Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Григоращенко А.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Кислун О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Григоращенку Артему Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent*

2. Керівник роботи *Кислун Олег Андрійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 35-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Григоращенко А.В. Дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи менеджера завантажень з використанням протоколу BitTorrent.

Метою розробки є дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Об'єктом дослідження є процес менеджера завантажень з використанням протоколу BitTorrent.

Предметом дослідження є методи менеджера завантажень з використанням протоколу BitTorrent.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi.

Ключові слова: комп'ютерна інженерія, BitTorrent

ABSTRACT

Hryhorashchenko A.V. Research and software implementation of the download manager system using the BitTorrent protocol. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the download manager system using the BitTorrent protocol.

The goal of the development is the research and software implementation of the download manager system using the BitTorrent protocol.

The object of the study is the download manager process using the BitTorrent protocol.

The subject of the study is download manager methods using the BitTorrent protocol.

Research methods are based on telecom theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the download manager system using the BitTorrent protocol.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi environment.

Keywords: computer engineering, BitTorrent

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛОМ	–	локальна обчислювальна мережа
ОП	–	оперативна пам'ять
ОС	–	операційна система
ПК	–	персональний комп’ютер
САЗ	–	система автоматизованого завантажування
ЦП	–	центральний процесор
DNS	–	сервер доменних імен
HTTP	–	HyperText Transfer Protocol – «протокол передачі гіпертексту»
HTTPS	–	Hypertext Transfer Protocol Secure
FTP	–	File Transfer Protocol – протокол передачі файлів
IP	–	Internet Protocol
LAN	–	локальна мережа
MAC	–	Media Access Control – управління доступом до носія
NAT	–	Network Address Translation
NTLM	–	NT LAN Manager – протокол мережної автентифікації
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol – протокол користувальницьких дейтаграм

ВСТУП

Актуальність теми. Торрент (torrent) – це мережний протокол для обміну файлами. Файли розбиваються на невеликі частини й у такому виді передаються по мережі. Торрент-клієнт (torrent-client) завантажує ці частини й потім збирає в себе файл воєдино. У процесі завантаження шматочків клієнт також віддає вже заколисані частини, що дозволяє передавати торренти з великою швидкістю й без очікування звільнення джерела (сідера, seed).

Для того щоб скачати торрент файл, клієнт з'єднується з торрент трекером (torrent tracker), передає йому інформацію про свою IP адресу й хеш суму файлу, що потрібно скачати. Трекер відправляє клієнтові IP адреси інших клієнтів, що також роздають або качають торрент. У процесі завантаження клієнт регулярно спілкується із сервером, повідомляючи інформацію про завантаження й одержуючи оновлений список IP адрес.

Клієнти передають інформація прямо між собою без участі торрент-трекеру. Трекер тільки збирає дані із клієнтів про процес завантаження, підключених клієнтах і іншій інформації. Для оптимальної роботи торрент протоколу потрібно, щоб максимальна кількість клієнтів могли приймати й віддавати файли. При некоректному налаштуванні міжмережний екран/брандмауера або трансляції адрес/NAT, швидкість передачі може значно зменшитися або припинитися зовсім.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем менеджера завантажень з використанням протоколу BitTorrent.

– Дослідження системи менеджера завантажень з використанням протоколу BitTorrent.

– Програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Об'єктом дослідження є процес менеджера завантажень з використанням протоколу BitTorrent.

Предметом дослідження є методи менеджера завантажень з використанням протоколу BitTorrent.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод менеджера завантажень з використанням протоколу BitTorrent.

– Розроблено вітчизняний продукт менеджера завантажень з використанням протоколу BitTorrent, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі менеджера завантажень з використанням протоколу BitTorrent.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації програмного забезпечення менеджера завантажень з використанням протоколу BitTorrent. Коли клієнти приєднуються друг до друга, вони відразу ж передають дані про шматочки торренту, наявного в них. Якщо в першого клієнта є відсутній сегмент, то другий клієнт посилає запит «скачати торрент». Перший клієнт віддає запитувану частину торренту, якщо є така можливість. Після одержання запитаної частини, другий клієнт перевіряє контрольну суму й доводить до відома про те, що він також має цю частину, що всі інші підключені клієнти могли скачати torrent з його.

Клієнт може призупинити віддачу частин torrent файлу іншому клієнтові, що робиться для оптимізації роздачі. Пріоритет віддається тому, що сам віддав більшу кількість частин, тобто чим більше віддав peer (пір), тим більше ви віддаєте йому. Завдяки такій особливості більшу швидкість завантаження одержують ті торрент клієнти, які віддають із великою швидкістю. Девіз torrent протоколу: «Скачав сам, дай скачати іншому!»

Особливості торрент протоколу:

1. Немає черг завантаження, тобто не потрібно всім стояти в черзі до джерела (сідеру), для того щоб скачати файл.

2. Торрент завантажується частинами – чим менше розповсюджений фрагмент файлу, тим частіше він буде віддаватися. Це значить, що наявність що роздає (сідера) не обов'язково для того, щоб скачати torrent. Протокол розподіляє фрагменти між клієнтами так, щоб надалі вони могли обмінюватися частинами без участі що роздає.

3. Клієнти працюють напряму один з одним. Торрент-трекер напряму не приймає участі в обміні.

4. Завантажені частини файлу відразу ж стають доступними для інших.

5. За допомогою контрольної суми відслідковується цілісність для кожного завантаженого шматочка.

6. Передавати можна вкладені файли, наприклад каталог з файлами усередині.

1.2 Область застосування

Областю застосування менеджера завантажень з використанням протоколу BitTorrent є мережа Інтернет.

Мережні протоколи й порти

Торрент клієнт приєднується до сервера за TCP протоколом (за замовчуванням порт 6969).

Клієнти з'єднуються між собою по TCP протокол. Порти: 6881-6889, але номера портів не є обов'язковими й можуть бути іншими. Множина torrents трекерів використовують 80 HTTP порт, а в клієнтах рекомендується виставляти опцію «випадковий вибір порту».

Також у клієнтах закладена можливість використовувати UDP порти, але вона не є офіційно затвердженою. Не всі клієнти підтримують її. UDP порти використовуються для роботи опції DHT, тобто для режиму роботи без зв'язку з torrent трекером.

Файл метаданих

Для того щоб можна було поширити інформацію про торрент, потрібно створити torrent файл, що містить метадані. Такий файл має розширення .torrent і містить наступну інформацію:

– адреса (URL) трекеру;

– інформацію про файл (атрибути, розмір і т.д.);

– контрольну суму / хеш суму фрагментів.

Для того щоб скачати torrent потрібно одержати його файл, що містить метадані. Сам він може бути переданий будь-яким доступним способом: по електронній пошті, завантажений із сайту або FTP сервера й т.д.

Торрент-трекер

1. Торрент трекер (torrent tracker) – це виділений сервер, що працює по протоколі HTTP. Він дозволяє клієнтам взаємодіяти один з одним, тобто знаходити один одного й обмінюватися інформацією, тобто які торренти в них є. Трекер містить інформацію про IP адресу, використовувані порти і хеш суми torrents файлів що роздаються. Самі файли на трекері, як правило, не присутсвуют і визначити їх з хеш суми не можна. Але часто сервер крім безпосереднього зберігання хеш суми торрентів також є Веб сайтом, на якому може втримуватися опис файлів. Також часто на ньому втримується інформація:

– опис торрент файлу;

– статистика завантажувань;

– статистика сідерів / лічерів і т.д.

Робота без торрент трекеру (torrents tracker)

У сучасній версії торрент протоколу розроблена підтримка роботи без центрального трекеру, режим DHT. У такому режимі, у випадки неприступності центрального сервера, клієнти можу продовжувати працювати підтримуючи зв'язок між собою.

Функція роботи заснована на протоколі Kademlia. При такому режимі роботи торрент трекер доступний децентралізовано на клієнтах у вигляді хеш таблиць.

У даний момент не всі клієнти здатні працювати в цьому режимі. Також протоколи роботи без трекеру реалізовані в багатьох клієнтах по-різному й тому вони можуть не працювати один з одним.

Супер-сід (Super seeding)

Супер-сід – це режим роботи, коли є всього один що роздає. Що роздає торрент клієнт віддає підключеним до нього частинам торренту й зупиняє його

віддачу іншим, доти поки ця частина не з'явитися в будь-якого іншого torrent клієнта. Тому що не всі учасники мають гарну швидкість віддачі, а в деяких заблоковані порти, то загальна швидкість поширення torrent файлу сильно знижується. При нормальній роботі кожний клієнт одержує ту частину файлу, що запитує, тому в цьому режимі середня швидкість поширення звичайно вище.

Супер-сід ефективний при участі великої кількості тих, хто завантажує. При участі декількох качающих їхні клієнти можуть бути не здатні з'єднуватися між собою, що приведе практично до повної зупинки. При кількості тих, хто завантажує, більше десяти, така ситуація менш імовірна, тому досить роздати кожному свою частину. Клієнт обмінюється своїм фрагментом з усіма іншими, і в усіх в остаточному підсумку буде торрент цілком.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Не обов'язково використовувати таку корисну технологію, як торрент, лише для поповнення своєї колекції відеофільмів. На трекерах повнісінько дійсно важливої й потрібної інформації, що поширюється по ліцензії freeware, тобто на безкоштовній основі. Звичайно, скачати її без спеціальної програми навряд чи вийде, тому даний огляд присвячений розгляду найкращих на сьогоднішній день клієнтів для роботи з торрент-трекерами.

uTorrent

Цей клієнт знаком всім і кожному – у нашій країні він найбільш популярний, тому що являє приклад майже ідеального програмного забезпечення. Він має малий розмір, але при цьому має максимальну функціональність.

Донедавна uTorrent поширювалася зовсім безкоштовно, але зараз політика розроблювачів дещо змінилася.

Відтепер є просто uTorrent, а є uTorrent Plus – останній коштує грошей і не набридає користувачеві всюдисущою рекламою.

Якщо ж вам не заважає баннер над списком файлів, то нема чого заохочувати розроблювачів рублем – їм ця реклама принесе більше прибутку.

До речі, до мінусів можна віднести зовсім непотрібні бічні панелі й бари, що вбудовуються прямо в браузер.

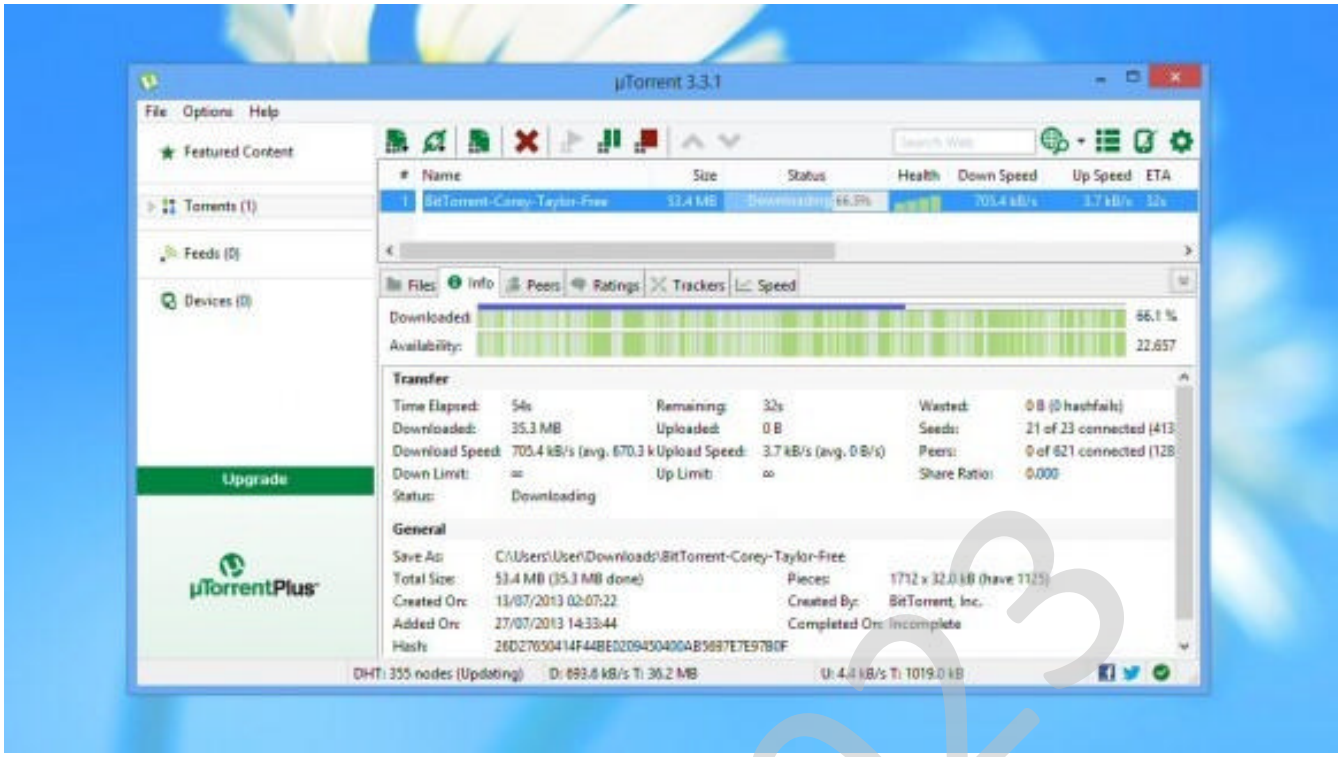

Рисунок 2.1 – Інтерфейс користувача uTorrent

qBittorrent

Це, фактично, той же uTorrent, але з невеликими відмінностями. По-перше, ніякої реклами, по-друге, відкритість вихідного коду й, по-третє, кроссплатформенність. Це значить, що та сама програма однаково добре працює й під Windows, і під MAC OS разом з Linux.

Інтерфейс qBittorrent спрощений донезмоги, так що навіть дитина розбереться. Скажемо так, таким був Utorrent пару років тому, поки його не стали перевантажувати непотрібними доповненнями, які псують все враження від програми.

В qBittorrent убудована система пошуку торрентів для завантаження, що значно скорочує час, затрачуваний на серфінг по трекерам. Дана програма для тих, кому потрібно максимум функціонала при мінімумі зайвої мішури.

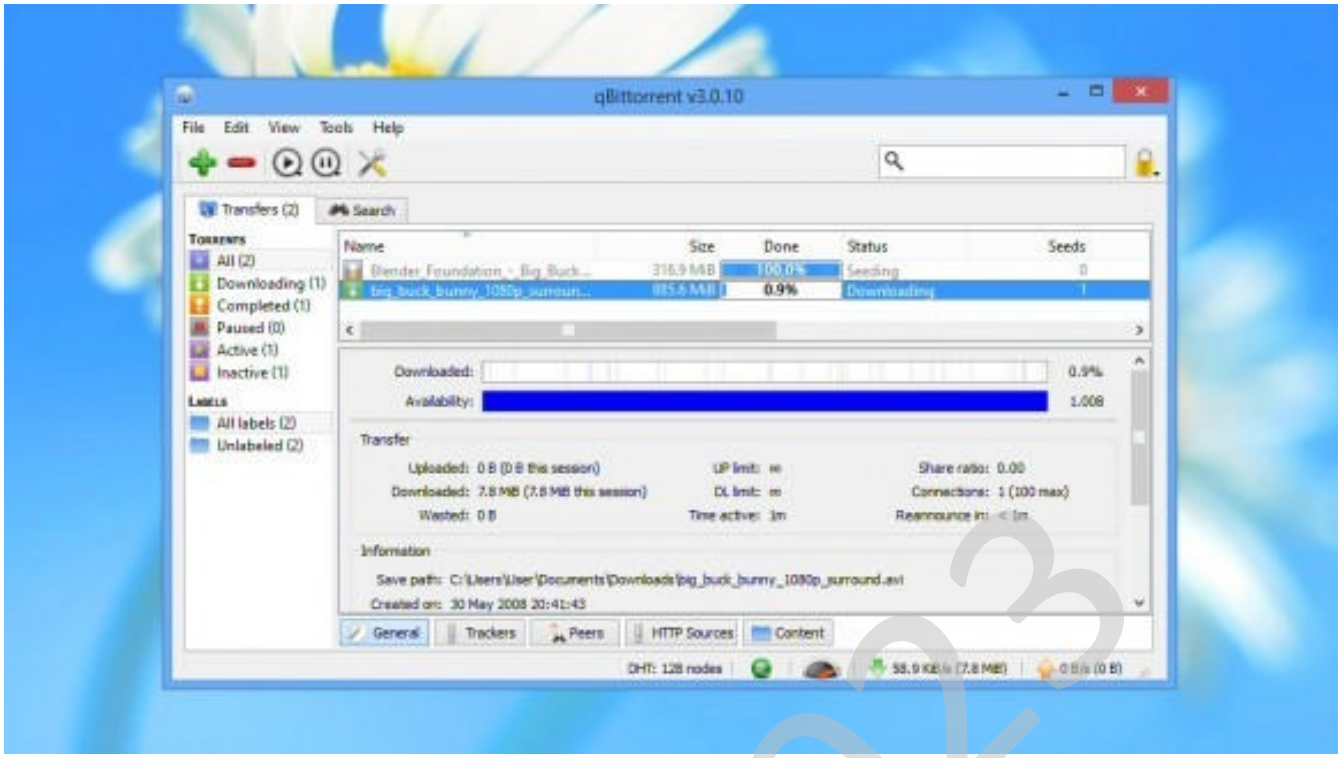

Рисунок 2.2 – Інтерфейс користувача qBittorrent

MediaGet

Даний торрент-клієнт активно просувається багатьма трекерами, що викликає певного роду недовіру до нього. Однак його не було б у нашому рейтингу, засумнівайся ми в його якості й можливостях.

Головною відмітною рисою цієї програми є наймогутніша система пошуку необхідного медіаконтенту, інтегрована прямо в неї.

Інтерфейс простим назвати не можна, але графікою він не перевантажений. Виглядає стильно – і не більше. Як альтернатива нудному й оброслій рекламі Utorrent MediaGet ідеально підійде. Це вибір тих, кому важлива візуальна краса інтерфейсу.

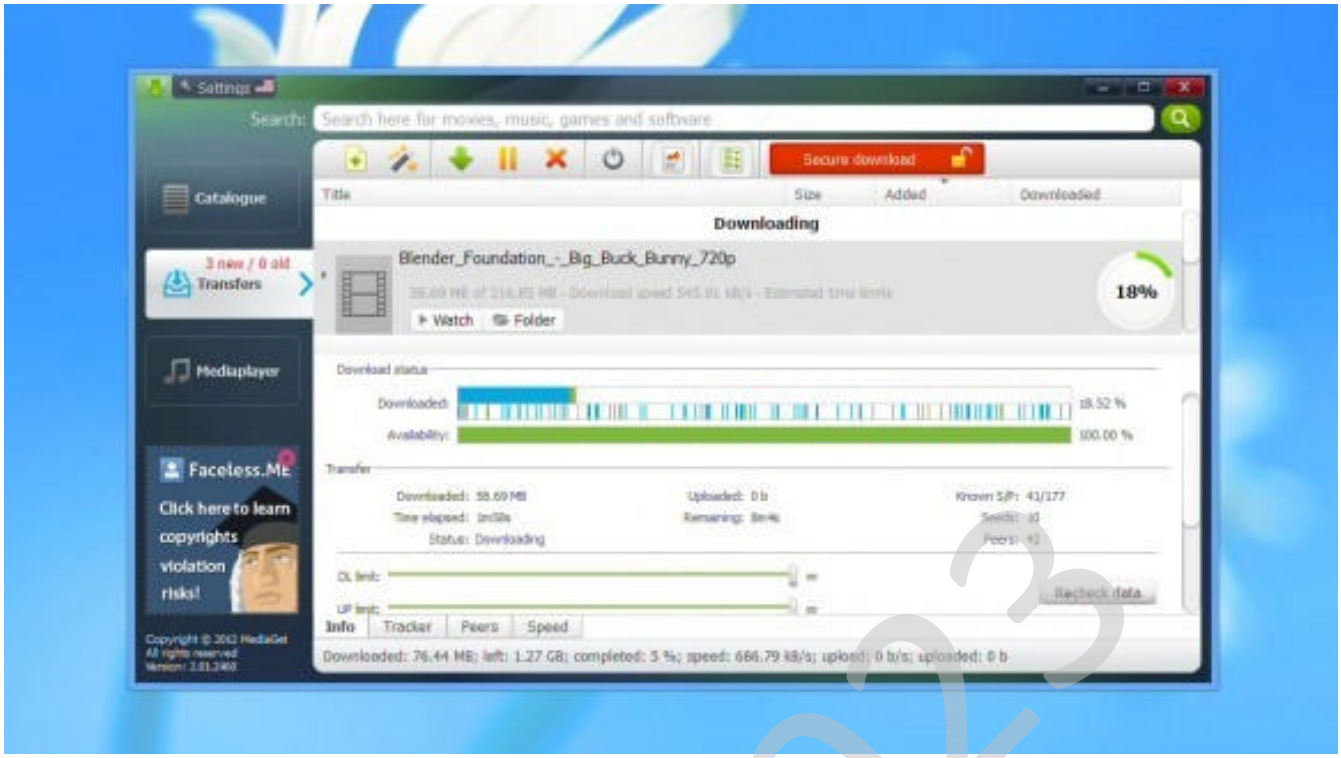

Рисунок 2.3 – Інтерфейс користувача MediaGet

BitTorrent

Це один з найперших клієнтів, що з'явилися в Мережі дуже давно, ще на зорі торрент-трекерів і безлімітного Інтернету. Він поширюється на безкоштовній основі, але при погляді на його інтерфейс виникає найсильніше почуття дежавю.

І неспроста. Насправді, BitTorrent – це все той же Utorrent, тільки ребрендинговий. Той самий вихідний код, та сама команда розроблювачів, але при цьому декілька різні інтерфейси. На цьому відмінності закінчуються. В BitTorrent немає (або, принаймні, у більше ранніх версіях не було) реклами, що не може не радувати.

До численних функцій програми відноситься можливість запуску завантажених файлів, будь те відео, фото або музика. Те ж саме є й в Utorrent, навіть у безкоштовній версії. Так що так, це два клієнти-близнюки.

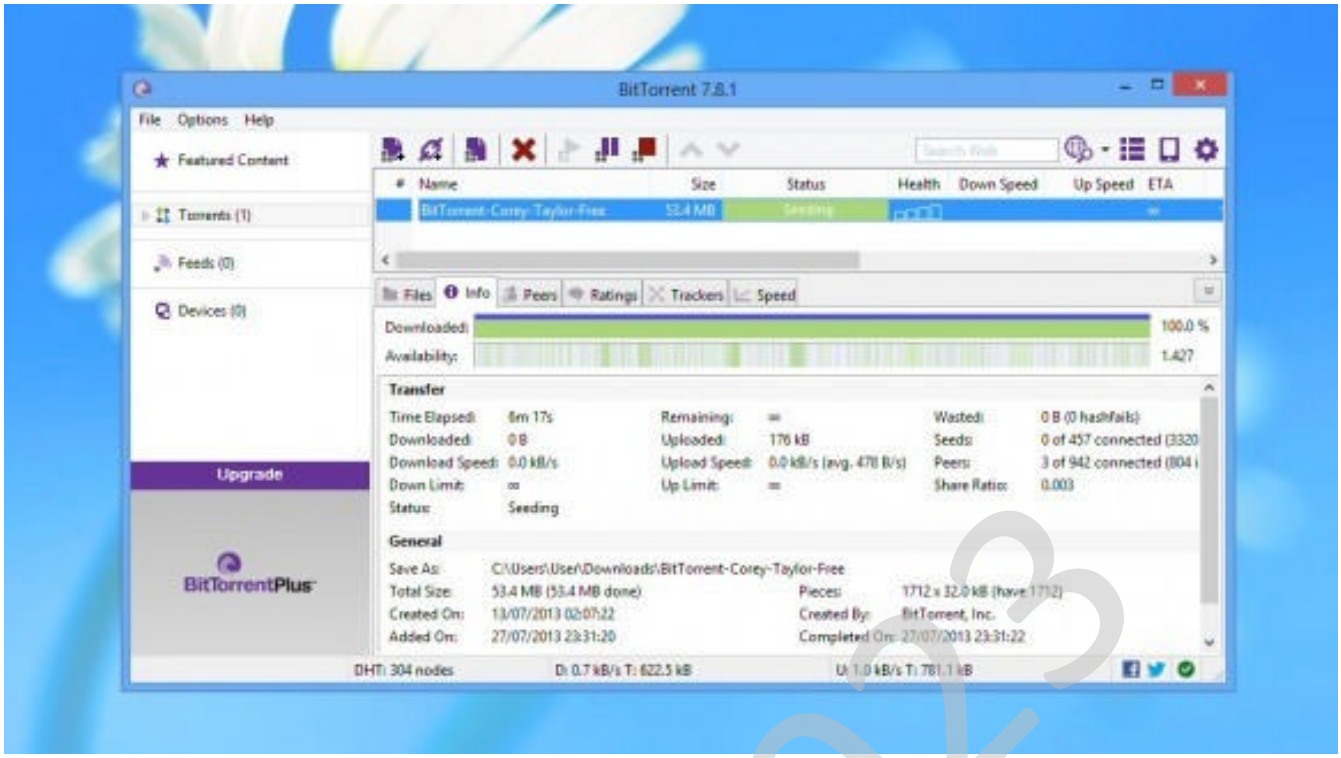

Рисунок 2.4 – Інтерфейс користувача BitTorrent

Vuze

Раніше цей клієнт називався Azureus, і це далеко не просто програма для завантаження контенту. Це як Bentley у світі торрент-клієнтів – у ньому є взагалі всі, але занадто всього багато. Наприклад, Vuze можна інтегрувати з iTunes, а множина плагінів, які він підтримує, дозволять зробити з нього мультимедійного монстра, здатного на будь-які дії з файлами.

Сам по собі Vuze безкоштовний, але якщо заплатити 30 доларів (абонентська плата на 12 місяців) те можна одержати версію Plus, у якій немає ніякої реклами, але зате є DVD-програвач і навіть убудована антивірус. Це, на наш погляд, уже невеликий перебір.

Vuze – великоваговий поїзд у світі торрент-клієнтів. Багатьом здасться, що весь набір його можливостей залишається незатребуваним у більшості випадків, однак є ті, хто все це використовує, інакше він не був би настільки популярним.

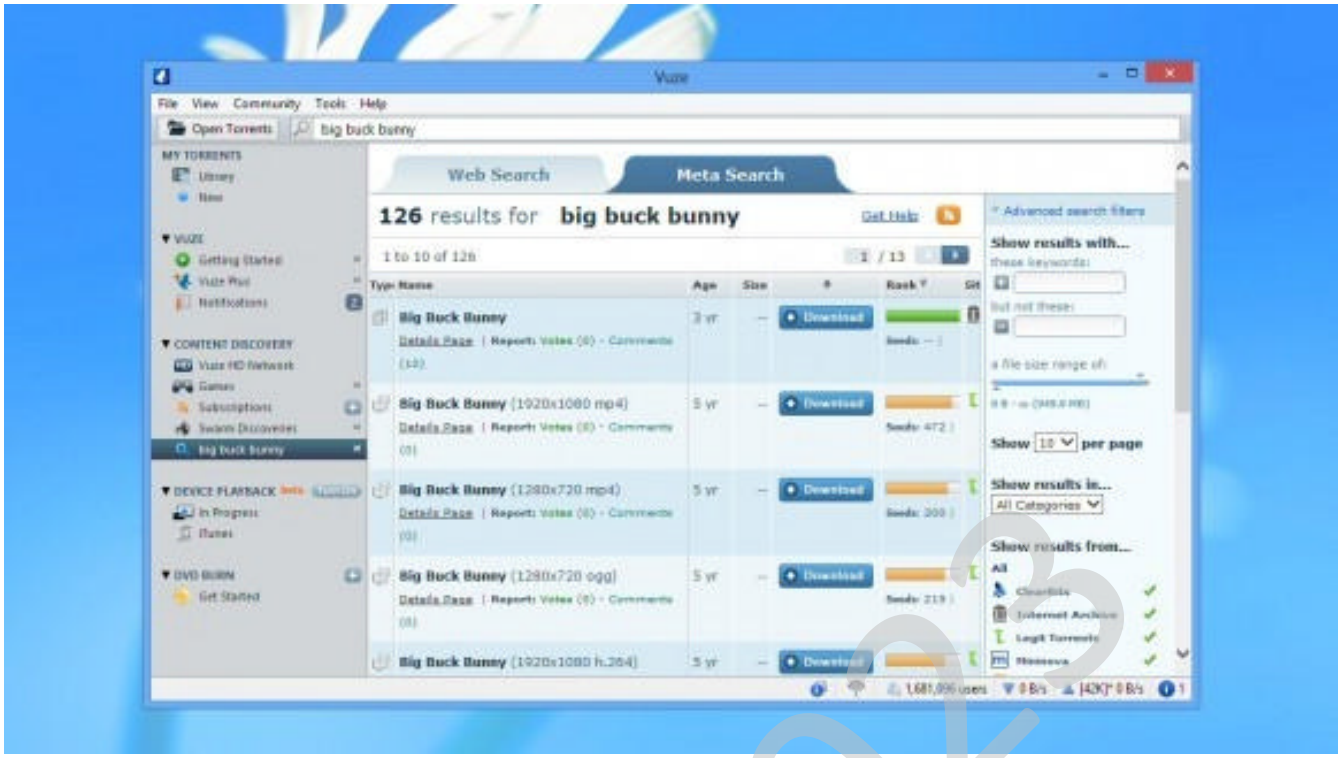

Рисунок 2.5 – Інтерфейс користувача Vuze

Opera

Ні, це не торрент-клієнт, це браузер з убудованим торрент-клієнтом. Більше того, Opera є одним з найпоширеніших веб-оглядачів, особливо в Україні, і те, що вона вміє качати з торрент-трекерів – це величезний плюс.

