ВВЕДЕНИЕ

Изучение проектирования и создания сетевых программных средств является обязательным для всех специалистов, связанных с компьютерными технологиями. В качестве разработки рекомендуется пройти все этапы создания низкоуровневых программных средств. С одной стороны, современные языки программирования с дополнительными библиотеками очень абстрактны и позволяют минимизировать решение проблем сетевого взаимодействия между программами. С другой стороны, в процессе передачи по сети сообщения проходят через все уровни стека сетевых протоколов, от прикладного до физического. Чтобы писать более интеллектуальные программы, вам нужно понимать, как генерируются сами данные, как программные системы устанавливают соединения и как они передают данные друг другу.

Характеристика клиент-сервер описывает взаимосвязь взаимодействующих программ в приложении. Серверный компонент предоставляет функцию или услугу одному или нескольким клиентам, которые инициируют запросы на такие услуги. Серверы классифицируются в соответствии с предоставляемыми ими услугами. Например, веб-сервер обслуживает веб-страницы, в то время как файловый сервер обслуживает компьютерные файлы. Общим ресурсом может быть любое программное обеспечение и электронные компоненты серверного компьютера, от программ и данных до процессоров и устройств хранения. Совместное использование ресурсов сервера - это услуга.

Является ли компьютер клиентом, сервером или и тем, и другим, определяется характером приложения, которому требуются службы. Например, веб-серверы и программное обеспечение файлового сервера могут работать одновременно на одном компьютере для обслуживания

разных данных для клиентов, отправляющих разные типы запросов. Клиентское программное обеспечение также может взаимодействовать с серверным программным обеспечением на том же компьютере. Связь между серверами, например, для синхронизации данных, иногда называют межсерверной.

Вообще говоря, сервис - это абстракция компьютерных ресурсов, и клиенту не нужно беспокоиться о том, как работает сервер при отправке запроса и доставке ответа. Клиенту нужно только понять ответ, основанный на известном прикладном протоколе, т.е. содержимое и форматирование данных для запрашиваемой услуги.

Клиенты и серверы обмениваются сообщениями по схеме запрос-ответ. Клиент отправляет запрос, а сервер возвращает ответ. Этот обмен сообщениями является примером межпроцессного взаимодействия. Для общения компьютеры должны иметь общий язык и следовать правилам, чтобы и клиент, и сервер знали, чего ожидать. Язык и правила общения определяются в протоколе связи. Все протоколы клиент-серверной модели работают на прикладном уровне. Протокол прикладного уровня определяет основные шаблоны общения. Для дальнейшей формализации обмена данными сервер может реализовать интерфейс прикладного программирования (API). API - это уровень абстракции для доступа к сервису. Ограничивая ссылку определенным форматом содержимого, это упрощает синтаксический анализ. Абстрагируя доступ, он облегчает кроссплатформенный обмен данными.

Сервер может получать запросы от множества разных клиентов за короткий промежуток времени. Компьютер может выполнять только ограниченное количество задач одновременно и полагается на систему планирования для определения приоритетов входящих запросов от клиентов для их удовлетворения. Для предотвращения злоупотреблений и обеспечения максимальной доступности серверное программное обеспечение может ограничить доступность для клиентов.

Атаки типа "Отказ в обслуживании" используют ответственность сервера за обработку запросов, такие атаки работают путем перегрузки сервера чрезмерной частотой запросов. Шифрование должно использоваться, если конфиденциальная информация должна передаваться между клиентом и сервером.

1 Архитектура системы

Архитектура системы в общих чертах представляет собой сеть внутри другой сети. Всего в системе 2 уровня, и на каждом уровне есть сервер этого уровня. Итак, существует базовый сервер для работы с протоколами стека IP (в основном с UDP), который развертывает между своими клиентами другую сеть, работающую по собственному серверному протоколу (basic server protocol).

В рамках сети, построенной по протоколу базового сервера, работает сервер сообщений, который отвечает за обслуживание клиентов чата.

