DOI 10.37882/2223–2966.2021.09.01

# СПОСОБЫ РЕАЛИЗАЦИИ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ ПОТОКОВОЙ ПЕРЕДАЧИ ДАННЫХ НА ОСНОВЕ TCP ПРОТОКОЛА

## METHODS FOR IMPLEMENTING CLIENT-SERVER DATA STREAMING APPLICATIONS BASED ON THE TCP PROTOCOL

***M. Ablyaev***

*Summary.* In this article discusses ways to implement client-server data streaming applications. The main data transmission protocols TCP and UDP are described, with an indication of their characteristics. The author provides the necessary requirements for the implementation of data streaming and describes their key advantages and limitations based on TCP and UDP protocols within the framework of this task. In the article, based on the results of the study, the author presents an algorithm that allows solving the identified limitations and increasing the performance of data streaming software implemented on the basis of the TCP protocol. As a result, it was concluded that the presented algorithm allows you to reduce the cost of developing network code, increase the efficiency of data streaming, and minimize the delays that occur with standard data streaming based on the TCP protocol.

*Keywords:* streaming data, protocol, TCP, UDP, client-server application, algorithm, networking.

***Абляев Марлен Рефатович***
*Преподаватель, ГБОУ ВО РК «Крымский инженерно-педагогический университет имени Февзи Якубова»*
*ablyaev.marlen@gmail.com*

*Аннотация.* В данной статье рассматриваются способы реализации клиент-серверных приложений потоковой передачи данных. Описываются основные протоколы передачи данных TCP и UDP, с указанием их характеристик. Автор приводит необходимые требования для реализации потоковой передачи данных и на основе TCP и UDP протоколов описывает ключевые их преимущества и ограничения в рамках данной задачи. В статье на основе результатов исследования автор приводит алгоритм, который позволяет решить выявленные ограничения и увеличить производительность программного обеспечения потоковой передачи данных реализованного на основе протокола TCP. В результате чего был сделан вывод о том, что представленный алгоритм позволяет снизить затраты на разработку сетевого кода, повысить эффективность потоковой передачи данных, а также минимизировать задержки, возникающие при стандартной потоковой передаче данных основе TCP протокола.

*Ключевые слова:* потоковая передача данных, протокол, TCP, UDP, клиент-серверное приложение, алгоритм, сетевое взаимодействие.

С развитием общества и информационных технологий растет потребность в информации и потребляемом контенте. В связи с этим меняются и подходы в организации предоставления передачи данных от сервера к клиенту и от клиента к серверу. Различные способы обмена данными между клиентом и сервером накладывают на разработчиков определённые ограничения, дают некоторые преимущества и создают требования в процессе разработки соответствующего программного обеспечения, как на стороне сервера, так и на стороне клиента. Разработчики в свою очередь стремятся усовершенствовать обмен данными для увеличения скорости передаваемой информации, минимизации потери пакетов и снижения нагрузки на клиентское и серверное оборудование. Использование различных типов протоколов, методов структурирования информации и алгоритмов формирования данных способствует появлению самых различных сценариев написания программного кода и создает определенные требования для сетевого оборудования.

В своей работе А. Миланович, С. Србльич и В. Срук (A. Milanovic, S. Srbljic, V. Sruk) [1] описывают, как различные аппаратные и программные компоненты персонального компьютера влияют на производительность двух наиболее часто используемых протоколов передачи данных в интернете, TCP и UDP. Авторы сравнивают скорость передачи данных для трех различных операционных систем: Windows 95, Windows NT 4.0 Workstation и Linux 2.1.132. Также показывают, как размер пакета данных влияет на скорость передачи данных.

Авторы Донджин Ли, Брайан Э. Карпентер и Невил Браунли (DongJin Lee, Brian E. Carpenter, Nevil Brownlee) [2] в своем исследовании раскрывают вопросы влияния тенденций роста приложений области потоковой передачи мультимедиа и влияние на характеристики трафика. В работе отображены результаты значительной изменчивости и признаки систематической тенденции с течением времени и широкий разброс использования номеров портов.