Ніякої реклами, ніяких зайвих функцій, все просто, зрозуміло й раціонально.

Вибрав файл на трекері, нажав «Завантажити», вибрав папку збереження – і все.

Залишиться лише небагато почекати, і потрібний файл буде скопійований з Інтернету прямо в потрібну директорію.

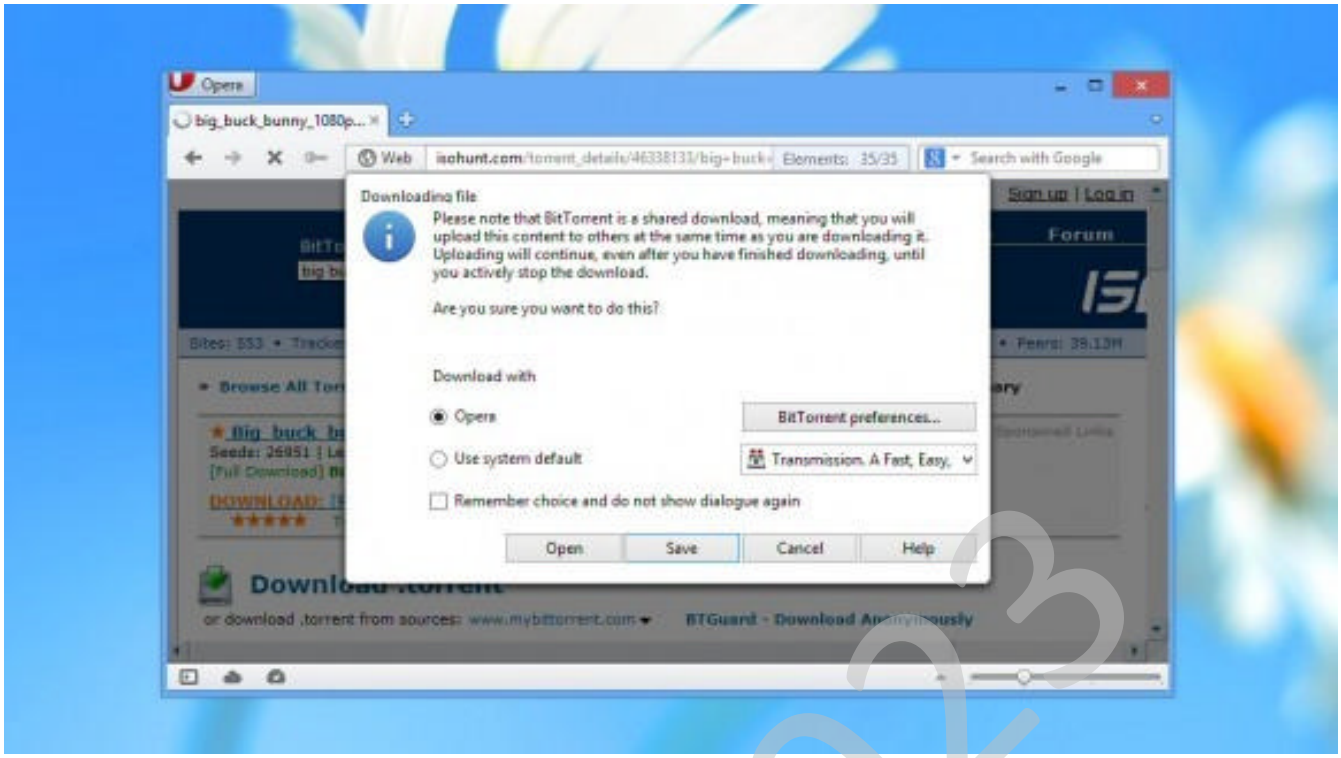

Рисунок 2.6 – Інтерфейс користувача Opera

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення

виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи менеджера завантажень з використанням протоколу BitTorrent.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Специфікація протоколу BitTorrent v 1.0

Ідентифікація

BitTorrent – це протокол для поширення файлів, заснований на принципі " крапка-крапка" і розроблений Бремом Кохеном (Bram Cohen). Відвідаєте його сторінку за адресою http://www.bittorrent.com. BitTorrent розроблений для полегшення передачі файлів безлічі пірів по ненадійних мережах.

Ціль цієї специфікації полягає в тому, щоб зародділювати у деталях специфікацію протоколу BitTorrent версії 1.0. Сторінка специфікації протоколу Брема обмежена основними поняттями, і втрачає супутні деталі в деяких областях. Я сподіваюся, що цей розділ буде написаний у ясних і однозначних визначеннях, які можуть послужити основою для обговорень і реалізації в майбутньому.

Предметна область

Цей розділ відноситься до першої версії (тобто версії 1.0) специфікації протоколу BitTorrent. На даний момент, він відбиває файлову структуру торрентів, протокол обміну даними між пірами (peer wire), і специфікації HTTP/ HTTPS-трекерів. З появою нових версій цих протоколів, вони повинні бути специфіковані на своїх окремих сторінках, але не тут.

У цьому розділі використовується ряд умовних позначок з метою представити інформацію в короткому й однозначному виді:

– Peer vs клієнт: У цьому розділі peer – це будь-який ***клієнт*** (це слово варто замінити адекватним синонімом) BitTorrent, що бере участь у завантаженні. Клієнт – це теж peer, однак це BitTorrent-клієнт, запущений на локальній машині. Читачі цієї специфікації можуть думати про нього як про клієнта, з'єднаним з деяким числом пірів.

– Шматок vs блок: У цьому розділі шматок відноситься до деякої порції завантажених даних, що описана в метафайлі, і яка може бути перевірена за допомогою хешу SHA1. Блок – це порція даних, що клієнт може запросити в піру. Два або більша кількості блоки становлять шматок, що потім може бути перевірений.

– Стандарт "де-факто": більші блоки тексту курсивом описують загальноприйняті підходи в різних реалізаціях BitTorrent-клієнтів, що стали стандартом "де-факто".

– Bencoding – це спосіб визначення й організації даних у стислому форматі. Він підтримує наступні типи: байтові рядки (byte strings), цілі числа (integers), списки (lists) і хеш-таблиці (dictionaries).

– Байтові рядки кодуються в такий спосіб: <довжина рядка, кодуєма в десяткової ASCII, у якій є тільки символи цифр>:<строкові дані>.

Треба помітити, що тут немає фіксованого початкового й кінцевого роздільника.

Приклад:

```
4:spam представляє рядок "spam"
```

Цілі числа кодуються в такий спосіб: i<ціле число, закодоване в десяткової ASCII>e

Початковий символ i і кінцевий e – початковий і кінцевий роздільники відповідно. Ви можете кодувати негативні числа, такі як i-3e. Цілому числу не може передувати префікс, що складається з нулів, як наприклад i04e. Разом з тим, i0e є є коректним записом нуля.

Приклад:

```
i3e представляє число "3".
```

Максимальна кількість біт цілого числа не специфікується, але для розгляду "більших файлів" aka .torrent для більше 4-х гігабайт необхідно знакове 64-бітне ціле.

Списки

Списки кодуються в такий спосіб:

l< bencode-закодовані значення>e

Початковий символ l і кінцевий e – початковий і кінцевий роздільники відповідно. Списки можуть містити bencode-кодовані значення будь-яких типів, включаючи цілі числа, рядки, хеш-таблиці й інші списки.

Приклад:

l4:spam4:eggse представляє список із двох рядків:

["spam", "eggs"]

Хеш-таблиці кодуються в такий спосіб:

d< bencode-кодована рядок-ключ>< bencode-кодований елемент>e

Початковий символ d і кінцевий e – початковий і кінцевий роздільники відповідно. Помітьте, що ключі повинні бути bencode-кодованими рядками. Значення хеш-таблиці можуть бути bencode-кодованими значеннями будь-якого типу, включаючи цілі числа, рядки, списки й інші хеш-таблиці. Ключі можуть бути тільки строковими й повинні фігурувати у відсортованому порядку (sorted as raw strings, not alphanumerics). Рядки повинні рівнятися з використанням бінарного порівняння, а не культурно^-специфічного "природного" порівняння.

Приклад:

d3:cow3:moo4:spam4:eggse являє собою хеш-таблицю { "cow" => "moo", "spam" => "eggs" }

Приклад:

d4:spaml1:a1:bee представляє хеш таблицю виду { "spam" => ["a", "b"] }

Приклад:

d9:publisher3:bob17: publisher-webpage15:www.example.com18:publisher.location4:homee представляє хеш-таблицю { "publisher" => "bob", " publisher-webpage" => "www.example.com", "publisher.location" => "home" }

Реалізація мовою Perl:

http://search.cpan.org/dist/ Convert-Bencode/lib/Convert/Bencode.pm

Реалізація мовою Java:

http://www.koders.com/java/fid47111A56F2466C232E09AEF75A39915EC70D3536.aspx

Структура файлу позначка-даних

Файл позначка-даних закодований в bencode-форматі, докладний опис якого наведено вище.

Уміст файлу позначка-даних (розширення файлу – ".torrent") – це bencode-кодована хеш-таблиця, що містить перераховані нижче ключі. Всі строкові величини закодовані в UTF-8:

– info: Хеш-таблиця, що описує файл(и) торренту. Є дві можливих форми: перша – для випадку з 'одне-файловим' торрентом, без опису структури директорій; друга – для 'багато-файлового' торренту;

– announce: URL трекеру для публікації торренту (рядок);

– announce-list: (опціональний) Це розширення до офіційної специфікації зі збереженням зворотної сумісності. Ключ використовується для реалізації списку резервних трекерів. Повний опис можна знайти тут – http://home.elp.rr.com/tur/ multitracker-spec.txt;

– creation date: (опціональний) Дата створення торренту, у стандартному форматі UNIX-Часу (ціле число секунд минулих з 01 січня 1970 00:00:00 по UTC);

– comment: (опціональний) Текстовий коментар у вільній формі від автора (рядок);

– created by: (опціональний) Ім'я й версія програми, що використовувалася для створення torrent-файлу (рядок).

Цей розділ описує загальні поля й для "одне-файлового", і для "багато-файлового" торренту:

– piece length: Розмір кожного шматка в байтах (ціле);

– pieces: Рядок, складений об'єднанням 20-байтових значень SHA 1-хешів кожного шматка (один шматок – один хеш) (байтовий рядок);

– private: (опціональний) Це поле є цілим числом. Якщо воно встановлено в значення "1", клієнт ЗОБОВ'ЯЗАНИЙ одержувати список пірів ТІЛЬКИ за допомогою трекерів, перерахованих у файлі позначка-даних. Якщо поле

встановлене в "0" або взагалі відсутній, клієнт може одержувати список пірів іншими способами, наприклад за допомогою PEX (обмін пірами) або DHT. Таким чином, слово "приватний" можна читати як "без зовнішніх джерел списку пірів".

Не всі згодні з таким описом. От наприклад визначення з вики клієнта Azureus: http://www.azureuswiki.com/index.php/Secure_Torrents.

Крім того, слід зазначити, що навіть якщо це поле й використовується на практиці, воно не є частиною офіційної специфікації.

Info в одне-файловому режимі

У випадку одне-файлового режиму, хеш-таблиця info доповнюється наступними ключами:

– name: Ім'я файлу, що містить торрент. Носить рекомендаційний характер. (рядок);

– length: Розмір файлу в байтах (ціле);

– md5sum: (опціональний) 32-символьна шістнадцятковий рядок відповідна MD 5-сумі файлу. Вона не використовується в BitTorrent, але записується в торрент деякими програмами для кращої сумісності.

Info у багато-файловому режимі

У випадку багато-файлового режиму, хеш-таблиця info доповнюється наступними ключами:

– name: Ім'я кореневої директорії, що містить торрент. Носить рекомендаційний характер. (рядок);

– files: Список з хеш-таблиць, по однієї на кожний файл. Кожна хеш-таблиці в цьому списку містить наступні ключі:

– length: Розмір файлу в байтах (ціле);

– md5sum: (опціональний) 32-символьна шістнадцятковий рядок відповідна MD 5-сумі файлу. Вона не використовується в BitTorrent, але записується в торрент деякими програмами для кращої сумісності;

– path: Список, що містить один або більше строкових елементів, об'єднання яких дає шлях і ім'я файлу. Кожний елемент у списку відповідає або ім'я директорії, або (у випадку з останньому елементом) – ім'я файлу. Наприклад,

файл "dir1/dir2/file.ext" повинен складатися із трьох строкових елементів: "dir1", "dir2" і "file.ext". Він кодується як список з рядків в bencode-форматі от так:

```
l4:dir14:dir28:file.exte
```

Ключ 'piece length' установлює номінальний розмір шматка, що, як правило, кратний 2-м. Розмір шматка звичайно вибирається виходячи із загальної кількості файлових даних у торренті, зважаючи на те, що занадто великий розмір шматка неефективний, а занадто малий дає в результаті великий торрент файл. При виборі найменшого розміру шматка керуйтеся здоровим глуздом. Бажано не робити торрент-файл розміром більше, ніж 50-75КБ (щоб полегшити його завантаження на сервер). Однак, зараз, коли розміри хостинга й ширина каналів не сильно обмежуються, кращий розмір шматка для більше ефективної роздачі файлів дорівнює 512КБ або менше (принаймні для торрентів приблизно до 8-10ГБ), навіть якщо це приведе до збільшення торрент-файлу. Часто використовувані розміри – 256КБ, 512КБ і 1МБ. Кожний шматок дорівнює обраному розміру, за винятком останнього, розмір якого може бути менше. Кількість шматків обчислюється розподілом загального розміру файлів торренту на розмір шматка й округляється в більшу сторону. Для обчислення границь шматків у багато-файловому торренті, файлові дані розглядаються як один великий безперервний потік, складений об'єднанням умісту всіх файлів у порядку їхнього проходження в списку 'files'. Число шматків і їхніх границь визначаються в такий же спосіб, як і у випадку одного файлу. Шматки можуть перекривати границі файлів (тобто початок шматка може перебуває наприкінці попереднього файлу, а кінець – на початку наступні).

Кожний шматок має відповідний йому SHA 1-хеш. Для формування значення ключа 'pieces' (див. опис хеш-таблиці 'info' вище) хеши всіх шматків об’єднуються в один рядок (зверніть увагу – у рядок, а не в список). Її довжина повинна бути кратна 20-ти.

Протокол трекеру (HTTP/HTTPS)

Трекер – це HTTP/HTTPS сервіс, що відповідає на HTTP GET запити. Запити містять у себе метрику від клієнтів, що допомагає трекеру вести статистику для торренту. У відповідях утримується список пірів, щоб клієнт міг брати участь у роздачі. Базовий URL запиту складається з 'announce URL', що визначений у файлі позначка-даних (торрент-файл), і параметрів, які додаються до цьому URL за допомогою стандартного CGI-Методу (т.е. додавання '?' після announce-URL, за яким треба послідовності 'параметр=значення', розділені символом '&').

Зверніть увагу, що всі бінарні дані в URL (особливо info_hash і peer_id) повинні бути правильно екрановані. Це означає, що будь-який байт, що не входить у безлічі " 0-9", " a-z", " A-Z" і "$-_.+!*'()," повинен бути закодований у форматі "%nn", де nn – шістнадцяткове значення байта. (см. RFC 1738 для подробиць).

Для наступного 20-байтового хешу:

\x12\x34\x56\x78\x9a\xbc\xde\xf1\x23\x45\x67\x89\xab\xcd\xef\x12\x34\x56\x78\x9a

правильно закодованої є рядок

%124Vx%9A%BC%DE%F1%23Eg%89%AB%CD%EF%124Vx%9A

Параметри запиту до трекеру

Далі треба опис параметрів, використовуваних в GET запиті від клієнт до трекеру:

– info_hash: 20-байтовий SHA 1-хеш від значення ключа 'info' файлу позначка-даних, що є хеш-таблицею в bencode-форматі. Опис 'info' було дано раніше;

– peer_id: 20-байтовий рядок, що використовується як унікальний ідентифікатор клієнта, згенерований ним же при запуску. Значення може бути кожним, у тому числі й бінарним. На даний момент немає ніяких рекомендацій з генерації цього ідентифікатора. Однак, справедливо припустити, що він повинен

бути унікальним для локальної машини. Таким чином, імовірно, варто включати в нього таку інформацію, як ідентифікатор процесу й, можливо, тимчасову мітку, записану їм при запуску. Способи кодування цього поля основними клієнтами опис нижче в розділі peer_id;

– port: Номер порту, що прослуховує клієнт. Стандартні порти, які зарезервовані для BitTorrent – 6881-6889. Клієнт може використовувати будь-який інший порт, якщо він не може відкрити його в зазначеному діапазоні;

– uploaded: Сумарна кількість відданих даних (після того, як клієнт послав подію 'started' трекеру) записане десятковим числом. Поки це точно не визначено в офіційній специфікації, уважається, що тут повинне бути загальне число відданих байт;

– downloaded: Сумарна кількість завантажених даних (після того, як клієнт послав подію 'started' трекеру) записане десятковим числом. Поки це точно не визначено в офіційній специфікації, уважається, що тут повинне бути загальне число завантажених байт;

– left: Число байт десятковим числом, що клієнт ще повинен скачати;

– compact: Устанавленное значення "1" сигналізує, що клиет може приймати компактні відповіді. Список пірів заміняється рядком – по 6 байт на одного піру. Перші чотири байти – це хост (у мережному порядку байтів), останні два байти – порт (знову ж, у мережному порядку байтів). Варто пам'ятати, що деякі трекери підтримують тільки компактні відповіді (для економії трафіку) і ігнорують запити без параметра "compact=1" або просто посилають компактну відповідь, навіть при "compact=0";

– no_peer_id: Говорить про те, що трекер може зневажити полем 'peer id' у хеш-таблиці 'peers'. Цей параметр ігнорується, якщо включено компактний режим;

– event: Значенням може бути 'started', 'completed', 'stopped', або порожнє, що рівнозначно невизначеному. Якщо параметр не визначений, значить цей запит виконується через регулярні інтервали часу. Докладніше про значення:

– started: Перший запит до трекеру обов'язково повинен бути з параметром "event=started";

– stopped: Повинен бути посланий трекеру, якщо клієнт правильно завершує роботу;

– completed: Повинен бути посланий трекеру при завершенні завантажування. Однак ця подія не повинне посилати, якщо при запуску клієнта завантажування вже на 100% завершена. Очевидно, це потрібно для того, щоб дати можливість трекеру правильно збільшувати показник завершених завантажувань, що залежить від цієї події;

– ip: (опціональний) Реальний IP-адреса клієнтської машини, формат адреси – чотири байти (десятковими числами) розділених крапками або шістнадцяткова IPv6-адреса (RFC 3513). Взагалі, цей параметр не є необхідним, тому що адреса клієнта може бути взятий з IP-адреси, з якого відправлений запит. Параметр потрібний тільки у випадку, коли IP-адреса, з якого прийшов запит, не є IP-адресою клієнта. Це відбувається, коли клієнт з'єднується із трекером через проксі-сервер. А також, це необхідно, коли клієнт і трекер перебувають в одній локальній частині NAT-шлюзу, т.к. інакше трекер буде видавати внутрішній (RFC 1918) адреса клієнта, що не є маршрутизуємим. Тому, клієнт повинен однозначно встановити IP-адресу (зовнішній, маршрутизуємий), для видачі його зовнішнім пірам. Різні трекери обробляють цей параметр по-різному. Деякі приймають його, якщо IP-адреса, з якого прийшов запит, перебуває в діапазоні RFC 1918, інші – приймають беззастережно, треті – повністю його ігнорують. Якщо передано IPv 6-адру (наприклад, 2001:db8:1:2::100), значить клієнт може спілкуватися тільки по протоколі IPv6;

– numwant: (опціональний) Кількість пірівів, що клієнт хоче одержати від трекеру. Значення може бути нулем. Якщо параметр не заданий, по-умовчанню, звичайно віддається 50 пірів;

– key: (опціональний) Додаткова ідентифікація, що не доступна іншим користувачам. Призначена для того, щоб клієнт міг підтвердити свою дійсність при зміні IP-адреси;

– trackerid: (опціональний) Якщо попередня відповідь містила значення 'tracker id', це значення потрібно вписати сюди.

Трекер відповідає текстом (text/plain), що містить у собі хеш-таблицю в bencode-форматі з наступними ключами:

– failure reason: Якщо є присутнім, то є єдиним ключем у хеш-таблиці. Значення ключа – це текстове повідомлення про помилку, що повідомляє про те, чому запит не вдався (рядок);

– warning message: (новий, опціональний) Схожий на 'failure reason', але відповідь буде повним. Попередження відображається також, як і помилка;

– interval: Інтервал у секундах, що клієнт повинен витримувати між посилкою регулярних запитів трекеру;

– min interval: Мінімальний інтервал для оповіщень. Якщо задано, клієнт не повинен робити оповіщення частіше, ніж це значення;

– tracker id: Рядок, що клієнт повинен посилати назад у наступних оповіщеннях. Якщо попереднє оповіщення містило 'tracker id', а в поточній відповіді ключ відсутній, використовуйте старе значення;

– complete: Число пірів, що мають всі файли торренту. Їх називають сідерами (ціле)

– incomplete: Число пірів, що не має всі файли торренту. Їх називають личерами (ціле)

– peers: (модель на хеш-таблицях) Значенням є список, що складається з хеш-таблиць, кожна з яких містить наступні ключі:

– peer id: Ідентифікатор піру для запитів трекеру, що він сам собі й вибрав. Був описаний раніше (рядок);

– ip: IP-адреса піру у форматі IPv6 або IPv4, або DNS-Ім'я (рядок);

– peers: (бінарна модель) Замість використання хеш-таблиць, значенням кожного елемента списку може бути рядок, що складається з 6 байт. Перші 4 байти – це IP-адреса; останні 2 байти – порт. Всі байти записуються в мережному порядку байтів (big endian нотація).

Як згадувалося раніше, список пірів, по-умовчанню, має 50 записів. Якщо торрент має невелика кількість пірів, список буде менше. У противному випадку, трекер вибирає піри для списку випадковим образом. Для здійснення вибірки пірів для списку, трекер може використовувати більше інтелектуальний алгоритм. Наприклад, не повідомляти про наявних на роздачі сідерах іншим сідерам.

Клієнти можуть посилати запити трекеру частіше, ніж через зазначений інтервал: якщо відбулася яка-небудь подія (наприклад, зупинка (stopped) або завершення (completed) завантажування), або клієнт хоче одержати ще один список пірів. Проте, постійне опитування трекеру (в оригіналі, hammer – бити, наносити удари) для одержання списків пірів – це погано. Якщо клієнт хоче одержати список більшого розміру, йому варто використовувати в запиті параметр 'numwant'.

Примітка розроблювача: Навіть 30 пірів досить. Насправді, офіційний клієнт 3-їй версії створює нові з'єднання, тільки якщо має менш 30 пірів, і відмовляє в з'єднанні, при більш ніж 55 пірах. Це значення має велике значення для продуктивності. Коли новий шматок повністю отриманий, більшості активних пірів повинне бути послане HAVE-повідомлення. У результаті, кількість трафіку збільшується пропорційно кількості пірів. Якщо їх більше 25-ти, досить малоймовірно, що нові піри піднімуть швидкість завантаження. Розроблювачам клієнтів настійно рекомендується зробити так, щоб цей параметр був непомітний і складний для зміни, тому що він буде корисний у рідких випадках.

Метод scrape

Scrape – збирати, скребти, зскрібати.

Більшість трекерів підтримують іншу форму запиту, що використовується для одержання інформації з певного торренту (або всіх торрентів), якими управляє трекер. Замість незручного парсингу сторінки зі статистикою, клієнт може відправити такий запит, і трекер відповість так званою scrape-сторінкою.

Для запиту scrape-сторінки клієнт використовує HTTP GET метод, як у стандартного запиту описаного раннє, але по іншому URL'у. Щоб одержати scrape-url, потрібно проробити наступне. Шукаємо в announce-url останній символ '/' (слеш). Якщо текст безпосередньо наступний за '/' не 'announce', це ознака того, що трекер не підтримує scrape. У противному випадку, заміняємо 'announce' на 'scrape'.

Приклади: (announce-url -> scrape-url)

~http://example.com/announce -> ~http://example.com/scrape
~http://example.com/x/announce -> ~http://example.com/x/scrape
~http://example.com/announce.php -> ~http://example.com/scrape.php
~http://example.com/a -> (scrape не підтримується)
~http://example.com/announce?x2%0644 -> ~http://example.com/scrape?x2%0644
~http://example.com/announce?x=2/4 -> (scrape не підтримується)
~http://example.com/x%064announce -> (scrape не підтримується)

Цей стандарт зароздільован Bram'ом у списку розсилання BitTorrent development: http://groups.yahoo.com/group/BitTorrent/message/3275

Scrape-url може бути доповнений опціональним параметром 'info_hash' з 20-байтовим значенням Це обмежить відповідь трекеру scrape-сторінкою, що буде містити інформацію тільки про запитуваний торренті. У противному випадку, статистика віддається по всім торрентам, якими управляє трекер. Якщо це можливо, для зменшення навантаження на трекер і канал, використання параметра 'info_hash' строго рекомендується.

Також, можна вказати кілька параметрів 'info_hash', якщо трекер це підтримує. Поки це не є частиною офіційної специфікації, хоча вже стало стандартом де-факто. Приклад:

http://example.com/scrape.php?info_hash=aaaaaaaaaaaaaaaaaaaa&info_hash=bbbbbbbbbbbbbbbbbbbb&info_hash=cccccccccccccccccccc

На scrape-запит трекер відповідає текстовим розділом (text/plain), іноді – стислим по методу gzip, що містить у собі хеш-таблицю в bencode-форматі з наступними ключами:

– files: Хеш-таблиця, що містить одну пару ключ/значення для кожного торренту, по якому є статистика. Якщо задано валідний параметр 'info_hash', то

таблиця містить одну пару ключ/значення. Кожний ключ – це 20-байтове бінарне значення 'info_hash'. Значення ключа – це ще одна хеш-таблиця:

– complete: Число пірів, що мають всі файли торренту (сідери) (ціле);

– downloaded: Загальна кількість завершених завантажувань, зареєстрованих трекером (реєструються при події 'event=complete', тобто коли клієнт закінчила завантаження торренту);

– incomplete: Число пірів, що не має всі файли торренту (личери) (ціле);

– name: (опціональний) Внутрішнє ім'я торренту, зазначене в ключі 'name' роздягнула 'info' торрент-файлу.

Зверніть увагу, що ця відповідь має три рівні вкладених хеш-таблиць. От Приклад:

```
d5:filesd20:....................d8:completei5e10:downloadedi50e10:incomplet
ei10eeee
```

Де ..... – це 20-байтове значення параметра 'info_hash' для торренту, зі статистикою: 5 сідеров, 10 лічерів і 50 завершених завантажувань.

Неофіційні розширення до scrape

Нижче описані ключі, які можуть використовуватися у відповіді, але їх немає в офіційній специфікації. Тому, поки вони є опціональними:

– failure reason: Текстове повідомлення про помилку, що повідомляє про те, чому запит не вдався (рядок). Клієнти, що обробляють цей ключ: Azureus;

– flags: Хеш-таблиця, що містить різноманітні прапори. Значення прапорів – це ще одна вкладена хеш-таблиця, що може містити:

– min_request_interval: Значення цього ключа – ціле число, що визначає, скільки секунд клієнт повинен чекати перед відправленням наступного scrape-запиту трекеру. Трекери, що посилають цей ключ: BNBT. Клієнти, які його обробляють: Azureus.

Протокол зв'язку між пірами (TCP)

Огляд

Протокол зв'язку між пірами (далі, просто peer-протокол) полегшує обмін шматками (pieces), перерахованих у торрент-файлі.

Зверніть увагу, що при описі peer-протоколу в оригінальній специфікації використовується термін "шматок", однак це не той "шматок", що використовується при описі торрент-файлу. Тому, у цій специфікації буде використовуватися термін "блок" для позначення даних, якими обмінюються піри по мережі.

