УДК 004.4

РАЗРАБОТКА ПРОГРАММНОГО КОМПОНЕНТА ДЛЯ КОМПРЕССИИ ПЕРЕДАВАЕМЫХ ПО СЕТИ ДАННЫХ

Ященков И.А. ФГБОУ ВО «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича»

Научный руководитель – преподаватель Кривоносова Н.В.

ФГБОУ ВО «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича»

ВВЕДЕНИЕ

Разработка большинства программных комплексов включает в себя проектирование такой архитектуры, в которой будут реализованы адаптивность и кроссплатформенность. Современные библиотеки и фреймворки предоставляют широкий спектр различных технологий для создания программного комплекса с учетом вышеуказанных требований. К сожалению, не всегда получается учесть одновременно оба этих фактора. Актуальность представленной системы обоснована необходимостью универсального компонента для подобных комплексов, позволяющего эффективно проводить компрессию и декомпрессию данных для получения и передачи различными модулями программы или клиент-серверной системы, использующих протокол TCP.

ОСНОВНАЯ ЧАСТЬ

Разработанная система представляет собой подключаемый к проекту модуль, который включает в себя библиотечные файлы и два класса: API компрессора, взятый из open-source библиотеки и классы, использующие предыдущий модуль, в которых определены необходимые переменные и методы для компрессии и декомпрессии получаемых и отправляемых пакетов.

API компрессора предоставляет методы для непосредственно компрессии и декомпрессии, а также метод, возвращающий объём сжатого пакета в самом неудачном случае, то есть размер исходного пакета без сжатия. Таким образом определяется размер буфера при компрессии данных, позволяющий не потерять информацию, которая потенциально не сожмется.

Классы работы с компрессором и декомпрессором используют в качестве входного и выходного буфера вектор целочисленных значений. Для его использования данные, подлежащие сжатию или декомпрессии, следует привести к указанному типу данных. Рекомендуется использовать массив байтов для хранения передаваемой или получаемой информации, так как он идеально подходит для приведения к типу вектор.

Также в данных классах определены три метода и два перегруженных оператора: метод сброса внутреннего буфера класса, метод проверки наличия данных во внутреннем буфере и метод чтения пакета входных данных, а также перегруженные операторы для процесса сжатия и декомпрессии информации, содержащейся в векторах целочисленных значений. Для каждой операции в модуле предусмотрен блокировщик предоставленного мьютекса, который позволяет перехватить используемый поток для процесса компрессии/декомпрессии, чтобы основная программа не использовала не до конца обработанные данные.

Первые два вышеуказанных метода не требуют дополнительного пояснения, так как содержат базовую логику. Отдельно следует описать метод чтения пакета и перегрузки операторов.

Классы именованы как входной и выходной потоки. Каждый имеет свои перегруженные операторы. Входной поток является классом для декомпрессии, а выходной – для сжатия данных. Перегрузки потоков загружают во внутренний буфер полученные векторы и считывают из буфера данные для компрессии или декомпрессии. В перегрузке входного потока используется вышеупомянутый метод чтения пакета, который находит в сжатых

данных байт с информацией о размере исходного пакета и возвращает его для дальнейшего использования.

Для описания принципа работы системы используется схема отправки сжатых данных с клиента и получения и обработки их на сервере. Алгоритм использования следующий:

В клиенте формируется пакет данных в виде массива байтов и инициализируется класс выходного потока. После этого массив байтов приводится к вектору целочисленных значений, который с помощью перегруженного оператора обрабатывается потоком: данные сжимаются и передаются во внутренний буфер. Затем, используя второй перегруженный оператор, поток возвращает в вектор выходного буфера сжатые данные, которые готовы к отправке. Рекомендуется перед отправкой привести вектор выходного буфера к массиву байтов.

После получения сервером передаваемого пакета, данные считываются в инициализированный входной буфер входного потока. Далее этот вектор с помощью перегруженного оператора обрабатывается входным потоком, загружается во внутренний буфер, после чего вызывается второй оператор, декомпрессирующий данные в выходной буфер. Наконец, готовые данные используются сервером для дальнейшей работы.

ВЫВОДЫ

Внедрение представленной системы осуществлено в приложение-клиент, написанное для обмена данных с сервером, в который также внедрена система, а также отображения полученной информации в GUI программы. Программный комплекс создан с целью демонстрации работы представленного модуля. В комплексе наглядно представлено решение проблемы компрессии JSON пакетов в фреймворке Qt для передачи по протоколу TCP.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Qt Documentation – URL: https://doc.qt.io/ (дата обращения 15.02.2023)
2. LZ4 - Extremely fast compression – URL: https://github.com/lz4/lz4 (дата обращения 15.02.2023).