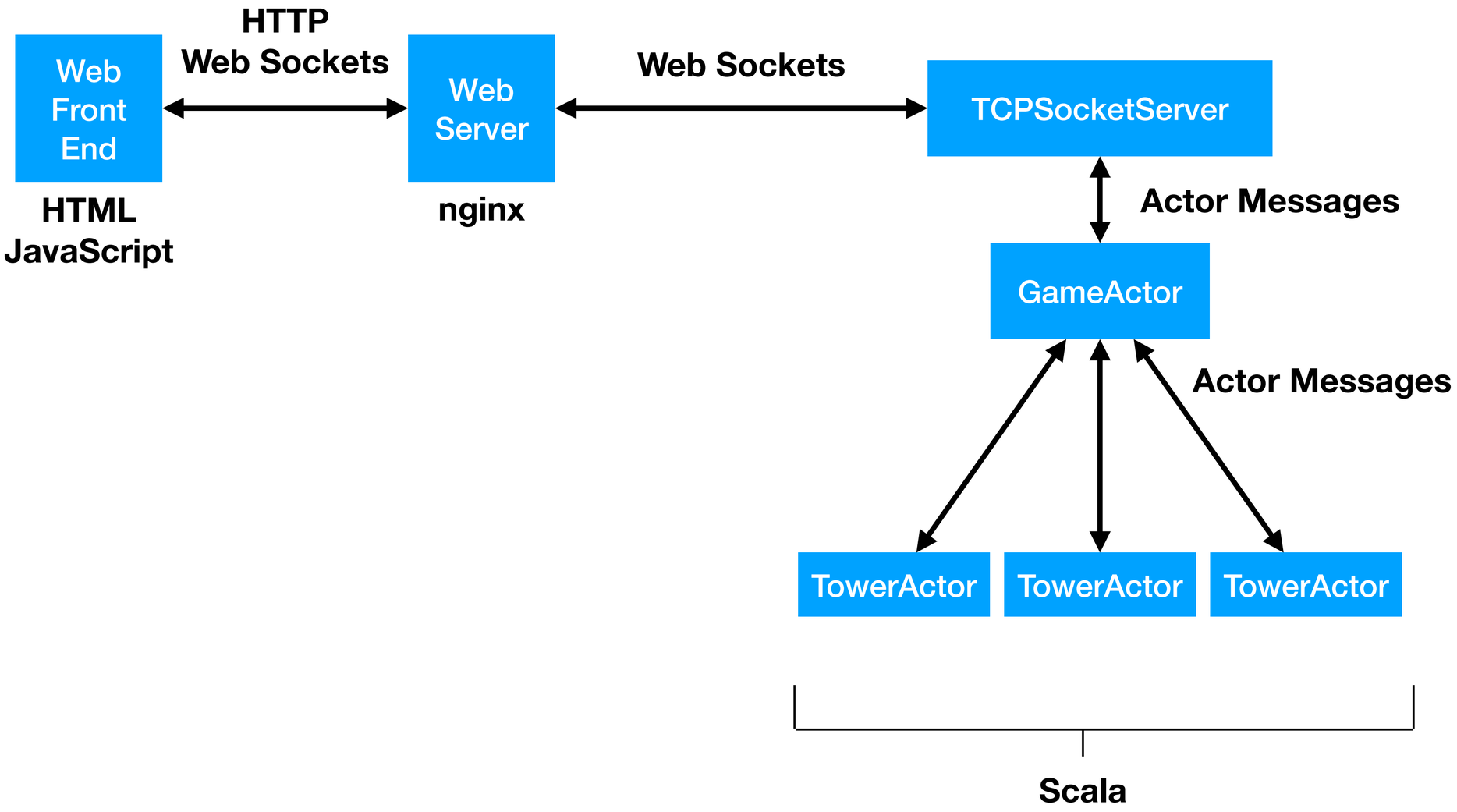
