УДК 004.772

СРАВНЕНИЕ ПРОТОКОЛОВ FILE TRANSFER PROTOCOL И HYPERTEXT TRANSFER PROTOCOL ПРИ ПЕРЕДАЧЕ ФАЙЛОВ В СЕТИ С ПОМЕХАМИ

Макушин А.В., Синявина А.П.

ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Москва,
e-mail: makushinav@student.bmstu.ru, sinyavinaap@student.bmstu.ru

Цель работы заключается в сравнении прикладных сетевых протоколов File Transfer Protocol и HyperText Transfer Protocol при передаче данных с точки зрения анализа их производительности в условиях возникновения помех в сети. В статье приведены результаты ряда проведенных экспериментов, моделирующих среду передачи данных, содержащую помехи, влияющие на скорость доставки пакетов от сервера клиенту. При проведении эксперимента осуществлялась передача файлов разного размера по внутренней сети между двумя виртуальными машинами. На одной из них были развернуты серверы FileZilla и Nginx для передачи по File Transfer Protocol и HyperText Transfer Protocol соответственно. На вторую машину были установлены клиентские программы для скачивания файлов посредством данных протоколов. Помехи в сеть вносились с помощью специальной утилиты и предполагали потерю, дупликацию и повреждение пакетов, а также разрыв соединения. В рамках одного эксперимента было проведено несколько передач, для каждой из которых осуществлялся замер времени; после значения были приведены к среднему арифметическому. На основании анализа полученных результатов сделан и обоснован вывод о схожих показателях скорости передачи при использовании обоих протоколов с учетом функционала, предоставляемого ими по умолчанию и не предполагающего установки дополнительного программного обеспечения.

Ключевые слова: протокол, протокол передачи файлов, протокол передачи гипертекста, передача файлов, помехи в сети, клиент, сервер

COMPARISON OF FILE TRANSFER PROTOCOL AND HYPERTEXT TRANSFER PROTOCOL FOR FILE TRANSFER IN A NETWORK WITH INTERFERENCE

Makushin A.V., Sinyavina A.P.

Bauman Moscow State Technical University, Moscow,
e-mail: makushinav@student.bmstu.ru, sinyavinaap@student.bmstu.ru

The aim of the study is to compare the application network protocols File Transfer Protocol and HyperText Transfer Protocol in data transfer from the point of view of analysing their performance under the conditions of interference in the network. The article presents the results of a number of experiments modelling the data transmission environment with the interference affecting the speed of packet delivery from the server to the client. The experiment involved transferring files of different sizes over an internal network between two virtual machines. On one of them were deployed «FileZilla» and «Nginx» servers for transfer via File Transfer Protocol and HyperText Transfer Protocol respectively. On the second machine there were installed client programmes for downloading files via the given protocols. Interference to the network was introduced using a special utility and implied packet loss, duplication and corruption, as well as connection breakage. Within one experiment several transmissions were conducted, for each of which time was measured; afterwards the values were summarised to the arithmetic mean. On the basis of the analysis of the obtained results the conclusion about similar indicators of transmission speed when using both protocols is made and substantiated taking into account the functionality provided by them by default and not assuming installation of additional software.

Keywords: protocol, File Transfer Protocol, HyperText Transfer Protocol, file transfer, network interference, client, server

Введение

В настоящее время возрастают требования к скорости и постоянству передачи данных в информационных сетях: это достигается с помощью модернизации сетевого оборудования, а также посредством внедрения более эффективных протоколов [1-3]. Тем не менее рассмотрение вопроса обеспечения быстродействия в сети с точки зрения оборудования и протоколов физического и сетевого уровней не учитывает ключевую роль протоколов прикладного уровня модели Department of Defense (DoD). При этом данные протоколы непосредственно определяют, как данные обрабатываются, передаются и принимаются на уровне конечных пользователей.

Несмотря на активное развитие за счет широкого представления среди других протоколов, внедрение новых прикладных протоколов также сталкивается с вызовами: от обновления сетевой инфраструктуры до обеспечения совместимости различных частей информационных систем [2; 4]. В связи с этим традиционные протоколы, такие как File Transfer Protocol (FTP) и HyperText

Transfer Protocol (HTTP), продолжают широко использоваться.

Поскольку в основе FTP и HTTP лежит идентичный стек технологий [5, с. 623–624], возникает вопрос, какой из данных протоколов при различных условиях предоставляет большую скорость передачи данных по сети.

Цель работы заключается в определении наиболее производительного протокола из сравниваемых HTTP и FTP при передаче файлов по сети, содержащей помехи.

Материалы и методы исследования

В рамках работы был проведён ряд экспериментов, сравнивающих скорость работы протоколов HTTP и FTP в рамках сценария передачи файлов разного размера по сети. При этом в экспериментах не применялись методы оптимизации передачи, предлагаемые различными реализациями протоколов.

