Assignment 1

Request-reply protocol

Packet structure

Packet structure

Field	Type	Size	Description
`type`	`MessageT`	4 bytes	Type of message
`clientId`	`unsigned int`	4 bytes	Unique client ID for the specific server
`serviceId`	`unsigned int`	4 bytes	The targeted service by the request
`sequenceNumber`	`unsigned int`	4 bytes	Sequence number for this client
`payloadLength`	`unsigned int`	4 bytes	Length of the following buffer
`payload`	`void *`	8 bytes	Payload buffer

Message types

MessageType enum

Value	Description
REQUEST	Used by client to send a request to the server
REPLY	Used by server to reply to the client
EXPIRED	Used by server when a client sends a duplicate request for an expired reply
NOT_SERVING	Used by server when it's not serving a specific service
ACKNOWLEDGE	Used by client to acknowledge a reply so that the server can clear the cache
CANCEL	Used by client to cancel any pending requests after a restart

Request structure

Request structure

Field	Type	Description
`packet`	`PacketT`	Packet containing all the information
`id`	`unsigned int`	Local request ID
`clientAddr`	`unsigned int`	Contact info (IP and port) of client
`clientAddrLen`	`unsigned int`	Length of previous field

Data structures

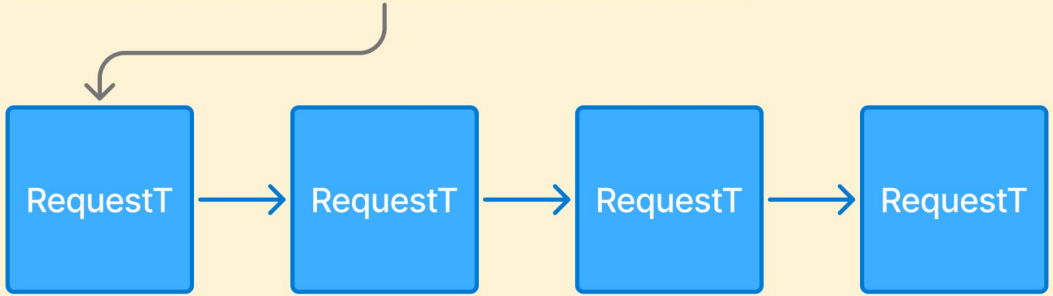
2 different types of lists were used:

- `RequestListT`: Lists containing requests and necessary synchronization tools
- `ServiceListT`: Lists containing services and necessary synchronization tools

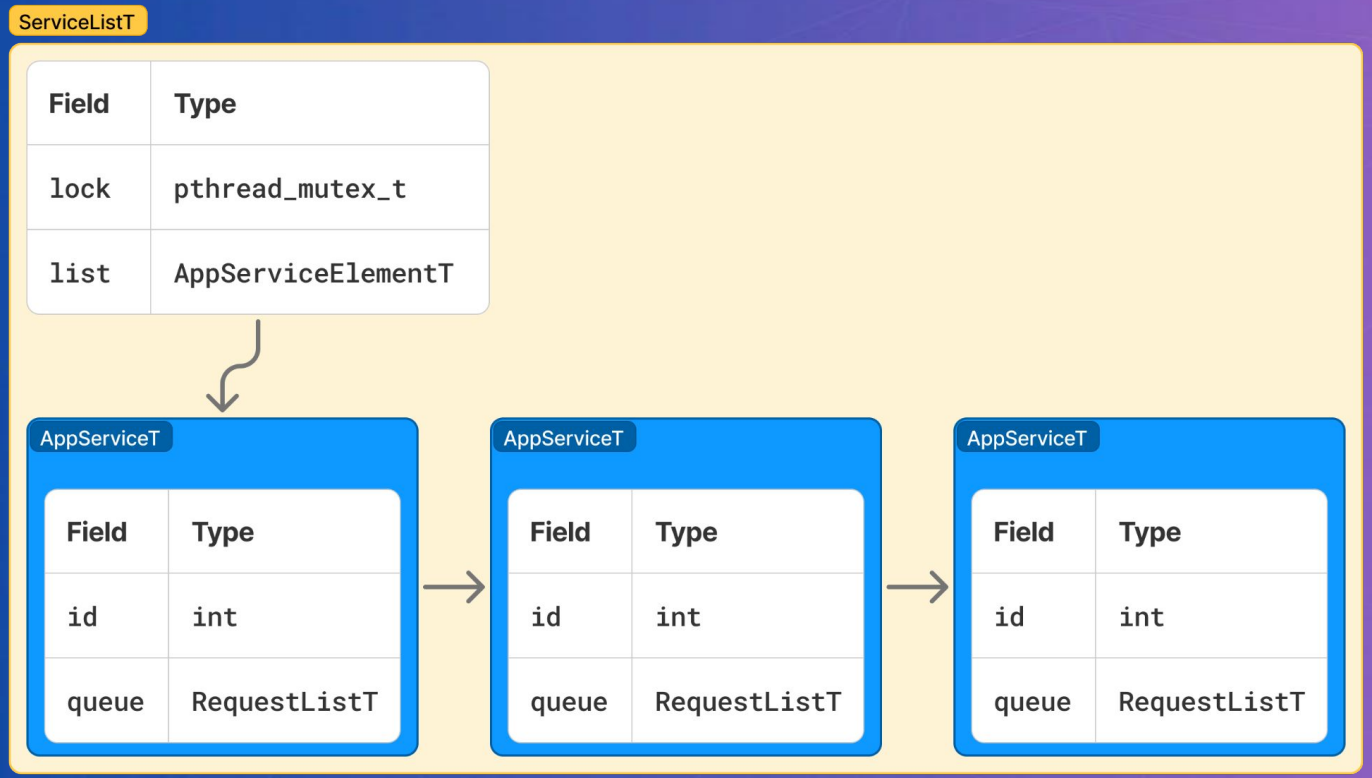
Data structures - `RequestListT`

RequestListT

Field	Type	Used for
`lock`	`pthread_mutex_t`	Mutual exclusion on the list
`cond`	`pthread_cond_t`	Waiting when list is empty
`list`	`RequestElementT`	Finding requests