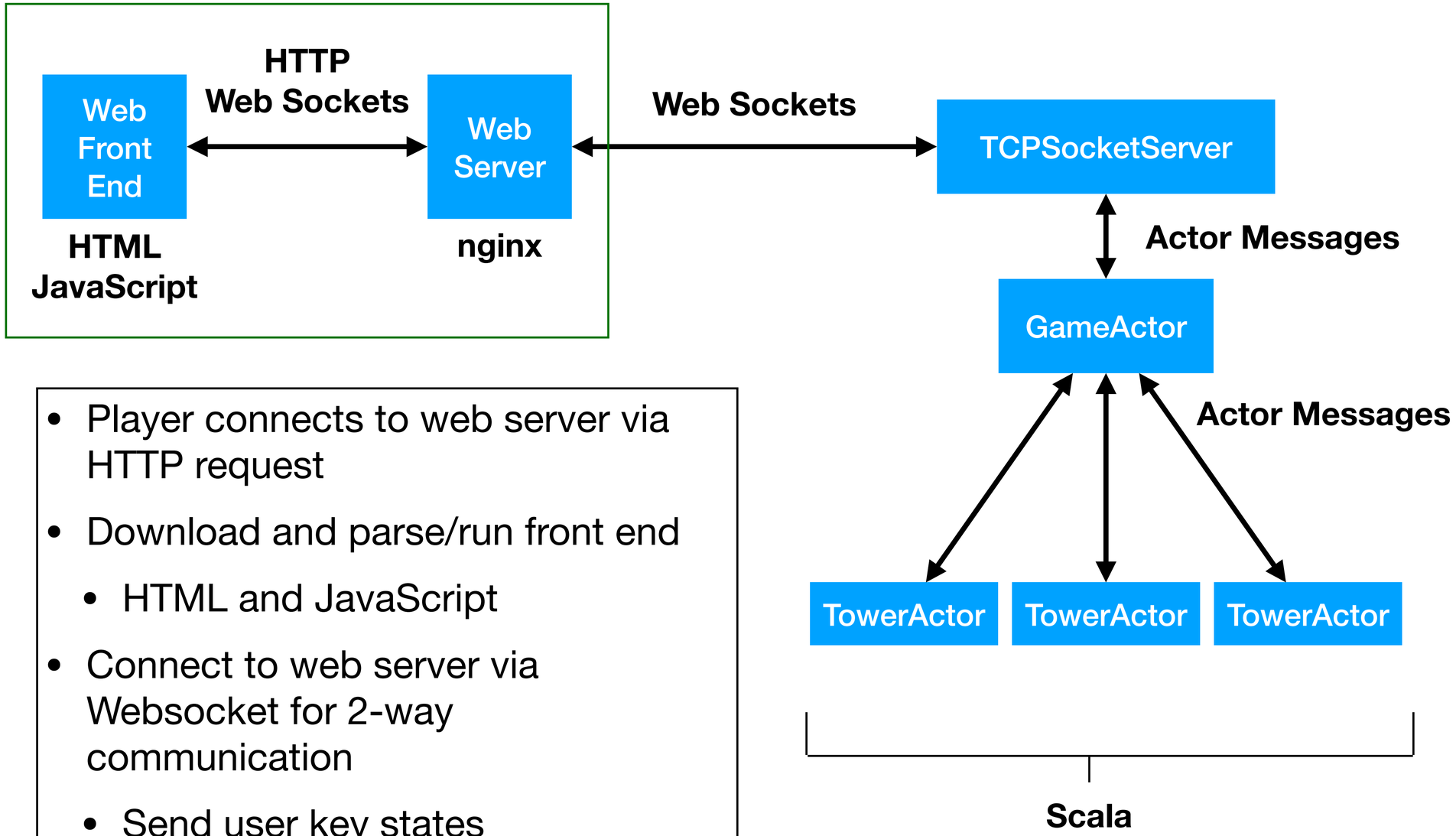
Towers Architecture