Мин Чжан, Маурицио Дуси, Вольфганг Джон и Чанцзя Чен (Min Zhang, Maurizio Dusi, Wolfgang John, Changjia Chen) [4] предоставляют сравнительный анализ передачи данных UDP и TCP в нескольких трассировках трафика, собранных из разных сетей и географических местоположений в разное время. Авторы описывают, что TCP по-прежнему имеет преимущество с точки зрения пакетов и байтов, но UDP часто отвечает за самую большую часть потоков в данном канале. Рассматривается анализ на основе портов и предполагается увеличение потоков UDP на анализируемых каналах, что происходит в основном из-за приложений P2P, использующих UDP для своего служебного трафика.

Сетевой обмен данными между двумя устройствами производится средствами различных протоколов, которые формируют данные по определенным требованиям и отправляют получателю. В зависимости от протокола меняется и способы передачи данных, а именно контроль последовательности пакетов, проверка утерянных пакетов, проверка оригинальности пакета и другие подобные надстройки, которые могут варьироваться в зависимости от выбранного типа протокола.

В случае отсутствия определенной необходимой надстройки разработчиками создаются собственные алгоритмы, которые удовлетворяют возникшие потребности. Данные действия затрачивают рабочее время программистов и в зависимости от качества созданного алгоритма возрастает нагрузка на серверное и клиентское оборудование.

В случае избыточности существующих надстроек определенного протокола, что может влиять на производительность клиентского и серверного оборудования, зачастую, разработчиками выбирается другой, наиболее подходящий тип протокола.

Выбираемый тип протока напрямую зависит от архитектуры разрабатываемого приложения и определяет вектор разработки соответствующего проекта программного продукта. Существует большое количество протоколов передачи данных с собственными возможностями и ограничениями. Большая часть из них узконаправленные и практически не используется в массовой разработке. Это связано с тем, что для их полноценной работы требуется дополнительная программная настройка, соответствующая поддержка и вероятная настройка сетевых устройств. В связи с этим, широкую популярность в области разработки получили протоколы передачи данных TCP и UDP [3].

TCP (англ. Transfer Control Protocol — протокол управления передачей) — стандарт связи, который позволяет прикладным программам и цифровым устройствам обмениваться сообщениями по сети. Он предназначен для отправки пакетов через интернет и обеспечения успешной доставки данных и сообщений.

TCP является одним из основных стандартов, определяющих правила сети интернет, и включен в стандарты, определенные Инженерной группой Интернета (IETF). Это один из наиболее часто используемых протоколов в цифровых сетевых коммуникациях, обеспечивающий сквозную доставку данных.

Протокол организует данные таким образом, чтобы их можно передавать между сервером и клиентом. Это гарантирует целостность данных, передаваемых по сети. Перед передачей данных TCP устанавливает соединение между отправителем и получателем, которое, как он гарантирует, остается активным до конца связи. Затем он разбивает большие объемы данных на более мелкие пакеты, обеспечивая при этом целостность данных на протяжении всего процесса.

TCP может быть затратным по отношению к оборудованию сетевых инструментов, поскольку он включает в себя проверку отсутствующих или поврежденных пакетов и защищает доставку данных с помощью таких элементов управления, как подтверждения, запуск соединения и управление потоком.

TCP — надежный протокол, поскольку он следует алгоритму контроля потока и ошибок. Он также поддерживает алгоритм подтверждения, который проверяет состояние и прибытия данных. В алгоритме подтверждения получатель отправляет либо положительное, либо отрицательное подтверждение, чтобы отправитель мог узнать, был ли пакет данных получен или его нужно отправить повторно.

Данный протокол гарантирует, что данные достигнут предполагаемого получателя в том же порядке, в котором они были отправлены. Он упорядочивает и нумерует каждый сегмент данных, чтобы получатель на уровне TCP мог повторно собрать их в соответствии с предполагаемым порядком. Кроме этого, данные могут передаваться в обоих направлениях одновременно.

Протокол TCP поточно-ориентированный, поскольку он позволяет отправителю предавать данные в виде потока байтов, а также позволяет получателю принимать данные в виде потока байтов. Он создает среду, в которой отправитель и получатель соединены воображаемой трубкой, известной, как виртуальный канал. Этот виртуальный канал передает поток байтов через интернет.

Модель данного протокола насчитывает пять уровней — это уровень приложений, транспортный уровень, сетевой уровень, уровень канала передачи данных и физический уровень. Транспортный уровень играет важную роль в обеспечении сквозной связи непосредственно с процессами приложений. Он позволяет создавать 65 000 портов, чтобы можно было получить доступ к нескольким приложениям одновременно. Он берет данные с верхнего уровня, разделяет их на более мелкие пакеты и затем передает их на сетевой уровень.