Клієнт повинен підтримувати інформацію про стан кожного з'єднання з вилученим піром:

– choked: чи Блокує (від англ. choke – душити, віджимати) вилучений peer цього чи клієнта ні. Якщо peer блокує клієнта, це означає, що peer не буде відповідати на будь-який запит клієнта доти, поки не розблокує його. Клієнтові не слід намагатися запитувати блоки, тому що всі ці запити будуть зігноровані;

– interested: чи Зацікавлений вилучений peer у чомусь, що може запропонувати клієнт. Це означає, що вилучений peer почне запитувати блоки, коли клієнт розблокує його.

Зверніть увагу, що сам клієнт теж стежить і за тим, чи зацікавлений він у пірі (interested), а також, чи заблокований peer чи клієнтом ні (choked/unchoked). Тому, реальний список виглядає приблизно так:

```
am_choking: Клієнт блокує піру
am_interested: Клієнт зацікавлений у пірі
peer_choking: Peer блокує клієнта
peer_interested: Peer зацікавлений у клієнті
```

Клієнт починає з’єднання як "заблокований" і "не зацікавлений". Іншими словами:

```
am_choking = 1
am_interested = 0
peer_choking = 1
peer_interested = 0
```

Блок завантажується клієнтом тоді, коли він зацікавлений у пірі, а peer, у свою чергу, не блокує клієнта. Блок віддається клієнтом тоді, коли він не блокує пір, і peer зацікавлений у клієнті.

Для клієнта важливо інформувати піри про те, чи зацікавлений він у них чи ні. Інформацію про цьому варто вчасно обновляти для кожного піру, навіть

якщо клієнт ним заблокований. Це дозволяє пірам знати, чи почне клієнт завантаження, коли він його розблокує (і навпаки).

Типи даних

Якщо не зазначений інший спосіб, всі цілі числа в peer-протоколі кодуються як чотирьохбайтові значення в big-endian форматі. У тому числі й префіксний розмір всіх повідомлень, які приходять після установки зв'язку.

Потік повідомлень

Peer-протокол складається з початкової установки зв'язку і наступного обміну повідомленнями із префіксним розміром

"Рукостискання" (handshake)

"Рукостискання" – це обов'язкове й перше в потоці повідомлення, що повинен передати клієнт. Його розмір – це 49 байт + довжина pstr.

```
handshake: <pstrlen><pstr><reserved><info_hash><peer_id>
```

– pstrlen: Довжина рядка <pstr> (один байт);

– pstr: Строковий ідентифікатор протоколу;

– reserved: Вісім зарезервованих байт. Всі поточні реалізації заповнюють їхніми нулями. Кожний біт у цих байтах може використовуватися для зміни режиму роботи протоколу. У своєму email Брем пропонує використовувати молодші біти, щоб старші біти можна було використовувати для зміни значення молодших;

– info_hash: 20-байтовий SHA 1-хеш ключа 'info' торрент-файлу. Це той же 'info_hash', що передається в запитах трекеру;

– peer_id: 20-байтовий рядок, використовується як унікальний ідентифікатор клієнта. Це той же 'peer_id', що передається в запитах трекеру (правда не завжди, наприклад Azareus не передає при включеній опції анонімності).

Для протоколу BitTorrent v1.0 дані такі:

```
pstrlen = 19 і pstr = "BitTorrent protocol".
```

Ініціатор з'єднання негайно посилає handshake-повідомлення. Адресат може відкласти відповідь ініціаторові, якщо він обслуговує трохи торрентів

одночасно (торренти однозначно ідентифікуються по їх 'info_hash'). Незважаючи на це, адресат повинен відповісти відразу, як тільки одержить значення поля 'info_hash' у handshake-повідомленні. Трекерна функція NAT-перевірки не посилає поле 'peer_id' при рукостисканні.

Клієнт повинен обірвати з'єднання, якщо одержав handshake-повідомлення з 'info_hash' торренту, що він не обслуговує.

Якщо ініціатор з'єднання одержує handshake-повідомлення, у якому 'peer_id' не збігається з очікуваним 'peer_id', то ініціатор закриває з'єднання. Зверніть увагу, що ініціатор, очевидно, одержує інформацію про peer від трекеру, що містить у собі 'peer_id' цього піру. Тому, peer_id від трекеру й peer_id при рукостисканні повинні збігатися.

Ідентифікатор піру (peer_id)

peer_id повинен бути довжиною в 20 байт.

Є дві основних угоди кодування інформації про клієнта і його версію в peer_id: Azareus-стиль і Shadow-стиль.

В Azareus-стилі використовується наступне кодування:

'-'; два символи для ідентифікатора клієнта; чотири ASCII цифри для номера версії; '-'; випадкові числа.

Наприклад:

'-AZ2060-'...

Відомі клієнти, які використовують цей стиль кодування:

Далі йде список ID для відомих клієнтів

Клієнта, які зустрічаються в природі, але поки не ідентифіковані:

'BD' (Приклад: -BD0300-) 'NP' (Приклад: -NP0201-) 'w' (Приклад: -w2200-)

'hk' (example: -hk0010-) Chinese IP address, unrequestedly sends info dict in message 0x, reconnects immediately after being disconnected, reserved bytes = 01,01,01,01,00,00,02,01

В Shadow-стилі використовується наступне кодування: один альфа-цифровий ASCII символ, що ідентифікує клієнта; до п'яти символів для номера версії (розбивається за допомогою '-', якщо більше п'яти); три символи (звичайно '---', але не завжди); випадкові символи. Кожний символ у рядку з версією клієнта

позначає число від 0 до 63: '0'=0, ..., '9'=9, 'A'=10, ..., 'Z'=35, 'a'=36, ..., 'z'=61, '.'=62, '-'=63.

Повний опис Shadow-стилю, включаючи інформацію про існуючі угоди, як кодувати версію трьома символами, можна знайти тут.

Наприклад:

'S58 B-B-----'... для клієнта Shadow's 5.8.11

Відомі клієнти, які використовують цей стиль кодування:

```
'A' - ABC
'O' - Osprey Permaseed
'Q' - BTQueue
'R' - Tribler
'S' - Shadow's client
'T' - BitTornado
'U' - UPn NAT Bit Torrent
```

Клієнт Bram'a зараз використовує такий стиль... 'M 3-4-2 -' або 'M 4-20-8-'.

BitComet робить щось інше. Його peer_id складається із чотирьох символів ASCII "exbc", за яким ідуть два байти X і Y, а потім випадкові символи. X – десятковий (in decimal) номер версії до коми, а Y – відповідає за два десяткових знаки після коми. BitLord використовує ту ж схему, але додає, "LORD" після байтів версії. Неофіційний патч для BitComet перемінив "exbc" на "FUTB". Кодування ідентифікаторів піру в BitComet було наведено до стилю Azureus, як в BitComet версії 0,59.

XBT Client також має свій власний стиль. Його peer_id складається із трьох заголовних символів "XBT" і потім випливають три ASCII цифри, що представляють номер версії. Якщо клієнт є відлагоджувальним складаням, сьомий байт – символ "d", у противному випадку це '-'.

Після цього треба '-', а потім випадкові цифри, заголовні й малі літери. Приклад:

"XBT054 d-d-" на початку буде означати відлагоджувальне складаня версії 0.5.4.|

Opera 8 previews і Opera 9.x releases використовують наступну схему peer_id: Перші два знаки "OP", а далі чотири цифри означають номер складаня. Всі наступні символи – випадкові шістнадцяткові цифри в нижньому регістрі.

MLdonkey використовує наступну peer_id схему: перші символи – це "-ML", далі розділений крапкою номер версії, потім "-", і далі послідовність випадкових символів. Наприклад:

```
'-ML2.7. 2-kgjjfkd "
```

Bits on Wheels використовує модель '- BOWxxx-yyyyyyyyyyyy', де y – випадковий символ (заголовна буква), а x залежить від версії. Версія 1.0.6 має XXX = A0C.

Queen Bee використовує новий стиль Брама: "Q 1-0-0 – 'або' Q 1-10-0-", а далі послідовність випадкових байт.

BitTyrant є Azureus віткою, і просто використовує "AZ2500BT '+ випадкові байти, як peer ID в 1.1 версії. Помітьте: відсутнє тире.

TorrenTopia версія 1.90 претендує бути або є похідна від Mainline 3.4.6. Його peer ID починається з '346 ---i ---i ---i ".

BitSpirit має кілька режимів peer ID. В одному режимі він читає ID пірів і переконнектується, використовуючи перші вісім байт як основу для своїх власних ID. Його реальний ID відображається з використанням '\ 0 \ 3BS "(З нотації), як перші чотири байти для версії 3.x и' \ 0 \ 2BS" – для версії 2.x. У всіх режимах ID кінці може закінчуватися як "UDP0".

Rufus використовує свою версію у вигляді десяткових ASCII значень для перших двох байт. Третій і четвертий байти – "RS". Потім випливають nickname користувача й деякі випадкові байти.

В G3 Torrent ID починається з '-G3' і додається до 9 символів nickname користувача.

FlashGet використовує Azureus стиль із "PG", але без замикаючого символу '-'. Версія 1.82.1002 – як і раніше використовує цифри версії 1.82: "0180".

BT Next Evolution походить від BitTornado, але намагається імітувати стиль Azureus. Результатом є те, що його peer ID починається з '-cB ", як і раніше

з 4-значним номером версії, а потім триває трьома символами, які описують тип клієнта в стилі Shad0w peer ID.

AllPeers приймає SHA1 хеш залежний від користувача й заміняє перші кілька знаків на "ZC" + рядок версії + "-".

Багато клієнтів використовують всі випадкові числа, або 12 нулів після випадкових чисел (наприклад, старі версії клієнта Bram'a).

Повідомлення

Всі інші повідомлення в протоколі приймають форму <length prefix><message ID><payload>. Довжина префікса складається із чотирьох байт big-endian значення. Ідентифікатор повідомлення – це один десятковий символ. Корисне навантаження (payload) безпосередньо залежить від повідомлення.

```
keep-alive: <len=0000>
```

keep-alive повідомлення – це повідомлення з нульовими байтами, length prefix установлений у нуль. Не існує ідентифікатора повідомлення й ніякого корисного навантаження повідомлення не несе. Peer може закрити з'єднання, якщо він не одержують ніяких повідомлень (keep-alive або будь-якого іншого повідомлення) протягом певного періоду часу, тому keep-alive повідомлення націлене на підтримку зв'язку. Це час, звичайно дорівнює двом хвилинам.

```
choke: <len=0001><id=0>
```

Choke-повідомлення – це повідомлення фіксованої довжини без корисного навантаження.

```
unchoke: <len=0001><id=1>
```

Unchoke-повідомлення – це повідомлення фіксованої довжини без корисного навантаження.

```
interested: <len=0001><id=2>
```

Interested-повідомлення – це повідомлення фіксованої довжини без корисного навантаження.

```
not interested: <len=0001><id=3>
```

Non interested-повідомлення – це повідомлення фіксованої довжини без корисного навантаження.

```
have: <len=0005><id=4><piece index>
```

Have-повідомлення фіксованої довжини. Корисне навантаження – це з вказанням нулів (zero-based) індекс шматка, що тільки що були успішно завантажений і перевірені за допомогою хешу.

Конструкторське зауваження: Це строге визначення, у реальності some games may be played. Зокрема, оскільки вкрай малоймовірно, щоб піри завантажували шматки, які вони вже мають, peer може не рекламувати (advertise) наявність шматків пірам, які ці шматки мають. Придушення HAVE-повідомлень ("HAVE supression") як мінімум приведе до 50% скороченню числа повідомлень, а це скорочення приблизно на 25-35% накладних витрат протоколу (protocol overhead). У той же час, можливо доцільно відправити HAVE-повідомлення пірам, які вже мають цей шматок, оскільки він буде корисний у визначенні його рідкості.

Шкідливі піри також можуть вибирати оголошення (advertise) наявних шматків, які peer точно ніколи не завантажить. Due to this attempting to model peers using this information is a bad idea

```
bitfield: <len=0001+X><id=5><bitfield>
```

Bitfield повідомлення може бути спрямоване відразу ж після того, послідовність "рукостискань" буде завершена, і до будь-яких інших повідомлень. Воно є необов'язковим, і клієнтам, що не має шматки, немає потреби відсилати його.

Bitfield повідомлення змінної довжини, де X – це довжина bitfield'a. Корисне навантаження повідомлення – bitfield поданя шматків, які були успішно завантажені. Старший розряд у першому байті відповідає шматку з індексом 0. Біти, які порожні вказують зниклий шматок, а встановлені біти позначають валідні й доступні шматки. Запасні біти наприкінці встановлюються в нуль.

Bitfield невірної довжини вважається помилковим. Клієнти повинні розірвати з'єднання, якщо вони одержують bitfields невірного розміру, або якщо bitfield має довільний набір запасних біт.

```
request: <len=0013><id=6><index><begin><length>
```

Повідомлення-Запит фіксованої довжини, використовується для запити блоку. Корисне навантаження повідомлення містить наступну інформацію:

index: ціле число, що визначає із вказівкою нулів (zero-based) індекс шматка

begin: ціле із вказівкою нулів зсув байтів усередині шматка

length: ціле число, що визначає запитувану довжину.

This section is under dispute! Please use the discussion page to resolve this!

View 1. Відповідно до офіційних специфікацій, "Всі поточних реалізацій використовують 2^15 (32KB) шматки, і закривають з'єднання, які запитують кількість даних більше 2^17 (128Kb)." Уже у версії 3 або 2004, це поводження було змінено на використання 2^14 (16Кб) блоків. Починаючи з версії 4.0 або mid-2005, з'єднання в Mainline при запитах більше, ніж 2^14 (16Кб), і деякі клієнти пішли цьому прикладу. Помнете, що block-запити менше, ніж шматки (>= 2^18 байт), тому будуть необхідні численні запити, щоб скачати весь шматок.

Властиво, специфікація дозволяє 2^15 (32Кб) запити. Реальність така, що всі клієнти починаючи із сьогоднішнього моменту будуть використовувати 2 ^ 14 (16Кб) запити. Через клієнтів, які прив'язані до такого розміру запитів, рекомендується реалізовувати програми, що роблять запити саме такого розміру. Менші розміри запитів приводять до підвищення накладних витрат у зв'язку зі збільшенням кількості необхідних запитів, проектувальники радять не робити розмір запитів менше, ніж 2 ^ 14 (16Кб).

Вибір граничного розміру запитуваного блоку не дуже ясний. Mainline версії 4 здійснює 16 Кб-Ие запити, більшість клієнтів будуть використовувати цей розмір. У той же час розмір 2^14 (16Кб) представляється підлоги-офіційним (наполовину офіційним, тому що офіційна розділяція протоколу не обновлялася), тому, по суті, неправильним (не відповідної специфікації). У той же час, дозвіл бо'льших запитів розширює набір можливих пірів, і при виключенні дуже низької пропускної здатності з'єднання (<256кб/сек), кілька блоків буде завантажене в один choke-timeperiod, у такий спосіб просте приписання старої межі розміру

блоку викликає мінімальне погіршення роботи. Через цього фактора, рекомендується тільки старе 2^17 (128 КБ) максимальне обмеження розміру.

View 2. Поточна версія має принаймні наступні помилки: Mainline почали використовувати 2^14 (16384) байт запити, коли він був єдиним з існуючих клієнтів, тільки "офіційна специфікація" усе ще говорила про застарілому 32768-байтовомц значенні, що не було в дійсності ні розміром значення за замовчуванням, ні дозволеним максимумом. У версії 4 поводження запитів не змінилося, але максимально припустимий розмір запиту став рівним значенню розміру за замовчуванням. В останній версії Mainline максимум змінився до 32768 (помітьте, що ця перша поява 32768 або для значення за замовчуванням, або для максимального розміру запиту з моменту появи першої версії). Твердження: "більшість старих клієнтів використовують 32KB запити" – є помилковим. Обговорення запитів не приймає наслідки латентності до уваги.

```
piece: <len=0009+X><id=7><index><begin><block>
```

Piece-повідомлення змінної довжини, де X – довжина блоку. Корисне навантаження повідомлення містить наступну інформацію:

– index: ціле визначальне із вказівкою нулів індекс шматка;

– begin: ціле, що визначає із вказівкою нулів байтове зсув усередині шматка;

– block: блок даних, що є підмножина шматка з певним індексом.

```
cancel: <len=0013><id <= 8><index><begin><length>
```

Cancel-повідомлення фіксованої довжини, використовується для скасування блокування запитів. Корисне навантаження повідомлення ідентичне тої, котра була в " повідомленні-запиті" ("request" message). Повідомлення звичайно використовується під час стратегії "Кінця гри" (End game).

```
port: <len=0003><id=9>< listen-port>
```

Port-повідомлення відсилається за допомогою нових версій Mainline, що реалізує DHT Tracker. Порт для прослуховування є портом який DHT вузол прослуховує. Цей peer повинен бути вставлений у локальну таблицю маршрутизації (якщо DHT Tracker підтримується).

Алгоритми

Черги

View 1. Взагалі пірам рекомендується тримати кілька невиконаних запитів для кожного з'єднання. Інакше повний круговий обіг повідомлення (туди-назад, round trip, RT) зажадає завантажити блок до завантаження нового блоку (круговий обіг PIECE-повідомлення, і далі REQUEST-повідомлення). У зв'язку з високим BDP (результат затримки смуги пропущення, високої латентності або high bandwidth), це може привести до істотної втрати ефективності.

Конструкторське зауваження: Це найбільш важливий показник продуктивності. Статична черга з 10 заявок є прийнятною для 16 Кб-их блоків при зв'язку 5 мб/сек з латентності 5 мс. Стає дуже розповсюдженим зв'язок з великою пропускною здатністю, так це підштовхує проектувальників інтерфейсів передбачати можливість змін.

View 2. більша частина інформації в розділі "Черги" є помилковою або вводить в оману. Просто до відома, що "установки за замовчуванням для 5 вихідних запитів" не можуть бути вірними в плині довгого часу, "32 KB блоки" – є помилковими, оскільки ви, як правило, не використовуєте 32 КБ блоки, і набудовуєте довжину черги, змінивши цей параметр (видимий розмір блоку) і намагаєтеся виміряти ефект, це погана ідея.

Супер-сідування

Властивість супер сід (super-seed) в і над S-5.5 є новим алгоритмом сідування, спроектованим для допомоги ініціаторові торренту з обмеженою пропускною здатністю "завантажувати" великий торрент, зменшуючи кількість обсягу даних, необхідних для вивантаження (upload) і для породження нових сідів у торренті.

Коли клієнт-сід переходить у режим " супер-сід" , він не буде виступати в якості стандартного сіду, але маскується як звичайний клієнт без яких-небудь даних. Як тільки клієнти підключаться до нього, він буде інформувати їх про те, що він одержав шматок – шматок, що ніколи посилав, або, якщо всі шматки вже

послав, він є дуже рідким. Це спонукає клієнта до спроби скачати тільки цей шматок.

Коли клієнт закінчив завантаження шматка, сід не буде інформувати його про будь-які інші шматки доти, поки він бачить ці шматки відіслані раніше, принаймні в одного іншого клієнта. Доти, клієнт не буде мати доступ до будь-яких інших шматків, і тому не буде витрачатися пропускна здатність сіду.

Цей метод привів до набагато більше високої ефективності сідування, за допомогою примуса пірів в одержанні тільки найрідших даних, це й скорочення надлишкової кількості посилаємих даних, і обмеження обсягу даних, що посилаються пірам, які не сприяють поширенню цих даних(do not contribute to the swarm). До цього, сід повинен був вивантажити від 150% до 200% від загального розміру торренту перед тим як інші клієнти стають сідами. Проте, великий торрент , сідуємий єдиним клієнтом, запущеним у режимі 'супер сід' у стані зробити це тільки після 105% вивантаження даних. Це на 150-200% ефективніше, ніж при використанні стандартних сідів.

Режим 'супер-сід' не рекомендується для повсюдного використання. Хоча він допомагає в поширенні більших рідких даних, оскільки він обмежує вибір шматків, які клієнт може завантажувати, він також обмежує можливості цих клієнтів, у плані завантаження даних для вже частково отриманих шматків <з даного сіду>. Таким чином, супер-сід режим рекомендується тільки для ініціюючих сідування серверів.

Стратегія завантаження шматків

Клієнти можуть вибирати для завантаження шматки у випадковому порядку

Краща стратегія полягає в тім, щоб завантажувати найрідші шматки в першу чергу. Клієнт може визначити їх за допомогою зберігання первісного bitfield кожного піру й наступних їхніх відновлень при одержанні have-повідомлень. Потім клієнт може скачати шматки, що зустрічаються з мінімальною частотою в bitfield'ах пірів. Помітьте, що будь-яка стратегія

визначення найбільш рідкого шматка (Rarest First стратегія) повинна включати елемент випадкового вибору з, принаймні, декількох найбільш рідких шматків, тому що одночасна спроба багатьох клієнтів перейти до того самого "самого рідкого" шматку є непродуктивною.

Кінець гри (End game)

Коли завантаження майже завершене, є присутньою тенденція повільного завантажування останніх декількох блоків. Для прискорення цієї операції, клієнт посилає запити про всі свої загублені блоки всім своїм пірам. Щоб це не стало жахливо неефективно, при стрибку необхідного блоку клієнт посилає cancel-повідомлення всім іншим пірам.

Існують не роздільовані граничні значення, рекомендовані відсотки, або кількість блоків, які повинні використовуватися як орієнтир або як Recommended Best Practice.

Коли переходити до стратегії "Кінець гри" – питання спірний і вимагає обговорення. Деякі клієнти переходять до стратегії "Кінець гри", коли всі шматки були запитані. Інші чекають доти, поки кількість незавантажених блоків (blocks left) стане менше, ніж кількість переданих блоків(blocks in transit, очевидно запитаних, але не прийнятих), і не більш ніж 20. Ідея зберегти кількість незакінчених блоків до 1 або 2 блоків представляється гарною для мінімізації накладних витрат, і якщо ви випадковим образом запитуєте блоки, то маєте низькі шанси скачати дублікати. Докладніше про протокольні витрати (protocol overhead), можна прочитати тут: http://hal.inria.fr/ inria-00000156/en

Блокування(Choking) і оптимістичне розблокування (Optimistic Unchoking)

Блокування (Choking, удушення, засміття) відбувається з кількох причин. Відстеження перевантажень в TCP відбувається дуже погано, коли одночасно відбувається відсилання через кілька з'єднань. Крім того блокування дозволяє кожному піру використовувати алгоритм "зуб за зуб" (for-tat-ish) для одержання доброї швидкості завантажування.

Описаний нижче алгоритм блокувань застосовується на справжній момент. Надто важливо, щоб всі наступні алгоритми однаково добре працювали й у їхніх мережах, що в повністю икористовують, і в мережах, великою частиною що використовують поточний алгоритм.

Є кілька критеріїв гарного алгоритму блокування, яким він повинен задовольняти. Він повинен обмежити число одночасних віддач для гарної продуктивності TCP. Алгоритм повинен дозволяти уникнути частого чергування блокування й розблокування, такий механізм відомий як "фібриляція"(known as 'fibrillation'). Алгоритм припускає "оплату" пірам, які дозволяють <клієнтові> завантажувати. Нарешті, алгоритм повинен з деякою частотою випробовувати невикористовувані з'єднання, щоб перевірити краще вони використовуваних у цей час чи ні, такий механізм називається оптимістичним розблокуванням (optimistic unchoking).

Поточний алгоритм блокування уникає фибрілляції змінюючи заблоковані піри не частіше ніж раз в 10 секунд.

Піри, що мають кращу швидкість віддачі (у порівнянні зі завантажуючими), але не зацікавлені ... Коли вони стає зацікавленим, піри, що здійснюють завантаження з поганою швидкістю віддачі, відключаються. Якщо клієнт має повний екземпляр файлу, він ґрунтується на швидкості віддачі...

Офіційні розширення протоколу

На сучасний момент є кілька офіційних розширень протоколу.

Розширення "Fast Peers"

Зарезервований біт: третій біт в 8-ом захищеному байті (reserved byte), тобто reserved [7] |= 0x04

Специфікація роздільована на сайті BitTorrent тут: http://bittorrent.org/beps/bep_0006.html

Розподілені хеш-таблиці (Distributed Hash Table)

Зарезервований біт: останній біт в 8-ом захищеному байті (reserved byte), тобто reserved[7] |= 0x01

Це розширення дозволяє пірам працювати без використання стандартного трекеру. Peer, що реалізує цей протокол, сам стає "трекером" і зберігає списки інших вузлів/пірів, які можуть використовуватися для знаходження нових пірів.

Специфікація роздільована на сайті BitTorrent тут: http://bittorrent.org/beps/bep_0005.html

Шифрування з'єднання

Це розширення дозволяє створювати зашифровані з'єднання між пірами. Це може допомогти обійти обмеження, які накладають деякі провайдери інтернету на BitTorrent-трафік.

Специфікація описана в Wiki проекту Azureus: http://www.azureuswiki.com/index.php/Message_Stream_Encryption

Розділяція є досить повної, однак потрібно уточнити пари моментів щодо шифрованих з'єднань: коли вони повинні прийматися й відкіт до звичайних з'єднань у випадку невдачі при встановленні шифрованих з'єднань.

Неофіційні розширення протоколу

Протокол повідомлень Azureus

Зарезервований біт: 1.

WebSeeding

Можливість роздавати торрент через веб-сервер звичайно називається WebSeeding. Вона дозволяє HTTP-серверу виступати в ролі піру в BitTorrent мережі.

Протокол розширень

Зарезервований біт: 44, четвертий найбільш значимий біт в 6-ом зарезервованому байті, тобто reserved[5] |= 0x10.

Розширення протоколу переговорів

Зарезервовані біти: 47 і 48.

Зарезервований біт: 21.

Протокол, що враховує місце розташування піру (у географічному змісті), для кращої продуктивності. Специфікація може бути знайдена тут – http://wiki.theory.org/BitTorrent_ Location-aware_Protocol_1.0_Specification

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.1. З неї ми бачимо, що система представляє собою взаємодію наступних структурних блоків:

1. Інтерфейс користувача головного вікна програми.

2. Панель швидкого доступу до основних функцій менеджера завантажень з використанням протоколу BitTorrent, яке включає в себе наступні функції:

– Додати завантаження.

– Запустити завантаження.

– Зробити паузу.

– Перервати завантаження.

– Швидкість завантаження.

– Кількість одночасних завантажень.

– Посилання до форуму на сайті підтримки.

3. Вікно статусів, яке включає в себе:

– Перелік усіх завантажень.

– Категорії файлів, які завантажуються (програми, архіви, музика, відео).

– Топ завантажень (програми, архіви, музика, відео, пошук).

– Новини.

– Стан (завантаження, чекання завантаження, заплановано, помилки, пауза, завантажено).

– Історія.

– Видаленні завантаження.

4. Основна панель менеджера завантажень з використанням протоколу BitTorrent:

– Файл (топ завантажень, імпорт завантажень, імпорт, експорт, вихід).

– Завантаження (додати завантаження, додати групу завантажень, перевірити оновлення, пауза, видалити, видалити заразом з файлом, запланувати, перезавантажити заново, копіювати URL, відкрити файл, відкрити папку, скопіювати файл, перемістити файл, меню Windows, робота з архівом, коментарі,

знайти дзеркала, додати в менеджер сайтів, властивості).

– Дії (стартувати все, призупинити все, тимчасова зупинка завантажуємих, видалити все, знайти, знайти далі, швидкість).

– Вид (налаштування кнопок, сортування списку, список завантажень, звук, категорії, лог завантажень, плаваюче вікно, скіни, мова інтерфейсу: українська, англійська).

– Автоматизація (стартувати усі завантаження при запуску програми, стартувати усі завантаження при появі інтернету, стартувати усі завантаження по часу, відновити зв'язок при обриві, відключитися від інтернету після завершення завантажень, перевірити завантажені файли на віруси).

– Інструменти (пошук, історія, менеджер сайтів, розклад, плагіни, налаштування: загальні, з’єднання, завантаження, проксі, автоматизація, менеджер сайтів, розклад, інтерфейс, інші, плагіни).

– Довідка (зміст, домашня сторінка, он-лайн підтримка, повідомити про помилку, форум, перевірити оновлення, про програму).

5. Вікно завантажень:

– Ім’я файлу.

– Стан завантаження.

– Розмір файлу.

– Скільки залишилося об’єму даних для завантаження файлу.

– Швидкість завантажень.

– Коментарі.

6. Блок основних функцій менеджера завантажень з використанням протоколу BitTorrent, який розташовується у треї.