Для функционирования системы необходимо создать функционирующую IP-сеть. Вам необходимо запустить базовый сервер в этой сети. Сервер сообщений (один) должен быть подключен к базовому серверу. Каждый клиент чата должен сначала войти в IP-сеть базового сервера, затем подключиться к базовому серверу, а затем войти на сервер сообщений. Кроме того, клиент может использовать доступные функциональные возможности сервера сообщений для обмена данными с другими клиентами.

В целом архитектура показана на рисунке 3. Зеленые линии на рисунке отражают подключение на уровне IP (в центре - маршрутизатор). Желтые линии представляют связь на уровне протокола базового сервера (на рисунке слева). Изгиб линии отражает путь сообщения на этом уровне, а также "искажение маршрута" из-за характера базового протокола. Таким образом, связь на уровне базового сервера проходит через IP-сеть, но это происходит прозрачно для клиентов базового сервера. Аналогично - для сервера сообщений (красная строка). Красная линия проходит поверх желтой, но на этот раз базовый сервер также прозрачен. На рисунке эта линия представлена в упрощенном виде (чтобы не загромождать диаграмму).

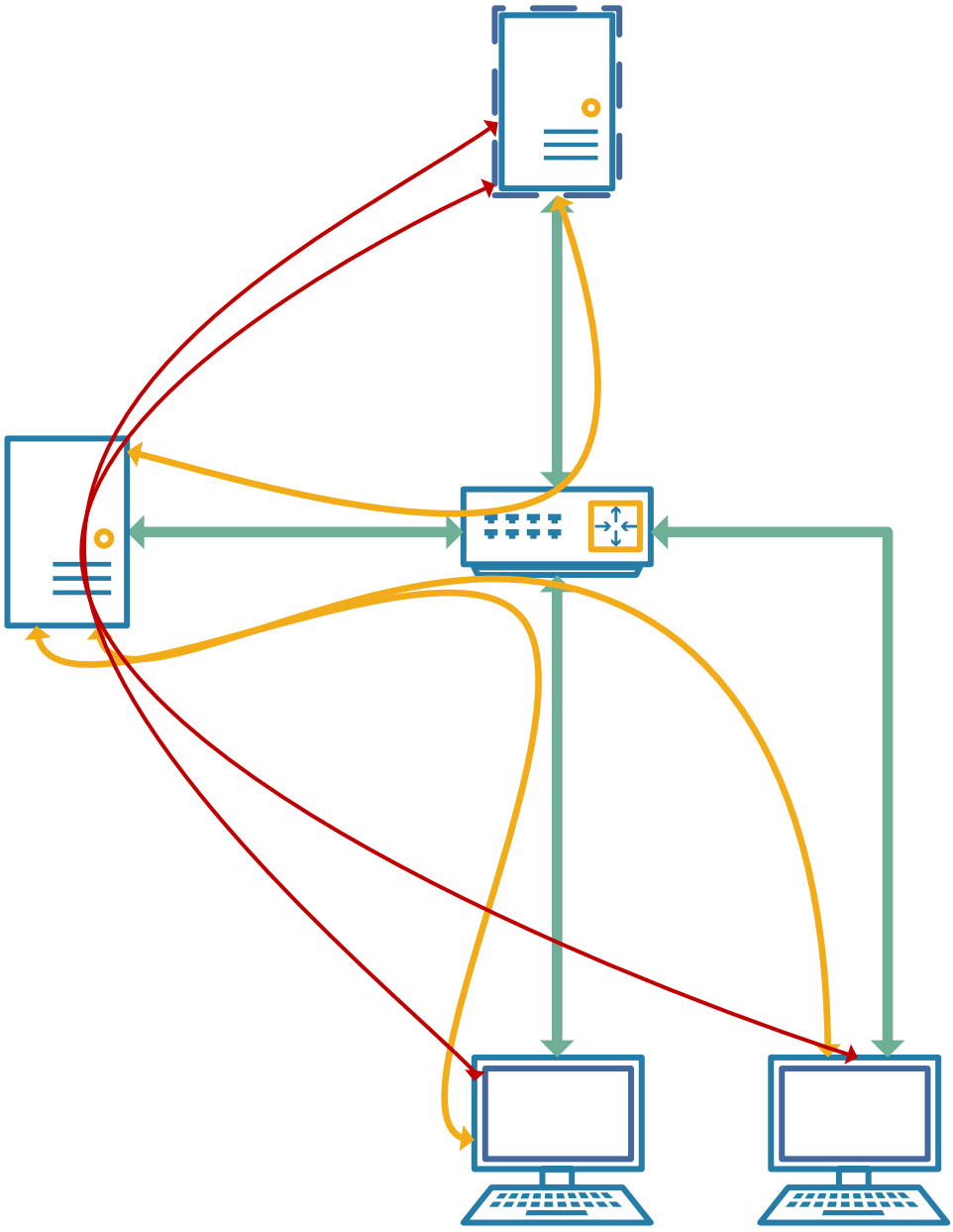

Рисунок 3 – архитектура системы в общем виде

2 Сервер сообщений

2.1 Общий принцип общения и подключения

Протокол сервера сообщений построен поверх протокола базового сервера, поэтому, когда речь идет о пакетах сервера сообщений, подразумевается, что эти пакеты также содержат заголовки базового сервера и работают внутри него.

Как часть протокола, сервер транслирует свои объявления в широковещательном режиме, чтобы клиенты могли найти этот сервер. Объявление каждого из серверов представляет собой строку некоторого контента, который клиент оценивает и выбирает сервер на основе этого. Пример анонса: «QUALITATIVE VERY FAST RELIABLE VERY ORIGINAL FAST PRIVATE MESSAGE SERVER». Строка анонса создается путем выбора случайного количества случайных слов из следующих: «SERVER», «MESSAGE», «PRIVATE», «RELIABLE», «QUALITATIVE», «FAST», «VERY», «PRO», «EFFECTIVE», «ORIGINAL».

