# Group Communication Server: A Design Exercise

**Daniel Amyot**

**TRIOsoft project, August 1996**

## 1.0 Introduction

### 1.1 Motivation

### 1.2 Goals

### 1.3 Structure of this document

## 2.0 GCS Informal Description

### 2.1 Use

### 2.2 Functionalities

## 3.0 Architecture

### 3.1 Alternatives

Several architectural patterns for structuring concurrency in a Group Communication Server [Schmidt 95].

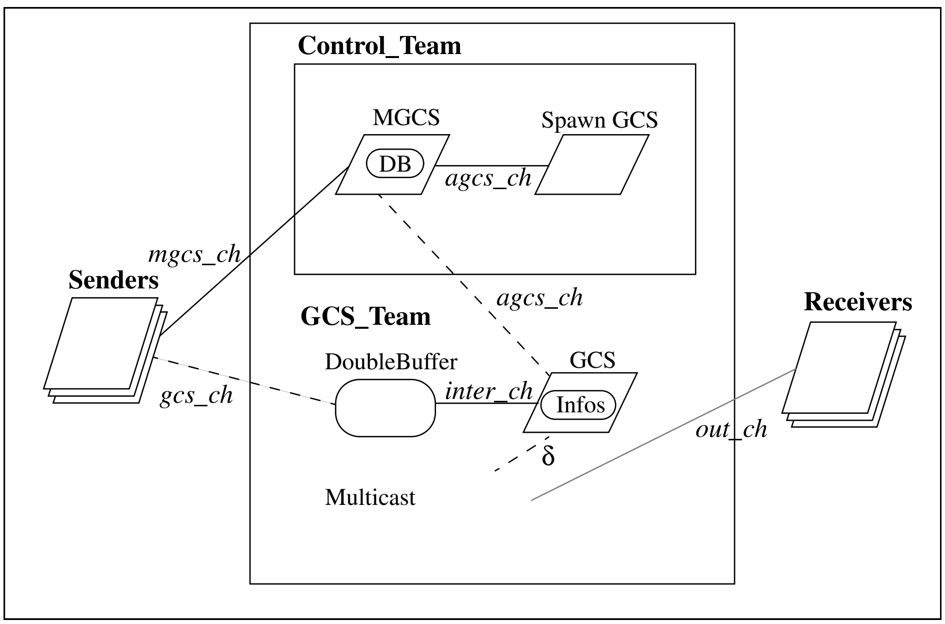
- *Reactive* : Not really concurrent. One monolithic server process that answers all requests for the managed groups. Only one message queue.

- *Thread-per-request* : Spawns an independant thread for every incoming request. No limitation, but high cost for the dynamic creation of so many threads.
- *Thread-per-session* : Less dynamic and costly than Thread-per-request. No limitation, but requires ressources while a session (group) exists (active or not). Each group could have its own process and queue and execute concurrently with the others (but the internal execution would be sequential).
- *Thread-pool* : Fixed number of threads (allocated once), thus saving the dynamic allocation performance problems. Limitations due to the fixed number of instances (the overflow of requests might be resolved with a global queue)

### 3.2 Selected GCS architecture

Thread-per session for GCS_Team, and Thread-per-request for Multicast.

**FIGURE 1.** GCS Architecture




### 3.3 Data structures

## 4.0 Use Case Maps

Here are some UseCase Maps:

**FIGURE 2.** Group Creation

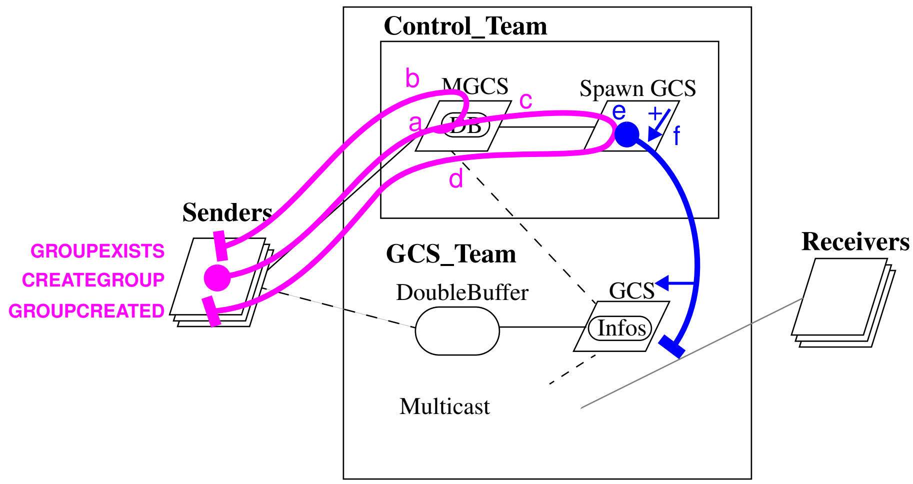



**TABLE 1.** Activities of "Group Creation"

| Activity | Type | Input/Output | Comment |
|---|---|---|---|
| CREATEGROUP | Request | *newgroupid, infos* | Group creation. Anyone can create. |
| GROUPCREATED | Ack | *newgroupid* | Group created. |
| GROUPEXISTS | Error | *newgroupid* | Group *newgroupid* already exists. |
| a | Internal | | MGCS checks the database (*DB*). |
| b | Condi-tion | | *newgroupid* is in *GCSlist (DB).* |
| c | Condi-tion | | *newgroupid* is not in *GCSlist.* |
| d | Internal | | MGCS updates the database (inserts *newpropupid* in *GCSlist*) |
| e | Internal | | Relays the CREATEGROUP request. |
| f | Internal | | Instantiates a new GCS_Team with user-provided informations (*infos*) |

**FIGURE 3.** List of Groups

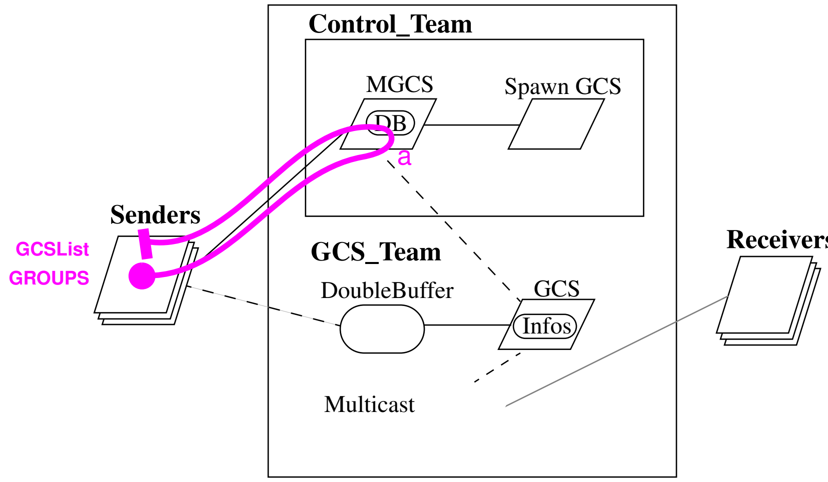



**TABLE 2.** Activities of “List of Groups”

| Activity | Type | Input/Output | Comment |
|---|---|---|---|
| GROUPS | Request | | List the existing groups within server. |
| GCSList | Ack | *GCSList* | List of groups. |
| a | Internal | | MGCS checks the database (*DB*). |