В реальных условиях передача файлов осуществляется между двумя различными устройствами, взаимодействующими друг с другом по сети. Проведение эксперимента требовало наличия изолированной и контролируемой среды. Данные условия были достигнуты с помощью средств виртуализации: созданы две виртуальные машины, размещенные на одном физическом устройстве [6]. В рамках данной работы использовалась программа для виртуализации Oracle VM Virtual Box, на каждую используемую виртуальную машину было выделено 8 виртуальных ядер процессора, 4 ГБ оперативной памяти и 25 ГБ дискового пространства, были добавлены виртуальные сетевые адаптеры для работы с локальной сетью и для выхода в Интернет через физическое устройство, на котором работают машины [7]. В качестве операционной системы был установлен дистрибутив Linux «Ubuntu».

Для передачи данных между виртуальными машинами была произведена настройка внутренней сети, предполагающая включение сетевого адаптера с типом подключения «Внутренняя сеть». Таким образом, машины были объединены в локальную сеть. Для установления связи между виртуальными машинами необходимо, чтобы каждой из них соответствовал уникальный Internet Protocol (IP) адрес. В связи с этим была произведена установка статических адресов с помощью утилиты ip [8, с. 41–42]. Это позволило «напрямую» связать машины в виртуальной сети, что соответствует соединению устройств через патч-корд или коммутатор в физической среде.

Для настройки FTP-сервера было установлена программа FileZilla Server, к преимуществам которой можно отнести кроссплатформенность, бесплатную лицензию и простоту настройки. В связи с этим FileZilla является одним из самых популярных серверов, поддерживающих протокол FTP. Также программа позволяет рассматривать базовый протокол FTP без дополнительных расширений, что позволяет сузить область влияния внешних переменных, обеспечивая стабильную тестовую среду. После установки и запуска программы был создан пользователь и заданы его параметры доступа к точке монтирования, содержащей файлы для передачи между машинами. Дополнительно были отключены механизмы реализации FTP, ускоряющие передачу данных: сжатие данных на лету, многопоточная передача и долгосрочное кеширование. В качестве публичного IP-адреса сервера был использован настроенный ранее адрес в локальной сети.

Для настройки HTTP-сервера был выбран «nginx», поскольку он является одним из самых используемых веб-серверов и обладает высокой производительностью и гибкостью. Функциональность HTTP-сервера «nginx» включает в себя раздачу файлов. Для раздачи из локального каталога в файле конфигурации /etc/nginx/nginx.conf был отредактирован блок «http», внутрь которого был добавлен блок «server» с настроенным блоком «locations». В блоке «locations» был задан префикс, сравниваемый с Uniform Resource Identifier (URI) из запроса для перенаправления данного запроса в директорию, указанную в директиве «alias», что обеспечивает доступ к запрашиваемому ресурсу (листинг 1) [9, с. 16–19].

```
http {
   server {
      listen 192.168.1.10:8081;
         location /files/ {
            autoindex on;
            alias /local_dir/;
   }}}
```

Листинг 1. Настройки HTTP-сервера

Для проведения экспериментов были созданы тестовые файлы различных размеров: 100 МБ, 1 ГБ и 5 ГБ – для проверки работы протоколов при передаче малых, средних и больших объемов информации. Генерация файлов осуществлялась на «серверной» виртуальной машине с использованием утилиты fallocate, позволяющей быстро создавать файлы заданного размера с произвольным содержимым. Пример команды для создания файлов представлен в листинге 2.

```
fallocate -l 1G test_1GB.txt
```

Листинг 2. Команда для создания файла заданного размера

С помощью утилиты tc были установлены различные ограничения на входящий и исходящий трафик с виртуального адаптера. Для всех проведённых экспериментов была задана пропускная способность канала, равная 100 МБит/с; это значение можно считать максимально возможной скоростью передачи, поскольку локальная сеть использовалась только для передачи файлов. Вводились следующие виды помех: потеря (2%), повреждение (2%) и дублирование (5%) пакетов. Пример установки помехи приведен в листинге 3. В рамках каждого отдельного эксперимента устанавливалась только одна помеха. Данные уровни (вероятность возникновения помехи) были выбраны экспериментально: при превышении указанных уровней скорость передачи падает до менее чем 1 МБит/с, что заметно увеличивает время исследования без привнесения дополнительных эффектов. Все ограничения устанавливались на машине, с которой происходила «отдача» файлов.

```
tc qdisc add dev enp0s8 root netem loss 2%
```