У каждого клиента есть свои "предпочтения", на основе которых делается выбор в пользу конкретного сервера. Предпочтением клиента является взвешенный граф ссылок (случайные веса) всех возможных комбинаций двух слов из массива, который представлен в программе квадратной матрицей со случайными значениями в ячейках.

После выбора сервера клиент начинает транслировать свой выбор в режиме широковещательной передачи, чтобы все серверы (и клиенты) могли видеть этот выбор.

Первоначальная идея заключается в том, что серверы могут корректировать свои объявления, чтобы увеличить количество клиентов (каждый сервер хочет иметь больше клиентов, чем другие). В то же время

клиенты могут со временем корректировать свои предпочтения, если они отличаются от объявления сервера.

Поскольку связь между серверами не была реализована, в основной серверной сети может быть только один сервер сообщений. Если существует более одного сервера сообщений, то клиенты одного сервера не смогут связаться с клиентами другого. Такая ситуация противоречит идее чата, где любой участник должен иметь возможность связаться с любым другим. Следовательно, должен быть ровно один сервер сообщений.

Если сервер сообщений не подключен к сети, клиенты будут ждать его появления, пока они не будут выключены или сервер не появится.

2.2 Структура пакетов

Чат-клиенты взаимодействуют с сервером, используя специальную структуру пакетов. Если мы обратимся к таблице формата пакетов UDP (таблица 6), мы можем отметить в ней поля, используемые непосредственно при общении сервером сообщений.

Таблица 6 – структура UDP-пакета

0	1	2	3	4	5	6	7	8	9	10
		type	control							addr.

Поле "тип" используется для хранения адреса отправителя пакета (в поле "addr." хранится только адрес получателя). Поле "control" содержит сквозные биты (4 бита), которые используются для указания команды (отправка от клиента на сервер) и кода успеха (отправка от сервера клиенту).

Следующий байт за номером 10 - это сами данные переменной длины (длина указана в поле "длина"). Частью этой длины является uid пакета (просто число), который занимает 4 байта (11-14 позиций) и присутствует во всех сообщениях клиента и сервера (таблица 7).

Таблица 7 – размещение uid в пакете

0-10	11	12	13	14
Заголовок базового сервера	uid пакета			

Содержимое следующих байтов зависит от инструкции.

