Enabling Group Communication in Global Networks

Markus Hofmann

Institute of Telematics, University of Karlsruhe, Zirkel 2, 76128 Karlsruhe, Germany
E-Mail: hofmann@acm.org
WWW: http://www.telematik.informatik.uni-karlsruhe.de/~hofmann

ABSTRACT

Group Communication is becoming more and more popular in global networks. New applications and protocols are under development to support cooperative work and information dissemination in worldwide, distributed systems. Most of them are based on a communication form in which a single source transmits data to multiple receivers.. This form of interaction is called multicast communication. As the geographic span and the size of multicast groups increase, efficient connection management schemes including scalable error and congestion control become more and more important. A single multicast flow might reach the entire global network while interfering with a large number of different data streams. Hence, carefully designed error and traffic control is necessary especially for multicast data transmission. The paper shortly presents the design of a multicast algorithm, named Local Group Concept (LGC), supporting applications with no or just loose time constraints. It provides reliable and semi-reliable data transfer and incorporates scalable mechanisms for error recovery.

1 Introduction

Groups show an ubiquitous form of relationship and inter-action in human society. People get together in groups to share common interests or to work on collaborative projects. Distributed computer systems are arranged into cooperative groups to master complex problems. Emerging applications, such as collaborative distributed work or information dissemination, support group interaction and are expected to require information exchange between a large number of geographically dispersed components. The recent success of applications deployed over the MBone [8] illustrates the enormous potential of group communication and demonstrates the instant need for economic multicast services in the Internet.

Recently, multicast data transmission based on Deering's IP multicast extensions [2] has been widely available in the Internet. But the bearer service provided by IP does not fit the requirements of all applications. It offers a best-effort service leaving it up to the application to provide the required reliability. However, the meaning of reliability is not well defined in the context of multicast communica-tion. There is a variety of widely different interpretations depending on the application environment. While one application requires correct and totally ordered data delivery to a well-known set of receivers, another one is satisfied with the correct delivery to a predetermined number of unknown receivers. Due to the wide range of different applications, a single multicast protocol suitable for all environments is infeasible. Instead, protocols should incorporate application specific semantics according to the model of Application Level Framing (ALF) [1]. The concept described in this paper defines a set of common protocol mechanisms which are fundamental for the provision of an all-reliable, scalable multicast service. The generic mechanisms are solely determined by network specific issues and apply to a wide variety of reliable multicast applications. They define some kind of protocol template that has to be filled by the application according to its specific requirements. The congestion signal, for example, is separated from the congestion control algorithm. It is up to the application to use the provided congestion information according to the specific application semantics. On one hand, it may slow down the transmitter during periods of high loss. On the other hand, receivers on congested links may be forced to leave multicast communication. The suitability of a certain action strongly depends on the application and the given group semantic. However, the basic mechanism for scalable retrieval of status information could be used by a wide range of different applications.

As the size and the geographic span of communication groups increase, efficient connection management schemes including scalable error and traffic control become more

and more essential [5]. Communication systems must be able to handle an arbitrary large number of receivers as well as a large number of different multicast data streams. Data and software distribution in global networks, for example, may lead to thousands of receivers being involved in a single multicast transmission. Due to their global character, multicast applications have an inherent potential to damage the Internet by causing congestion disasters. Moreover, measurements performed by Kurose, Towsley and Yajnik [14] have shown that packet loss in the global Multicast Backbone (MBone) is significant. Data sets collected in the MBone state that in one scenario more than 70% of all transmitted packets were not successfully received by at least one receiver. This illustrates the need for powerful error recovery and fair congestion control.

The characteristics of today's global networks have strongly influenced the design and the development of the Local Group Concept (LGC). The defined mechanisms provide support for all-reliable and semi-reliable data transmission in large-scale, heterogeneous networks. The algorithm supports stream-based applications for continuous data delivery as well as transfer of a fixed-size file. The protocol mechanisms are based on a best-effort delivery model with multicast support. This is in accordance with the core design principle of IP multicast. It is not necessary to change the underlying network protocol nor to modify any internal network equipment, such as routers or switches.

This paper outlines a concept for reliable multicast to large groups. Section 2 presents the protocol mechanisms defined by the Local Group Concept (LGC), while Section 3 describes an algorithm for dynamic establishment of a Local Group hierarchy. Finally, Section 4 concludes the paper.

2 The Local Group Concept (LGC)

The basic principle of LGC is to distribute the burden of acknowledgment handling and of error recovery among all the members of a global multicast group. LGC divides global communication groups into so-called *Local Groups* to improve scalability of point-to-multipoint services. The receivers which pertain to the same Local Group should be located in close vicinity. Different application-specific metrics are supported to establish a hierarchy of Local Groups. In each subgroup a *Group Controller (GC)* is responsible for processing status information from the assigned receivers. The integration of acknowledgment

processing capabilities into Group Controllers reduces the implosion problem [4]. A GC evaluates control messages from all the members of its subgroup and forwards them to the multicast sender or a higher-level Group Controller in a single composite control unit. This concerns error reports as well as messages to control data flow. Parallel processing of status reports and their combination into a single message per Local Group relieves the multicast sender as it reduces the number of control units to be processed at the transmitter.