TCP обеспечивает надежную передачу данных с установлением соединения, что гарантирует доставку пакетов. Если пакет данных потерян в сети, TCP отправит их повторно. Он устраняет перегрузку с помощью алгоритма предотвращения перегрузки сети, который включает в себя различные схемы, такие как аддитивное увеличение / мультипликативное уменьшение, медленный запуск и окно перегрузки.

К ограничениям данного протокола можно отнести большие накладные расходы на аппаратную часть, поскольку каждый сегмент получает свой собственный заголовок TCP, поэтому фрагментация маршрутизатором увеличивает на потребление вычислительной мощности расходы, что приводит к снижению производительности.

В результате все протоколы высокого уровня, которые должны передавать данные, используют протокол TCP. К примерам можно отнести методы однорангового обмена, такие как протокол передачи файлов (FTP), Secure Shell (SSH) и Telnet. Данный протокол также используется для отправки и получения электронной почты через протокол доступа к сообщениям в интернете (IMAP), протокол почтового отделения (POP) и простой протокол передачи почты (SMTP), а также для доступа в Интернет через протокол передачи гипертекста (HTTP).

UDP (англ. User Datagram Protocol — протокол пользовательских датаграмм) — это протокол транспортного уровня, который является частью Internet Protocol Suite и альтернативным протоколом связи протоколу управления передачей (TCP).

UDP — самый простой протокол транспортного уровня, предназначенный для отправки данных через интернет. Он выбирает дейтаграмму с сетевого уровня и прикрепляет заголовок, а затем пересылает его пользователю либо на сервер. Это быстрый, ненадежный протокол без сохранения состояния, что делает его подходящим для использования с приложениями, которые могут допускать потерю данных.

Его можно использовать для протоколов на основе транзакций, таких как DNS или протокол сетевого времени NTP.

Кроме этого, его можно использовать для решений, когда подключено много клиентов и где исправление ошибок в реальном времени не требуется, например, игры, голосовые вызовы или видеоконференции, а также потоковое видео.

Данный протокол не требует установления соединения, поскольку для передачи данных не требуется виртуальный канал.

Он предлагает минимальные транспортные услуги, при которых доставка до получателя не гарантируется, а также не предусмотрен механизм контроля перегрузок.

Кроме того, UDP использует заголовки для передачи данных по соединениям. Его заголовки содержат набор параметров, называемых полями. Заголовок UDP имеет четыре поля, а именно:

Порт источника: это 2-байтовое поле, в котором указывается номер порта отправителя.

Порт назначения: это также 2-байтовое поле, в котором указывается номер порта получателя.

Длина: это общая длина UDP, включая заголовок и данные. Это 16-битное поле.

Контрольная сумма: это поле длиной 2 байта, которое используется для проверки ошибок, например, в IPv6, а иногда и в IPv4.

UDP отправляет дейтаграмму (блок данных) с одного устройства на другое, используя интернет-протокол. UDP инкапсулирует данные в пакет и добавляет к нему информацию заголовка. Данные включают порт источника, порт назначения, длину пакета и контрольную сумму. После того, как пакеты UPD инкапсулируются в пакет интернет-протокола, они начинают перемещаться по назначению.

Учитывая описанные характеристики и принципы работы данных протоколов можно сделать вывод, что TCP более надежный, но медленный протокол, а UDP более быстрый, но не имеет достаточной надежности. Поэту большинство разработчиков не выбирают один конкретный протокол для разрабатываемого программного продукта, а используют комбинированное решение, в котором TCP протокол отвечает за авторизацию, пересылку сообщений и другую важную отправку ценной информации, а UDP применяется после авторизации или установления соединения для конкретной потоковой передачи данных, которые не требуют детальной проверки.