Towers Architecture

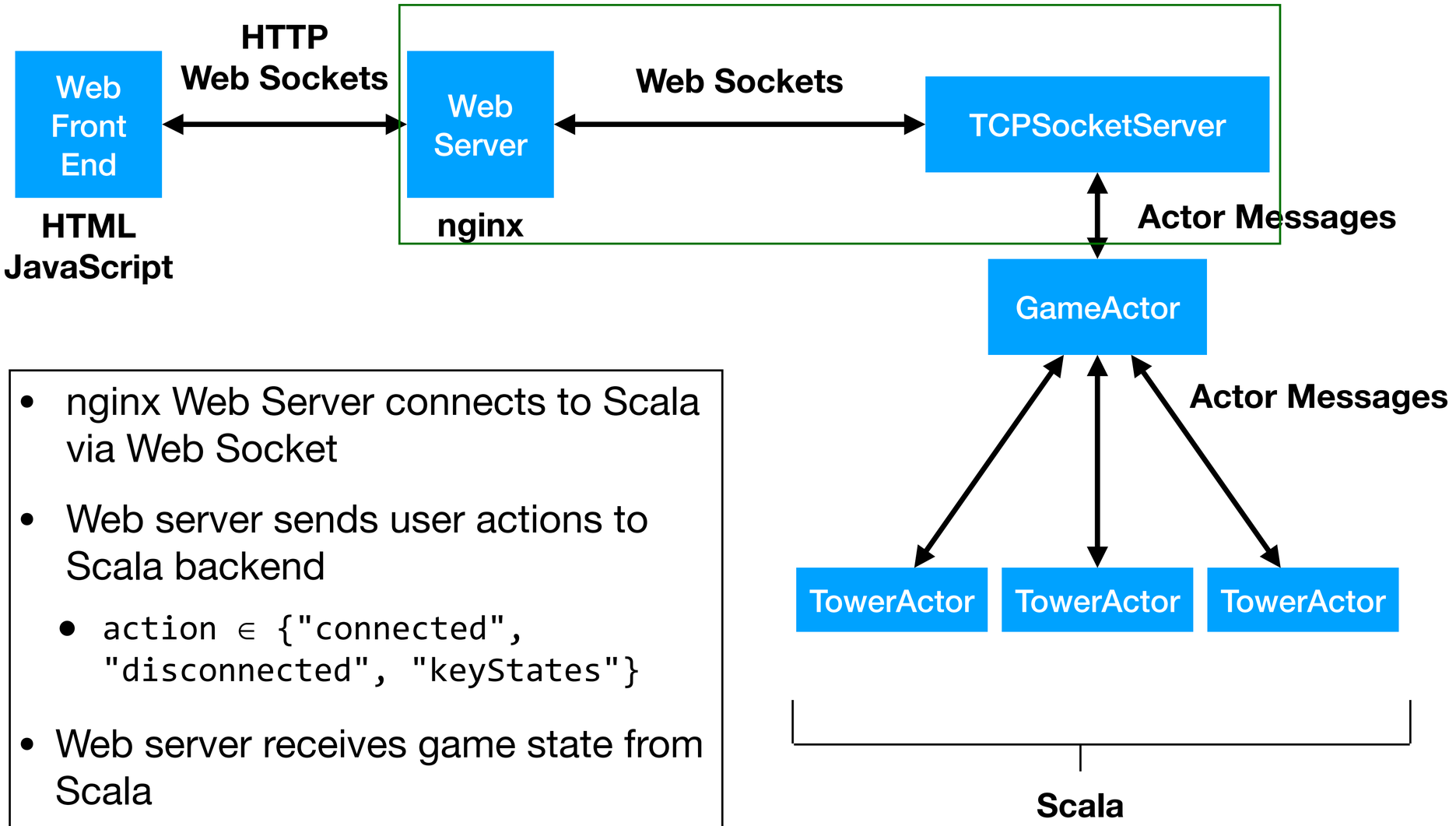


Towers Architecture



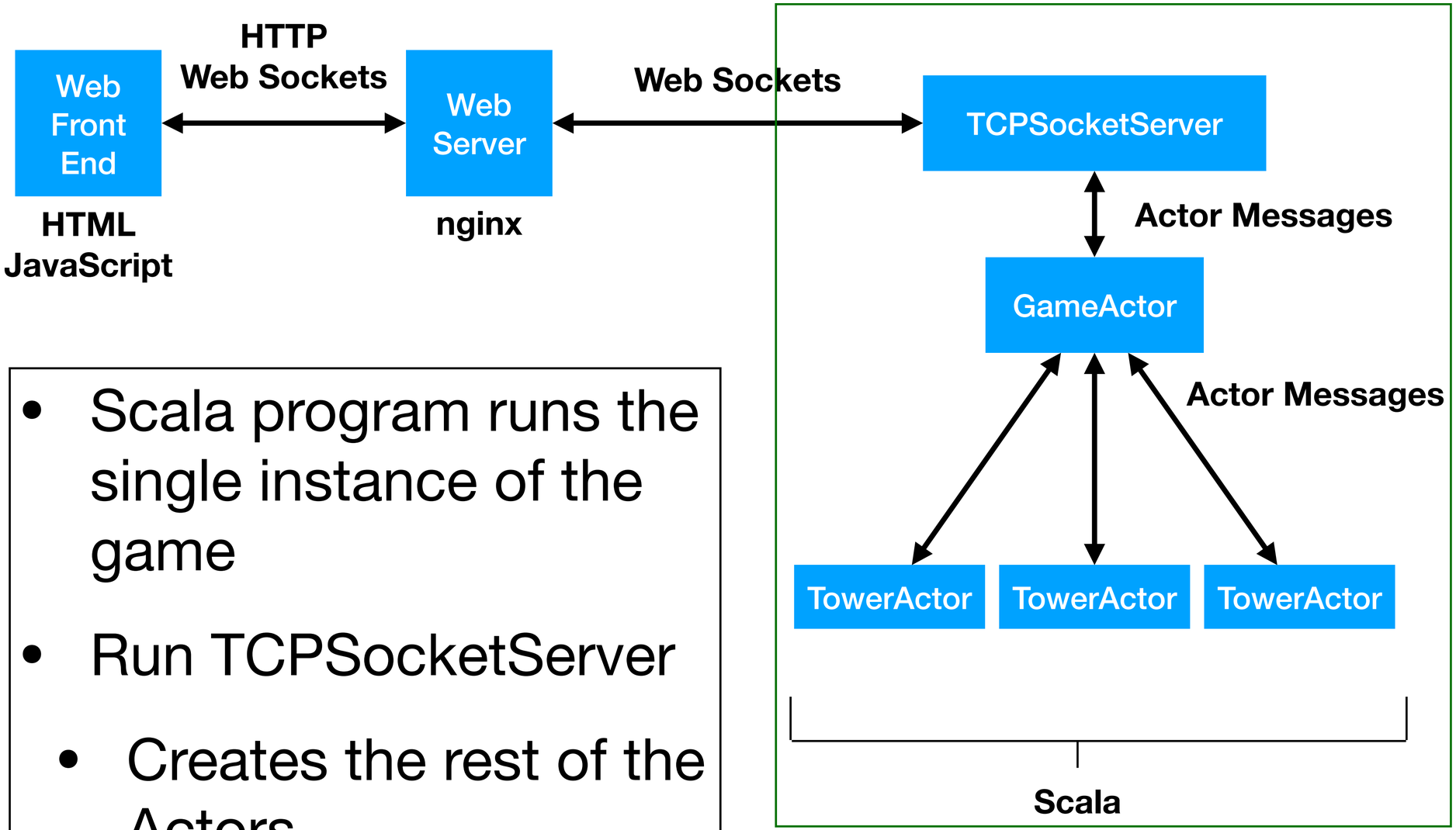
- Player connects to web server via HTTP request
- Download and parse/run front end
  - HTML and JavaScript
- Connect to web server via Websocket for 2-way communication
  - Send user key states
