Разработка протокола прикладного уровня

Задание на этап:
1 Составить диаграмму состояний программы-клиента(ов) и программы-сервера.
2 На основе диаграммы состояний определить необходимые типы запросов и ответов протокола.
3 Составить диаграмму последовательности действий (взаимодействия) протокола.
4 Разработать форматы сообщений протокола прикладного уровня.

Результатом выполнения этапа являются:
- перечисление функций программ (из задания на курсовую работу);
- диаграмма состояний программы-клиента;
- диаграмма состояний программы-сервера;
- таблица с типами запросов и ответов на них;
- диаграммы последовательности;
- форматы сообщений протокола прикладного уровня с описанием значений полей.

Протокол прикладного уровня должен описывать способ сетевого взаимодействия между программой-клиентом (далее - клиент) и программой-сервером (далее - сервер).

Протокол прикладного уровня должен содержать сообщения, интерпретируемые клиентом и сервером для смены их состояний. Единицей информационного обмена служит сообщение. Изначально клиент и сервер находятся в некоторых состояниях. Посредством обмена сообщениями происходит смена состояний и выполнение каких-либо действий.

Сначала необходимо определить возможные состояния, в которых могут находиться клиент и сервер. Для их описания используется **Диаграмма состояний** языка UML.

Диаграммы состояний представляют собой граф состояний и переходов конечного автомата, содержащий дополнительные детали и подробности. На диаграмме состояний применяют один основной тип сущностей — состояния, и один тип отношений — переходы. При этом состояния конечного автомата соответствуют состояниям объекта, т. е. различным наборам значений атрибутов, а переходы соответствуют выполнению операций.

На диаграмме используются следующие обозначения:
- начальное состояние (круг)
- конечное состояние (круг в окружности)
- состояние (скругленный прямоугольник), которое имеет следующую структуру:
  - имя;
  - действие при входе (entry);
  - действие при выходе (exit);
  - множество внутренних переходов и внутренняя активность (do);
  - множество отложенных событий (defer);
- переход (стрелка), которая обозначается названием события (если есть), вызывающего переход. Может содержать событие перехода, сторожевое условие и действия на переходе – если наступает событие, то проверяется сторожевое условие. Если условие истинно – выполняется действие и происходит смена состояния.

Диаграмма состояний позволяет определить перечень событий, наступление (или завершение) которых определяется передачей сообщения протокола.

Сообщения протокола можно построить по схеме запрос-ответ. Протокол будет содержать сообщения-запросы, которые позволят одной стороне взаимодействия

изменить состояние другой стороны, и сообщения-ответы, позволяющие сообщить об успешности или неудаче.

Разработку диаграммы начнем с определения состояний. Состояние может отображать только название состояния, либо содержать дополнительный список внутренних действий (entry, exit, do, defer). В первом случае состояние изображается простым блоком, во втором используется расширенное описание, представленные на рисунке 1,а) и 1,б) соответственно.

а) б)

Рисунок 1 – Условные обозначения состояний

Первоначально на диаграмму добавляются псевдосостояния – начальное и конечное, обозначаемые кругом и кругом в окружности соответственно, как показано на рисунке 2,а) и 2,б) соответственно.

а) б)

Рисунок 2 – Начальное и конечное состояния

Выполнение этапа начинается с разработки диаграммы состояний программы-клиента. При сетевом взаимодействии программа-клиент отправляет запросы серверу, получает ответ и, в зависимости от полученного ответа, переходит в другое состояние. После запуска программы-клиента необходимо выполнить подключение к программе-серверу, поэтому первое состояние программы-клиента можно обозначить как «Подключение к серверу». Подключение может быть как успешным, так и неуспешным. Если подключение произошло успешно, то программа-клиент переходит в следующее состояние, иначе – либо возвращается в предыдущее, либо переходит в состояние, в котором выполняется обработка ошибок подключения.

Переход в следующее состояние изображается стрелкой. Переход может быть обозначен именем события, которое вызвало смену состояния (событие-триггер), и сторожевым условием. Рассмотрим смену состояний программы-клиента из состояния «Подключение к серверу» в следующее. При успешном подключении к серверу программа-клиент должна формировать и отправлять запросы на получение от сервера информации, необходимой для выполнения основной функции. Например, такой функцией является получение с сервера файла. Это состояние можно назвать «Запрос файла». При неуспешном подключении программа-клиент вернется в состояние «Подключение к серверу». Теперь необходимо определиться с событиями, которые приводят к смене состояний. Таким событием-триггером является отправка запроса на подключение к программе серверу. Сторожевым условием перехода в успешное состояние является ответ программы-сервера о подключении, сторожевым условием перехода в неуспешное состояние является либо сообщение об ошибке, либо отсутствие ответа в течение определенного момента времени. Сторожевое условие является булевым выражением и принимает значение «истина» или «ложь». Сторожевые условия переходов, выходящих из одного состояния, не должны одновременно быть истинными, т.к.

