ЗМІСТ

ЗМІСТ……...3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ…….5
ВСТУП……..6
1. АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ ……..…9
2. ВИБІР ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТІВ ……10
2.1 Аналіз аналогів.....10
2.1.1 Переваги та недоліки передачі відеопотока за допомогою Wi-Fi.....10
2.1.2 Технології передачі відеопотоку.....11
2.1.3 Безпосередній аналіз мікрокомп'ютерів доступних на ринку.....16
2.1.4 Вибір програмної оболонки та інструментів.....22
2.1.4.1 Вибір мови програмування.....22
2.1.4.2 Вибір інструментів реалізації.....28
2.1.4.2.1 Вибір інтегрованої середи розробки.....28
2.1.4.3 Вибір Бази Даних.....31
Висновки…….38
3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ТА АЛГОРИТМУ 41
3.1 Розробка алгоритму.....41
3.2 Розробка компонентів програми.....43
2.3 Зберігання інформації 48
3.4 Налаштування Raspberry Pi для передачу відеопотока…….....53
3.4.1 Протокол RTP.....53
Висновки…….56
4 АНАЛІЗ ВПЛИВУ НЕГАТИВНИХ ФАКТОРІВ НА ЯКІСТЬ ВІДЕО ПОТОКУ.....57
4.1 Якість передачі відеопотоку.....57
4.1.1 Модель мережі, в якій передається відеопотік.....58
4.1.2 Середовище моделювання.....60
4.1.3 Топологія мережі і доступна швидкість.....62

4.1.4 Організація передачі RTP-пакетів…….....67
4.1.5 Результати моделювання.....68
4.2 Втрати відеопотока у мережі Wi-Fi.....72
4.2.1 Емуляція помилок.....74
4.2.2 Розмір потоку і кодування.....76
4.2.4 Чутливість до втрат пакетів при потокової передачі.....81
4.2.5 Чутливість до втрат пакетів при відтворенні файлів.....83
4.3 Огляд видів кодування звуку та проблеми при його передачі……....83
Висновки…….87
5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ……....89
5.1 Визначення потенційно небезпечних і шкідливих факторів при використанні ВДТ ПЕОМ…….....90
5.2. Технічне рішення та організаційні заходи з безпеки та гігієни праці…….....91
5.3 Безпека в надзвичайних ситуаціях……100
ЗАГАЛЬНІ ВИСНОВКИ.....106
ПЕРЕЛІК ПОСИЛАНЬ……...107
ДОДАТКИ……108

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних.

СУБД – Система управління базами даних.

ПЕОМ – Персональна електроно-обчислювальна машина.

ЕОМ – Електроно-обчислювальна машина.

ПК – Персональний комп’ютер.

ЕПТ – Електронно-променева трубка.

ВДТ – Візуально дисплейний термінал.

ЕМП – Електромагнітне поле.

ДСТУ – Державний стандарт України.

API – Application programming interface.

JDBC – Java DataBase Connectivity.

JDK – Java Development Kit.

GPL – General Public License.

IDE – Integrated Development Environment.

ВСТУП

В даній дипломній роботі виконане дослідження керування відеопотоком з використанням мікрокомп’ютерів. Підставою для дослідження слугує наказ №298/2-с по РТФ від 11.03.2015. Сьогодні все більшої популярності набувають мікрокомп’ютери. Все частіше пропонуються пристрої та системи для так званого Інтернету речей.

Інтернет речей— концепція комунікаційної мережі фізичних або віртуальних об'єктів («речей»), які мають технології для взаємодії між собою та з оточуючим середовищем, а також можуть виконувати певні дії без втручання людини. Концепція полягає в тому, щоб всі предмети побуту, товари, вузли технологічних процесів тощо, були оснащені вбудованими комп'ютерами та сенсорами, мали змогу обробляти інформацію, що надходить із навколишнього середовища, обмінюватися нею та виконувати різні дії в залежності від отриманої інформації.

Результатом виконання дипломної роботи очікується система керування відепотоком за допомогою мікрокомп’ютера та пристрою на операційній системі Android, яка дозволяю с будь – якого Android пристроя підключитись до налаштованого мікрокомп’ютера в мережі та приймати відеопотік від нього. Так як на сьогоднішній день кожен другий телефон підключений до мережі інтернет розроблена система дозволяє створити цілу мережу відео спостереження, до якої власник цієї мережі зможе підключитися з будь – якої точки світу.

Актуальність теми

Враховуючи швидке зростання популярності мікрокомп’ютерів, зростає також велика кількість програмного забезпечення, які дозволяють використовувати дані пристрої в повсякденному житті.

Дослідження, які проводяться в магістерській дисертаціх сприяють розвитку такого програмного забезпечення, а саме, більш ефективному впрровадженню системи відеоспостереження.

Зв’язок роботи з науковими програмами, планами, темами

На кафедрі радіоприймання та оброблення сигналів проводиться активне впровадження сучасних пристроїв, зокрема мікроконтролерів та мікрокомп’ютерів, на яких студенти реалізують проекти для Інтернету речей.

Мета і задачі дослідження

Метою роботи є розробка та дослідження системи для керування відео потоком з використанням мікрокомп’ютерів.

Об’єкти досліджень — 1) мікрокомп’ютер Raspberry Pi розроблений британським фондом Raspberry Pi Foundation; 2) об’єкто орієнтовна мова програмування Java; 3) Операційна система мобільних пристроїв Android розроблена компанією Google.

Предмет досліджень — характеристики керування відео потоком з використання в Android пристроях та мікрокомп’ютерах.

Головними задачами дослідження є:

- Створення програмного забезпечення операційної системи Android, для приймання відеопотока від мікрокомп’ютерів;
- Налаштування мікрокомп'ютера для передачі відеопотока;
- Дослідження основних характеристик відеопотока для вибору оптимальних параметрів;

Наукова новизна одержаних результатів

В магістерській дисертації:

- дослідження впливу зовнішніх факторів на передачу відеопотока;
- розроблені програмне забезпечення для передачі відеопотоку для операційної системи Android за допомогою Java.

Провівши аналіз існуючих рішень керування відеопотоком стало зрозуміло, що не існує жодної системи передачі відео від мікрокомп’ютера до мобільних пристроїв без використання пристроїв – посередників.

Публікації:

Під час виконання досліджень написана стаття по результатам супутніх досліджень:

Дідух О. І., Тищенко В. В. Порівняння швидкодії Java на мікрокомп’ютері Raspberry Pi. // Вісник НТУУ „КПІ”. Серія радіотехніка, радіоапаратобудування. — 2015. — № 60. — С. 107—113.

Могильний С.Б., Дідух О.І, Тищенко В.В. Raspberry pi, mathematica і Wolfram як основа створення елементів навчальних курсів. — Науково-практична конференція "Мультимедійні технології в освіті та інших сферах діяльності": Тези доповідей. - К.:НАУ, 2014 - 94 с.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

В технічному завданні була поставлена задача на розробку та дослідження системи керування відепотоком з використанням мікрокомп’ютерів. Виходячи з технічного завдання, дана система повинна бути призначена для передачі відеопотоку, яки транслюється мікрокомп’ютером та приймайться пристроєм на операційній системі Android.

Вимоги до апаратного та програмного забезпечення, на якому буде функціонувати система:

Мікрокомп’ютер: операційна система сімейства Unix, RAM 256 MB, мережний протокол TCP/IP або UDP, наявність веб камери.

Мобільний пристрій: операційна система Android 2.1 та вище.

2 ВИБІР ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТІВ

2.1 Аналіз аналогів

До розділу «Аналіз аналогів» входить вибір методу та підходу до реалізації системи та безпосередній аналіз готових рішеннь.

Проведемо обгрунтовуання та вибір критичних параметрів, що характеризують систему, з метою вирішення поставлених завдань. Сформулюємо критерії, та проводемо порівняльний аналіз представлених на сьогоднішній день систем керування відеопотоком з метою уточнення напрямів дослідження.

2.1.1 Переваги та недоліки передачі відеопотока за допомогою Wi-Fi

Перед початком розгляду конкретних технологій передачі відео та звуку за допомогю Wi-Fi, потрібно визначити основні переваги та недоліки передачі таким способом.

Переваги:

1) Більшість комп'ютерів (і не тільки їх) вже оснащено Wi-Fi - не потрібен окремий передавач, все, що потрібно для трансляції вже є;
2) Можна використовувати не тільки для бездротової передачі відео та звуку але і для отримання доступу до мережі;
3) Із за широкої поширеності звертає на себе увагу найбільших учасників IT-індустрії, таких як Intel, Apple, Qualcomm, Cavium Networks.

Недоліки:

1) Бездротова передача відео та звуку віднімає частину ефіру у прямого призначення Wi-Fi - доступу в мережу;
2) Роботі вашої мережі можуть заважати навколишні Wi-Fi мережі, яких з кожним днем стає все більше;

3) Для того, щоб HD-відео і звук потрапляли в смугу пропускання Wi-Fi, потрібно «упакувати» їх відповідним кодеком (у більшості випадків - h.264), що дає несуттєву втрату якості;
4) Потреба в стисненні народжує потребу в софті, який може працювати на одній, але не працювати на іншій ОС / платформі;
5) Робота кодека зі стиснення контенту вимагає апаратних ресурсів, при тому чималих;
6) Із за роботи кодека передача сигналу затримується на дельту часу, що йде на стиск (від 20мс до 2 секунд, залежно від відстані і потужності стискає апаратури).

2.1.2 Технології передачі відео потоку

1) DLNA

Це найпростіший спосіб і, напевно, найпоширеніша на сьогоднішній день технологія передачі фільмів і музики по LAN / Wi-Fi. Сам принцип DLNA полягає в тому, що на комп'ютері запускається серевер, в якому прописана відкрита для перегляду папка з кіно та музикою. Сам (розумний) ТБ, або якийсь посередник, типу SetTop box'a або тюнера або ігрової приставки, підключається по Wi-Fi до расшаренной папці і виводить з неї контент на HDTV.

Переваги:

- Вбудований в більшість сучасної телевізійної техніки.

Недоліки:

- Низька продуктивність при передачі робочого столу / програм;
- Складність в налаштуванні із за обміну ролями клієнт <-> сервер.

2) Remote Desktop

Для отримання відео та звуку з вашого комп'ютера на ньому повинен бути запущений сервер одного з протоколів віддаленого робочого столу, а біля

телевізора повинен знаходиться клієнт, який повинен ініціювати підключення до вашого комп'ютера. Продуктивність цієї технології вважається невисокоїю, інколи - дуже низькою.

З іншого боку, це вихід для малобюджетних рішень, коли біля телевізора досить поставити тільки старий комп'ютер і оптимально його налаштувати.

Переваги:

- Для роботи достатньо будь-якого старого ПК або навіть смартфона з виведенням на ТБ.

Недоліки:

- Анологічні тим, що і у DLNA.

3) Бездротові сервери презентацій(wireless presentation gateway)

До Wi-Fi точки доступу, крім інтернету, можна підключити телевізор або монітор з колонками, і за допомогою невеликої утиліти виводити картинку і звук з ПК на ці ТВ / монітор / колонки. Це і є схема роботи бездротового сервера презентації. Грубо кажучи, це точка доступу (наприклад) wi-fi роутер, який крім інтернету надає вашому ПК або смартфону пристрій виведення зображення і звуку.

У більшості випадків, коли ви хочете передати через сервер презентацій вже не робочий стіл і програми, а кіно і музику, в передавальну утиліту вбудований плеєр, відкриваючи яким ваші медіафайли вони передаються на точку доступу вже по протоколу DLNA, тобто у віконному режимі ви відео переглянути не зможете, тільки на повний екран. Коли фільм фбо музика закінчується, додаток автоматично перемикається назад, в режим передачі зображення робочого столу.

У різний час подібні точки доступу випускали Dlink, Planet, Edimax, ViewSonic й інші, менш відомі автори. Але найкращого результату за якістю передачі й зниженню затримки досягла фірма Awind зі своїм продуктом McTivia: це невелика точка доступу 802.11n, що має HDMI вихід (максимальний дозвіл - 1280x720, звук - стерео). Також є вхід ethernet для надання загального доступу до

мережі (до речі, зв'язок утиліти і точки доступу працює і по LAN); а також є USB-вхід для клавіатури / миші, щоб можна було керувати комп'ютером, на якому запущена утиліта, віддалено. Оскільки протокол передачі помітно оптимізований в порівнянні з попередниками, вдалося прибрати DLNA частину і запускати відео й звук безпосередньо, через будь-який програвач.

При цьому у утиліти є кілька режимів якості: при найменшій якості, відповідно, найнижча затримка - це для презентацій та іншого інтерактиву; при найвищій якості затримка майже секунда, але це для кіно, де в хорошій якості запустили і не чіпаєте, так що затримка не заважає.

Переваги:

- Працює практично на будь-якій платформі, в тому числі і на мобільних ОС;
- Оптимізована якість, прийнятна при середньостатистичному комп'ютері;
- Софт частково використовує GPU для прискорення стиснення
- Є можливість керувати віддаленим компьтером прямо з точки доступу (USB-клавіатурою / мишею);
- Можна використовувати одночасно ще й як точку доступу для роздачі мережі.

Недоліки:

- Чим слабкіше комп'ютер - тим гірше продуктивність;
- Відео обмежено здатністю 1280x720, про 3D мови немає, звук - тільки стерео;
- Досить велика затримка, можуть виникнути проблеми з іграми;
- Необхідно витрачати кошти на купівлю WPG-обладнання.

4) Intel WiDi

Інтел є ключовим виробником чипів на ринку процесорів для ПК і ноутбуків, і має ряд рішень, розроблених величезними research-центрами та дослідницькими лабораторіями, по всьому світу.

Оскільки Intel все тісніше з'єднує всі компоненти сучасного комп'ютера, такі як CPU, GPU і Wi-Fi адаптер, в рамках своєї єдиної платформи, то вони можуть дозволити собі випускати софт, оптимально адаптований під цю платформу, так як вона - масова. WiDi працює тільки на процесорах Core другого покоління і тільки з Intel Wi-Fi чіпом. Стисненню потоку для передачі допомагає вбудований в кристал процесора графічний адаптер.

Для прийому сигналу на телевізорі використовується спеціальна приставка WiDi, якість FullHD 1080p, але на сьогодні тілки 30fps. Звук - багатоканальний. Затримка - від 20мс і більше (зазвичай - більше). Для з'єднання з WiDi-приймачем використовується спеціальна технологія від Intel, подібна множинному Wi-Fi p2p, тобто ваш ПК може бути приєднаний до WiDi-адаптеру одночасно разом із з'єднанням з інтернетом через точку доступу.

В якості альтернативи приймачу Intel обіцяє вбудовувати технологію в сучасні телевізори. Або ж, що ще більш цікаво: пропонують відправляти / приймати контент по DLNA (як підтвердження на CES2012 показували PS3 і Xbox360, що приймають сигнал від ноутбука з WiDi).

Переваги:

- Працює на всіх ПК з сучасною платформою Intel;
- Софт оптимізований під платформу, активно використовується потужність GPU;
- Постійні оновлення, висока якість сигналу (FullHD + 5.1), зниження затримки;
- Поява вбудованих в ТБ і Sat / IPTV-приставки приймачів і передача через повсюдний DLNA.

Недоліки

- Тільки залізо від Intel - CPU Core, працює на вбудованій відеокарті, потрібен Wi-Fi чіп Intel;
- Працює тільки під Windows;
- На мобільній платформі тільки в стадії прототипу на платформі Intel (Cedar Trail), на ARM відсутня;
- Відчутна затримка передачі сигналу;
- Потрібно купувати окремий приймач WiDi.

5) Apple AirPlay

Це точка доступу, до якої можна підключити колонки (AirPort Express) і транслювати музику з iTunes. Потім це переросло ще й у трансляції на AppleTV (вся технологія грунтувалася на DLNA). Потім з'явилися гаджети і технологія AirPlay: тепер з вбудованого плеєра гаджетів стало можливо слати як відео так і музику на AppleTV і тільки музику на AirPort Express.

Але з появою двоядерних айфонів і айпедів все змінилося. Тепер потужності стало достатньо і одне ядро процесора може відповідати за основні завдання, повністю відповідаючи характеристикам попереднього покоління, а друге ядро може стискати все, що відбувається на екрані і з динаміках, і відправляти це на AppleTV. Так що AirPlay виріс до повноцінної передачі робочого столу мобільних пристроїв на телевізор (з роздільною здатністю 1280x720).

Переваги:

- Працює практично на всіх сучасних пристроях від Apple, в тому числі і мобільних, і через iTunes на Windows;
- Софт оптимізований під платформу, активно використовуються апаратні особливості;
- Сигнал має гарантовану якість і відсутність градацій;
- Абсолютно точно буде розвиватися далі;
- AirPlay вбудовується в безліч аксесуарів для Apple-пристроїв.

Недоліки:

- Тільки залізо від Apple (Linux і Android не підтримуються);
- Працює тільки під MacOS / iOS, за винятком DLNA функцій через iTunes на Windows (або Airfoil)
- Якість 720p з затримкою.
- Невідомо, чи буде можливість вбудовувати AirPlay приймачі HD-відео та звуку в іншу техніку, крім існуючої AppleTV.

2.1.3 Безпосередній аналіз мікрокомп’ютерів доступних на ринку

Мікрокомп’ютер - самодостатній комп'ютер, зібраний на одній друкованій платі, на якій встановлені мікропроцесор, оперативна пам'ять, системи введення-виведення і інші модулі, необхідні для функціонування комп'ютера. Це одноплатні комп'ютери, які виготовляються в якості демонстраційних систем, систем для розробників або освіти, або для використання в ролі промислових комп'ютерів.

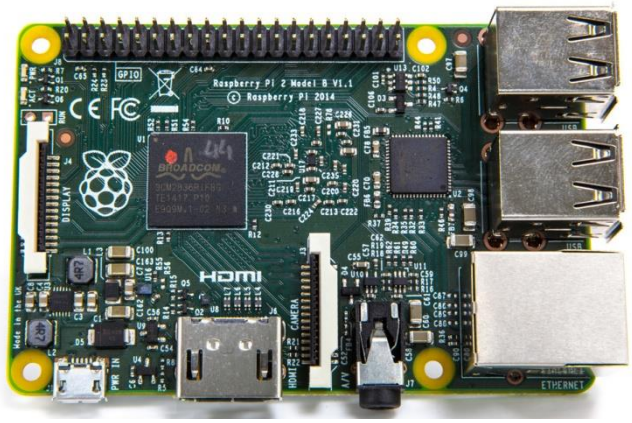
На відміну від традиційних персональних комп'ютерів форм-фактора «desktop» (стандарти AT, ATX, і т. П.), Одноплатні комп'ютери часто не вимагають установки якихось додаткових периферійних плат. Деякі одноплатні системи виготовлені у вигляді компактної плати з процесором і пам'яттю, що підключаються до backplane для розширення можливостей, наприклад, для збільшення кількості доступних роз'ємів.

Найчастіше ці рішення повинні бути захищені або дуже компактні, тому всі компоненти повинні розташовуватися на одній платі.

Така економія з одного боку робить цілий пристрій більш компактним, з іншого боку, розширення можливостей - зміна процесора пам'яті - ускладненні, оскільки найчастіше ці компоненти напаяні на плату.

Розглянемо основні рішення на ринку одноплатних комп’ютерів і виберемо один з них для подальшої роботи

1) **Raspberry Pi 2B**



Raspberry Pi - одноплатний комп'ютер розміром з банківську карту, спочатку розроблений як бюджетна система для навчання інформатиці. Згодом отримав набагато більш широке застосування і популярність, ніж очікували його автори.

Безумовно, Raspberry Pi є повнофункціональним комп'ютером. Він володіє всіма атрибутами справжнього комп'ютера: виділеним процесором, пам'яттю і графічним драйвером для виводу через HDMI. На ньому навіть працює спеціальна версія операційної системи Linux. Тому на Raspberry Pi легко встановити більшість програм для Linux. Варто трохи попрацювати - і Raspberry Pi можна використовувати як повноцінний медіа-сервер або емулятор відеоігор.

Хоча в Pi і відсутнє внутрішнє сховище даних, на цьому комп'ютері можна використовувати смарт-картки в якості флеш-пам'яті, яка обслуговує всю систему. Таким чином, можна швидко вивантажувати для налагодження різні версії операційної системи або програмні оновлення. Оскільки цей пристрій забезпечує незалежну з'єднуваність по мережі, його можна налаштовувати і для доступу по SSH, або пересилати на нього файли по протоколу FTP. Також цей пристрій підтримує широкий спектр операційних систем: Debian, Fedora, Arch Linux,Gentoo, RISC OS, Android,Firefox OS, NetBSD, FreeBSD,Slackware, Tiny Core Linux,Windows 10.

Основні характеристики

SoC	BCM2835
Процесор	Broadcom BCM2836 - 4 ядра ARM Cortex-A7 1ГГц
Графічний процесор	Broadcom VideoCore IV
ОЗП	1 Гб

Вбудована пам’ять	MicroSD
Мережа	10/100/1000 Ethernet
Габарити	85,6 × 53,98×17 мм
Ціна	$35

2) Banana Pi

Banana Pi - одноплатний комп'ютер. Може працювати під управлінням різних операційних систем: Android 4.2, Android 4.4, Lubuntu, Raspbian, Debian, Fedora, Arch Linux, Gentoo, openSUSE, Berryboot, FreeBSD, OpenWrt, FreeBSD, Slackware, Tiny Core Linux. Banana Pi розроблений компанією «Lemaker» (Китай) з метою сприяння STEAM (науці, технології, інженерії, мистецтва і математики) в загальноосвітніх школах.

Banana Pi не має ніякого відношення до Raspberry Pi, хоча їх схожість очевидно . Журнал «Linux user & Developer» не вважає його прямим клоном але зазначає значну схожість , в той час як linux.com вважає його клоном з поліпшеними характеристиками . Компонування друкованої плати лише незначно відрізняється, хоча відносні розміри різні. Крім того, не всі аксесуари сумісні з Raspberry Pi.

Основні характеристики

SoC	Allwinner A20
Процесор	ARM Cortex-A7 dual core
Графічний процесор	ARM Mali400MP2 dual GPU core
ОЗП	1 Гб
Вбудована пам’ять	SD слот
Мережа	10/100/1000 Ethernet
Габарити	92 × 60 мм

Ціна	$74

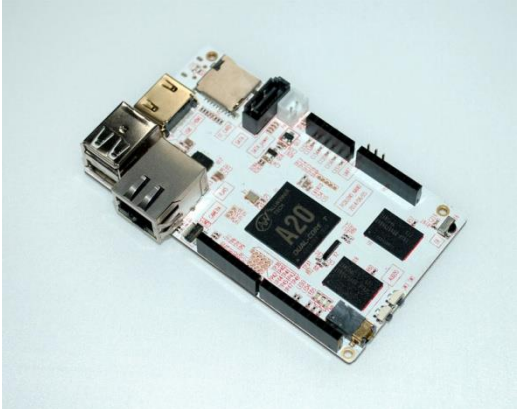
3) Cubieboard

Cubieboard - це материнська плата, що продається в якості одноплатного комп'ютера, проводиться в Шеньчжень, провінції Гуандун, Китай. Перший прототип плати почав продаватися на міжнародному рівні у вересні 2012 року, а серійну версію почали продавати в жовтні того ж року. Плата може працювати на Android 4 ICS, Ubuntu 12.04 desktop, Fedora 19 ARM Remix desktop , XBMC media player system, Archlinux ARM , OpenBSD / armv7 або basic Debian server через Cubian дистрибутив .

Основні характеристики

SoC	Allwinner A80
Процесор	4x Cortex-A15
Графічний процесор	PowerVR G6230
ОЗП	2Гб
Вбудована пам’ять	SD слот
Мережа	10/100/1000 Ethernet
Габарити	111.43мм × 111.43мм
Ціна	125$

4) PcDuino3 Nano



PcDuino3 Nano – одноплатний комп’ютер розроблений компанією Linksprite являє собою відлагоджувальну плату побудовану на двоядерному AllWinner A20 SoC, з Arduino-

сумісним майданчиком. Це дозволяє підключати Arduino плати розширення до PcDuino3 Nano, як ніби це Arduino UNO. Додатково є SATA роз'єм для підключення HDD або SSD.

PcDuino3 Nano, це менш габаритна і дешевша версія PcDuino3, без Wi-Fi модуля, LVDS, і I2C інтерфейсу. Плата не підтримує підключення Li-Po акумуляторних батарей, але доданий ще один USB порт в порівнянні зі старшою моделлю.

Основні характеристики

SoC	AllWinner A20
Процесор	Cortex™-A7 Dual-Core
Графічний процесор	ARM® Mali400MP2
ОЗП	1 Гб
Вбудована пам’ять	4GB NAND пам'ять, SD слот
Мережа	10/100/1000 Ethernet
Габарити	91.4mm x 53.3mm
Ціна	38$

5) ODROID-C1

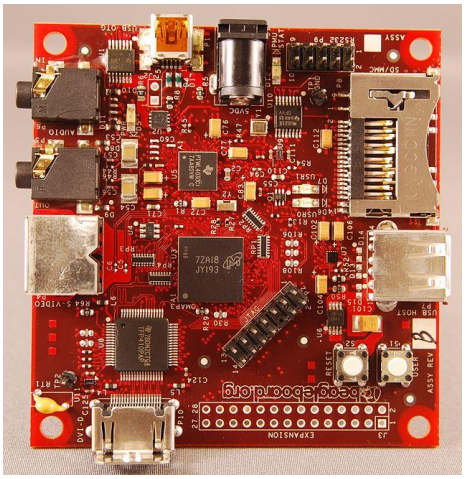
У грудні 2014 року южнокорейсая компанія HardKernel представила свою нову платформу ODROID-C1, яка за ціною, формфактору і функціоналу близька до RPi. Вартість комп'ютера 35 $ як і RPi. Потужність цього рішення в 2-4 рази вище ніж аналогічного від Raspberry Pi B+ , але нова модель 2B має такі-ж показники в тестах. В якості операційних систем використовуються Ubuntu 14.04 LTS, Android KitKat

Основні характеристики

SoC	Amlogic S805

Процесор	Quad Core Cortex™-A5
Графічний процесор	Mali-450
ОЗП	1 Гб
Вбудована пам’ять	microSD
Мережа	10/100/1000 Ethernet
Габарити	91.4mm x 53.3mm
Ціна	35$

6) Beagle board



Beagle board - недорогий одноплатний комп'ютер розроблений компаніями Texas Instruments і Digi-Key. Спочатку Beagle board розроблявся у тісній співпраці з спільнотою open source з метою демонстрації можливостей системи на мікропроцесорі OMAP3530.