Group Controllers also support the coordination of local retransmissions. In principle, any group member that has correctly received a certain data unit is able to perform retransmissions. It is not just the original sender which may retransmit lost data units. LGC recovers packet errors inside Local Groups firstly. A Group Controller requests missing data units from the sender or a higher-level GC only if not a single member of its subgroup has received the missing data unit correctly. However, retransmissions performed by regular receivers have to be coordinated in some way. Redundant retransmissions should be avoided and an appropriate group member should perform the retransmission. The receivers have to come to an agreement on which of them is performing a requested retransmission. They coordinate local recovery from data loss to avoid expensive retransmissions from the multicast sender or a higher level Group Controller. This reduces delay and decreases the load for sender and network.

In principle, any regular receiver can become a Group Controller. However, each communication participant decides on its own whether it is willing to be a potential Group Controller or not. A receiver may reject to perform local acknowledgment processing and local retransmissions due to its restricted processing power or memory space. Other reasons for the rejection could be the cost structure of the Internet service or security considerations. It is also possible to establish dedicated systems to take on the task of a Group Controller. ISPs, for example, may want to introduce Group Controllers in order to provide a value-added service to their customers.

2.1 Multicast Data Transfer

The Local Group Concept follows the core design principle of the Internet protocol family. It is based on the best-effort delivery service of IP, while reliability and ordering is built on an end-to-end basis. It is not necessary to change any network internal equipment such as routers or switches, nor to modify any existing network layer protocol.

Basic data transmission in LGC is quite simple. The data source simply addresses all the data units to a global multicast group without requiring any knowledge of the group members. The data units travel directly to the receivers along the multicast routing tree. Thus, packet forwarding is done solely by the network delivery service in the most efficient manner.

In order to perform local retransmissions, a Group Controller must be able to address all assigned receivers. At best, retransmitted data units reach exactly the members of a Local Group. Using current implementations based on IP multicast, the scope of global multicast packets could be restricted by a threshold time-to-live value. Administrative scoping of IPv6 will improve this method. However, this mechanism does not ensure that no other receivers except members of the Local Group get the local retransmission. Therefore, LGC assigns a separate multicast address to each Local Group. Using this local multicast address, retransmissions and control messages can be exclusively transmitted to the members of a Local Group without sending unwanted information to other receivers.

2.2 Acknowledgment Scheme

Common ARQ protocols use *positive* acknowledgments (ACK) to release data units from send buffer and *negative* acknowledgments (NACK) to request missed data units for retransmission. The Local Group Concept defines a third type of acknowledgment, called *semi-negative* acknowl-edgment (SNACK). A semi-negative acknowledgment indicates that a data unit has not yet been received correctly, but it does not request the data unit for retransmission.

Group Controllers use semi-negative acknowledgments to indicate that at least one member of the Local Group has received a data unit correctly, but another one is still missing it. Because packet errors are firstly recovered inside Local Groups, the sender should not retransmit the missed data unit. Instead, failed receivers will try to get it from one of their neighbors. For this reason, Group Controllers must not yet acknowledge such a data unit negatively. However, the sender is not allowed to release the data unit from its buffer. If the successful receiver dies, leaves multicast communication, or changes to another subgroup, all the members of its Local Group must have the chance to get the missing data unit from someone else. In the worst case, they will get it directly from the sender. Therefore, Group Controllers must not acknowledge such a data unit positively either. Instead, they will use a semi-negative acknowledgment to indicate the status of the data

unit. The meaning of each acknowledgment type and its interpretation in the context of LGC is summarized in Table 1.

Table 1: Types of Acknowledgments

Acknowledgment	Meaning
positive	*All* members of the Local Group have received the data unit correctly
semi-negative	Data unit has been received correctly by *at least one* member of the Local Group, while another member is still missing the data unit
negative	*Not a single* member of the Local Group has received the data unit correctly

Receivers and Group Controllers use a list of spans to indicate which data units have been received correctly. As Figure 1 illustrates, each acknowledgment segment of a status report starts with a packet sequence number *seq*. Data units with a sequence number smaller than *seq* have been received correctly by all members of the Local Group (cumulative positive acknowledgment).
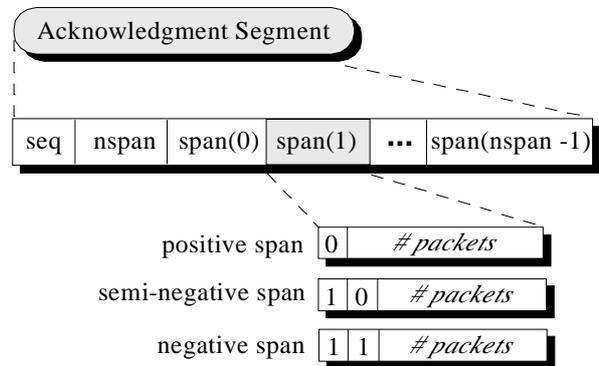

