Assignment 2

Causal and total group communication

GMS architecture

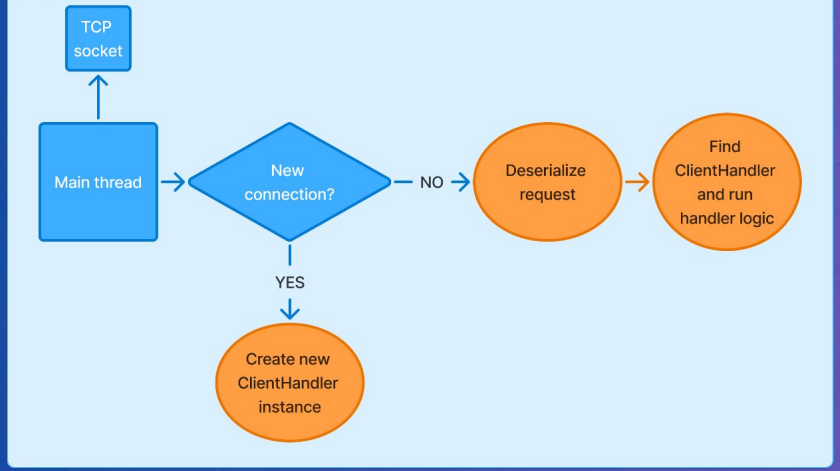
ClientInfo

Type	Field	Description
String	username	Username of a client in a group
String	ipAddress	IP of the client
int	port	Port of the client

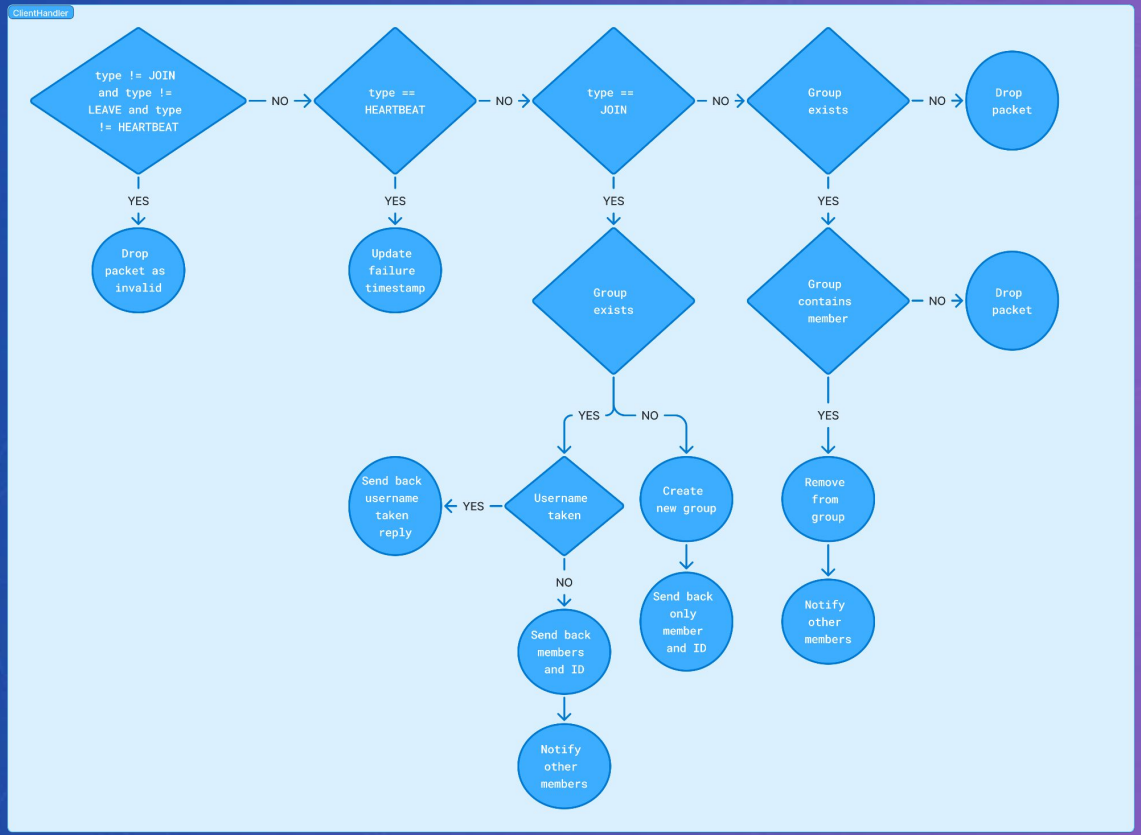
Group

Type	Field	Description
int	id	ID of the group
String	name	Name of the group
List<ClientInfo>	clients	List of all the members of the group

Main



GMS architecture



GMS requests

GMSRequest

Type	Field	Description
RequestType	type	JOIN,LEAVE or HEARTBEAT
String	username	Name of the group
int	futureId	Used to unblock sender

GMSJoinRequest

Type	Field	Description
String	groupName	Group to join
String	ipAddress	IP of the client
int	port	Port used for group communication

GMSLeaveRequest

Type	Field	Description
int	groupId	The group to leave

GMS replies

GMSReply

Type	Field	Description
ReplyType	type	JOIN,LEAVE or UPDATE
int	groupId	ID of the group

GMSUpdateReply

Type	Field	Description
ClientInfo	user	Client involved in the update
Action	clientAction	The action taken by the client (JOINED, LEFT)

GMSJoinReply

Type	Field	Description
int	futureId	Used to unblock sender
FailReason	reason	The reason for failure (if failed)
List<ClientInfo>	clients	All the clients in the group

GMSLeaveRequest

Type	Field	Description
int	futureId	Used to unblock sender
FailReason	reason	The reason for failure (if failed)