Основні характеристики

SoC	TI DM3730
Процесор	ARM Cortex-A8
Графічний процесор	PowerVR SGX
ОЗП	512 Мб
Вбудована пам’ять	4 ГБ microSD з Angstrom Linux
Мережа	10/100/1000 Ethernet
Габарити	82мм x 82мм
Ціна	55$

2.1.4 Вибір програмної оболонки та інструментів

2.1.4.1 Вибір мови програмування

Спочатку треба провести аналіз та порівняння наявних мов з допомогою яких можна вирішити задачу поставлену в технічному завданні.

Мова програмування Асемблер

До мов низького рівня належать мова Асемблер. Свою назву він отримав від імені системної програми Асемблер, яка перетворює вихідні програми, написані на таких мовах, безпосередньо в коди машинних команд. Частинами тут служать оператори, а результатом збірки послідовність машинних команд Мова Асемблера об'єднує в собі переваги мови машинних команд і деякі риси мов високого рівня. Асемблер забезпечує можливість застосування символічних імен у вихідній програмі і позбавляє програміста від стомлюючої праці (неминучого при програмуванні на мові машинних команд) з розподілу пам'яті комп'ютера для команд, змінних і констант.

Асемблер дозволяє також гнучко і повно використовувати технічні можливості комп'ютера, як і мову машинних команд. Транслятор вихідних програм в Асемблері простіше транслятора, потрібного для мови програмування високого рівня. На Асемблері можна написати настільки ж ефективну за розміром і часу виконання програму, як і програму мовою машинних команд. Ця риса відсутня у мов високого рівня. Ця мова часто застосовують для програмування систем реального часу, технологічними процесами та обладнанням, забезпечення роботи інформаційно-вимірювальних комплексів. До таких систем звичайно пред'являються високі вимоги за обсягом займаної машинної пам'яті. Часто мова Асемблера доповнюється засобами формування макрокоманд, кожна з яких еквівалентна цілій групі машинних команд. Таку мову називають мовою макроассемблера. Застосування мак "будівельних" блоків і наближає мову

Асемблера до мови високого рівня. Асемблер - машинно-залежна мова, тобто він відображає особливості архітектури конкретного типу комп'ютера.

Мова програмування Бейсік

Бейсік (BASIC - BASIC - Beginner’s All-Purpose Symbolic Instruction Code - "універсальний символічний код інструкцій для початківців"). Прямий нащадок Фортрана. З'явився Бейсік в 1963 році (назвати автора було б важко, але основна заслуга в його появі безсумнівно належить американцям Джону Кемені і Томас Курц). Як і будь-які переваги, простота Бейсика оберталася, особливо в ранніх версіях труднощами структурування; крім того, Бейсік не допускав рекурсію - цікавий прийом, що дозволяє складати ефективні і в той же час короткі програми.

Розроблено потужні компілятори Бейсіка, які забезпечують не тільки багату лексику і високу швидкодію, але і можливість структурного програмування. На думку деяких програмістів, найцікавішими версіями є GWBASIC, Turbo-Basic і Quick Basic.

Свого часу поява Quick Basic ознаменувало народження другого покоління систем програмування на мові Бейсік. Він надає можливість модульного і процедурного програмування, створення бібліотек, компіляції готових програм та інше, що вивело його на рівень таких класичних мов програмування, як Сі, Паскаль, Фортран та ін. Більше того, у зв'язку з відсутністю офіційного стандарту мови Бейсік, його реалізація у вигляді Quick Basic стала фактичним стандартом. Безумовними лідерами серед різних версій Бейсика були Quick Basic 4.5 і PDS 7,1 фірми Microsoft, що з'явилися в кінці 80-х років.

Мова програмування Паскаль

Мова програмування Паскаль була розроблена професором кафедри обчислювальної техніки Швейцарського Федерального інституту технології Ніколасом Віртом в 1968 році як альтернатива існуючим і ускладнювавшимися мовам програмування, таким, як PL / 1, Алгол, Фортран. Інтенсивний розвиток

Паскаля призвело до появи вже в 1973 році його стандарту у вигляді переглянутого повідомлення, а число трансляторів з цієї мови в 1979 році перевалило за 80. Найважливішою особливістю Паскаля є втілена ідея структурного програмування. Іншою істотною особливістю є концепція структури даних як одного з фундаментальних понять.

Основні причини популярності Паскаля полягають в наступному:

- Простота мови дозволяє швидко його освоїти і створювати алгоритмічно складні програми

- Розвинені засоби представлення структур даних забезпечують зручність роботи як з числовою, так і з символьної і бітової інформацією

- Наявність спеціальних методик створення трансляторів з Паскаля спростило їх розробку і сприяло широкому поширенню мови

- Оптимізує властивості трансляторів з Паскаля дозволяють створювати ефективні програми. Це послужило однією з причин використання Паскаля в якості мови системного програмування

- У мові Паскаль реалізуються ідеї структурного програмування, що робить програму наочною і дає хороші можливості для розробки та налагодження.

Мова програмування С

Співробітник фірми Bell Labs Денис Рітчі створив мову С в 1972 році під час спільної роботи з Кеном Томпсоном, як інструментальний засіб для реалізації операційної системи Unix, проте популярність цієї мови швидко переросла рамки конкретної операційної системи і конкретних завдань системного програмування. В даний час будь-яка інструментальна і операційна система не може вважатися повною якщо до її складу не входить компілятор мови С. Мова програмування Сі була розроблена як інструмент для програмістів-практиків. Відповідно до цього головною метою його автора було створення зручного і корисної в усіх відношеннях мови.

С є знаряддям системного програміста і дозволяє глибоко влізати в найтонкіші механізми обробки інформації на ЕОМ. Хоча мова вимагає від програміста високої дисципліни, вона не строга у формальних претензіях і допускає короткі формулювання.

С - сучасна мова. Вонв включає в себе ті управляючі конструкції, які рекомендовані теорією і практикою програмування. Її структура спонукає програміста використовувати у своїй роботі нисхідне проектування, структурне програмування і покрокову розробку модулів.

С - потужна і гнучка мова. Велика частина операційної системи Unix, компілятори та інтерпретатори мов Фортран, Паскаль, Лисп, і Бейсік написані саме з її допомогою.

С - зручна мова. Вона досить структурована, щоб підтримувати хороший стиль програмування і разом з тим не пов'язана жорсткими обмеженнями. Крім набору засобів, притаманних сучасним мовам програмування високого рівня (структурність, модульність, певні типи даних), в неї включені засоби для програмування практично на рівні асемблера. Великий набір операторів і засобів вимагають від програміста обережності, акуратності і доброго знання мови з усіма її перевагами і недоліками.

Мова програмування С ++

Мова С ++ з'явилася на початку 80-х років. Створена Бьерном Страуструпом з первинною метою позбавити себе і своїх друзів від програмування на асемблері, Сі чи різних інших мовах високого рівня.

На думку автора мови, відмінність між ідеологією С і С ++ полягає приблизно в наступному: програма на С відображає "спосіб мислення" процесора, а

С ++ - спосіб мислення програміста. Відповідаючи вимогам сучасного програмування, С ++ робить акцент на розробці нових типів даних найбільш повно відповідних концепціям обраної галузі знань і завданням програми. Клас є

ключовим поняттям С ++. Опис класу містить опис даних, потрібних для подання об'єктів цього типу і набір операцій для роботи з подібними об'єктами.

На відміну від традиційних структур С і Паскаля, членами класу є не тільки дані, але й функції. Функції - члени класу мають привілейований доступ до даних усередині об'єктів цього класу і забезпечують інтерфейс між цими об'єктами і решті програмою. При подальшій роботі зовсім не обов'язково пам'ятати про внутрішню структуру класу і механізм роботи вбудованих функцій. У цьому сенсі клас подібний електричному приладу - мало хто знає про його пристрої, але всі знають, як ним користуватися.

Мова С ++ є засобом об'єктного програмування, новітньої методики проектування та реалізації програм, яка в поточному десятилітті, швидше за все, замінить традиційне процедурне програмування. Головною метою творця мови доктора Бьерна Страустрапа було оснащення мови С ++ конструкціями, що дозволяють збільшити продуктивність праці програмістів і полегшити процес оволодіння великими програмними продуктами.

Абстракція, реалізація, спадкування та поліморфізм є необхідними властивостями якими володіє мова С ++, завдяки чому вона не тільки універсальна, як і мова С, але і є об'єктною мовою.

Мова програмування Java

Мова Java зародилася як частина проекту створення передового програмного забезпечення (ПО) для різних побутових приладів. Реалізація проекту була розпочата на мові С ++, але незабаром виник ряд проблем, найкращим засобом боротьби з якими була зміна самого інструмента - мови програмування. Стало очевидним, що необхідна платформо-незалежна мова програмування, що дозволяє створювати програми, які не доводилося б компілювати окремо для кожної архітектури і можна було б використати на різних процесорах під різними операційними системами. Мова Java потрібна для створення інтерактивних продуктів для мережі Internet. Фактично, більшість архітектурних рішень, прийнятих при створенні Java, було продиктовано

бажанням надати синтаксис, схожий з C і C ++. У Java використовуються практично ідентичні угоди для оголошення змінних, передачі параметрів, операторів і для управління потоком виконання коду. У Java додані всі хороші риси C ++.

Три ключові елементи об'єдналися в технології мови Java:

- Java надає для широкого використання свої аплети (applets) - невеликі, надійні, динамічні, незалежні від платформи активні мережеві додатки, що вбудовуються в сторінки Web. Аплети Java можуть налаштовуватися і поширюватися споживачам з такою ж легкістю, як будь-які документи HTML.

- Java звільнює міць об'єктно-орієнтованої розробки додатків, поєднуючи простий і знайомий синтаксис з надійним та зручним у роботі середовищем розробки. Це дозволяє широкому колу програмістів швидко створювати нові програми і нові аплети.

- Java надає програмісту багатий набір класів об'єктів для ясного абстрагування багатьох системних функцій, використовуваних при роботі з вікнами, мережею і для вводу-виводу. Ключова риса цих класів полягає в тому, що вони забезпечують створення незалежних від використовуваної платформи абстракцій для широкого спектра системних інтерфейсів.

На рисунку 2.1 представлена схема компіляції та виконання коду написаного на мові Java.

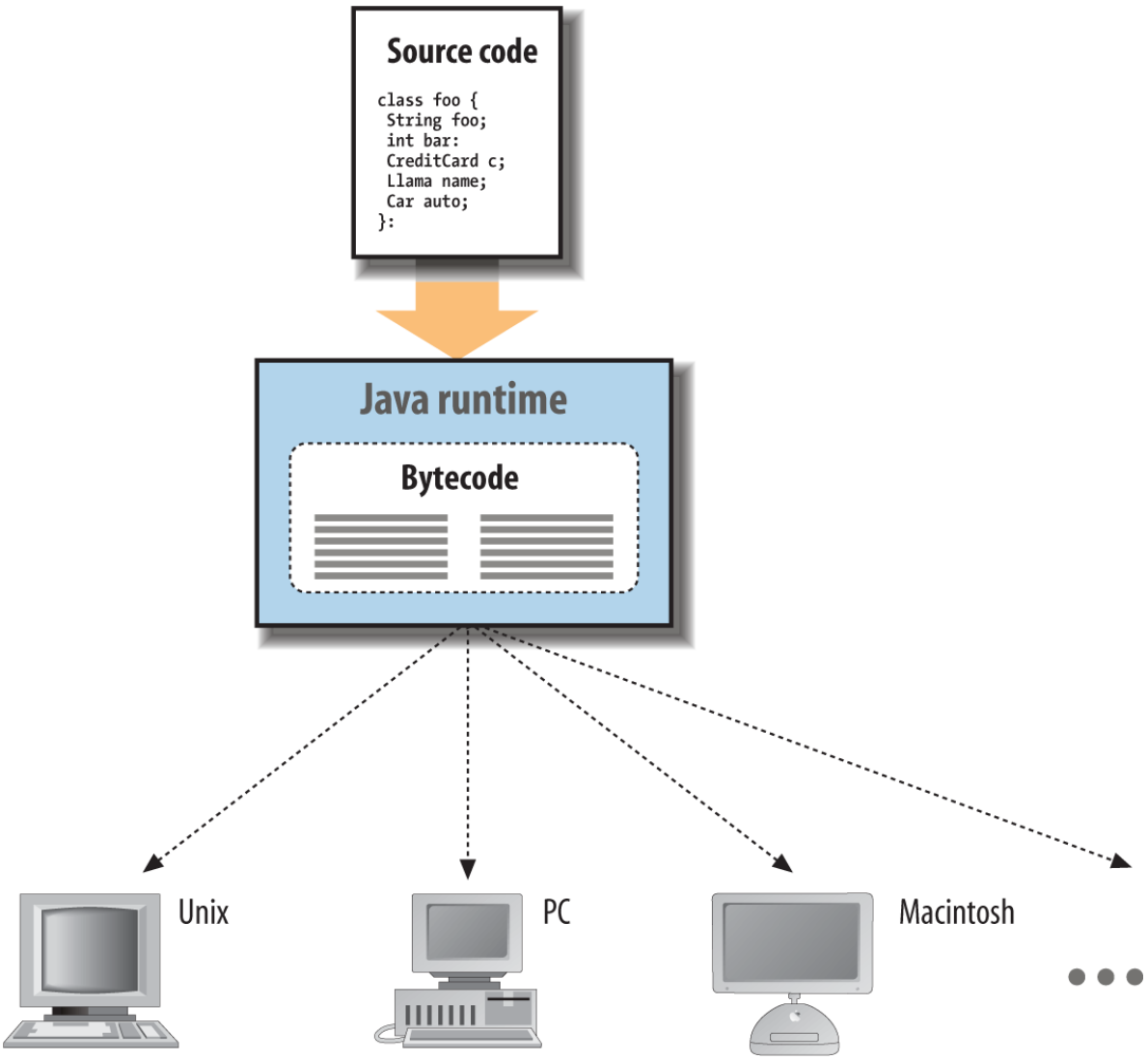


Рис. 2.1

(де Source code – вихідний код, Java runtime – середовище виконання Java, bytecode – машиний код)

2.1.4.2 Вибір інструментів реалізації.

Треба провести вибір та обґрунтування всіх необхідних інструментів для реалізації поставленої в технічному завданні задачі.

2.1.4.2.1 Вибір інтегрованої середи розробки.

Так як в попередньому розділі було прийнято рішення розробляти дану роботу на мові програмування Java, то відповідно треба порівняти та вибрати інтегровану середу розробки. Розглянемо популярні на сьогоднішній день рішення.

1) Eclipse - вільне інтегроване середовище розробки модульних кроссплатформенних додатків. Розвивається і підтримується Eclipse Foundation.

Eclipse служить в першу чергу платформою для розробки розширень: будь-який розробник може розширити Eclipse своїми модулями. Вже існують Java Development Tools (JDT), C / C ++ Development Tools (CDT), що розробляються інженерами QNX разом з IBM, і мов Ada (GNATbench, Hibachi), COBOL, FORTRAN, PHP, X10 (X10DT) та ін. Від різних розробників. Безліч розширень доповнює середу Eclipse диспетчерами для роботи з базами даних, серверами додатків і ін.

Eclipse JDT (Java Development Tools) - найбільш відомий модуль, націлений на групову розробку: середа інтегрована з системами керування версіями - CVS, GIT в основній поставці, для інших систем (наприклад, Subversion, MS SourceSafe) існують плагіни. Також пропонує підтримку зв'язку між IDE і системою управління завданнями (помилками). В основній поставці включена підтримка трекера помилок Bugzilla, також є безліч розширень для підтримки інших трекерів (Trac, Jira та ін.). В силу безкоштовності і високої якості, Eclipse в багатьох організаціях є корпоративним стандартом для розробки додатків.

Eclipse написан на Java, тому є платформо-незалежним продуктом, за винятком бібліотеки SWT, яка розробляється для всіх поширених платформ . Бібліотека SWT використовується замість стандартної для Java бібліотеки Swing. Вона повністю спирається на платформу (операційну систему), що забезпечує швидкість і натуральний зовнішній вигляд для користувача інтерфейсу, але іноді викликає на різних платформах проблеми сумісності і стійкості додатків.

2) JetBrains IntelliJ IDEA (Android Studio) - це провідне середовище швидкої розробки мовою Java. IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторінга коду, вбудовану підтримку технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами керування версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів. Включає в себе:

- вбудований декомпілятор Java-класів
- оптимізований Java-відладчик, що дозволяє всередині редактора бачити значення змінних і на що посилаються об'єкти
- виявлення стильових настройок коду на ходу
- нові опції редактора - "розумна" клавіша Backspace і множинне виділення
- вдосконалений аналіз коду з авто-виділення анотацій NotNull, Nullable і Contract
- підтримка та додаткова допомога при написанні коду для Thymeleaf
- покращений процес інтеграції з Gradle і Maven
- вбудований JavaFX Scene Builder
- підтримка Drools 6
- поліпшена інтеграція з Git і Mercurial
- Scratch files дозволяє здійснювати швидке прототипування коду без створення файлів в проекті
- нові опції з Android Studio Beta, в т.ч. підтримка Android Wear і TV
- постфікси-додаток коду для JavaScript
- підтримка PhoneGap, Cordova і Ionic
- нові інструменти для роботи з БД.

Висновок: Звичайно видно, що більш досконалою середою розробки є Android Studio, саме цей інструмент зараз біль розвинений для розробки додатків

під Android, компанія Google кожні 3 місяця робить великі оновлення і додає нові можливості, що робить розробку Android додатків більш зручною.

2.1.4.3 Вибір Бази Даних.

Вибір бази даних являється важливим рішенням при розробці програмного забезпечення, тому що треба зважити всі плюси і мінуси того чи іншого рішення, так як від швидкості доступу до Бази Даних залежить швидкодія всієї системи в цілому та простота реалізації всієї системи.

1) SQLite

База даних, що легко вбудовується в додатки. Так як це система базується на файлах, то вона надає досить широкий набір інструментів для роботи з нею, в порівнянні з мережевими СУБД. При роботі з цією СУБД звернення відбуваються безпосередньо до файлів (в цих файлах зберігаються дані) замість портів і сокетів в мережевих СУБД. Саме тому SQLite дуже швидка, а також потужна завдяки технологіям обслуговуючих бібліотек.

Типи даних SQLite

NULL - значення NULL

INTEGER - знакова цілочисельне значення, використовує 1, 2, 3, 4, 6, або 8 байт в залежності від порядку числа

REAL - число з плаваючою точкою, займає 8 байт для зберігання числа у форматі IEEE

TEXT - текстовий рядок, при зберіганні використовуються кодування UTF-8, UTF-16BE або UTF-16LE

BLOB - тип даних BLOB, масив двійкових даних (призначений, в першу чергу, для зберігання зображень, аудіо та відео).

Переваги SQLite

Файлова структура - вся база даних складається з одного файлу, тому її дуже легко переносити на різні машини

Використовувані стандарти - хоча може здатися, що ця СУБД примітивна, але вона використовує SQL. Деякі особливості є опущеними (RIGHT OUTER JOIN або FOR EACH STATEMENT), але основні все-таки підтримуються

Відмінна при розробці та тестуванні - в процесі розробки додатків часто з'являється необхідність масштабування. SQLite пропонує все що необхідно для цих цілей, тому що складається всього з одного файлу і бібліотеки написаної мовою C.

Встановлено в кожній Android пристрій.

Недоліки SQLite

Відсутність системи користувачів - більш великі СУБД включають в свій склад системи управління правами доступу користувачів. Зазвичай застосування цієї функції не так критично, оскільки ця СУБД використовується в невеликих додатках.

Переваги SQLite

Вбудовані додатки - якщо важлива можливість легкого перенесення програми і не важлива масштабованість. Наприклад однокористувацькі додатки, мобільні додатки або ігри

Прямий доступ до диска - при необхідності безпосередньо звертатися до диска виграш при переході на цю СУБД в функціоналі і простоті використання SQL мови

Тестування - використання додаткових процесів при тестуванні функціоналу, дуже уповільнює додаток.

Коли відмовитися від SQLite

Якщо необхідно забезпечити доступ до даних для декількох користувачів, та й до того ж розрізняти їх з прав доступу, то, напевно, повноцінна СУБД (наприклад: MySQL) буде більш логічним вибором

Запис великих обсягів даних - одне з обмежень SQLite це операції запису. Дозволений тільки один процес запису в проміжок часу, що сильно обмежує продуктивність.

2) MySQL

MySQL - це найпоширеніша повноцінна серверна СУБД. MySQL дуже функціональна, що вільно розповсюджується СУБД, яка успішно працює з різними сайтами та веб додатками.

Незважаючи на те, що в ній не реалізований весь SQL функціонал, MySQL пропонує досить багато інструментів для розробки додатків. Так як це серверна СУБД, додатки для доступу до даних, на відміну від SQLite працюють зі службами MySQL.

Типи даних MySQL

TINYINT - дуже малі цілочисельні значення

SMALLINT - малі цілочисельні значення

MEDIUMINT - середні цілочисельні значення

INT або INTEGER - стандартні цілочисельні значення

BIGINT - великі цілочисельні значення

FLOAT - маленькі значення з плаваючою точкою (точність до одного значення після крапки). Завжди знакові значення

DOUBLE, BOUBLE PRECISION, REAL - Стандартні значення з плаваючою крапкою. завжди знакові

DECIMAL, NUMERIC - розпаковане значення з плаваючою точкою, завжди знакова.

DATE – дата

DATETIME - дата і час в одному значенні

TIMESTAMP - тимчасова відмітка timestamp

TIME – час

YEAR - рік, 2 або 4 числа (4 - по-замовчуванню)

CHAR - строкове значення фіксованої довжини, праворуч завжди додаються прогалини до вказаної довжини при сортуванні

VARCHAR - строкове значення змінної довжини

TINYBLOB, TINYTEXT - значення типу BLOB або TEXT, 255 (2 ^ 8 - 1) символів - максімальния довжина

BLOB, TEXT - значення типу BLOB або TEXT, 65535 (2 ^ 16 - 1) - символів - максімальния довжина

MEDIUMBLOB, MEDIUMTEXT - значення типу BLOB або TEXT, 16777215 (2 ^ 24 - 1) символів - максімальния довжина

LONGBLOB, LONGTEXT - значення типу BLOB або TEXT, 4294967296 (2 ^ 32 - 1) символів - максімальния довжина

ENUM – перерахування

SET - безліч

Переваги MySQL

Простота в роботі - встановити MySQL досить просто. Додаткові програми, наприклад GUI, дозволяє досить легко працювати з БД

Багатий функціонал - MySQL підтримує більшість функціоналу SQL.

Безпека - велика кількість функцій забезпечують безпеку, які підтримується за замовчуванням

Масштабованість - MySQL легко працює з великими обсягами даних і легко масштабується

Швидкість - спрощення деяких стандартів дозволяє MySQL значно збільшити продуктивність.

Недоліки MySQL

Відомі обмеження - за задумом в MySQL закладені деякі обмеження функціоналу, які іноді необхідні в особливо вимогливих додатках.

Проблеми з надійністю - через деякі способи обробки даних MySQL (зв'язку, транзакції, аудити) іноді поступається іншим СУБД по надійності.

Повільна розробка - Хоча MySQL технічно відкрите ПЗ, існують скарги на процес розробки. Варто зауважити, що існують інші досить успішні СУБД створені на базі MySQL, наприклад MariaDB.

Для Android пристроїв необхідно використовувати зовнішній сервер.

Коли слід використовувати MySQL

Розподілені операції - якщо функціоналу SQLite не вистачає, то варто розглянути MySQL. Так як ця СУБД поєднує в собі просунутий функціонал і вільний доступ до вихідного коду.

Високий рівень безпеки - система безпеки MySQL включає в себе прості і в той же час гідні способи захисту доступу до даних

Веб сайти і веб додатки - більшість сайтів і онлайн додатків спокійно працюють з MySQL незважаючи на деякі обмеження. Будучи легкою в налаштуванні і масштабованої системою - MySQL перевірена часом.

Індивідуальні рішення - якщо ви працюєте з яким або специфічним проектом, MySQL легко допомогає завдяки широким можливостям в налаштуванні і функціоналом.

Коли краще відмовитися від MySQL

Відповідність стандартам -Так як MySQL не ставить для себе мету - повністю відповідати стандартам SQL, то ця СУБД неповністю підтримує SQL.

Багатопоточність - хоча деякі движки БД досить легко виконують паралельне читання, паралельні операції читання-запису можуть створити проблеми

Недолік функціоналу - деякі движки MySQL, наприклад, не підтримують повнотекстовий пошук.

3) PostgreSQL

PostgreSQL є самим професійним з усіх трьох розглянутих нами СУБД рішенням. Вона вільно розповсюджується і максимально відповідає стандартам SQL. PostgreSQL або Postgres намагаються повністю застосовувати ANSI / ISO SQL стандарти своєчасно з виходом нових версій.

Від інших СУБД PostgreSQL відрізняється підтримкою затребуваного об'єктно-орієнтованого та / або реляційного підходу до баз даних. Наприклад, повна підтримка надійних транзакцій, тобто атомарность, послідовність, ізоляційні, міцність (Atomicity, Consistency, Isolation, Durability (ACID).) Завдяки потужним технологіям Postgre дуже продуктивна. Паралельність досягнута не за рахунок блокування операцій читання, а завдяки реалізації управління

багатоваріантним паралелізмом (MVCC), що також забезпечує відповідність ACID. PostgreSQL дуже легко розширювати своїми процедурами, які називаються збережені процедури. Ці функції спрощують використання постійно повторюваних операцій.

Хоча PostgreSQL і не може похвалитися великою популярністю у відмінності від MySQL, існує досить велике число додатків, які полегшують роботу з PostgreSQL, незважаючи на всю потужність функціоналу. Зараз досить легко встановити цю СУБД використовуючи стандартні менеджери пакетів операційних систем.

