2, G_3,$ and G_4 , too.

A typical *traffic* integrity condition demands that the data in $G_1, G_2,$ and G_3 remain synchronized.

Note that MM(G) does not care how the G_2 members are distributed among subgroups G_2^0 and G_2^1 . If we now further assume that $G_3^0 \rightarrow G_2^1$, but *not* $G_3^0 \rightarrow G_2^0$, an organizational integrity condition is imposed on G_2 (and enforced by $M(G_2)$, not $MM(G)$) stating that G_2^1 *super-mirror* G_2^0 . It is then up to $M(G_2)$ to decide whether this be best accomplished by deploying a transformer $T(G_2^0, G_2^1)$ or by requiring that all sources in G_2^0 must also be sources in G_2^1 .

As can be seen from the above example, any transformer $T(G_a^k, G_b^j)$ with $a \neq b$ introduces additional complexity at the meta group management level. Thus, such inter-group dependencies are best avoided when a meta group is created.

At meta group level, *action policies* and *transition policies* state how to re-establish meta group integrity and how to handle *group manager* requests directed towards the meta group manager, respectively.

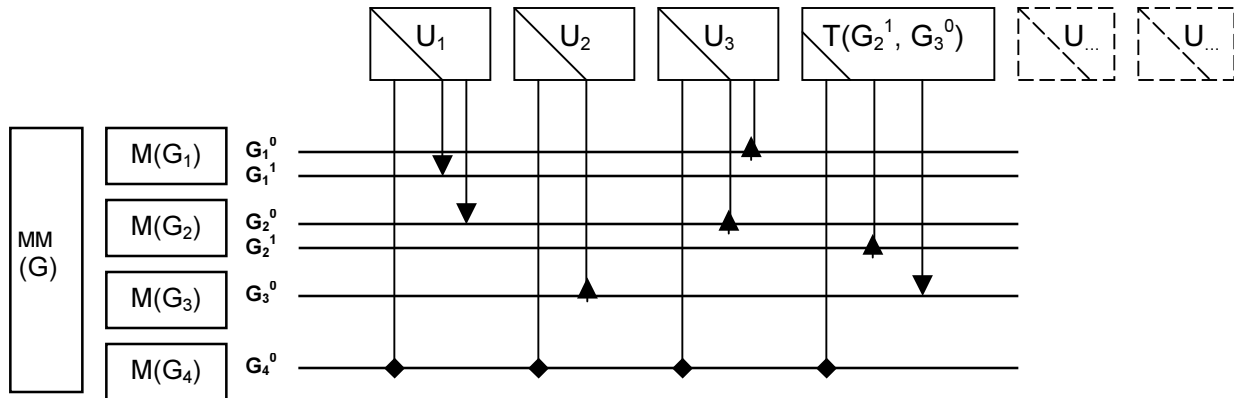


Figure 4 - Meta Group

³ An exception might be appropriate for a transformer for monitoring

5 Conclusions and Future Work

Having observed that current multicast models are not well suited for management of group integrity, we have suggested a new framework for group management in multimedia multicasting. Groups were decomposed into subgroups, allowing for different formats of one media type; inter-related groups were said to form a meta group. We introduced centralized group management, to be performed by group and meta group managers. Particular attention was devoted to integrity conditions imposed on subgroups, groups and meta groups, with regard to state, state transitions, and traffic. Additionally, we included action policies and transition policies in our policy set. It was shown that our framework is applicable to common problems in monomedia and multimedia multicasting.

Several extensions of this framework are possible and left for future work: *Fault tolerance*, especially in case of group manager failure, is an issue to be resolved. We have, up to this point, considered only centrally managed groups (and sub- and meta groups). However, in future work, we will drop this limitation and analyze *decentralized* scenarios, e.g. with peering members who vote in order to reach a group management decision by themselves. We have so far assumed that some *directory service* provides information on users to group managers. This service is subject to further specification. In some cases, a group should be permitted to belong to *several* meta groups. This raises new inter-group integrity issues since no single meta group manager is responsible for such a group, and conflicting control measures may be applied to that group by different meta group managers.