Figure 1: Acknowledgment Segment of a Status Report

The number of spans carried in an acknowledgment seg-ment is given by the value of *nspan*. The value of *span(0)* indicates the number of successive data units starting at packet number *seq* that are acknowledged either semi-negatively or negatively. The acknowledgment type of a span is given by its first two bits according to Figure 1. The value of *span(1)* acknowledges $n = $ size(1) successive data units starting at packet number *seq*+size(0), where size(i) is the number of data units acknowledged by the value of *span(i)* $(0 \leq i \leq nspan\text{-}1)$. In general, *span(i)*

($0 \leq i \leq$ *nspan*-1) acknowledges data units with sequence numbers {seq(i), seq(i) + 1, ... , seq(i) + size(i) - 1}, whereat the term seq(i) ($0 \leq i \leq$ *nspan*-1) is given by

$$\text{seq(i)} = seq + \sum \text{size(k)} \quad 0 \leq k \leq i\text{-}1$$

Receivers send status reports periodically to their Group Controller, indicating which data units they have received correctly. The time interval TACK(i), after which a receiver R(i) has to send a status report, depends on the measured round trip time *rtt(i)* between itself and its Group Controller. To avoid redundant retransmissions, we set

$$rtt(i) < \text{TACK(i)} < rtt(i) + \varepsilon \quad \forall R(i), \ \varepsilon > 0$$

If a Group Controller GC(j) does not get a status report from a certain receiver R(i) within the time interval TALIVE(j), it will assume that the receiver R(i) is not active any more or has left his Local Group. In this case, the Group Controller will not take care about receiver R(i) any longer. To avoid wrong assumptions concerning group membership, the time interval must be set to

$$\text{TALIVE(j)} > \text{TACK(i)} \quad \forall R(i) \in LG(j)$$

We propose to use time intervals, so that

$$\text{TALIVE(j)} \approx 3 \cdot \text{TACK} \ \land$$
$$\text{TACK} \geq \text{TACK(i)} \quad \forall R(i) \in LG(j)$$

Like regular receivers, Group Controllers periodically send status reports to a higher-level Group Controller or directly to the sender. At the end of each interval TACK(j), a Group Controller GC(j) calculates the status of its Local Group and places it into a single composite control message. The interval TACK(j) does not solely depend on the round trip time *rtt(j)* between GC(j) and its parent. Instead, it also depends on the round trip time *rtt(j,i)* between itself and each of its children R(i).

$$\text{TACK(j)} = \max\{rtt(j), rtt(j,i) \mid R(i) \in LG(j)\}$$

2.3 Error Recovery using Local Retransmissions

Common protocols use retransmission-based schemes for error recovery. Receivers do request missing data units directly from the multicast sender. However, any multicast receiver which holds a copy of the requested data unit is able to reply to the request. It is not solely the original packet source which may perform retransmissions. An optimal error correction scheme would stimulate the retransmission of missing data units by the receiver located closest to the requesting receiver. However, such local retransmissions need to be organized in some way to avoid a flood of redundant repair requests and corresponding retransmissions. Scalable Reliable Multicast (SRM) [3], for example, uses timers carefully set to suppress redundant requests and repetitive retransmissions. Tree-based approaches such as the Reliable Multicast Transport Protocol (RMTP) [9] or the Tree-Based Multicast Transport Protocol (TMTP) [15] solve this problem by following strong hierarchical guidelines. Missed data units are always requested from the parent, while there is no possibility for data exchange between hosts at the same level of a hierarchy. LGC uses a hybrid mechanism. Local Groups form a tree-like hierarchy, supporting hierarchical data exchange between Local Groups. However, packet errors are firstly recovered inside Local Groups. A Group Controller requests missing data units from the sender or a higher-level GC only if not a single member of its sub-group holds a copy of the missing data unit. Otherwise, errors will be recovered by local retransmissions.

There are several ways to organize local retransmissions. Group Controllers may either unicast or multicast missing data units within their subgroup. It is also possible for regular receivers to reply in an SRM-like manner to repair request. The suitability of each approach mainly depends on the application-specific environment and on the characteristics of the underlying network. Therefore, LGC defines two different modes of performing local retransmissions: a *load-sensitive* mode and a *delay-sensitive* mode. It is up to the application to choose the appropriate mode according to its specific requirements. It is also possible for different Group Controllers to operate in different modes.