  - Receive and render game state

Towers Architecture



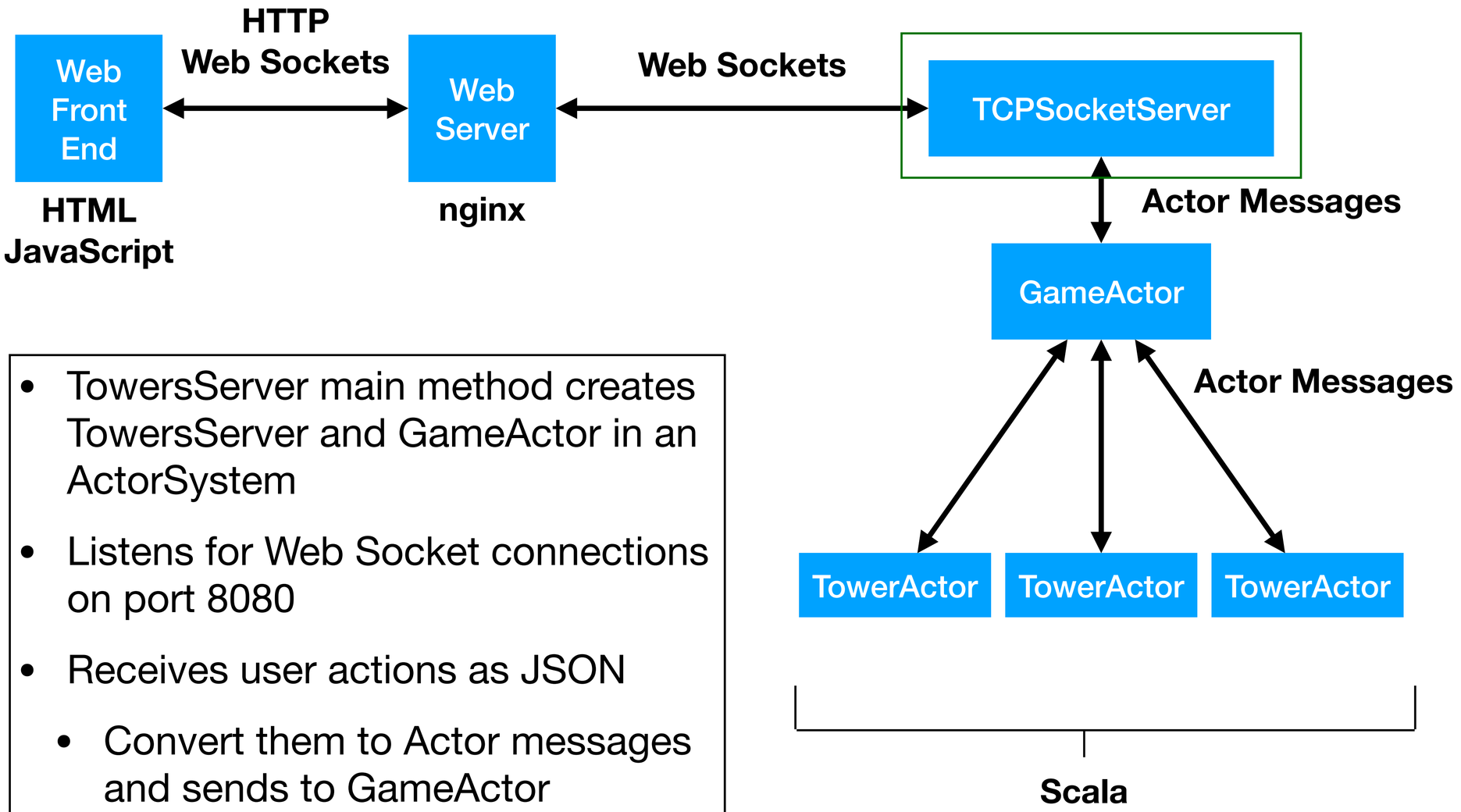
- nginx Web Server connects to Scala via Web Socket
- Web server sends user actions to Scala backend
  - `action ∈ {"connected", "disconnected", "keyStates"}`
- Web server receives game state from Scala
  - Broadcast game state to all players