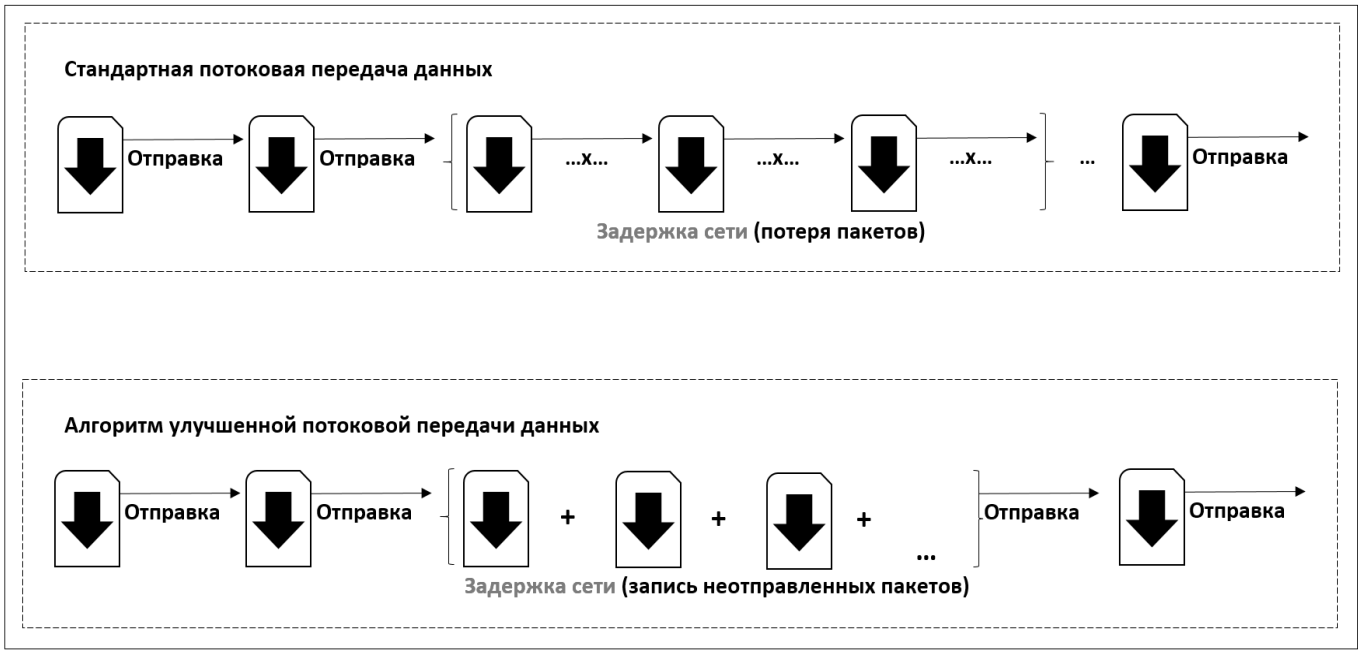


Рис. 1. Алгоритм потоковой передачи данных на основе TCP протокола

Использование подобного комбинированного способа передачи данных несет за собой последствия, такие как увеличение затрат и времени на разработку программного продукта, создание собственных алгоритмов над протоколом UDP для обеспечения нужд соответствующего разрабатываемого продукта, создание алгоритмов контроля пакетов UDP протокола и другие.

Для того чтобы этого избежать в проект необходимо внедрять и в дальнейшем использовать один протокол TCP. Но применение данного протокола в потоковой передаче данных несет за собой значительное снижение скорости, что является существенным недостатком и фактически задуманная потоковая передача перестает быть таковой, в результате чего на стороне клиента начнут появляться задержки и отставания в отправке или получении данных.

TCP протокол имеет возможность асинхронно отправлять и принимать данные, но получаемые данные от сервера или клиента считываются устройством последовательно. То есть данные доставляются поочередно и пока не дойдет первый пакет, получателю не будет доступен следующий пакет. Данный алгоритм работы TCP является одной из существенных причин замедления передачи данных.

Кроме это при помощи TCP протокола на стороне отправителя можно отслеживать статус отправленного пакета. Это дает возможность для реализации большого количества различных алгоритмов, направленных на отслеживание, детальное структурирование контролируемых протоколом пакетов.

Используя данную функциональную составляющую протокола TCP можно реализовать алгоритм, который будет основан на отправке сформированного пакете, а во время ожидания на стороне отправителя будет осуществляться формирование следующего пакета. После успешной доставки, начнется отправление уже сформированного пакета (рисунок 1).