ChatApp data structures

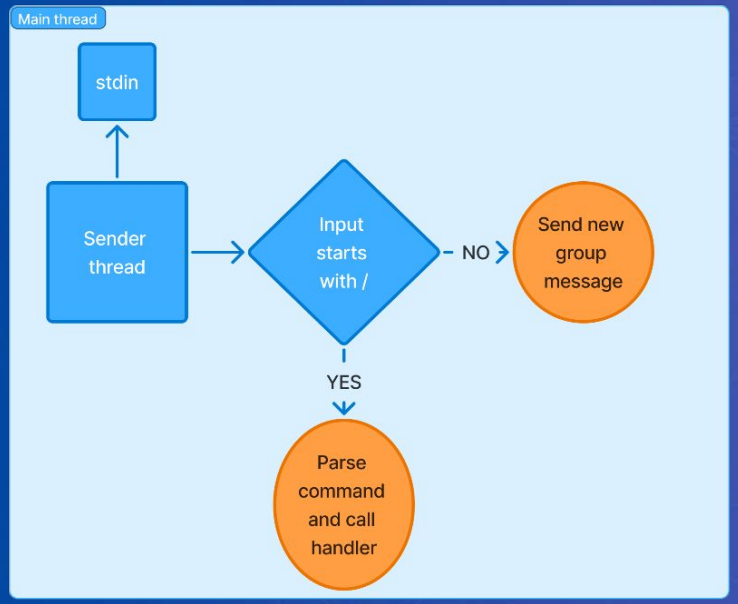
ChatMessage class

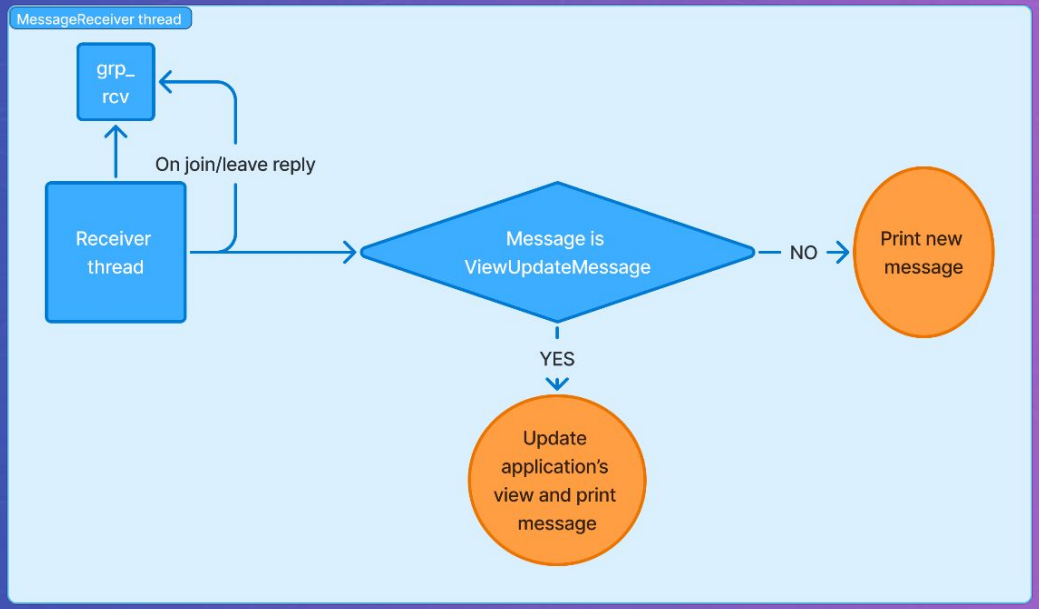
Type	Field	Description
`String`	`author`	Name of the sender
`Date`	`timestamp`	Timestamp of the sender
`String`	`message`	The content

GroupChat class

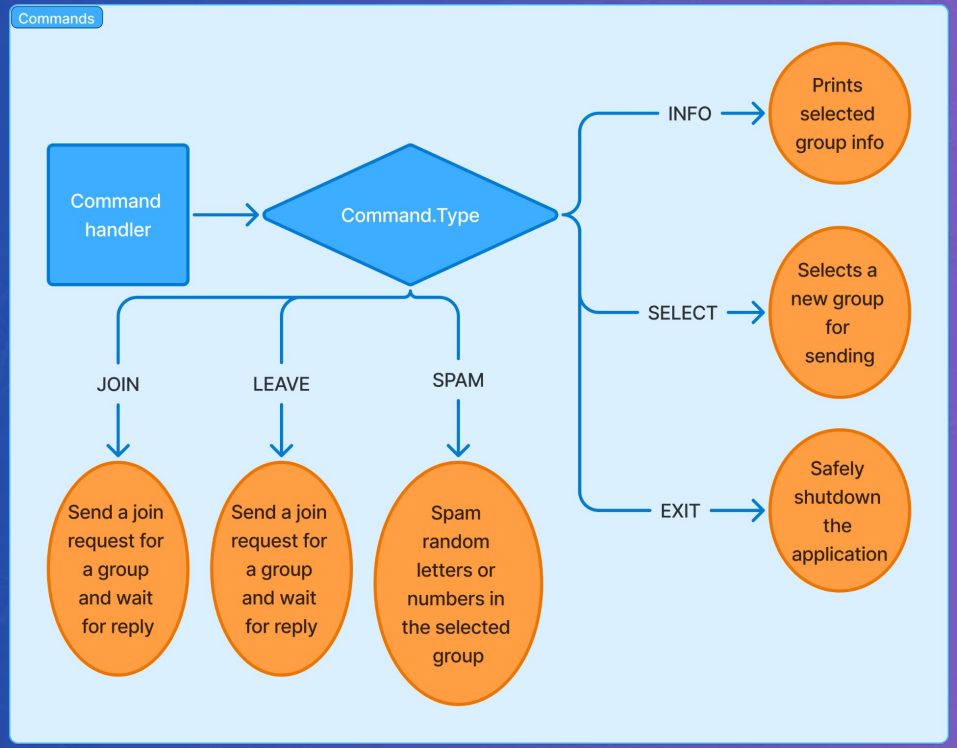
Type	Field	Description
`int`	`id`	Unique ID of the group
`String`	`name`	Name of the group
`String`	`username`	Username of the client in this group chat
`List<String>`	`members`	Usernames of all members in the chat