Towers Architecture



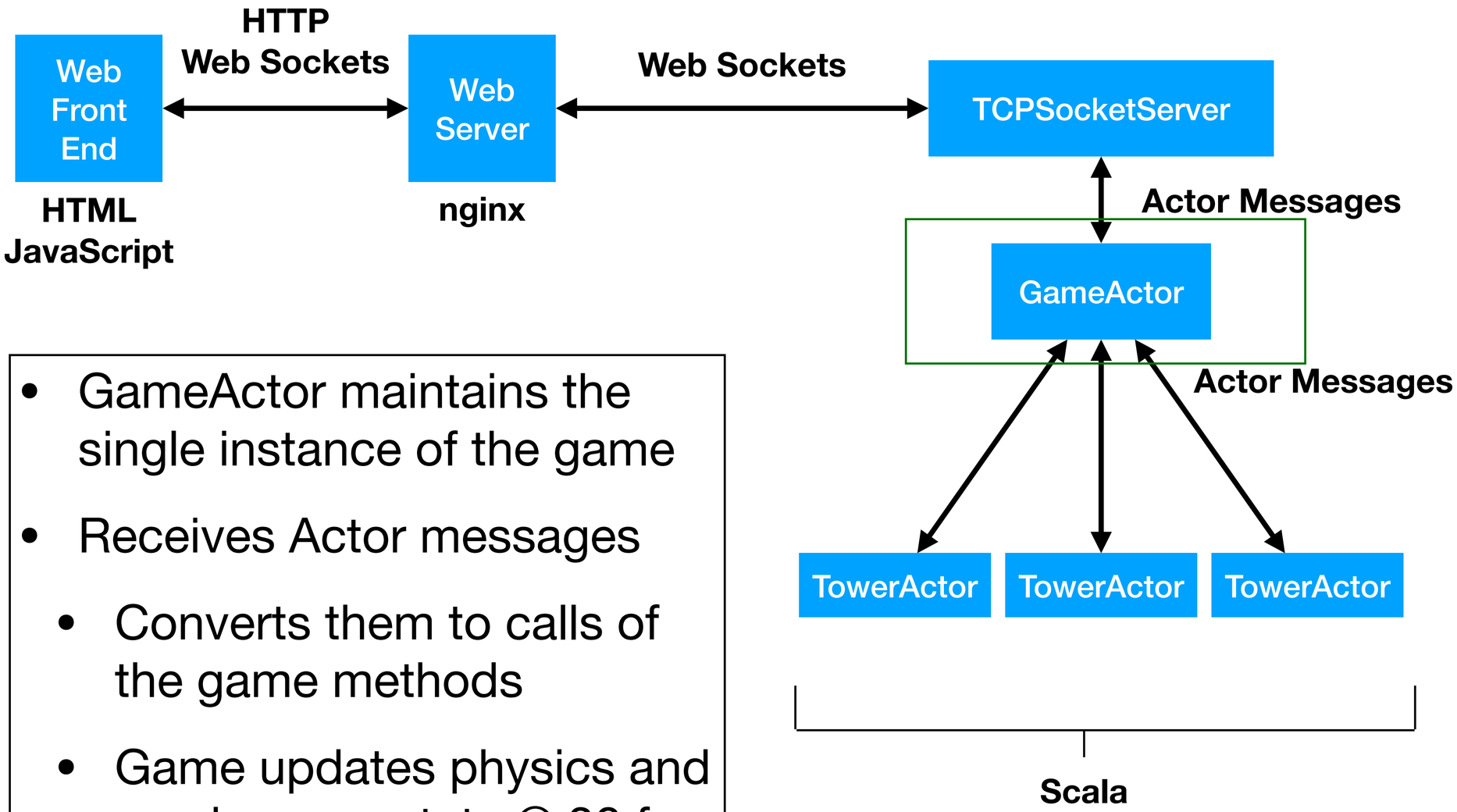
- Scala program runs the single instance of the game
- Run TCPSocketServer
  - Creates the rest of the Actors