Типи даних PostgreSQL

bigint - знакова 8-ми бітове цілочисельне значення

bigserial - автоматично інкрементіруемое 8-ми бітове цілочисельне значення

bit [(n)] - рядок постійної довжини

bit varying [(n)] - рядок змінної довжини

boolean - логічне значення (true / false)

box - прямокутник на площині

bytea - бінарні дані (масив байтів)

character varying [(n)] - строкове значення змінної довжини

character [(n)] - строкове значення постійної довжини

cidr - IPv4 / IPv6 мережеву адресу

circle - круг на площині

date - календарна дата (рік, місяць, день)

double precision - число з плаваючою точкою подвійної точності (8 байт)

inet - IPv4 / IPv6 адресу хоста

integer - знакова 4-ох байтовое цілочисельне значення

interval [fields] [(p)] - відрізок часу

line - нескінченна пряма на площині

lseg - відрізок на площині

macaddr - MAC адреса

mony - валютне значення

numeric [(p, s)] - точне чисельне значення з обраної точністю

path - геометрична крива на площині

point - геометрична точка на площині

polygony - багатокутник на площині

real - число з плаваючою точкою одинарної точності (4 байти)

smallint - знакове цілочисельне значення (4 байти)

serial - автоматичне інкрементіруєме цілочисельне значення (4 байти)

text - строкове значення змінної довжини

time [(p)] [without time zone] - час доби (без часового поясу)

time [(p)] with time zone - час доби (включаючи часовий пояс)

timestamp [(p)] [without time zone] - дата і час (без часового поясу)

timestamp [(p)] with time zone - дата і час (з часовим поясом)

tsquery - текстовий пошуковий запит

tsvector - документ текстового пошуку

txid_snapshot - користувальницький знімок транзакції з ID

uuid - універсальний унікальний ідентифікатор

xml - XML дані

Переваги PostgreSQL

Відкрите ПЗ відповідне стандарту SQL - PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом. Ця СУБД є дуже потужною системою.

Велика спільнота - існує досить велика спільнота в якому можна знайти відповіді на запитання.

Велика кількість доповнень - незважаючи на величезну кількість вбудованих функцій, існує дуже багато доповнень, що дозволяють розробляти дані для цієї СУБД і керувати ними.

Розширення - існує можливість розширення функціоналу за рахунок збереження своїх процедур.

Об'єктність - PostrgreSQL це не тільки реляційна СУБД, але також і об'єктно-орієнтована з підтримкою успадкування та багато іншого

Недоліки PostgreSQL

Продуктивність - при простих операціях читання PostgreSQL може значно уповільнити сервер і бути повільніше своїх конкурентів, таких як MySQL

Популярність - за своєю природою, популярністю ця СУБД похвалитися не може, хоча і присутня досить велика спільнота.

Хостинг - в силу вище перерахованих факторів іноді досить складно знайти хостинг з підтримкою цієї СУБД.

Коли використовувати PostgreSQL

Цілісність даних - коли надійність і цілісність даних - є вимогами, PostgreSQL буде, мабуть, кращим вибором

Складні користувальницькі процедури - якщо необхідно використовувати користувальницькі процедури, то PostgreSQL має вбудовану підтримку для них

Інтеграція - якщо в майбутньому буде перехід на платні СУБД, наприклад Oracle, то зробити це з PostgreSQL буде досить просто в порівнянні з іншими безкоштовними СУБД

Складна структура даних - порівняно з іншими відкритими СУБД PostgreSQL надає більше можливостей для створення складних структур даних без необхідності жертовать якими або аспектами.

Недоліки PostgreSQL

Швидкість - якщо швидке читання єдиний фактор, то варто вибрати іншу СУБД

Проста настройка - якщо не потрібна цілісність даних, відповідність ACID або складні структури даних, то настройка PostgreSQL є дуже затратною справою.

Висновки

Передача HD-відео та звуку через Wi-Fi зараз у стадії самого найактивнішого розвитку. І постійні новини про Intel WiDi і Apple AirPlay тому підтвердження. Ще зараз подібним проектом зайнялися разом Qualcomm (з Atheros) і Cavium: перша обіцяє з боку своєї мобільної платформи високу потужність і швидкість, а друга - свою реалізацію протоколу передачі

бездротового HD-сигналу, а як відомо, Qualcomm - зовсім не останній виробник на ринку мобільних платформ.

Узагальнюючи, всі намагаються зробити рішення вбудованим, використовуючи вже існуючу апаратну платформу і Wi-Fi адаптер, але, навіть самі останні реалізації поки не досягли абсоютної HD-якості і дуже низької затримки. Вони будуть розвиватися і поліпшуватися за рахунок виходу нового, більш потужного устаткування.

За рахунок проведеного аналізу, можна вважати тему магістерської десертації актуальною для подальших досліджень.

Проаналізувавши існуючі рішення можна зробити такий висновок: Raspberry Pi 2B являє собою ідеальне співвідношення ціни та функціоналу. Кількість підтримуванних операційних систем та зовнішніх модулів робить процесс розробки программного більш комфортним та гнучким. В даній роботі керувати відео потоком будемо за допомогою саме цієї моделі.

Порівнявши велику кількість мов програмування можемо зазначити, що найкраще для поставленої задачі нам підходить мова Java. Тому що, вона є мовою програмування, що реалізує в собі всі необхідні інструменти для швидкого та якісного написання програмного забезпечення. Але, все ж таки, основною причиною для вибору саме цієї мови послужило те, шо програми написані на ній є мультиплатформними на відміну від всіх інших вищерозглянутих мов. Тобто це означає, що написаний код не потрібно перекомпілювати окремо під кожну систему і він буде правильно працювати, тому що він компілюється в байт код , а не машинний код, як на інших мовах. Нижче наведена структурна схема для наглядності цієї особливості мови.

Порівнявши три СУБД, та виходячи з того, що для розроблюємої системи БД буде знаходитись на машині користувача (Android пристрій), та в основному так як розроблюється система моніторинга, то буде більше операцій запису до БД, а не читання з неї. В цьому в повній мірі нас задовольняє SQLite СУБД, так як сама база даних являє собою один файл, в достатній мірі відповідає стандарту

SQL, та взагалі дане рішення є єдиним з вишепредставлених, як СУБД для локального, а не серверного використання.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ТА АЛГОРИТМУ

Виходячи з даного технічного завдання треба реалізувати додаток для Android пристроїв, який буде приймати відео потік з будь-якого мікрокомп'ютера Raspbery Pi. Який включає в себе включає наступні можливості:

1) Вибір доступних мікрокомп'ютерів в мережі для перегляду відео через веб камери, які підключені.
2) Додавання IP адрес в favourite list, щоб можно було без сканування всієї мережі підключатись до пристроїв, які використовуються найчастіше
3) Аналізування швидкісті передачі відео потоку між android пристроєм та мікрокомп'ютером.
4) Програвання кількох відео потоків з різних пристроєв одночасно на одному екрані.

3.1 Розробка алгоритму

Реалізація основного класу програму виходить достатньо простою в результаті правильного проектування архітектури проекту вцілому. Програма реалізована за допомогою шаблонну проектування «модель – представлення – контролер», який забезпечує лаконічність коду, гнучкість програми при додавані або зміні різних можливостей. Шаблон проектування «модель – представлення – контролер» зображений на рис. 3.1

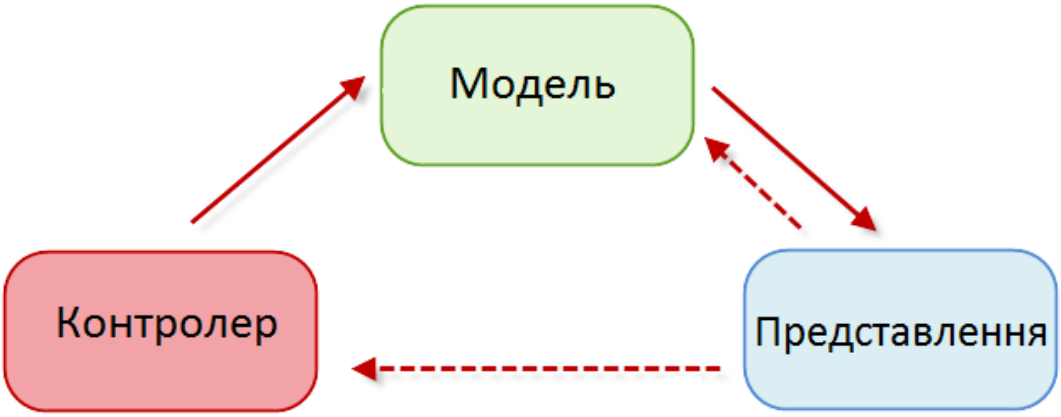


Рис. 3.1 Шаблон проектування «модель – представлення – контролер»

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Архітектурний шаблон Модель-Вид-Контролер (MVC) поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

3.2 Розробка компонентів програми

В основі програми лежить контролер, тобто клас, який відповідає за встановлення з’єднання з мікрокомп’ютером. Так як цей класс запускається один раз при встановленні з’єднання, необхідно реалізувати цей класс таким чином, щоб не можливо було його запустити більше одного разу та попередити дублікаті данних будемо використовувати шаблон проектування, ща моє назву: сінглон (одинак).

Проведемо аналіз даного підходу проектування.

Одинак (англ. Singleton) - породжує шаблон проектування, який гарантує, що в однопоточном додатку буде єдиний екземпляр класу з глобальною точкою доступу.

Гарантує, що у класу є тільки один екземпляр, та надає до нього глобальну точку доступу. Суттєво те, що можна користуватися саме екземпляром класу, так як при цьому в багатьох випадках стає доступною більш широка функціональність. Наприклад, до описаних компонентів класу можна звертатися через інтерфейс, якщо така можливість підтримується мовою.

Глобальний «самотній» об'єкт - саме об'єкт, а не набір процедур, які не прив'язаних ні до якого об'єкту - буває потрібен:

- якщо використовується існуюча об'єктно-орієнтована бібліотека;

- якщо є шанси, що один об'єкт коли-небудь перетвориться на декілька;

- якщо інтерфейс об'єкта (наприклад, ігрового світу) занадто складний, і не варто засмічувати основний простір імен великою кількістю функцій;

- якщо, залежно від яких-небудь умов і налаштувань, створюється один з декількох об'єктів. Наприклад, залежно від того, ведеться лог чи ні, створюється або справжній об'єкт, що пише у файл, або «заглушка», нічого не робить.

Розглянемо Singleton безпосередньо в Java.

Існують декілька різних підходів, реалізації шаблону Singleton, але всі вони мають загальні принципи.

- Private конструктор - для заборони ініціалізації екземпляра класу з іншого класу через конструктор.
- Private static змінну того ж класу, яка і буде єдиним екземпляром цього класу.
- Public static метод, який повертає екземпляр класу. Це - глобальна точка доступу для зовнішнього світу дозволяє отримати екземпляр класу Singleton.

Розглянемо різні підходи реалізацій шаблону Singleton, а також проблеми пов'язані з кожною реалізацією.

1) Рання реалізація (Eager initialization)

У ранній реалізації, екземпляр класу Singleton ініціалізується одночасно із завантаженням класу. Це найпростіший варіант створення класу Одинаку, але він має недолік: екземпляр створюється в будь-якому випадку, навіть якщо їм ніхто так і не скористається.

Покажемо статичну ініціалізацію класу.

EagerInitializedSingleton.java

```
package com.journaldev.singleton;

public class EagerInitializedSingleton {

  private static final EagerInitializedSingleton instance = new
EagerInitializedSingleton();

  // конструктор private, чтобы не было возможности создать экземпляр класса
извне.
  private EagerInitializedSingleton(){}

  public static EagerInitializedSingleton getInstance(){
    return instance;
  }
}
```

Якщо клас не використовує великої кількості ресурсів, то такий підхід виправданий. Однак найчастіше Singleton-класи створюються для роботи з файловими системами, базами даних тощо, коли слід уникати ініціалізації до моменту виклику класом-клієнтом методу getInstance.

Також цей метод неможливо обробляє виключення.

2) Ініціалізація в статичному блоці

Реалізація статичного блоку ініціалізації схожа на «ранню ініціалізацію», за винятком того, що екземпляр класу створюється в статичному блоці, що забезпечує можливість обробки винятків.

StaticBlockSingleton.java

```
package com.journaldev.singleton;

public class StaticBlockSingleton {

  private static StaticBlockSingleton instance;

  private StaticBlockSingleton(){}

  // блок статичної ініціалізації з можливістю обробки виняткових ситуацій
  static{
    try{
      instance = new StaticBlockSingleton();
    }catch(Exception e){
      throw new RuntimeException("При створенні об'єкта «Singleton» сталася
помилка ");
    }
  }

  public static StaticBlockSingleton getInstance(){
    return instance;
  }
}
```

Як рання ініціалізація, так і ініціалізація в статичному блоці створюють екземпляр класу ще до того як він викликається, що не робить їх кращими прийомами.

3) Лінива ініціалізація. (Lazy Initialization)

Лінива ініціалізація - спосіб реалізації шаблону Singleton з глобальним методом доступу до екземпляру класу. Приклад коду для створення класу Singleton з таким підходом.

LazyInitializedSingleton.java

```
package com.journaldev.singleton;

public class LazyInitializedSingleton {

  private static LazyInitializedSingleton instance;

  private LazyInitializedSingleton(){}

  public static LazyInitializedSingleton getInstance(){
    if(instance == null){
      instance = new LazyInitializedSingleton();
    }
    return instance;
  }
}
```

Реалізація вище відмінно працює, якщо використовується тільки один потік, але якщо справа доходить до багатопоточних систем, вона може викликати проблеми, коли всередині циклу одночасно присутні декілька потоків. В цьому випадку втрачається суть шаблону Singleton і два різних потоку отримуватимуть різні об'єкти класу.

4) Поток-безпечний Сінглтон

Найпростіший спосіб створити потоко-безпечний Singleton клас - це синхронізувати глобальний метод доступу за допомогою synchronized, таким чином тільки один потік зможе використовувати даний метод. Загальна реалізація такого класу наведена нижче.

ThreadSafeSingleton.java

```
package com.journaldev.singleton;

public class ThreadSafeSingleton {

  private static ThreadSafeSingleton instance;

  private ThreadSafeSingleton(){}

  public static synchronized ThreadSafeSingleton getInstance(){
    if(instance == null){
      instance = new ThreadSafeSingleton();
    }
    return instance;
  }

}
```

Реалізація вище прекрасно працює і забезпечує потоко-безпеку, але при цьому знижується продуктивність через витрати пов'язаних із синхронізацією, хоча остання необхідна тільки для перших декількох ниток які могли б створити

незалежний екземпляр класу. Щоб прискорити роботу програми, використовується принцип подвійного блокування з перевіркою. При такому підході всередині синхронізуючий блок за допомогою конструкції if проводиться додаткова перевірка, для гарантії, що буде створений тільки один екземпляр класу Singleton.

```
public static ThreadSafeSingleton getInstanceUsingDoubleLocking(){
  if(instance == null){
    synchronized (ThreadSafeSingleton.class) {
      if(instance == null){
        instance = new ThreadSafeSingleton();
      }
    }
  }
  return instance;
}
```

5) Singleton за допомогою Enum.

Java гарантує, що будь-яке Enum значення має тільки один екземпляр в програмі Java. З тих пір як Enum значення можуть мати глобальний доступ вони використовуються в Singleton. Недоліком є те, що повертається Enum тип, кілька негнучкий, наприклад, не підтримує ледачу ініціалізацію.

EnumSingleton.java

```
package com.journaldev.singleton;

public enum EnumSingleton {

  INSTANCE;

  public static void doSomething(){
    //do something
  }
}
```

Проаналізувавши п’ять підходів обираємо спосіб реалізації потокозахищенного сінглтону з подвійним блокуванням з перевіркою, тому що по-перше він потоко захищений, реалізує ліниву ініціалізацію та краще сприймається ніж спосіб реалізації через Enum.

2.3 Зберігання інформації

Зберігати історію, яка містить інформацію про підключення до різних мікрокомп’ютерів будемо зберігати в базі даних SQLite. Для цього реалізовано класс, який реалізує інтерфейс. Це необхідно для того, щоб при необхідності можно було легко змінити базу даних, або зберігати їх в мережі тощо. Інтерфейс повинен мати методи add(), delete(), get().

Розглянемо чому краще використовувати саме такий підхід.

Інтерфейс - семантична і синтаксична конструкція в коді програми, використовувана для специфікації послуг, що надаються класом або компоненто.

Інтерфейс визначає межу взаємодії між класами або компонентами, специфікуючи певну абстракцію, яку реалізує сторона. На відміну від концепції інтерфейсів в багатьох інших областях, інтерфейс в ООП є строго формалізованим елементом об'єктно-орієнтованої мови і в якості семантичної конструкції широко використовується кодом програми.

Інтерфейси дозволяють налагодити множинне спадкування об'єктів і в той же час позбутися ромбовидного спадкування.

Використання інтерфейсів можливо двома способами:

Клас може реалізовувати інтерфейс. Реалізація інтерфейсу полягає в тому, що в описі класу даний інтерфейс вказується, що реалізовується, а в коді класу обов'язково визначаються всі методи, які описані в інтерфейсі, в повній відповідності з сигнатурами з опису цього інтерфейсу. Тобто, якщо клас реалізує інтерфейс, для будь-якого екземпляру цього класу існують і можуть бути викликані всі описані в інтерфейсі методи. Один клас може реалізувати кілька інтерфейсів одночасно.

Можливо оголошення змінних і параметрів методів як мають тип-інтерфейс. У таку змінну або параметр може бути записаний екземпляр будь-якого класу, що реалізовує інтерфейс. Якщо інтерфейс оголошений як тип значення функції, це означає, що функція повертає об'єкт класу, що реалізовує даний інтерфейс.

Як правило, в об'єктно-орієнтованих мовах програмування інтерфейси, як і класи, можуть успадковуватися один від одного. У цьому випадку інтерфейс-

нащадок включає всі методи інтерфейсу-предка і, можливо, додає до них свої власні.

Таким чином, з одного боку, інтерфейс - це «договір», який зобов'язується виконати клас, який реалізує його, з іншого боку, інтерфейс - це тип даних, тому що його опис досить чітко визначає властивості об'єктів, щоб нарівні з класом типізувати змінні. Слід, однак, підкреслити, що інтерфейс не є повноцінним типом даних, так як він задає тільки зовнішню поведінку об'єктів. Внутрішню структуру і реалізацію заданого інтерфейсом поведінки забезпечує клас, який реалізує інтерфейс; саме тому «екземплярів інтерфейсу» в чистому вигляді не буває, і будь-яка змінна типу «інтерфейс» містить екземпляри конкретних класів.

Використання інтерфейсів - один з варіантів забезпечення поліморфізму в об'єктних мовах і середовищах. Всі класи, що реалізують один і той же інтерфейс, з точки зору визначається ними поведінки, поводяться зовні однаково. Це дозволяє писати узагальнені алгоритми обробки даних, що використовують як типів параметри інтерфейсів, і застосовувати їх до об'єктів різних типів, всякий раз отримуючи необхідний результат.

Наприклад, інтерфейс «Cloneable» може описати абстракцію клонування (створення точних копій) об'єктів, специфіковані метод «Clone», який повинен виконувати копіювання вмісту об'єкта в інший об'єкт того ж типу. Тоді будь-який клас, об'єкти якого може знадобитися копіювати, повинен реалізувати інтерфейс Cloneable і надати метод Clone, а в будь-якому місці програми, де потрібна клонування об'єктів, для цієї мети в об'єкта викликається метод Clone. Причому, що використовує цей метод коду достатньо мати лише опис інтерфейсу, він може нічого не знати про фактичне класі, об'єкти якого копіюються. Таким чином, інтерфейси дозволяють розбити програмну систему на модулі без взаємної залежності коду.

Для реалізації збереження даних в розділі 1 було прийняте рішення використовувати SQLite. В Java для реалізації з’єднання та доступу до бази даних використовується стандартна бібліотека JDBC.

JDBC, також як і Microsoft ODBC і Borland DataBase Engine (BDE), базується на X / Open SQL CLI (Call Level Interface). Основне завдання полягає в описі основних абстракцій і концепцій, визначених у X / Open CLI, у вигляді натуральних ("рідних") інтерфейсів Java.

На відміну від класів, що містять в собі як оголошення, так і реалізацію методів, інтерфейси забезпечують більш високий рівень абстракції, описуючи тільки оголошення методів. Враховуючи можливість наслідування, причому спадкування множинного (на відміну від класів), такий підхід дозволяє створювати програми, призначені для роботи з базами даних, які не залежать від конкретної реалізації як самої СУБД, так і методів доступу до неї. Для того щоб звернутися до конкретної СУБД (мається на увазі сервери баз даних), будь це Oracle, Informix, InterBase або щось інше, необхідний JDBC-драйвер.

У концепції універсіалізаціі доступу до даних через стандартні інтерфейси , JDBC-драйвер є сукупність класів, що реалізують JDBC-інтерфейси. Під реалізацією інтерфейсу в Java розуміється створення класу, що посилається в своєму оголошенні на інтерфейс і що пропонує конкретну реалізацію методів інтерфейсу вже у вигляді методів даного класу.

JDBC передбачає передачу запиту до бази даних у вигляді рядка. Таким чином, можуть використовуватися конструкції SQL, специфічні для даної бази даних та / або її JDBC-драйвера.

Визначаючи якийсь універсальний метод роботи з базами даних, JDBC передбачає і конкретні вимоги до базового SQL. Базовою вимогою JDBC (те, що називається JDBC compliantTM), є задоволення вхідного рівня (Entry Level) ANSI-стандарту SQL-92 (SQL-2). Такі вимоги обумовлені наявністю певного понятійного поля, тобто мінімального (аксіоматичного) набору зумовлених сутностей, характерних для РСУБД. Наприклад, за властивостями і поведінці транзакцій, базовим типам даних і т. п.

Збереження по можливості суворої статичної типізації, що повинно забезпечити більшу ступінь захисту від помилок на рівні компіляції в байт-код. Однак ця вимога не є абсолютною, оскільки SQL DML (Data Manipulation

Language) є за своєю природою динамічною. Наприклад, такі сутності, як набори рядків результуючих множин, містять динамічно визначається число і типи стовпців.

Збільшення числа методів інтерфейсів, з одночасною "атомізацією" (спрощенням, навіть мінімізацією) функціональності цих методів. Визначаючи роботу за принципом "одна дія - один метод", цей підхід повинен забезпечити кращу чіткість і прозорість логіки коду. Іншими словами, краще використовувати набір простих методів, ніж багатоцільові (в оригіналі multi-purpose) методи з великими наборами параметрів-прапорів.

java.sql.DriverManager забезпечує завантаження драйверів і створення нових сполук (connection) з базою даних; це стрижневий інтерфейс JDBC, що визначає коректний вибір і ініціалізацію драйвера для даної СУБД в даних умовах;

java.sql.Connection визначає характеристики і стан з'єднання з БД; крім того, він надає засоби для контролю транзакцій і рівня їх ізольованості;

java.sql.Statement виконує функції контейнера по відношенню до

SQL-виразу; при цьому під виразом розуміється не тільки сам текст запиту, а й такі характеристики, як параметри і стан вираження;

java.sql.ResultSet надає доступ до набору рядків, отриманому в результаті виконання даного SQL-вирази.

На рисунку 3.2 представлені основні інтерфейси JDBC.

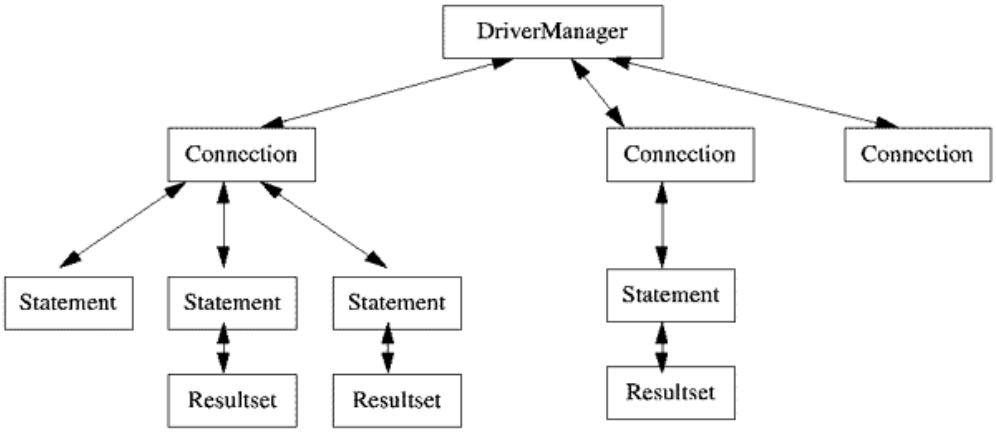


Рисунок 3.2 Основні інтерфейси JDBC.

(*де DriveManager, Connection, Statement, Resultset – назви інтерфейсів)*

Інтерфейс вираження java.sql.Statement виступає як предок для інших двох важливих інтерфейсів: java.sql.PreparedStatement і java.sql.CallableStatement, перший з яких призначений для виконання прекомпілірованние SQL-виразів, другий - для виконання викликів збережених процедур. Відповідно Statement виконує звичайні (статичні) SQL-запити, а зазначені два спадкоємця працюють з параметризованих SQL-виразами.

API (середній рівень) - функцій драйвера для отримання базових інтерфейсів - сполуки, виразів і результуючих даних, а також описового (нижній рівень), що відповідає за конкретну реалізацію JDBC-драйвера, як системно / СУБД-залежної частини (наприклад, як платформно-залежний міст JDBC-ODBC або, повністю написані на Java так звані драйвери JDBC-Net).

Крім перерахованих базових інтерфейсів JDBC описує й інший, не менш важливий специфічний набір інтерфейсів - для роботи з метаданими, тобто зі структурою БД (ми розглянемо їх трохи пізніше). Ці дві групи інтерфейсів є наріжними каменями JDBC.

На рисунку 2.3 показана взаємодія Java коду з JDBC.