Load-Sensitive Mode

In load-sensitive mode, Group Controllers aim to minimize network load caused by local retransmissions and control messages. The decision whether to perform local retransmissions via unicast or via multicast is taken dynamically based on the current group status. If the number of repair requests *req(i)* for a certain data unit data(i) exceeds a predefined threshold thresh, a Group Controller will multicast the corresponding retransmission. Otherwise, it will send it directly to the requesting receivers using unicast. However, local retransmissions performed by regular receivers will always be multicasted to the Local Group.

On receiving a repair request, a Group Controller marks the corresponding data unit data(i) for retransmission and increases its request counter *req(i)*. After a predefined

time interval `TACK(j)`, a Group Controller GC(j) initiates all outstanding local retransmissions, taking care of each request counter *req(i)* and the parameter `thresh`. Afterwards, it resets all request counters and unmarks the retransmitted data units.

On detecting a packet error, a Group Controller queues the sequence number of the missed data unit to be acknowledged negatively. If it receives a positive acknowledgment for such a data unit before expiration of timer `TACK(j)` (see (1) in Figure 2), it will unicast a repair request to the receiver which has confirmed the missing data unit first (see (2) in Figure 2). In addition, the Group Controller GC(i) removes the sequence number of the data unit from the NACK queue. On receiving such a repair request (see (3) in Figure 2), a regular receiver multicasts the requested data units to the Local Group (see (4) in Figure 2).
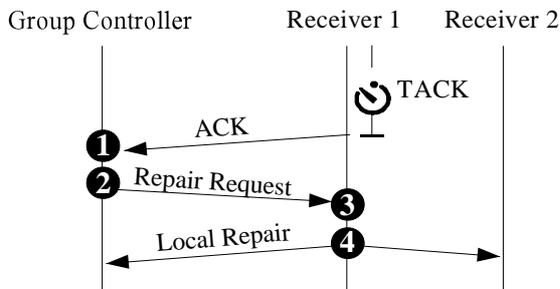

Figure 2: Repair Request by GC in Load-Sensitive Mode

At the end of each time interval `TACK(j)`, a Group Controller GC(j) requests the remaining data units in the NACK queue from its parent by sending a negative acknowledgment. Data units requested for local retransmission are acknowledged semi-negatively.

Delay-Sensitive Mode

A Group Controller GC(j) operating in load-sensitive mode delays local retransmission until expiration of `TACK(j)`. Such a delay is not acceptable for applications with some time constraints. Therefore, the delay-sensitive mode defines a slightly modified mechanism for local error recovery.

On receiving a NACK, a Group Controller GC(j) in delay-sensitive mode immediately multicasts the requested data units to the Local Group. Further requests for the same data units will be ignored until the expiration of `TACK(j)`. If a Group Controller itself is missing a certain data unit, it will try to get it from one of its assigned receivers right away (see Figure 2).

On detecting a packet error, the Group Controller immediately multicasts a repair request to all members of its subgroup (see (1) in Figure 3). Each receiver R(i) holding a copy of the requested data unit delays the retransmission for a certain time interval `TDELAY(i)`. (see (2) and (3) in Figure 3). The intervals `TDELAY(i)` should be set to

$$\text{TDELAY(i)} \ne \text{TDELAY(j)}, \textit{ for most } i \ne j$$

For example, receivers could set their own `TDELAY(i)` depending on the number of hops between themselves and their assigned Group Controller. If the timer `TDELAY(i)` expires, the receiver transmits the requested data unit to the local multicast group (see (4) in Figure 3). Receivers that are still waiting to perform the same local retransmission will stop their repair timers on receiving the local retransmission (see (5) in Figure 3).
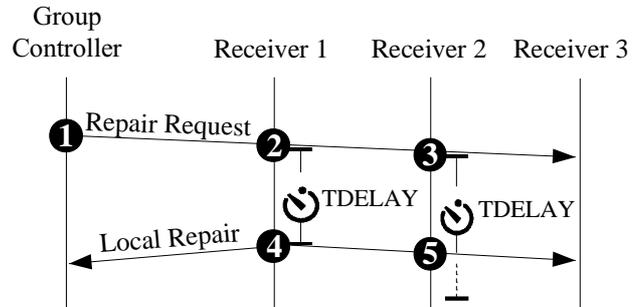

Figure 3: Repair Request by GC in Delay-Sensitive Mode

At the end of each interval `TACK(j)`, a Group Controller negatively acknowledges all data units which it has not received yet.

2.4 Congestion Control

The Local Group Concept (LGC) separates the congestion signal from the congestion control algorithm. It provides mechanisms to detect network congestion based on the status reports of each receiver, while it is up to the application to choose the appropriate algorithm to deal with it. For example, LGC allows to slow down the sender during periods of high loss in a TCP-like manner [6]. It also provides mechanisms to dynamically create and announce additional IP groups, each of them transmitting at a different rate. Receivers may chose an appropriate rate and join the corresponding IP multicast group [10]. Of course, it is also possible to drop out and ignore slow receivers. More research is going on to develop and to evaluate several algorithms for congestion control.