одновременный переход сразу в два и более состояния невозможен. Сторожевые условия на диаграмме обозначаются в квадратных скобках.

Получается два перехода:

```
запрос на подключение [успешное подключение]
запрос на подключение [ошибка подключения | истекло время ожидания]
```

Формально в диаграмме переход для возврата в предыдущее состояние можно не указывать, т.к. переход в состояние «Запрос файла» возможен только при срабатывании сторожевого условия, однако в этом случае на диаграмме не будет отображено сообщение от сервера, соответствующее ошибке подключения.

Изобразим состояния и переходы на диаграмме (рисунок 3).

Рисунок 3

В результате описания двух состояний определены первые сообщения, которыми обмениваются программа-клиент и программа-сервер:

запрос клиент – сервер	ответ сервер – клиент
запрос на подключение	успешное подключение
	ошибка подключения

Из состояния «Запрос файла» программа-клиент переходит в состояние «Загрузка файла». Для того чтобы получить файл, программа-клиент должна отправить запрос с указанием файла и либо получить файл в ответном сообщении, либо сообщение об ошибке (например, если запрашиваемый файл не существует или у программы-клиента нет прав на его получение). Переход в состояние «Запрос файла» аналогичен по структуре предыдущим:

```
запрос файла[успешный запрос]
```

Событиям и сторожевым условия рекомендуется давать уникальные названия. Сообщения об ошибках могут быть универсальными, т.е. иметь одинаковый формат независимо от вида ошибки, либо быть различными для определения последующего состояния, отображения сведений об ошибках пользователю и т.д.

Вернемся к переходу, который описывает ошибку при выполнении запроса. Например, требуется отобразить пользователю программы-клиента причину ошибки. Возможно два вида ошибок – неверно указано имя файла или версия программы не соответствует типу файла. В этом случае особенность функционирования программы-клиента можно отобразить на диаграмме через выражение действия (action expression) на переходе. Выражение выполняется, когда срабатывает переход и представляет собой операцию, выполняемую до перехода в следующее состояние. Таким выражением может быть «вывод файл не найден» и «вывод неправильная версия». Получается еще два перехода:

```
запрос файла[файл не найден]/вывод файл не найден
```

```
запрос файла[неправильная версия]/вывод неправильная версия
```

Подробнее рассмотрим состояние «Загрузка файла». Переход из этого состояния в следующее произойдет, когда завершится передача файла, при этом запросы и ответы отправляться не будут, т.е. смена состояния произойдет без участия внешних воздействий, а по завершении действий, выполняемых в этом состоянии. В этом случае переход в следующее состояние не будет содержать событий и условий, а сработает по завершении действий. Такой переход иногда называется «нетриггерный», при этом в блоке, описывающем состояние должно содержать метку внутренней активности (do). Переход сработает, когда завершится внутренняя активность. После загрузки файла клиент опять перейдет в состояние «Запрос файла».

Из состояния «Запрос файла» добавим переход в конечное состояние, при этом программе-серверу будет отправляться сообщение о закрытии соединения и завершение работы программы, ответа от сервера при этом не предусмотрено.

Диаграмма представлена на рисунке 4.