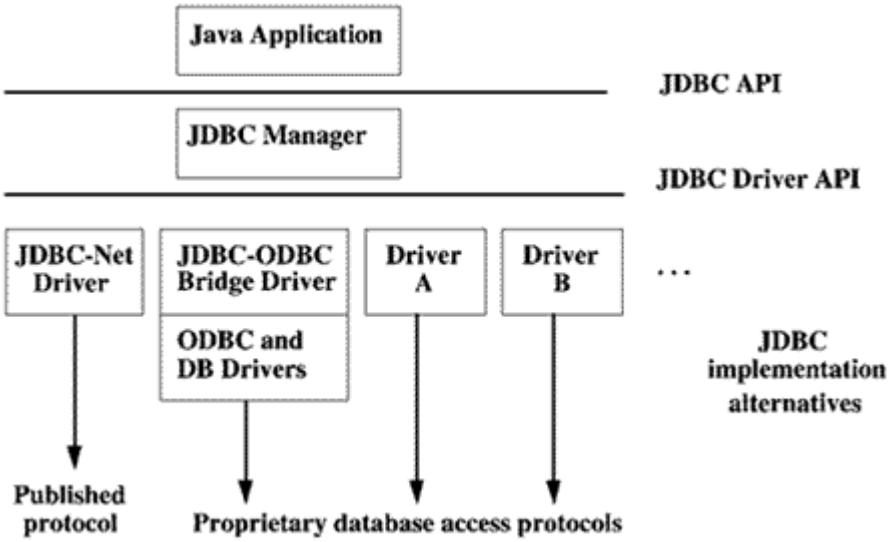


Рисунок 3.3 Взаємодія Java з JDBC

(*де Java Application – додаток Java, implementation alternativs – існуючи реалізаціх драйверів, Pubkished protcol – публічний протокол, Proprietary database access protocols – приватны протоколи)*

3.4 Налаштування Raspberry Pi для передачу відеопотока

Налаштування відбувається досить просто, існуючими інструментами, яке має Raspberry Pi. Для встановлення необхідного пакету програм в терміналі необхідно виконати команду:

```
sudo apt-get install gstreamer0.10-plugins-base gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-tools
```

Далі необхідно ввести:

```
$sudo gst-launch-0.10 v4l2src device=/dev/video0 ! 'video/x-raw-yuv,width=640,height=480, framerate=20/1'! ffmpegcolorspace ! rtpvrawpay ! udpsink host=192.168.0.50 port=4000 sync=false
```

де

device=/dev/video0 — джерело зображення

'video/x-raw-yuv,width=640,height=480, framerate=20/1' — формат

udpsink host=192.168.0.50 port=4000 sync=false — адреса Raspberry Pi.

Відео будемо передавати по протоколу rtp, який працює на транспортному рівні.

3.4.1 Протокол RTP

Поле даних RTP: інформація, пересилається в пакеті RTP, наприклад фрагменти звуку або стислі відео дані.

Пакет RTP: інформаційний пакет, що містить фіксований заголовок. Один пакет транспортного нижнього рівня, наприклад UDP, зазвичай містить один RTP-пакет, але це вимога не є обов'язковим. Поле джерел інформації може бути порожнім.

Пакет RTCP: керуючий пакет, що містить фіксований заголовок подібний RTP, за яким йдуть структурні елементи, які залежать від типу RTCP-пакету. Зазвичай кілька RTCP-пакетів надсилаються як складової RTCP-пакет, вкладений в дейтограмму нижчого рівня.

Транспортна адреса: комбінація мережевої адреси і порту, яка ідентифікує кінцеву точку каналу (наприклад, IP-адреса і UDP порт). Пакети йдуть від транспортного адреси відправника до транспортного адресу одержувача.

Сесія RTP: період з моменту встановлення групи учасників RTP-обміну до її зникнення. Для кожного з учасників сесія визначається конкретною парою транспортних адрес (мережевий адреса і номери портів для RTP і RTCP). Транспортний адреса місця призначення може бути загальним для всіх учасників сесії. Допускається реалізація декількох сесій для кожного з учасників одночасно.

Джерело синхронізації (SSRC): джерело потоку RTP-пакетів, визначається 32-бітним числовим SSRC-ідентифікатором, який записується в заголовок RTP-пакету і не залежить від мережної адреси. Всі пакети від джерела синхронізації утворюють частина з ідентичною тимчасової прив'язкою і нумерацією. Ці дані використовуються приймаючою стороною при відтворенні. Джерелами синхронізації можуть служити первинного джерела сигналу (мікрофони або відеокамери), а також RTP-змішувачі. SSRC-ідентифікатор являє собою випадкове число, яке є унікальним для даної RTP-сесії. Учасник сесії не повинен використовувати один і той же SSRC-ідентифікатор для всіх RTP-сесій мультимедійного набору. Якщо учасник формує кілька потоків в рамках однієї RTP-сесії (наприклад, від декількох відеокамер), кожен учасник повинен бути забезпечений унікальним SSRC-ідентифікатором.

Інформаційне джерело CSRC (contributing source): джерело потоку RTP-пакетів, котрий робить внесок у загальний потік, що формується RTP-змішувачем. Змішувач вставляє список SSRC-ідентифікаторів, які ідентифікують парціальні джерела, в заголовок RTP-пакетів. Цей список називається CSRC-списком. Прикладом програми може бути аудіо-конференція, де змішувач відзначає всіх людей, чий голос породжує вихідні пакети. Це дозволяє приймаючій стороні ідентифікувати мовця, хоча всі пакети мають один і той же SSRC-ідентифікатор.

Кінцева система: програма, яка генерує або сприймає дані, що їх посилають у вигляді RTP-пакетів. Кінцева система може виступати в якості одного або декількох джерел синхронізації для конкретної сесії.

Змішувач: проміжна система, яка отримує RTP-пакети від одного або декількох джерел, при необхідності змінює їх формат, об'єднує і пересилає їх адресатам. Так як тимчасова прив'язка вхідних пакетів може відрізнятися,

змішувач здійснює їх синхронізацію і генерує свій власний потік RTP-пакетів. Таким чином все що пакети мають в якості джерела синхронізації змішувач.

Транслятор: Проміжна система, яка переадресує RTP-пакети, не змінюючи їх ідентифікатори джерела синхронізації. Такі пристрої використовуються для перетворення системи кодування, переходу від мультикастинг- до традиційної уникаст-адресації або при роботі з Firewall.

Монітор: додаток, яке отримує RTCP-пакети, надіслані учасниками RTP-сесії, зокрема діагностичні повідомлення, проводить оцінку стану зв'язку, накопичує довгострокову статистику обміну.

Всі цілочисельні поля передаються згідно з мережевим порядком, тобто старший байт слід першим (big-endian). Порядок передачі докладно описаний у роботі [3]. Якщо не обумовлено зворотного всі цифрові константи є десятковими. Всі поля заголовка вирівнюються їх природним кордонів, тобто. 16-бітові поля мають парне зміщення, а 32-бітні мають адреса, кратний 4. Октети-заповнювачі містять нулі.

Абсолютний час видається з допомогою часових позначок згідно з форматом NTP (network time protocol), який характеризує час у секундах від початку доби (UTC) 1 січня 1900 [4]. Повне дозвіл тимчасової мітки NTP визначається 64-бітовим числом з фіксованою комою без знаку. Цілочисельна частина задається першими 32 бітами, а дробова частина останніми. У деяких полях, де припустимо компактніше подання, використовуються тільки середні 32 біта (16 біт цілочисельна частина і 16 біт дробова).

Структура пакета:

+ Біти	0-1	2	3	4-7	8	9-15	16-31
0	Ver.	P	X	CC	M	PT	Порядковий номер
32	Мітка часу						
64	SSRC-ідентифікатор						
96	... CSRC-ідентифікатори ...						
96+(CC×32)	Додатковий заголовок (необов'язковий), містить довжину блоку даних — «AHL»						
96+(CC×32) + (X×(AHL+16))	Дані						

- Ver. (2 біта) вказує версію протоколу. Поточна версія - 2.
- P (один біт) використовується у випадках, коли RTP-пакет доповнюється порожніми байтами на кінці.
- X (один біт) використовується для зазначення розширень протоколу, задіяних в пакеті.
- CC (4 біти) містить кількість CSRC-ідентифікаторів, що йдуть за постійним заголовком.
- M (один біт) використовується на рівні програми і визначається профілем. Якщо це поле встановлено, то дані пакету мають якесь особливе значення для програми.
- PT (7 біт) вказує формат payload і визначає її інтерпретацію додатком.

SSRC вказує джерело синхронізації.

Висновки

В результаті система реалізована згідно технічного завдання. В себе включає можливість підключення до мікрокомп’ютерів в мережі, зберігання історії підключень, підключення одночасто до декількох пристроїв, транслювання відеопотока на Android пристрої

4 АНАЛІЗ ВПЛИВУ НЕГАТИВНИХ ФАКТОРІВ НА ЯКІСТЬ ВІДЕО ПОТОКУ

4.1 Якість передачі відеопотоку

Одним з найбільш швидко розвиваються сегментів локальних бездротових мереж є ad-hoc мережі, що володіють такими привабливими якостями, як швидкість розгортання, гнучкість структури, мобільність і живучість. Проблеми, пов'язані з організацією ефективної передачі відео в бездротових ad-hoc-мережах, дуже різноманітні і породжуються, головним чином, наступними факторами. По-перше, для такого роду мереж характерна випадкова зміна в часі величини доступної швидкості передачі між передавальним і прийомним вузлами. Це пояснюється як мобільністю користувачів, так і зміною поточної топології мережі. По-друге, передача пакетів в такій мережі характеризується порівняно високим рівнем імовірності неправильного прийому і (або) стирання пакетів. В якості третього чинника слід виділити появу затримок і мережевого джиттера в процесі доставки пакетів, зміна порядку проходження пакетів, а також втрату пакетів внаслідок перевантаження мережі. В інтерактивних сервісах, таких як відеоконференції, допустима наскрізна затримка жорстко обмежена і, як правило, не повинна перевищувати 150 мс. Додаткові проблеми вносить і можлива втрата локальної синхронізації у бітовому потоці компрессированного відео. У зв'язку з тим, що доступна швидкість передачі апріорі невідома і змінюється в процесі передачі, а введення перепитувать призводить, як правило, до неприйнятно великих затримок, раціональний метод передачі відеоконтенту повинен базуватися на адаптації параметрів видеокодека (в першу чергу, швидкості кодування) до поточної доступній швидкості передачі, оцінюваної на основі одержуваної інформації про стан мережі. Інтуїтивно зрозуміло, що для найпростішої топології типу «точка-точка» передача відео зі швидкістю, близькою до доступної швидкості в каналі, забезпечить, мабуть, найкраща якість доставленого відео. У той же час, за наявності проміжних і (або) інтерферирующих вузлів така стратегія передачі, можливо, буде

супроводжуватися помітним зростанням частки втрачених та (або) помилкових пакетів, що призведе до істотного погіршення якості доставленого відео. При зменшенні швидкості кодування відео, з одного боку, знижується навантаження на мережу і тим самим зростають надійність і своєчасність доставки пакетів. З іншого боку, зменшення швидкості кодування супроводжується погіршенням відношення сигнал / шум (ВСШ) в компресованому відео. Крім того, високий ступінь стиснення призводить до більшої чутливості компресованого відео до помилок, що виникають в процесі передачі. Таким чином, можна говорити про свого роду оптимальної (в рамках деякої моделі) швидкості кодування (ступеня стиснення) відеоконтенту, призначеного для передачі в ad-hoc-мережі. У більш загальному контексті виникає завдання вибору швидкості кодування відео, узгодженої з протоколом передачі і поточним станом мережі, що включає такі характеристики, як топологія, метод маршрутизації, кількість проміжних і (або) інтерферуючих вузлів і т.д.

4.1.1 Модель мережі, в якій передається відеопотік

Дослідимо передачу відео в локальній бездротової WiFi-мережі, що функціонує на основі стандарту 802.11b. Номінальна швидкість передачі становить 1, 2, 5 і 11 Мбіт / с в залежності від обраного виду модуляції кодування. Розподіл загального ресурсу мережі між активними користувачами, що базується на процедурі CSMA / CA, підпорядковується правилу, згідно з яким швидкість передачі, що виділяється активному користувачеві, приблизно обернено пропорційна числу таких користувачів. В результаті при декількох активних користувачів в мережі доступна швидкість передачі виявляється помітно менше номінальної швидкості. Схема моделі передачі представлена на рис. 4.1. В якості переданого відео в роботі використовувалися стандартні відео-кліпи Foreman і Ref7, характерною особливістю яких є наявність істотної частки динамічних фрагментів.

Рис. 4.1. Схема моделі передачі

Для створення компресованих відеофайлів використовувався відкритий проект FFmpeg, на основі якого формується велике число мультимедіа додатків. Широкий вибір кодеків, налаштувань процедури кодування, висока швидкість роботи і постійні оновлення роблять FFmpeg ефективним інструментом кодування / декодування відео. У даній роботі FFmpeg використовувався для компресії відео і створення RTP-потоку, а також відновлення (декодування) RTP-потоку. Поточні обмеження в FFmpeg не дозволяють зберігати RTP-пакети у файл або читати їх з файлу. Висновок і введення виробляються UDP-пакетами на адресу і порт, зазначені у файлі протоколу управління сесією (Session Description Протокол - SDP). Допоміжна утиліта зберігає пакети, які будуть використовуватися в подальшому для моделювання процесу передачі. Ця ж утиліта здатна відтворювати збережені RTP-пакети для відновлення відео за допомогою FFmpeg. Для оцінки якості відновленого відео в використовується найбільш поширена метрика - пікове ВСШ (Peak Signal-to-Нойс Ratio - PSNR). Приблизне співвідношення між PSNR і метрикою, що базується на суб'єктивному сприйнятті (Mean Option Score -MOC), наведено в роботі .

4.1.2 Середовище моделювання

В якості моделюючої програми був обраний Network Simulator 3 (NS-3). Цей симулятор дискретних подій призначений для дослідження протоколів мереж (у тому числі Інтернету) і моделювання великих систем. Розробка моделей з використанням NS-3 близька за своєю суттю до створення класичних об'єктно-орієнтованих багатопоточних додатків мовою C ++.

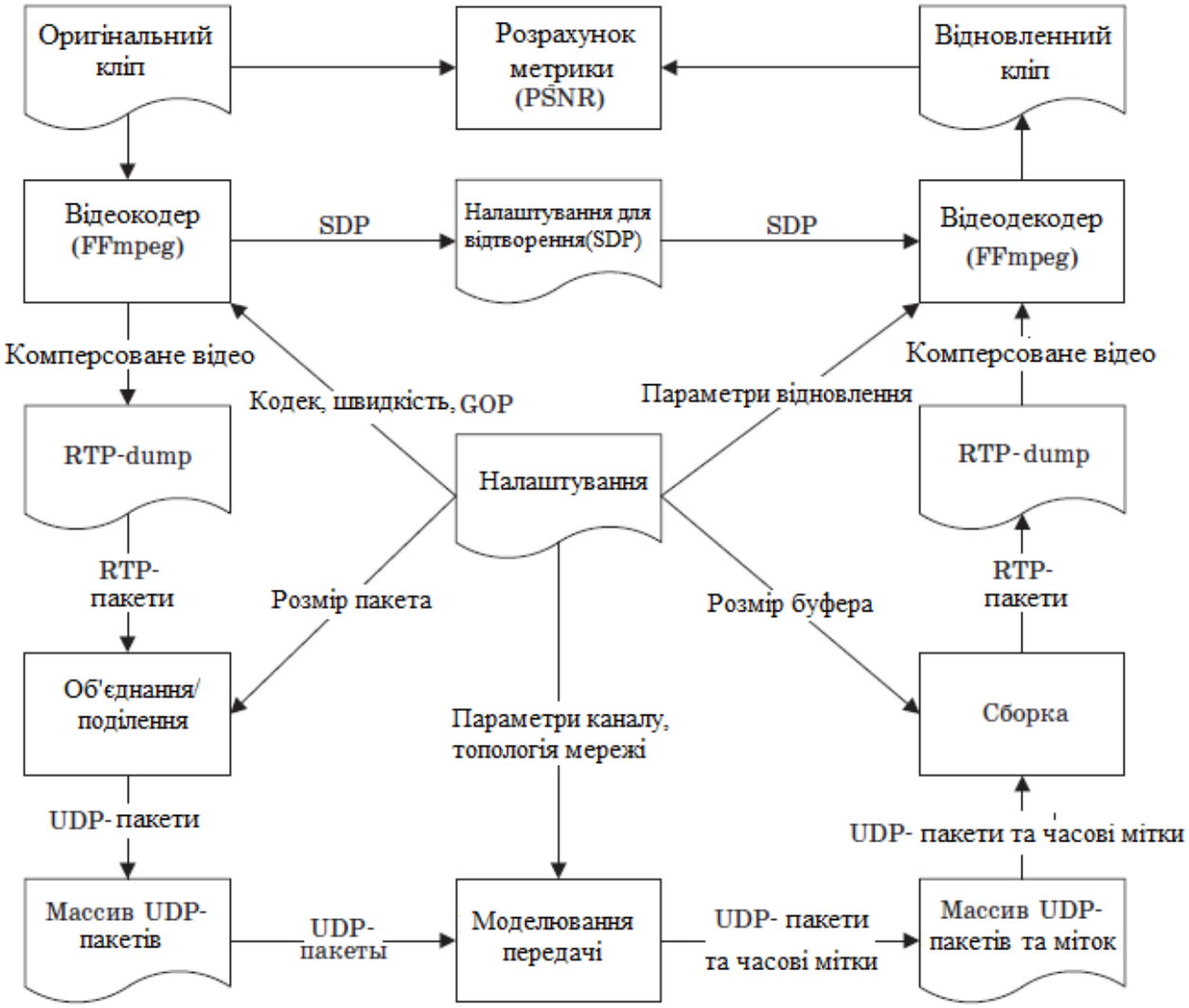


Рис. 4.2 Схема програмної моделі

Можна виділити три основні частини програми:

1.Ініціалізація вузлів мережі (тип трафіку, розташування тощо.), настройка характеристик (швидкості передачі, черги приймачів і передавачів тощо). Вибір топології мережі та її характеристик.

2. Безпосередньо моделювання роботи мережі. Слід зазначити, що моделювання роботи мережі ніяк не зводиться до організації передачі трафіку між вузлами. Під час моделювання можливе включення і (або) видалення вузлів з мережі, зміна інтенсивності трафіку і його типу, переміщення вузлів у просторі та зміна характеристик середовища.

3. Моделювання закінчується збором статистики та її обробкою. Умова закінчення моделювання налаштовується користувачем. Модель передачі відео в ad-hoc-мережі, побудована з використанням симулятора NS-3 (рис. 4.2), включає в себе фіксовані і настроюються параметри. Під параметрами розуміються налаштування всіх програм у комплексі моделювання, а саме NS-3, FFmpeg і утиліти. До фіксованих, тобто жорстко запрограмованим параметрами належать:

- розмір UDP-пакета, рівний 2000 Б;
- мінімальний квант часу, який використовується при описі топології і становить 1 мс;
- протокол Маршрутизації в мережі (в моделі використовується OLSR (Optimized Link-State Routing) - оптимізована маршрутизація станом каналу);
- Стандарт зв'язку - IEEE 802.11b.

Всі вузли вважаються нерухомими, але розташування вузлів при запуску може бути випадковим. До настроюванних параметрів моделі відносяться:

- передані відеокліпи: 1) Foreman (розширення SIF 352 X 288 пікселів з частотою 30 кадрів / с, колірний простір YCbCr 4: 2: 0), тривалість 300 кадрів і 2) Ref7 (розширення DVD 720 X 576 пікселів з частотою 30 кадрів / с, колірний простір YCbCr 4: 2: 0), тривалість 220 кадрів;
- кодек зі MPEG1, MPEG2 і MPEG4;
- швидкість кодування від 200 Кбіт / с до 4000 Мбіт / с;
- порядок слідування кодованих кадрів (group of pictures - GOP). Як правило, GOP-послідовність вибирається з умови, що час між сусідніми кадрами не перевищує 1 с;
- номінальна швидкість передавання V (1; 2; 5,5 і 11 Мбіт / с);
- топологія мережі;

- допустима затримка на приймальній стороні від початку прийому до початку декодування.

Очевидно, багато з настроюваних (а також частково фіксованих) параметрів будуть змінюватися при розгляді інших кодеків (наприклад, VP8), кліпів і бездротових мереж (наприклад, IEEE 802.16s).

4.1.3 Топологія мережі і доступна швидкість

У програмі моделювання використовується стек протоколу IPv4 (з невеликими змінами його можна замінити на IPv6). Всі вузли лежать в одній підмережі 10.1.1.0/24, починаючи з адреси 10.1.1.1 і до 10.1.1.254. Розташування вузлів в одній мережі дозволяє з мінімальними змінами конфігураційних файлів легко налаштовувати топологію. Топологія мережі описується в файлі конфігурації. Кожен рядок файлу, яка не починається зі знака дієз #, описує одну сутність - вузол або з'єднання.

Вузол описується наступним рядком:

node тип_вузла з'єднання тип_координат вектор_координат.

Можливі значення і опис параметрів вузла наведено в табл. 4.1.

Приклад опису вузла:

- node normal 0 const 30 90 0
- node src 2 uniformr 15 30 45 5 0 0

Для завдання параметрів з'єднання використовується схожий синтаксис:

link з'єднання тип_трафіку параметри_трафіку розмір_пакета

Можливі значення і опис параметрів з'єднання наведено в табл. 4.2.

Приклад опису сполук:

- link 0 const 30 1000
- link 2 uniform 20 50 1000

У роботі використовуються дві топології, що ілюструють односкачковую (single-hop) і многоскачковую (multi-hop) ad-hoc-мережі. Топологія 1 тривіальна і

містить два вузли (передавач і приймач), розташованих в прямої видимості один від одного. Така топологія дозволяє отримати результати, які є верхньою межею (в сенсі якості передачі) для інших, більш складних топологій. Топологія 2 включає проміжні та інтерферуючі вузли.

Таблиця 4.1. Значення і опис параметрів вузла

Ім'я параметра	Значення	Опис
Тип вузла	src	джерело пакетів
	dst	приймач пакетів
	normal	проміжний вузол
з'єднання	Ціле число, більше або рівне 0	Номер з'єднання, значення 0 зарезервовано для передачі відео
Тип координат	const	Постійне положення задається трьома координатами по осям X, Y, Z в метрах в параметрі вектор координат
	uniform	Положення вибирається випадково з рівномірним розподілом в інтервалах від Xmin до Xmax, від Ymin до Ymax і від Zmin до Zmax в метрах в параметрі вектор_координат
		Положення вибирається випадково з рівномірним розподілом

	uniformr	в інтервалах від X - Rx до X + Rx, від Y - Ry до Y + Ry і від Z - Rz до Z + Rz в метрах. Таким чином, в параметрі вектор_координат задаються пари центр і радіус для кожної координати
Вектор координат	Вектор дійсних чисел	Див. тип координат

Таблиця 4.2. Значення і опис параметрів з'єднання

ім'я параметра	значення	опис
з'єднання	Ціле число, більше або рівне 0	Номер з'єднання; значення 0 зарезервовано для передачі відео, яке ігнорує нижченаведені параметри
Тип трафіку	const	Постійна інтенсивність, що задається в скалярному параметрі параметри трафіку
	uniform	Випадкова інтенсивність, рівномірно розподілена в інтервалі від tmin до tmax в параметрі параметри

		трафіка
Параметри трафіку	Вектор цілих чисел	Див. Тип_трафіку
Розмір пакету	Ціле число	Ціле позитивне число, яке задає розмір пакета

Топология 1:

node src 0 const 0 0 0

node dst 0 const 30 0 0

link 0 const 30 1000

Топология 2:

node src 0 const 0 0 0

node normal 0 const 30 0 0

node normal 0 const 60 0 0

node normal 0 const 90 0 0

node dst 0 const 120 0 0

node src 2 uniformr 15 30 45 5 0 0

node dst 2 uniformr 45 30 45 5 0 0

node src 1 uniformr 15 30 65 5 0 0

node dst 1 uniformr 45 30 65 5 0 0

link 0 const 30 1000

link 2 uniform 20 50 1000

link 1 uniform 500 600 1000

Як видно з опису топології 2, відео має пройти від вузла (0, 0, 0) до вузла (120, 0, 0) через три проміжних вузла. Два інтерферуючих з'єднання розташовані в безпосередній близькості від проміжних вузлів, що беруть участь у передачі відео. Схематичне зображення топологій 1 і 2 представлено на рис. 4.3(а і б). Відстань між сусідніми вузлами вибрано з урахуванням того, що при

максимальній швидкості 11 Мбіт / с гранична дальність (Tx range), що дозволяє забезпечити прийнятну ймовірність помилки, на практиці близька до 30 м.

Для обох топологій шляхом моделювання передачі UDP-пакетів були визначені значення доступної швидкості передачі при номінальних швидкостях 1; 2; 5,5 і 11 Мбіт / с. За результатами, представленим в табл. 4.3, видно, що доступна швидкість виявляється істотно менше номінальної швидкості навіть для з'єднання «точка-точка».

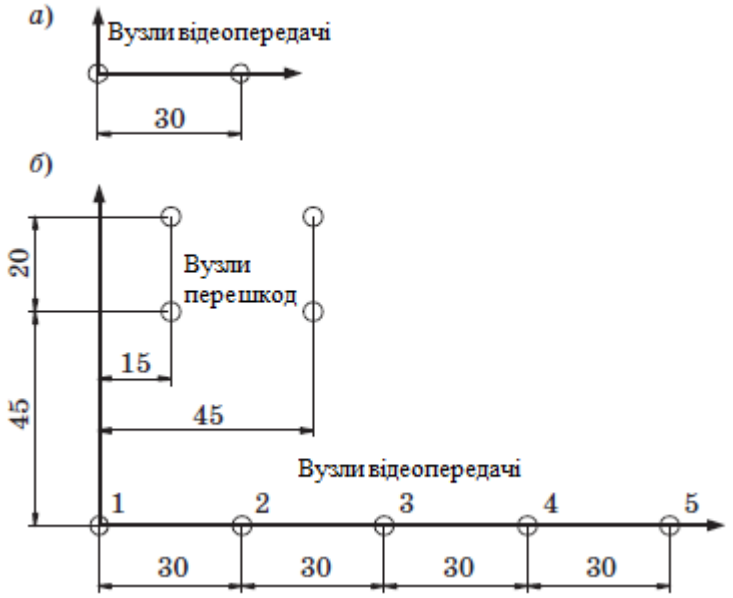


Рис. 4.3. Схематичне зображення топологій 1 (а) і 2 (б)

Це пояснюється, головним чином, тим, що помітну частку в загальному трафіку складає службова інформація, а саме заголовки і RTS / CTS / ACK пакети. Для многоскачковой мережі до зазначеного фактору додається ефект поділу швидкості між частиною вузлів, що в ще більшій мірі знижує доступну швидкість. Більше того, у розглянутій многоскачковой мережі збільшення номінальної швидкості з 1 до 2 Мбіт / с супроводжується зниженням доступної швидкості з 0,42 до 0,21 Мбіт / с.