Следующие команды определяются от клиента к серверу:

0 - подтверждение получения, uid содержит uid принятого пакета, никаких дополнительных байтов (14 байт - последний).

1 – запрос на регистрацию (таблица 8), uid – новый. Этот запрос выполняется только вне сеанса. Если сеанс уже существует, он либо остается неизменным (если пользователь тот же), либо завершается и создается новый (если пользователь другой). Сервер отправит ответ об успешном выполнении этой команды.

Таблица 8 – запрос регистрации

0-10	11-14	15	16…	16+L…
Заголовок базового сервера	uid	длина логина (L)	логин	пароль

2 – запрос на удаление регистрации (таблица 9), идентификатор пользователя не важен. Этот запрос выполняется только в контексте сеанса. Сервер отправит ответ об успешном выполнении этой команды.

Таблица 9 – запрос удаления регистрации

0-10	11-14	15…
Заголовок базового сервера	uid	пароль

3 – запрос на авторизацию (таблица 10), идентификатор пользователя не важен. Этот запрос выполняется только вне сеанса. Если сеанс уже существует, он либо остается неизменным (если пользователь тот же), либо завершается и создается новый (если пользователь другой). Сервер отправит ответ об успешном выполнении этой команды.

Таблица 10 – запрос авторизации

0-10	11-14	15	16…	16+L…
Заголовок базового сервера	uid	длина логина (L)	логин	пароль

4 – запрос на отмену авторизации (таблица 11), идентификатор пользователя не важен. Этот запрос выполняется только вне сеанса. Сервер не отправит ответ об успешном выполнении этой команды.

Таблица 11 – запрос отмены авторизации

0-10	11-14	15	16…	16+L…
Заголовок базового сервера	uid	длина логина (L)	логин	пароль

5 – запрос списка онлайн-пользователей (таблица 12), идентификатор пользователя не важен. Этот запрос выполняется в любом контексте. В ответ сервер отправит список.

Таблица 12 – запрос списка пользователей онлайн

0-10	11-14
Заголовок базового сервера	uid

6 - запрос списка онлайн-пользователей на всех серверах (межсерверное взаимодействие - не реализовано) - таблица 13. Этот запрос запрещен (сервер выдаст сообщение об ошибке при выполнении команды).

Таблица 13 – запрос списка пользователей онлайн на всех серверах

0-10	11-14
Заголовок базового сервера	uid

7 – запрос на отправку личного сообщения (таблица 14), uid новый. Этот запрос выполняется только в контексте сеанса. Указанный логин - это логин получателя сообщения. Формат даты отправки обсуждается в таблице 23. Сервер отправит ответ об успешном выполнении этой команды.

Таблица 14 – запрос отправки личного сообщения

0-10	11-14	15-18	19	20…	20+L…
Заголовок базового сервера	uid	дата отправки	длина логина (L)	логин	сообщение

8 – запрос на отправку сообщения всем (таблица 15), новый идентификатор пользователя. Этот запрос выполняется только в контексте сеанса. Сервер отправит ответ об успешном выполнении этой команды.

Таблица 15 – запрос отправки сообщения всем

0-10	11-14	15-18	19…
Заголовок базового сервера	uid	дата отправки	сообщение

9 - запрос на отправку сообщения всем, кто находится в Сети (таблица 16), uid является новым. Этот запрос выполняется только в контексте сеанса. Сервер отправит ответ об успешном выполнении этой команды.

Таблица 16 – запрос отправки сообщения всем, кто онлайн

0-10	11-14	15-18	19…
Заголовок базового сервера	uid	дата отправки	сообщение

Следующие команды определяются от сервера к клиенту:

0 - подтверждение получения, uid содержит uid принятого пакета, никаких дополнительных байтов (14 байт - последний).

1 – ошибка при выполнении команды (таблица 17), uid является новым.

Таблица 17 – ошибка при выполнении команды

0-10	11-14	15…
Заголовок базового сервера	uid	текстовая расшифровка

2 – команда выполнена успешно (таблица 18), uid является новым.

Таблица 18 – команда выполнена успешно

0-10	11-14	15…
Заголовок базового сервера	uid	описание ошибки

5 – список пользователей онлайн (таблица 19), uid является новым.