Таким образом, во время обычной потоковой передачи данных отправитель либо ждет пока информация будет доставлена, либо продолжит отправку данных даже если до получателя не дошел предыдущий пакет. Соответственно, в первом случае появляются задержки и отставания, так как отправитель бездействует во время отправки данных. Во втором случае на стороне отправителя бессмысленно затрачиваются ресурсы устройства, а на стороне получателя возможна потеря пакетов, что может привести к потери данных, некорректной обработки полученной информации или получению данных в неправильной последовательности. Для того чтобы это решить, на стороне получателя и на стороне отправителя необходимо реализовывать дополнительный функционал, который будет отвечать требованиям каждой отдельно взятой ситуации.

Предложенный алгоритм решает вышеуказанные проблемы. Проток TCP на своем уровне обеспечивает корректную отправку сформированных пакетов,

а на программном уровне происходит формирование для последующей отправки.

Первым этапом происходит подключение, которое остается активным до отключения одной из сторон. Далее в пакет собираются необходимые первичные данные и начинается безопасная отправка. Во время ожидания ответа от получателя, в новый пакет записываются новые данные. Соответственно, чем больше время ожидания, тем больше данных будет записано в пакет. Количество записываемых данных ограничивается лишь размером пакета, который может быть изменен разработчиком. Как только отправитель будет уведомлен об успешной отправке, начнется следующая передача заново сформированного пакета. Таким образом на стороне получателя во время задержки, на аппаратном и программном уровне, конечному пользователю не будет заметно отставаний и замедления скорости по сравнению с протоколом UDP. Кроме этого, дополнительным преимуществом данного алгоритма является настройка частоты передачи пакетов, т.е. сколько пакетов будет передаваться за одну секунду. В связи с этим, в случае возникновения кратковременных проблем с подключением к сети количество тактов передачи данных за одну секунду уменьшится, но при этом последующие отправленные пакеты будут содержать информацию, которая должна была передаваться во время возникшей задержки. Алгоритм работает таким образом, что возникшие задержки компенсируются разбиением неотправленной информации на сегменты и поочередным добавлением ее в отправляющиеся пакеты. На стороне получателя эта информация распаковывается уже в правильной последовательности, без потери данных и с указанием дополнительной информации. Данный алгоритм можно настраивать под различные сценарии и добавлять дополнительный функционал для более детального обозначения и обработки данных, которые были отправлены с задержкой, что в свою очередь является значительным с своей мере для оптимизации и повешения производительности потоковой передачи данных.

В результате чего данный алгоритм обеспечивает потоковую передачу данных с использованием одного типа протокола TCP без дополнительных затрат на реализацию надстроек над протоколом. Кроме этого, алгоритм имеет уже существующий функционал безопасной передачи данных и решает проблемы снижения скорости и производительности. В связи с этим можно сделать вывод, что использование данного алгоритма является наиболее рациональным для потоковой передачи данных с экономией ресурсов аппаратной части реализуемого программного продукта, затрат времени на разработку и минимизации задержек при отправке и получении данных.

ЛИТЕРАТУРА

1. A. Milanovic, S. Srbljic and V. Sruk, "Performance of UDP and TCP communication on personal computers," 2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No.00CH37099), 2000, pp. 286–289 vol.1, doi: 10.1109/MELCON.2000.880422. URL: https://ieeexplore.ieee.org/abstract/document/880422/authors#authors.
2. D. Lee, B.E. Carpenter and N. Brownlee, "Observations of UDP to TCP Ratio and Port Numbers," 2010 Fifth International Conference on Internet Monitoring and Protection, 2010, pp. 99–104, doi: 10.1109/ICIMP.2010.20. URL: https://ieeexplore.ieee.org/abstract/document/5476872/authors#authors.
3. I. Coonjah, P.C. Catherine and K. M.S. Soyjaudah, "Experimental performance comparison between TCP vs UDP tunnel using OpenVPN," 2015 International Conference on Computing, Communication and Security (ICCCS), 2015, pp. 1–5, doi: 10.1109/CCCS.2015.7374133. URL: https://ieeexplore.ieee.org/abstract/document/7374133/authors#authors.
4. M. Zhang, M. Dusi, W. John and C. Chen, "Analysis of UDP Traffic Usage on Internet Backbone Links," 2009 Ninth Annual International Symposium on Applications and the Internet, 2009, pp. 280–281, doi: 10.1109/SAINT.2009.65. URL: https://ieeexplore.ieee.org/abstract/document/5230775/authors#authors.



Журнал «Современная наука: актуальные проблемы теории и практики»