Цей парадокс пояснюється наступним чином. При номінальних швидкостях 1 і 2 Мбіт / с гранична дальність передавача перевищує 60 м, тому протокол маршрутизації виявляє, що вузол 3 доступний для вузла 1 без проміжного вузла 2 і, відповідно, призначає йому маршрут без проміжного вузла 2. У результаті

відношення сигнал / перешкода на вході приймача вузла 3 падає приблизно на 6 дБ, що виявляється більш критичним при номінальній швидкості 2 Мбіт / с (для неї спочатку відношення сигнал / перешкода на 3 дБ менше, ніж для швидкості 1 Мбіт / с).

4.1.4 Організація передачі RTP-пакетів

Кожен компрессированний кадр відео у форматі RTP розбивається згідно із зазначеними в роботі правилам на n пакетів. Величина n залежить від швидкості компресованого відео (чим вище швидкість, тим більше величина n), типу кадру (ключовий кадр вимагає, в загальному випадку, більшого значення n) і природи відео. За час тривалості кадру τ слід відправити n пакетів

довжини lk, k = 1, 2, ..., n, для одного видеокадра при доступній швидкості передачі C.

$$t = \frac{1}{c}\sum_{k=1}^{n} l_k \qquad (1)$$

Для великих бітових швидкостей більшість RTP -пакетів піддаються розбиттю на пакети максимально можливої довжини lmax; це означає, що час передачі одного кадру можна оцінити зверху у вигляді

$$t \leq \frac{n \cdot l_{max}}{c} \qquad (2)$$

Якщо t ≤ τ, то передача наступного кадру починається через інтервал τ щодо початку передачі поточного. Якщо час передачі поточного кадру виявляється більше τ, то кадри, наступні за поточним, відкидаються до тих пір, поки поточний кадр не переданий. Кількість відкинутих кадрів

$$n_d = \left[\frac{t}{\tau}\right] - 1 = \left[\frac{1}{\tau C} \cdot \sum_{k=1}^{n} l_k\right] - 1 \qquad (3)$$

Використовуючи оцінку (2) у виразі (3), отримуємо верхню межу для кількості відкинутих кадрів

$$n_d \leq \tau C \left[\frac{n \cdot l_{max}}{\tau C}\right] - 1 \qquad (4)$$

дозволяє, у свою чергу, оцінити ступінь погіршення якості прийнятого відео. Зазначена стратегія гарантує відправку поточного кадру і, як показує аналіз, у поєднанні з відповідною процедурою маскування пропущених кадрів в більшості ситуацій виявляється більш ефективною з точки зору якості, ніж часткова відправка декількох кадрів.

4.1.5 Результати моделювання

У процесі моделювання обидва відеокліпа кодувалися трьома відеокодеками (MPEG1, MPEG2 і MPEG4) зі швидкостями кодування (компресії) в діапазоні 200 - 4000 Кбіт / с (для кліпу Foreman) і 1000 - 4000 Кбіт / с (для кліпу Ref7). Відзначимо, що для обох кліпів навіть на нижній межі діапазону швидкостей кодування забезпечується відмінне або гарна якість за шкалою MOS. Кодоване відео передавалося по мережі з можливими номінальними швидкостями V = 1; 2; 5,5 і 11 Мбіт / с.

Топологія 1. На рис. 4.4 представлені результати обчислення ВСШ (PSNR) в декодованому кліпі Foreman для зазначених значень доступних швидкостей. В якості верхньої межі на рис. 4.4 (і наступних малюнках) використовується функція «PSNR - швидкість кодування», відповідна відеокодеку MPEG4. Для кліпу Ref7 результати, що характеризують PSNR в декодованому відео, показані на рис. 4.5.

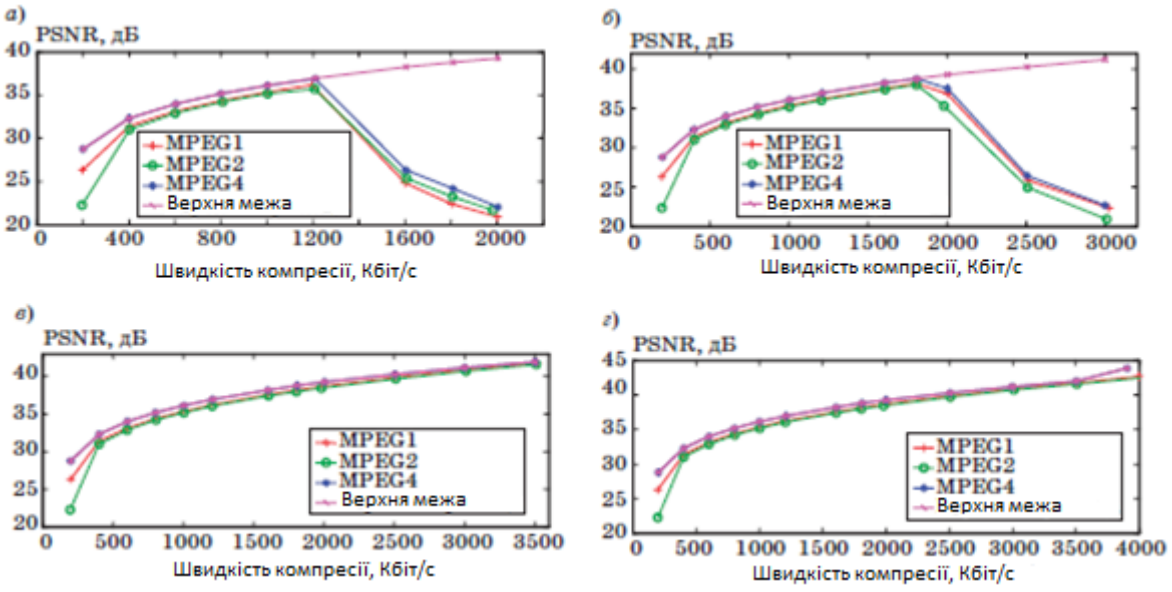


Рис.4.4 PSNR в декодованому відео: топологія 1, кліп Foreman, номінальна швидкість V = = 1 Мбіт / с (а); 2 Мбіт / с (б); 5,5 Мбіт / с (в); 11 Мбіт / с (г)

За результатами можна зробити висновки, що, по-перше, при номінальних швидкостях 1 і 2 Мбіт / с оптимальна (в сенсі максимуму PSNR) швидкість кодування R0 кілька перевершує доступну швидкість C.

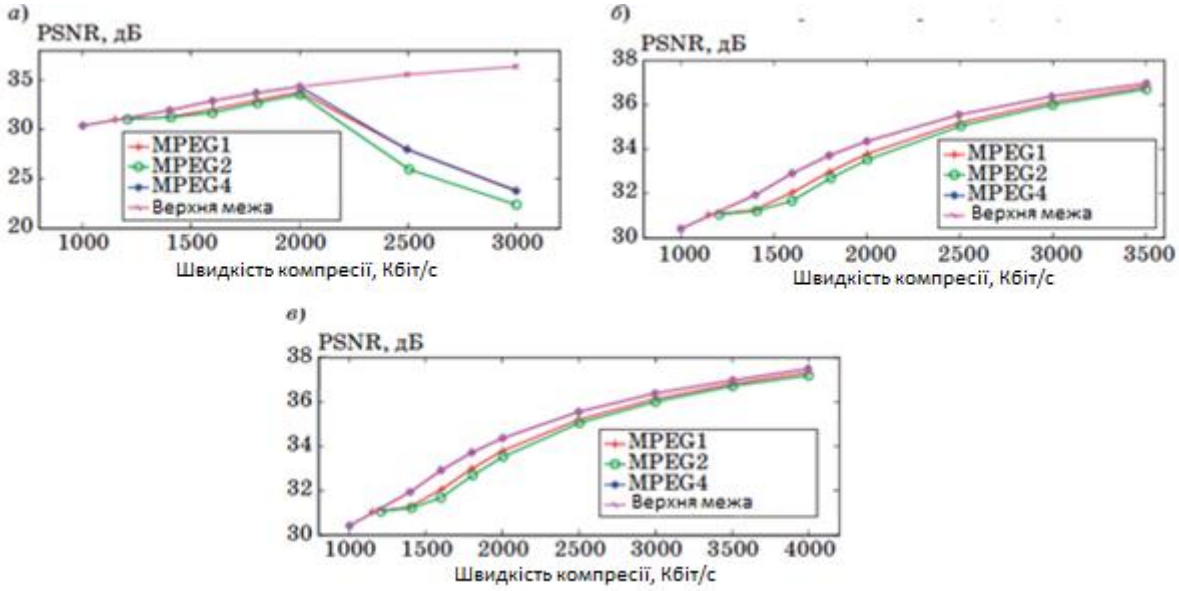


Рис.4.5 PSNR в декодованому відео: топологія 1, кліп Ref7, номінальна швидкість V = 2 Мбіт / с (а); 5,5 Мбіт / с (б); 11 Мбіт / с (в)

Вплив інтерферуючої перешкоди і проміжних вузлів проявляється наступним чином. Интерферуючий трафік, по-перше, споживає якусь частину доступної швидкості передачі в мережі (в силу механізму доступу CSMA / CA) і, по-друге, спотворює сигнал що передається, приводячи до додаткових помилок в передачі. Проміжні вузли, в свою чергу: а) також займають якусь частину доступній швидкості передачі; б) можуть відкидати пакети через переповнення буфера і в) можуть пропускати відправку кадру, якщо попередній кадр ще не завершив передачу. Рис. 3.6 ілюструє вплив перерахованих факторів при передачі кліпу Foreman. Для мережі з топологією 2 характерне наступне (див. Рис. 3.6). По-перше, оптимальна швидкість кодування R0 знову опиняється близька до доступної швидкості C (відношення R0 / C знаходиться в інтервалі від 0,95 до 1,04); по-друге, програш (щодо варіанту передачі в топології 1 становить 5-6 дБ.

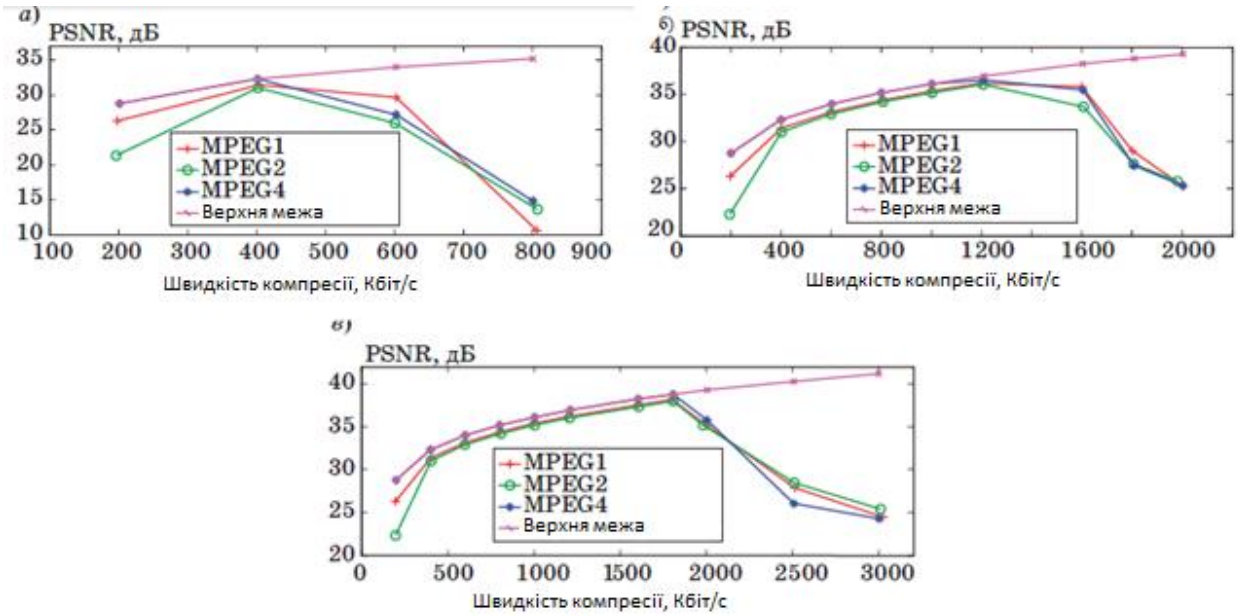


Рис 4.6. PSNR в декодованому відео: топологія 2, кліп Foreman, номінальна швидкість V = 1 Мбіт / с (а); 5,5 Мбіт / с (б); 11 Мбіт / с (в)

Для кліпу Ref7 передача навіть з найбільшою компресією з розглянутого діапазону (швидкість кодування R = 1 Мбіт / с) супроводжувалася неприйнятною якістю при номінальних канальних швидкостях 1 і 2 Мбіт / с. Результати моделювання для швидкості 11 Мбіт / с показані на рис. 3.7. Як бачимо, тут

ставлення R0 / C = 1,15, а програш (відносно варіанти передачі в топології 1 складає близько 5 дБ.

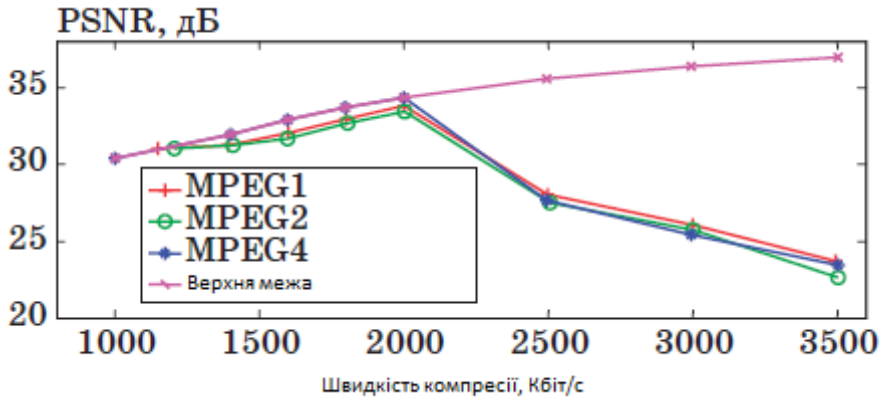


Рис.4.7 PSNR в декодованому відео: топологія 2, кліп Ref7, номінальна швидкість V = 11 Мбіт / с

Таким чином, проведені експерименти показують, що:

1) для обох топологій оптимальна швидкість кодування R0 виявляється близька до доступної швидкості передачі C

2) збільшення дальності передачі в 4 рази, що досягається при використанні многоскачкової мережі з інтерферуючиви вузлами, супроводжується погіршенням PSNR на 5 - 6 дБ (при цьому якість залишається відмінною за шкалою MOS).

Сумарний час затримки**.** Аналіз причин затримки при доставці відеокадрів в рамках топології 2 показав наступне. Затримки виникають внаслідок витрат часу на передачу повідомлення, його очікування у черзі та необхідність ретрансляції на інший вузол. Сусідні вузли можуть займати загальне середовище, збільшуючи час на передачу. Основним фактором, що визначає величину затримки, є співвідношення швидкості кодування і доступною швидкості передачі. Для оцінки впливу цього співвідношення були розглянуті три варіанти передачі з номінальною швидкістю передачі V = 1 Мбіт / с, що відповідає доступній швидкості C = 420 Кбіт / с. У першому варіанті швидкість кодування R становила 200 Кбіт / с. На рис. 3.8 наведено поточні значення сумарного часу затримки при доставці відеокадрів для кліпу Foreman.

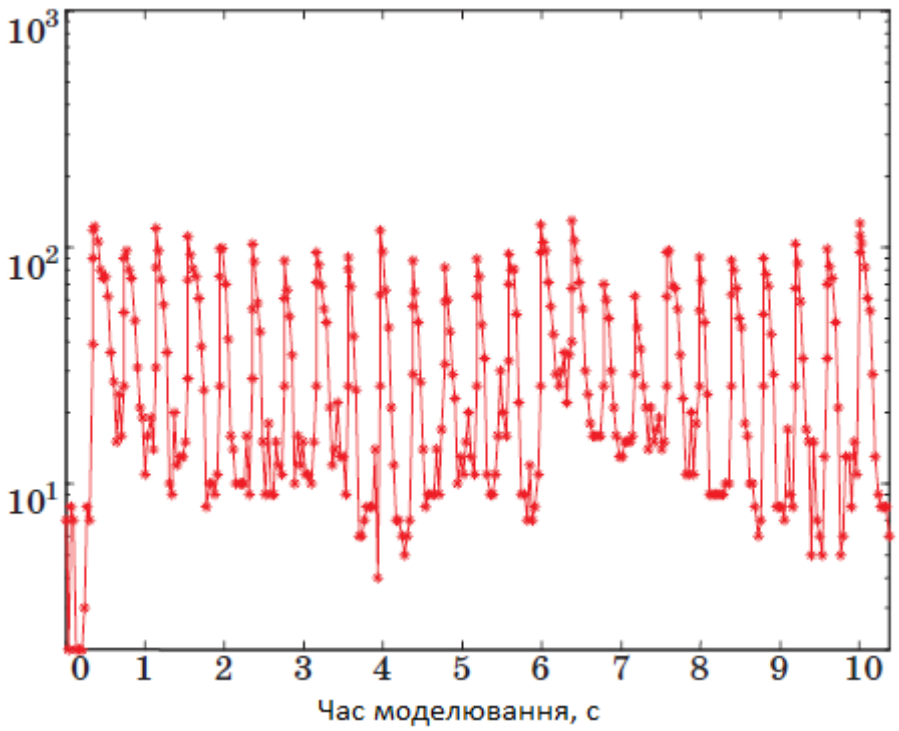


Рис. 4.8. Поточний сумарний час затримки для кліпу Foreman: топологія 2, швидкість кодування R = 200 Кбіт / с, номінальна швидкість V = 1 Мбіт / с

Як видно з графіка, максимальне значення затримки не перевершує 150 мс, а джіттер знаходиться в прийнятному діапазоні, так що буфер відповідного розміру забезпечить плавне відтворення відео. При невеликому збільшенні швидкості кодування (до 230 Кбіт / с) затримка різко збільшилася майже в 7 разів, досягнувши 1с. , при R = 400 Кбіт / с затримка протягом першої ж секунди з моменту початку передачі стала лавиноподібно наростати і досягла 8с. Така затримка допустима, якщо мова йде просто про доставку відео. Разом з тим для інтерактивних додатків прийнятна швидкість кодування, як бачимо, виявилася приблизно вдвічі менше величини C, що, в свою чергу, призвело до зниження PSNR приблизно на 3 дБ.

4.2 Втрати відеопотока у мережі Wi-Fi

Оскільки передача потового відео відбувається по мережі Wi-Fi, то ми стикаємося з проблемою завад в цьому діапазоні. Діапазон 2,4 ГГц

використовується також мікрохвильовими печами, бездротовими телефонами та іншим обладнанням, не кажучи про Wi-Fi, то існує величезна кількість потенційних джерел перешкод. Проблем від побутового обладнання можуть бути не такими сильними, то кілька сусідніх бездротових мереж зможуть зайняти весь доступний діапазон частот. В результаті багато хто з тих, хто використовує бездротову мережу, навряд чи зможуть користуватися її ресурсами навіть для web-серфінгу, читання електронної пошти, чатів та ігор, не кажучи вже про такі вимогливих до смуги пропускання додатках, як потокове відео. Більшість користувачів стикаються із завантаженістю діапазону. Очевидно, що багатоквартирні будинки мають найбільший потенціал для прояву цієї проблеми. Але навіть у передмістях, де щільність жителів не така висока, ризик перекриття кількох сусідніх мереж все ж є. Навіть якщо мереж немає, то завжди залишається ризик, що діапазон буде періодично перевантажуватися різним побутовим обладнанням.

Ми прийшли до висновку, що в більшості випадків діапазон 2,4 ГГц, використовуваний обладнанням 802.11b / g, виявиться занадто переповненим, щоб використовувати його для додатків потокового відео. Що ж таке перевантаженість діапазону і чим вона відрізняється від простого браку швидкості на бажаній відстані?

Ось три основних ефекти перевантаженості спектра.

- Зменшення смуги пропускання

- Безліч сусідніх мереж намагаються використовувати одні й ті ж частоти. Це загрожує не тільки зниженням швидкості роботи, але й істотними її коливаннями, оскільки всі сусідні мережі намагаються отримати шматок загального каналу.

- Великі втрати пакетів

- Високий рівень "шуму", створюваний сусіднім каналом і джерелами, відмінними від Wi-Fi (мікрохвильові печі, телефони та інше), може стати причиною зниження швидкості, оскільки при втраті або пошкодженні пакету

відбувається його повторна пересилка. У гіршому випадку, з'являються невиправні помилки.

- Нестабільне підключення

- Користувачі мереж Wi-Fi в місцях де їх багато знайомі з проблемою підключення до чужої мережі. Це пов'язано з тим, що сигнал сусідньої точки доступу або маршрутизатора виявляється сильнішим сигналу власної. Це доповнюється залишеними за замовчуванням налаштуваннями вбудованої утиліти конфігурування бездротової мережі в Windows.Очевидно, що при виході за мінімальну смугу пропускання, або якщо з'єднання буде постійно пропадати, то додаток не працюватиме.

4.2.1 Емуляція помилок

Компанія Apposite Technologies розробила пристрій Linktropy 4500 WAN Emulator (рис. 4.9), який є апаратним емулятором, який дозволяє виконувати налаштування швидкості вхідного і вихідного потоку, затримки і втрати. Для налаштування використовується web-інтерфейс, для доступу до якого потрібно підключитися до спеціального порту Ethernet.

Рис. 4.9. Apposite Technologies Linktropy 4500 WAN Emulator.

Для роботи емулятор потрібно підключити його між будь-якими двома сегментами Ethernet LAN. Для підключення на емуляторі є два порти 10/100/1000: LAN A і B. За замовчуванням пристрій працює як прозорий інтелектуальний міст з будь-яким протоколом. Однак, при бажанні, його можна перемкнути в режим

маршрутизатора для роботи з IP, але в цьому режимі не підтримується робота з трафіком multicast-додатків. При підключенні на інтерфейсі управління задаються бажані параметри мережі (рис. 4.10). Емуляцію можна ввімкнути або вимкнути за допомогою спеціальної кнопки на інтерфейсі (Emulation On / Off) без зміни інших налаштувань.Як уже згадувалося раніше, можна налаштовувати пропускну здатність, затримки і втрати для всього трафіку, що проходить через 4500 в обох напрямках. Затримка (delay) - єдиний параметр, який може не тільки бути фіксовано заданим (Constant), але і може динамічно змінюватися з рівномірним (Uniform) або нормальним (Normal) статистичним розподілом. Взагалі, непогано було б отримати подібну динамічну зміну параметрів втрат і, можливо, пропускної здатності. Тоді ми змогли б краще емулювати бездротове з'єднання. Втім, подивитися на ефект від втрати кадрів можна і в статичному режимі.

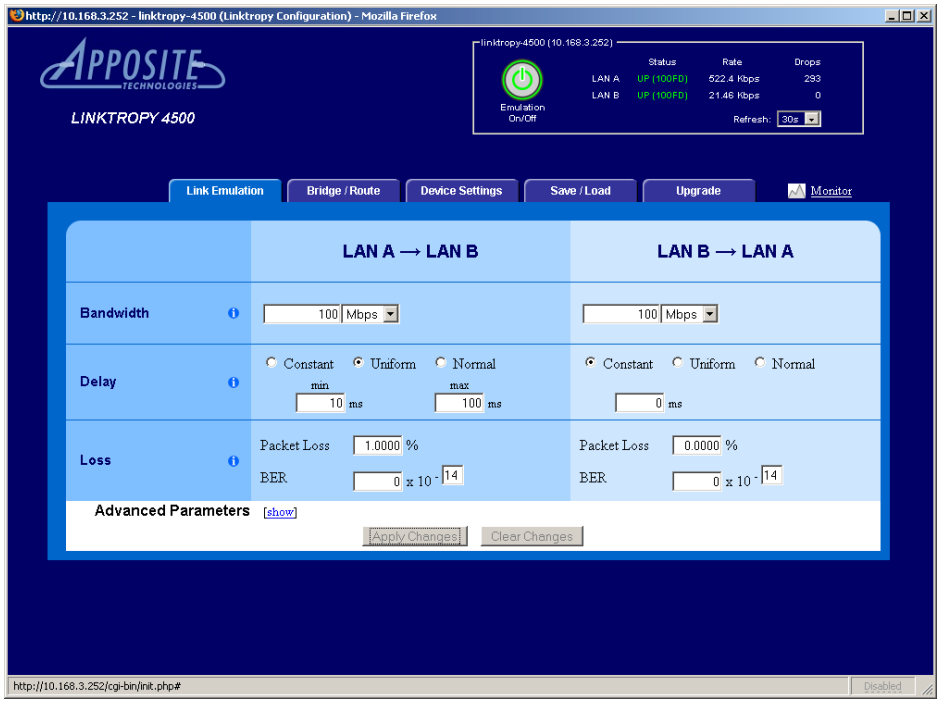


Рис. 4.10. Екран емуляції з'єднання Link Emulation.

