**Общие требования:**

1. Приложение должно быть кроссплатформенным (работать под Android и Windows).
2. Обеспечить отказоустойчивость: приложение не должно аварийно завершаться при потере соединения или других сбоях.
3. Все модули должны обеспечивать корректное восстановление работы в случае сбоев.
4. Основные задачи:
   * Публикация собственного сервиса.
   * Обнаружение устройств с запущенной программой.
   * Реализация сервера и клиента для взаимодействия между устройствами.
   * Сервис проверки доступности связи с устройством (ping).
   * Координация всех компонентов.

**1. Публикация собственного сервиса**

**Задачи:**

* **Цель:** Оповещать другие устройства в сети о доступности сервиса.
* **Функциональность:**
  + Устройство публикует уникальное имя и параметры своего сервиса (например, имя устройства, IP-адрес, порт WebSocket сервера).
  + Использовать протокол **mDNS (Zeroconf)** для автоматического обнаружения.
  + В случае недоступности сети и прочих сбоях необходимо оповещать (например, паттерн Observer или Event Emitter).

Для реализации можно использовать существующие библиотеки например, Qt Service Discovery (Qt-обертка для mDNS/Zeroconf).

**2. Обнаружение своих устройств (поиск сервисов)**

**Задачи:**

* **Цель:** Находить другие устройства в сети на которых запущенна наша программа.
* **Функциональность:**
  + Способ обнаружения связан с выбранным способом публикации. Например можно также использовать mDNS для поиска устройств, публикующих сервис (а можно использовать BROADCAST - пытаться установить соединение со всеми устройствами в сети).
  + Реагировать на добавление новых устройств и потерю существующих, и оповещать (например, паттерн Observer или Event Emitter).
  + Должна быть возможность конфигурирования набора параметров для отбора устройств.
  + Хранить список обнаруженных устройств и их параметров (IP, порт).
  + Обновлять список в реальном времени, удаляя устройства, которые стали недоступны.
  + Предоставлять API для получения текущего списка устройств.

Под вопросом т.к. нарушает принцип SRP, и может быть реализовано на следующем уровне.

**3. WebSocket сервер для обработки множественных подключений**

**Задачи:**

* **Цель:** Обрабатывать входящие соединения от других устройств.
* **Функциональность:**
  + Запуск WebSocket сервера на порту, указанном в параметрах.
  + Обеспечивать обработку нескольких подключений одновременно.
  + Поддерживать обмен сообщениями (команды, файлы).
  + Поддержка таймаутов и проверка состояния соединений.
  + Оповещать о различных событиях и реагировать на разрыв соединений (например, паттерн Observer или EventEmitter).

Можноиспользовать Qt WebSocket для реализации сервера.

**4. WebSocket клиент**

**Задачи:**

* **Цель:** Устанавливать соединения с WebSocket серверами других устройств.
* **Функциональность:**
  + Подключаться ко всем обнаруженным устройствам.
  + Поддерживать двусторонний обмен данными.
  + Реагировать на потерю соединения и оповещать об этом.
* **Требования:**
  + Поддержка автоматического переподключения.
  + Обеспечить очередь сообщений в случае временной недоступности сервера.

**5. Сервис проверки соединения**

**Задачи:**

* **Цель:** Обеспечивать контроль состояния соединений между устройствами.
* **Функциональность:**
  + Периодически проверять связь с устройствами (ping).
  + Фиксировать разрывы соединений и уведомлять об этом другие модули.
* **Требования:**
  + Оптимизация проверки для минимальной нагрузки на сеть.
  + Предоставление API для получения состояния соединений.

**6. Сервис координирующий всё вышеперечисленное**

**Задачи:**

* **Цель:** Управлять взаимодействием всех модулей и координировать их работу.
* **Функциональность:**
  + Определять параметры сети (IP-адрес, доступные интерфейсы).
  + Запускать и контролировать все модули (сервер, клиент, публикацию, обнаружение).
  + Обеспечивать синхронизацию между модулями (например, при изменении списка устройств).
  + Реагировать на сбои (перезапускать модули при необходимости).
* **Требования:**
  + Централизованное управление с логгированием всех действий.
  + Реализация API для взаимодействия с приложением.

**Логика работы на каждом устройстве**

1. **Определение интерфейса связи и параметров устройства:**
   * Выбор активного сетевого интерфейса (Wi-Fi, Ethernet и т.д.).
   * Определение IP-адреса и порта для публикации.
2. **Запуск WebSocket сервера.**
   * Ожидание входящих подключений.
3. **Запуск WebSocket клиента.**
   * Подключение к другим устройствам.
4. **Запуск публикации собственного сервиса.**
   * Регистрация сервиса через mDNS.
5. **Запуск обнаружения своих устройств.**
   * Сканирование сети для поиска других сервисов.
6. **Запуск проверки доступности соединения с устройствами.**
   * Периодическая проверка связи.

**Отказоустойчивость**

1. **Публикация сервиса и обнаружение:**
   * Повторные попытки при сбоях сети.
   * Обновление данных при изменении IP.
2. **WebSocket сервер и клиент:**
   * Переподключение при разрыве соединения.
   * Очередь сообщений для недоступных устройств.
3. **Сервис проверки соединения:**
   * Уведомление о проблемах в соединении.
   * Автоматическое восстановление.
4. **Координирующий сервис:**
   * Перезапуск модулей при их сбоях.
   * Логирование и уведомления о критических ошибках.