3 DCS: Management of Local Groups

The placement of Group Controllers and the assignment of receivers to appropriate Local Groups are essential for the efficiency of data transfer. Simulations of MBone scenarios have shown that inappropriate group hierarchies result in inefficient network utilization, in increased transfer delay, and in reduced throughput [13]. The burden of acknowledgment processing and of doing retransmissions should be fairly distributed among all the hosts according to their capabilities. Another important issue is robustness as regards inconsistency and fault. The methods used to define and maintain Local Groups must be able to handle failures of any host. As controlling entities are a single point of failure, an auto-recovery mechanism is necessary to repair the dropping out of a Group Controller independently. All these mechanisms have to work correctly under heavy load and must not be susceptible to temporarily high packet loss or increased transfer delay. In addition, they must be able to deal with dynamic groups and should support mobile environments in which hand-over could take place. As receivers and Group Controllers may join and leave dynamically at any time, Local Groups must be self-maintaining. Due to changes in membership, it might be necessary to reconfigure a group hierarchy. Affected hosts must be able to detect such a need and to execute the reconfiguration on their own without any administrative support.

Due to the identified requirements, Local Groups and the implicitly defined group hierarchy will not be static or stable in nature. They will consist of loosely coupled receivers which may join or drop out at any time. Such a continuous agitation makes it difficult to maintain a consistent view on the membership and on the status of each Local Group. Therefore, one design decision has been the use of "soft states", a principle that has been proven worthwhile for several Internet protocols. The state information necessary to maintain a Local Group will be distributed periodically according to a predetermined refresh interval, thus adding a high level of robustness. Another design decision has been that the definition of Local Groups is receiver-driven and decentralized. There is no central instance deciding which Local Group a receiver is assigned to and where to place the Group Controllers. Such a mechanism would presume a global view on the multicast group which is no easy task for highly dynamic groups with possibly failing receivers. Therefore, it is up to the receivers to arrange themselves in Local Groups. Each receiver has to decide whether there is an appropriate Group Controller to attach to or not. If it is

useful, a receiver may also define a new Local Group and declare itself as a Group Controller.

The establishment and maintenance of logically structured group hierarchies is not a task of data-level protocols. Thus, rather than integrate mechanisms to define Local Groups into LGC, a new protocol, named *Dynamic Configuration Service (DCS),* has been defined. State information maintained by DCS is distinct and independent from data-level protocols. Therefore, DCS is not restricted to LGC, but it can interact with any other protocol that requires a logically structured receiver hierarchy. DCS finds out the address to which receivers should deliver their status reports. However, DCS does not define the parameters for selection of an appropriate Group Controller. It just provides all the mechanisms necessary to get the required state information. It is the task of session-level control and of QoS management to set the right parameters (e.g., to decide whether to choose the closest Group Controller with respect to transfer delay or number of hops). Figure 4 illustrates the integration of LGC and DCS in an overall protocol architecture.
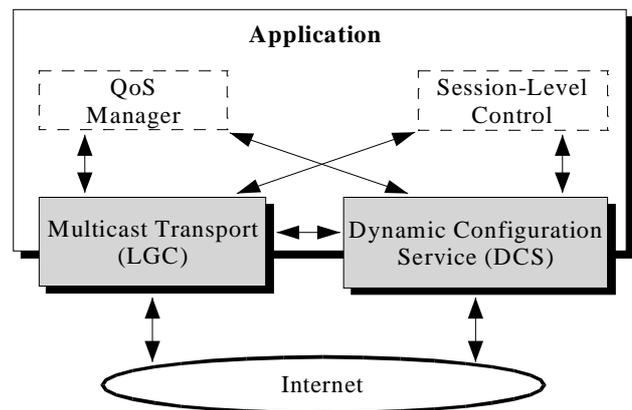

Figure 4: Protocol Architecture

In order to define and to maintain a group hierarchy, several tasks have to be performed. First of all, Local Groups need to be established by assigning certain receivers the role of a Group Controller. This is called the *placement of Group Controllers.* Group Controllers are determined dynamically as receivers join and leave a multicast session. Once Local Groups have been established and their corresponding Group Controllers have been chosen, joining receivers have to select an appropriate Local Group to attach to. This is called the *selection of Group Controllers.* A prerequisite for the selection of Group Controllers is their advertisement, making it possible for the receivers to get knowledge about existing

Group Controllers. Each of these protocol mechanisms will be explained in a separate section. To improve clarity, the *advertisement of Group Controllers* will be explained first. Comments on the selection and placement of Group Controllers will follow.

3.1 Advertisement of Group Controllers

Each Group Controller periodically sends packets of type `LG_ADVERTISE`. These messages are addressed to a separate, group-specific multicast address. Their scope is restricted by setting the time-to-live (TTL) to a variable value *ttl_send*. Communication participants listen to the group-specific DCS address and use advertise messages to identify existing Group Controllers.