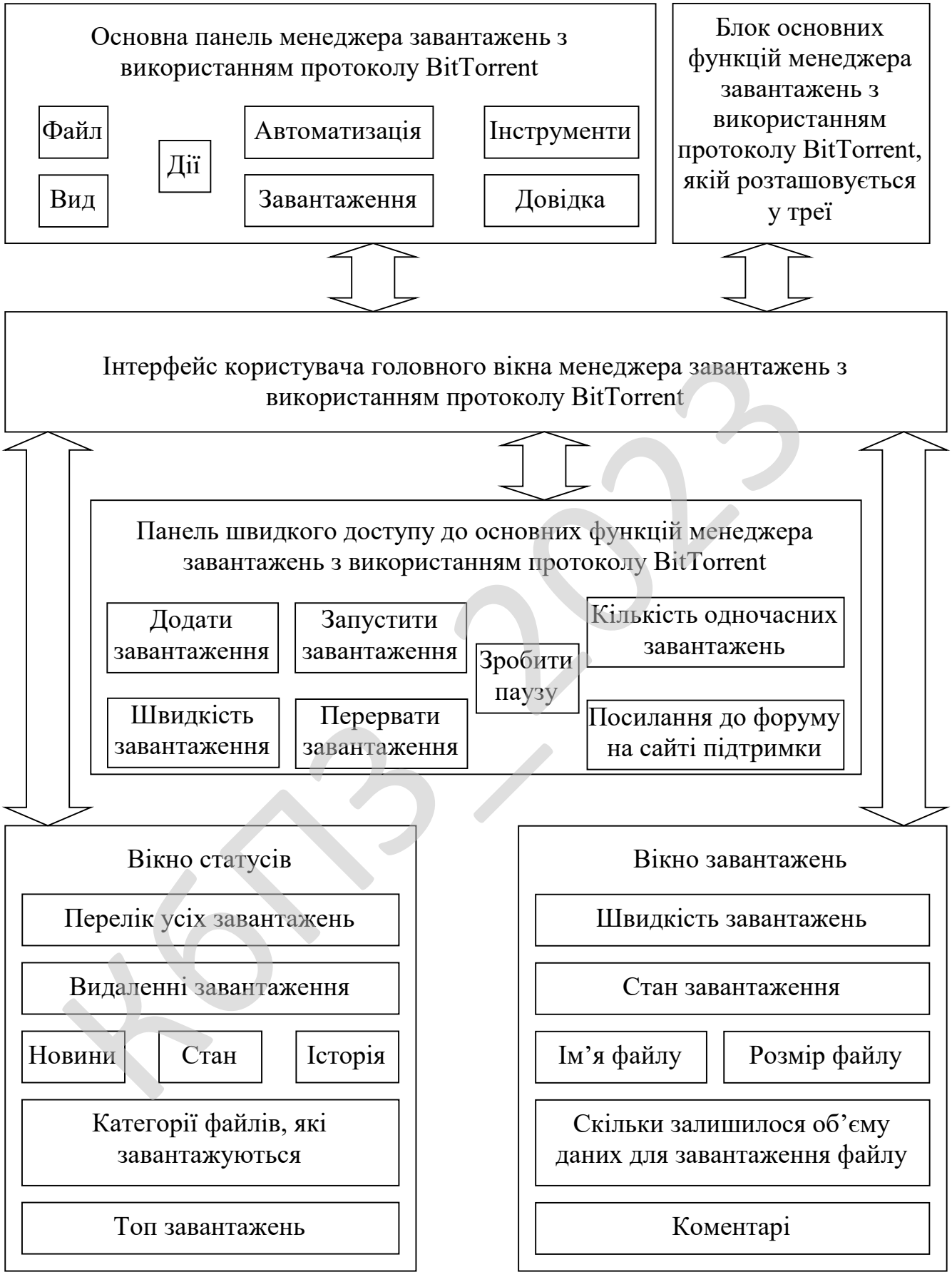

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціонально система складається з наступних блоків:

– функціональний блок керування категоріями завантажень;

– функціональний блок розпізнавання контенту при роботі через HTTP;

– функціональний блок роботи з командним рядком;

– функціональний блок синхронізації (автооновлення) файлів на сервері й локальному ПК;

– функціональний блок можливості послухати/подивитися музичні й відео файли в процесі завантаження. Автоматичне одержання інформації про MP3 файлах при старті завантаження;

– функціональний блок можливості завантажувати html-сторінки разом з картинками;

– функціональний блок налаштування параметрів з'єднання, HTTP, HTTPS і FTP протоколів;

– функціональний блок роботи через HTTP і FTP проксі-сервера, підтримка NTLM і NTLM-проксі автентифікації;

– функціональний блок звірення MD5 суми завантаженого файлу;

– функціональний блок історії завантажень;

– функціональний блок перевірки завантажених файлів на відновлення;

– функціональний блок динамічного багатопотокового завантаження;

– функціональний блок робота з розкладом;

– функціональний блок дозавантаженя після обриву зв'язку з HTTP, HTTPS і FTP серверів;

– функціональний блок відключення ПК після завершення завантаження;

– функціональний блок оптимальних налаштувань для роботи з різними типами з'єднань (dial-up, ISDN, ADSL, LAN) і на різних швидкостях;

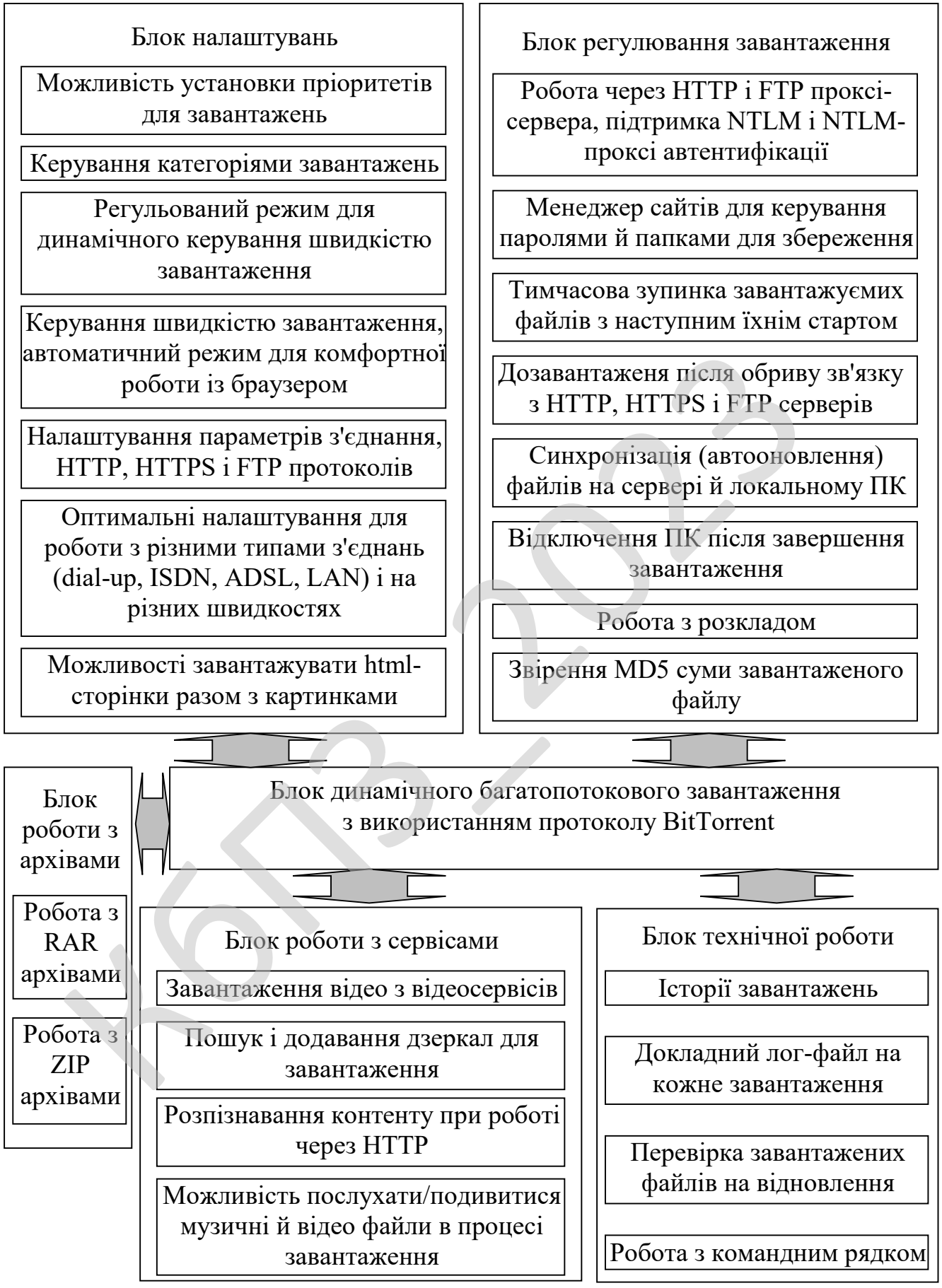

Рисунок 3.2 – Функціональна схема системи

– функціональний блок тимчасової зупинки завантажуємих файлів з наступним їхнім стартом у тому же стані;

– функціональний блок пошуку і додавання дзеркал для завантаження;

– функціональний блок завантаження відео з відеосервісів;

– функціональний блок роботи з ZIP архівами: можливість перегляду вмісту ZIP архівів перед завантаженням, можливість завантажувати тільки обрані файли з архіву, можливість перевіряти ZIP архіви й відновлювати ушкоджені файли, а також можливість розпаковувати архіви;

– функціональний блок роботи з RAR архівами: можливість перевіряти RAR архіви, можливість розпаковувати архіви;

– докладний лог-файл на кожне завантаження;

– функціональний блок менеджеру сайтів для керування паролями й папками для збереження;

– функціональний блок керування швидкістю завантаження, автоматичний режим для комфортної роботи із браузером;

– функціональний блок можливість установки пріоритетів для завантажень;

– регульований режим для динамічного керування швидкістю завантаження.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Підсистема налаштувань.

– Налаштування ПЗ.

– Налаштування проксі-сервера.

– Підсистема завантажень.

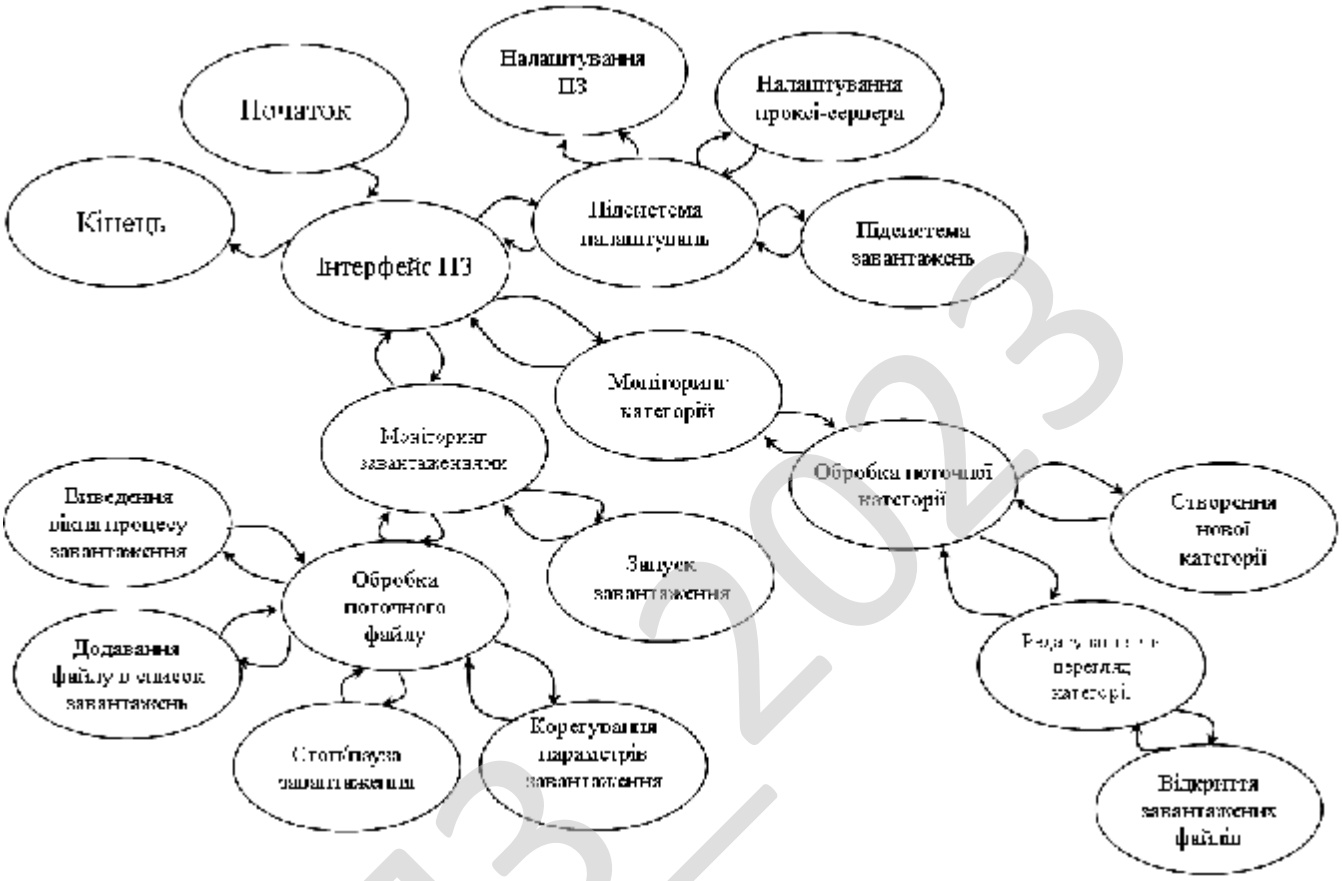

Рисунок 3.3 – Діаграма взаємодії процесів

– Моніторинг категорій.

– Обробка поточної категорії.

– Створення нової категорії.

– Редагування чи перегляд категорії.

– Відкриття завантажених файлів.

– Моніторинг завантаженнями.

– Запуск завантаження.

– Обробка поточного файлу.

– Виведення вікна процесу завантаження.

– Додавання файлу в список завантажень.

– Стоп/пауза завантаження.

– Корегування параметрів завантаження.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

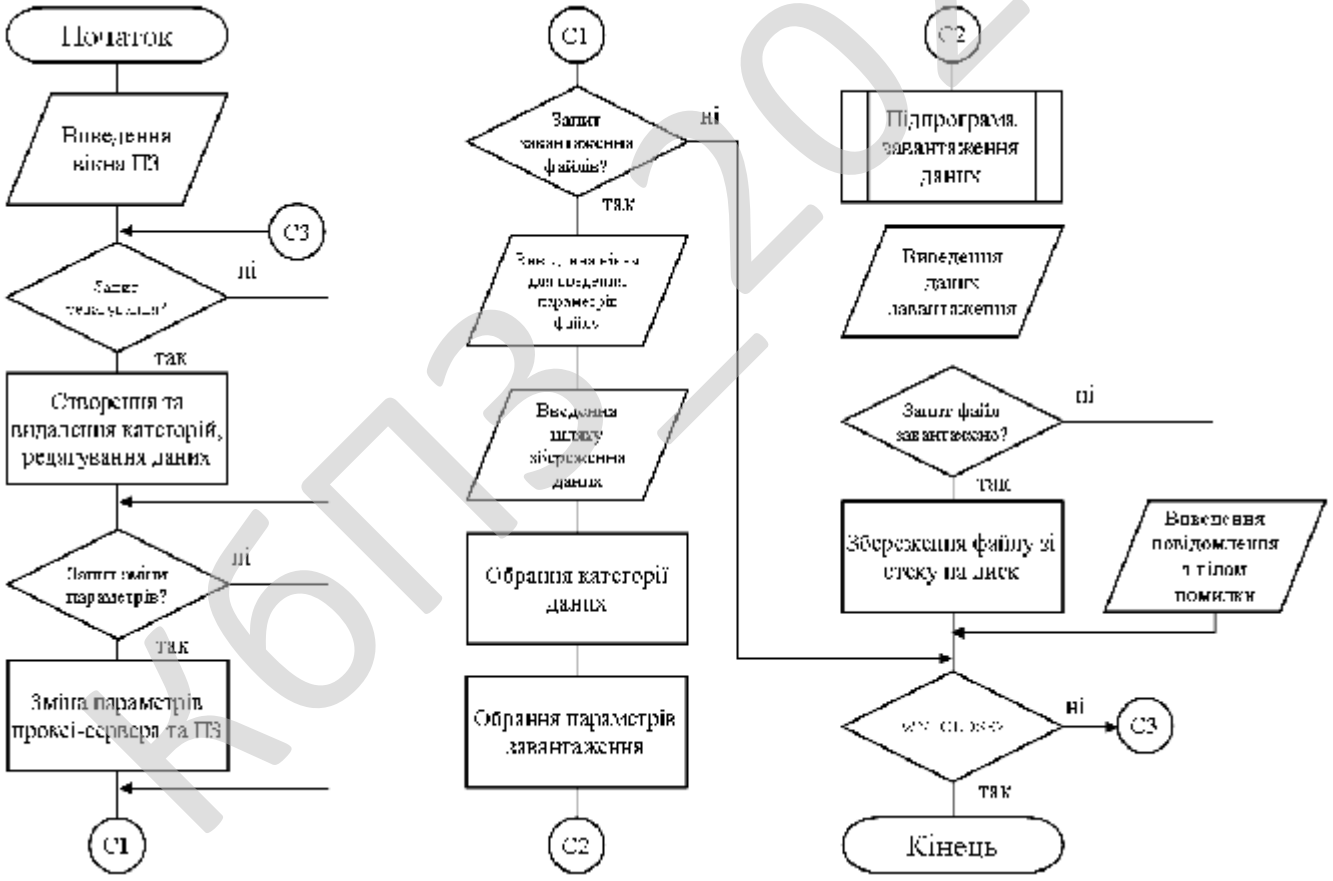

Рисунок 4.1 – Блок схема основної програми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного

циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

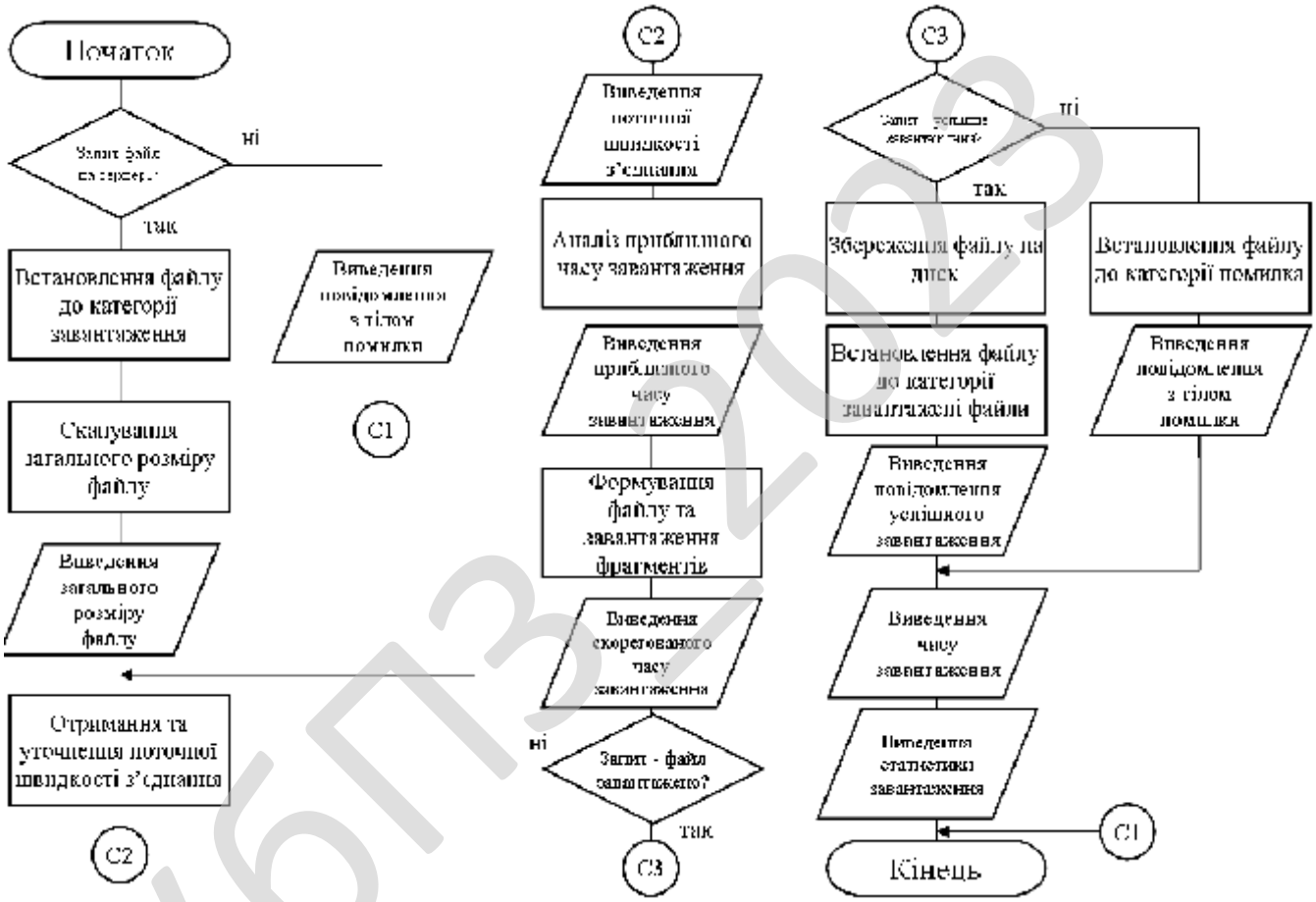

Рисунок 4.2 – Блок схема підпрограми

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає оптимізації програми високого рівня, також те,

що при розробці програми слід надати особливу увагу модулям обробки категорій.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Розглянемо реалізацію проекту. Торрент менеджер – це практично тільки GUI, що одержить у своє розпорядження інтерфейс завантажувача.

Сам завантажувач буде створюватися/знищуватися усередині системи автоматизованого завантажування (далі САЗ), назовні будуть виведені тільки її методи Пауза/Продовжити/Скасувати + статистика й лог. От цей інтерфейс (у такому виді він зараз у САЗ):

```
IRegionTilesDownload = interface ['{4BF3D0E8-3971-4EDC-97F3-44461D6A54FD}']
procedure SaveSession (ASessionFileName: string);
procedure Abort;
function GetPaused: Boolean;
procedure SetPaused (APaused: Boolean);
property Paused: Boolean read GetPaused write SetPaused;
function GetFinished: Boolean;
property Finished: Boolean read GetFinished;
function GetTotalInRegion: Int64;
property TotalInRegion: Int64 read GetTotalInRegion;
function GetDownloaded: Int64;
property Downloaded: Int64 read GetDownloaded;
function GetProcessed: Int64;
property Processed: Int64 read GetProcessed;
function GetDownloadSize: Double;
property DownloadSize: Double read GetDownloadSize;
function GetElapsedTime: TDateTime;
property ElapsedTime: TDateTime read GetElapsedTime;
function GetStartTime: TDateTime;
property StartTime: TDateTime read GetStartTime;
function GetZoom: Byte;
property Zoom: Byte read GetZoom;
end;
```

Інтерфейс лога (повідомлення які зараз виводяться в Memo):

```
ILogForTaskThread = interface (ILogSimple) ['{8583BE 82-1239-4E0 F-9B96-
D2810232282A}']
function GetLastMessages (AMaxRowsCount: Cardinal; var ALastId: Cardinal;
out AcntLines: Cardinal): WideString; safecall;
end;
```

Використання проксі з'єднань.

```
procedure InHttp(Port: Word; IP, Login, Password: String; Http:
                TidHTTP);
begin
      Http. Request. ProxyPort := Port;
      Http. Request. ProxyServer := IP;
      Http. Request. ProxyUsername := Login+'5';
      Http. Request. ProxyPassword := Password;
end;
function LoadPrice (const URL, FileName: String; Http: TIdHTTP): Boolean;
var tmpStream: TFileStream;
begin
      tmpStream := TFileStream. Create (FileName, fmCreate);
      tryHttp. Get (URL, tmpStream);
      tmpStream. Free;
      exceptOn E: Exception do
      begin
            if E.ClassName = 'EIdSocketError'
            then MessageText := 'Невірно зазначений порт'
            else
            if E.ClassName = 'EIdProtocolReplyError'
            then MessageText := 'Неправильний логін/пароль';
            Result := MessageError (MessageText);
            tmpStream. Free;
            DeleteFile (PChar (FileName));
      end;
      end;
end;
```

Використання методу.

```
Uses OleCtnrs, IdHTTP;
constPrice = 'price. xls';
URL = ' https://www.youtube.com/watch?v=gtc11GYhddw';
Port = 3128;
IP = '192. 168.1. 2';
Login = 'internet';
```

```
Password = 'www';
Var Http: TIdHTTP;
begin
      Http := TIdHTTP. Create (nil);
      InHttp (Port, IP, Login, Password, Http);
      if LoadPrice (URL, Price, Http) then Convert;
      Http.Free;
      Readln;
end.
```

Перевірка URL адреси за допомогою TNMHTTP.

```
procedure Button1Click (Sender: TObject);
begin
      {Намагаємося одержати заголовок}
      NMHTTP1. Head (Edit1. Text);
      {Якщо URL невірний, то тут вискочить помилка}
end;
```

Одночасне закачування зазначених URL у заданий каталог.

```
type
 THTTPThread = class (TThread)
 private
 {Для кожного процесу – створюємо свій компонент TNMHTTP}
 FHTTP: TNMHTTP;
 protected
// Execute викликається при запуску процесу;
 override – заміняємо
// існуючу процедуру базового класу TThread
 procedure Execute;
 override;
// створена нами функція, виконання якої синхронізується в Execute
 procedure Do Work;
 public
// створена нами рядок, що вказує процесу, який URL йому потрібно скачати
 URL: string;
 end;
// У форму потрібно помістити три кнопки TButton,
// одне поле TEdit і один список
// TListBox. При натисканні на кнопку Button1 викликається оброблювач події
// OnClick – Button1Click. Перед цим в TEdit потрібно
// ввести шлях до каталогу, в
// якому будуть зберігатися закачанні файли, а ListBox1
// потрібно заповнити списком
```

```
// URL-ів для закачування (за допомогою кнопок Add
// (Button2) і Delete (Button3)).
procedure TForm1. Button3Click (Sender: TObject);
begin
      {Видалення виділеного URL зі списку}
      if ListBox1. ItemIndex >= 0 then
      ListBox1. Items. Delete (ListBox1. ItemIndex);
end;
procedure TForm1. Button2Click (Sender: TObject);
 var s: string;
begin
       {Додавання URL у список}
       s := InputBox ('Додати', 'Уведіть URL: ', '');
       if s '' then
       ListBox1. Items. Add (s);
end;
procedure TForm1. Button1Click (Sender: TObject);
 var i: Integer;
begin
       {Перевірка на існування каталогу}
       if Length (Edit1. Text) > 0 then
       if not DirectoryExists (Edit1. Text) then
       MkDir (Edit1. Text);
       {Далі йде створення для кожного URL у списку свого процесу}
       for i := 0 to ListBox1. Items. Count-1 do
      begin
            with THTTPThread. Create (True) do
          begin
              {Створюємо припинене завдання, указуємо їй її URL і
              запускаємо її}
               URL := ListBox1. Items[i];
               Resume;
           end;
       end;
end;
// Оператори процесу THTTPThread
procedure THTTPThread. Execute;
begin
// Робимо так, щоб кожний процес виконувався
// одночасно з іншими (синхронізація)}
 Synchronize (Do Work);
end;
```

```
procedure THTTPThread. Do Work;
var i: Integer;
begin
       {Створюємо компонент TNMHTTP}
       FHTTP := TNMHTTP. Create (Form1);
      {Результат треба записувати у файли}
       FHTTP. InputFileMode := True;
      {Підбираємо імена для файлів}
       i := 1;
      while FileExists (Form1. Edit1. Text+'\page'+IntToStr (i)+'. htm') do
       Inc (i);
      {Указуємо, у які саме файли класти результат}
       FHTTP. Body := Form1. Edit1. Text+'\body'+IntToStr (i)+'. htm';
      FHTTP. Header := Form1. Edit1. Text+'\header'+IntToStr (i)+'. txt';
      {Намагаємося надіслати запит}
       FHTTP. Get (URL);
      {Перед завершенням процесу не забуваємо звільнити пам'ять з-під
      компонента}
       FHTTP. Free;
end;
```

Опишемо загальний інтерфейс (ISASCommonAPI) з парою методів, щоб можна було стартувати збережені сесії з менеджера (автоматично, при перезапусках САЗ), і можливо, управляти видимістю головного вікна САЗ. Це, що стосується з боку САЗ. З боку dll, повинна експортуватися одна функція:

```
function GetDownloadManager(ASASCommonAPI:ISASCommonAPI):IDownloadManager;
```

яка як параметр одержує загальний інтерфейс, і повинна повернути інтерфейс менеджера (він буде зберігатися протягом усього часу існування програми, і відповідно викликатися ця функція буде при завантаженні й ініціалізації САЗ). У цьому інтерфейсі повинні бути як мінімум метод:

```
procedure ShowUI;
```

(користувач нажав на тулбарі кнопку "Показати менеджера") і метод:

```
procedure AddTask (ADownloader: IRegionTilesDownload; ALog:
ILogForTaskThread);
```

(користувач виділив регіон і запустив закачування).