Емулятор може відображати статистику трафіку для обох напрямів (але тільки при включеній емуляції), а також будувати графік швидкості (рис. 4.11)

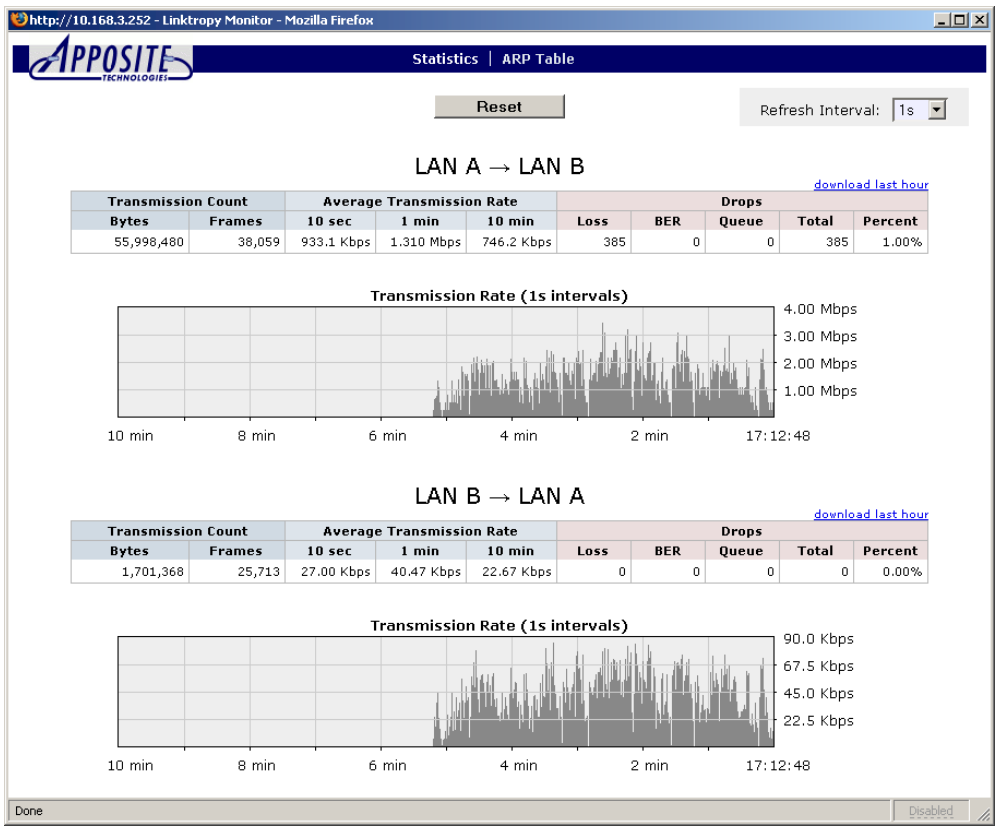


Рис. 4.11. Екран статистики з'єднання Link Статистика.

4.2.2 Розмір потоку і кодування

Існує два способи відтворення віддалених медіа-файлів. Можна просто взяти ПК або інший пристрій, здатний працювати з локальними і мережевими файлами. Тоді достатньо буде знайти в мережі і запустити на відтворення потрібний файл. Він буде відтворюватися через ту мережеву файлову систему, яку використовує ваша ОС. У більшості випадків це буде SMB (Server Message Block), що працює на верхніх рівнях стека TCP / IP.

Інший спосіб - використовувати для відтворення медіа-сервер і протокол потокового мовлення, який доставлятиме медіа-потік від сервера до плеєра. Для передачі потоку використовуються такі протоколи, як RTP і RTCP, що працюють поверх UDP.

Відмінність у тому, що TCP / IP забезпечує надійну доставку, а UDP - ні, оскільки TCP має вбудовані механізми контролю доставки та цілісності даних. Однак TCP не можна назвати кращим рішенням для передачі мультимедіа, оскільки цей протокол додає в пакети даних велика кількість службової

інформації. Для TCP головне - безпомилково передати дані, а час доставки стоїть на другому місті.

З іншого боку, UDP використовує набагато менше службових даних, ніж TCP, тому він краще підходить для додатків, що працюють з потоковими даними, де на перший план виходить час доставки інформації. Що стосується пропусків і спотворень пакетів, то вирішення цієї проблеми покладається на приймаючу сторону.

Ми протестували обидва описаних вище способи, використовуючи для цього три тестових файли:

-DVD VOB, кодек MPEG 2;

-файл DivX з кодеком DX50 в дозволі 640x300 при 24 кадрах в секунду;

-PSP avc1 (MPEG 4), дозвіл 368x208 при 30 кадрах в секунду.

Для мовлення та відтворення потоку ми використовували VideoLAN VLC Media player 0.8.5. Вибір припав на плеєр VLC, тому що він дозволяє працювати з усіма перерахованими типами файлів і може як просто відкривати медіа-файл, так і відтворювати потік з іншого VLC-плеєра, що працює в серверному режимі. Мовлення виконувалося в режимі VLC UDP при настройках за умовчанням без перекодування. Розмір потоку (бітрейт) безпосередньо впливає на якість відтворення, від нього також багато чого залежить і те, чи можна буде дивитися відео по мережі. Розмір потоку можна дізнатися у властивостях файлу, однак багато кодеків використовують динамічнозмунюваний бітрейт, тому навіть вказаному значенню іноді не слід вірити. Щоб дізнатися реальний бітрейт відтвореного файлу, потрібно взяти утиліту, яка показує потік даних, що проходить через мережевий адаптер. Ми використаємо Hoo Technologies Net Meter. Утиліта дозволяє практично все, що можна очікувати від вимірювача швидкості (крім кнопки очищення графіка; для цього доводилося перезапускати утиліту). На рис. 4.12 і 4.13 показані графіки, отримані при відтворенні перших чотирьох хвилин тестового VOB-файлу в обох режимах.

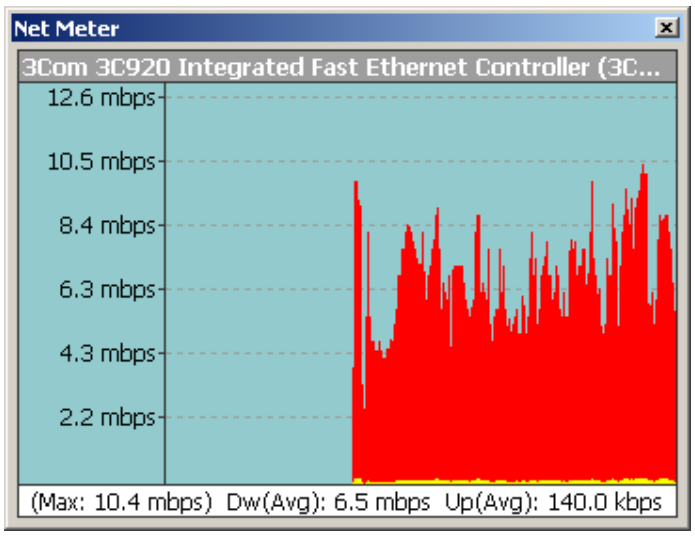


Рис. 4.12. Графік для відтворення файлу VOB (MPEG 2).

Для VOB значення максимальної та середньої швидкості при відтворенні потоку виявилися приблизно на 20% нижче, ніж при відтворенні файлу. При цьому відношення максимальної швидкості потоку до середньої в обох випадках склало близько 1,6.

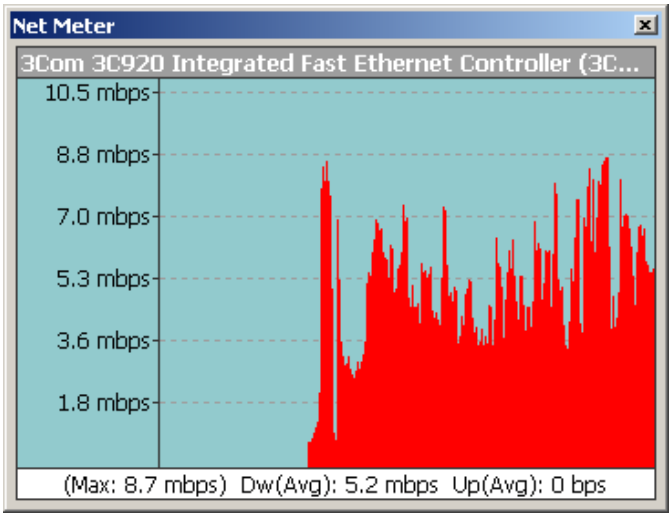


Рис. 4.13. Графік для відтворення VOB (MPEG 2) в потоковому режимі.

На рис. 4.14 і 14.15 показані графіки, отримані при відтворенні тестового DivX файлу в обох режимах.

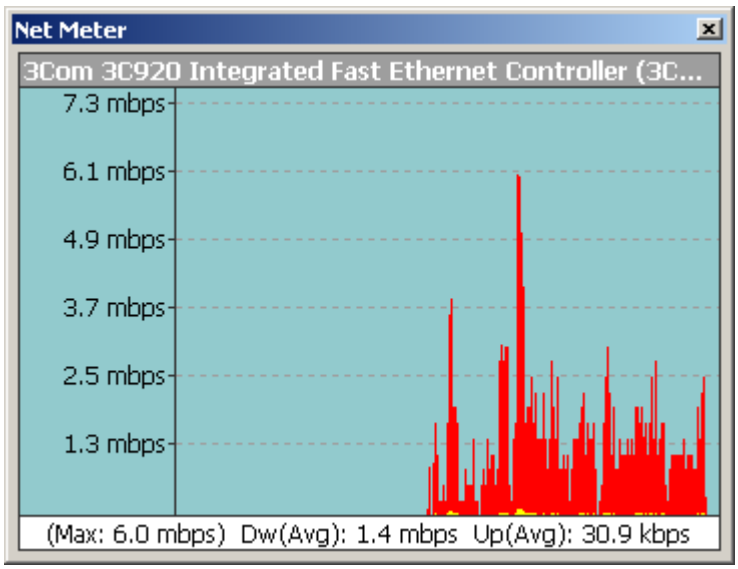


Рис. 4.14. Графік для відтворення файлу DivX.

Цього разу максимальний бітрейт при відтворенні файлу DivX виявився приблизно на 20% нижче, ніж при відтворенні потоку, при цьому середні значення практично не відрізнялися. Відношення максимальної швидкості до середньої при файловому режимі склало 4,3, в потоковому - 4,8, максимум з усіх тестів.

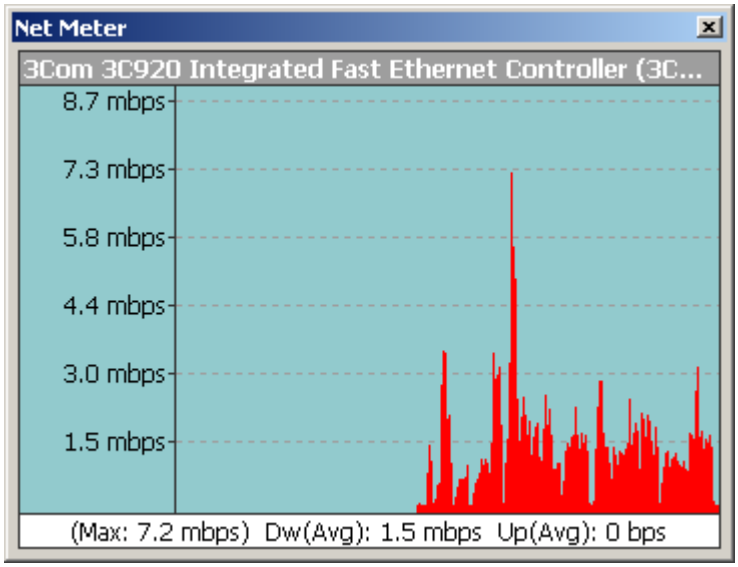


Рис. 4.15. Графік для відтворення DivX в потоковому режимі.

І, нарешті, на рис. 4.16 і 4.17 показані графіки для PSP / MPEG4.

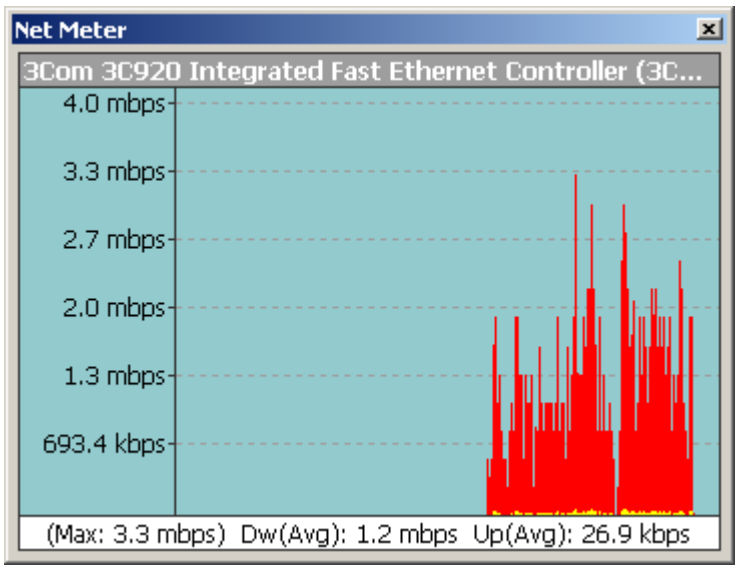


Рис. 4.16. Графік для відтворення файлу PSP.

І знову середня швидкість для потоку і для файлу виявилася практично однаковою, а максимальна для потоку виявилася нижче, ніж для файлу, приблизно на 10%. Відношення максимальної швидкості до середньої склало 2,8 для файлу і 2,2 для потоку.

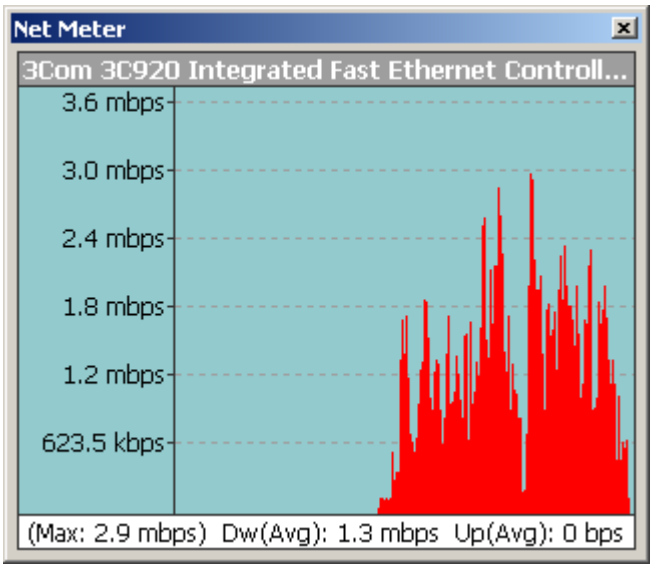


Рис. 4.17. Графік для відтворення потоку PSP.

Оскільки в тестах використовувалися різні фільми, результати не можна використовувати для порівняння кодеків. Зате за графіками наочно видно, що процес передачі мультимедіа-файлу і потоку не можна описувати одним значенням бітрейту.

4.2.4 Чутливість до втрат пакетів при потокової передачі

Для початку ми встановили втрати в 1% для потокового режиму, в результаті побачили спотворення (квадрати). Потім ми запустили серію тестів, результати яких показані в таблицях нижче. Також наведемо скріншоти артефактів, отриманих при відтворенні.

Таблиця 4.3. Результати для потоку VOB

Втрати пакетів	Результат
5%	Неможливо дивитися
1%	Неможливо дивитися
0,5%	Дивитися можна, помітні квадрати
0,25%	Дивитися можна, часто проскакують невеликі квадрати
0,1%	Дивитися можна, квадрати проскакують рідше
0,05%	Дивитися можна, квадрати проскакують дуже рідко

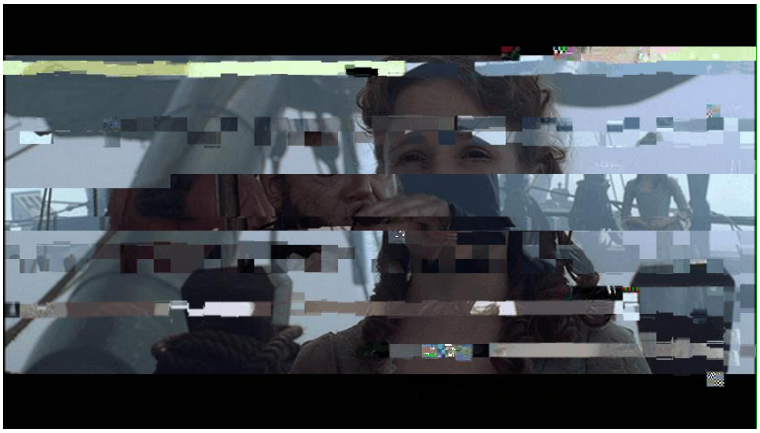


Рис. 4.18. Потік VOB при втратах 5%.

Таблиця 4.4. Результати для потоку DivX.

Втрати пакетів	Результат
5%	Неможливо дивитися, відео та звук перериваються
1%	Неможливо дивитися, постійно проскакують квадрати
0,5%	Дивитися можна, але складно, квадрати

0,25%	Дивитися можна, часто проскакують невеликі квадрати
0,1%	Дивитися можна, квадрати проскакують рідше
0,05%	Дивитися можна, квадрати проскакують дуже рідко

Рис. 4.19. Потік DivX з 1% втрат.

Таблиця 4.5. Результати для потоку PSP / MPEG4.

Втрати пакетів	Результат
5%	Неможливо дивитися, плеєр не працює
1%	Неможливо дивитися, але плеєр працює
0,5%	Дивитися можна, але складно
0,25%	Дивитися можна, періодично проскакують квадрати
0,1%	Дивитися можна, іноді проскакують квадрати
0,05%	Дивитися можна, спотворення з'являються рідко

Рис. 4.20. Потік PSP / MPEG4 з втратами 0,05%.

4.2.5 Чутливість до втрат пакетів при відтворенні файлів

Після тестування потокового режиму ми вирішили повторити тести при файловому відтворенні і отримали зовсім інші результати. Як і слід було очікувати, через здатність TCP / IP коригувати помилки передачі, тут набагато складніше домогтися проблем з відтворенням відео. А якщо проблеми і виникнуть, то у вигляді пауз або пропусків, але не як спотворення картинки. Щоб відео відмовилося відтворюватися, нам довелося вдатися до затримок с рівномірним розподілом і втрат пакетів. При втратах 1% пакетів і затримках між 10 і 500 мс відтворення VOB-файлу періодично завмирало і, врешті-решт, через кілька хвилин зупинився повністю. При тих же налаштуваннях і тому ж ролику, але вже стислому у формат DivX (якість Home Theater Quality), паузи стали рідше, і ми змогли додивитися ролик до кінця

Тут ми прийшли до наступних висновків:

- якщо відтворювати файли через TCP / IP, то чутливість до втрат пакетів виявляється менше, ніж з протоколом UDP;

- бітрейт і спосіб кодування впливають на сприйнятливість до помилок передачі.

Висновок

Навіть незначні втрати пакетів призводять до того, що потокове відео стає просто неможливо дивитися. Навпаки, якщо використовувати протокол TCP / IP, то створити перешкоди під час перегляду набагато складніше.

4.3 Огляд видів кодування звуку та проблеми при його передачі

Аудіокодеком називають програму або алгоритм, який стискає, або розтискає цифрові звукові дані, дозволяючи знизити вимоги до пропускної здатності каналу передачі даних. На сьогоднішній день найбільш поширене перетворення допомогою кодека G.729, а також стиснення G.711 по А-закону (alaw) і μ-закону (ulaw).

G.729 є кодеком, який стискає вихідний сигнал з втратою даних. Основна ідея, закладена в G.729 - передача не самого оцифрованого сигналу, а його параметрів (спектральної характеристики, кількості переходів через нуль), достатніх для подальшого синтезування на приймаючій стороні. При цьому всі основні характеристики голосу, такі як амплітуда і тембр зберігаються.

Пропускна здатність каналу, на яку розрахований даний кодек - 8 кбіт / с. Довжина кадру оброблюваного G.729 - 10 мс, частота дискретизації - 8 кГц. Для кожного з таких кадрів визначаються параметри математичної моделі, які в подальшому і передаються в канал у вигляді кодів.

При використанні кодування G.729 затримка становить 15 мс, з яких 5 мс витрачається на заповнення попереднього буфера.

G.711 - голосовий кодек, який передбачає ніякого стиснення, крім компадування - методу зменшення ефектів каналів з обмеженим динамічним діапазоном. В основі цього методу лежить принцип зменшення кількості рівнів квантування сигналу в області високій гучності, зберігаючи при цьому якість звуку. Зараз широко використовуються два види кодування - alaw і ulaw. Сигнал в даному кодеку надано потоком величиною 64 кбіт / с. Частота дискретизації - 8000 кадрів по 8 біт в секунду. Якість голосу суб'єктивно краще, ніж при застосуванні кодека G.729.

Alaw або А-закон - алгоритм стиснення звукових даних з втратою інформації. В основному використовується на території Європи та Росії.

Для сигналу x перетворення за алгоритмом alaw виглядає наступним чином

$$F(x) = sgn(x)\begin{cases} \dfrac{A|x|}{1+\ln(A)}, & |x| < \dfrac{1}{A} \\ \dfrac{1+\ln(A|x|)}{1+\ln(A)}, & \dfrac{1}{A} \le |x| \le 1, \end{cases}$$

Де A - параметр стиснення (звичайно приймається рівним 87,7).

Ulaw або μ-закон - алгоритм стиснення звукових даних з втратою інформації. В основному використовується на території Японії та Північної Америки.

Для сигналу x перетворення за алгоритмом ulaw виглядає наступним чином:

$$F(x) = sgn(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \quad -1 \le x \le 1$$

де μ приймається рівним 255 (8 біт) в стандартах Північної Америки та Японії.

Імпульсно кодова модуляція - передача неперервної функції у вигляді серії послідовних імпульсів.

Для отримання на вході каналу зв'язку модульованого сигналу, миттєве значення несучого сигналу вимірюється АЦП з певним періодом. При цьому кількість оцифрованих значень в секунду (інакше, частота дискретизації) повинно бути більшим або рівним дворазової максимальній частоті в спектрі аналогового сигналу.

Далі отримані значення округлюються до одного з заздалегідь прийнятих рівнів. Зауважимо, що кількість рівнів необхідно приймати кратним ступеня двійки. Залежно від того, скільки було визначено рівнів, сигнал кодується певною кількістю біт.

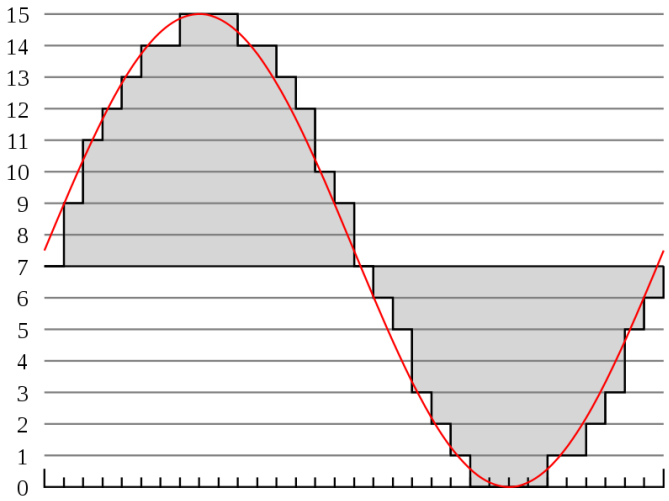


Рис. 4.21 Квантування сигналу

На даному малюнку представлено кодування за допомогою чотирьох бітів (тобто всі проміжні значення аналогового сигналу заокруглюватимуть до одного з наперед заданих 16 рівнів). Для прикладу, при часу рівному нулю сигнал буде представлений подібним чином: 0111.

При демодуляції послідовність нулів і одиниць перетворюється в імпульси демодулятором, рівень квантування якого дорівнює рівню квантування

модулятора. Після цього ЦАП на основі даних імпульсів відновлює сигнал, а згладжує фільтр остаточно прибирає неточності.

Число рівнів квантування має бути більшим або рівним 100, тобто мінімальна кількість біт, яким може кодуватися сигнал - 7.

Джитер - небажані фазові та/або частотні випадкові спотворення при передачі сигналу. Виникають внаслідок нестабільності тактового генератора, змін параметрів лінії передачі в часі і різної швидкості поширення частотних складових одного і того ж сигналу.

Обумовлюється джіттер трьома факторами:

-Обмежена смуга пропускання або некоректна робота активних мережевих пристроїв;

-Висока затримка поширення сигналу;

-Тепловий шум.

Найбільш часто застосовується метод боротьби з джиттером - джіттер-буфер, який зберігає певну кількість пакетів.

Зазвичай передбачається динамічне підстроювання довжини буфера протягом усього часу існування з'єднання. Для вибору найкращої довжини використовуються евристичні алгоритми.

Для компенсації нерівномірної швидкості надходження пакетів на приймальній стороні створюють тимчасове сховище пакетів, або так званий джиттер буфер. Його завдання, зібрати пакети, що поступають в правильному порядку відповідно до часових відміток і видати їх кодеку з правильними інтервалами і правильному порядку.

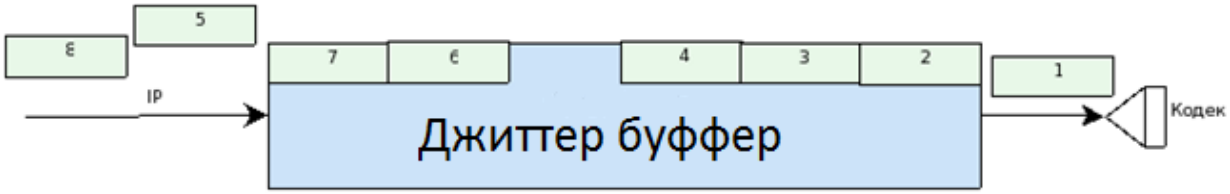


Рис.4.22 Джиттер буфер

Розмір буфера розраховується в процесі роботи, або примусово задається в настройках. З одного боку він не може бути занадто великим, щоб не збільшувати транспортну затримку. З іншого боку, маленький розмір буфера викликає втрати пакетів при змінах часу затримки в IP мережі.

На практиці втрата більше 1% викликає певні неприємні відчуття. При 2% розмова виявляється ускладеною. При значеннях більше 4% розмова вже практично неможлива.