**FIGURE 4.** Group Deletion

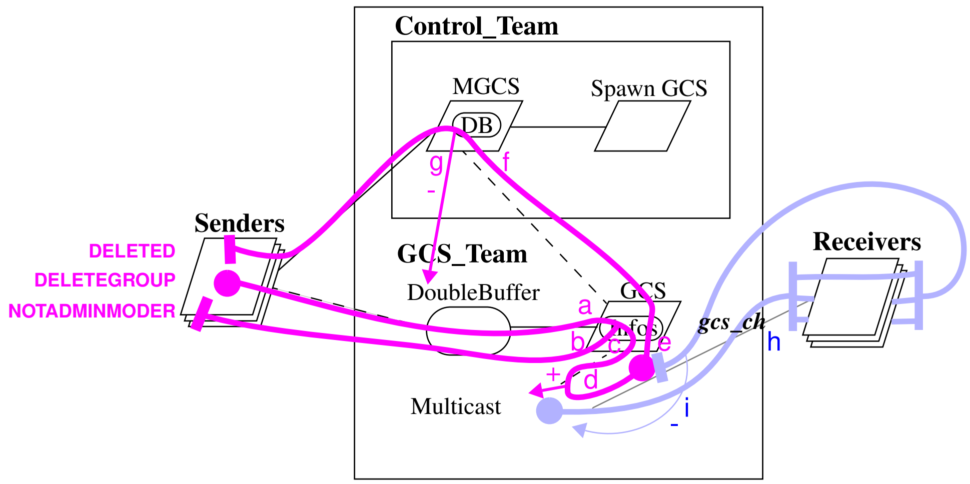



**TABLE 3.** Activities of "Group Deletion"

| Activity | Type | Input/Output | Comment |
|---|---|---|---|
| DELETEGROUP | Request | | Group deletion. |
| DELETED | Ack | | Group deleted. |
| NOTADMIN-MODER | Error | | Sender does not have sufficient rights. |
| a | Internal | | GCS checks the database (*infos*). |
| b | Condi-tion | | Group administered and sender is not the admin. |
| c | Condi-tion | | Group not administered or sender is the admin. |
| d | Internal | | Instantiates Multicast processes (one per member) |
| e | Internal | | Wait until all messages are sent |
| f | Internal | | MGCS updates the database (removes the group from *GCSlist*) |
| g | Internal | | MGCS destroys the GCS_Team |
| h | External message | Message: GROUP WAS DELETED | Announce group deletion to all mem-bers (using the control channel ***gcs_ch*** instead of ***out_ch***). |
| i | Internal | | Destroys the Multicast processes. |

**FIGURE 5.** Member Registration

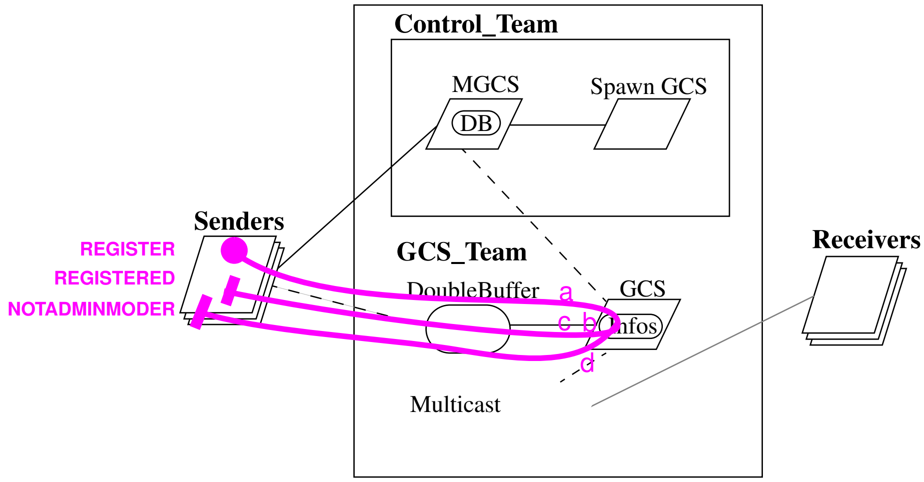



**TABLE 4.** Activities of "Member Registration"

| Activity | Type | Input/Output | Comment |
|---|---|---|---|
| **REGISTER** | Request | *ChanId* | Register sender (or another user if admin) in the group. Also used to modify ChanId. |
| **REGISTERED** | Ack | | Member registered. |
| **a** | Internal | | GCS checks the database (*mbrL*). |
| **b** | Condition | | Group is not private, or sender is the admin. |
| **c** | Internal | | GCS updates the database (inserts the pair *MemberID.ChanId* in *mbrL*). |
| **d** | Condition | | Group is private and sender is not the admin. |

