A mechanism for Jam Session Management

Purpose: This goal of this document is to solicit discussion and requirements for Jam Session Management.

Use Cases and Requirements:

- Allow the scheduling a jam session available for a controlled list of participants.
- Allow the scheduling of a jam session for a controlled list of participants AND a limited number of observers/visitors.
- Allow the use of a Jamulus server in a similar way to a music school rehearsal room. There can be a series of scheduled times and times for open use.
-
-
- Allow the scheduling of a jam session without being the owner or system admin of a server.
- Encourage the owner of a server to share the unscheduled time with the public.
-
-

Proposed Solution Architecture

The proposed solution requires a minimal change to the Jamulus client and server. These changes are backwards compatible to earlier versions of Jamulus.

The proposed solution adds the following functionals/modules.

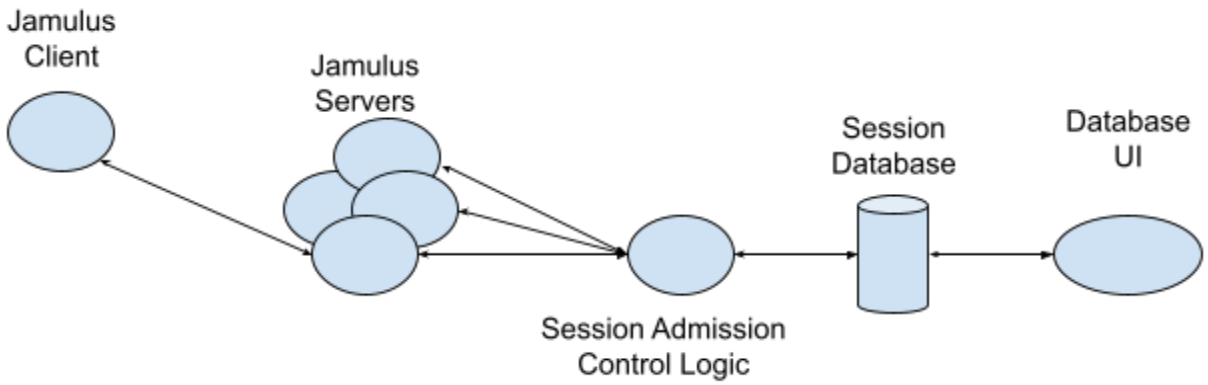
A. Jamulus client sends connection request to the Jamulus server
    a. The original connection request continues to be supported.
    b. Add a new (optional) connection request carrying username, session name, session ticket, Jamulus server
    c. Response to the client are reject, connect, or display message to user.
B. Jamulus server receives connection request from a Jamulus client.
    a. It is optional for the Jamulus server to be configured to communicate with a Jam Session Manager.
    b. If the Jamulus server does not use a Jam Session Manager, then only the original connection protocol is supported.
    c. If the Jamulus server is configured to communicate with a Jam Session Manager, then the new connection request (sent by the client) is forwarded to the Jam Session Manager.

i. The Jam Session Manager will reply with connect, reject, or send a message to the user. (For this discussion, this message is a short HTML message. Example: the session ticket is not recognized, please resend.)

C. Session admission control logic.
   a. This module receives a connection request from the Jamulus client and looks up defined sessions to determine if a user is a participant.

D. Planned session database
   a. This is a database of planned sessions.
   b. This database could be shared by many servers.
   c. Common elements for the database: server name, session name, session start time, session stop time, <userid, session credential>
   d. For simplicity, this version does not address the handling of session stop times.

E. Session creator
   a. This could be a web front end to the planned session database.



Recommended implementation

The author believes all the use cases for session admission control can be mapped into a RADIUS database. The demands on the database are minimal. Even a simple CSV list, and XLSX table, or a LDAP table could suffice.

The key component, the Session Admission Control Logic, can be implemented with FreeRADIUS. The protocol between the Jamulus servers and FreeRADIUS are available in Linux as a PAM (Pluggable Authentication Module).