Towers Architecture



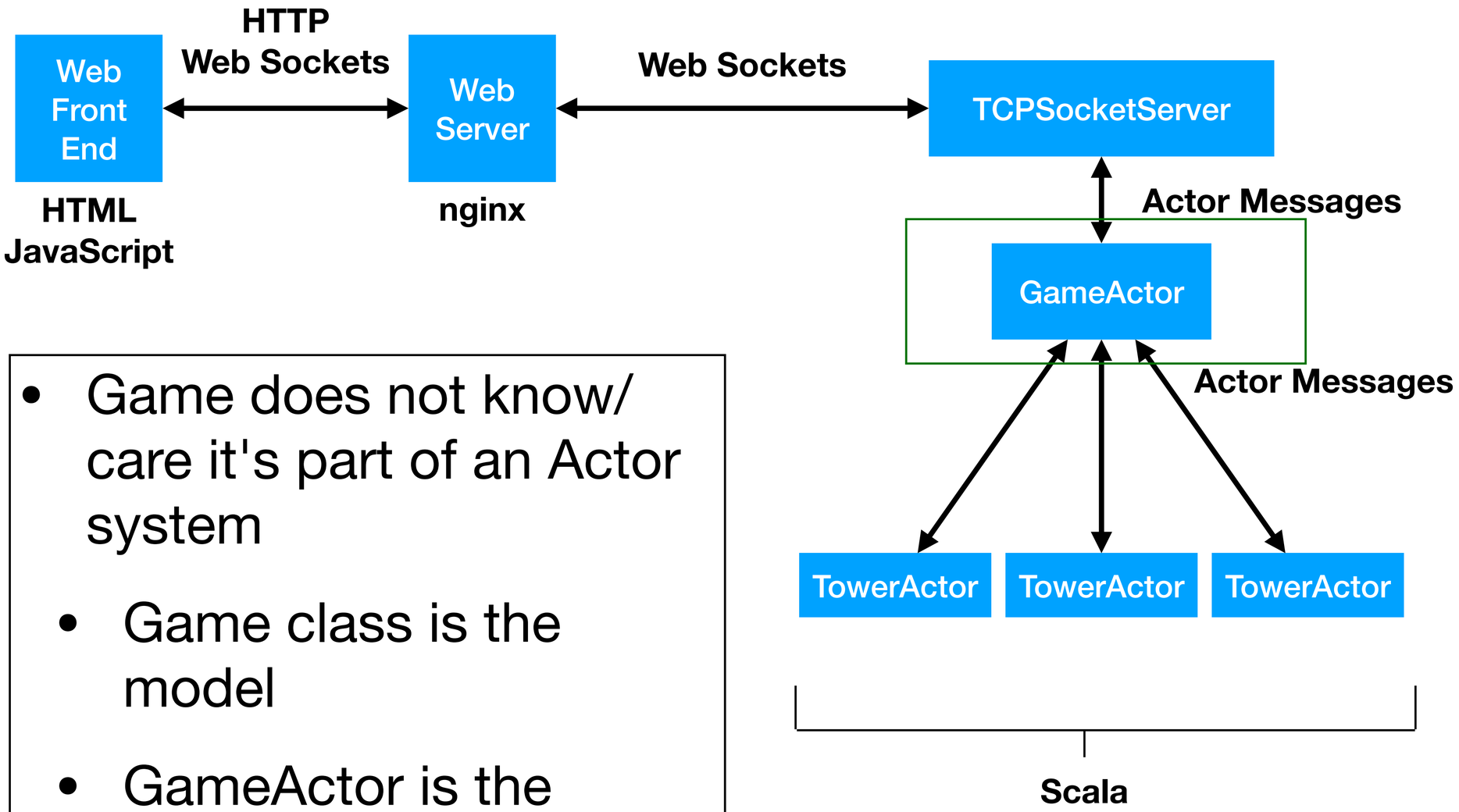
- TowersServer main method creates TowersServer and GameActor in an ActorSystem
- Listens for Web Socket connections on port 8080
- Receives user actions as JSON
  - Convert them to Actor messages and sends to GameActor
- Sends game state over Web Socket