ChatApp architecture





ChatApp architecture

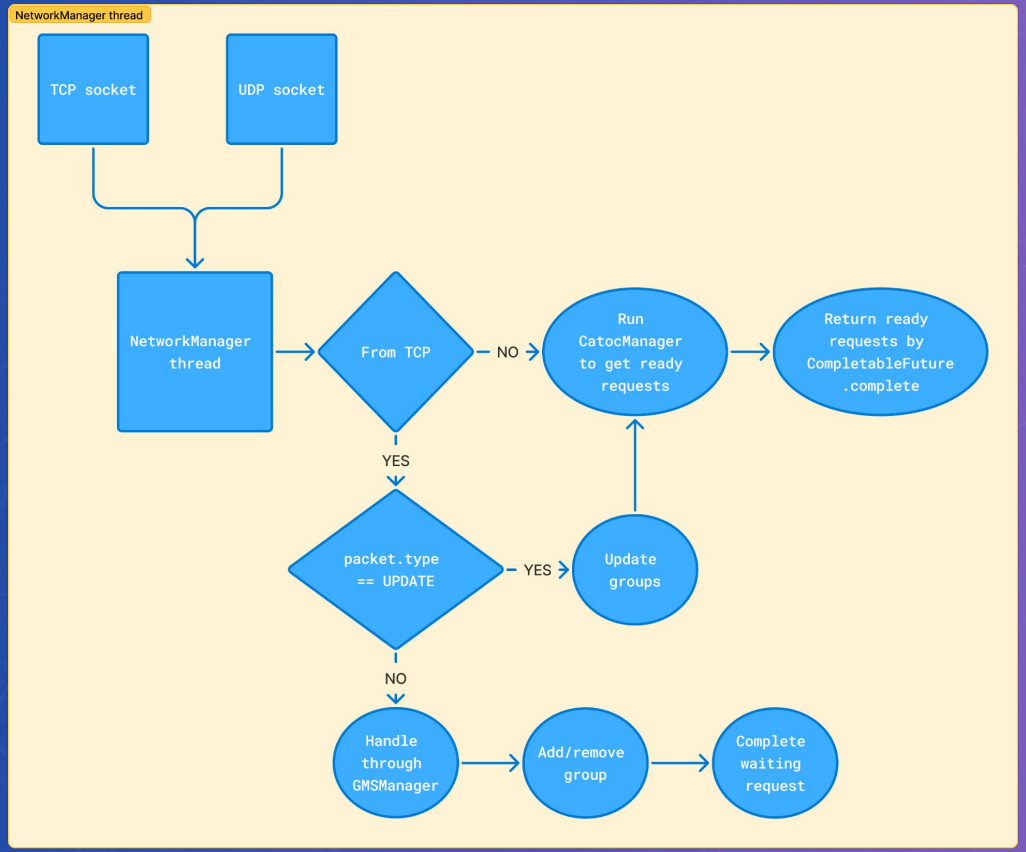


Group communication API

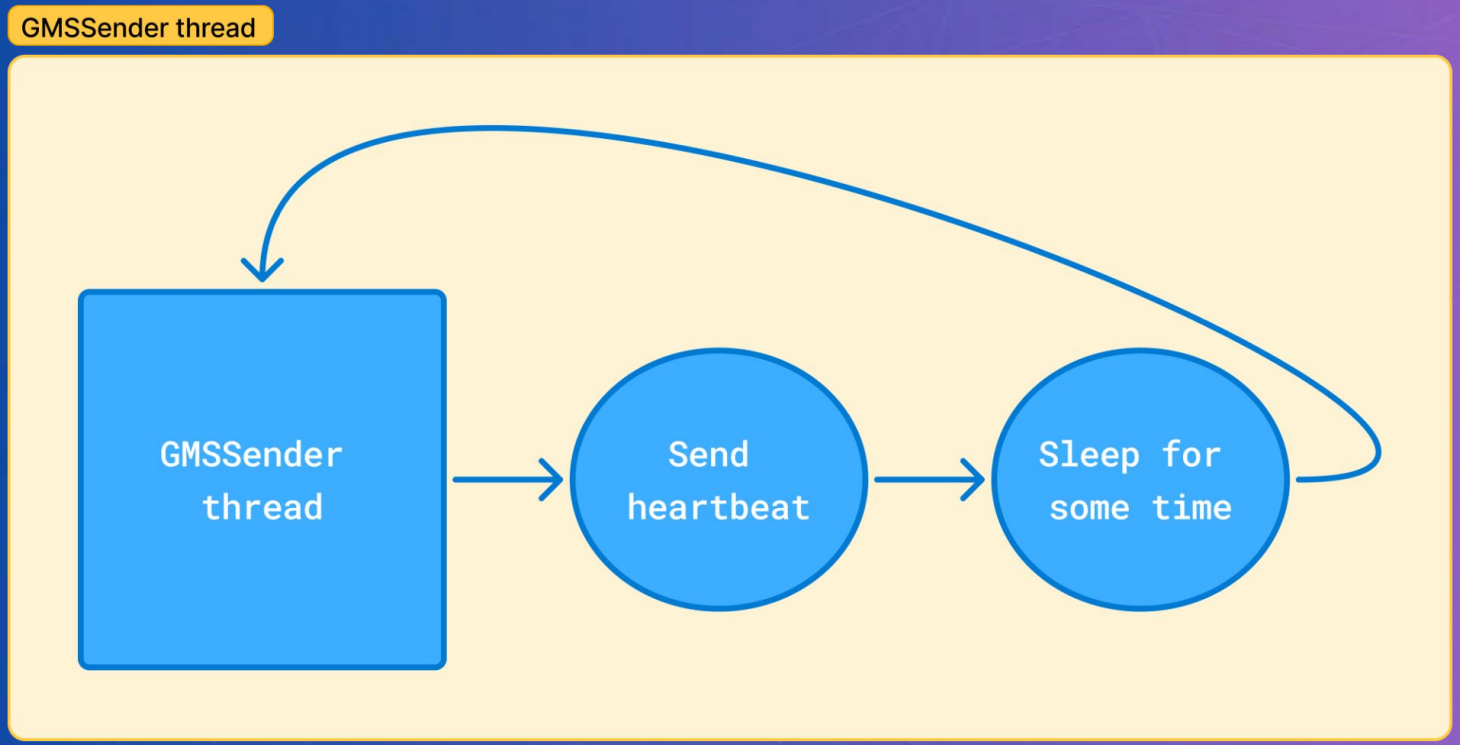
```java
public GroupAPI(String gmsAddress, int gmsPort, Function<ByteBuffer, SerializableObject> deserializer);

public JoinReply join(String groupName, String username);
public LeaveReply leave(int groupId);
public int send(int groupId, SerializableObject message);
public GroupMessagePair receive(Set<Integer> groups);
```

Group library - Network thread



Group library - GMS sender thread



Group library - CatocManager

The `CatocManager` is split in 3 different layers that have different guarantees. Each send request passes from the top to the bottom layer and each receive request starts from the bottom layer to the top:

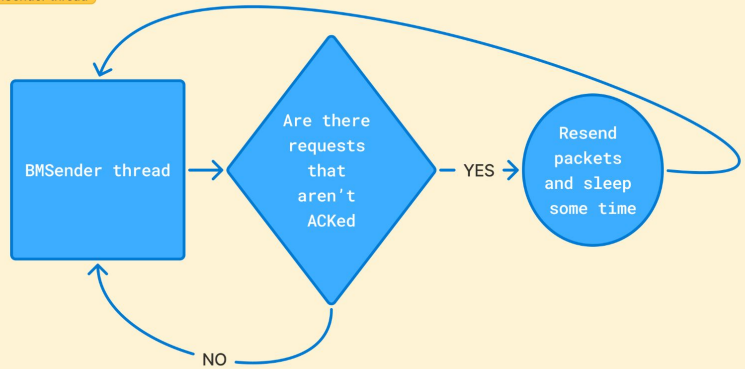
- `BMState`: The basic multicast layer that guarantees reliable delivery.
- `URMState`: Uniform reliable multicast which guarantees a message is delivered once everyone has received it and also filters duplicates.
- `TRMState`: Total multicast that guarantees total order across all members.

CatocManager - Basic Multicast

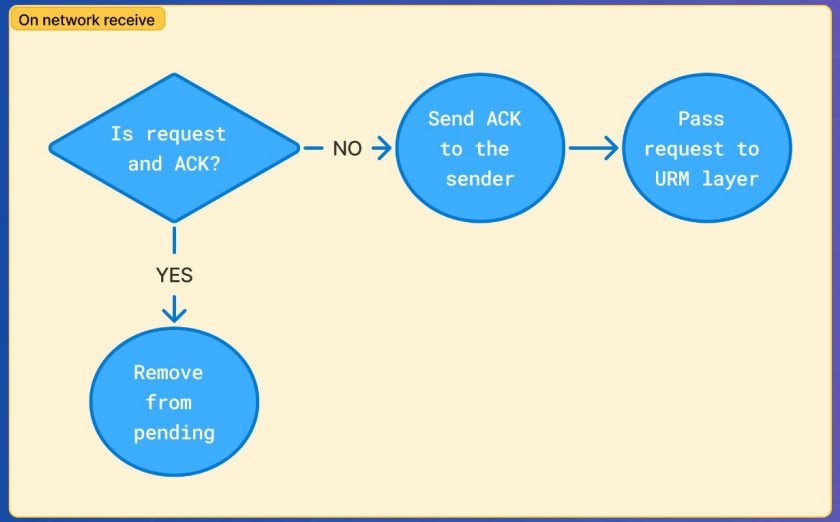
Fields

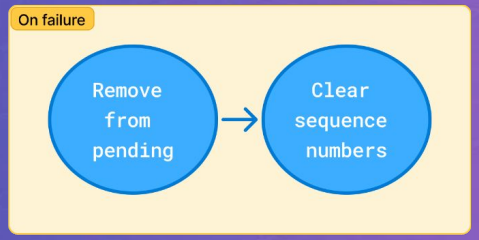
Type	Field	Description
`Map<String, Integer>`	`sequenceNumbers`	Highest sequence number for each client
`Map<String, Map<Integer, URMRequest>>`	`pending`	All pending packets for a client
`Object`	`conditional`	Conditional for waking up BMSender