Advertise messages contain information that allows receivers to select the most appropriate Group Controller according to their requirements. By default, an advertise message includes the smoothed error probability of a Group Controller, the number of receivers currently controlled by the Group Controller as well as the multicast address of the represented Local Group. Optional fields have been defined to include additional information, for example a timestamp for the calculation of transfer delay between Group Controller and receiver. A Group Controller copies the local system time into the advertise message. If the system time of all participating hosts is synchronized, for example by using the Network Time Protocol (NTP) [11], transfer delay could easily be calculated. It will be the difference between the local arrival time of an advertisement and the time carried in the message. To determine the number of hops between Group Controller and receiver, each Group Controller enters the time-to-live value *ttl_send* used to send the advertise message into the packet. Getting an LG_ADVERTISE, receivers know that the number of hops between themselves and the message source is smaller then the value of *ttl_send*. Group Controllers use dynamically changing values of *ttl_send* (*expanded ring advertisement*), so that the number of hops *h* between receivers and Group Controllers could be estimated to the minimum value of *ttl_send* received during a certain time interval TTTL:

$$h = \min\{ ttl\_send \mid ttl\_send \text{ received within last } \mathtt{TTTL}\}$$

Reset after the time interval TTTL ensures that *h* adapts to possibly changing routes in the Internet. Additional packet fields to support further metrics (e.g., carrier fees, security options, round trip time) could easily be added by setting the right payload type of an advertise message.

3.2 Selection and Placement of Group Controllers

Once the service user has issued a listen request, a receiver initializes an association control block. Each of these blocks contains an entry named `redirect`, which is undefined at startup. This entry will identify the controller to which receivers should deliver their status reports. While the value of `redirect` is undefined, LGC will address all status reports to the data source.

With the establishment of an association control block, the receiver activates an initialization timer named `INIT-TIMER` and changes from the *inactive* to the *pending* state. It joins the group-specific DCS group and, therefore, stimulates the transmission of an IGMP Host Membership Report. Now, the host is set up to receive packets addressed to the global DCS group and it buffers all the information obtained from received advertise messages. After expiration of timer `INIT-TIMER`, a receiver evaluates the buffered information, selects one of the discovered Group Controllers, sets the `redirect` entry of the association control block to the address of the chosen Group Controller, and changes to the *active* state. While being in *active* state, a receiver will continue to process advertise messages and to update the `redirect` entry dynamically.

The selection of an appropriate Group Controller could be based on several criteria. While existing approaches are restricted to use the number of hops for the selection of an appropriate Group Controller, it is preferable to consider multiple parameters. The suitability of a certain metric, such as delay, bandwidth, throughput, error probability, reliability, carrier fees, or number of hops, mainly depends on the given application environment. While a time constrained application may wish to minimize transfer delay, a user transferring files is interested in reducing financial cost of a transfer. Of course, it could also be suitable to combine several metrics and to weight them according to the intention of a service user. For this reasons, DCS does not define how to choose the best-fit Group Controller. DCS itself does not have all the knowledge necessary to make the right decision. The protocol just provides mechanisms necessary to identify existing Group Controllers and their current characteristic. It is the task of session-level control to choose the right parameters. The basic idea is to share common protocol mechanisms across multiple application instances and to allow the application semantics to be reflected back into the established group hierarchy. Of course, receivers should choose their Group Controller in a way such that load will

be balanced within the global group hierarchy.

If no appropriate Group Controller could be found according to the application requirements, the joining receiver has two possibilities. On one hand, it could attach itself directly to the Local Group represented by the multicast transmitter. In this case, the `redirect` entry of its association control block will be undefined and LGC will address all status reports to the multicast transmitter. On the other hand, it could establish a new Local Group and appoint itself as Group Controller of the new subgroup. One of the identified Group Controllers or the multicast transmitter itself will be defined to be the parent, of the newly established subgroup. All reports about the status of the new Local Group will be addressed to the parent Group Controller, thus building a group hierarchy.

Initially, it is the founder of a Local Group which will become the Group Controller. Due to the joining and dropping out of receivers, the group structure has to be reconfigured dynamically during the lifetime of an association. It might be beneficial to split a growing Local Group or to merge several waning subgroups. In addition, a joining receiver might be a better Group Controller than the current one due to its network connection or its processing capacity. The following section describes the scheme used to perform such a dynamic reconfiguration of the global group structure.

3.3 Dynamic Reconfiguration of Local Groups

As receivers and Group Controllers may join and leave during the lifetime of a connection, it is necessary to adjust the placement of Group Controllers dynamically according to the current group status, the current network load, and the current characteristics of each communication participant. For example, it could be advantageous to place the Group Controller in the center of a Local Group. Various schemes based on different criteria could be used to determine the optimal Group Controller among all the members of a Local Group. Existing approaches define the host located next to the sender as controller of a subgroup [15]. However, the number of hops between transmitter and host does not indicate the suitability of a certain receiver to take the role of a Group Controller. A Group Controller has to perform local acknowledgment processing. For this task, it needs sufficient processing capacity and buffer space. In addition, a Group Controller should be able to locally retransmit as many data units as possible. Therefore, a much better metric for the placement of Group Controllers seems to be the current error probability of a host.