Towers Architecture



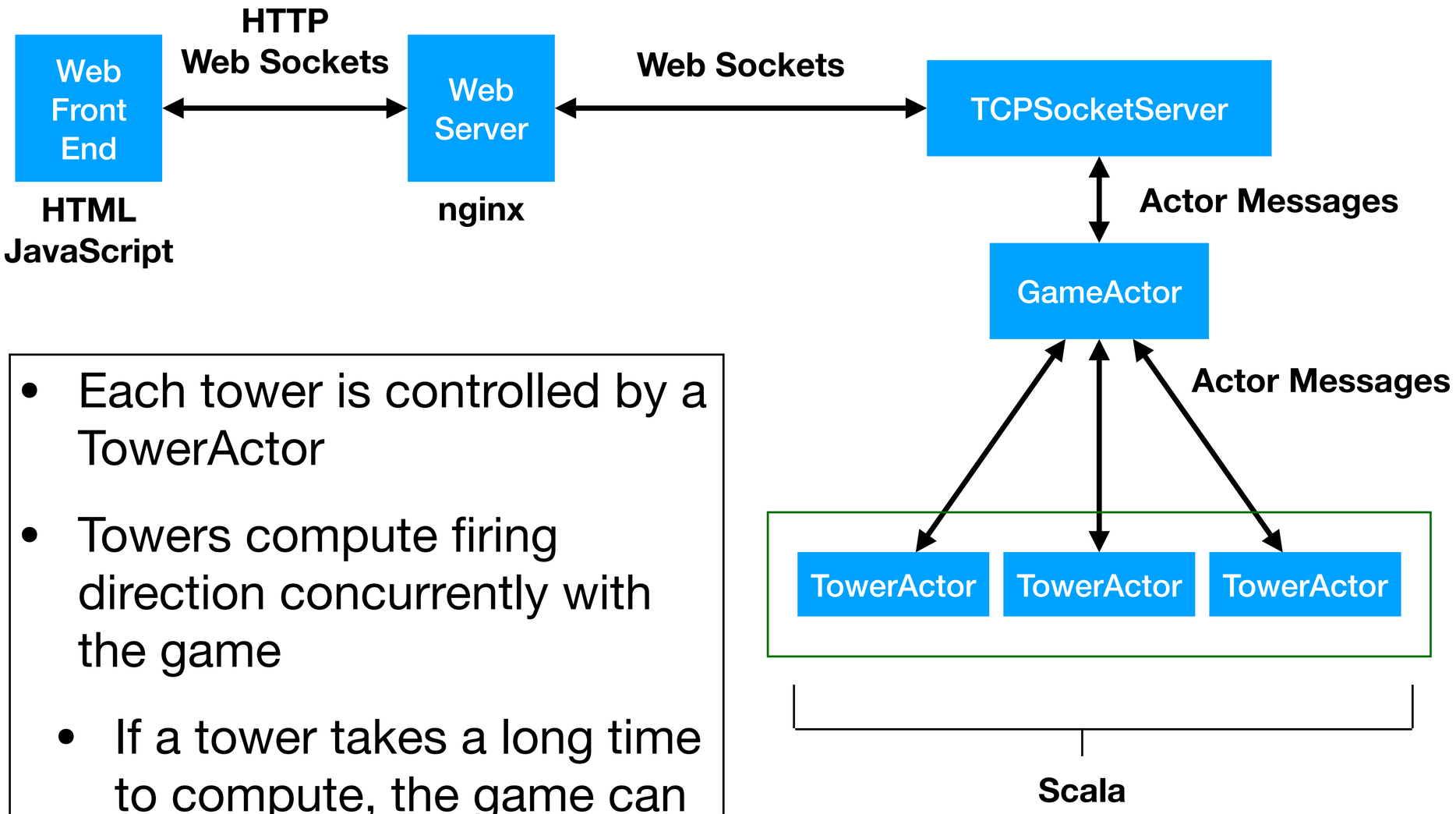
- GameActor maintains the single instance of the game
- Receives Actor messages
  - Converts them to calls of the game methods
  - Game updates physics and sends game state @ 30 fps