От і все. Менеджерові дається повна свобода дій над закачуваннями (розклад, черговість і т.д.), плюс, можна буде зробити щоб при згортанні менеджера він приховував головне вікно САЗ, а сам ховався в трей.

У лог скидає відповідь від сервера: хедер і властиво, сам контент файлу.

```
var type TBuf = array [0.. 65535] of char;... .
Procedure DownLoad ();
Var Send_buf, Recv_buf: TBuf;
Sock : TSocket;
sockAddr : TSockaddr;
Cookies : String;
Query : TBuf;
Tmp : String;
buf : array [0.. 1023] of Char;
RcvLen : Integer;
Begin
      Sock:=socket (AF_INET,SOCK_STREAM,IP PROTO_TCP);
      sockAddr. sin_family := AF_INET;
      sockAddr. sin_port := htons (80);
      sockAddr. sin_addr. S_addr := Resolve ('ismily.ua');
      connect (Sock, sockAddr, sizeof (TSock ddr));
      if (Sock <> INVALID_SOCKET) then
      begin
            StrPCopy (Send_buf, 'GET /111. jpg HTTP/1.1'#13#10 +'Host:
            ismily.ua'#13#10+' User-Agent: Opera/9.63 (Windows NT 5.1; U;
            ua) Presto/2.1.1'#13#10+'Accept: text/html, application/xml;
            q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif,
            image/ x-xbitmap, */*; q=0.1'#13#10+' Accept-Language: ua-UA,
            ua; q=0.9, en; q=0.8'#13#10+' Accept-Charset: iso-8859-1, utf-
            8, utf-16, *; q=0.1'#13#10+' Accept-Encoding: deflate,
            identity, *; q=0'#13#10+'Connection: Close, TE'#13#10+'TE:
            deflate, chunked, identity, trailers'#13#10#13#10);
            send (Sock, Send_buf, strlen (Send_buf), 0);
            Tmp:='';
            ZeroMemory (@buf, 1024);
            RcvLen:=recv (Sock, buf, 1024, 0);
            while RcvLen > 0 do
            begin
                  Tmp := Tmp + Copy (buf, 0, RcvLen);
                  RcvLen := recv (sock, buf, 1024, 0);
            end;
            StrPCopy (Recv_buf, tmp);
            closesocket (Sock);
      end;
end;
```

Завантаження файлу з Інтернет можлива декількома шляхами. За допомогою компонента TNMHTTP. Даний компонент дає можливість працювати по протоколі HTTP. Зокрема, метод NMHTTP. Get дозволяє виконувати стандартний запит GET протоколу HTTP для завантаження інформації. Крім методу GET підтримуються POST, PUT, DELETE, OPTIONS.

За допомогою компонента TNMFTP – даний компонент дозволяє працювати по протоколі FTP.

TIdFTP і TIdHTTP – компоненти Indy для роботи із протоколу FTP і HTTP відповідно.

Реалізація протоколу FTP і HTTP "вручну" шляхом установлення з'єднання із сервером через TClientSocket або TIdTCPClient. Даний метод складніше в реалізації, тому що прийде реалізовувати роботу з обраного протоколу згідно RFC, проводити обробку помилок і т.п.

Використання API функцій бібліотеки wininet. Dll (найбільш уживаної функції InternetOpen, InternetOpenURL, InternetReadFile / InternetWriteFile і InternetCloseHandle).

З описаних методів останній має перевага, тому що його реалізація досить проста й немає потреби замислюватися про тонкості (наприклад, про проксі-сервер і його настроювання). Крім того, в wininet. Dll утримується кілька десятків функцій для роботи з FTP, дозвону до провайдеру.

Завантаження файлу за допомогою TIdHTTP. Компонент бібліотеки Indy TIdHTTP містить ряд властивостей, які управляють роботою компонента. Розглянемо найцікавіші Властивості:

– HandleRedirects – Якщо дана властивість дорівнює True, те компонент обробляє коди протоколу HTTP (переадресація).

– Port – Порт, по якому виробляється з'єднання із сервером, за замовчуванням 80.

– ProtocolVersion – Версія протоколу. Можливий один із двох варіантів – pv1_0 для HTTP 1.0, pv1_1 для HTTP 1.1 (установлено за замовчуванням).

– RedirectMaximum – Максималана кількість переадресацій, що може бути виконано. Має сенс при HandleRedirects = True, за замовчуванням дорівнює 15. Призначення даного досить очевидно – може виявитися, що при запиті ресурсів відбудеться редиректна x1, звідти – на x2 і так далі нескінченно (у принципі, велика ймовірність, що ланцюжок редиректів може утворити кільце). При досягненні заданої граничної кількості редиректів буде викликаний оброблювач Do Request, і Якщо він не обробить дану ситуацію, то виникне виняткова ситуація EIdProtocolReplyError. На мій погляд значення 15 велике і я рекомендую використовувати значення порядку 5.

– Request – Параметри в заголовку HTTP запиту.

– UseNagle – Використання алгоритму Наггла (Nagle algorithm) при обміні. За замовчуванням використання алгоритму Наггла включене й виключати його не рекомендується.

Методи:

– Do Request – Виконання запиту й прийом відповіді.

– Get – Виконання методу Get.

Нижче наведене короткий опис основних властивостей, методів і подій компонента TNMHTTP.

Властивості:

– Body – рядок, що містить або шлях до файлу, у який буде записане тіло http-документа (Якщо властивість InputFileMode дорівнює True), або безпосередньо саме тіло (Якщо властивість InputFileMode дорівнює False). Тип: string.

– Header – рядок, що містить або шлях до файлу, у який буде записаний заголовок http-документа (Якщо властивість InputFileMode дорівнює True), або безпосередньо сам заголовок (Якщо властивість InputFileMode дорівнює False). Тип: string.

– HeaderInfo – структура, що містить різну інформацію про http-документ. Тип: THeaderInfo.

– InputFileMode – тип запису результату. Значення True – запис у файли, зазначені у властивостях Body і Header, False – запис у самі ці властивості. Тип: Boolean.

– OutputFileMode – тип відсилаємих даних (методами Put, Post і Trace). Значення True – дані для відправлення втримуються у файлах, зазначених при виклику цих методів, а False – у самих аргументах цих методів. Тип: Boolean.

Далі деякі властивості, успадковані від TPowerSock:

– BytesRecvd, BytesSent, BytesTotal – кількість відправлена, прийнята й загальна кількість байтів відповідно. Тип: LongInt.

– Connected – показує, чи встановлено в цей момент з'єднання. Тип: Boolean.

– BeenCanceled – показує, чи було перерване з'єднання із сервером. Тип: Boolean.

– Host – рядок, що містить хост-ім'я віддаленого комп'ютера. Заповнювати не треба, тому що ця властивість установлюється автоматично при виклику методів Get, Put, Postи т.д. Тип: string. Port-Integer, що містить порт віддаленого комп'ютера (заповнюється теж автоматично).

– TimeOut – таймаут у мілісекундах. Тип: Integer.

Методи:

– Get (URL: string) – надсилає запит на зазначений URL. Дані після виконання цього запиту записуються у файли або в самі властивості Body і Header (залежно від значення властивості InputFileMode).

– Head (URL: string) – надсилає запит на зазначений URL. Дані після виконання цього запиту записуються у файл або в саме властивість Header (залежно від значення властивості InputFileMode). На відміну від методу Get, при виклику Head запит відсилається тільки на заголовок http-документа.

– Post (URL, PostData: string) – надсилає запит на зміну http-документа (з адресою URL) на дані, що втримуються в параметрі PostData. Якщо OutputFileMode дорівнює True, то в PostData повинен утримуватися шлях до

файлу, що містить потрібні дані. Put (URL, PutData: string) – надсилає запит на створення http-документа (з адресою URL), що містить дані, передані в параметрі PutData.

Якщо OutputFileMode дорівнює True, то в PostData повинен утримуватися шлях до файлу, що містить потрібні дані. Trace (URL, TraceData: string) – надсилає запит на одержання відладочних даних (для налагодження з'єднання з HTTP-сервером). Дані для запиту потрібно вказати в параметрі TraceData. Якщо OutputFileMode дорівнює True, то в TraceData повинен утримуватися шлях до файлу, що містить потрібні дані. Delete (URL: string) – надсилає запит на видалення http-документа (з адресою URL).

Далі деякі методи, успадковані від TPowerSock:

– Abort і Cancel – переривають з'єднання й обмін даними.

– Disconnect – від'єднання від HTTP-сервера.

Події:

– OnAuthenticationNeeded – виникає, коли сервер вимагає вказівки імені користувача й пароля. В оброблювачі цієї події (Якщо воно виникне) Ви повинні відповісти серверу, заповнивши потрібними значеннями відповідні змінні. Примітка: Перед установленням з'єднання можна відразу заповнити поля UserID і Password у властивості HeaderInfo.

– OnAboutToSend – виникає, коли компонент TNMHTTP збирається відправляти дані (запит). В оброблювачі цієї події можна заповнити додатковою інформацією властивість SendHeader.

– OnFailure – виникає, коли поточна операція завершилася невдало, тобто відбулася помилка.

– OnRedirect – виникає, сервер переадресував посилання із зазначеної URL на інше посилання. Установивши параметр handled у значення True можна заборонити переадресацію й зупинитися на запитаної URL. Значення за замовчуванням – False.

– OnSuccess – виникає, коли поточна операція завершилася успішно, тобто запит був виконаний без помилок.

Далі деякі методи, успадковані від TPowerSock:

– OnConnect – виникає, коли з'єднання із сервером успішно встановлено.

– OnDisconnect – виникає, коли з'єднання із сервером завершено.

– OnConnectionFailed – виникає, коли з'єднання із сервером установити не вдалося.

– OnError – виникає, коли остання операція була завершена з помилкою.

– OnHostResolved – виникає, коли від DNS отримана IP-адреса зазначеного хосту.

– OnInvalidHost – виникає, коли DNS повернув помилку при спробі визначити IP-адресу зазначеного хосту.

– OnPacketRecvd – виникає, коли значення властивостей BytesRecvd та BytesTotal змінені, тобто була прийнята нова порція даних від сервера.

– OnPacketSent – виникає, коли значення властивостей BytesSent та BytesTotal змінені, тобто була відправлена нова порція даних на сервер.

– OnStatus – виникає, коли статус компонента був змінений (для відновлення візуального оповіщення користувача).

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки

підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'a даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо

використовує xor підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Блок меню.

– Блок клавіш швидкого доступу.

– Блок відображення категорій.

– Вікно завантаження.

На верхній частині вікна можна побачити навігаційне меню, яке складається з: Файл; Завантаження; Вид; Параметри; Довідка.

Блок відображення категорій включає в себе наступні категорії: Прийняті; Задачі; Історія; Видалені.

Задачі включають у себе наступні елементи: Очікування; Завантаження; Завантажуються; Зупинені; Помилка; Історія.

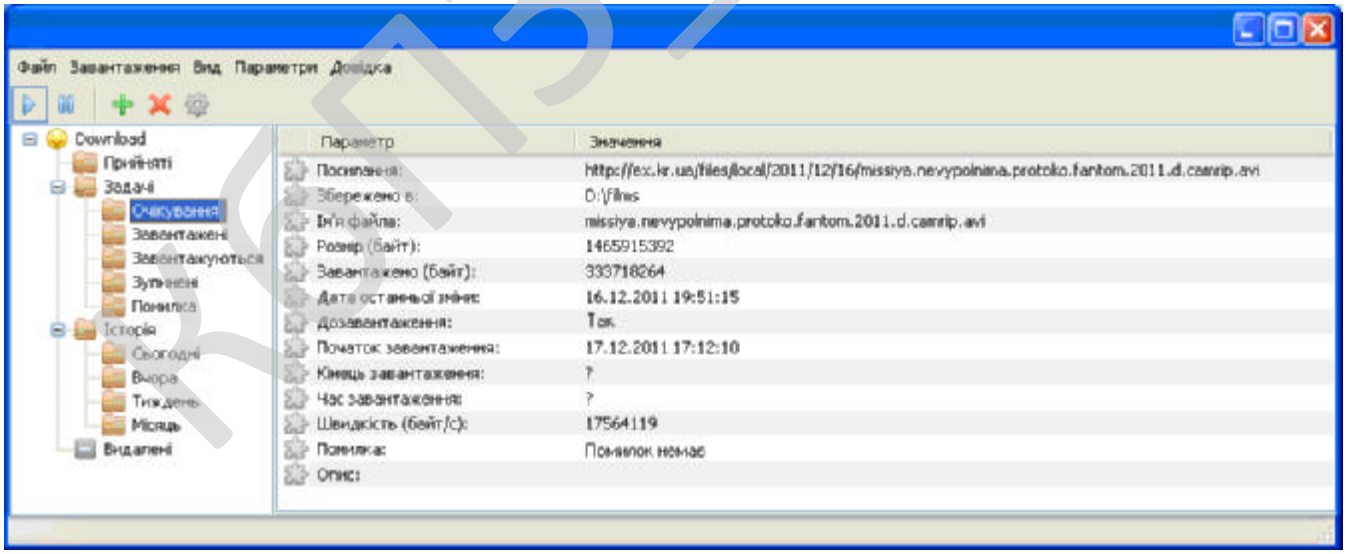

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

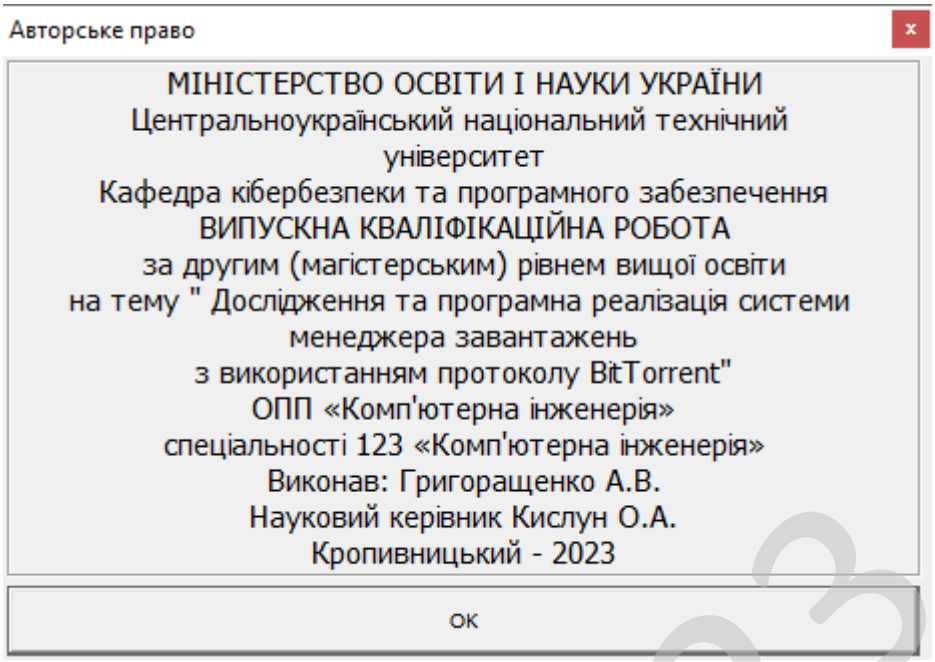

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права.

Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж.

Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень. Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію.

Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення.

Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи менеджера завантажень з використанням протоколу BitTorrent.

Метою розробки є дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Об'єктом дослідження є процес менеджера завантажень з використанням протоколу BitTorrent.

Предметом дослідження є методи менеджера завантажень з використанням протоколу BitTorrent.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод менеджера завантажень з використанням протоколу BitTorrent.

– Розроблено вітчизняний продукт менеджера завантажень з використанням протоколу BitTorrent, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	99
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	100000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 70 = 118 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	118	Ф 7.1-7.4
Впровадження	13	Д13
Всього	159	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{159 \cdot 1}{60 - 5} = 2,9 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	160	400	6,67
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	37,33

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{37 \cdot 3}{1,2} = 92,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 92,5/(60 \cdot 8) = 0,2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	7870	23610
Продакт-менеджер	0,25	7000	5250
Інженер-програміст	2,9	8500	73950
Інженер - електронщик	0,2	7000	4200
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,1	7000	2100
Дизайнер WEB	0,25	7000	5250
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,1	7000	2100
Всього за період розробки	$R_{сп}$ = 5,8	-	$Ф_{роб}$ = 137460

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{137460}{5,8 \cdot 60} = 395 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 29000 = 1858000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 185800 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 15.10.23 – джерело http://computorg.ua/ru/price.html.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	AMD Ryzen 3 4100 (100-100000510BOX AM4, 4 ядра, 8 потоків, 3.8 GHz, 4.0 GHz	-
Системна плата	Biostar A520MH 3.0 - AM4, DDR4 макс 6 ГБ, макс. частота ОП - 4400 MHz, LAN - Гбит/с, D-Sub (VGA), HDMI, 1 x M.2 2280, x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	GeForce GT730 4Gb Biostar (VN7313TH41 PCI-Express 2.0, 4 ГБ, GDDR3, 128 Bit,	-
Жорсткий диск	SSD M.2 2280 240GB Apace (AP240GAST280-1) 240 GB, TLC, M.2 SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2400 MHz eXceleram (E408247D DDR4, 4 ГБ, В наборі - 2	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" LG 22MP58VQ-P 5 мс IPS 1920x108 250/1000M:1 178/178 D-Sub+HDMI+DVI	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1858000	-	-
2. Передавальні пристрої	185800	-	-
Всього по групі	2043800	5	102190
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	100000	10	10000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	37857,75
Разом	$К_р$ = 2523008		$А_р$ = 249636,25

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 395 \cdot 159/99 = 635 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 635 \cdot 10 \cdot 0{,}01 = 64 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(635+64) = 154 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 635 \cdot 15 \cdot 0{,}01 = 95 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 1,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 206 \cdot 1,5 = 309 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 51):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 23,6 \cdot 50+35 = 1215 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з.}, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (309 + 1215 + 1702)/99 = 33 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0,01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 635 \cdot 15 \cdot 0,01 = 95 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 99 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 249636 \cdot 3/(99 \cdot 12) = 631 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	635
2. Додаткова зарплата виконавців	$З_д$	64
3. Відрахування на соціальні потреби	$С_{оц}$	154
4. Загальногосподарські витрати	$Г_{осп}$	95
5. Витрати на матеріали	$З_М$	33
6. Освоєння нових операційних систем, мов програмування	$О_п$	95
7. Амортизація основних фондів	$А_м$	631
8. Повна собівартість програмного забезпечення	$С_п$	1707
9. Плановий прибуток	$П_р$	940
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2647
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	529,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3176,4

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 635+64+154+95+33+95+631 = 1707 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 55 \cdot 1707 = 940 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3176
Всього капітальних витрат	–	3176

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	44286	36905
2. Витрати на електроенергію	$З_{ел}$	432	360
3. Витрати на амортизацію	$З_{ам}$	0	794
Всього витрат за рік	$І$	44718	38059

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування системи на рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 300 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 300 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 44286 \text{ грн},$$

до:

$$З_{р\,нов} = 250 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 36905 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\,баз} = 0{,}45 \cdot 300 \cdot 3{,}2 = 432 \text{ грн}.$$

$$З_{ел\,нов} = 0{,}45 \cdot 250 \cdot 3{,}2 = 360 \text{ грн}.$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3176	–	794
Всього відрахувань	-	–	3176	–	794

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Eв = (2647\text{-}1707) \cdot 99\text{-}(0{,}05 \cdot 2043800 + 0{,}5 \cdot 199177 + 0{,}25 \cdot 37031 + $$
$$+0{,}1 \cdot 100000 + 0{,}2 \cdot 143000) \cdot 3/12 = 30651 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{479208}{(2647\text{-}1707)\cdot 99\cdot 12/3} = 1{,}3 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	99
2. Повна собівартість розробленої програми	Грн.	1707
3. Ціна розробленої програми	Грн.	2647
4. Плановий прибуток від реалізації розробленої програми	Грн.	940
5. Рентабельність програмної продукції	%	55
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2523008
7. Загальний прибуток від реалізації програмної продукції	Грн.	93060
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	30651
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3176
11.Величина економічного ефекту у користувача програмної продукції	Грн.	5865
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,5

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (44718\text{-}38059)\text{-}0{,}25 \cdot 3176 = 5865 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{3176}{44718 - 38059} = 0{,}5 \text{ } року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми

роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці.

Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [4]

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців ІТ-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи

«оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали

максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розрахункова частина

Для захисного штучного заземлення будем застосовувати ветикальні електроди з сталевого прокату круглого перерізу діаметром *45 мм.*, довжиною *L=2 м.,* та горизонтальний електрод – метелева полоса з перетином *45·5* мм. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,6 м*. Відстань між вертикальними заземлювачами (електродами) *A=2,5 м*. Глибина закладення горизонтального контура заземлення *t=0,65 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,65+2/2=1,75 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр ветикальньного електрода (задан):

$$Dв = 45 \text{ мм.} = 0,045 \text{ м.}$$

Відношення $A/L=2,5/2=1,25$.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O=0,366(\rho/L)[lg(2L/Dв)+(1/2)lg((4T+L)/(4T-L))]=$$
$$=0,366(54,5/2)[(lg(2\cdot2/0,045)+(1/2\ )lg((4\cdot1,75+2)/(4\cdot1,75-2))]=$$
$$=20,6 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів $Кев=0,53$ при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [11].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 20,6 / (0,53\cdot4)= 9,75 \approx 10 \text{ шт.}$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2,5\cdot10=25,5\approx 26 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П =0,366(\rho\cdot K_П/L_П\ )lg(2(L_П\cdot L_П\ )/(B\cdot t))=$$
$$=0,366(40\cdot5/26)\cdot lg((2\cdot26^2)/(0,045\cdot0,65\ ))=20,2 \text{ Ом.}$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [11]:

$B = 45 \text{ мм.} = 0,045 \text{ м.}$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [11]:

$$R =(R_O\cdot R_П)/(\ R_O\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$
$$=(20,6\cdot\ 20,2)/(20,6\cdot\ 0,55+\ 10\cdot\ 20,2\cdot\ 0,53)=3,36 \text{ Ом.}$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \leq R_{ЗН}$ виконується ($3,36 \leq 4$).

Так як R суттево більше $R_{ЗН}$, зменшимо кількість вертикальних електродів до *9* і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює *9* при $R = 3,7$ *Ом.*

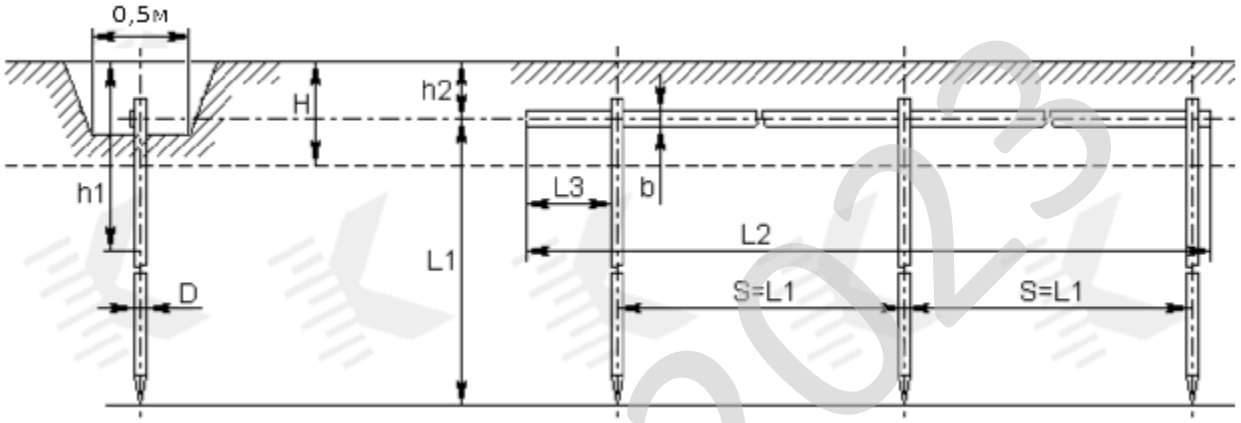

Рисунок 8.1 – Схема штучного заземлення .

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому. З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів о питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини. Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи менеджера завантажень з використанням протоколу BitTorrent.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів менеджера завантажень з використанням протоколу BitTorrent.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем менеджера завантажень з використанням протоколу BitTorrent.

– Досліджена система менеджера завантажень з використанням протоколу BitTorrent.

– На основі отриманих результатів досліджень створена програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання менеджера завантажень з використанням протоколу BitTorrent.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 5865 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,5 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Григоращенко А.В. Дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

4. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

5. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

6. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

7. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

8. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

9. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

10. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

11. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

12. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

13. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

14. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

15. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

16. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

17. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

18. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

19. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

20. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

21. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

22. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

23. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

24. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

25. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

26. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

27. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

28. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

29. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

31. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

32. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

33. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

34. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

35. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

36. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

37. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

38. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

40. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

41. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

42. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

43. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

44. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

45. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

46. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

49. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

50. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. – Випуск 2 (46) – Х.: ХУПС – 2016. – С. 146-149.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи менеджера завантажень з використанням протоколу BitTorrent.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи менеджера завантажень з використанням протоколу BitTorrent;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті пропозиції щодо підвищення працездатності ІТ-фахівців.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 118 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 13.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Кислун О.А.