Висновоки

У даній роботі аналізувалося якість передачі відео в односкачковій і багатоскачкової ad-hoc-мережах. Ключовим фактором, що визначає якість передачі, є відношення швидкості кодування відео R до доступної швидкості передачі в мережі C. З точки зору максимізації PSNR у доставленому відео, оптимальне значення відношення R / C виявилося близьким до 1 для обох топологій. Збільшення дальності передачі вчетверо, що досягається при переході від односкачкової топології до розглянутої в роботі багатоскачкової топології з интерферуючими вузлами, супроводжувалося погіршенням величини PSNR на 5-6 дБ. Тим не менш, якість доставленого відео залишалося відмінною за шкалою MOS. В інтерактивних додатках (з допустимою затримкою близько 150 мс) оптимальне відношення R / C при передачі в багатоскачковій мережі виявилося істотно менше 1: в наших експериментах це ставлення було близько до 0,5. Як наслідок, PSNR у доставленому відео додатково знизилося приблизно на 3 дБ. Одним з інструментів поліпшення якості передачі відео в ad-hoc-мережах представляється вдосконалення використаного в роботі алгоритму маршрутизації OLSR з тим, наприклад, щоб виключити появу парадоксів, аналогічних тому, що був відзначений при обговоренні отриманих при моделюванні доступних швидкостей передачі в мережі (див. табл. 3). Бажано, зокрема, щоб алгоритм маршрутизації враховував не тільки доступність вузлів, але і рівень відношення сигнал / шум на вході вузла. Це дозволить трафіку з високою чутливістю до втрат пакетів проходити по більш надійним каналам, нехай і з великим числом

проміжних вузлів. Для менш чутливого трафіку або трафіку з низькою інтенсивністю можна використовувати «прямий» маршрут з гіршим значенням сигнал / шум на вході вузла. Така модифікація протоколу маршрутизації дозволить, як видається, збільшити доступну швидкість передавання в багатоскачковій ad-hoc-мережі.

Тестування привело нас до наступних висновків:

-для появи проблем з потоковим відео досить дуже малих втрат пакетів;

-при втратах пакетів бітрейт і спосіб кодування практично не впливають на якість відтворення.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Так як дана робота має суто теоретичний характер і виконувалась із застосуванням засобів обчислювальної техніки, то в даному розділі визначені основні потенційно небезпечні і шкідливі фактори при використанні візуальних дисплейних терміналів персональних електронно-обчислювальних машин (ВДТ ПЕОМ) відповідно до вимог ДсанПіН 3.3.2.007-98 та ДНАОП 0.00-1.21-99 та запропоновані технічні рішення та організаційні заходи з безпеки та гігієни праці і виробничої санітарії, а також визначені основні заходи з безпеки в надзвичайних ситуаціях (БНС).

5.1 Визначення основних потенційно небезпечних і шкідливих факторів при використанні ВДТ ПЕОМ

Приміщення, в якому відбувалася розробка програмного забезпечення та його тестування, знаходиться на другому поверсі адміністративної будівлі. Загальна площа становить 45 кв.м, висота – 2,8 м. Кількість працюючих у відділі – 6 чоловік. Отже, на одного працюючого в приміщенні припадає 45:6=7,5 (89в.м/89вто.) робочої площі. Згідно із СниП 2.09.04 – 87 на кожного працюючого в управлінських приміщеннях повинно припадати не менше 4 (89в.м/89вто.) робочої площі. Висота приміщення – не менше 2,5м. Отже, нормативи розмірів та забезпечення працюючих робочою площею в офісі дотримано.

План приміщення, в якому проводилася розробка програмного забезпечення та його тестування, показаний на рис. 5.1.

У приміщенні розташовано 6 комп’ютерів. Напруга джерела живлення комп’ютерів у приміщенні – 220 В. У приміщенні розміщено 6 письмових столів, одна шафа для зберігання документів. За небезпекою ураження електричним струмом приміщення відділу належить до приміщень без підвищеної небезпеки ураження електричним струмом працюючих.

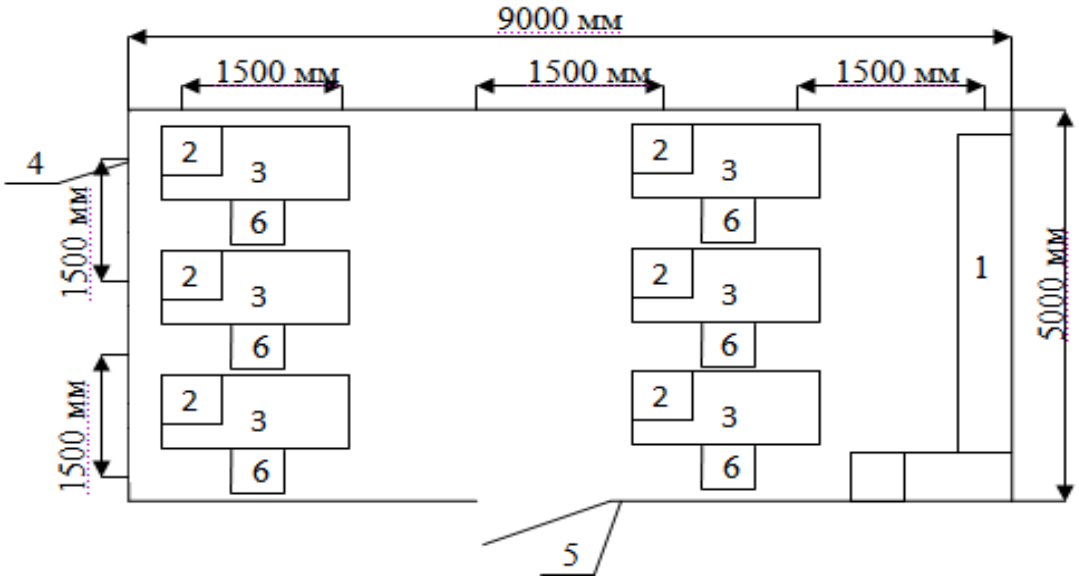


Рис. 5.1. План приміщення

Умовні позначення:

1. шафи для зберігання документів;
2. комп’ютери;
3. стіл;
4. вікно;
5. двері;
6. стілець.

При роботі з ВДТ ПЕОМ можна виділити наступні потенційно небезпечні і шкідливі фактори:

- підвищена напруга в електромережі;
- м’яке рентгенівське випромінювання ВДТ;
- електромагнітне випромінювання;
- ультрафіолетове і інфрачервоне випромінювання;
- електростатичне поле між екраном і оператором;
- наявність пилу, озону, оксидів азоту й аероіонізації;
- недостатнє освітлення приміщення та наявність відблисків від екрану монітора;
- неоптимальні параметри мікроклімату (температура, відносна вологість, швидкість руху, рівень іонізації повітря тощо).

5.2. Технічні рішення та організаційні заходи з безпеки та гігієни праці і виробничої санітарії

5.2.1 Відповідність параметрів мікроклімату робочої зони існуючим санітарним вимогам

Порівняємо фактичні значення параметрів мікроклімату: температури повітря, відносної вологості, швидкості переміщення повітря з нормованими значеннями, встановленими відповідно до нормативних документів.

Фактичні значення одержували за допомогою безпосередніх власноручних вимірів, а також використовували готові дані вимірів в санітарно-технічних паспортах досліджуваних приміщень.

Відповідно до ГОСТ 12.1.005-88 роботи, які проводяться в приміщенні, відносяться до категорії 1а (легкі роботи з затратами енергії до 150 ккал/год). Швидкість переміщення повітря у приміщенні знайдемо за формулою:

$$V_{\text{п}} = \frac{L_{\text{ф}}}{\text{висота}\cdot\text{глибина}\cdot\mu\cdot 3600} = \frac{2679{,}831}{2{,}8\cdot 5\cdot 0{,}55\cdot 3600} = 0{,}096 \text{ м/с}$$

Зведемо дані у таблицю 5.2.5:

Таблиця 5.2.5.

Підсумкова таблиця

Параметр	Значення параметра		Нормативний документ
	фактичне	нормоване	
1) Освітленість штучна (лк)	378	300	СниП II-4-79
2) Значення коефіцієнта природного освітлення (%)	0,5897	1,35	СниП II-4-79
3) Температура повітря (^{o}C):			
взимку	18-20	21-25	ДСН 3.3.6.042-99
влітку	24	22-28	ДСН 3.3.6.042-99
4) Відносна вологість повітря (%)			
взимку	60	<75	ДСН 3.3.6.042-99
влітку	55	<60	ДСН 3.3.6.042-99
5) Повітрообмін ($м^3$/год)			
взимку	2679,831	90	СниП 2.09.04-87

влітку	1704,78	90	СниП 2.09.04-87
6) Швидкість переміщення повітря м/с	0,096	<0,2	ДСН 3.3.6.042-99

5.2.2 Розрахунок природної вентиляції щодо її відповідності існуючим санітарним вимогам

При дослідженні достатності вентиляції в адміністративно-управлінських приміщеннях слід мати на увазі, що відповідно до ДБН 3.2.5-28-2006 об`єм виробничого приміщення, яке доводиться на одного працюючого, повинен складати не менше 15 м^3. Інакше для нормальної роботи в приміщенні необхідно забезпечити постійний повітрообмін за допомогою вентиляції не менше ніж L_1=30 м^3/год на кожного працюючого. Таким чином, потрібний повітрообмін $L_н$, м^3/год, розраховується за формулою:

$$L_н = L_1 \cdot n = 30 \cdot 6 = 180(м3/год),$$

де n=6 – найбільша можлива кількість працюючих у приміщенні.

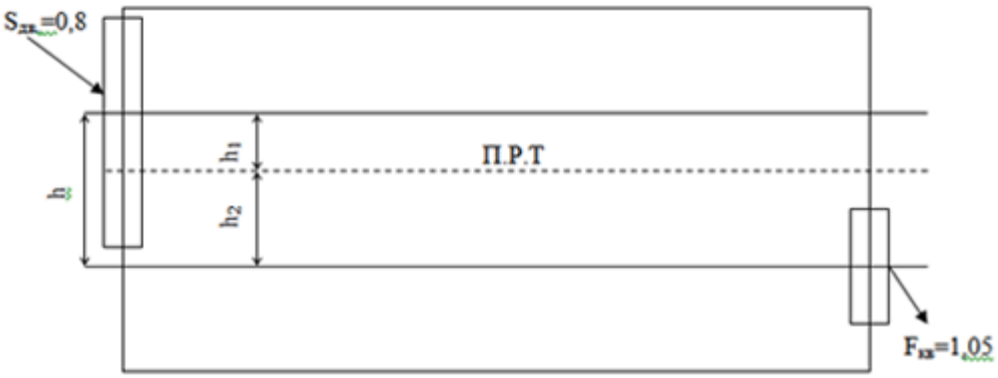


Рис. 5.2.3.1. Схема розрахунку природної вентиляції

Знайдемо фактичне значення повітрообміну, користуючись схемою, наведеною на рисунку 5.2.3.1, за формулою:

$$L_ф = \mu \cdot F \cdot V \cdot 3600\ ,$$

де $\mu = 0{,}55$ – коефіцієнт витрати повітря;

$F_{кв}$ – площа кватирки ($F_{кв}$ = 1,4·0,75=1,05 м^2);

V – швидкість виходу повітря через кватирку або вентиляційний канал, м/с, яка розраховується за формулою:

$$V = \sqrt{\frac{2 \cdot g \cdot \Delta H_2}{\gamma_c}},$$

де g=9,8 м/с^2 – прискорення вільного падіння;

ΔH_2 – тепловий тиск, під дією якого буде виходити повітря з кватирки. Розраховується за формулою:

$$\Delta H_2 = h_2 \cdot (\gamma_з - \gamma_{вн}),$$

де $\gamma_з, \gamma_{вн}$ – відповідно об'ємна вага повітря ззовні приміщення та всередині його, кг/м3.

Об'ємна вага повітря розраховується за формулою:

$$\gamma = 0{,}465 \cdot \frac{P_б}{T},$$

де $P_б$=750 мм.рт.ст – барометричний тиск;

Т – температура повітря в К (для теплового періоду року у приміщенні t=28 ^{o}С, або Т=310К, для холодного періоду року t =-17 ^{o}С або Т=290 К; ззовні приміщення для літа t=24 ^{o}С або Т=297 К, для зими t=-11 ^{o}С, або Т=262 К (СниП 2.04.05-91).

$\Gamma_{с(л)}$ = 0,465·750/310 = 1,125; $\gamma_{с(з)}$ = 0,465·750/290 = 1,203;

$\gamma_{з(л)}$ = 0,465·750/297 = 1,174; $\gamma_{з(з)}$ = 0,465·750/262 = 1,331;

Знайдемо h_2 із співвідношень: $\frac{h_1}{h_2} = \frac{F_{кв}{}^2}{S_{дв}{}^2}$;

h=0,8+1,4/2=1,1 (м)

$F_{кв}{}^2$=(1,05)·2=1,1025 (м^2)

$S_{дв}{}^2$=(2·0,8)·2=2,56 (м^2)

Розв'яжемо систему:

$$\begin{cases} h_1 + h_2 = h \\ \frac{h_1}{h_2} = \frac{F_{кв}{}^2}{S_{дв}{}^2} \end{cases} => \begin{cases} h_1 + h_2 = 1{,}1 \\ \frac{h_1}{h_2} = \frac{1{,}1025}{2{,}56} \end{cases} => \begin{cases} h_1 = 1{,}1 - h_2 \\ 1{,}1 - h_2 = 0{,}431 \cdot h_2 \end{cases} =>$$

$$=> \begin{cases} h_1 = 1{,}1 - h_2 \\ 1{,}431 \cdot h_2 = 1{,}1 \end{cases} => \begin{cases} \text{h2} = 0{,}8 \text{ (м)} \\ \text{h1} = 0{,}3 \text{ (м)} \end{cases}$$

Звідси:

H_2(л) = 0,8·(1,174-1,125) = 0,039; H_2(з) = 0,8·(1,331 – 1,203) = 0,102;

Vл = $\sqrt{\frac{2 \cdot 9{,}8 \cdot 0{,}039}{1{,}125}} = 0{,}82$ (м/с); Vз = $\sqrt{\frac{2 \cdot 9{,}8 \cdot 0{,}102}{1{,}203}} = 1{,}289$ (м/с);

$L_ф$(л) = 0,55·0,7·0,82·3600 =1704,78 ($м^3$/год)

$L_ф$(з) = 0,55·0,7·1,289 ·3600 = 2679,831 ($м^3$/год)

Оскільки фактичне значення повітрообміну значно перевищує нормативне значення як взимку, так і влітку, то природна вентиляція(аерація) ефективна.

5.2.3 Неіонізуючі електромагнітні випромінювання, електростатичні та магнітні поля ВДТ ПЕОМ

В даному приміщенні використовуються рідкокристалічні монітори, тому гранично допустима напруженість електростатичного поля на робочих місцях не повинна перевищувати рівнів, наведених в ГОСТ 12.1.045 “ССБТ. Электростатические поля. Допустимые на местах и требования к проведению контроля», СН № 1757-77 “Санитарно-гигиенические нормы допустимой напряженности электростатического поля”.

Рівні інфрачервоного випромінювання не перевищують граничних відповідно до ГОСТ 12.1.005 та СН № 4088-86 з урахуванням площі тіла, яка опромінюється, та ДсанПіН 3.3.2-007-98.

Рівні ультрафіолетового випромінювання не перевищують допустимих відповідно до СН № 4557-88 «Санітарні норми ультрафіолетового випромінювання у виробничих приміщеннях», затверджених Міністерством охорони здоров‘я СРСР, та ДсанПіН 3.3.2-007-98.

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05 м від екрана та корпусу при будь-яких положеннях регулювальних пристроїв відповідно до Норм радіаційної безпеки України (НРБУ-97), затверджених постановою державного санітарного лікаря Міністерства охорони здоров’я

України від 18.08.97 №58, не повинна перевищувати 7,74..12 А/кГ, що відповідає еквівалентній дозі 0,1 мбер/год (100 мкР/год).

Відповідно до ГОСТ 12.1.005-88 вміст озону в повітрі робочої зони не повинен перевищує 0,1 мг/м3; вміст оксидів азоту – 5 мг/м3; вміст пилу – 4 мг/м3.

5.2.4 Розрахунок штучного освітлення на робочих місцях користувачів ВДТ ПЕОМ щодо його відповідності існуючим санітарним вимогам

При дослідженні штучного освітлення треба аналогічно природному освітленні порівняти фактичне значення з нормованим.

Нормоване значення освітлення для адміністративно-управлінських приміщень при загальній освітленості по ДБН 3.2.5-28-2006 при використанні люмінесцентних ламп $E_н$= 400 люкс.

Значення фактичного освітлення $E_ф$ може розрахуватися за допомогою методу коефіцієнта використання світлового потоку по формулі:

$$E_ф = \frac{F_л \cdot \eta_B \cdot N \cdot n}{S \cdot K_З \cdot Z},$$

де: $F_л$ – світловий потік лампи, лм (для люмінесцентних ламп потужністю 40 Вт – $F_л$ =3120 лм);

η_B – коефіцієнт використання світлового потоку (η_B=0,4..0,6=0,5);

N=6 – кількість світильників, 95вт;

n=3 – кількість ламп у світильнику, 95вт;

S=45 м^2 – площа приміщення, м^2;

$K_З$ – коефіцієнт запасу, $K_З$=1,5..2;$K_З = 1{,}5$

Z=1,1 – коефіцієнт нерівномірності освітлення.

$$E_ф = \frac{3120 \cdot 0{,}5 \cdot 6 \cdot 3}{45 \cdot 1{,}5 \cdot 1{,}1} = 378(\text{люкс})$$

Допустиме відхилення фактичного освітлення від нормативного є +20%..-10%.

Схема розміщення світильників у приміщенні наведено на рис. 5.2.2.1.

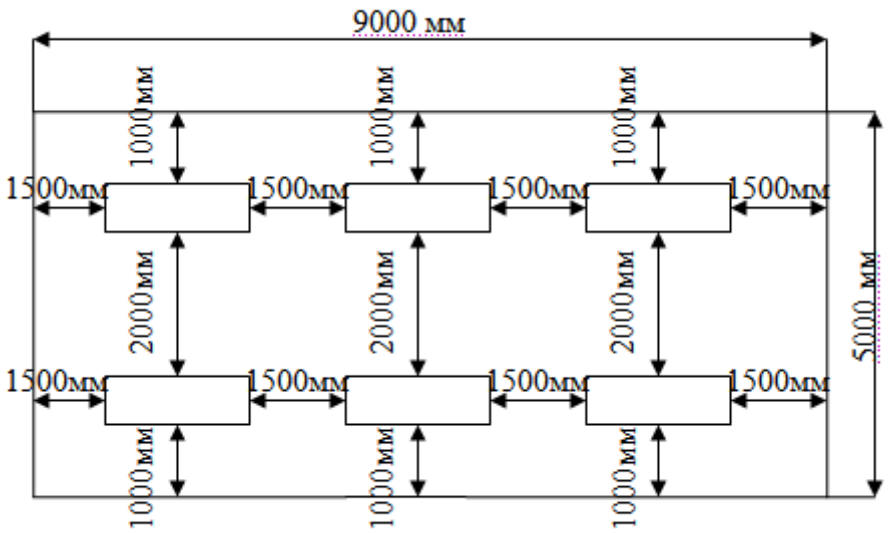


Рис. 5.2.2.1. Схема розміщення світильників

5.2.5 Електробезпека

Згідно ГОСТ 12.2.007.0-75 за ступенемзахисту людини від ураження електричним струмом ВДТ ПЕОМ та периферійні пристрої відносяться до I та II класів електробезпеки.

Приміщення відноситься до сухих, так як відносна вологість у приміщеннях не перевищує 60%.

За ступенем небезпеки ураження електричним струмом дане приміщення відноситься до приміщень без підвищеної небезпеки, тому що в ньому відсутні умови, які створюють підвищену або особливу небезпеку: вологість, струмопровідний пил, середовище, струмопровідна підлога, підвищена температура.

ЕОМ, периферійні пристрої, інше устаткування (чайник, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту відповідають класу зони за ПВЕ, мають апаратуру захисту від струму короткого замикання та інших аварійних режимів.

В приміщенні знаходиться більше 5 персональних ЕОМ, тому для прокладення захищеного проводу використано гнучкі металеві рукава та на помітному, доступному місці встановлено аварійний резервний вимикач. Таким

способом реалізована електромережа штепсельних розеток для живлення персональних ЕОМ, периферійних пристроїв, електрочайника тощо. Самі розетки монтуються на негорючих пластинах з урахуванням вимог ПВЕ та Правил пожежної безпеки в Україні та виконані за магістральною схемою.

Заземлення відповідає вимогам ДНАОП 0.00-1.21-98 “Правила безпечної експлуатації електроустановок споживачів”.

Для протирання підлоги застосовують рідини, пара яких не утворює вибухопожежонебезпечних сумішей з повітрям та не викликає корозії контактів електричних з`єднань.

Основними заходами захисту від ураження електричним струмом для даного приміщення є забезпечення недоступності струмоведучих частин, що знаходяться під напругою, від випадкового дотику, організація безпечної експлуатації електронагрівальних приладів а також занулення устаткування з використанням автоматів струмового захисту. При наявності занулення замикання фази на корпус перетворюється в одноразове коротке замикання струму, від я кого спрацьовує пристрій максимального струмового захисту і вимикає пошкоджений електроприлад. Так як в приміщеннях зберігаються горючі матеріали, то електрична проводка в приміщенні прихована під шаром штукатурки 2,5 см.

Розрахунок електромережі на вимикаючу здатність включає знаходження величини струму КЗ і розрахунок номінального струму спрацювання пристрою максимального струмового захисту.

Вихідні дані для розрахунку:

- $U_ф$=220 В – фазна напруга;
- кабель трьохжильний 3х1,5 мм^2 плюс 1х0,85 мм^2, матеріал – алюміній(ρ=0,031 Ом·мм^2/м);
- L=150 м – відстань від трансформатора до споживача;

Розрахуємо активний опір фазного та нульового проводів:

$$r_H = r_ф = \frac{\rho \cdot L}{S_ф} = \frac{0{,}031 \cdot 150}{1{,}5} = 3{,}1 \text{ (Ом)}$$

Струм однофазного КЗ знаходиться по формулі:

$$\mathrm{I}_{\text{КЗ}} = \frac{U_{\text{ф}}}{r_{\text{ф}} + r_{\text{Н}} + r_{u}} = \frac{220}{3{,}1 + 3{,}1 + 0{,}3} = 33{,}9\ (\text{А}),$$

де $r_{u} = 0{,}3$ Ом – розрахований опір трансформатора потужністю 250 Вт.

Номінальний струм спрацювання автомату струмового захисту розраховується за формулою (К – необхідна кратність струму КЗ до струму спрацювання автомату струмового захисту):

$$\mathrm{I}_{\text{ном}} = \frac{\mathrm{I}_{\text{КЗ}}}{\text{К}} = \frac{33{,}9}{1{,}4} = 24{,}2\ (\text{А})$$

З розрахунків видно, що при однофазному КЗ номінальний струм спрацювання автомату захисту повинен бути рівним 25 А.

При однофазному КЗ нульовий провід і з'єднаний з ним корпус електроустаткування за час спрацювання максимального струмового захисту знаходяться під напругою ($\mathrm{U}_{\text{пр}}$) відносно землі:

$$U_{\text{дот}} = \mathrm{I}_{\text{КЗ}} \cdot r_{\text{Н}} = 33{,}9 \cdot 3{,}1 = 105{,}1\ (\text{В})$$

Розрахована напруга $U_{\text{дот}} < U_{\text{дот.доп.}}$ у відповідності з ГОСТ 12.1.038-88 при t< 0,2 с ($U_{\text{дот.доп.}} = 250\text{В}$).

Державний контроль за станом пожежної безпеки ведуть інспектора Держпожнагляду.

5.2.6. Вимоги з ергономіки до робочих місць користувачів ВДТ ПЕОМ

Робоче місце – це частина простору, в якому інженер здійснює трудову діяльність, і проводить велику частину робочого часу. При правильній організації робочого місця продуктивність праці програміста зростає з 8 до 20 відсотків. Розглянемо питання та заходи з ергономіки.

Згідно ГОСТ 12.2.032-78 конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам.

Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови, крім попередньо згаданих:

- оптимальне розміщення устаткування, що входить до складу робочого місця;

- достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення;

Головними елементами робочого місця програміста є письмовий стіл і крісло. Основним робочим положенням є положення сидячи. Робоче місце для виконання робіт у положенні сидячи організується відповідно до ГОСТ 12.2.032-78 (Рис. 5.2.6.1).

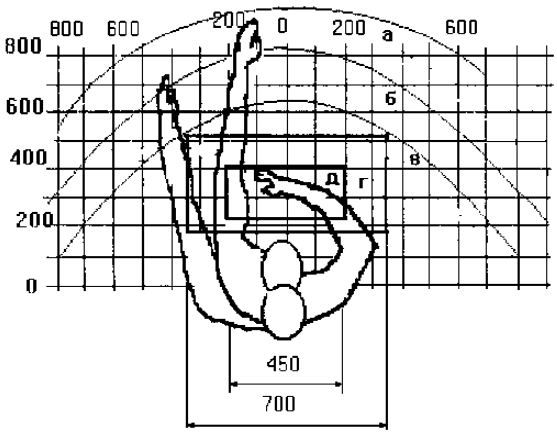


Рис. 5.2.6.1. Зони досяжності рук в горизонтальній площині

а – зона максимальної досяжності;

б – зона досяжності пальців при витягнутій руці;

в – зона легкої досяжності долоні;

г – оптимальний простір для грубої ручної роботи;

д – оптимальний простір для тонкої ручної роботи.

Розглянемо оптимальне розміщення предметів праці і документації в зонах досяжності рук:

- ДИСПЛЕЙ розміщається в зоні а (у центрі);

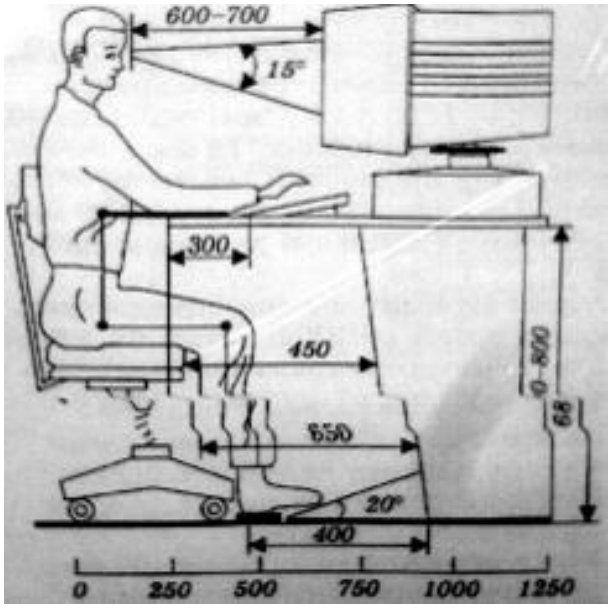


Рис. 5.2.6.2. Оптимальні метричні параметри робочого місця (мм)

- КЛАВІАТУРА – у зоні г / д;
- СИСТЕМНИЙ БЛОК розміщається в зоні б (зліва);
- ПРИНТЕР знаходиться в зоні а (праворуч);
- ДОКУМЕНТАЦІЯ:

○ в зоні легкої досяжності долоні – в (ліворуч) – література і документація, необхідна при роботі;

○ у висувних ящиках столу – література, якою не постійно.