Data structures - `ServiceListT`



Data structures

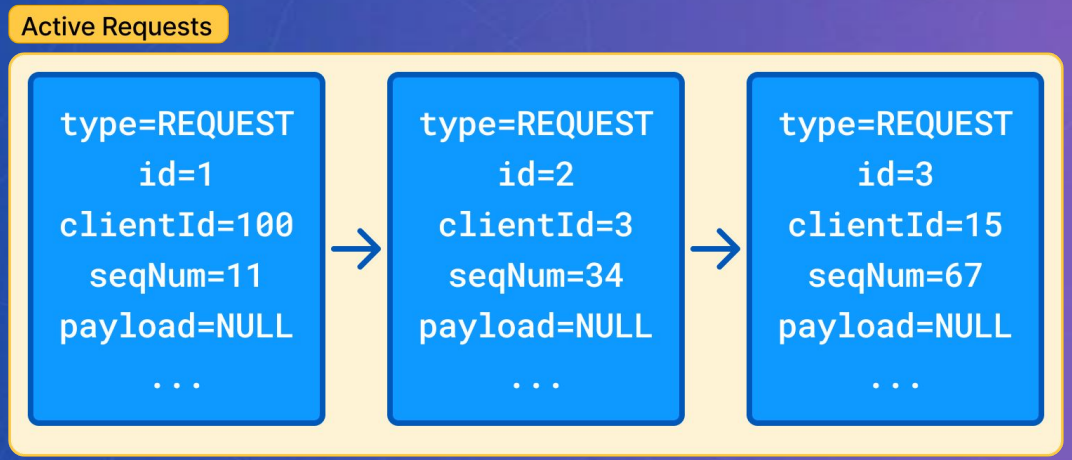
3 different linked lists were used for storing requests in different phases and 1 linked list was used to stored registered services by the server:

- `serviceRegistry`: Stores all the registered services along with its ID and pending requests
- `activeRequests`: Stores all the requests that are currently being processed by the server.
- `cachedRequests`: The memory of the server which also temporarily stores responses.
- `replies`: Stores replies ready to be sent back to the client by the reply thread.

Data structures - `activeRequests`

Contains all the requests that have been removed from the corresponding service's queue by the application calling `getRequest` for the specific service.

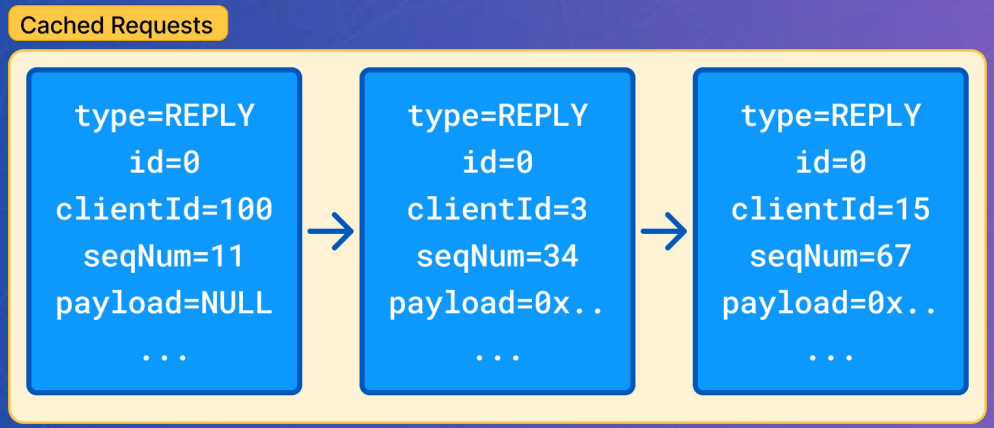
- Payload is no longer stored in the `payload` field because it's been delivered.
- All requests in this list contain a valid local ID



Data structures - `cachedRequests`

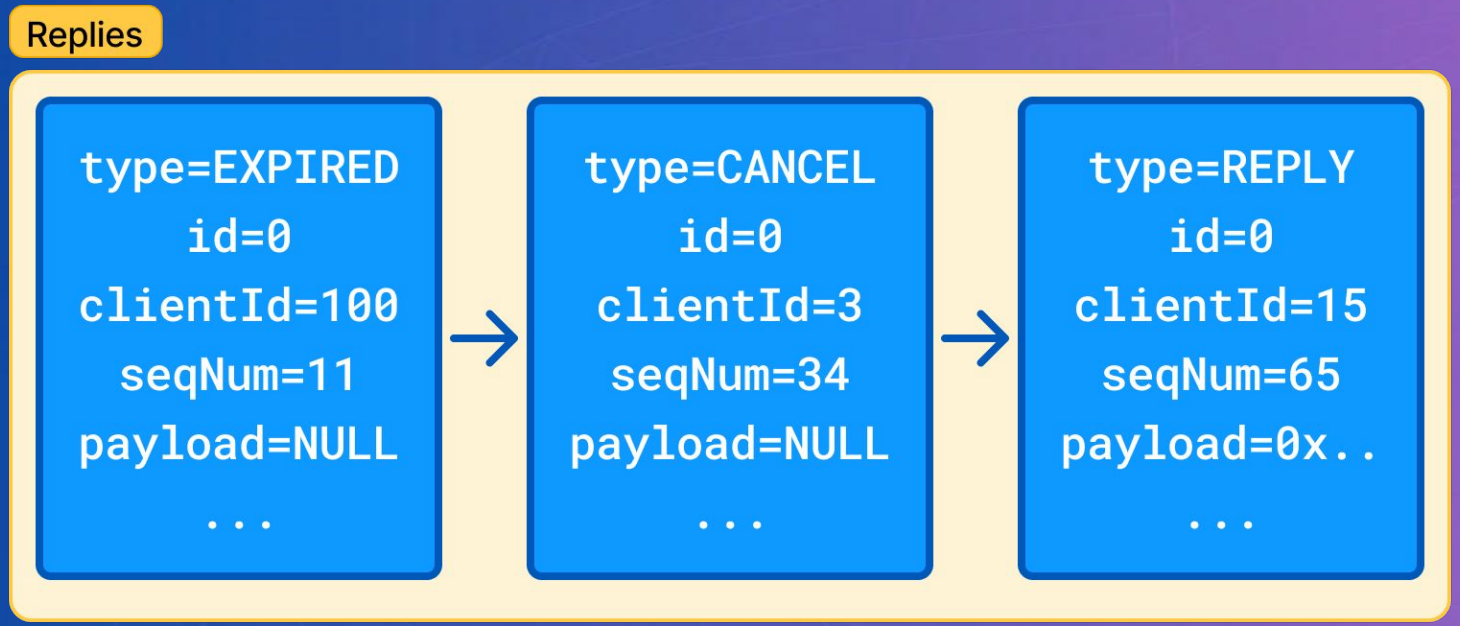
Acts as the memory of the server.

- Contains one entry per client with the latest sequence number received from the client.
- May contain the actual payload for the last reply, but it may have timed out or ack'ed.
- All entries are fetched from the persistent memory on initialization



Data structures - `replies`

Contain all the replies ready to be sent back to the client.