Таблица 19 – список пользователей онлайн

0-10	11-14	15	16…	16+L1	16+L1…	…
Заголовок базового сервера	uid	длина L1	логин	длина L2	логин	…

7 – входящее личное сообщение (таблица 20), новый идентификатор пользователя. Указанный логин - это логин отправителя сообщения.

Таблица 20 – входящее личное сообщение

0-10	11-14	15-18	19	20…	20+L…
Заголовок базового сервера	uid	дата отправки	длина логина (L)	логин	сообщение

8 – входящее сообщение для всех (таблица 21), новый идентификатор пользователя. Указанный логин - это логин отправителя сообщения.

Таблица 21 – входящее сообщение для всех

0-10	11-14	15-18	19	20…	20+L…
Заголовок базового сервера	uid	дата отправки	длина логина (L)	логин	сообщение

9 - входящее сообщение для всех, кто находится в Сети (таблица 22), идентификатор пользователя новый. Указанный логин - это логин отправителя сообщения.

Таблица 22 – входящее сообщение для всех, кто онлайн

0-10	11-14	15-18	19	20…	20+L…
Заголовок базового сервера	uid	дата отправки	длина логина (L)	логин	сообщение

Таблица 23 – формат поля даты отправки

31-26	25-22	21-17	17-0
год считая от 2022	месяц	день месяца	время суток в секундах

2.3 Общий принцип работы с пакетами

Технологическая схема упаковки показана на рисунке 4. В случае сервера поток обработки объединяется с потоком отправки. Однако логически это разные части потоковой процедуры (сначала выполняется обработка, затем отправка).

