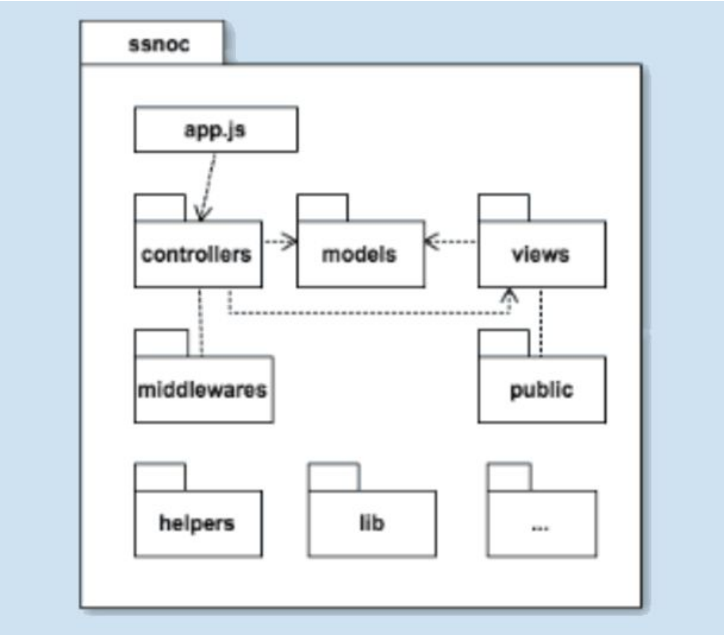
Survivable Social Network on a Chip — Team SV-4

SSNOC-V1.0.0: Social Network on Beaglebone board provides social network during emergencies

Technical Constraints

- **Hardware**: App server runs on a Beaglebone Black with wireless dongle and powered by a rechargeable battery. Clients connect to the app server via their mobile phone browsers. Memory and performance limited by hardware.
- **Client Side Software**: no native app, only web stack (HTML5, CSS, JS) on mobile browser (initially only Chrome will be supported)



High-Level Functional Requirements

- join community - access to the application
- chat publicly - chat messages are available to all
- chat privately - allows private chat
- share status - online / offline
- post announcement - System notification
- search information
- measure performance
- administer user profile

Top 3 Non-Functional Requirements

Works Out-of-the-Box > Usability > Your NFR3

- *Performance:* the application performs well on BBB
- *Maintainability:* the application is easy to maintain, since it is for emergency purpose
- *Testability:* the application is convenient to test, in order to detect bugs or faults

Architectural Decisions with Rationale

- Client-Server as main architectural style
- Server-side JS (node.js) for low footprint and reasonable performance (event-based, non-blocking asynchronous I/O, easily configurable pipe-and-filter for processing incoming requests via middleware)
- Lightweight MoVC on the server side via the **express** framework
- RESTful API for core functionality to reduce coupling between UI and back-end
- Event-based fast dynamic updates via web-sockets

Design Decisions with Rationale

- **Encapsulate** data and behavior in models for easy testing and better modularization
- Use **Adapter** design pattern is used to be able to substitute a test database for the production database during testing
- Use **Singleton** to the online listing of users; each time the Server needs only to use the same list
- **Open-closed principle:** the application open to extension, but closed to modification.

Responsibilities of Main Components

- **Socket.io:** dynamic updates from server to client, clients' views are automatically updated when new messages are post or when new new users login
- **Node.js**: back-end development environment
- **Bootstrap**: responsive design, clean, scalable UI layout
- **SQLite**: lightweight DB
- **express.js**: a popular framework for Node.js
- **jQuery**: a cross-platform JavaScript library designed to simplify the client-side scripting of HTML.