Besides the move of Group Controllers, the splitting and merging of Local Groups might become necessary due the changes in group membership. The burden of acknowledgment processing and of doing local retransmissions has to be distributed fairly among all the Group Controllers, thus, resulting in a well-balanced group structure.

Receivers use information contained in LG_ADVERTISE messages to maintain a table of reachable Group Controllers. On receiving an advertise message, a host will add a new entry to the table or it will update an existing one. Each entry represents a Group Controller and indicates its error probability, the estimated number of hops between receiver and Group Controller as well as the size and the multicast address of the Local Group. If there is some more information about the characteristic of a Group Controller included in its advertise messages, this information will also be added to the table (e.g. transfer delay or carrier fees). While updating their table, Group Controllers ignore their own advertise messages.

Each entry is valid for a predetermined time interval `TVAL`. When the timer expires and no further advertise message of a certain Group Controller has been received within the last time interval, receivers will delete the corresponding entry in the table. Therefore, each host has an up-to-date view on active Group Controllers, their identity, and their current status. There is no additional information exchange necessary to keep the table valid. If a Group Controller fails or leaves, the corresponding table entry will time out and be deleted. To ensure correctness of this mechanism, the expiration time `TVAL` must be longer than the time `TADV` between two successive advertise messages of one and the same Group Controller. We propose to choose

$$TVAL \approx (3 \cdot TADV) + \varepsilon$$

to counterbalance the loss of two successive advertise messages. If three successive announce messages are lost, a receiver will erroneously delete the corresponding entry. However, on receiving the next advertise message the receiver will add the Group Controller again.

The table size could be reduced by setting filters for processing advertise messages. For example, a receiver not interested in Group Controllers beyond a scope of `max_ttl` could establish a filter to avoid the addition of Group Controllers with an estimated distance $h(GC)$ higher than `max_ttl` hops to the table.

Receivers periodically rate the suitability of their current Group Controller GC(i). If the rating $r(j)$ of another Group

- 8 -

Controller GC(j) is better than the rating $r(i)$, the `redirect` entry will be set to the address of GC(j). However, the difference between $r(i)$ and $r(j)$ should be higher than a given threshold to avoid oscillatory changing between Local Groups. A problem might also occur in case a large number of receivers decide to assign themselves to a newly defined Group Controller. Due to an overwhelming number of new members, the new Group Controller might get under heavy load, thus decreasing its rating. This would probably result in a further reconfiguration. Therefore, receivers delay spontaneous reconfiguration for a random time to check the rating of a newly detected Group Controller again.

In addition, a receiver R(i) periodically calculates its own rating $r(i)$. If $r(i)$ is better than the rating $r(j)$ of its current Group Controller by some non-negligible amount, the receiver will establish a new Local Group claiming itself to be a Group Controller. It will start to send LG_ADVERTISE messages to advertise its existence and its current status. Nearby receivers may now join the newly established Local Group, thus relieving their previous Group Controller.

3.4 Fault Tolerance

Fault tolerance is inherent to the mechanisms defined by DCS, because each table entry will time out without being refreshed. If a Group Controller GC(j) fails, receivers will not get advertise messages generated by GC(i) any more. On expiration of timer `TVAL`, assigned receivers will remove the corresponding table entry and will select another Group Controller. If a receiver does not succeed and is not willing to be a Group Controller, the `redirect` entry will be undefined and all status reports will be addressed to the sender.

3.5 Security

It might be desirable to restrict the members of a Local Group to a subset of all potential group members. A company having defined Group Controllers, for example, might not be interested in supporting receivers of their competitors. ISPs charging their customers for using installed Group Controllers want to restrict the set of assigned receivers to the set of their customers. This could be achieved based on cryptography by sealing the content of advertise messages, and charging for decrypt keys.

On the other hand, receivers probably do not want to depend on repairs from a company site that is a competitor. However, receivers themselves decide to which Group Controller they assign themselves. They just need to activate a positive or a negative filter to limit the set of acceptable Group Controllers.

4 Future Work

Currently, LGC uses a retransmission-based scheme to recover from packet errors. In networks with uncorrelated loss at the receivers, hybrid error recovery using both Forward Error Correction (FEC) and Automatic Repeat Request (ARQ) might be quite advantageous. Work is in progress to integrate such a hybrid approach into LGC in order to support heterogeneous communication groups.

As proposed in [12], a sender will not retransmit copies of original data units. Instead, it will send FEC packets to recover from packet loss. Furthermore, LGC will use multiple groups to transmit different FEC packets. Receivers will be able to receive the required number of FEC packets by joining and leaving appropriate multicast groups.