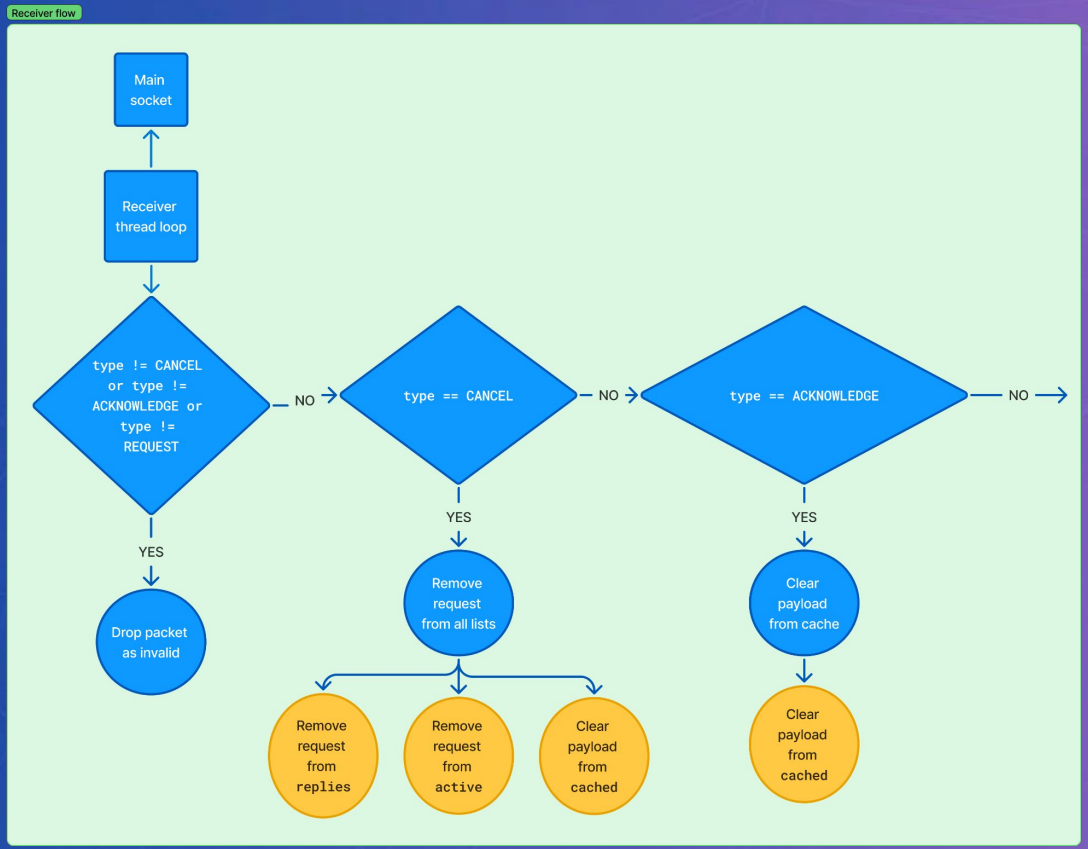


Server - Architecture

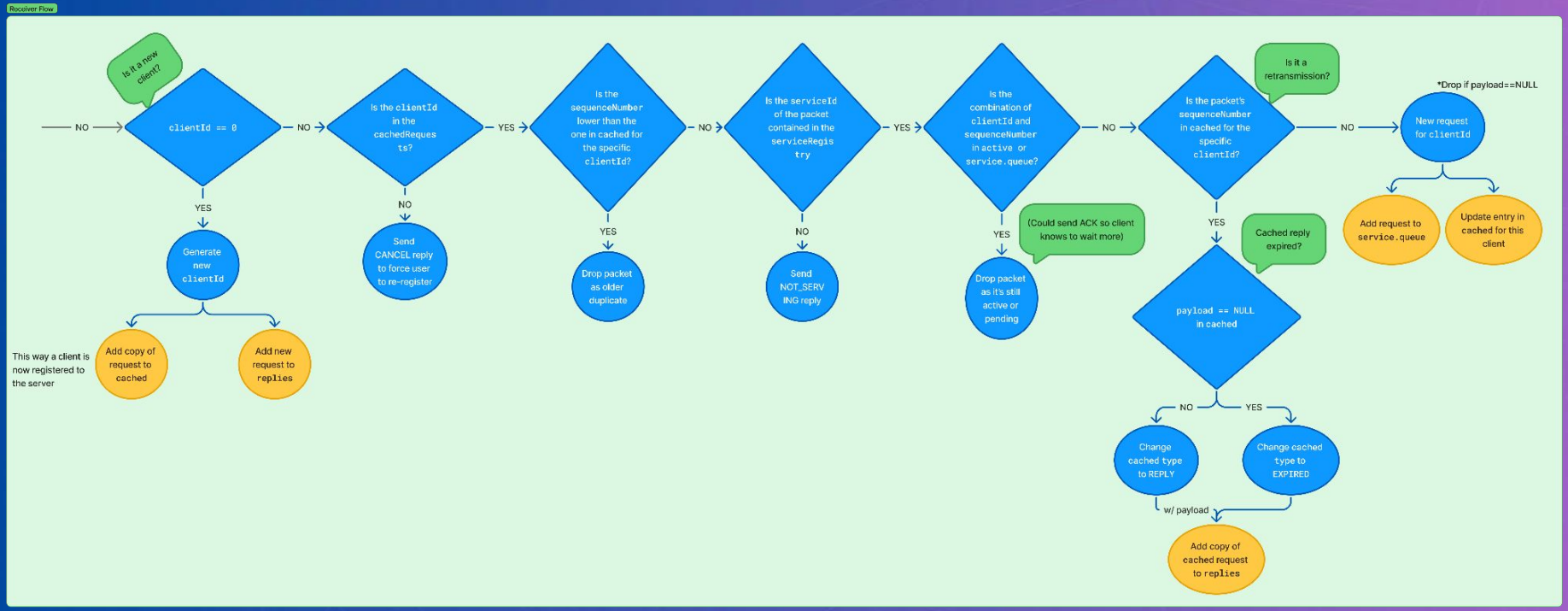
The server API uses 3 separate threads for receiving requests, sending replies and handling the discovery:

- Receiver: Listens on a randomly selected port, deserializes any received packet and processes it according to its header
- Sender: Waits on the `replies` request list until it isn't empty, pops a request from the list, serializes it and sends it back to the client
- Discovery: Listens on a different predefined port for the message "`DISCOVERY`" and responds with "`DISCOVERY_RPL`" followed by the main port.