Дослідження та програмна реалізація системи менеджера завантажень з використанням протоколу BitTorrent

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 53

Літера: РП

Кропивницький – 2023 року

BitTorrent_Procedures.pas - основні процедури менеджера завантажень з використанням протоколу BitTorrent

```
unit uProcedures;

interface

uses
  SysUtils, Windows, ShellAPI, ShlObj;

function BytesToText(Bytes : Integer) : String;
function GetFileVersion(FileName : String) : String;
function BrowserFolder(Owner : THandle) : String;
function CreateFileName(Url : String) : String;
function LocalAddress(Url : String) : Boolean;
function ExtractAddress(Url : String) : String;
function ExtractFileName(Url : String) : String;
function GetFreeSpace(Disk : String) : Int64;
function IsRun : Boolean;
function GetTimeStr(Secs : Integer) : String;

implementation

////////////////////////////////////////////////////////////////////////////////

function BytesToText(Bytes : integer) : String;
begin

  if Bytes div 1000 < 1 then Result := IntToStr(Bytes) + ' байт';

  if Bytes div 1000 >= 1 then Result := FloatToStr(Bytes/1000, ffNumber, 18, 1)
+ ' Кб';

  if Bytes div 1000 >= 1000 then Result := FloatToStr(Bytes/1000000, ffNumber,
18, 1) + ' Мб';

end;

////////////////////////////////////////////////////////////////////////////////

function GetFileVersion(FileName : String) : String;
var
  Data : Pointer;
  DataSize, InfoSize : Dword;
  Dummy : Cardinal;
  Buffer: array [0..MAX_PATH] of Char;
  Major1, Major2, Minor1, Minor2 : Integer;
  FileInfo : PVSFixedFileInfo;

begin

  StrCat(Buffer, PChar(FileName));
  DataSize := GetFileVersionInfoSize(Buffer, Dummy);

  if DataSize > 0 then
  begin

    GetMem(Data, DataSize);
    GetFileVersionInfo(Buffer, 0, DataSize, Data);
    VerQueryValue(Data, '\', Pointer(FileInfo), InfoSize);

    Major1 := FileInfo.dwFileVersionMS shr 16;
    Major2 := FileInfo.dwFileVersionMS and $FFFF;
    Minor1 := FileInfo.dwFileVersionLS shr 16;
    Minor2 := FileInfo.dwFileVersionLS and $FFFF;
```

```
    Result := IntToStr(Major1) + '.' + IntToStr(Major2) + '.' + IntToStr(Minor1)
+ ' build ' + IntToStr(Minor2);

    FreeMem(Data, DataSize);

  end;

end;

////////////////////////////////////////////////////////////////////////////////

//Дерево каталогів

function BrowserFolder(Owner : THandle) : String;
var
  TitleName   : String;
  lpItemID    : PItemIDList;
  BrowseInfo  : TBrowseInfo;
  DisplayName : Array [0..MAX_PATH] of char;
  TempPath    : Array [0..MAX_PATH] of char;

begin

  FillChar(BrowseInfo, SizeOf(TBrowseInfo), #0);

  BrowseInfo.hwndOwner := Owner;
  BrowseInfo.pszDisplayName := @DisplayName;

  TitleName := 'Виберіть директорію';

  BrowseInfo.lpszTitle := PChar(TitleName);
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;

  lpItemID := SHBrowseForFolder(BrowseInfo);

  if lpItemId <> nil then
  begin

    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);

    Result := TempPath;

  end else
  begin

    Result := '';

  end;

end;

////////////////////////////////////////////////////////////////////////////////

//Створення імені файлу

function CreateFileName(Url : String) : String;
var
  i : Integer;

begin

  Result := '';

  if Pos('//', Url) > 0 then Delete(Url, 1, Pos('//', Url) + 1);

  if Url = '' then Exit;

  if Pos('/', Url) > 0 then Delete(Url, 1, Pos('/', Url))
```

```
  else Delete(Url, 1, Length(Url));

  if Url = '' then
  begin

    Result := 'index.html';
    Exit;

  end;

  if Url[Length(Url)] <> '/' then
  begin

    i := Length(Url);

    repeat

      i := i - 1;

    until Url[i] = '/';

    Result := Copy(Url, i + 1, Length(Url) - i);

  end else Result := 'index.html';

end;

////////////////////////////////////////////////////////////////////////////////

//Завдання адреси

function LocalAddress(Url : String) : Boolean;
begin

  Result := False;

  if Pos('//', Url) > 0 then Delete(Url, 1, Pos('//', Url) + 1);
  if Pos('/', Url) > 0 then Delete(Url, Pos('/', Url), Length(Url));

  if LowerCase(Url) = 'localhost' then Result := True;
  if LowerCase(Url) = '127.0.0.1' then Result := True;

end;

////////////////////////////////////////////////////////////////////////////////

function ExtractAddress(Url : String) : String;
begin

  if Pos('//', Url) > 0 then Delete(Url, 1, Pos('//', Url) + 1);
  if Pos('/', Url) > 0 then Delete(Url, Pos('/', Url), Length(Url));

  Result := Url;

end;

////////////////////////////////////////////////////////////////////////////////

function ExtractFileName(Url : String) : String;
begin

  if Pos('//', Url) > 0 then Delete(Url, 1, Pos('//', Url) + 1);
  if Pos('/', Url) > 0 then Delete(Url, 1, Pos('/', Url));

  Result := Url;

end;

////////////////////////////////////////////////////////////////////////////////
```

```
function GetFreeSpace(Disk : String) : Int64;
var
  TotalBytes     : Int64;
  TotalFreeBytes : PLargeInteger;
  FreeBytesCall  : Int64;

begin
  New(TotalFreeBytes);
  try
    GetDiskFreeSpaceEx(PChar(Disk), FreeBytesCall, TotalBytes, TotalFreeBytes);;
    Result := TotalFreeBytes^;
  finally
    Dispose(TotalFreeBytes);

  end;
end;

////////////////////////////////////////////////////////////////////////////////

//Запуск завантаження

function IsRun : Boolean;
var
  Mutex : integer;

begin
  Result := False;
  Mutex := CreateMutex(nil , True, 'BankClientServer');
  if GetLastError <> 0 then
  begin
    CloseHandle(Mutex);
    Result := True;
  end;
end;

////////////////////////////////////////////////////////////////////////////////

//Рядок з часом

function GetTimeStr(Secs : Integer) : String;

  function LeadingZero(N:Integer) : String;
  begin
    if N < 10 then Result := '0' + IntToStr(N) else Result := IntToStr(N);

  end;

var
  Hours, Mins : Integer;
begin
  Hours := Secs div 3600;
  Secs  := Secs - Hours * 3600;
  Mins  := Secs div 60;
  Secs  := Secs - Mins * 60;

  Result := LeadingZero(Hours) + ':' + LeadingZero(Mins) + ':' +
LeadingZero(Secs);
end;
end.
```

BitTorrent_Main.pas - головна програма менеджера завантажень з використанням протоколу BitTorrent

```
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ToolWin, ComCtrls, ImgList, StdActns, Menus, ExtCtrls, ActnList,
  XPStyleActnCtrls, ActnMan, ActnCtrls, ActnMenus, StdCtrls, XPMan, Clipbrd,
  DateUtils, ShellAPI;

type
  TfMain = class(TForm)
    ilMenu: TImageList;
    ActionMainMenuBar1: TActionMainMenuBar;
    ActionManager1: TActionManager;
    actExit: TAction;
    actAdd: TAction;
    actOptions: TAction;
    lvTasks: TListView;
    StatusBar: TStatusBar;
    actStatusBar: TAction;
    actInfo: TAction;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Splitter1: TSplitter;
    tvFolders: TTreeView;
    Splitter2: TSplitter;
    Panel2: TPanel;
    lvParams: TListView;
    XPManifest1: TXPManifest;
    ilTasks: TImageList;
    actLoad: TAction;
    actDelete: TAction;
    ilTree: TImageList;
    actParams: TAction;
    actStop: TAction;
    ilTray: TImageList;
    actClearDel: TAction;
    actEdit: TAction;
    actAbout: TAction;
    actAddCategory: TAction;
    actDeleteCategory: TAction;
    actEditCategory: TAction;
    pmTasks: TPopupMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N5: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    pmTree: TPopupMenu;
    N9: TMenuItem;
    N10: TMenuItem;
    N11: TMenuItem;
    N12: TMenuItem;
    N14: TMenuItem;
    tmUpdate: TTimer;
    N15: TMenuItem;
    N16: TMenuItem;
    ActionToolBar1: TActionToolBar;
    pmTray: TPopupMenu;
    N17: TMenuItem;
    N18: TMenuItem;
```

```
    N20: TMenuItem;
    N21: TMenuItem;
    N22: TMenuItem;
    N19: TMenuItem;
    N23: TMenuItem;
    N24: TMenuItem;
    actOpenFolder: TAction;
    N4: TMenuItem;
    N6: TMenuItem;
    actOpenFile: TAction;
    N13: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure actExitExecute(Sender: TObject);
    procedure actAddExecute(Sender: TObject);
    procedure actOptionsExecute(Sender: TObject);
    procedure actStatusBarExecute(Sender: TObject);
    procedure actInfoExecute(Sender: TObject);
    procedure actLoadExecute(Sender: TObject);
    procedure actDeleteExecute(Sender: TObject);
    procedure lvParamsCustomDrawItem(Sender: TCustomListView; Item: TListItem;
State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure actParamsExecute(Sender: TObject);
    procedure lvTasksSelectItem(Sender: TObject; Item: TListItem; Selected:
Boolean);
    procedure tvFoldersChange(Sender: TObject; Node: TTreeNode);
    procedure actStopExecute(Sender: TObject);
    procedure TrayIconClick(Sender: TObject);
    procedure lvTasksCustomDrawItem(Sender: TCustomListView; Item: TListItem;
State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure actClearDelExecute(Sender: TObject);
    procedure actEditExecute(Sender: TObject);
    procedure actAboutExecute(Sender: TObject);
    procedure Panel2Resize(Sender: TObject);
    procedure Panel1Resize(Sender: TObject);
    procedure actAddCategoryExecute(Sender: TObject);
    procedure lvTasksKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    procedure actDeleteCategoryExecute(Sender: TObject);
    procedure actEditCategoryExecute(Sender: TObject);
    procedure tmUpdateTimer(Sender: TObject);
    procedure N17Click(Sender: TObject);
    procedure actOpenFolderExecute(Sender: TObject);
    procedure actOpenFileExecute(Sender: TObject);
  private
    NextViewerHandle : THandle;
    procedure OnDrawClipboard(var Message : TMessage); Message WM_DRAWCLIPBOARD;
    procedure OnChangeCBCHain(var Message : TMessage); Message WM_CHANGECBCHAIN;
  public
    procedure InsertItem(P : Pointer);
    procedure UpdateItems;
    procedure RefreshTasks;
    procedure SaveTreeNode(Node : TTreeNode);
  end;

var
  fMain: TfMain;

implementation

uses uAddTask, uObjects, uProcedures, uOptions, uLoading, uThreads, uEditTask,
uAbout, uAddCategory, uEditCategory,
  uDownload;

{$R *.dfm}

procedure TfMain.UpdateItems;
var
 Data: TTask;
```

```
 i: integer;
begin
 for i:=0 to lvTasks.Items.Count-1 do
  begin
   Data := lvTasks.Items[i].Data;
   if Data.Status=tsLoading
   then
    begin
     if Data.TotalSize>0
     then lvTasks.Items[i].SubItems[1]:=BytesToText(Data.TotalSize)
     else lvTasks.Items[i].SubItems[1]:='?';
     lvTasks.Items[i].SubItems[2]:=BytesToText(Data.LoadSize);
     if Data.TotalSize>0
     then
lvTasks.Items[i].SubItems[3]:=FloatToStr((Data.LoadSize/Data.TotalSize)*100,
ffFixed, 18, 0)
     else lvTasks.Items[i].SubItems[3]:='?';
     if Data.Speed>0
     then lvTasks.Items[i].SubItems[4]:=BytesToText(Data.Speed)+'/c'
     else lvTasks.Items[i].SubItems[4]:='?';
    end;
  end;
end;

//Створення головної форми

procedure TfMain.FormCreate(Sender: TObject);
begin
 Options:=TOptions.Create;
 Options.Path:=ExtractFileDir(Application.ExeName);
 Options.Version:=GetFileVersion(Application.ExeName);
 Options.Name:='Менеджер завантажень файлів з мережі';
 Options.Load;
 NextViewerHandle:=SetClipboardViewer(Handle);
 Application.Title:=Options.Name+' ';
 Caption:=Options.Name;
 tvFolders.FullExpand;
 actParams.Checked:=Panel2.Visible;
 actStatusBar.Checked:=StatusBar.Visible;
 tvFoldersChange(Self, tvFolders.Selected);
end;

//Закриття голвної форми

procedure TfMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 ChangeClipboardChain(Handle, NextViewerHandle);
 Options.Save;
 Options.Free;
end;

//закінчення роботи

procedure TfMain.actExitExecute(Sender: TObject);
begin
 Close;
end;

//додання нового посилання на навантаження

procedure TfMain.actAddExecute(Sender: TObject);
begin
 fAddTask:=TfAddTask.Create(Application);
 fAddTask.ShowModal;
end;

//Встановлення опцій

procedure TfMain.actOptionsExecute(Sender: TObject);
```

```
begin
 fOptions:=TfOptions.Create(Application);
 fOptions.ShowModal;
end;

//Виведеннястатусбара

procedure TfMain.actStatusBarExecute(Sender: TObject);
begin
 StatusBar.Visible:=actStatusBar.Checked;
end;

//Виведеня інформації про завантаження

procedure TfMain.actInfoExecute(Sender: TObject);
var
 ThreadHttp: TGetOptionsHttp;
 ThreadFtp: TGetOptionsFtp;
begin
 if lvTasks.SelCount=0
 then Exit;
 if (TTask(lvTasks.Selected.Data).Protocol=ptHttp)
       or (TTask(lvTasks.Selected.Data).Protocol=ptHttps)
 then
  begin
   ThreadHttp:=TGetOptionsHttp.Create(true, lvTasks.Selected.Data);
   ThreadHttp.Priority:=Options.Priority;
   ThreadHttp.FreeOnTerminate:=true;
   ThreadHttp.Resume;
  end;
 if TTask(lvTasks.Selected.Data).Protocol=ptFtp
 then
  begin
   ThreadFtp:=TGetOptionsFtp.Create(true, lvTasks.Selected.Data);
   ThreadFtp.Priority:=Options.Priority;
   ThreadFtp.FreeOnTerminate:=true;
   ThreadFtp.Resume;
  end;
end;

//Редагування завантаження

procedure TfMain.actLoadExecute(Sender: TObject);
var
 LoadOne: TLoadOne;
begin
 if lvTasks.SelCount=0
 then Exit;
 if Options.MinOnRun
 then Application.Minimize;
 if TTask(lvTasks.Selected.Data).Status=tsLoad
 then
  begin
   if MessageBox(Application.Handle, PChar('Файл
"'+TTask(lvTasks.Selected.Data).FileName + '" вже був завантажений. Завантажити
його знову?'), PChar(Options.Name), MB_YESNO or MB_ICONERROR)=IDYES
   then
    begin
     TTask(lvTasks.Selected.Data).LoadSize:=0;
     TTask(lvTasks.Selected.Data).Status:=tsReady;
    end
   else Exit;
  end;
 if Options.ShowLoadingForm
 then
  begin
   fLoading:=TfLoading.Create(Application);
   fLoading.Data:=lvTasks.Selected.Data;
   fLoading.Show;
```

```
  end;
 LoadOne:=TLoadOne.Create(true, lvTasks.Selected.Data);
 LoadOne.FreeOnTerminate:=true;
 LoadOne.Resume;
end;

//Видалення завантаження

procedure TfMain.actDeleteExecute(Sender: TObject);
var
 i: integer;
begin
 if not Assigned(lvTasks.Selected)
 then Exit;
 if MessageBox(Application.Handle, 'Видалити виділені елементи?',
PChar(Options.Name), MB_YESNO or MB_ICONWARNING)=IDYES
 then
  begin
   for i:=0 to lvTasks.Items.Count-1 do
    begin
     if lvTasks.Items[i].Selected
     then
      begin
       if TTask(lvTasks.Selected.Data).Status=tsDeleted
       then TTask(lvTasks.Selected.Data).Status:=tsDelete
       else TTask(lvTasks.Selected.Data).Status:=tsDeleted;
      end;
    end;
  end;
 lvTasks.Selected.Delete;
 lvTasks.Repaint;
end;

//Рисування вікна параметрів

procedure TfMain.lvParamsCustomDrawItem(Sender: TCustomListView;
  Item: TListItem; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
 if Item.Index mod 2=0
 then
  begin
   Sender.Canvas.Font.Color:=clBlack;
   Sender.Canvas.Brush.Color:=$F6F6F6;
  end
 else
  begin
   Sender.Canvas.Font.Color:=clBlack;
   Sender.Canvas.Brush.Color:=clWhite;
  end;
end;

//Виведеня параметрів

procedure TfMain.actParamsExecute(Sender: TObject);
begin
 if ActParams.Checked
 then
  begin
   Panel2.Height:=200;
   Panel1.Height:=Panel4.Height-200;
   Splitter2.Visible:=true;
  end
 else
  begin
   Panel2.Height:=0;
   Panel1.Height:=Panel4.Height;
   Splitter2.Visible:=false;
  end;
end;
```

```
//Вибір вікна у якому потрібно працювати

procedure TfMain.lvTasksSelectItem(Sender: TObject; Item: TListItem; Selected:
Boolean);
var
 Data: TTask;
begin
 if (lvParams.Items[0]=nil) or (Item<>lvTasks.Selected)
 then Exit;
 Data:=Item.Data;
 lvParams.Items[0].SubItems[1]:=Data.LinkToFile;
 lvParams.Items[1].SubItems[1]:=Data.Directory;
 lvParams.Items[2].SubItems[1]:=Data.FileName;
 if Data.TotalSize>0
 then lvParams.Items[3].SubItems[1]:=IntToStr(Data.TotalSize)
 else lvParams.Items[3].SubItems[1]:='?';
 lvParams.Items[4].SubItems[1]:=IntToStr(Data.LoadSize);
 if Data.LastModified<>0
 then lvParams.Items[5].SubItems[1]:=FormatDateTime('dd.mm.yyyy hh:mm:ss',
Data.LastModified)
 else lvParams.Items[5].SubItems[1]:='?';
 if Data.TotalSize>0
 then lvParams.Items[6].SubItems[1]:='Так'
 else lvParams.Items[6].SubItems[1]:='Немає';
 if Data.TimeBegin<>0
 then lvParams.Items[7].SubItems[1]:=FormatDateTime('dd.mm.yyyy hh:mm:ss',
Data.TimeBegin)
 else lvParams.Items[7].SubItems[1]:='?';
 if Data.TimeEnd<>0
 then lvParams.Items[8].SubItems[1]:=FormatDateTime('dd.mm.yyyy hh:mm:ss',
Data.TimeEnd)
 else lvParams.Items[8].SubItems[1]:='?';
 if Data.TimeTotal<>0
 then lvParams.Items[9].SubItems[1]:=FormatDateTime('hh:mm:ss', Data.TimeTotal)
 else lvParams.Items[9].SubItems[1]:='?';
 if Data.Speed>0
 then lvParams.Items[10].SubItems[1]:=IntToStr(Data.Speed)
 else lvParams.Items[10].SubItems[1]:='?';
 lvParams.Items[11].SubItems[1]:=Data.ErrorText;
 lvParams.Items[12 ].SubItems[1]:=Data.Description;
end;

//Зміна папки куди записувати завантажений файл

procedure TfMain.tvFoldersChange(Sender: TObject; Node: TTreeNode);
var
 i: integer;
 Data: TTask;
 SortStatus: TTaskStatus;
 SortDate: TDate;
begin
 if (Node.Level=0) or (Node<>tvFolders.Selected)
 then Exit;
 lvTasks.Items.Clear;
 SortStatus:=tsReady;
 SortDate:=0;
 if (Node.Level=1) and (Node.Index=0)
 then
  begin
   for i:=0 to Options.Task.Count-1 do
    begin
     Data:=Options.Task[i];
     if Data.Status=tsLoad
     then InsertItem(Data);
    end;
  end;
 if (Node.Level=1) and (Node.Index=1)
 then
```

```
 begin
  for i:=0 to Options.Task.Count-1 do
   begin
    Data:=Options.Task[i];
    if (Data.Status<>tsDelete) and (Data.Status<>tsDeleted)
    then InsertItem(Data);
   end;
 end;
if (Node.Parent.Level=1) and (Node.Parent.Index=1)
then
 begin
  case Node.Index of
   0: SortStatus:=tsReady;
   1: SortStatus:=tsLoad;
   2: SortStatus:=tsLoading;
   3: SortStatus:=tsStoped;
   4: SortStatus:=tsError;
  end;
 for i:=0 to Options.Task.Count-1 do
  begin
   Data:=Options.Task[i];
   if Data.Status=SortStatus
   then InsertItem(Data);
  end;
 end;
if (Node.Level=1) and (Node.Index=3)
then
 begin
  for i:=0 to Options.Task.Count-1 do
   begin
    Data:=Options.Task[i];
    if Data.Status=tsDeleted
    then InsertItem(Data);
   end;
 end;
if (Node.Parent.Level=1) and (Node.Parent.Index=2)
then
 begin
  if (Node.Index=0) or (Node.Index=1)
  then
   begin
    case Node.Index of
      0: SortDate:=Date;
      1: SortDate:= Date-1;
    end;
  for i:=0 to Options.Task.Count-1 do
   begin
    Data:=Options.Task[i];
    if (SameDate(Data.TimeEnd, SortDate)) and (Data.Status=tsLoad)
    then InsertItem(Data);
   end;
 end;
if (Node.Index=2) or (Node.Index=3)
then
 begin
  case Node.Index of
    2: SortDate:= Now-8;
    3: SortDate:= Now-31;
  end;
  for i:=0 to Options.Task.Count-1 do
   begin
    Data:=Options.Task[i];
    if (Data.TimeEnd>SortDate) and (Data.Status=tsLoad)
    then InsertItem(Data);
   end;
  end;
 end;
end;
```

```
//Остановка завантаження

procedure TfMain.actStopExecute(Sender: TObject);
begin
 if not Assigned(lvTasks.Selected)
 then Exit;
 if TTask(lvTasks.Selected.Data).Status=tsLoading
 then TTask(lvTasks.Selected.Data).Status:=tsStoped;
end;

//Виведення вікна кліпбоарда

procedure TfMain.OnDrawClipboard(var Message : TMessage);
begin
 if Options.HookClipboard
 then
  begin
   if Clipboard.HasFormat(CF_TEXT)
   then
    begin
     if (Pos('http://', Trim(Clipboard.AsText))=1)
      or (Pos('https://', Trim(Clipboard.AsText))=1)
      or (Pos('ftp://', Trim(Clipboard.AsText))=1)
     then
      begin
       SetForegroundWindow(fMain.Handle);
       fAddTask:=TfAddTask.Create(Application);
       fAddTask.ShowModal;
      end;
    end;
  end;
 Message.Result:=SendMessage(WM_DRAWCLIPBOARD, NextViewerHandle, 0, 0);
end;

procedure TfMain.OnChangeCBCHain(var Message: TMessage);
begin
 if Message.wParam=Integer(NextViewerHandle)
 then
  begin
   NextViewerHandle:=Message.lParam;
   Message.Result:=0;
  end
 else Message.Result:=SendMessage(NextViewerHandle, WM_CHANGECBCHAIN,
Message.wParam, Message.lParam);
end;

//Звертання системи у трей

procedure TfMain.TrayIconClick(Sender: TObject);
begin
 SetForegroundWindow(Handle);
end;

//вибір вікна поточних завдань

procedure TfMain.lvTasksCustomDrawItem(Sender: TCustomListView;
  Item: TListItem; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
 if Item.Index mod 2=0
 then
  begin
   Sender.Canvas.Font.Color:=clBlack;
   Sender.Canvas.Brush.Color:=$F6F6F6;
  end
 else
  begin
   Sender.Canvas.Font.Color:=clBlack;
   Sender.Canvas.Brush.Color:=clWhite;
```

```
  end;
end;

//Очищення вікна завантаження

procedure TfMain.actClearDelExecute(Sender: TObject);
var
 i: integer;
begin
 if MessageBox(Application.Handle, 'Ви дійсно хочете очистити папку
"Вилучені"?', PChar(Options.Name), MB_OKCANCEL or MB_ICONWARNING)=ID_OK
 then
  begin
   for i:=0 to Options.Task.Count-1 do
    begin
     if TTask(Options.Task[i]).Status=tsDeleted
     then TTask(Options.Task[i]).Status:=tsDelete;
    end;
   lvTasks.Clear;
  end;
end;

//додавання даних про завантаження

procedure TfMain.InsertItem(P : Pointer);
var
 ListItem: TListItem;
 Data: TTask;
begin
 Data:=P;
 ListItem:=lvTasks.Items.Insert(0);
 ListItem.SubItems.Add(Data.LinkToFile);
 if Data.TotalSize>0
 then ListItem.SubItems.Add(BytesToText(Data.TotalSize))
 else ListItem.SubItems.Add('?');
 ListItem.SubItems.Add(BytesToText(Data.LoadSize));
 if Data.TotalSize>0
 then ListItem.SubItems.Add(FloatToStr((Data.LoadSize/Data.TotalSize)*100,
ffFixed, 18, 0))
 else ListItem.SubItems.Add('?');
 if Data.Speed>0
 then ListItem.SubItems.Add(BytesToText(Data.Speed)+'/c')
 else ListItem.SubItems.Add('?');
 ListItem.ImageIndex:=1;
 ListItem.Data:=Data;
end;

//Додавання завантаження

procedure TfMain.actEditExecute(Sender: TObject);
begin
 if lvTasks.SelCount=0
 then Exit;
 fEditTask:=TfEditTask.Create(Application);
 fEditTask.Data:=lvTasks.Selected.Data;
 fEditTask.ShowModal;
end;

//Інформація про завантаження

procedure TfMain.actAboutExecute(Sender: TObject);
begin
 fAbout:=TfAbout.Create(Application);
 fAbout.ShowModal;
end;

//Встановлення розмірів вікон

procedure TfMain.Panel2Resize(Sender: TObject);
```

```
var
 i: integer;
begin
 if lvParams.Width>400
 then
  begin
   if lvParams.Height<240
   then i:=40
   else i:=24;
   lvParams.Columns[2].Width:=lvParams.Width-lvParams.Columns[1].Width-i;
  end;
end;

procedure TfMain.Panel1Resize(Sender: TObject);
var
 i: integer;
begin
 if lvTasks.Width>500
 then
  begin
   if lvTasks.Height<lvTasks.Items.Count*16
   then i:=40
   else i:=24;
   lvTasks.Columns[1].Width:=lvTasks.Width-lvTasks.Columns[2].Width -
lvTasks.Columns[3].Width  - lvTasks.Columns[4].Width-lvTasks.Columns[5].Width-i;
  end;
end;

//Додавання категорії завантаження

procedure TfMain.actAddCategoryExecute(Sender: TObject);
begin
 fAddCategory:=TfAddCategory.Create(Application);
 fAddCategory.ShowModal;
end;

procedure TfMain.lvTasksKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
begin
 if Key=VK_DELETE
 then actDeleteExecute(nil);
end;

//Видалення категорії завантаження

procedure TfMain.actDeleteCategoryExecute(Sender: TObject);
begin
 if MessageBox(Application.Handle, PChar('Ви дійсно хочете видалити папку "' +
tvFolders.Selected.Text + '"?'), PChar(Options.Name), MB_OKCANCEL or
MB_ICONWARNING)=ID_OK
 then tvFolders.Selected.Delete;
end;

//Редагування категорії завантаження

procedure TfMain.actEditCategoryExecute(Sender: TObject);
begin
 fEditCategory:=TfEditCategory.Create(Application);
 fEditCategory.ShowModal;
end;

procedure TfMain.tmUpdateTimer(Sender: TObject);
begin
 UpdateItems;
end;

procedure TfMain.N17Click(Sender: TObject);
begin
 SetForegroundWindow(Handle);
```

```
end;

procedure TfMain.RefreshTasks;
var
 Data: TTask;
 i: integer;
begin
 for i:=0 to lvTasks.Items.Count-1 do
  begin
   Data:=lvTasks.Items[i].Data;
   if Data.TotalSize>0
   then lvTasks.Items[i].SubItems[1]:=BytesToText(Data.TotalSize)
   else lvTasks.Items[i].SubItems[1]:='?';
   lvTasks.Items[i].SubItems[2]:=BytesToText(Data.LoadSize);
   if Data.TotalSize>0
   then
lvTasks.Items[i].SubItems[3]:=FloatToStr((Data.LoadSize/Data.TotalSize)*100,
ffFixed, 18, 0)
   else lvTasks.Items[i].SubItems[3]:='?';
   if Data.Speed>0
   then lvTasks.Items[i].SubItems[4]:=BytesToText(Data.Speed)+'/c'
   else lvTasks.Items[i].SubItems[4]:='?';
  end;
end;


procedure TfMain.SaveTreeNode(Node : TTreeNode);
var
 i: integer;
begin
 for i:=0 to Node.Count-1 do
  begin
   ShowMessage(Node.Item[i].Text);
   SaveTreeNode(Node.Item[i]);
  end;
end;

//Відкриття папки завантаження

procedure TfMain.actOpenFolderExecute(Sender: TObject);
begin
 if not Assigned(lvTasks.Selected)
 then Exit;
 ShellExecute(0, 'open',     PChar(TTask(lvTasks.Selected.Data).Directory), '',
'', SW_SHOWNORMAL);
end;



//Відкриття файлу завантаження

procedure TfMain.actOpenFileExecute(Sender: TObject);
begin
 if not Assigned(lvTasks.Selected)
 then Exit;
 ShellExecute(0, 'open',     PChar(TTask(lvTasks.Selected.Data).Directory + '\'
+ TTask(lvTasks.Selected.Data).FileName), '', '', SW_SHOWNORMAL);
end;

end.
```

BitTorrent_Download.pas - модуль завантажень

```
unit uDownload;

interface

uses
  Classes, uThreads, uObjects, uProcedures;

type

// Class TLoadOne //

TLoadOne = class(TThread)
  public
    constructor Create(CreateSuspended : Boolean; P : Pointer);
  private
    Data : TTask;
  protected
    procedure Execute; override;
end;

implementation

uses uMain;

//Створення завантаження

constructor TLoadOne.Create(CreateSuspended : Boolean; P : Pointer);
begin
 Data:=P;
 inherited Create(CreateSuspended);
end;

//Редагування завантаження

procedure TLoadOne.Execute;
var
 ThreadHttp: TGetFileHttp;
 ThreadHttpOpt: TGetOptionsHttp;
 ThreadFtp: TGetFileFtp;
 ThreadFtpOpt: TGetOptionsFtp;
begin
 if (Data.Protocol=ptHttp) or (Data.Protocol=ptHttps)
 then
  begin
   ThreadHttpOpt:=TGetOptionsHttp.Create(True, Data);
   ThreadHttpOpt.Priority:=Options.Priority;
   ThreadHttpOpt.Resume;
   ThreadHttpOpt.WaitFor;
   ThreadHttpOpt.Free;
   if Data.Status=tsError
   then Exit;
   if (Data.TotalSize<0) and not (Options.ResumeLoad)
   then Exit;
   if Data.TotalSize>0
   then
    begin
     if Data.TotalSize>GetFreeSpace(Data.Directory)
     then Exit;
    end;
   ThreadHttp:=TGetFileHttp.Create(true, Data);
   ThreadHttp.Priority:=Options.Priority;
   ThreadHttp.Resume;
   ThreadHttp.WaitFor;
   ThreadHttp.Free;
```

```
  end;

 if Data.Protocol=ptFtp
 then
  begin
   ThreadFtpOpt:=TGetOptionsFtp.Create(True, Data);
   ThreadFtpOpt.Priority:=Options.Priority;
   ThreadFtpOpt.Resume;
   ThreadFtpOpt.WaitFor;
   ThreadFtpOpt.Free;
   if Data.Status=tsError
   then Exit;
   if Data.TotalSize>0
   then
    begin
     if Data.TotalSize>GetFreeSpace(Data.Directory)
     then Exit;
    end;
   ThreadFtp:=TGetFileFtp.Create(True, Data);
   ThreadFtp.Priority:=Options.Priority;
   ThreadFtp.Resume;
   ThreadFtp.WaitFor;
   ThreadFtp.Free;
  end;
 fMain.tmUpdate.Enabled:=false;
 fMain.RefreshTasks;
end;

end.
```