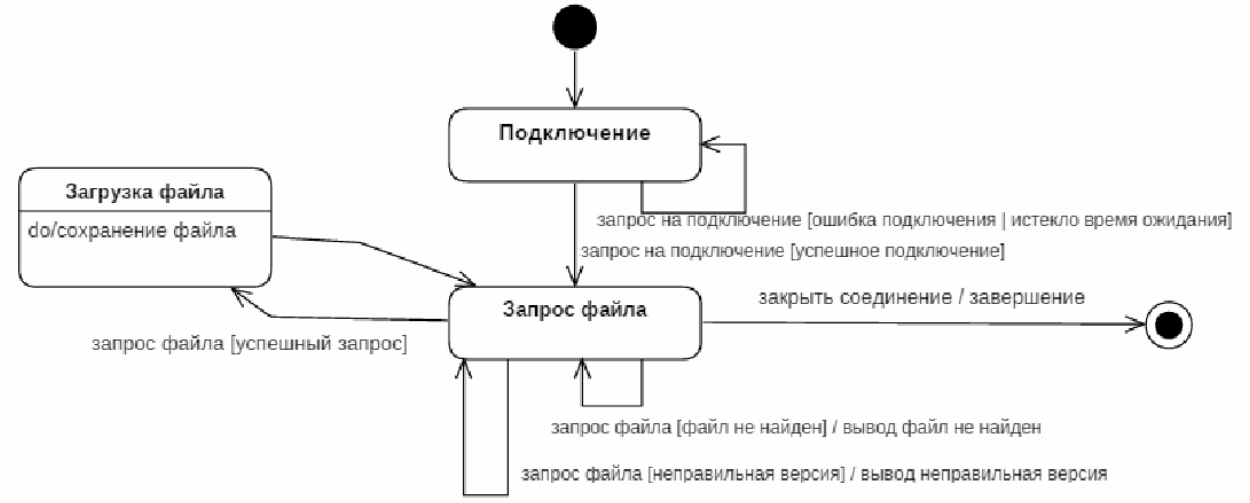

Рисунок 4 – Диаграмма состояний программы-клиента

По диаграмме можно определить следующие сообщения протокола, приведенные в таблице 1.

Таблица 1 – Запросы и ответы

запрос клиент – сервер	ответ сервер – клиент
запрос на подключение	успешное подключение
	ошибка подключения
запрос файла	успешный запрос
	файл не найден
	неправильная версия
закрыть соединение	

Диаграмма состояний программы-сервера отличается состояниями. Первоначально программа-сервер находится в состоянии ожидания подключений. Смена состояния происходит, когда поступает запрос на подключение от программы-клиента. Если подключение успешно, программа-сервер переходит в состояние ожидания запроса на передачу файла, иначе – возвращает сообщение об ошибке и возвращается в состояние ожидания. Переходы программы-сервера в целом аналогичны переходам программы-клиента. Отличие заключается в том, что сторожевое условие будет означать результат обработки запроса программой-сервером, поэтому для детализации логики работы программы-сервера можно добавить дополнительные состояния обработки запроса. Для детализации действий переходов можно добавить выражение действия, соответствующее отправке соответствующих ответов на запросы. Пример обработки запроса на подключение от программы-клиента приведен на рисунке 5.

Таким образом, описание перехода на диаграмме состояний программы-клиента имеет формат: `запрос клиента [ответ сервера]`, а на диаграмме состояний программы-сервера описание разбивается на два перехода. Первый: `запрос клиента`, второй `[результат обработки]/ответ сервера`.