Towers Architecture



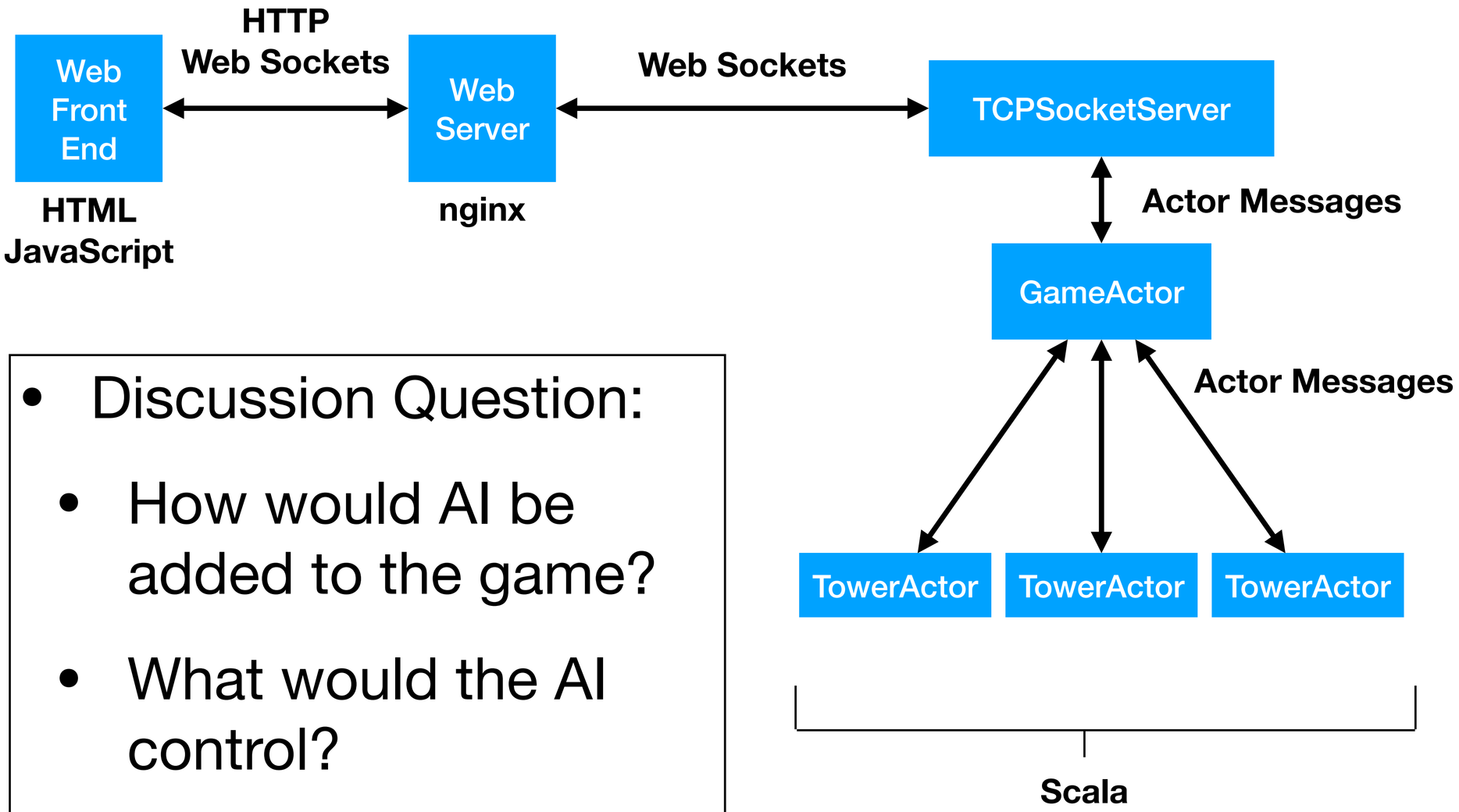
- Game does not know/ care it's part of an Actor system
  - Game class is the model
  - GameActor is the controller

Towers Architecture



- Each tower is controlled by a TowerActor
- Towers compute firing direction concurrently with the game
  - If a tower takes a long time to compute, the game can still run without lag

Towers Architecture



- Discussion Question:
  - How would AI be added to the game?
  - What would the AI control?
  - How would it behave?

Slow Towers Demo

- Let's run the game with slow towers and see if the actors behave as expected