Original data units $\{data(1),..., data(k)\}$ are sent to a multicast address $addr(0)$. An FEC encoder at the sender calculates so-called FEC packets $\{p(1),...,p(n-k)\}$ depending on the original data units. An FEC encoder at the receiver is able to reconstruct the original data units as soon as it has received any k out of the packets $\{data(1),..., data(k), p(1),...,p(n-k)\}$. This is advantageous especially in networks with uncorrelated packet loss, because a single parity packet can repair the loss of different data units.

In addition, the set of FEC packets is partitioned into subsets, for example

$$s(1) = \{p(1),...,p(4)\}$$
$$s(2) = \{p(5),...,p(9)\}$$
$$s(3) = ...$$

Each of these subsets is addressed to a different multicast group. For example, the sender might address all packets in subset $s(1)$ to $addr(1)$ and packets in subset $s(2)$ to $addr(2)$. Receivers permanently join $addr(0)$ to receive original data units. They also join additional multicast groups according to the error probability of their connections. Receivers getting all original data units correctly do not need to join any other group. Receivers missing up to four original data units will also join $addr(1)$ to get the required number of FEC packets. Receivers missing up to 9 data units will join both, $addr(1)$ and $addr(2)$. Group membership could change dynamically during data transfer. However, frequent join/leave should

be avoided to reduce multicast routing load. Therefore, receivers base the decision whether to join or leave a certain group on their smoothed error probability. If a receiver is not able to recover a lost packet, it could also use negative acknowledgments to request the original data unit.

FEC coding requires additional processing capabilities at the endsystems. Less powerful receivers might not be able to keep up with encoding FEC packets for error recovery. Such receivers will gain by attaching themselves to a powerful Group Controller using FEC to improve its error probability. Thus, they will use local retransmission to get missed data units and are not enforced to decode FEC packets.

5 Additional Information

More information about the Local Group Concept (LGC) and the Dynamic Configuration Service is available on the LGC Homepage at http://www.telematik.informatik.uni-karlsruhe.de/~hofmann/LocalGroups.html.

6 Acknowledgments

7 References

[1] Clark, D.; Tennenhouse, D.: *Architectural Considerations for a New Generation of Protocols;* Proceedings of ACM SIGCOMM'90, pp. 201-208, September 1990.

[2] Deering, S.: *Host Extensions for IP Multicasting;* RFC 1112, August 1989.

[3] Floyd, S.; Jacobson, V.; McCanne, S.; Liu, C.; Zhang, L.: *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing;* Computer Communication Review, Vol. 25, No. 4, Proceedings of ACM SIGCOMM '95, August 1995.

[4] Hofmann, M.: A *Generic Concept for Large-Scale Multicast;* in: B.Plattner (ed.), Broadband Communications, Proceedings of International Zurich Seminar on Digital Communications (IZS '96), LNCS, No. 1044, Springer Verlag, February 1996.

[5] Hofmann, M.: *Scalable Multicast Communication in the Internet;* ConneXions - The Interoperability Report, Vol. 10, No. 10, October 1996.

[6] Jacobson, V.: *Congestion Avoidance and Control;* Proceedings of ACM SIGCOMM '88, pp.314-329, Stanford, CA, August 1988.

[7] Kasera, S. K.; Kurose, J.; Towsley, D.: *Scalable Reliable Multicast Using Multiple Multicast Groups;* Technical Report TR 96-73, University of Massachusetts, October 1996.

[8] Kumar, V.: MBone: *Interactive Multimedia on the Internet;* New Riders Publishing, Indianapolis, USA 1995.

[9] Lin, J. C.; Paul, S.: *RMTP: A Reliable Multicast Transport Protocol;* Proceedings of IEEE INFOCOM '96, 1996.

[10] McCanne, S.; Jacobson, V.; Vetterli, M.: *Receiver-driven Layered Multicast;* Proceedings of ACM SIGCOMM, 1996.

[11] Mills, D. L.: *Network Time Protocol (Version 3);* RFC 1305, March 1992.

[12] Nonnenmacher, J.; Biersack, E.; Towsley, D.: *Parity-Based Loss Recovery for Reliable Multicast Transmission;* Technical Report 97-17, University of Massachusetts, March 1997.

[13] Rohrmüller, M.: *Bewertung von Verfahren zur Strukturierung globaler Kommunikationsgruppen;* Institut für Telematik, Universität Karlsruhe, 1997.

[14] Yajnik, M.; Kurose, J. F.; Towsley, D.: *Packet Loss Correlation in the MBone Multicast Network;* IEEE Global Internet '96, London, England, November 20-21, 1996.

[15] Yavatkar, R.; Griffioen, J.; Sudan, M.: *A Reliable Dissemination Protocol for Interactive Collaborative Applications;* Proceedings of ACM Multimedia '96, 1996