Листинг 3. Установка потери 2% пакетов с помощью утилиты tc

Также был рассмотрен сценарий, при котором происходит разрыв соединения на некоторый отрезок времени: 5, 30, 60 или 120 с. Для этого был написан bash-скрипт, производящий физическое отключение сетевого адаптера на принимающей машине с помощью утилиты iptables [8, с. 113–117; 10, с. 356–360]. В листинге 4 представлен вариант скрипта для разрыва соединения на 120 с.

```
#!/bin/bash
INTERFACE=”enp0s8”
iptables -A INPUT -i “$INTERFACE” -j DROP
iptables -A OUTPUT -o “$INTERFACE” -j DROP
sleep 120
iptables -D INPUT -i “$INTERFACE” -j DROP
iptables -D OUTPUT -o “$INTERFACE” -j DROP
```

Листинг 4. Скрипт для имитации разрыва соединения на 120 с

После настройки тестовой среды был разработан алгоритм для измерения скорости передачи данных по протоколам FTP и HTTP, предусматривающий последовательную загрузку файлов заданного размера с фиксацией времени передачи. После каждой успешной загрузки файл удалялся, и устанавливалась пауза длительностью 30 с для обеспечения очистки кеша как на стороне сервера, так и на стороне клиента. По истечении паузы процедура передачи повторялась. Для каждого файла проводилось 5 последовательных итераций для минимизации влияния случайных факторов на результат и обеспечения репрезентативности измерений. Данный подход применялся для обоих протоколов и всех типов сетевых помех. Для автоматизации данного процесса были разработаны специальные скрипты, выполняющие последовательность шагов алгоритма для передачи по FTP (листинг 5) и HTTP (листинг 6) без вмешательства пользователя [10, с. 436–439]. Для загрузки данных по FTP использовалась утилита lftp с поддержкой командного интерфейса и гибкой настройки параметров передачи, что позволяет использовать её в автоматизированных сценариях, описанных выше. Для HTTP применялась утилита wget как одна из наиболее распространённых для загрузки файлов по HTTP [10, с. 207].

```
#!/bin/bash
for size in 100MB 1GB 5GB do
     for ((i=0;i<=5;i++)); do
          time lftp -e “set ssl:verify-certificate no; get /local_
dir/test_$size.txt; bye” -u ftpuser,11 ftp://192.168.1.10
          rm test_$size.txt
          sleep 30
     done
done
```

Листинг 5. Скрипт для проведения эксперимента с протоколом FTP

```
#!/bin/bash
for size in 100MB 1GB 5GB do
     for ((i=0;i<=5;i++)); do
          time
wget http://192.168.1.10:8081/files/test_$size.txt
          rm test_$size.txt
          sleep 30
     done
done
```

Листинг 6. Скрипт для проведения эксперимента с протоколом HTTP

В ходе экспериментов виртуальные машины не подвергались дополнительной нагрузке для исключения влияния фоновых процессов на производительность сети. Также был отключен переход в режим сна и энергосбережения для избежания прерывания экспериментов.

Результаты исследования и их обсуждение

На основании проведённых экспериментов были собраны данные о времени передачи файлов различных размеров при разных типах сетевых помех для FTP и HTTP. Для каждого случая было выполнено 5 последовательных измерений; в качестве итогового значения времени передачи использовалось среднее арифметическое по результатам этих измерений.

Также для сравнения протоколов была определена разница во времени передачи, рассчитываемая по формуле:

$$\text{Разница} = \frac{t_{HTTP} - t_{FTP}}{t_{FTP}} \cdot 100\%,$$

где t_{HTTP} и t_{FTP} – среднее время передачи по HTTP, FTP соответственно. Данное значение показывает, насколько быстрее происходит передача посредством FTP по сравнению с HTTP. Полученные результаты отражены в таблице 1.

Результаты эксперимента показывают, что при отсутствии сетевых помех время передачи по HTTP и FTP практически совпадает для средних и больших файлов. Незначительные различия объясняются скорее деталями реализации используемых инструментов, а не особенностями протоколов. Тем не менее при отсутствии помех передача малых файлов по HTTP происходит в среднем на 8% быстрее. Это можно объяснить тем, что FTP имеет более долгую процедуру инициализации передачи, требующую установления соединения для передачи команд, проведения аутентификации, запроса файла, открытия соединения для его передачи. В связи с этим на коротких временных отрезках передачи малого объема данных подобные задержки более заметны.

При появлении помех FTP в ряде случаев обеспечивает более стабильную и быструю передачу. Это может быть связано с тем, что FileZilla, применяемая в качестве FTP-сервера, оптимизирована для быстрых переотправок повреждённых или потерянных пакетов.