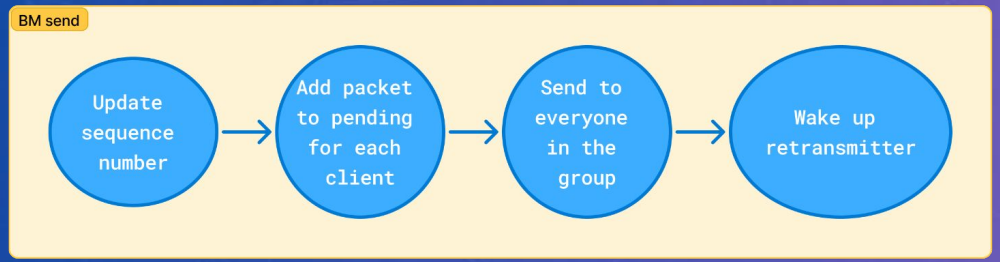
BMSender thread



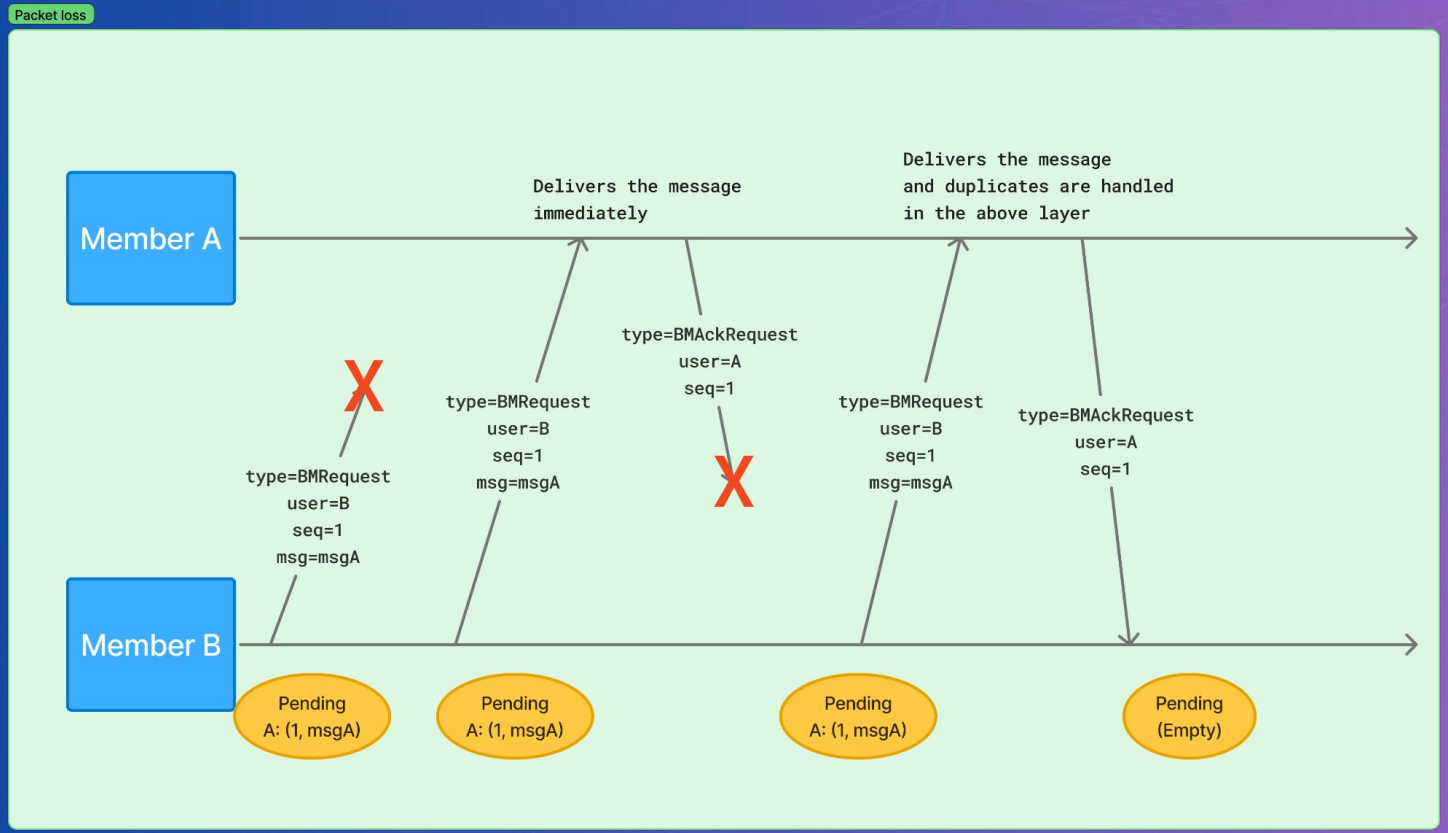
CatocManager - Basic Multicast







CatocManager - Basic Multicast

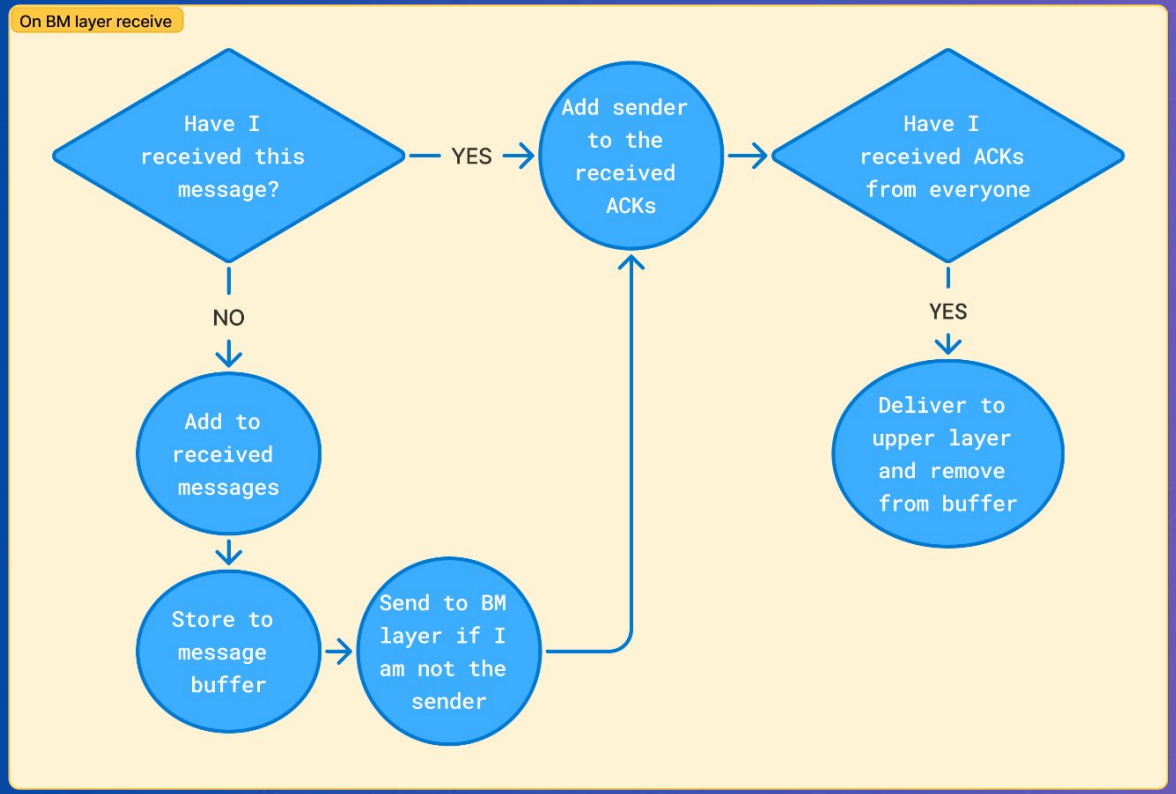


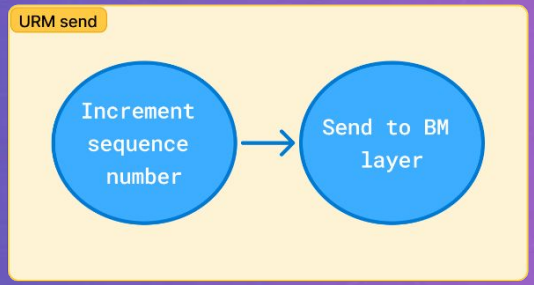
CatocManager - Uniform Reliable Multicast

Fields

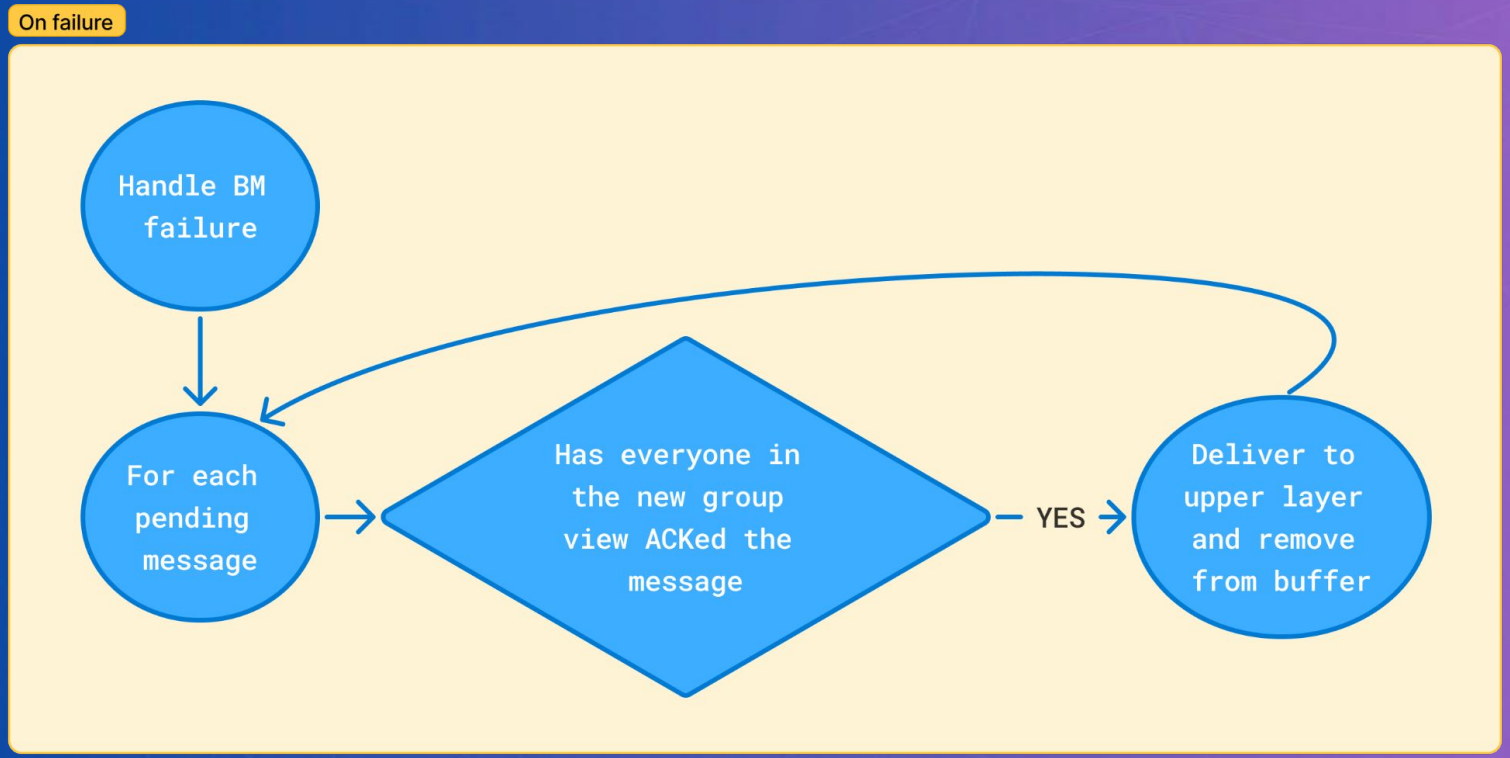
Type	Field	Description
`int`	`sequenceNumber`	Current sequence number for the client
`Set<String>`	`receivedMessages`	All IDs of the received messages
`Map<String, Set<String>>`	`receivedAcks`	Received ACKs for each message ID and for each client
`Map<String, Object>`	`messageBuffer`	Buffer to hold not ready messages