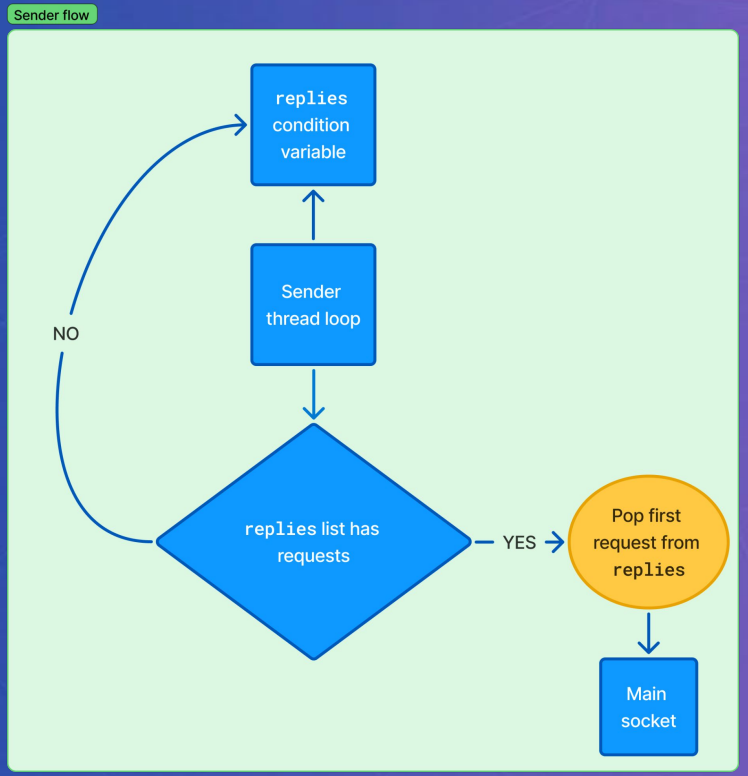
Server - Receiver (Part 1)



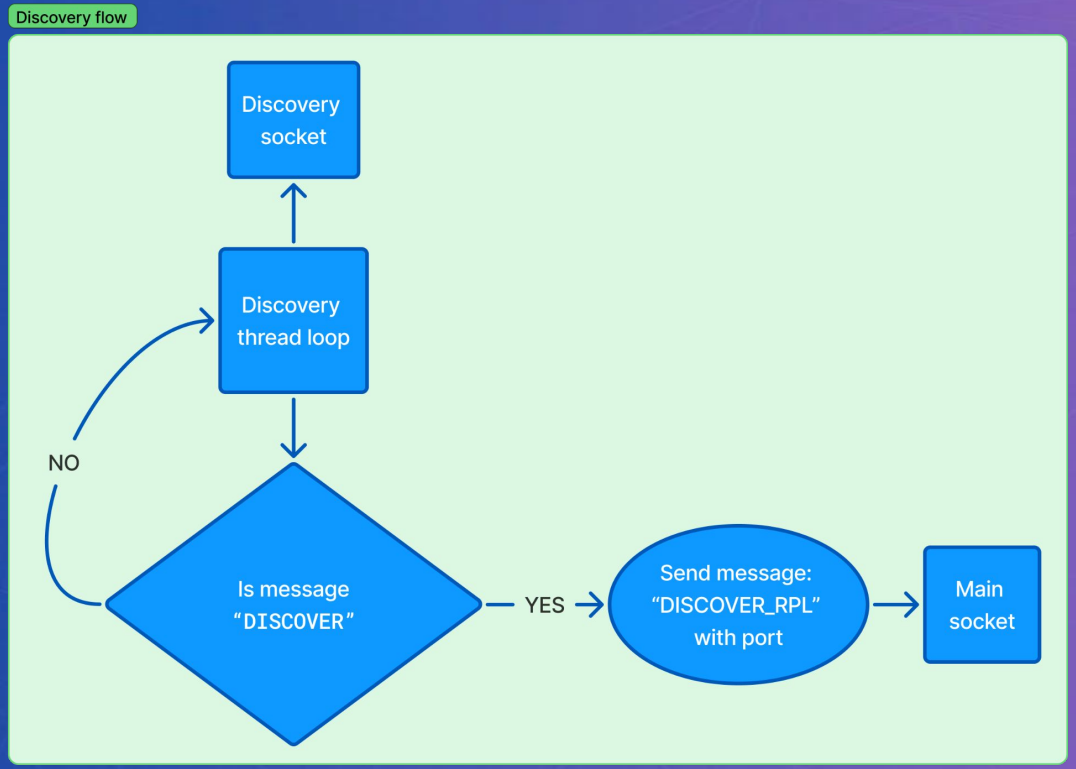
Server - Receiver (Part 2)