BitTorrent_Loading.pas - візуалізація процесу завантаження

```
unit uLoading;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Gauges, ExtCtrls, uObjects, uProcedures;

type
  TfLoading = class(TForm)
    ProgressBar: TGauge;
    Button1: TButton;
    Button2: TButton;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Timer: TTimer;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure TimerTimer(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Data : TTask;
  end;

var
  fLoading: TfLoading;

implementation


{$R *.dfm}

procedure TfLoading.Button1Click(Sender: TObject);
begin
 Close;
end;

procedure TfLoading.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action := caFree;
end;

procedure TfLoading.TimerTimer(Sender: TObject);
var
 TimeRemind : Integer;
begin
 Label11.Caption := FormatDateTime('hh:mm:ss', Data.TimeBegin - Now);
 if Data.TotalSize > 0 then
  begin
   Label7.Caption := BytesToText(Data.LoadSize) + ' з ' +
BytesToText(Data.TotalSize);
   ProgressBar.MinValue := 0;
   ProgressBar.MaxValue := Data.TotalSize;
   ProgressBar.Progress := Data.LoadSize;
```

```
   if Data.Speed > 0 then
    begin
     TimeRemind := (Data.TotalSize - Data.LoadSize) div Data.Speed;
     Label13.Caption := GetTimeStr(TimeRemind);
    end;
   end
  else
   begin
    ProgressBar.MinValue := 0;
    ProgressBar.MaxValue := 8;
    ProgressBar.AddProgress(1);
    if ProgressBar.Progress = 8 then ProgressBar.Progress := 0;
    Label7.Caption := BytesToText(Data.LoadSize);
  end;
 Label9.Caption := BytesToText(Data.Speed) + '/c';
 if (Data.Status = tsLoad) then
  begin
   Timer.Enabled := False;
   ProgressBar.Progress := ProgressBar.MaxValue;
   if Options.AutoCloseLoadingForm then Close;
  end;
 if (Data.Status = tsError) then
  begin
   Timer.Enabled := False;
   ProgressBar.BackColor := clRed;
   if Options.AutoCloseLoadingForm then Close;
  end;
end;

procedure TfLoading.FormShow(Sender: TObject);
begin
 Timer.Enabled := True;
 Label2.Caption := Data.LinkToFile;
end;

procedure TfLoading.Button2Click(Sender: TObject);
begin
 Data.Status := tsStoped;
 Close;
end;

end.
```

BitTorrent_Objects.pas - основні об'єкти

```
unit uObjects;

interface

uses
  Classes, XMLDoc, XMLIntf, SysUtils, Windows, ComCtrls, Menus;

type

// Class TTaskStatus //

TTaskStatus = (tsReady, tsError, tsLoad, tsLoading, tsStoped, tsDelete,
tsDeleted);

// Class THTTPVersion //

THTTPVersion = (hvHttp10, hvHttp11);

// Class TProtocol //

TProtocolType = (ptHttp, ptHttps, ptFtp);

// Class TCategory //

TCategory = record
  Name      : String;
  Directory : String;
  Extension : String;
end;

// Class TProxy //

TProxy = record
  Host     : String;
  Port     : Integer;
  UserName : String;
  Password : String;
  UseProxy : Boolean;
end;

// Class TTask //

TTask = class(TObject)
  public
    LinkToFile    : String;
    FileName      : String;
    Directory     : String;
    TotalSize     : Integer;
    LoadSize      : Integer;
    StartPosition : Integer;
    EndPosition   : Integer;
    Login         : String;
    Password      : String;
    Port          : Integer;
    LastModified  : TDateTime;
    TimeBegin     : TDateTime;
    TimeEnd       : TDateTime;
    TimeTotal     : TDateTime;
    ScheduleOn    : Boolean;
    Speed         : Integer;
    Status        : TTaskStatus;
    Protocol      : TProtocolType;
```

```
    UseSpecial    : Boolean;
    ErrorText     : String;
    Category      : String;
    Description   : String;
end;

// Class TOptions //

TOptions = class(TObject)
  public
    Name      : String;
    Version   : String;
    Path      : String;
    AgentName : String;
    ShowLoadingForm      : Boolean;
    AutoCloseLoadingForm : Boolean;
    HookClipboard        : Boolean;
    RunWithWindows : Boolean;
    MinToTray      : Boolean;
    MinOnRun       : Boolean;
    AlwaysInTray   : Boolean;
    UseProxyLocal  : Boolean;
    ResumeLoad     : Boolean;
    HTTPVersion    : THTTPVersion;
    Redirect  : Boolean;
    Priority  : TThreadPriority;
    HotKey    : TShortCut;
    HTTPProxy : TProxy;
    FTPProxy  : TProxy;
    Url       : TStrings;
    Directory : TStrings;
    Task      : TList;

    procedure Save;
    procedure Load;
end;


var
  Options : TOptions;


implementation


uses uMain;


////////////////////////////////////////////////////////////////////////////
//                              Class 'TOptions'                          //
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

procedure TOptions.Save;
var
  Xml    : TXMLDocument;
  Parent : IXMLNode;
  Child  : IXMLNode;
  Value  : IXMLNode;
  i, n   : Integer;
  Data   : TTask;

begin

  Xml := TXMLDocument.Create(nil);
  Xml.Active := True;
```

```
  if Xml.IsEmptyDoc then Xml.DocumentElement := Xml.CreateElement('XMLOptions',
'');

  Xml.DocumentElement.Attributes['Name'] := Options.Name;
  Xml.DocumentElement.Attributes['Version'] := Options.Version;

  Parent := Xml.DocumentElement.AddChild('Options');
  Child := Parent.AddChild('AgentName');
  Child.Text := AgentName;
  Child := Parent.AddChild('ShowLoadingForm');
  Child.Text := BoolToStr(ShowLoadingForm);
  Child := Parent.AddChild('AutoCloseLoadingForm');
  Child.Text := BoolToStr(AutoCloseLoadingForm);
  Child := Parent.AddChild('HookClipboard');
  Child.Text := BoolToStr(HookClipboard);
  Child := Parent.AddChild('MinToTray');
  Child.Text := BoolToStr(MinToTray);
  Child := Parent.AddChild('AlwaysInTray');
  Child.Text := BoolToStr(AlwaysInTray);
  Child := Parent.AddChild('MinOnRun');
  Child.Text := BoolToStr(MinOnRun);
  Child := Parent.AddChild('RunWithWindows');
  Child.Text := BoolToStr(RunWithWindows);
  Child := Parent.AddChild('UseProxyLocal');
  Child.Text := BoolToStr(UseProxyLocal);
  Child := Parent.AddChild('Redirect');
  Child.Text := BoolToStr(Redirect);
  Child := Parent.AddChild('ThreadPriority');
  Child.Text := IntToStr(Integer(Priority));
  Child := Parent.AddChild('HotKey');
  Child.Text := ShortCutToText(HotKey);
  Child := Parent.AddChild('HTTPVersion');
  Child.Text := IntToStr(Integer(HTTPVersion));
  Child := Parent.AddChild('ResumeLoad');
  Child.Text := BoolToStr(ResumeLoad);

  Parent := Xml.DocumentElement.AddChild('MainForm');
  Child := Parent.AddChild('MainWindowTop');
  Child.Text := IntToStr(fMain.Top);
  Child := Parent.AddChild('MainWindowLeft');
  Child.Text := IntToStr(fMain.Left);
  Child := Parent.AddChild('MainWindowHeight');
  Child.Text := IntToStr(fMain.Height);
  Child := Parent.AddChild('MainWindowWidth');
  Child.Text := IntToStr(fMain.Width);
  Child := Parent.AddChild('CategoryPanelWidth');
  Child.Text := IntToStr(fMain.Panel3.Width);
  Child := Parent.AddChild('TasksPanelHeight');
  Child.Text := IntToStr(fMain.Panel1.Height);
  Child := Parent.AddChild('ParamsVisible');
  Child.Text := BoolToStr(fMain.actParams.Checked);
  Child := Parent.AddChild('StatusBarVisible');
  Child.Text := BoolToStr(fMain.actStatusBar.Checked);
  Child := Parent.AddChild('TasksColumn1Width');
  Child.Text := IntToStr(fMain.lvTasks.Columns[1].Width);
  Child := Parent.AddChild('TasksColumn2Width');
  Child.Text := IntToStr(fMain.lvTasks.Columns[2].Width);
  Child := Parent.AddChild('TasksColumn3Width');
  Child.Text := IntToStr(fMain.lvTasks.Columns[3].Width);
  Child := Parent.AddChild('TasksColumn4Width');
  Child.Text := IntToStr(fMain.lvTasks.Columns[4].Width);
  Child := Parent.AddChild('TasksColumn5Width');
  Child.Text := IntToStr(fMain.lvTasks.Columns[5].Width);
  Child := Parent.AddChild('ParamsColumn1Width');
  Child.Text := IntToStr(fMain.lvParams.Columns[1].Width);
  Child := Parent.AddChild('ParamsColumn2Width');
  Child.Text := IntToStr(fMain.lvParams.Columns[2].Width);
  Child := Parent.AddChild('SelectedTreeItem');
```

```
  if Assigned(fMain.tvFolders.Selected) then Child.Text :=
IntToStr(fMain.tvFolders.Selected.AbsoluteIndex) else Child.Text := '0';

  Parent := Xml.DocumentElement.AddChild('Proxy');

  Child := Parent.AddChild('HTTPProxy');

  Value := Child.AddChild('Host');
  Value.Text := HTTPProxy.Host;
  Value := Child.AddChild('Port');
  Value.Text := IntToStr(HTTPProxy.Port);
  Value := Child.AddChild('User');
  Value.Text := HTTPProxy.UserName;
  Value := Child.AddChild('Password');
  Value.Text := HTTPProxy.Password;
  Value := Child.AddChild('UseProxy');
  Value.Text := BoolToStr(HTTPProxy.UseProxy);

  Child := Parent.AddChild('FTPProxy');

  Value := Child.AddChild('Host');
  Value.Text := FTPProxy.Host;
  Value := Child.AddChild('Port');
  Value.Text := IntToStr(FTPProxy.Port);
  Value := Child.AddChild('User');
  Value.Text := FTPProxy.UserName;
  Value := Child.AddChild('Password');
  Value.Text := FTPProxy.Password;
  Value := Child.AddChild('UseProxy');
  Value.Text := BoolToStr(FTPProxy.UseProxy);

  if Url <> nil then
  begin

    Parent := Xml.DocumentElement.AddChild('Url');

    if Url.Count > 15 then n := 15 else n := Url.Count - 1;

    for i := 0 to n do
    begin

      Child := Parent.AddChild('Url');
      Child.Text := Url.Strings[i];

    end;

  end;

  if Directory <> nil then
  begin

    Parent := Xml.DocumentElement.AddChild('Directory');

    if Directory.Count > 15 then n := 15 else n := Directory.Count - 1;

    for i := 0 to n do
    begin

      Child := Parent.AddChild('Directory');
      Child.Text:= Directory.Strings[i];

    end;

  end;

  if Task <> nil then
  begin

    Parent := Xml.DocumentElement.AddChild('Tasks');
```

```
    for i := 0 to Task.Count - 1 do
    begin

      Data := Task.Items[i];

      if Data.Status <> tsDelete then
      begin

        Child := Parent.AddChild('Task');

        Value := Child.AddChild('LinkToFile');
        Value.Text := Data.LinkToFile;
        Value := Child.AddChild('FileName');
        Value.Text := Data.FileName;
        Value := Child.AddChild('Directory');
        Value.Text := Data.Directory;
        Value := Child.AddChild('TotalSize');
        Value.Text := IntToStr(Data.TotalSize);
        Value := Child.AddChild('LoadSize');
        Value.Text := IntToStr(Data.LoadSize);
        Value := Child.AddChild('StartPosition');
        Value.Text := IntToStr(Data.StartPosition);
        Value := Child.AddChild('EndPosition');
        Value.Text := IntToStr(Data.EndPosition);
        Value := Child.AddChild('Login');
        Value.Text := Data.Login;
        Value := Child.AddChild('Password');
        Value.Text := Data.Password;
        Value := Child.AddChild('Port');
        Value.Text := IntToStr(Data.Port);
        Value := Child.AddChild('LastModified');
        Value.Text := DateTimeToStr(Data.LastModified);
        Value := Child.AddChild('TimeBegin');
        Value.Text := DateTimeToStr(Data.TimeBegin);
        Value := Child.AddChild('TimeEnd');
        Value.Text := DateTimeToStr(Data.TimeEnd);
        Value := Child.AddChild('TimeTotal');
        Value.Text := DateTimeToStr(Data.TimeTotal);
        Value := Child.AddChild('ScheduleOn');
        Value.Text := BoolToStr(Data.ScheduleOn);
        Value := Child.AddChild('Speed');
        Value.Text := IntToStr(Data.Speed);
        Value := Child.AddChild('Status');
        Value.Text := IntToStr(Integer(Data.Status));
        Value := Child.AddChild('Protocol');
        Value.Text := IntToStr(Integer(Data.Protocol));
        Value := Child.AddChild('UseSpecial');
        Value.Text := BoolToStr(Data.UseSpecial);
        Value := Child.AddChild('ErrorText');
        Value.Text := Data.ErrorText;
        Value := Child.AddChild('Category');
        Value.Text := Data.Category;
        Value := Child.AddChild('Description');
        Value.Text := Data.Description;

      end;

    end;

  end;

  Xml.SaveToFile(Options.Path + '\' + Options.Name + '.xml');
  Xml.Free;

end;

////////////////////////////////////////////////////////////////////////////////
```

```
procedure TOptions.Load;
var
  Xml    : IXMLDocument;
  Parent : IXMLNode;
  Child  : IXMLNode;
  Value  : IXMLNode;
  i      : Integer;
  Data   : TTask;

begin

  Url := TStringList.Create;
  Directory := TStringList.Create;
  Task := TList.Create;

  if not FileExists(Options.Path + '\' + Options.Name + '.xml') then Exit;

  Xml := TXMLDocument.Create(nil);
  Xml.Active := True;
  Xml.LoadFromFile(Options.Path + '\' + Options.Name + '.xml');

  Parent := Xml.DocumentElement.ChildNodes['Options'];
  Child := Parent.ChildNodes['AgentName'];
  AgentName := Child.Text;
  Child := Parent.ChildNodes['ShowLoadingForm'];
  ShowLoadingForm := StrToBool(Child.Text);
  Child := Parent.ChildNodes['AutoCloseLoadingForm'];
  AutoCloseLoadingForm := StrToBool(Child.Text);
  Child := Parent.ChildNodes['HookClipboard'];
  HookClipboard := StrToBool(Child.Text);
  Child := Parent.ChildNodes['MinToTray'];
  MinToTray := StrToBool(Child.Text);
  Child := Parent.ChildNodes['MinOnRun'];
  MinOnRun := StrToBool(Child.Text);
  Child := Parent.ChildNodes['AlwaysInTray'];
  AlwaysInTray := StrToBool(Child.Text);
  Child := Parent.ChildNodes['RunWithWindows'];
  RunWithWindows := StrToBool(Child.Text);
  Child := Parent.ChildNodes['UseProxyLocal'];
  UseProxyLocal := StrToBool(Child.Text);
  Child := Parent.ChildNodes['Redirect'];
  Redirect := StrToBool(Child.Text);
  Child := Parent.ChildNodes['HotKey'];
  HotKey := TextToShortCut(Child.Text);
  Child := Parent.ChildNodes['ThreadPriority'];
  Priority := TThreadPriority(StrToInt(Child.Text));
  Child := Parent.ChildNodes['HTTPVersion'];
  HTTPVersion := THTTPVersion(StrToInt(Child.Text));
  Child := Parent.ChildNodes['ResumeLoad'];
  ResumeLoad := StrToBool(Child.Text);

  Parent := Xml.DocumentElement.ChildNodes['MainForm'];
  Child := Parent.ChildNodes['MainWindowTop'];
  fMain.Top := StrToInt(Child.Text);
  Child := Parent.ChildNodes['MainWindowLeft'];
  fMain.Left := StrToInt(Child.Text);
  Child := Parent.ChildNodes['MainWindowHeight'];
  fMain.Height := StrToInt(Child.Text);
  Child := Parent.ChildNodes['MainWindowWidth'];
  fMain.Width := StrToInt(Child.Text);
  Child := Parent.ChildNodes['CategoryPanelWidth'];
  fMain.Panel3.Width := StrToInt(Child.Text);
  Child := Parent.ChildNodes['TasksPanelHeight'];
  fMain.Panel1.Height := StrToInt(Child.Text);
  Child := Parent.ChildNodes['ParamsVisible'];
  fMain.Panel3.Visible := StrToBool(Child.Text);
  Child := Parent.ChildNodes['StatusBarVisible'];
  fMain.StatusBar.Visible := StrToBool(Child.Text);
  Child := Parent.ChildNodes['TasksColumn1Width'];
```

```
fMain.lvTasks.Columns[1].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['TasksColumn2Width'];
fMain.lvTasks.Columns[2].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['TasksColumn3Width'];
fMain.lvTasks.Columns[3].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['TasksColumn4Width'];
fMain.lvTasks.Columns[4].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['TasksColumn5Width'];
fMain.lvTasks.Columns[5].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['ParamsColumn1Width'];
fMain.lvParams.Columns[1].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['ParamsColumn2Width'];
fMain.lvParams.Columns[2].Width := StrToInt(Child.Text);
Child := Parent.ChildNodes['SelectedTreeItem'];
fMain.tvFolders.Items[StrToInt(Child.Text)].Selected := True;

Parent := Xml.DocumentElement.ChildNodes['Proxy'];

Child := Parent.ChildNodes[0];

Value := Child.ChildNodes['Host'];
HTTPProxy.Host := Value.Text;
Value := Child.ChildNodes['Port'];
HTTPProxy.Port := StrToInt(Value.Text);
Value := Child.ChildNodes['User'];
HTTPProxy.UserName := Value.Text;
Value := Child.ChildNodes['Password'];
HTTPProxy.Password := Value.Text;
Value := Child.ChildNodes['UseProxy'];
HTTPProxy.UseProxy := StrToBool(Value.Text);

Child := Parent.ChildNodes[1];

Value := Child.ChildNodes['Host'];
FTPProxy.Host := Value.Text;
Value := Child.ChildNodes['Port'];
FTPProxy.Port := StrToInt(Value.Text);
Value := Child.ChildNodes['User'];
FTPProxy.UserName := Value.Text;
Value := Child.ChildNodes['Password'];
FTPProxy.Password := Value.Text;
Value := Child.ChildNodes['UseProxy'];
FTPProxy.UseProxy := StrToBool(Value.Text);

Parent := Xml.DocumentElement.ChildNodes['Url'];

for i := 0 to Parent.ChildNodes.Count - 1 do
begin

  Child := Parent.ChildNodes[i];
  Url.Add(Child.Text);

end;

Parent := Xml.DocumentElement.ChildNodes['Directory'];

for i := 0 to Parent.ChildNodes.Count - 1 do
begin

  Child := Parent.ChildNodes[i];
  Directory.Add(Child.Text);

end;

Parent := Xml.DocumentElement.ChildNodes['Tasks'];

for i := 0 to Parent.ChildNodes.Count - 1 do
begin
```

```
    Data := TTask.Create;

    Child := Parent.ChildNodes[i];
    Value := Child.ChildNodes['LinkToFile'];
    Data.LinkToFile := Value.Text;
    Value := Child.ChildNodes['FileName'];
    Data.FileName := Value.Text;
    Value := Child.ChildNodes['Directory'];
    Data.Directory := Value.Text;
    Value := Child.ChildNodes['TotalSize'];
    Data.TotalSize := StrToInt(Value.Text);
    Value := Child.ChildNodes['LoadSize'];
    Data.LoadSize := StrToInt(Value.Text);
    Value := Child.ChildNodes['StartPosition'];
    Data.StartPosition := StrToInt(Value.Text);
    Value := Child.ChildNodes['EndPosition'];
    Data.EndPosition := StrToInt(Value.Text);
    Value := Child.ChildNodes['Login'];
    Data.Login := Value.Text;
    Value := Child.ChildNodes['Password'];
    Data.Password := Value.Text;
    Value := Child.ChildNodes['Port'];
    Data.Port := StrToInt(Value.Text);
    Value := Child.ChildNodes['LastModified'];
    Data.LastModified := StrToDateTime(Value.Text);
    Value := Child.ChildNodes['TimeBegin'];
    Data.TimeBegin := StrToDateTime(Value.Text);
    Value := Child.ChildNodes['TimeEnd'];
    Data.TimeEnd := StrToDateTime(Value.Text);
    Value := Child.ChildNodes['TimeTotal'];
    Data.TimeTotal := StrToDateTime(Value.Text);
    Value := Child.ChildNodes['ScheduleOn'];
    Data.ScheduleOn := StrToBool(Value.Text);
    Value := Child.ChildNodes['Speed'];
    Data.Speed := StrToInt(Value.Text);
    Value := Child.ChildNodes['Status'];
    Data.Status := TTaskStatus(StrToInt(Value.Text));
    Value := Child.ChildNodes['Protocol'];
    Data.Protocol := TProtocolType(StrToInt(Value.Text));
    Value := Child.ChildNodes['UseSpecial'];
    Data.UseSpecial := StrToBool(Value.Text);
    Value := Child.ChildNodes['ErrorText'];
    Data.ErrorText := Value.Text;
    Value := Child.ChildNodes['Category'];
    Data.Category := Value.Text;
    Value := Child.ChildNodes['Description'];
    Data.Description := Value.Text;

    Task.Add(Data);

  end;

  Xml := nil;

end;

end.
```

BitTorrent_AddCategory.pas - створення нової категорії

```
unit uAddCategory;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  TfAddCategory = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    edName: TEdit;
    edPath: TEdit;
    Label3: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private

  public
    { Public declarations }
  end;

var
  fAddCategory: TfAddCategory;

implementation

uses uProcedures, uMain, uObjects;

{$R *.dfm}

procedure TfAddCategory.Button3Click(Sender: TObject);
begin
 Close;
end;

procedure TfAddCategory.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfAddCategory.Button1Click(Sender: TObject);
begin
 edPath.Text:=BrowserFolder(Handle);
end;

procedure TfAddCategory.Button2Click(Sender: TObject);
var
 Node: TTreeNode;
begin
 if Trim(edName.Text)=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказана назва категорії!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
```

```
 if Trim(edPath.Text)=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказане розміщення папки!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if not DirectoryExists(Trim(edPath.Text))
 then
  begin
   if MessageBox(Application.Handle, PChar('Папки "'+edPath.Text+'" не існує!
Створити?'), PChar(Options.Name), MB_OKCANCEL or MB_ICONERROR)=ID_OK
   then
    begin
     MkDir(Trim(edPath.Text));
    end
   else Exit;
  end;
 Node:=fMain.tvFolders.Items.AddChild(fMain.tvFolders.Selected, edName.Text);
 Node.ImageIndex:=2;
 Node.SelectedIndex:=2;
 Close;
end;

end.
```

BitTorrent_EditCategory.pas - редагування категорії

```
unit uEditCategory;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  TfEditCategory = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    edName: TEdit;
    edPath: TEdit;
    Label3: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private

  public
    { Public declarations }
  end;

var
  fEditCategory: TfEditCategory;

implementation

uses uProcedures, uMain, uObjects;

{$R *.dfm}

procedure TfEditCategory.Button3Click(Sender: TObject);
begin
 Close;
end;

procedure TfEditCategory.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfEditCategory.Button1Click(Sender: TObject);
begin
 edPath.Text:=BrowserFolder(Handle);
end;

procedure TfEditCategory.Button2Click(Sender: TObject);
begin
 if Trim(edName.Text)=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказана назва категорії!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if Trim(edPath.Text)=''
```

```
 then
  begin
   MessageBox(Application.Handle, 'Не вказане розміщення папки!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if not DirectoryExists(Trim(edPath.Text))
 then
  begin
   if MessageBox(Application.Handle, PChar('Папки "' + edPath.Text + '" не
існує! Створити?'), PChar(Options.Name), MB_OKCANCEL or MB_ICONERROR)=ID_OK
   then
    begin
     MkDir(Trim(edPath.Text));
    end
   else Exit;
  end;
 fMain.tvFolders.Selected.Text := Trim(edName.Text);
 Close;
end;

procedure TfEditCategory.FormCreate(Sender: TObject);
begin
 edName.Text:=fMain.tvFolders.Selected.Text;
end;

end.
```

BitTorrent_AddTask.pas - Створення задачі

```
unit uAddTask;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  StdCtrls, Clipbrd, Mask, Spin, Dialogs;

type
  TfAddTask = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    btnOK: TButton;
    btnCancel: TButton;
    Label3: TLabel;
    Label4: TLabel;
    edLogin: TEdit;
    edPassword: TEdit;
    cbxSpecial: TCheckBox;
    Label5: TLabel;
    Label9: TLabel;
    cbCategory: TComboBox;
    cbDirectory: TComboBox;
    Label10: TLabel;
    edFileName: TEdit;
    cbUrl: TComboBox;
    Label8: TLabel;
    edDescription: TEdit;
    sePort: TSpinEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnCancelClick(Sender: TObject);
    procedure cbxSpecialClick(Sender: TObject);
    procedure cbDirectoryClick(Sender: TObject);
    procedure cbDirectoryDropDown(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure cbUrlChange(Sender: TObject);
    procedure btnOKClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fAddTask: TfAddTask;

implementation

uses uObjects, uMain, uProcedures;

{$R *.dfm}

procedure TfAddTask.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfAddTask.btnCancelClick(Sender: TObject);
begin
 Close;
```

```
end;

procedure TfAddTask.cbxSpecialClick(Sender: TObject);
begin
 if cbxSpecial.Checked
 then
  begin
   edLogin.Enabled:=true;
   edPassword.Enabled:=true;
   sePort.Enabled:=true;
  end
 else
  begin
   edLogin.Enabled:=false;
   edPassword.Enabled:=false;
   sePort.Enabled:=false;
  end;
end;

procedure TfAddTask.cbDirectoryClick(Sender: TObject);
begin
 if cbDirectory.ItemIndex=0
 then
  begin
   cbDirectory.Items.Strings[0]:=BrowserFolder(Handle);
   cbDirectory.ItemIndex:=0;
  end;
end;

procedure TfAddTask.cbDirectoryDropDown(Sender: TObject);
begin
 cbDirectory.Items.Strings[0]:='Огляд папок...';
end;

procedure TfAddTask.FormCreate(Sender: TObject);
var
 ClipboardTemp: string;
 i: Integer;
begin
 if Clipboard.HasFormat(CF_TEXT)
 then
  begin
   ClipboardTemp:=Trim(Clipboard.AsText);
   if (Pos('http://', ClipboardTemp)=1)
      or (Pos('https://', ClipboardTemp)=1)
      or (Pos('ftp://', ClipboardTemp)=1)
   then cbUrl.Text:=ClipboardTemp;
   if Pos('www.', ClipboardTemp)=1
   then cbUrl.Text:='http://'+ClipboardTemp;
   if Pos('ftp.', ClipboardTemp)=1
   then cbUrl.Text:='ftp://'+ClipboardTemp;
  end;
 edFileName.Text:=CreateFileName(cbUrl.Text);
 cbUrl.Items:=Options.Url;
 sePort.Enabled:=false;
 for i:=0 to Options.Directory.Count-1 do
  begin
   cbDirectory.Items.Insert(1, Options.Directory.Strings[i]);
  end;
end;

procedure TfAddTask.cbUrlChange(Sender: TObject);
begin
 edFileName.Text:=CreateFileName(cbUrl.Text);
end;

procedure TfAddTask.btnOKClick(Sender: TObject);
var
 i, n: integer;
```

```
 Data: TTask;
begin
 if cbUrl.Text=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказане посилання!', PChar(Options.Name),
MB_OK or MB_ICONERROR);
   Exit;
  end;
 if cbDirectory.Text=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказаний каталог для збереження файлу!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if edFileName.Text=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказане ім'я файлу!', PChar(Options.Name),
MB_OK or MB_ICONERROR);
   Exit;
  end;
 if cbCategory.Text=''
 then
  begin
   MessageBox(Application.Handle, 'Не вказана категорія!', PChar(Options.Name),
MB_OK or MB_ICONERROR);
   Exit;
  end;
 if (Pos('http://', Trim(cbUrl.Text))=1)
  or (Pos('https://', Trim(cbUrl.Text))=1)
  or (Pos('ftp://', Trim(cbUrl.Text))=1)
 then
 else
  begin
   MessageBox(Application.Handle, 'Не вказаний протокол (http://, https://,
ftp://)!', PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if not DirectoryExists(Trim(cbDirectory.Text))
 then
  begin
   if MessageBox(Application.Handle, PChar('Папки "'+cbDirectory.Text + '" не
існує! Створити?'), PChar(Options.Name), MB_OKCANCEL or MB_ICONERROR)=IDOK
   then MkDir(Trim(cbDirectory.Text))
   else Exit;
  end;
 if FileExists(Trim(cbDirectory.Text)+'\'+Trim(edFileName.Text))
 then
  begin
   if MessageBox(Application.Handle, PChar('Файл "'+cbDirectory.Text + '\' +
edFileName.Text + '" вже існує! Хочете перезаписати його?'),
PChar(Options.Name), MB_OKCANCEL or MB_ICONERROR)=IDCANCEL
   then Exit;
  end;
 n:=0;
 for i:=0 to Options.Url.Count-1 do
  begin
   if Options.Url.Strings[I]=Trim(cbUrl.Text)
   then n:=n+1;
  end;
 if n=0
 then Options.Url.Insert(0, Trim(cbUrl.Text));
 n:=0;
 for i:=0 to Options.Directory.Count-1 do
  begin
   if Options.Directory.Strings[i]=Trim(cbDirectory.Text)
   then n:=n+1;
```

```
  end;
 if n=0
 then Options.Directory.Insert(0, Trim(cbDirectory.Text));
 Data:=TTask.Create;
 Data.LinkToFile:=Trim(cbUrl.Text);
 Data.FileName:=Trim(edFileName.Text);
 Data.Directory:=Trim(cbDirectory.Text);
 Data.Login:=Trim(edLogin.Text);
 Data.Password:=Trim(edPassword.Text);
 Data.Port:=sePort.Value;
 Data.UseSpecial:=cbxSpecial.Checked;
 Data.Description:=Trim(edDescription.Text);
 Data.Category:=Trim(edDescription.Text);
 Data.Status:=tsReady;
 Data.TotalSize:=0;
 Data.LoadSize:=0;
 Data.StartPosition:=0;
 Data.EndPosition:=0;
 Data.LastModified:=0;
 Data.TimeBegin:=0;
 Data.TimeEnd:=0;
 Data.TimeTotal:=0;
 Data.ErrorText:='';
 if Pos('http://', Data.LinkToFile)=1
 then Data.Protocol:=ptHttp;
 if Pos('https://', Data.LinkToFile)=1
 then Data.Protocol:=ptHttps;
 if Pos('ftp://', Data.LinkToFile)=1
 then Data.Protocol:=ptFtp;
 Options.Task.Add(Data);
 Close;
end;

end.
```