CatocManager - Uniform Reliable Multicast

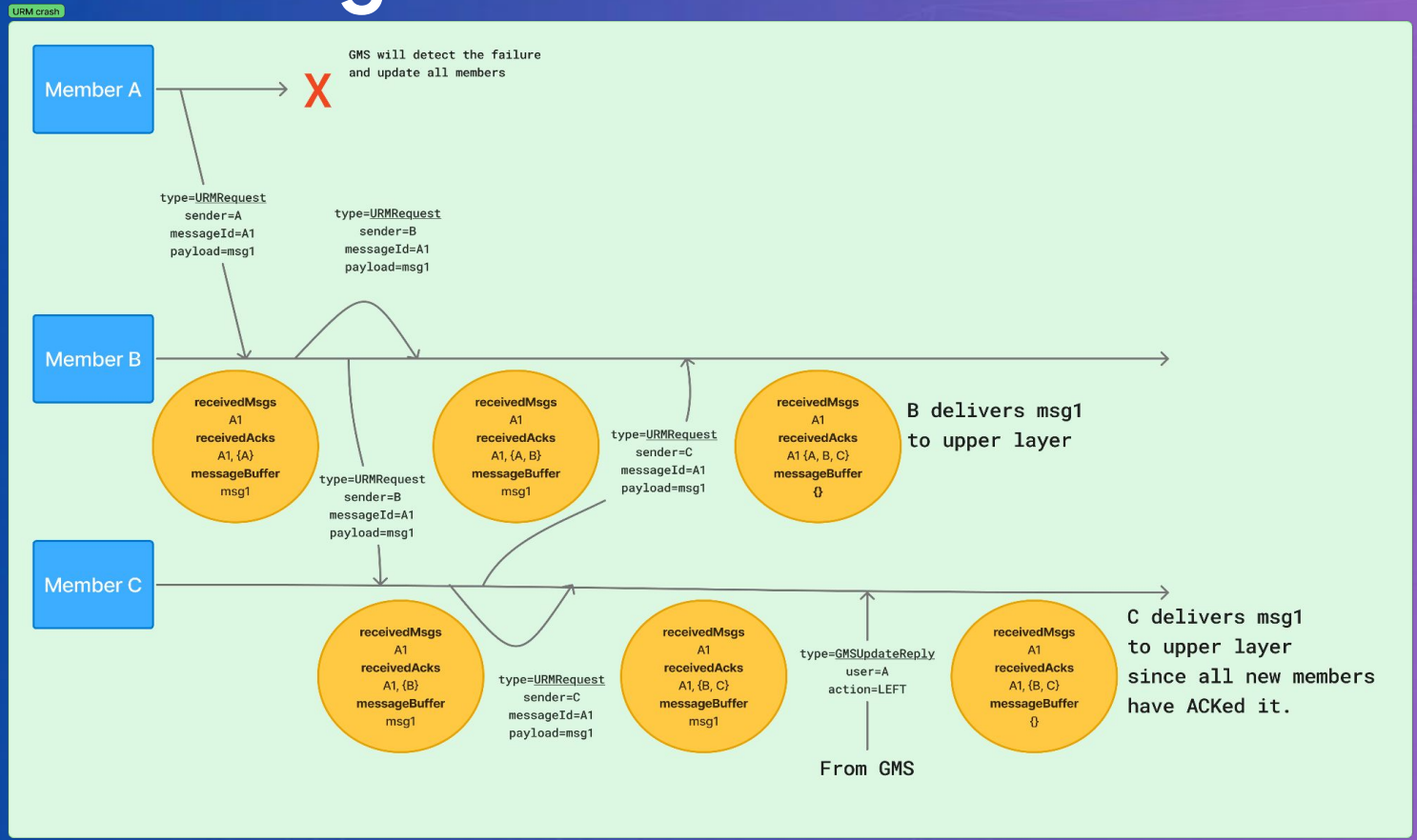




CatocManager - Uniform Reliable Multicast



CatocManager - Uniform Reliable Multicast

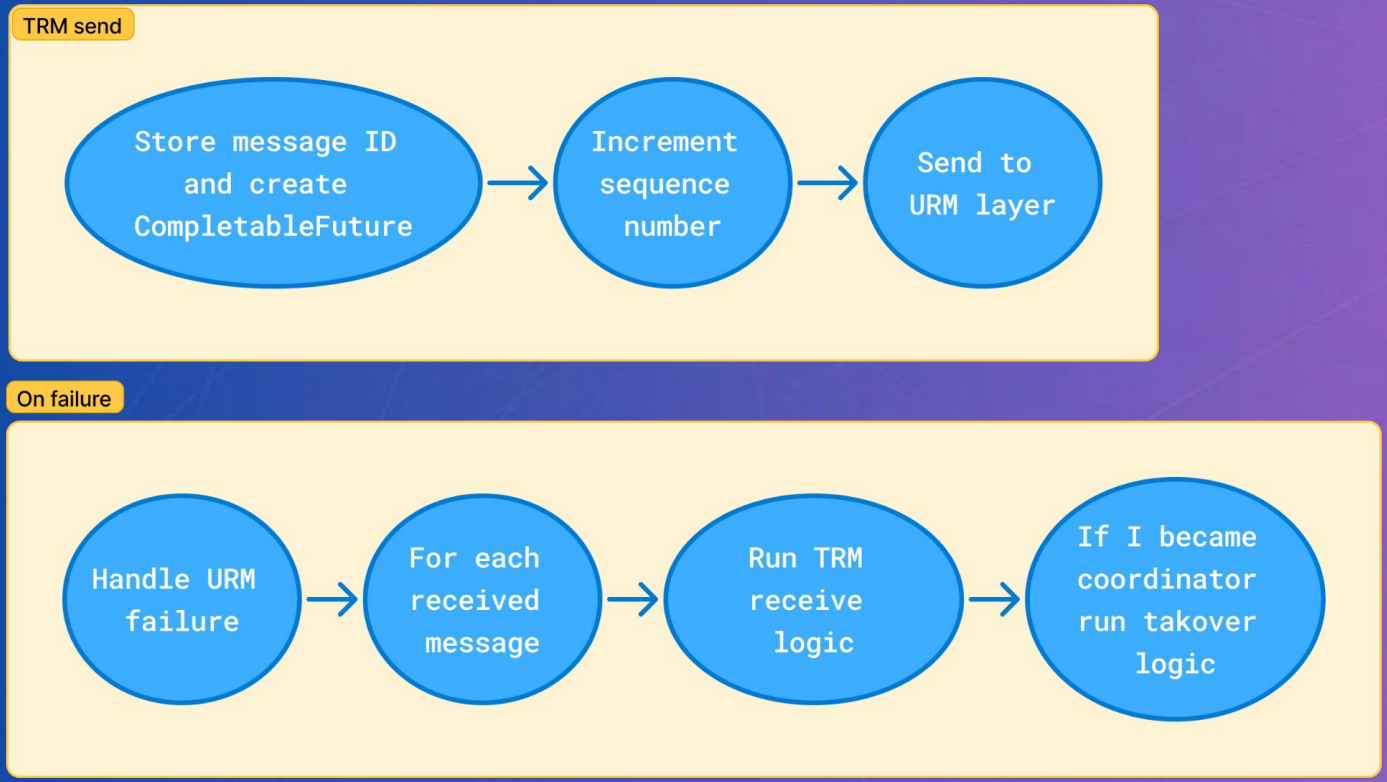


CatocManager - Total Reliable Multicast

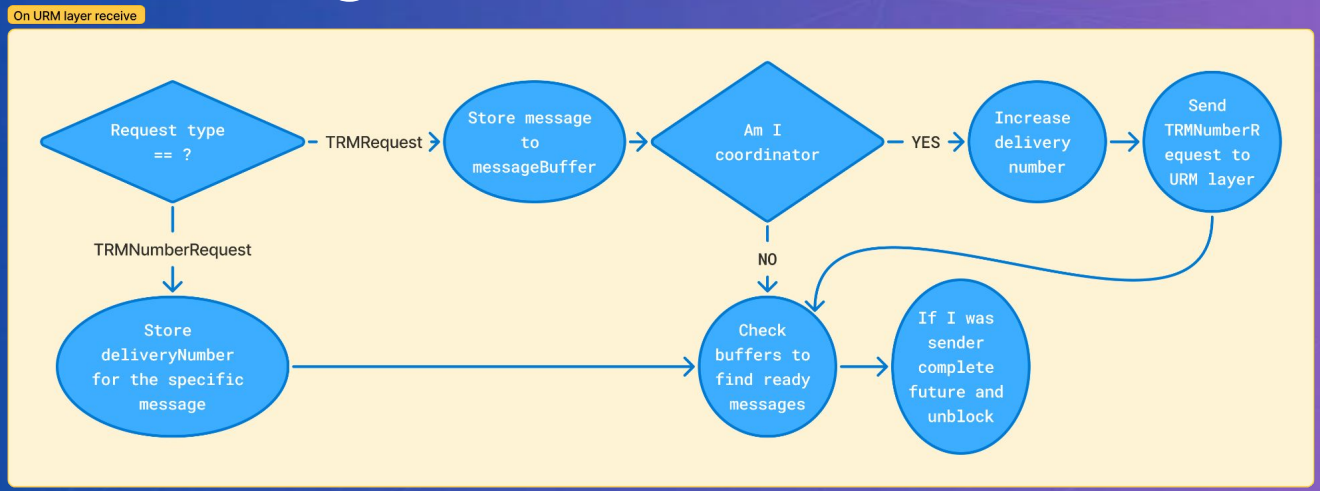
Fields

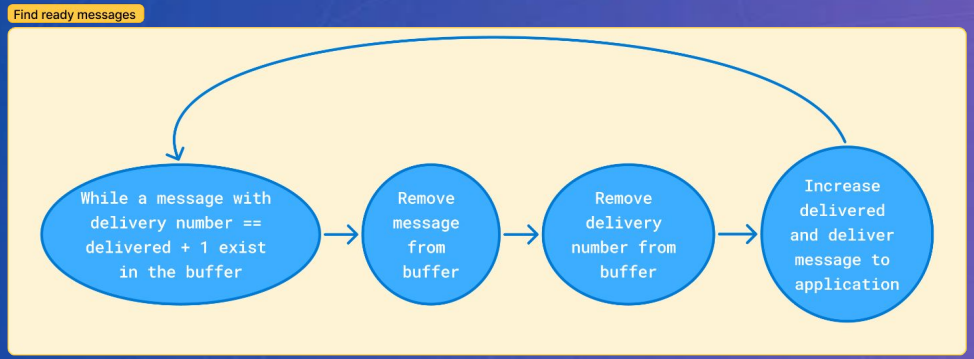
Type	Field	Description
int	sequenceNumber	Highest sequence number of the client
int	delivered	Count of delivered messages
int	deliveryNumber	Last delivery number got from coordinator
Map<String, SerializableObject>	messages	Messages waiting to receive delivery number
Map<Integer, String>	deliveryOrder	Delivery numbers waiting to receive the object
String	sendRequestId	Message ID of the last message sent by this client
CompletableFuture<Integer>	sendRequest	Future to complete when the last sent message of this client is received