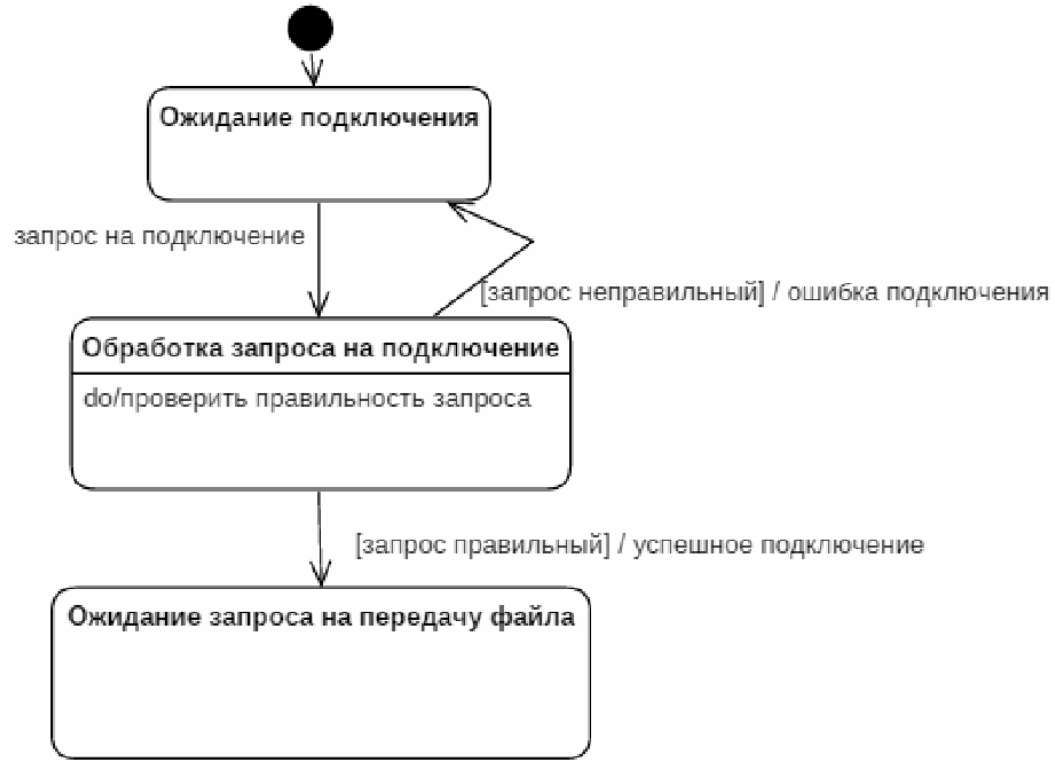

Рисунок 5 – Фрагмент диаграммы состояний программы-сервера

Выражений действий у переходов может быть несколько, в этом случае они разделяются символом «;» (точка с запятой). Например, если в программе-сервере должно выполняться протоколирование длительности сессии программы-клиента, то помимо отправки сообщения об успешном подключении, в переход может быть добавлено выражение действия «занести время подключения в журнал»:

```
[запрос правильный]/успешное подключение; занести время подключения в журнал
```

В процессе составления диаграммы состояний программы-сервера может появиться необходимость добавления дополнительных запросов и ответов. В этом случае также следует скорректировать диаграмму состояний программы-клиента и дополнить таблицу запросов и ответов на них.

Результатом разработки диаграмм состояний программы-клиента и программы сервера является описание логики взаимодействия, но основной задачей является **определение необходимых типов и количество возможных запросов и ответов протокола прикладного уровня**. При составлении диаграмм необходимо рассмотреть возможные ошибки, повторные запросы на выполнение функций, запросы на аутентификацию, закрытие установленного соединения и т.д.

После определения типов и количества запросов и ответов необходимо разработать описание последовательности обмена сообщениями. Последовательность обмена сообщениями можно описать с использованием **Диаграммы последовательности**.

Диаграмма последовательности предназначена для моделирования поведения в форме описания протокола сеанса обмена сообщениями между взаимодействующими объектами во время выполнения одного из возможных сценариев.

На диаграмме последовательности считается выделенным одно направление, соответствующее течению времени. По умолчанию считается, что время течет сверху вниз. Сообщения изображаются прямыми стрелками разного вида. Если передача

сообщения считается мгновенной, то стрелка горизонтальна. Если же нужно отобразить задержанную доставку сообщения, то стрелку немного наклоняют, так чтобы конец стрелки был ниже начала. Среди сообщений есть первое, которое кладет начало данному взаимодействию. Стрелка этого сообщения расположена выше всех других стрелок сообщений. Все объекты, которые находятся выше первого сообщения, существуют до начала данного взаимодействия; все объекты, которые расположены ниже, возникают в процессе данного взаимодействия. В направлении оси времени от всех участвующих во взаимодействии объектов отходит прямая пунктирная линия, которая называется линией жизни. Линия жизни представляет объект во взаимодействии: если стрелка отходит от линии жизни объекта, то это означает, что данный объект отправляет сообщение, а если стрелка сообщения входит в линию жизни, то это означает, что данный объект получает сообщение. Если в процессе взаимодействия объект заканчивает свое существование, то линия жизни обрывается и в этом месте ставится жирный косой крест.

Сообщение может передавать управление другому объекту. Чтобы показать, что некоторый объект в определенный период взаимодействия имеет фокус управления на диаграмме последовательности соответствующую часть линии жизни объекта изображают в виде узкой полоски. Фактически такая полоска означает выполнение операции объекта и называется активацией объекта. Начало активации соответствует приему сообщения вызова операции, а конец активации — завершению выполнения операции и возврату управления. Для наглядности на диаграмме последовательности можно показать в явном виде возврат управления (и, может быть, возвращаемое значение).

Диаграмма последовательности действий позволяет описать взаимодействие клиента и сервера и определить достаточность типов запросов-ответов протокола.

Следует описать все возможные сценарии взаимодействия программы-клиента и программы-сервера. Первый сценарий – правильное взаимодействие, затем рассмотреть ошибки подключения, аутентификации, неправильные запросы и т.д. Каждая стрелка диаграммы должна соответствовать запросу или ответу, определенному при разработке диаграммы состояний. В случае выявления необходимости добавления запроса или ответа, следует скорректировать диаграммы состояний программы-клиента и программы-сервера.

Пример диаграммы последовательности для правильного взаимодействия программы-клиента и программы-сервера приведен на рисунке 6.