BitTorrent_EditTask.pas - редагування задачі

```
unit uEditTask;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  StdCtrls, Clipbrd, Mask, Spin, Dialogs, uObjects;

type
  TfEditTask = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    btnOK: TButton;
    btnCancel: TButton;
    Label3: TLabel;
    Label4: TLabel;
    edLogin: TEdit;
    edPassword: TEdit;
    cbxSpecial: TCheckBox;
    Label5: TLabel;
    Label9: TLabel;
    cbCategory: TComboBox;
    cbDirectory: TComboBox;
    Label10: TLabel;
    edFileName: TEdit;
    cbUrl: TComboBox;
    Label8: TLabel;
    edDescription: TEdit;
    sePort: TSpinEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    rbSchedule: TRadioButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnCancelClick(Sender: TObject);
    procedure cbxSpecialClick(Sender: TObject);
    procedure cbDirectoryClick(Sender: TObject);
    procedure cbDirectoryDropDown(Sender: TObject);
    procedure cbUrlChange(Sender: TObject);
    procedure btnOKClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Data : TTask;
  end;

var
  fEditTask: TfEditTask;

implementation

uses uMain, uProcedures;

{$R *.dfm}

procedure TfEditTask.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfEditTask.btnCancelClick(Sender: TObject);
begin
```

```
 Close;
end;

procedure TfEditTask.cbxSpecialClick(Sender: TObject);
begin
 if cbxSpecial.Checked
 then
  begin
   edLogin.Enabled:=true;
   edPassword.Enabled:=true;
   sePort.Enabled:=true;
  end
 else
  begin
   edLogin.Enabled:=false;
   edPassword.Enabled:=false;
   sePort.Enabled:=false;
  end;
end;

procedure TfEditTask.cbDirectoryClick(Sender: TObject);
begin
 if cbDirectory.ItemIndex=0
 then
  begin
   cbDirectory.Items.Strings[0]:=BrowserFolder(Handle);
   cbDirectory.ItemIndex:=0;
  end;
end;

procedure TfEditTask.cbDirectoryDropDown(Sender: TObject);
begin
 cbDirectory.Items.Strings[0]:='Перегляд папок...';
end;

procedure TfEditTask.cbUrlChange(Sender: TObject);
begin
 edFileName.Text:=CreateFileName(cbUrl.Text);
end;

procedure TfEditTask.btnOKClick(Sender: TObject);
var
 i, n: integer;
begin
 if cbUrl.Text=''
 then
  begin
   MessageBox(Application.Handle, 'Не зазначене посилання!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if cbDirectory.Text = '' then
  begin
   MessageBox(Application.Handle, 'Не зазначений каталог для збереження файлу!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if edFileName.Text = '' then
  begin
   MessageBox(Application.Handle, 'Не зазначене ім'я файлу!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if cbCategory.Text = '' then
  begin
   MessageBox(Application.Handle, 'Не зазначена категорія!',
PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
```

```
 if (Pos('http://', Trim(cbUrl.Text)) = 1)
  or (Pos('https://', Trim(cbUrl.Text)) = 1)
  or (Pos('ftp://', Trim(cbUrl.Text)) = 1)
 then
 else
  begin
   MessageBox(Application.Handle, 'Не зазначений протокол (http://, https://,
ftp://)!', PChar(Options.Name), MB_OK or MB_ICONERROR);
   Exit;
  end;
 if not DirectoryExists(Trim(cbDirectory.Text)) then
  begin
    if MessageBox(Application.Handle, PChar('Папки "' + cbDirectory.Text + '" не
існує! Створити?'), PChar(Options.Name), MB_OKCANCEL or MB_ICONWARNING) = IDOK
      then MkDir(Trim(cbDirectory.Text)) else Exit;
  end;
  n := 0;
 for i := 0 to Options.Url.Count - 1 do
  begin
   if Options.Url.Strings[I] = Trim(cbUrl.Text) then n := n + 1;
  end;
  if n = 0 then Options.Url.Insert(0, Trim(cbUrl.Text));
  n := 0;
  for i := 0 to Options.Directory.Count - 1 do
  begin
    if Options.Directory.Strings[i] = Trim(cbDirectory.Text) then n := n + 1;
  end;
  if n = 0 then Options.Directory.Insert(0, Trim(cbDirectory.Text));
  Data.LinkToFile := Trim(cbUrl.Text);
  Data.FileName := Trim(edFileName.Text);
  Data.Directory := Trim(cbDirectory.Text);
  Data.Login := Trim(edLogin.Text);
  Data.Password := Trim(edPassword.Text);
  Data.Port := sePort.Value;
  Data.UseSpecial := cbxSpecial.Checked;
  Data.Description := Trim(edDescription.Text);
  if Pos('http://', Data.LinkToFile) = 1 then Data.Protocol := ptHttp;
  if Pos('https://', Data.LinkToFile) = 1 then Data.Protocol := ptHttps;
  if Pos('ftp://', Data.LinkToFile) = 1 then Data.Protocol := ptFtp;
  Close;
end;

////////////////////////////////////////////////////////////////////////////////

procedure TfEditTask.FormShow(Sender: TObject);
var
 i : Integer;
begin
  cbUrl.Text := Data.LinkToFile;
  edFileName.Text := Data.FileName;
  cbUrl.Items := Options.Url;
 for i := 0 to Options.Directory.Count - 1 do
  begin
   cbDirectory.Items.Insert(1, Options.Directory.Strings[i]);
  end;
  cbDirectory.Text := Data.Directory;
  cbCategory.Text := Data.Category;
  edDescription.Text := Data.Description;
  cbxSpecial.Checked := Data.UseSpecial;
 if Data.UseSpecial
 then
  begin
    edLogin.Enabled := True;
    edPassword.Enabled := True;
    sePort.Enabled := True;
  end
 else
  begin
   edLogin.Enabled := False;
```

```
    edPassword.Enabled := False;
    sePort.Enabled := False;
  end;
  edLogin.Text := Data.Login;
  edPassword.Text := Data.Password;
  sePort.Value := Data.Port;
  if Data.ScheduleOn then rbSchedule.Checked:=true;
end;

end.
```

BitTorrent_Options.pas - параметри програми

```
unit uOptions;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls, Spin, Registry;

type
  TfOptions = class(TForm)
    btnOK: TButton;
    btnCancel: TButton;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Label1: TLabel;
    cbAgent: TComboBox;
    TabSheet3: TTabSheet;
    cbxRunWithWindows: TCheckBox;
    cbxHookClipboard: TCheckBox;
    cbxMinToTray: TCheckBox;
    Label23: TLabel;
    mmExtention: TMemo;
    Label15: TLabel;
    TrackBar: TTrackBar;
    Label16: TLabel;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    cbxAutoCloseLoadingForm: TCheckBox;
    cbxShowLoadingForm: TCheckBox;
    cbxMinOnRun: TCheckBox;
    cbxRedirect: TCheckBox;
    Label6: TLabel;
    hkApplication: THotKey;
    Label7: TLabel;
    cbHttpVersion: TComboBox;
    cbxResumeLoad: TCheckBox;
    TabSheet2: TTabSheet;
    cbxUseProxyLocal: TCheckBox;
    cbUseHTTPProxy: TCheckBox;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    edHTTPProxyHost: TEdit;
    edHTTPProxyPort: TEdit;
    Label4: TLabel;
    Label5: TLabel;
    edHTTPProxyUser: TEdit;
    edHTTPProxyPass: TEdit;
    GroupBox2: TGroupBox;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    edFTPProxyHost: TEdit;
    edFTPProxyPort: TEdit;
    edFTPProxyUser: TEdit;
    edFTPProxyPass: TEdit;
    cbUseFTPProxy: TCheckBox;
    cbxAlwaysInTray: TCheckBox;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnCancelClick(Sender: TObject);
    procedure btnOKClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure TrackBarChange(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fOptions: TfOptions;

implementation

uses uObjects, uMain;

{$R *.dfm}

procedure TfOptions.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfOptions.btnCancelClick(Sender: TObject);
begin
 Close;
end;

procedure TfOptions.btnOKClick(Sender: TObject);
var
 Registry : TRegistry;
begin
  Options.AgentName := Trim(cbAgent.Text);
  Options.ShowLoadingForm := cbxShowLoadingForm.Checked;
  Options.AutoCloseLoadingForm := cbxAutoCloseLoadingForm.Checked;
  Options.HookClipboard := cbxHookClipboard.Checked;
  Options.MinToTray := cbxMinToTray.Checked;
  Options.MinOnRun := cbxMinOnRun.Checked;
  Options.AlwaysInTray := cbxAlwaysInTray.Checked;
  Options.RunWithWindows := cbxRunWithWindows.Checked;
  Options.HotKey := hkApplication.HotKey;
  Options.Redirect := cbxRedirect.Checked;
  Options.UseProxyLocal := cbxUseProxyLocal.Checked;
  Options.ResumeLoad := cbxResumeLoad.Checked;
  Options.HTTPProxy.UseProxy := cbUseHTTPProxy.Checked;
  Options.HTTPProxy.Host := Trim(edHTTPProxyHost.Text);
  Options.HTTPProxy.Port := StrToInt(Trim(edHTTPProxyPort.Text));
  Options.HTTPProxy.UserName := Trim(edHTTPProxyUser.Text);
  Options.HTTPProxy.Password := Trim(edHTTPProxyPass.Text);
  Options.FTPProxy.UseProxy := cbUseFTPProxy.Checked;
  Options.FTPProxy.Host := Trim(edFTPProxyHost.Text);
  Options.FTPProxy.Port := StrToInt(Trim(edFTPProxyPort.Text));
  Options.FTPProxy.UserName := Trim(edFTPProxyUser.Text);
  Options.FTPProxy.Password := Trim(edFTPProxyPass.Text);

  if cbHttpVersion.Text = 'HTTP 1.0' then Options.HTTPVersion := hvHttp10 else
Options.HTTPVersion := hvHttp11;

  case TrackBar.Position of

    0 : Options.Priority := tpIdle;
    1 : Options.Priority := tpLowest;
    2 : Options.Priority := tpLower;
    3 : Options.Priority := tpNormal;
    4 : Options.Priority := tpHigher;
    5 : Options.Priority := tpHighest;
    6 : Options.Priority := tpTimeCritical;

  end;

  Options.Save;

  Registry := TRegistry.Create(KEY_ALL_ACCESS);
```

```
  try

    Registry.RootKey := HKEY_LOCAL_MACHINE;
    Registry.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Run', True);

    if cbxRunWithWindows.Checked then Registry.WriteString('Download',
Application.ExeName)
    else Registry.DeleteValue('Download');

    Registry.CloseKey;

  finally

    Registry.Free;

  end;

  Close;

end;

procedure TfOptions.FormCreate(Sender: TObject);
begin

  cbAgent.Text := Options.AgentName;
  cbxShowLoadingForm.Checked := Options.ShowLoadingForm;
  cbxAutoCloseLoadingForm.Checked := Options.AutoCloseLoadingForm;
  cbxHookClipboard.Checked := Options.HookClipboard;
  cbxMinToTray.Checked := Options.MinToTray;
  cbxMinOnRun.Checked := Options.MinOnRun;
  cbxAlwaysInTray.Checked := Options.AlwaysInTray;
  cbxRunWithWindows.Checked := Options.RunWithWindows;
  hkApplication.HotKey := Options.HotKey;
  cbxRedirect.Checked := Options.Redirect;
  cbxUseProxyLocal.Checked := Options.UseProxyLocal;
  cbxResumeLoad.Checked := Options.ResumeLoad;
  cbUseHTTPProxy.Checked := Options.HTTPProxy.UseProxy;
  edHTTPProxyHost.Text := Options.HTTPProxy.Host;
  edHTTPProxyPort.Text := IntToStr(Options.HTTPProxy.Port);
  edHTTPProxyUser.Text := Options.HTTPProxy.UserName;
  edHTTPProxyPass.Text := Options.HTTPProxy.Password;
  cbUseFTPProxy.Checked := Options.FTPProxy.UseProxy;
  edFTPProxyHost.Text := Options.FTPProxy.Host;
  edFTPProxyPort.Text := IntToStr(Options.FTPProxy.Port);
  edFTPProxyUser.Text := Options.FTPProxy.UserName;
  edFTPProxyPass.Text := Options.FTPProxy.Password;

  if Options.HTTPVersion = hvHttp10 then cbHttpVersion.ItemIndex := 0 else
cbHttpVersion.ItemIndex := 1;

  case Options.Priority of

    tpIdle         : begin TrackBar.Position := 0; Label16.Caption := 'Низький';
end;
    tpLowest       : begin TrackBar.Position := 1; Label16.Caption := 'Нижче
середнього'; end;
    tpLower        : begin TrackBar.Position := 2; Label16.Caption :=
'Середній'; end;
    tpNormal       : begin TrackBar.Position := 3; Label16.Caption := 'Вище за
середнє'; end;
    tpHigher       : begin TrackBar.Position := 4; Label16.Caption := 'Високий';
end;
    tpHighest      : begin TrackBar.Position := 5; Label16.Caption := 'Більш
високий'; end;
    tpTimeCritical : begin TrackBar.Position := 6; Label16.Caption := 'Реального
часу'; end;
  end;
end;
```

```
procedure TfOptions.TrackBarChange(Sender: TObject);
begin
 case TrackBar.Position of
   0: Label16.Caption := 'Низький';
   1: Label16.Caption := 'Нижче середнього';
   2: Label16.Caption := 'Середній';
   3: Label16.Caption := 'Вище за середнє';
   4: Label16.Caption := 'Високий';
   5: Label16.Caption := 'Більш високий';
   6: Label16.Caption := 'Реального часу';
  end;
end;

end.
```

BitTorrent_Threads.pas - темы

```
unit uThreads;

interface

uses
  Classes, IdHTTP, IdFTP, IdComponent, IdFTPCommon, Windows, SysUtils, uObjects,
Forms;

type

// Class TGetFileHttp //

TGetFileHttp = class(TThread)
  public
    Item   : Integer;
    Reload : Boolean;

    constructor Create(CreateSuspended : Boolean; P : Pointer);

  private
    Data      : TTask;
    Tick      : Integer;
    StartSize : Integer;
    HTTP      : TIdHTTP;

    procedure OnWork(Sender : TObject; AWorkMode : TWorkMode; const AWorkCount :
Integer);
    procedure OnWorkBegin(Sender : TObject; AWorkMode : TWorkMode; const
AWorkCountMax : Integer);
    procedure OnWorkEnd(Sender: TObject; AWorkMode: TWorkMode);

  protected
    procedure Execute; override;
end;

// Class TGetOptionsHttp //

TGetOptionsHttp = class(TThread)
  public
    constructor Create(CreateSuspended : Boolean; P : Pointer);

  private
    Data : TTask;

  protected
    procedure Execute; override;
end;

// Class TGetFileFtp //

TGetFileFtp = class(TThread)
  public
    Item   : Integer;
    Reload : Boolean;

    constructor Create(CreateSuspended : Boolean; P : Pointer);

  private
    Data : TTask;
    Tick : Integer;
    FTP  : TIdFTP;

    procedure OnWork(Sender : TObject; AWorkMode : TWorkMode; const AWorkCount :
Integer);
```

```
    procedure OnWorkBegin(Sender : TObject; AWorkMode : TWorkMode; const
AWorkCountMax : Integer);
    procedure OnWorkEnd(Sender: TObject; AWorkMode: TWorkMode);

  protected
    procedure Execute; override;
end;

// Class TGetOptionsFtp //

TGetOptionsFtp = class(TThread)
  public
    constructor Create(CreateSuspended : Boolean; P : Pointer);

  private
    Data : TTask;

  protected
    procedure Execute; override;
end;


implementation


uses uMain, uProcedures;

////////////////////////////////////////////////////////////////////////////
//                              Class 'TGetFileHttp'                      //
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

constructor TGetFileHttp.Create(CreateSuspended : Boolean; P : Pointer);
begin

  Data := P;

  inherited Create(CreateSuspended);

end;

////////////////////////////////////////////////////////////////////////////

procedure TGetFileHttp.Execute;
var
  FileStream : TFileStream;

begin

  HTTP := TIdHTTP.Create(nil);

  if FileExists(Data.Directory + '\' + Data.FileName) then
  begin

    FileStream := TFileStream.Create(Data.Directory + '\' + Data.FileName,
fmOpenReadWrite);
    FileStream.Position := FileStream.Size;

  end else
  begin

    FileStream := TFileStream.Create(Data.Directory + '\' + Data.FileName,
fmCreate);

  end;

  HTTP.OnWork := OnWork;
  HTTP.OnWorkBegin := OnWorkBegin;
```

```
  HTTP.OnWorkEnd := OnWorkEnd;

  if Options.HTTPVersion = hvHttp10 then HTTP.ProtocolVersion := pv1_0 else
HTTP.ProtocolVersion := pv1_1;

  if (Options.HTTPProxy.UseProxy) and (LocalAddress(Data.LinkToFile) = False)
then
  begin

    HTTP.ProxyParams.ProxyServer := Options.HTTPProxy.Host;
    HTTP.ProxyParams.ProxyPort := Options.HTTPProxy.Port;
    HTTP.ProxyParams.ProxyUsername := Options.HTTPProxy.UserName;
    HTTP.ProxyParams.ProxyPassword := Options.HTTPProxy.Password;

  end;

  if Data.UseSpecial then
  begin

    HTTP.Port := Data.Port;

    HTTP.Request.Username := Data.Login;
    HTTP.Request.Password := Data.Password;

  end;

  HTTP.Request.ContentRangeStart := Data.LoadSize;
  HTTP.Request.ContentRangeEnd := Data.TotalSize;

  HTTP.HandleRedirects := Options.Redirect;

  StartSize := Data.LoadSize;

  try

    HTTP.Get(Data.LinkToFile, FileStream);

  except

    on E : Exception do
    begin

      Data.Status := tsError;
      Data.ErrorText := E.Message;

      MessageBox(Application.Handle, PChar('Помилка при завантаженні файлу.' +
#13#10 + E.Message), PChar(Options.Name), MB_OK or MB_ICONERROR);

    end;

  end;

  HTTP.Free;
  FileStream.Free;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileHttp.OnWork(Sender: TObject; AWorkMode: TWorkMode; const
AWorkCount: Integer);
var
  TickCount : Integer;
  Count     : Integer;

begin

  if AWorkMode = wmRead then
  begin
```

```
    Data.LoadSize := StartSize + AWorkCount;

    TickCount := GetTickCount;
    Count := (TickCount - Tick) div 1000;

    if (Data.LoadSize > 0) and (Count > 0) then Data.Speed := Data.LoadSize div
Count;

  end;

  if Data.Status = tsStoped then HTTP.Disconnect;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileHttp.OnWorkBegin(Sender : TObject; AWorkMode : TWorkMode;
const AWorkCountMax : Integer);
begin

  Tick := GetTickCount;

  Data.TimeBegin := Now;
  Data.Status := tsLoading;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileHttp.OnWorkEnd(Sender: TObject; AWorkMode: TWorkMode);
begin

  Data.TimeEnd := Now;
  Data.TimeTotal := Data.TimeBegin - Data.TimeEnd;

  if Data.Status <> tsStoped then Data.Status := tsLoad;

end;

////////////////////////////////////////////////////////////////////////////////
//                              'TGetOptionsHttp'                             //
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////

constructor TGetOptionsHttp.Create(CreateSuspended : Boolean; P : Pointer);
begin

  Data := P;

  inherited Create(CreateSuspended);

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetOptionsHttp.Execute;
var
  HTTP : TIdHTTP;

begin

  HTTP := TIdHTTP.Create(nil);

  if (Options.HTTPProxy.UseProxy) and (LocalAddress(Data.LinkToFile) = False)
then
  begin
```

```
    HTTP.ProxyParams.ProxyServer := Options.HTTPProxy.Host;
    HTTP.ProxyParams.ProxyPort := Options.HTTPProxy.Port;
    HTTP.ProxyParams.ProxyUsername := Options.HTTPProxy.UserName;
    HTTP.ProxyParams.ProxyPassword := Options.HTTPProxy.Password;

  end;

  if Data.UseSpecial then
  begin

    HTTP.Port := Data.Port;
    HTTP.Request.Username := Data.Login;
    HTTP.Request.Password := Data.Password;

  end;

  HTTP.HandleRedirects := Options.Redirect;

  if Options.HTTPVersion = hvHttp10 then HTTP.ProtocolVersion := pv1_0 else
HTTP.ProtocolVersion := pv1_1;

  try

    HTTP.Head(Data.LinkToFile);

    Data.Status := tsReady;
    Data.ErrorText := 'Помилок немає';

    Data.TotalSize := HTTP.Response.ContentLength;
    Data.LastModified := HTTP.Response.LastModified;

  except

    on E : Exception do
    begin

      Data.Status := tsError;
      Data.ErrorText := E.Message;

      Data.TotalSize := 0;
      Data.LastModified := 0;

      MessageBox(Application.Handle, PChar('Помилка при завантаженні файлу.' +
#13#10 + E.Message), PChar(Options.Name), MB_OK or MB_ICONERROR);

    end;

  end;

  HTTP.Free;

  fMain.RefreshTasks;

end;

////////////////////////////////////////////////////////////////////////////////
//                              Class 'TGetFileFtp'                           //
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////

constructor TGetFileFtp.Create(CreateSuspended : Boolean; P : Pointer);
begin

  Data := P;

  inherited Create(CreateSuspended);

end;
```

```
////////////////////////////////////////////////////////////////////////////////

procedure TGetFileFtp.Execute;
begin

  FTP := TIdFTP.Create(nil);

  FTP.OnWork := OnWork;
  FTP.OnWorkBegin := OnWorkBegin;
  FTP.OnWorkEnd := OnWorkEnd;

  if (Options.FTPProxy.UseProxy) and (LocalAddress(Data.LinkToFile) = False)
then
  begin

    FTP.ProxySettings.Host := Options.FTPProxy.Host;
    FTP.ProxySettings.Port := Options.FTPProxy.Port;
    FTP.ProxySettings.Username := Options.FTPProxy.UserName;
    FTP.ProxySettings.Password := Options.FTPProxy.Password;

  end;

  if Data.UseSpecial then
  begin

    FTP.Port := Data.Port;
    FTP.Username := Data.Login;
    FTP.Password := Data.Password;

  end else FTP.Username := 'anonymous';

  try

    FTP.TransferType := ftBinary;
    FTP.Host := ExtractAddress(Data.LinkToFile);
    FTP.Connect;
    FTP.Get(ExtractFileName(Data.LinkToFile), Data.Directory + '\' +
Data.FileName, False, True);
    FTP.Disconnect;

    Data.Status := tsReady;
    Data.ErrorText := 'Помилок немає';

  except

    on E : Exception do
    begin

      Data.Status := tsError;
      Data.ErrorText := E.Message;

      MessageBox(Application.Handle, PChar('Помилка при завантаженні файлу.' +
#13#10 + E.Message), PChar(Options.Name), MB_OK or MB_ICONERROR);

    end;

  end;

  FTP.Free;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileFtp.OnWork(Sender: TObject; AWorkMode: TWorkMode; const
AWorkCount: Integer);
var
  TickCount : Integer;
```

```
  Count     : Integer;

begin

  if AWorkMode = wmRead then
  begin

    Data.LoadSize := AWorkCount;

    TickCount := GetTickCount;
    Count := (TickCount - Tick) div 1000;

    if (Data.LoadSize > 0) and (Count > 0) then Data.Speed := Data.LoadSize div
Count;

  end;

  if Data.Status = tsStoped then FTP.Disconnect;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileFtp.OnWorkBegin(Sender : TObject; AWorkMode : TWorkMode; const
AWorkCountMax : Integer);
begin

  Tick := GetTickCount;

  Data.TimeBegin := Now;
  Data.Status := tsLoading;

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetFileFtp.OnWorkEnd(Sender: TObject; AWorkMode: TWorkMode);
begin

  Data.TimeEnd := Now;
  Data.TimeTotal := Data.TimeBegin - Data.TimeEnd;

  if Data.Status <> tsStoped then Data.Status := tsLoad;

end;

////////////////////////////////////////////////////////////////////////////////
//                              'TGetOptionsFtp'                              //
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////

constructor TGetOptionsFtp.Create(CreateSuspended : Boolean; P : Pointer);
begin

  Data := P;

  inherited Create(CreateSuspended);

end;

////////////////////////////////////////////////////////////////////////////////

procedure TGetOptionsFtp.Execute;
var
  FTP : TIdFTP;
  Details : TStrings;

begin
```

```
  FTP := TIdFTP.Create(nil);
  Details := TStringList.Create;

  if (Options.FTPProxy.UseProxy) and (LocalAddress(Data.LinkToFile) = False)
then
  begin

    FTP.ProxySettings.Host := Options.FTPProxy.Host;
    FTP.ProxySettings.Port := Options.FTPProxy.Port;
    FTP.ProxySettings.Username := Options.FTPProxy.UserName;
    FTP.ProxySettings.Password := Options.FTPProxy.Password;

  end;

  if Data.UseSpecial then
  begin

    FTP.Port := Data.Port;
    FTP.Username := Data.Login;
    FTP.Password := Data.Password;

  end else FTP.Username := 'anonymous';


  try

    FTP.TransferType := ftBinary;
    FTP.Host := ExtractAddress(Data.LinkToFile);
    FTP.Connect;

    FTP.List(Details, ExtractFileName(Data.LinkToFile), True);
    Data.TotalSize := FTP.DirectoryListing.Items[0].Size;
    Data.LastModified := FTP.DirectoryListing.Items[0].ModifiedDate;

    FTP.Disconnect;

    Data.Status := tsReady;
    Data.ErrorText := 'Помилок немає';

  except

    on E : Exception do
    begin

      Data.Status := tsError;
      Data.ErrorText := E.Message;

      Data.TotalSize := 0;
      Data.LastModified := 0;

      MessageBox(Application.Handle, PChar('Помилка при завантаженні файлу.' +
#13#10 + E.Message), PChar(Options.Name), MB_OK or MB_ICONERROR);

    end;

  end;

  FTP.Free;
  Details.Free;

  fMain.RefreshTasks;

end;

end.
```

BitTorrent_Threads.pas - довідка

```
unit uAbout;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TfAbout = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fAbout: TfAbout;

implementation

uses uObjects;
{$R *.dfm}

procedure TfAbout.Button1Click(Sender: TObject);
begin
 Close;
end;

procedure TfAbout.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action:=caFree;
end;

procedure TfAbout.FormCreate(Sender: TObject);
begin
 Label1.Caption:=Options.Name;
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи менеджера
завантажень з використанням протоколу BitTorrent');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник:  Кислун О.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив:  студент Григоращенко Артем Володимирович');
Memo1.Lines.Add('           гр. КІ-22М-2');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Кропивницький 2023');
Memo1.Lines.Add('');
end;
end;
end.
```