CatocManager - Total Reliable Multicast

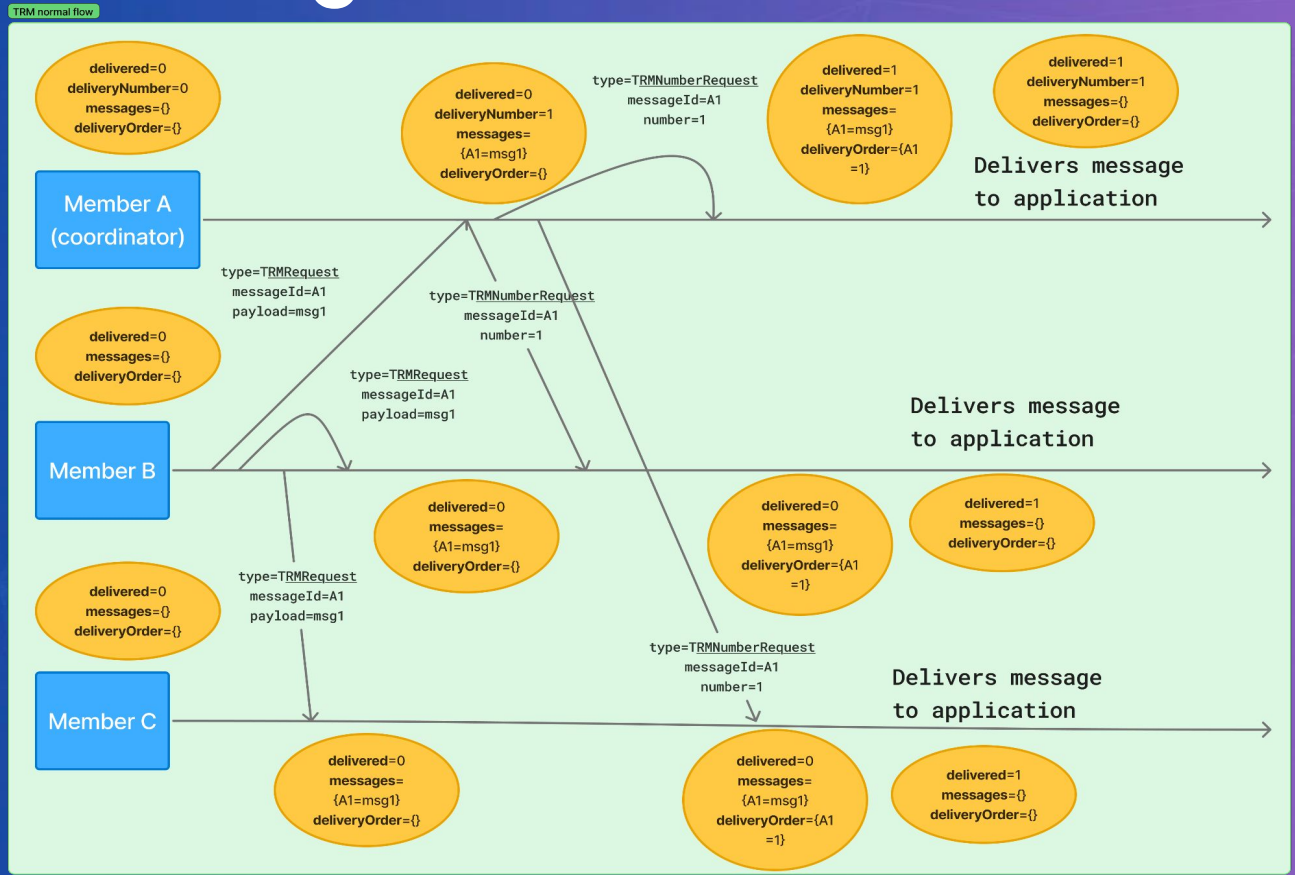


CatocManager - Total Reliable Multicast

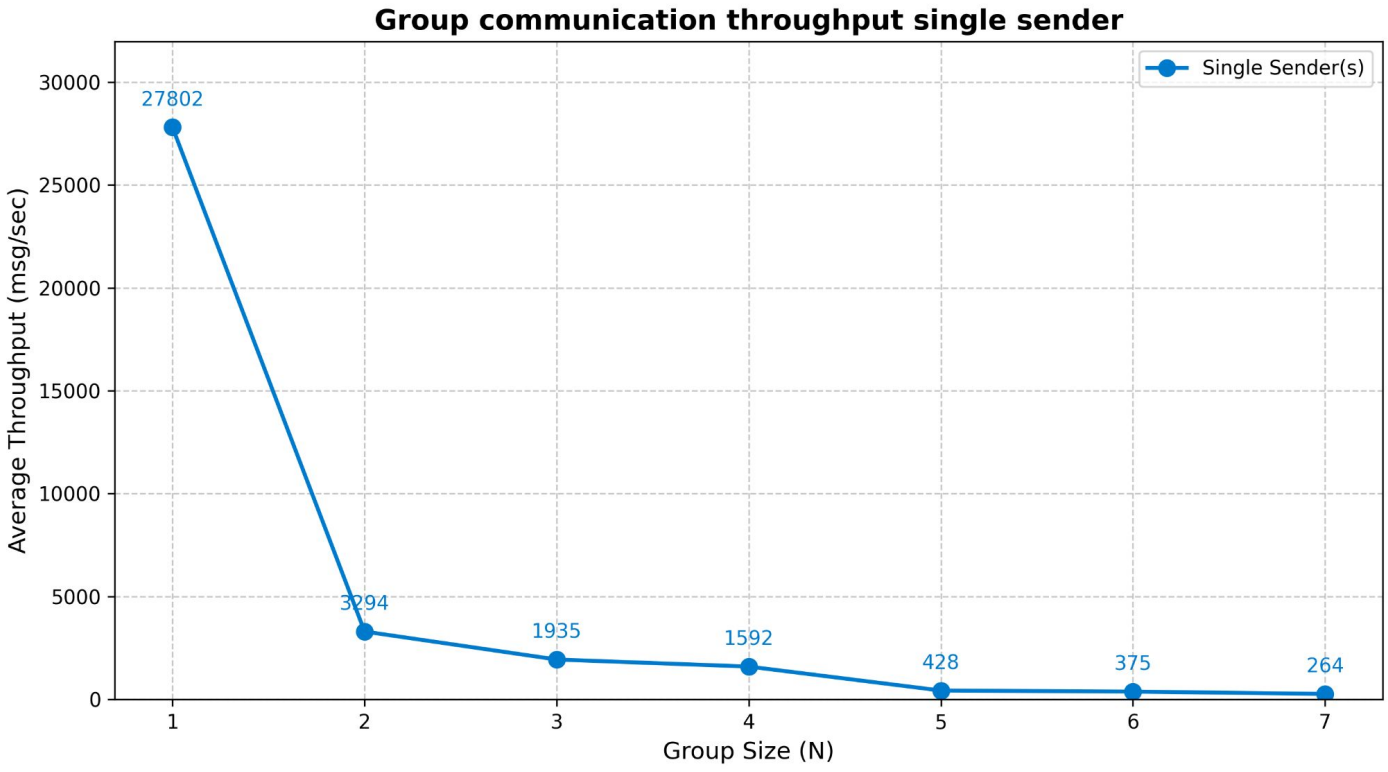




CatocManager - Total Reliable Multicast

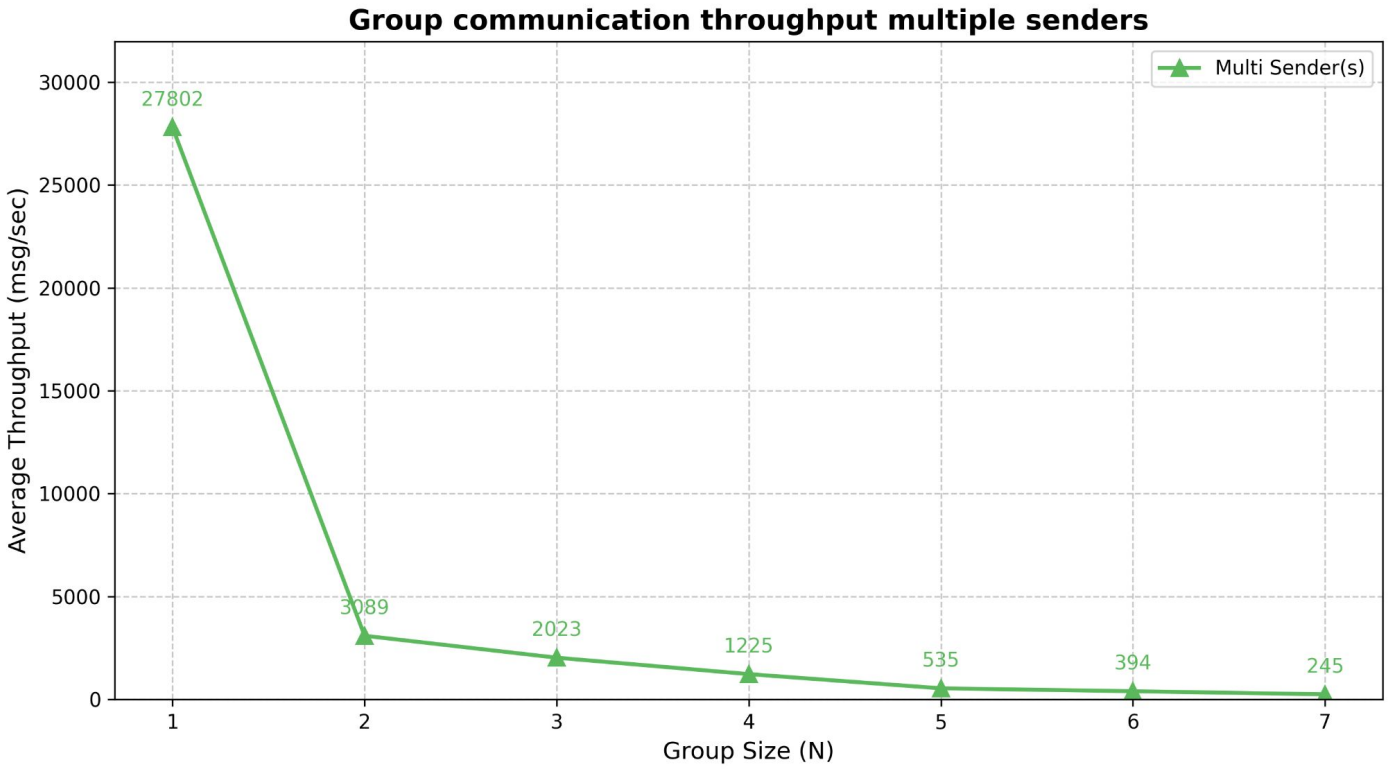


Performance - Single sender



** Data points represent the average of all iterations. Global min and max outliers are discarded if ≥3 total runs.*

Performance - Server throughput



** Data points represent the average of all iterations. Global min and max outliers are discarded if ≥3 total runs.*

Performance - Theoretical message count

For each message we need to send the following packets in each layer:

- `BM layer`: At best (no losses), N packets for the request and N packets for the ACK. Total 2N packets
- `URM layer`: The sender sends to N - 1 relayers with BM which costs 2N and then each of the N - 1 relayers sends their own URM packet with BM which costs another 2N each. So in total: 2N + 2N (N - 1) = 2N + 2N^2 - 2N = 2N^2
- `TRM layer`: For each application message we need 2 TRM messages, one for the request and one for the delivery number. Each are sent using URM so they cost 2N^2.

In total, for a single application we need to send: 2 * (2N^2) = 4N^2

For example, in a group of 4 members sending 10000 messages each we theoretically have: 4 * 10,000 * 4(4^2) = 2,560,000 packets

Performance - Actual message count

Group size of 4 (10000 messages each)			
Member	**Messages sent**	**Packets sent**	**Packets/Msgs**
1	10000	642724	64.2724
2	10000	642484	64.2484
3	10000	642418	64.2418
4	10000	642748	64.2748
Total	40000	2570374	