# Передача файла по UDP

Отправитель отправляет получателю файл.

В один момент времени конкретному получателю отправитель может передавать только один файл.

## Структура пакета

struct msg\_udp\_t {

sockaddrin addr; // IP Адрес получателя пи отправке или отправителя при приеме

size\_t size; // Размер того что ниже

uint32\_t hash; // Хэш того что ниже размером size – sizeof(hash)

uint8\_t type; // Тип сообщения. Старшие 4 бита задаются случайно( для изменения хэша)

uint8\_t data[UDP\_MAX\_SIZE]; // Данные

}

Отправляется пакет начиная с hash размером size.

## Типы пакетов

MSG\_TYPE\_EMPTY – пустое сообщение

MSG\_TYPE\_INIT – инициализация передачи, передается имя файла, размер.

MSG\_TYPE\_DATA – блок данных (часть файла). Номер блока, данные.

MSG\_TYPE\_DATA\_ASK –блок данных (часть файла) с запросом подтверждения.

MSG\_TYPE\_COMMIT – подтверждение приема. Размер блока, номер последнего принятого, список пропущенных блоков.

MSG\_TYPE\_END – подтверждение окончания отправки.

## MTU

MTU определяется путем отправки нескольких MSG\_TYPE\_INIT разных размеров (1472, 1464, 1372, 1272).

Для передачи берется MTU того пакета, ответ на который придет первым.

## Протокол обмена

**Отправитель** получает полное имя файла, определяет его размер и посылает MSG\_TYPE\_INIT. Размер пакета отдельно с каждое MTU. Если не пришло подтверждение от получателя, то делается еще 4 попытки с интервалом 100 мс, 200 мс, 300 мс, 400 мс. После последнего ожидание 500 мс и отправка считается несостоявшейся.

**Получатель**, приняв MSG\_TYPE\_INIT, инициирует прием под MTU с размер пакета и отправляет подтверждение. Если прием уже инициирован, то повторное подтверждение отправляется только если размер пакета равен ожидаемому (чтобы не слать лишние подтверждения).

**Отправитель**, получив первое подтверждение, переходит в режим отправки.

**Отправитель** каждые 20 мс отправляет одно сообщение MSG\_TYPE\_DATA\_ASK и окно из N пакетов MSG\_TYPE\_DATA (в каждом очередной блок файла, если есть запрос потерянных то сначала их). Отправка может начаться раньше, если придет подтверждение последнего MSG\_TYPE\_DATA\_ASK. В этом случае отсчет 20 мс начинается с отправки первого пакета.

**Получатель** сохраняет принятые блоки и следит за порядком их поступления. Если получен блок не по порядку, то пропущенные заносятся в таблицу пропущенных. При получении ранее пропущенного он убирается из таблицы.

**Получатель** отправляет подтверждение MSG\_TYPE\_COMMIT (список пропущенных)

**Отправитель** по приходу MSG\_TYPE\_COMMIT вносит неполученные в список пропущенных и изменяет размер окна (N) в зависимости от количества потерянных пакетов.

**Получатель** при получении последнего блока отправляет MSG\_TYPE\_COMMIT для извещения об успешном окончании приема. Сохраняет файл. Переходит в состояние ожидания следующего файла (MSG\_TYPE\_INIT). На пустые MSG\_TYPE\_DATA\_ASK отвечает MSG\_TYPE\_COMMIT.

**Отправитель** по приходу MSG\_TYPE\_COMMIT, сообщающем об окончании приема получателем, посылает MSG\_TYPE\_END. Надо ли ?

## Структура отправителя

1. **Инициализация передачи**. Формируется 4 сообщения MSG\_TYPE\_INIT разных размеров (1472, 1464, 1372, 1272). Сообщение передается на расчет хэша и отправку.
2. **Расчет хэша и отправка**. Рассчитывается хэш сообщения и оно отправляется.

## Структура получателя

1. **Поток приема сообщений**. Принимает сообщение и отдает его на проверку хэша.
2. **Проверка хэша**. Далее в зависимости от типа
   1. MSG\_TYPE\_INIT – создание получателя и регистрация его в таблице получателей. Ключ для поиска IP:порт отправителя. Отправителю посылается подтверждение.
   2. Остальные. По IP:порт находится получатель и сообщение передается ему.
3. **Получатель**. Сохраняет принятые данные в соответствии с их порядком. В случае если отправитель запрашивает ответ – посылает номер последнего принятого и список потерянных пакетов.

### Отправка

На вход дается задание с именем файла и IP:порт куда доставить.

~~Отправляемому файлу присваивается номер (счетчиком).~~

~~Создаются акторы Читатель (для чтения файла) и Отправитель (для отправки)~~

## Сообщение

Тип: входящее/исходящее

Контрагент: IP:порт

Обработчик: указатель на актор

Данные: номер файла, номер блока, размер данных, данные.

## Порт

Хранит таблицу соответствия: IP:порт контрагента, номер сообщения, Актор обработчик для входящих пакетов.

Получает на вход сообщение:

- Исходящее – отправляет данные в UDP.

- Входящее – по таблице проставляет обработчика, передает на проверку Контролеру.

## Читатель

При создании открывает файл на чтение.

Получает запрос с номером блока, читает из файла, формирует сообщение для отправки, передает сообщение Контролеру.

## Контролер

Получает на вход заполненное сообщение и вычисляет контрольную сумму. Передает Отправителю.

## Отправитель

На вход принимает пустые пакеты для отправки (в пакете настроены )

## Контроль и шифрование

В старшие 4 разряда типа сообщения добавляется случайное число. Затем считается хэш в msg\_udp\_t::hash. Затем шифруется msg\_udp\_t::data, ключ шифрования – хэш.

~~По каждому сообщению перед отправкой считается хэш и записывается в msg\_udp\_t::hash~~

~~Затем сообщение шифруется ключом передачи.~~

~~Ключ передачи – случайное число, выбирается отправителем и передается при инициализации передачи (MSG\_TYPE\_INIT)~~

~~Сообщение MSG\_TYPE\_INIT шифруется ключом по-умолчанию (0x12345678)~~

~~Как вариант: через DH получить секрет и посчитать хэш с паролем и им~~

## Управление созданием сообщений

Для создания сообщений используется очередь пустых сообщений msg\_udp\_t\*. Для получения пустого сначала проверяется очередь, если там нет то создается новое.

Отработанные сообщения помещаются в конец очереди с типом MSG\_TYPE\_EMPTY.

При начале передачи/приема увеличивается счетчик обменов, при завершении – уменьшается. При переходе счетчика в ноль – очередь очищается.