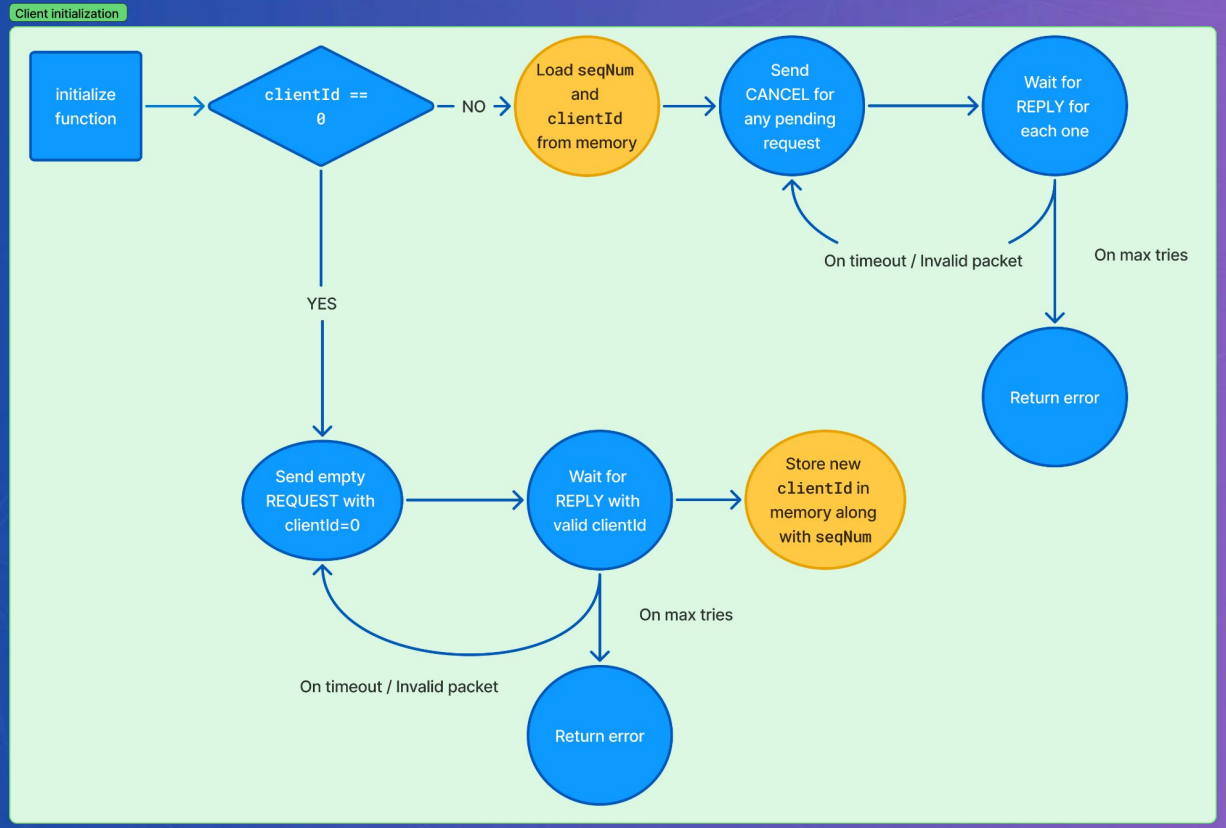
Server - Sender



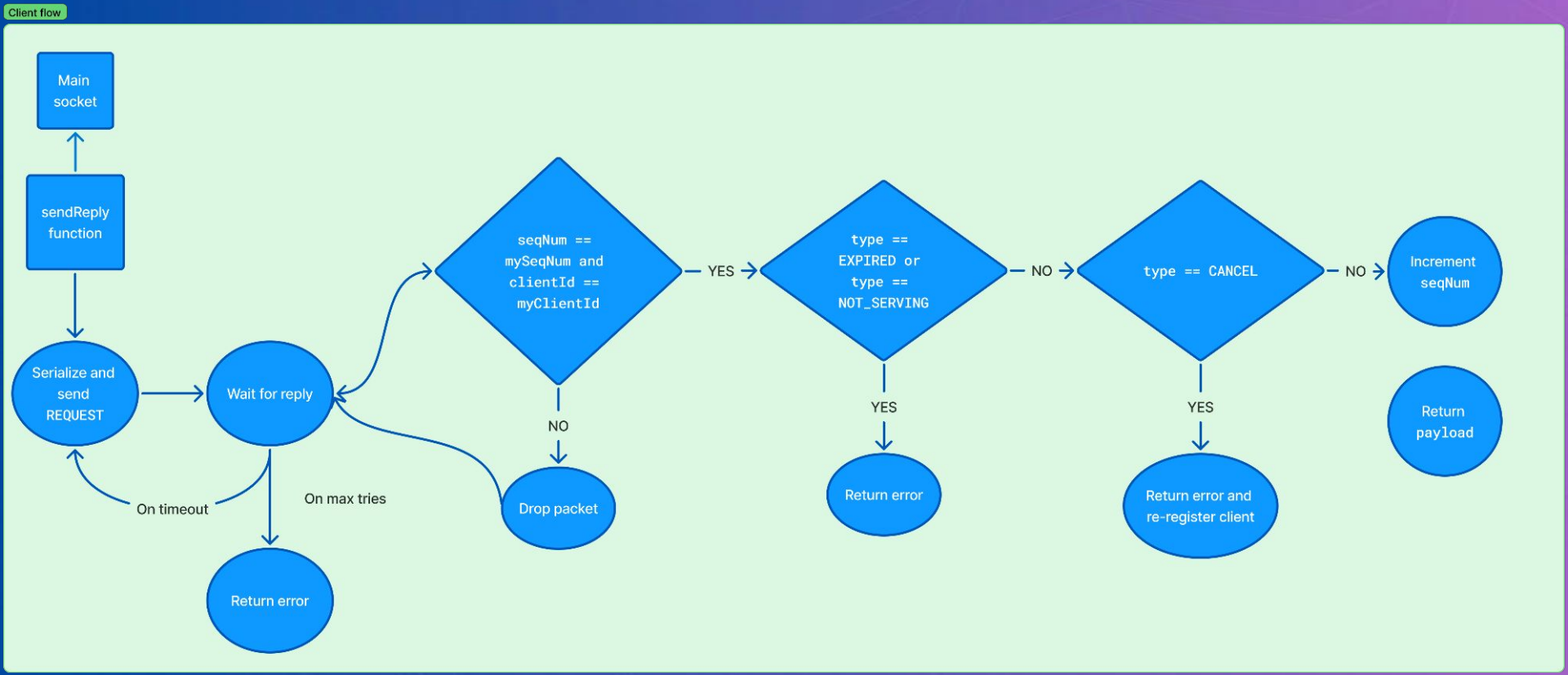
Server - Discovery



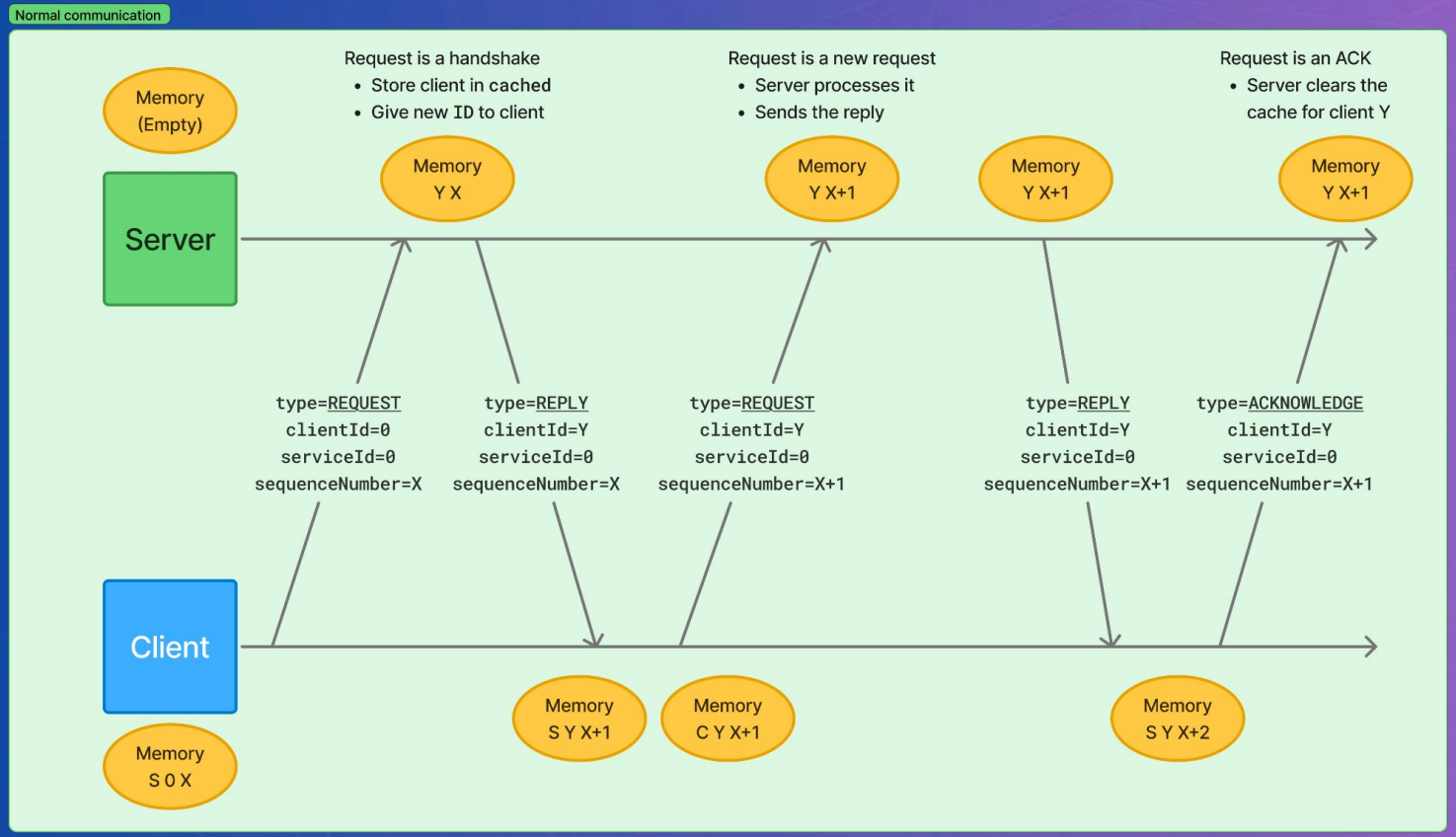
Client - Initialization and discovery



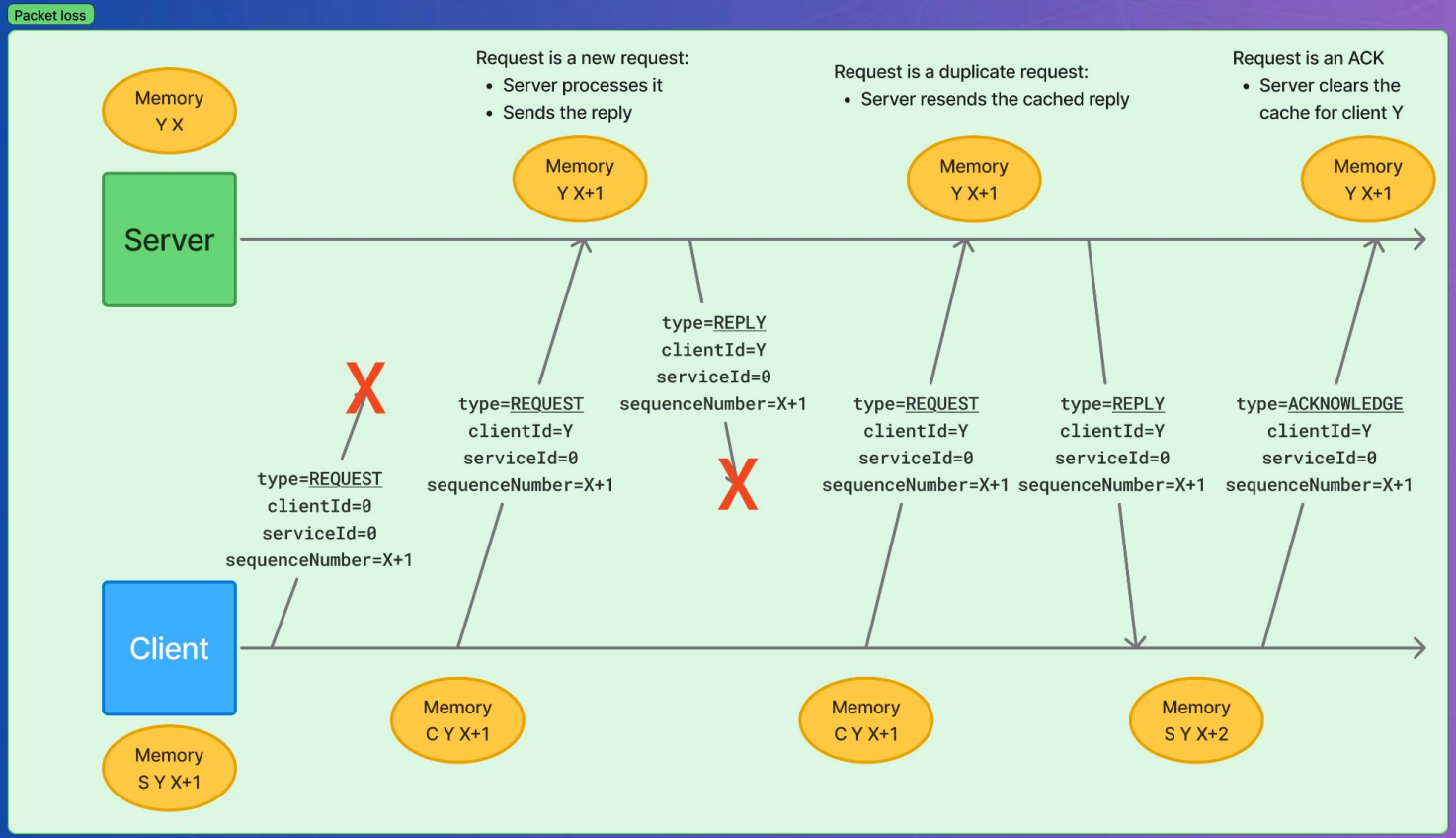
Client - Send/Receive flow



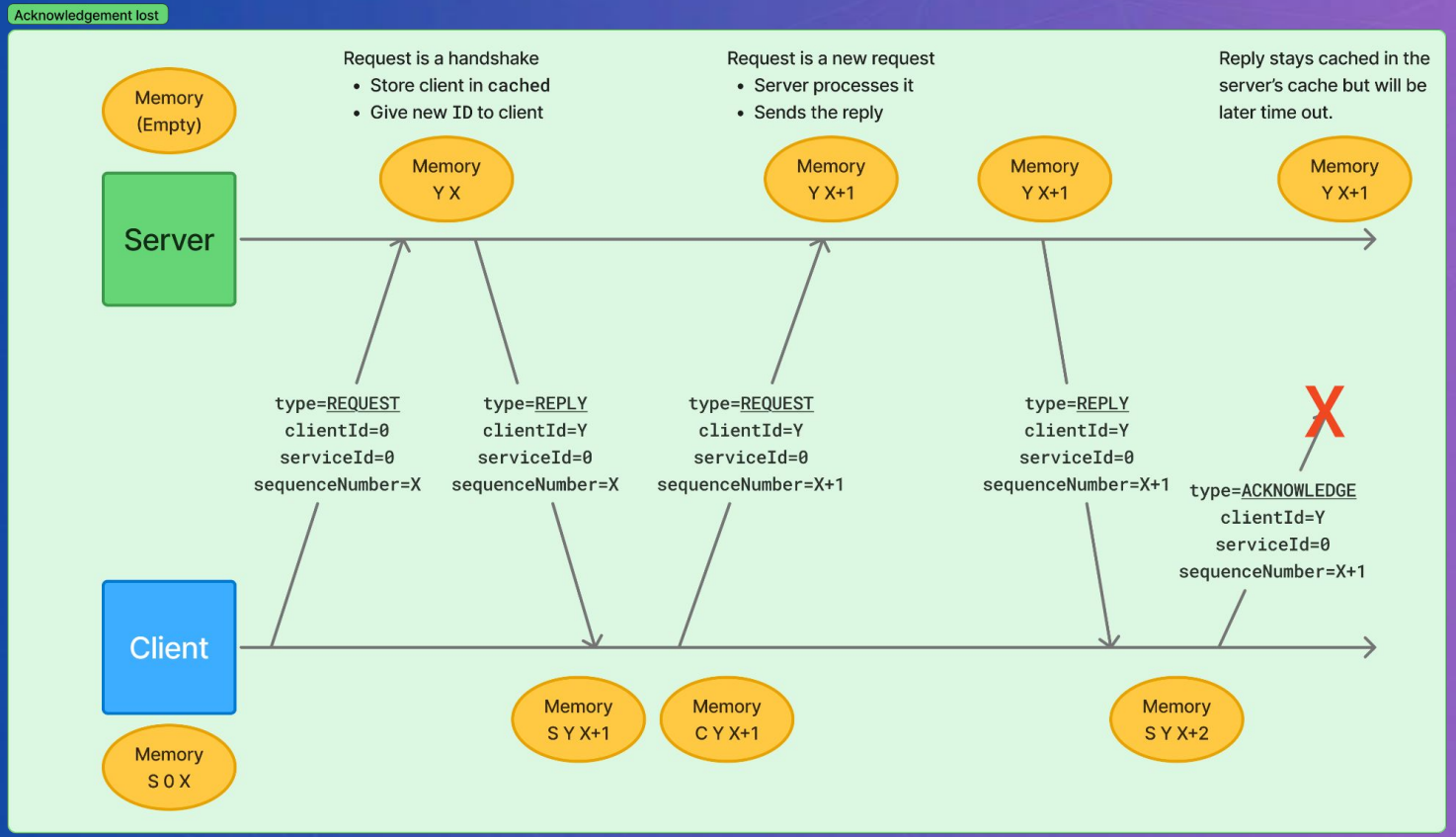
Example: Normal communication



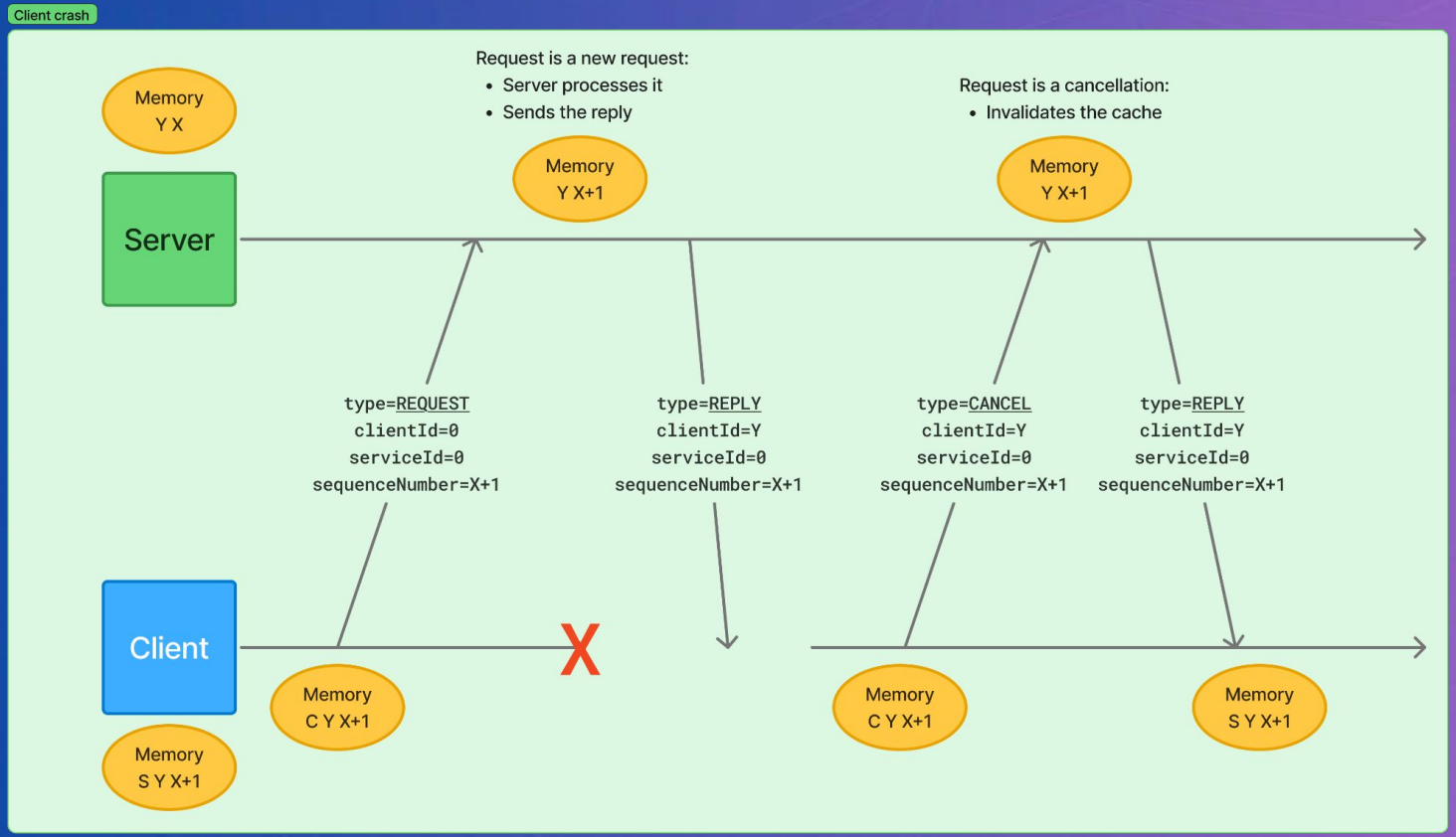