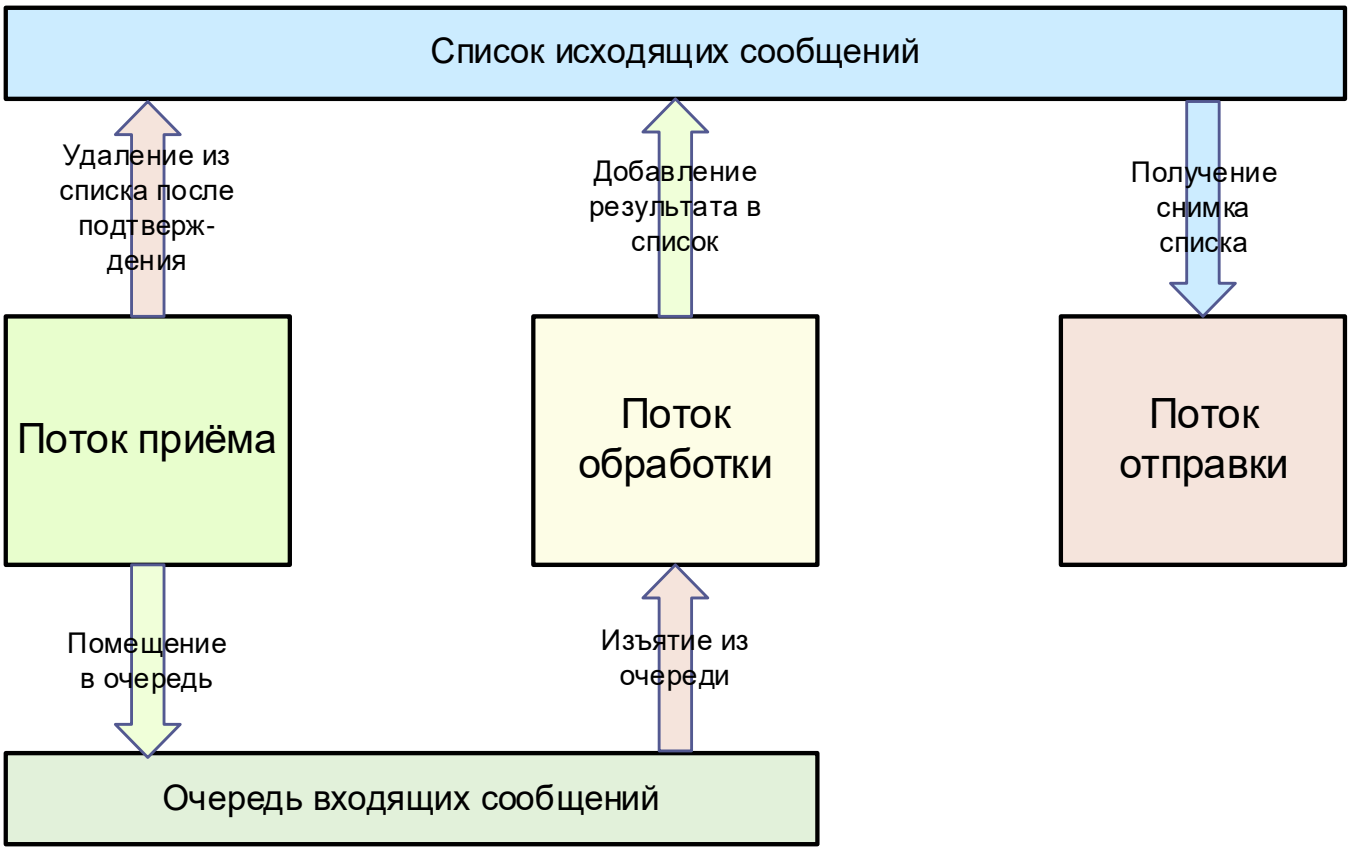

Рисунок 4 – общий принцип отправки и получения пакетов

Схема, показанная на рисунке 4, предназначена для обеспечения доставки пакетов по протоколу UDP.

Проблемы дублирования и смешивания пакетов решаются с помощью поля uid, содержащегося в пакете. В то же время это поле работает по-разному в пакетах для сервера и для клиента.

В случае сервера каждый сеанс каждого пользователя (не более одного сеанса на пользователя) связан с минимально допустимым uid.

Если в течение сеанса приходит пакет с идентификатором, меньшим или равным uid, то пакет отбрасывается. Если пакет поступает с большим uid, то пакет принимается, и новый uid записывается в данные сеанса. Этот метод подходит, поскольку у клиентов сервера сообщений нет сценариев, в которых одному клиенту необходимо отправлять несколько пакетов параллельно (то есть, не дожидаясь ответа на получение).

Однако у сервера бывают ситуации, когда он отправляет несколько сообщений параллельно, не дожидаясь ответа от клиента на получение каждого из них. Это происходит, например, когда несколько клиентов пишут сообщения третьему клиенту примерно в одно и то же время. Сервер будет параллельно отправлять сообщения третьему клиенту. Если бы клиент работал точно так же, как сервер, то гарантированно было бы доставлено только последнее сообщение, и хотя предыдущее сообщение получило бы ответ о получении, в действительности его можно было бы просто отбросить, поскольку его минимальный uid меньше ожидаемого.

Чтобы решить эту проблему на клиентах, используется комбинация использования набора принятых uid и минимального uid. Коллекция содержит 256 позиций для хранения uid. Когда количество превышает 256, наименьший из них извлекается из коллекции и устанавливается в качестве минимального uid. Таким образом, реализован гарантированный прием до 256 пакетов, отправляемых сервером параллельно, и этого достаточно, так как базовый сервер имеет ограничение на максимальное количество клиентов - 256 (с адресами от 0 до 255).

2.4 Сессии клиентов

Сервер мог бы обойтись без использования сеансов - достаточно было бы указать логин и пароль отправителя в каждой команде - и все. Это выглядит не очень удобно, но реальная причина использования сеансов кроется в трех других моментах:

- наличие сеанса позволяет реализовать создание списка онлайн-клиентов максимально естественно и логично;
- наличие сеанса позволяет клиентам иметь индивидуальный минимальный uid;
- наличие сеанса также подразумевает наличие регистрации, что, в свою очередь, играет важную роль в механике отправки сообщений.

