Вопрос 4
Сравнительные характеристики синхронных и асинхронных протоколов

1. Основные принципы работы
Синхронные протоколы (HTTP, FTP):
- Работают по схеме "запрос-ответ"
- Клиент ожидает ответа сервера для продолжения работы
- Соединение закрывается после каждого запроса (если не используется keep-alive)
- Пример: HTTP/1.1, где каждый запрос требует нового соединения (без pipelining)

Асинхронные протоколы (WebSocket, MQTT):
- Поддерживают постоянное двустороннее соединение
- Данные могут передаваться в любое время без явного запроса
- Сервер может инициировать передачу данных клиенту
- Пример: WebSocket, где после установки соединения данные передаются в реальном времени

2. Установка и поддержание соединения
Характеристика	Синхронные протоколы	Асинхронные протоколы
Установка соединения	TCP handshake + TLS (если HTTPS)	TCP + WebSocket handshake
Таймауты	Обычно 30-60 сек	Могут быть часами
Keep-alive	HTTP Keep-Alive заголовок	Встроенный ping/pong механизм
Проверка связи	Нет стандартного механизма	Ping/pong фреймы

3. Передача данных
Синхронные:
- Данные передаются либо в заголовках (метаданные), либо в теле запроса
- Каждый запрос содержит полный набор заголовков
- Пример: HTTP, где заголовки передают метаданные, а тело - содержимое

Асинхронные:
- Данные передаются в виде отдельных фреймов
- Заголовки устанавливаются при подключении, далее передается только полезная нагрузка
- Пример: WebSocket, где после handshake передаются только данные

4. Восстановление соединения
Синхронные:
- При разрыве требуется новый запрос
- Клиент должен обрабатывать ошибки и повторять запросы
- Нет стандартного механизма восстановления

Асинхронные:
- Автоматические попытки переподключения (reconnect)
- Сохранение состояния сессии (если поддерживается протоколом)
- Пример: WebSocket библиотеки обычно имеют встроенный reconnect

5. Пример работы WebSocket
1. Установка соединения:
   - Клиент отправляет HTTP-запрос с заголовком `Upgrade: websocket`
   - Сервер отвечает `101 Switching Protocols`
   - Происходит TLS handshake (для wss://)

2. Поддержание соединения:
   - Регулярные ping/pong фреймы для проверки связи
   - Таймауты определяются реализацией (обычно 30-60 сек)

3. Передача данных:
   - Данные разбиваются на фреймы
   - Поддержка текстовых и бинарных данных
   - Нумерация фреймов для контроля целостности

4. Восстановление:
   - При разрыве клиент может автоматически переподключаться
   - Возможность восстановления сессии (если сервер поддерживает)

Вывод
Синхронные протоколы лучше подходят для простых запросов, где не требуется постоянное соединение. Асинхронные протоколы оптимальны для приложений реального времени, где важна двусторонняя связь и минимальные задержки. WebSocket сочетает преимущества HTTP (совместимость с веб-инфраструктурой) с возможностями полноценного двустороннего соединения.