Example: Packet lost



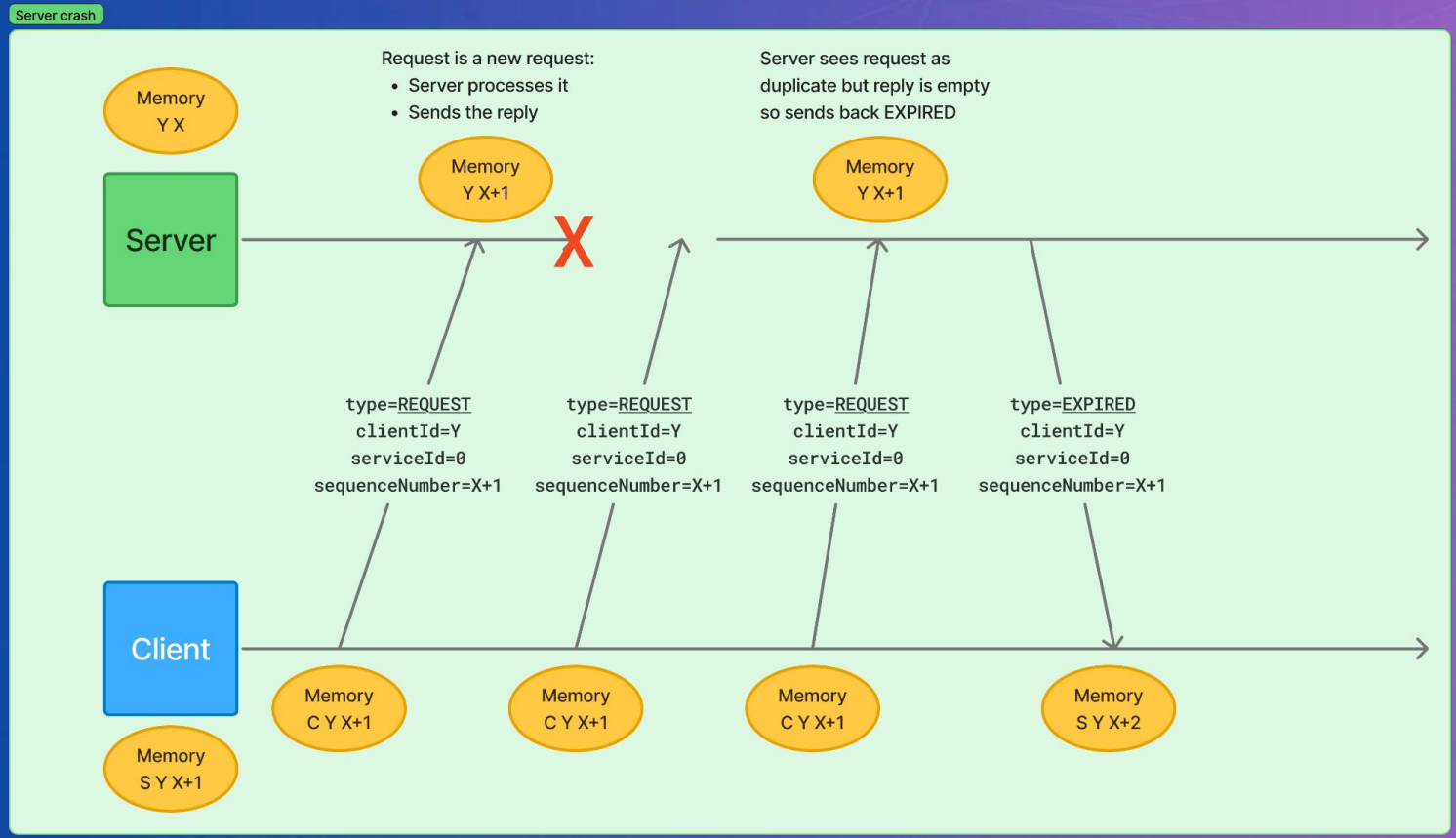
Example: Acknowledgement lost



Example: Client crash



Example: Server crash



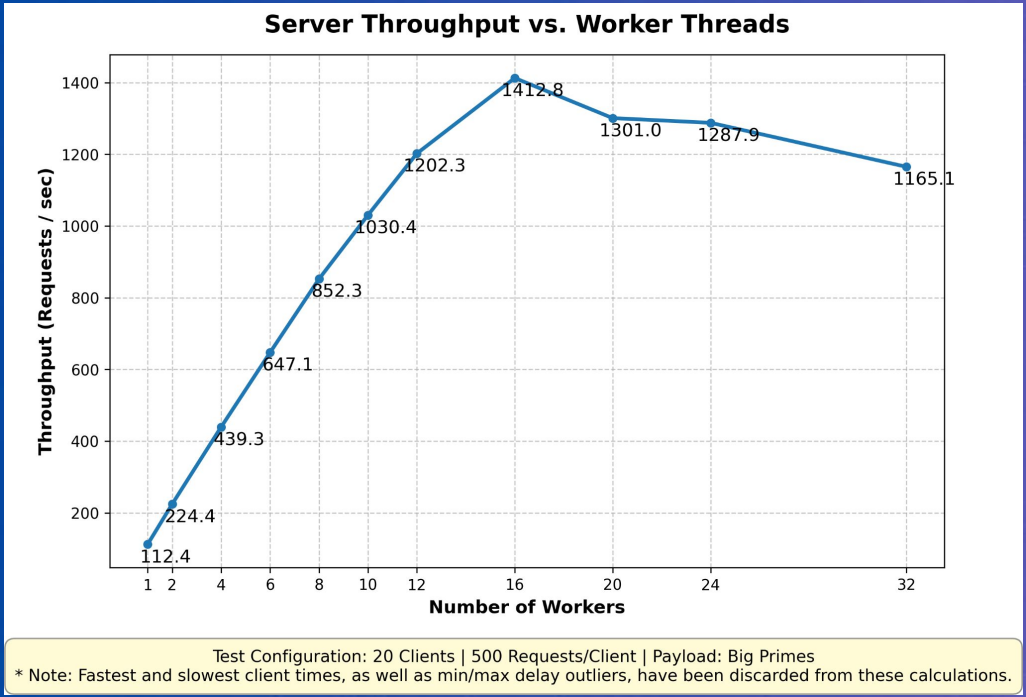
Performance - Server latency

Null Request-Reply Interaction Latency

Setup	Iterations	Successes	Avg Delay (ms)	Min / Max (ms)	Std Dev (ms)
Single	1000	1000	1.324	1.176 / 3.215	0.231
Multi	1000	1000	2.148	1.198 / 9.640	1.324

* Note: The minimum and maximum individual delay outliers have been discarded from these metrics.

Performance - Server throughput



Test Configuration: 20 Clients | 500 Requests/Client | Payload: Big Primes
* Note: Fastest and slowest client times, as well as min/max delay outliers, have been discarded from these calculations.

Workers	Avg Delay (ms)	Min / Max (ms)	Std Dev (ms)
1	177.84	16.84 / 210.18	4.89
2	89.06	14.09 / 115.96	2.64
4	45.48	12.58 / 61.23	1.56
6	30.87	12.18 / 67.75	2.28
8	23.43	11.14 / 45.73	1.81
10	19.33	10.87 / 104.09	3.55
12	16.55	10.03 / 42.73	2.52
16	13.96	10.86 / 28.63	2.23
20	15.26	10.56 / 28.12	2.82
24	15.40	10.78 / 30.65	2.51
32	17.03	10.58 / 562.60	24.20