Build a Dynamic Chat Application Using Goroutines, Channels, and APIs in Go

Task:
You are tasked with building a chat application in Go that allows clients to dynamically join, send messages, and leave the chat using HTTP APIs. The core of the application should leverage **goroutines** and **channels** for concurrency.

Requirements:

1. **ChatRoom**:
   - A central chat room where multiple clients can:
     - **Join** the chat room.
     - **Leave** the chat room.
     - **Send messages** that are broadcast to all connected clients.
   - The chat room should be thread-safe, handling concurrent clients using **channels** and **goroutines**.
2. **Client**:
   - Each client should have:
     - A **unique ID**.
     - A **message channel** to receive messages from the chat room.
   - Clients should be able to send messages and receive all broadcasted messages.
3. **Concurrency**:
   - Use **channels** for client interactions with the chat room.
   - Use **goroutines** to manage multiple clients concurrently.
4. **RESTful API**:
   - Implement the following HTTP endpoints:
     - **Join Chat**: Allows a client to join the chat room.
       - Endpoint: /join?id=<client_id>
     - **Send Message**: Allows a client to send a message to the chat room.
       - Endpoint: /send?id=<client_id>&message=<message>
     - **Leave Chat**: Allows a client to leave the chat room.
       - Endpoint: /leave?id=<client_id>
     - **Get Messages**: Allows a client to receive broadcast messages from the chat room.
       - Endpoint: /messages?id=<client_id>
5. **Concurrency Handling**:
   - Ensure that the chat room can handle multiple clients concurrently using goroutines and channels.
   - Safely manage access to shared data structures like the list of connected clients.

Example Usage:

- A client joins the chat by calling the /join endpoint.
- The client can send a message using the /send endpoint.
- The client can retrieve new messages using the /messages endpoint.
- A client leaves the chat by calling the /leave endpoint.

Bonus:

- Implement a timeout for the /messages endpoint so that it doesn't block indefinitely.
- Handle cases where a client leaves the chat and should no longer receive messages.

Hint: Use channels and goroutines effectively to handle concurrent clients, ensuring that the chat room's message broadcasting is thread-safe.