Таблица 1

Сравнение времени передачи по протоколам HTTP и FTP

Вносимая помеха	Размер файла	Время передачи по протоколу, с		Разница, %
		t_{HTTP}	t_{FTP}	
Отсутствует	100 МБ	8,50	9,25	-8,11
	1 ГБ	83,87	84,07	-0,24
	5 ГБ	419,50	421,52	-0,48
Потеря пакетов (2%)	100 МБ	16,42	15,27	7,53
	1 ГБ	145,28	152,32	-4,62
	5 ГБ	758,84	704,80	7,67
Повреждение пакетов (2%)	100 МБ	29,60	27,85	6,28
	1 ГБ	316,95	311,80	1,65
	5 ГБ	1568,01	1365,74	14,81
Дублирование пакетов (5%)	100 МБ	10,20	10,13	0,69
	1 ГБ	102,63	100,39	2,23
	5 ГБ	527,46	510,93	3,24

Также, согласно описанной ранее методике, были проверены сценарии разрыва соединения между машинами на интервалах в 5, 30, 60 и 120 с. Результаты тестов приведены в таблице 2.

Таблица 2

Результаты тестирования на разрыв соединения при передаче по HTTP и FTP

Время разрыва, с	Передача восстановлена	
	HTTP	FTP
5	Да	Да
30	Да	Да
60	Да	Да
120	Нет	Да

Видно, что передача по FTP оказалась более устойчивой к длительному разрыву соединения по сравнению с HTTP. Однако это также можно объяснить особенностями реализации FTP-сервера, который дольше сохраняет соединения открытыми. Аналогичного эффекта можно добиться и при использовании HTTP, например, указав заголовок Keep-Alive [9, с. 86].

Заключение

В ходе работы была проведена настройка тестовой среды для сравнения скорости передачи данных с использованием протоколов HTTP и FTP. Результаты экспериментов были обработаны и представлены в сравнительных таблицах.

На основании проведенного исследования сделаны выводы о производительности данных протоколов в условиях нестабильного сетевого соединения между узлами клиента и сервера. Поскольку оба исследуемых протокола используют TCP на транспортном уровне, их базовые характеристики передачи данных практически не отличаются. Наблюдаемые в экспериментах расхождения скорее обусловлены тонкостями реализации конкретных программных средств, а не принципиальными различиями между HTTP и FTP. Данный факт подчеркивает важность тщательного выбора и корректной настройки инструментов, а также осознанного подхода к оценке влияния сетевых условий на производительность конкретных реализаций протоколов.

Список литературы

1. Юхимук Р.А., Веревкин С.А. Анализ протоколов сетевого взаимодействия для повышения надежности, быстродействия и безопасности сети организации // Известия Тульского государственного университета. Технические науки. 2023. № 8. С. 286-296. DOI: 10.24412/2071-6168-2023-8-286-287.

2. Багиян Н.В., Беляев В.А., Тудвасев Д.А. Анализ современных сетевых протоколов: технологии и тенденции // Молодежь. Образование. Наука. 2024. № 1 (19). С. 328-332.

3. Лукичев А.Л., Лиманова Н.И., Козлов В.В. Сетевые протоколы нового поколения и их влияние на производительность сети // Тенденции развития науки и образования. 2024. № 110-17. С. 121-124. DOI: 10.18411/trnio-06-2024-944.

4. Биджиева С.Х., Шебзухова К.В. Сетевые протоколы передачи данных: преимущества и недостатки // Тенденции развития науки и образования. 2022. № 86-1. С. 43-45. DOI: 10.18411/trnio-06-2022-14.

5. Эндрю Таненбаум, Ник Фимстер, Дэвид Уэзеролл. Компьютерные сети. 6-е издание / Пер. с англ. С. Черникова. Под ред. М. Капранова. СПб.: Питер, 2023. 992 с.

6. Терешкин Д.О., Мартышкин А.И., Данилов Е.А. Исследование различных способов реализации сетевой инфраструктуры в виртуализированной среде // 21 век: итоги прошлого и проблемы настоящего плюс. 2021. № 1 (53). С. 51-56. DOI: 10.46548/21vek-2021-1053-0009.

7. Сукманов С.В. Современные технологии виртуализации // Актуальные проблемы экономики, социологии и права. 2016. № 2. С. 70-75.

8. Роб Ванденбринк. Linux для сетевых инженеров / Пер. с англ. А.Г. Гаврилова; под ред. Е. Строгановой. СПб.: Питер, 2024. 496 с.

9. Дерек де Йонге. NGINX. Книга рецептов / Пер. с англ. Д.А. Беликова; под ред. Д.А. Мовчана. М.: ДМК Пресс, 2020. 176 с.

10. Уильям Шоттс. Командная строка Linux. Полное руководство / Пер. с англ. А. Киселева; под ред. К. Тульцевой. СПб.: Питер, 2017. 480 с.