**FIGURE 6.** List of Members of a Group

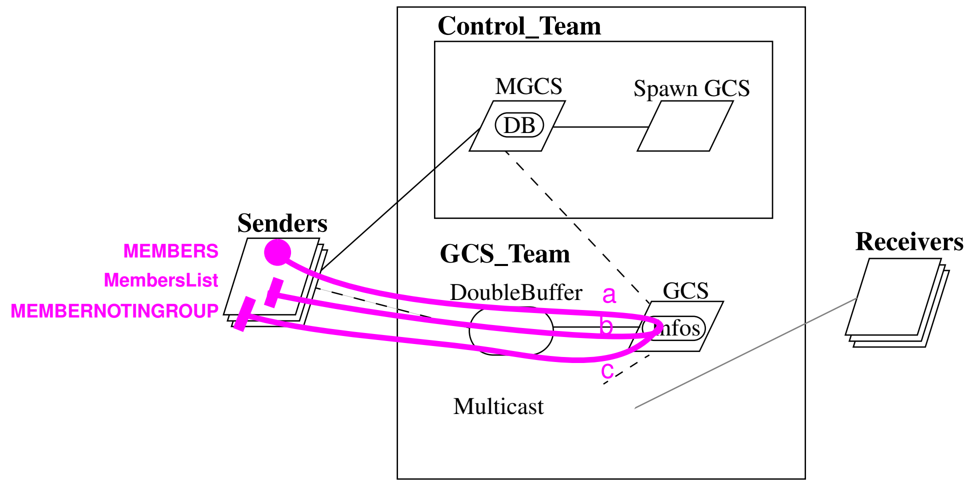



**FIGURE 7.** Member DeRegistration

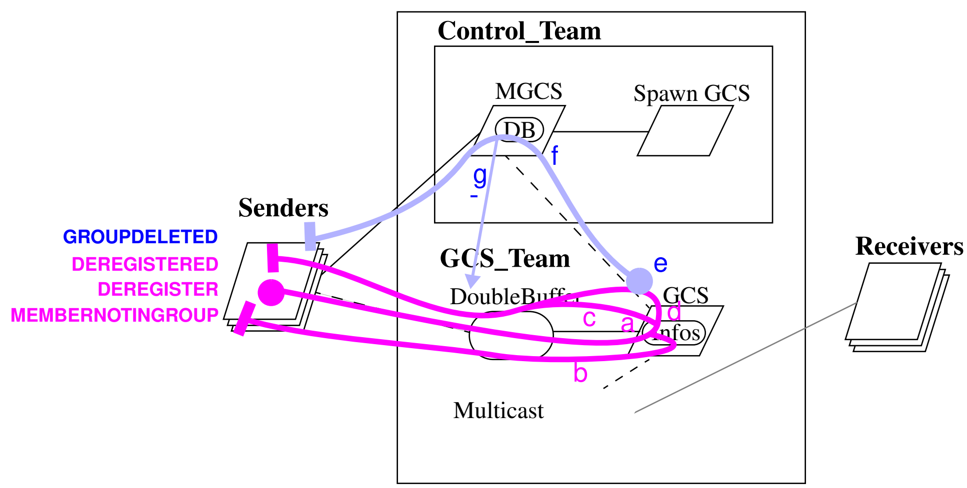


**FIGURE 8.** Message Sending (Multicast)

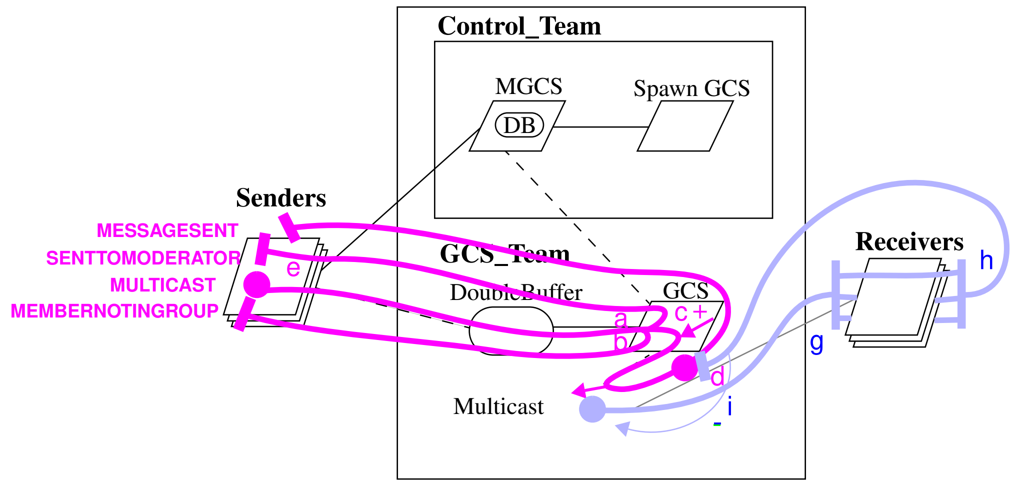



**FIGURE 9.** Administrator Changing

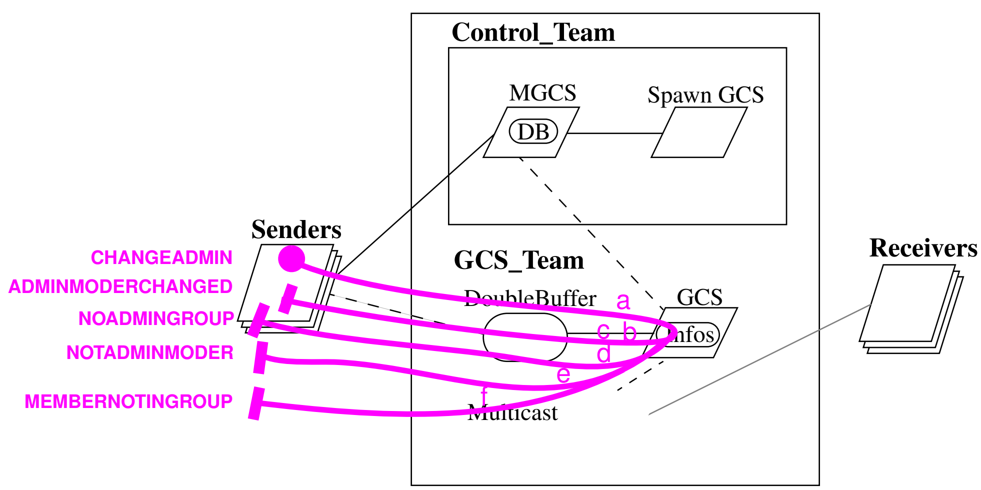
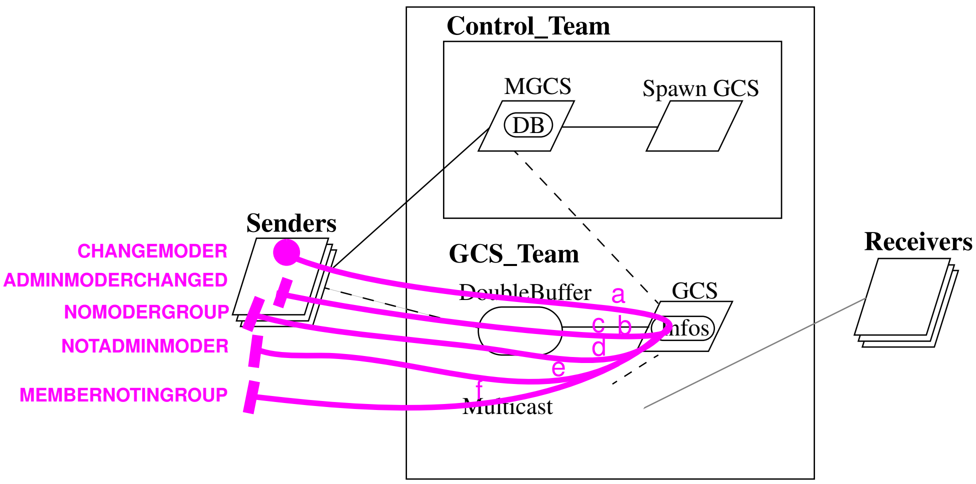


**FIGURE 10.** Moderator Changing



# 5.0 LOTOS Specification

## 5.1 General structure

## 5.2 Data Types

## 5.3 Processes

# 6.0 Test Processes

## 6.1 UCM-based tests

## 6.2 High-yield tests

# 7.0 Test Results

# 8.0 Properties and their Verification

# 9.0 Towards an ACE Specification

## 9.1 ACE

## 9.2 Specification of ACE components

## 9.3 GCS specified in ACE-oriented style

# 10.0 Conclusion

## 10.1 What's done

## 10.2 What's next