Если бы не было регистрации, любой клиент существовал только пока он был онлайн, а поскольку полноценная реализация сеанса без регистрации невозможна, онлайн пришлось бы определять косвенно (по наличию потока сообщений от клиента, это была бы “сессия”). В этом случае было бы невозможно отправить сообщение автономному клиенту, поскольку у сервера просто не было бы координат запрошенного клиента (сервер не знал бы, существует ли такой клиент вообще). В этом случае внедрение активного сервера сообщений было бы практически невозможно.

Поскольку есть регистрация, а вместе с ней и сервер в любой момент имеет хоть какие-то данные о клиенте (его логин и пароль), вы можете опираться на эти данные дальше и реализовывать работу с оффлайн-клиентами. Для других клиентов это ничем не будет отличаться от работы с онлайн-клиентами. В случае отправки сообщений автономным клиентам сервер просто сохраняет эти сообщения в профиле клиента, которому они были назначены. Когда клиент выходит в Интернет, эти сообщения немедленно отправляются ему и удаляются с

сервера. Именно здесь возможность параллельной отправки показывает себя гораздо более востребованной, чем могло показаться раньше, поскольку в этом случае с сервера может быть отправлено достаточно большое количество сообщений параллельно, и важно принять все. Способ приема вкл. параллельные сообщения, реализованные на клиенте (как обсуждалось ранее), позволяют вам это сделать.

Сервер также периодически запрашивает активных клиентов базового сервера (специальная команда базового сервера) и сравнивает полученный список со списком активных клиентских сеансов, что позволяет находить и завершать "зависающие" сеансы, то есть сеансы, которые были завершены неправильно: клиент отключился, и сервер не сообщил об этом.

Также стоит упомянуть "серый список" - это список пакетов для отправки клиентам, которые, вероятно, больше не доступны или эти пакеты просто не имеют к ним отношения. Пакеты из "серого списка" не требуют подтверждения доставки от клиента, но отправляются фиксированное количество раз (три), после чего удаляются из списка. Гарантии доставки в этом случае нет, но она и не нужна, так как сервер не уверен, что клиенту нужны эти пакеты. Обычно этот список включает ответы на запросы, либо сделанные вне сеанса, либо если клиент, отправивший запрос, закрыл сеанс до получения ответа. Сервер фиксирует тот факт, что клиент перешел в автономный режим, и передает пакет ответа в серый список.

3 Чат-клиент

У клиента несколько иной дизайн системы пакетов. Здесь уместно обратиться к схеме на рисунке 4 (общий принцип отправки и получения пакетов). В отличие от севера, клиентская функциональность потоков комбинируется по-разному.

Поток приема частично отвечает за обработку - он принимает входящие сообщения и сохраняет их в архиве сообщений. Пакеты, которые не являются сообщениями, отправляются в список запросов: если список запросов содержит запрос типа входящего пакета, то этот пакет закрывает запрос. Если нет запроса на пакет входящего типа, то он отбрасывается.

Поток обработки представлен основным потоком (потоком пользовательского интерфейса), поскольку он единственный, который размещает пакеты при отправке. Как правило, этот поток помещает пакет для отправки и ожидает ответа от сервера, помещая в список запросов типы всех возможных ответов. Ожидание ответа блокирует поток обработки.

4 Тестирование

При запуске базового сервера им будет задано несколько вопросов:

- IP для прослушивания;
- номер порта для прослушивания;
- максимальный адрес клиента.

IP имеет смысл указать либо 127.0.0.1 (для работы только на локальном хосте), либо 0.0.0.0 (прослушивание пакетов, поступающих на хост). Максимальное количество клиентов на базовом сервере ограничено 256, но поскольку сервер запрашивает максимальный адрес, вы можете указать только 255.

После запуска базового сервера вам необходимо запустить сервер сообщений. Он запросит IP-адрес и порт базового сервера.

После запуска сервера сообщений вы можете запустить чат-клиенты. Вам нужно выбрать пункт 1 (“подключиться”), ввести IP и порт базового сервера. Клиент напишет "Подключено!" (если соединение было успешным) и "Ожидание сервера сообщений...". На данный момент клиент ожидает объявления от сервера сообщений. При получении объявления клиент (из объявления) узнает адрес сервера сообщений и подключается к нему, что сопровождается сообщением “Сервер сообщений найден!”.