References

1. Y. Amir, D. Dolev, S. Kramer, D. Malki: Membership algorithms for multicast communication groups. 6th International Workshop on Distributed Algorithms, Springer, pp 292-312, November 1992
2. I. Beier, H. Koenig: GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management. Proc. of the 7th International Conference on Computer Communications and Networks ICCCN'98, Lafayette, USA, 1998, pp 594-598
3. S. T. Chanson, A. Hui, H. König, M Zühlke: Das OCTOPUS-Videokonferenzsystem, PIK 23 (2000) No. 4, pp 189-198 (in German)
4. D. R. Cheriton, W. Zwaenepol: Distributed Process Groups in the V-Kernel. ACM Transactions on Computer Systems, Vol. 3, No. 2, 1985, pp 77-107
5. C. Diot, W. Dabbous, J. Crowcroft: Multipoint Communications: A Survey of Protocols, Functions, and Mechanisms. IEEE Journal on Selected Areas in Communications, Vol. 15 (3), April 1997
6. GCAP IST Project: Global Communication Architecture and Protocols for new QoS Services over IPv6 Networks. <http://www.laas.fr/GCAP>
7. F.M. Kaashoek, A.S. Tanenbaum: An Evaluation of the Amoeba Group Communication System. Proceedings of the 16th International Conference on Distributed Computing Systems, pp 436-447, May 1996
8. N. Lynch, M. Tuttle: An Introduction to Input/Output automata. CWI-Quarterly (Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands), 2(3), pp 219-246, September 1989
9. L. Mathy, G. Leduc, O. Bonaventure, A. Danthine: A Group Communication Framework. Broadband Islands '94 Connecting with the End-User, W. Bauerfeld, O. Spaniol and F. Williams, eds., Elsevier North-Holland, 1994, pp 167-178
10. A. Mauthe, D. Hutchison, G. Coulson, S. Namuye: From Requirements to Services: A Study on Group Communication Support for Distributed Multimedia Systems. Technical Report MPG-95-10, Computing Department, Lancaster University, Lancaster, UK, 1995
11. A. Mauthe, G. Coulson, D. Hutchison, S. Namuye: Group Support in Multimedia Communications Systems. Proceedings of the 2nd COST 237 Workshop on Teleservices and Multimedia Communications, Copenhagen, Denmark, 1995
12. M. Nguyen, M. Schwartz: MCMP: A Transport/Session Level Distributed Protocol for Desktop Conference Setup. IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7, Sept. 1996
13. J. C. Pasquale, G. C. Polyzos, G. Xylomenos: The multimedia multicasting problem. Multimedia Systems (1998) 6, pp 43-59
14. J. F. de Rezende, A. Mauthe, S. Fdida, D. Hutchison: M-Connection Service: A Multicast Service for Distributed Multimedia Applications. Proc. of Multimedia Transport and Teleservices, International COST237 Workshop, (Copenhagen, Denmark), Nov. 1995
15. V. Roca, L. Costa, R. Vida, A. Dracinschi, S. Fdida: A Survey of Multicast Technologies. Technical Report RP-LIP6-2000-20, Université Pierre et Marie Curie LIP6, Paris, September 2000 (WiP)
16. T. Villemur, M. Diaz: A collaborative membership service and protocol for structured groups. International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'99), Las Vegas (USA), 28 June - 1 July 1999, Vol. IV, pp 2115-2121
17. R. Vitenberg, I. Keidar, G. V. Chockler, D. Dolev: Group Communication Specifications: A Comprehensive Study. Tech. Report CS99-31, Comp. Sci. Inst., The Hebrew University of Jerusalem and MIT Technical Report MIT-LCS-TR-790, September 1999
18. The Xpress Transport Protocol, XTP 4.0 Specification, see <http://www.ca.sandia.gov/xtp/biblio.html>