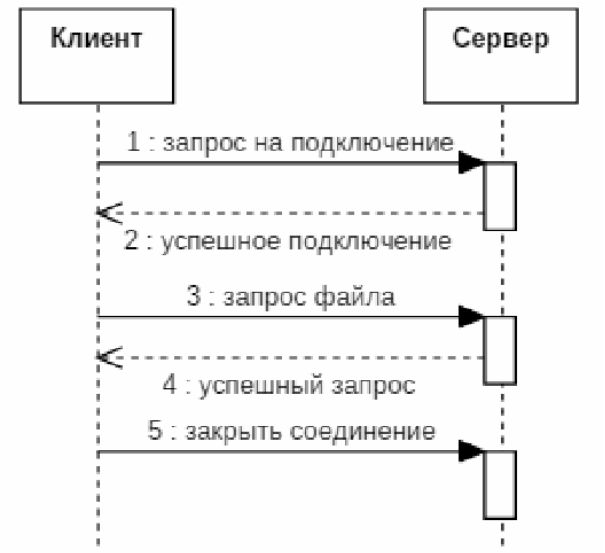

Рисунок 6 – Диаграмма последовательности правильного взаимодействия

Пример диаграммы последовательности при запросе несуществующего файла приведен на рисунке 7.

Обратите внимание, что на диаграммах все запросы и ответы соответствуют сообщениям, приведенным в таблице 1. Кроме того, при обработке ошибок отправляются только те сообщения, которые возможны из того состояния, в котором находится программы согласно диаграмме состояний.

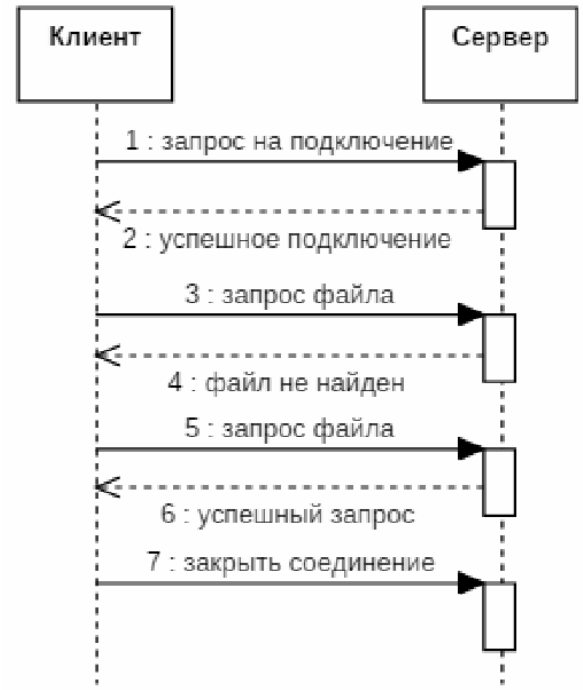

Рисунок 7 – Диаграмма последовательности с обработкой ошибки запроса несуществующего файла

После определения типов ответов следует разработать формат сообщений протокола. Сообщения-запросы, как правило, имеют в своем составе заголовок, описывающий тип запроса, и тело сообщения, описывающее подробности запроса.

Например, сообщение, предназначенное для запроса подключения клиента к серверу может иметь следующий формат:

Заголовок	connection
Тело	clientID: <значение> messageID: <значение> clientType: <значение> …

Поля сообщения могут быть различного формата (строка, число). Также поля сообщения могут быть выровнены по определенной границе (например, каждое поле 32 байта), могут быть произвольного размера. В первом случае упрощается интерпретация (разбор полей сообщения) приемной стороной, но уменьшается количество возможных значений поля. Во втором случае – усложняется интерпретация, но возможно большее количество значений полей.

Сообщение-ответ по формату может совпадать с запросами (заголовок-тело), может обозначаться кодами ответа (например, код 200 будет означать успешную обработку запроса, код 400 – ошибку подключения). Сообщения-ответы необходимо связывать с сообщениями-запросами. Для этого, например, в запрос можно включать идентификатор сообщения (`messageID` в примере).

Для разработанных форматов сообщений следует привести формат сообщения и описание полей сообщения. Например, формат запроса на подключение можно представить следующим образом:

Название: запрос клиента на подключение к серверу.

Формат:

connection
clientID: <значение> messageID: <значение> clientType: <значение>

Значения полей:

connection – заголовок запроса, строка символов в кодировке UTF-8;

clientID: <значение> - строка символов в кодировке UTF-8, поле «значение» содержит идентификатор программы-клиента (от 1 до 24 символов английского алфавита), используется для определения сессии каждой программы-клиента;

messageID: <значение> - строка символов в кодировке UTF-8, поле «значение» содержит целое число от 1 до 32, выбирается случайно, используется для сопоставления запросов и ответов каждой программы-клиента с одинаковым значением полей clientID;

clientType: <значение> - строка символов, поле «значение» содержит целое число 1 или 2, определяет номер версии подключенной программы-клиента.