Параметри робочого місця вибираються відповідно до антропометричними характеристиками. При використанні цих даних у розрахунках варто виходити з максимальних антропометричних характеристик.

5.3 Безпека в надзвичайних ситуаціях

Безпека в надзвичайних ситуаціях регламентується ПЛАС (ДНАОП 0.00-4.33-99). Одними з основних складових ПЛАС є розробка технічних рішень та організаційних заходів щодо оповіщення, евакуації та дій виробничого персоналу у разі виникнення надзвичайної ситуації (НС), а також визначення основних заходів з пожежної безпеки (ПБ).

5.3.1. Обов’язки та дії персоналу у разі виникнення НС

У разі виявлення НС працівник повинен:

– негайно повідомити про це Державну пожежну охорону засобами зв’язку органам МНС та вказати при цьому адресу, кількість поверхів, місце виникнення НС, наявність людей, а також своє прізвище;

– повідомити про НС керівника, адміністрацію, пожежну охорону підприємства;

– організувати оповіщення людей про НС;

– вжити заходів щодо евакуації людей та матеріальних цінностей;

– вжити заходів щодо ліквідації наслідків НС з використанням наявних засобів.

Керівник та пожежна охорона установи, яким повідомлено про виникнення НС, повинні :

– перевірити, чи викликано підрозділ МНС та Державної пожежної охорони;

– вимкнути у разі необхідності струмоприймачі та вентиляцію;

– у разі загрози життю людей негайно організувати їх евакуацію та їх рятування, вивести за межі небезпечної зони всіх працівників, які не беруть участь у ліквідації НС;

– перевірити здійснення оповіщення людей про НС;

– забезпечити дотримання техніки безпеки працівниками, які беруть участь у ліквідації наслідків НС;

– організувати зустріч підрозділів МНС та Державної пожежної охорони, надати їм допомогу у локалізації та ліквідації наслідків НС.

Після прибуття підрозділів МНС та Державної пожежної охорони повинен бути забезпечений безперешкодний доступ їх до місця, де виникла НС.

5.3.2 Вимоги до організації ефективної роботи системи оповіщення виробничого персоналу у разі виникнення надзвичайної ситуації

Для підвищення безпеки в надзвичайних ситуаціях (НС) пропонується встановлення системи оповіщення (СО) виробничого персоналу.

Оповіщення виробничого персоналу у разі виникнення НС, здійснюється відповідно до вимог НАПБ А.01.003-2009.

Необхідність обладнання виробничих приміщень певним типом СО визначається згідно з додатком Е до ДБН В.1.1-7-2002 «Захист від пожежі. Пожежна безпека об’єктів будівництва».

При обладнані виробничих будівель системою оповіщення, їх необхідно поділяти на зони оповіщування з урахуванням об’ємно-планувальних рішень будинків, шляхів евакуації, поділення на протипожежні відсіки тощо, а також з урахуванням вимог, що наведені в примітці 1 таблиці Е.1 додатка Е до ДБН В.1.1-7-2002.

Розміри зон оповіщування, черговість оповіщування та час початку оповіщування людей в окремих зонах визначаються, виходячи з умов забезпечення безпечної та своєчасної евакуації людей у разі виникнення НС.

Оповіщування про НС та управління евакуацією людей здійснюється одним з наступних способів або їх комбінацією:

- поданням звукових і (або) світлових сигналів в усі виробничі приміщення будівлі з постійним або тимчасовим перебуванням людей;

- трансляцією текстів про необхідність евакуації, шляхи евакуації, напрямок руху й інші дії, спрямовані на забезпечення безпеки людей;

- трансляцією спеціально розроблених текстів, спрямованих на запобігання паніці й іншим явищам, що ускладнюють евакуацію;

- розміщенням знаків безпеки на шляхах евакуації згідно з ДСТУ ISO 6309;

- ввімкненням евакуаційних знаків «Вихід»;

- ввімкненням евакуаційного освітлення та світлових покажчиків напрямку евакуації;

- дистанційним відкриванням дверей евакуаційних виходів;

- зв’язком оперативного (чергового) персоналу СО (диспетчера пожежного поста) із зонами оповіщування.

Як правило, СО вмикається автоматично від сигналу про НС, який формується системою пожежної сигналізації або системою пожежогасіння. Також з приміщення оперативного (чергового) персоналу СО (диспетчера пожежного поста) слід передбачати можливість запуску СО вручну, що забезпечує надійну роботу СО у разі виникнення будь-якої НС.

Повинен бути забезпечений розподіл пріоритетів щодо повідомлень для виробничого персоналу у такій послідовності:

I (найвищий) – повідомлення оперативного (чергового) персоналу СО (диспетчера пожежного поста) у разі виникнення будь-якої НС;

II – повідомлення, які записані на будь-якому носії та вмикаються автоматично від спрацювання систем пожежної автоматики, або за сигналом оперативного (чергового) персоналу СО (диспетчера пожежного поста);

III – службові повідомлення, що не стосуються організації та управління евакуацією людей.

У разі одночасного транслювання декількох повідомлень, що мають різні пріоритети, повідомлення, які мають нижчий пріоритет, повинні автоматично блокуватись.

СО повинна мати можливість одночасно передавати різні мовленнєві повідомлення в різні зони оповіщування.

Згідно з вимогами ДБН В.1.1-7-2002 необхідно можливість прямої трансляції мовленнєвого оповіщування та керівних команд через мікрофон для оперативного реагування в разі зміни обставин або порушення нормальних умов евакуації виробничого персоналу.

В разі виникнення НС у багатоповерхових виробничих будівлях, СО повинна діяти у такій послідовності:

- в першу чергу, здійснюється оповіщування людей про НС на поверсі, де виникла НС;

- потім оповіщування людей про НС на поверхах, що розташовані вище поверху, де виникла НС;

- в останню чергу, оповіщування людей про НС на поверхах, що розташовані нижче поверху, де виникла НС.

Затримку часу оповіщування про НС для різних поверхів будинку необхідно передбачати з урахуванням злиття потоків людей на шляхах евакуації відповідно до розрахунків по ГОСТ 12.1.004 «ССБТ. Пожарная безопасность. Общие требования».

У багатоповерхових виробничих будівлях, які поділені на протипожежні відсіки по вертикалі, СО повинна вмикатися одразу для всього протипожежного відсіку, де виникла НС. Затримку часу оповіщування про НС для інших вертикальних протипожежних відсіків будинку слід передбачати з урахуванням злиття потоків людей на шляхах евакуації відповідно до вимог додатка 2 згідно ГОСТ 12.1.004.

Оповіщення виробничого персоналу про НС здійснюється за допомогою світлових та/або звукових оповіщувачів – обладнуються всі виробничі приміщення.

СО повинна розпочати трансляцію сигналу оповіщування про НС, не пізніше трьох секунд з моменту отримання сигналу про НС.

Сигнал привертання уваги та сигнал оповіщування про НС, повинен відрізнятися від сигналів іншого призначення.

5.3.1 Пожежна безпека

Приміщення, в якому розроблялися алгоритми та структура ПЗ, за небезпекою виникнення пожежі відповідно до ОНТП 24-86 належить до категорії В (пожежонебезпечні – в ньому наявні легкозаймисті речовини, зокрема папір, дерево). За ступенем вогнестійкості дане приміщення можна віднести до важкого, що можуть продовжувати горіти тільки при постійному впливі джерела запалення.

Основними причинами пожежі в приміщенні можуть бути недотримання правил пожежної безпеки, їх порушення, неправильне використання електроприладів, несправність електричної проводки або електроприладів, а також куріння в не відведених місцях.

Протипожежний режим на об’єкті дотримується. Щороку всі працівники проходять протипожежний інструктаж. Кожен працівник ознайомлюється з правилами техніки безпеки при виникненні пожежі, з правилами евакуації, з використанням вогнегасників та інших засобів вогнегасіння. Усе це фіксується в журналі інструктажу підписами службовців.

Наявні шляхи евакуації забезпечують безпечну евакуацію людей і відповідає вимогам розділу „Евакуація людей з приміщень і будівель”. Для евакуації людей з будівлі є „План евакуації людей та матеріальних цінностей при виникненні пожежі”. Плани евакуації вивішені на кожному поверсі (рис. 5.3.1). Мінімальний

час евакуації з приміщення не перевищує допустиме значення (1,25 хв.) згідно СниП 2.01.02-85, СниП 2.09.02-85.

На поверсі, де розташоване виробниче приміщення, біля обох сходів є по одному вуглекислотному вогнегаснику (тип вогнегасників обрано згідно з ДСТУ 3675-98 та ISO 3941-77) та сповіщувачу пожежному ручному. Всі електронагрівальні прилади експлуатуються на підставі письмового дозволу на їх застосування.

Приміщення обладнано автоматичною системою пожежної сигналізації, яка відповідає вимогам ДНБВ 2-5-13-98

Для куріння на кожному поверсі обладнані спеціальні місця, розташовані в туалеті.

Відділ телефонізовано, в кожному кабінеті є телефон. Засоби пожежогасіння і зв'язку перебувають у справному стані.

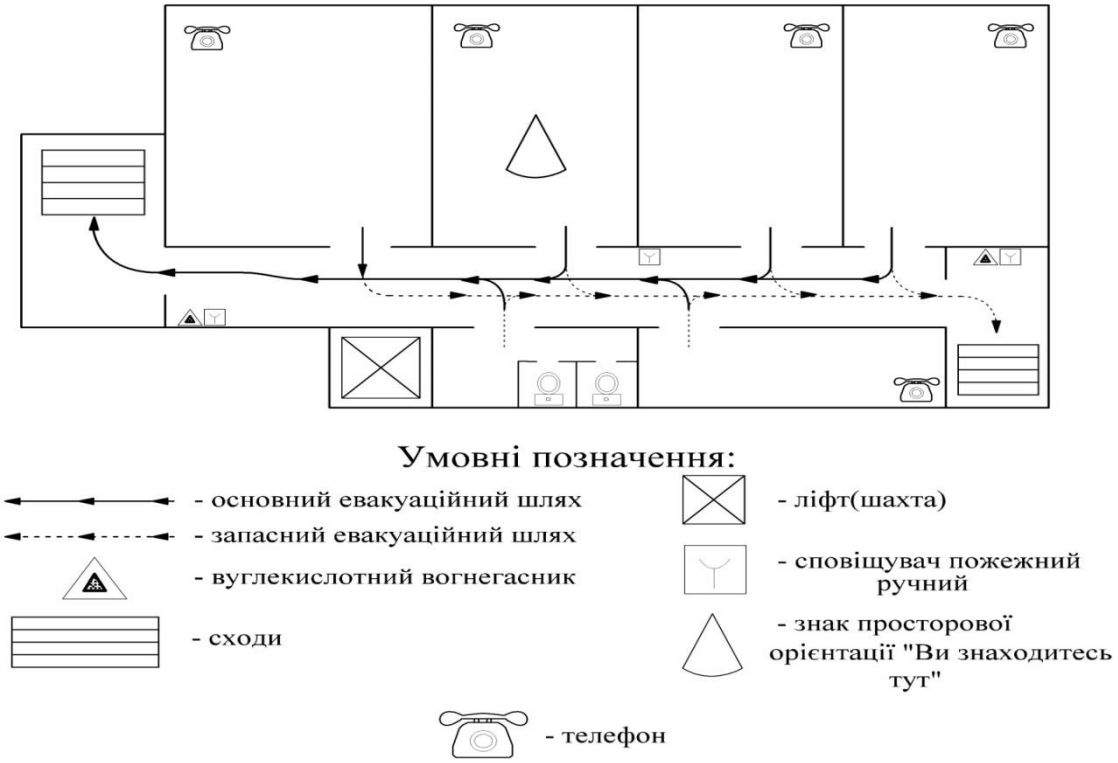


Рис. 5.3.1. План евакуації з приміщення у разі виникнення НС (пожежі).

ЗАГАЛЬНІ ВИСНОВКИ

В даній магістерській дисертації була досліджена та розроблена система керування відео потоком з використанням мікрокомп’ютерів.

Був проведений аналіз аналогів та прийнято рішення розробляли систему самостійно.

В результаті проведення порівняння мов програмування додаток під мобільний пристрій був реалізований на Java. Додаток включає у себе можливість додавання мікрокомп’ютера в історію пристроя, щоб надалі використовувати швидкий пошук доступник комп’ютерів, підключення одночасно до декількох каналів транслювання відеопотока і відображення одночасно на дисплеї мобільного пристрою, аналізувати швидкість приймання відеопотоку.

Досліджено втрати при передачі відеопотоку у мережі Wi-Fi.

Досліджено зовнішні фактори, які впливають на якість відеопотоку.

Була проведена оцінка охорони праці для безпечного дослідження та розробки даної системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Блинов И.Н. Java. Промышленное программирование : практ. пособие / В.С. Романчик. – Минск : УниверсалПресс, 2007. – 704 с.

2. Бернард В.Х. JDBC. Java и базы данных : Лори, 2000. – 324с.

3. Хассан Г. UML-проектирование систем реального времени параллельных и распределенных приложений : ДМК Пресс, 2011. – 711 с.

4. Jay K. Using SQLITE : O'Reilly Media, 2010. – 528 с.

5. M2M Summit 2013 : Raspberry Pi and Java — Love At First Sight [Електронний ресурс] / M2M Summit 2013 — Режим доступу : http://m2m-summit.com/files/m2m_oracle_java___raspberry_pi_v2.pdf. — Назва з екрану.

6. MikroTik : З початку продажів по світу розійшлося вже 5 млн мікрокомп'ютерів Raspberry Pi [Електронний ресурс] / MikroTik — Режим доступу: http://mikrotik.kpi.ua/index.php/component/k2/item/24-5-billion-raspberry-pi. — Назва з екрану.

7. Могильний С. Б. Мікрокомп'ютер Raspberry Pi — інструмент дослідника : посібник. - К. : «Талком», 2014. - 340 с. ISBN 978-617-7133-48-2

8. Geektimes : Потоковое видео в Android — Режим доступу : http://geektimes.ru/post/148754/ . — Назва з екрану.

9. Хабрахабр : Raspberry Pi. Передача видео для дистанционного управления — Режим доступу : http://habrahabr.ru/post/1536111. — Назва з екрану.

10. Хабрахабр : Безпроводная передача видео и звука по Wi-Fi — Режим доступу : http://habrahabr.ru/post/138670. — Назва з екрану.

11. ГОСТ 12.1.004-74 «Пожарная безопасность. Общие требования».

12. НАПБ Б.03.002-2007 "Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою".

13. Смирнов Г.В., Кодолова Л.И. Безопасность жизнедеятельности. – Томск: ТУСУР. 2003.–79с.

Додаток 1. Код классу ініціалізації бази даних

```
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DBManager {


    String DBName = "base.db";
    Connection conn;
    Statement st;
    private static DBManager singleInstance = new DBManager();

    private DBManager() {

    }

    public static DBManager getInstance(){
         return singleInstance;
    }

    synchronized public void connectToDB() throws Exception{

         Class.forName("org.sqlite.JDBC");
        String dbURL = "jdbc:sqlite:"+DBName;
        conn = DriverManager.getConnection(dbURL);
        st = conn.createStatement();
        DatabaseMetaData md = conn.getMetaData();

        if (!md.getTables(null, null, "%", null).next()){

            String tableKeylog = "CREATE TABLE keylog (" +
                   "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                   "user TEXT," +
                   "text TEXT," +
                   "timestamp TIMESTAMP);";

            st.execute(tableKeylog);

            String tableUserActivity = "CREATE TABLE user_activity
(" +
                     "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                     "user TEXT," +
                     "begin_time TIMESTAMP," +
                     "end_time TIMESTAMP);";
```

```
        st.execute(tableUserActivity);

        String tableDeviceInfo = "CREATE TABLE device_info (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                "os_version TEXT," +
                "os_arch TEXT," +
                "computer_name TEXT);";

        st.execute(tableDeviceInfo);

        String tableScreenshot = "CREATE TABLE screenshot (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                "user TEXT," +
                "image BLOB," +
                "timestamp TIMESTAMP);";

      st.execute(tableScreenshot);

    }
}



public void close() throws SQLException{
     st.close();
     conn.close();
}
```

Додаток 2. Код класу Activity, який викону відображення відеопотоку

```
import java.util.Timer;
import java.util.TimerTask;
import android.os.Bundle;
import android.os.StrictMode;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.view.Menu;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.ImageView;

public class MainClientView extends Activity {
            StrictMode.ThreadPolicy        policy        =        new
StrictMode.ThreadPolicy.Builder().permitAll().build();
         public Rtsp RTSP;
         public String state;  //START->READY->PLAY/PAUSE/TEAR
         public Timer timer;
         public Rtp RTP;

         @Override
         protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main_client_view);
            StrictMode.setThreadPolicy(policy);
            state = "START";
            timer = new Timer();
         }

         @Override
```

```
        public boolean onCreateOptionsMenu(Menu menu) {
           // Inflate the menu; this adds items to the action bar if
it is present.

        getMenuInflater().inflate(R.menu.activity_main_client_view,
menu);
           return true;
        }

        public void setup(View view){

           //Send Setup Message
           try{
             if(state.equals("START")){
                  System.out.println("Setup");

                  //Get the variables
                  String                movie_file                =
((EditText)findViewById(R.id.editMjpeg)).getText().toString();
                  String                  ip_addr                 =
((EditText)findViewById(R.id.editIP)).getText().toString();
                  String                string_port               =
((EditText)findViewById(R.id.editPort)).getText().toString();
                  int port = Integer.parseInt(string_port);

                  //Create RTSP and call setup
                  RTSP = new Rtsp(movie_file,ip_addr,port);
                  if(RTSP.setup()){
                       state = "READY";
                       RTP = new Rtp(port);
                       System.out.println("State is now ready");
                  }
             }
           }catch(Exception e){
             System.out.println(e);
```

```
    }

}

public void play(View view){
    if(state.equals("READY")){
      try{
           //Call Play
           if(RTSP.play()){
                state = "PLAY";
                System.out.println("Now Playing");

                //Init Timer
                timer.schedule(new TimerTask(){
                     @Override
                     public void run(){
                          TimerTick();
                     }
                }, 0,60);
           }
      }catch(Exception e){
           System.out.println(e);
      }
    }

    //If paused, notify the timer to run
    if(state.equals("PAUSE")){
      try{
           if(RTSP.play()){
                state = "PLAY";
                System.out.println("Video Unpaused");

                //Notify timer
                timer.notify();
           }
```

```
            }catch(Exception e){
                System.out.println(e);
            }
        }

    }
        public void pause(View view){
        if(state.equals("PLAY")){
            try{
                //Call Pause
                if(RTSP.pause()){
                    state = "PAUSE";
                    System.out.println("Video Paused");

                    //Stop Timer
                    timer.wait();
                }
            }catch(Exception e){
                System.out.println(e);
            }
        }
    }

    public void teardown(View view){
        if(state.equals("PLAY")   ||   state.equals("PAUSE")   ||
state.equals("READY")){
            try{
                //Call teardown
                if(RTSP.teardown()){
                    state = "START";
                    RTP.destroySocket();
                    //Destroy timer
                    timer.wait();
                }
            }catch(Exception e){
```

```
                System.out.println(e);
            }
        }
            }

        //Hides the keyboard
        public void hideKeyboard(View view){
            InputMethodManager          inputMethodManager          =
(InputMethodManager)
this.getSystemService(Activity.INPUT_METHOD_SERVICE);

inputMethodManager.hideSoftInputFromWindow(this.getCurrentFocus().ge
tWindowToken(), 0);
        }

        //Called when the timer goes off
        private void TimerTick(){

            //RTP should handle this stuff
            runOnUiThread(new Runnable(){
              public void run(){
                  Bitmap bmp = RTP.timerEvent();
                  if(bmp != null){
                      try{
                      ImageView               image               =
(ImageView)findViewById(R.id.imageView);
                      image.setImageBitmap(bmp);
                      }catch(Exception e){
                          System.out.println(e);
                      }
                  }
              }
            });
        }
}
```

Додаток 3. Код класу, який викону підключення до мікрокомп'ютера та приймання відеопотоку

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

import android.graphics.Bitmap;
import android.os.StrictMode;

public class Rtp {
        StrictMode.ThreadPolicy        policy        =        new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        //Each Rtp instance should have a packet
        //Previous packets will be overwritten by the current packet
        RtpPacket rtp_packet;
        private byte[] buf;
        private DatagramSocket socket;
        private int port;
        private DatagramPacket receive_packet;
        private int plength;
        private int largestFrame;
        private boolean finished;

        //Constructor
        public Rtp(int port){
           buf = new byte[15000];
           this.port = port;
           largestFrame = 0;

           try{
             socket = new DatagramSocket(25000);
             socket.setSoTimeout(10);
           }catch(Exception e){
             System.out.println("Error Creating Socket: " + e);
             finished = false;
           }
```

```
        }

        //Get the next video frame
        public Bitmap timerEvent(){
           finished = false;

           //Create packet
           receive_packet = new DatagramPacket(buf, buf.length);

           try{
             //Receive packet
             socket.receive(receive_packet);
             rtp_packet  =  new  RtpPacket(receive_packet.getData(),
receive_packet.getLength());
             Bitmap bmp = rtp_packet.getBmp();

             //Commented out now that we're getting BMP in packet
             //Get payload
             /*plength = rtp_packet.getLength();
             if(plength>largestFrame){
                  largestFrame = plength;
                  System.out.println("The  Largest  Frame  is:  "  +
largestFrame);
             }
             byte[] payload = new byte[plength];
             for(int i = 0; i < plength; i++){
                  payload[i] = rtp_packet.getPayload()[i];
             }*/

             System.out.println("Got Payload!!!!!");
             return bmp;
           }
           //Sometimes  the  socket  won't  be  made  correctly  if  the
user puts in an incorrect IP
           catch(java.lang.NullPointerException en){
```

```
            try{
                socket = new DatagramSocket(25000);
                socket.setSoTimeout(10);
            }catch(Exception e){
                System.out.println("Error     Creating     Socket
Again!");
            }
            return null;
          }
          catch(java.net.BindException be){
            System.out.println("UH oh, bind exception");
            return null;
          }
          catch(java.net.SocketTimeoutException se){
            System.out.println("Reached end of video, teardown");
            finished = true;
            return null;
          }
          catch(Exception e){
            System.out.println("Error with UDP, Packet Dropped");
            System.out.println(e);
            return null;
          }
        }
        public int getPlength(){
          return plength;
        }
        public void destroySocket(){
          socket.close();
        }
        public boolean getFinished(){
          return finished;
        }

}
```

Додаток 4. Клас обгортка rtp пакету

```
import java.util.Date;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;

public class RtpPacket {
   private Bitmap bmp;

   private byte[] header;
   private byte[] payload;

   //Constructor
   public RtpPacket(){
   }

   //Special Constructor for received packets
   public RtpPacket(byte[] packet, int size){
                  bmp  =  BitmapFactory.decodeByteArray(packet,
12, packet.length - 12);
   }

   public void initPacket(String server_message){
   }
   public byte[] getPayload(){
      return payload;
   }
   public int getLength(){
      return payload.length;
   }
   public Bitmap getBmp(){
      return bmp;
   }

}
```

Додаток 5. Клас контейнер зберігання інформації поточних підключень

```
import java.net.*;
import java.util.StringTokenizer;
import java.io.*;
import android.os.StrictMode;

public class Rtsp {
        private String movie_file;
        private String ip_addr;
        private int port;
        private int c_seq;
        private int session;
        private int id; //The RTSP ID given by the server
        private BufferedReader RTSPBufferedReader;
        private BufferedWriter RTSPBufferedWriter;
        private InetAddress addr;

        private Socket rtsp_socket;

        //Constructor
        public Rtsp(String movie_file, String ip_addr, int port)
throws IOException{
           this.movie_file = movie_file;
           this.ip_addr = ip_addr;
           this.port = port;
           this.c_seq = 1;
           this.id = 0;
           //Session given from setup
           this.session = 0;
           this.addr = InetAddress.getByName(ip_addr);

           //Create RTSP Socket
           this.rtsp_socket = new Socket(ip_addr,port);
           //Create RTSP Writers/Readers
```

```
            RTSPBufferedReader      =      new      BufferedReader(new
InputStreamReader(rtsp_socket.getInputStream()) );
            RTSPBufferedWriter      =      new      BufferedWriter(new
OutputStreamWriter(rtsp_socket.getOutputStream()) );
         }

         //SETUP
         public boolean setup(){

            //Create RTSP Message
            String rtsp_setup = "SETUP rtsp://" + ip_addr + ":" +
port + "/" + movie_file + " RTSP/1.0" + "\nCSeq: " + c_seq
+"\n"+"Transport: RTP/UDP; client_port= " + 25000;
            c_seq++;

            //Create & Send Packet
            try{
              RTSPBufferedWriter.write(rtsp_setup);
              RTSPBufferedWriter.flush();
            }catch(Exception e){
              return false;
            }

            //Receive response
            if(getRTSPResponse() == 200)
              return true;
            else
              return false;
         }
         public boolean play(){
            //Create RTSP Message
            String rtsp_play = "PLAY rtsp://" + ip_addr + ":" + port
+ "/" + movie_file + " RTSP/1.0" + "\nCSeq: " + c_seq
+"\n"+"Session: " + id;
            c_seq++;
```

```
            //Create & Send Packet
            try{
              RTSPBufferedWriter.write(rtsp_play);
              RTSPBufferedWriter.flush();
            }catch(Exception e){
              return false;
            }
            //Receive Response
            if(getRTSPResponse()==200)
              return true;
            else
              return false;
          }
          public boolean pause(){
            //Create RTSP Message
            String rtsp_pause = "PAUSE rtsp://" + ip_addr + ":" +
port + "/" + movie_file + " RTSP/1.0" + "\nCSeq: " + c_seq
+"\n"+"Session: " + id;
            c_seq++;

            //Create and send packet
            try{
              RTSPBufferedWriter.write(rtsp_pause);
              RTSPBufferedWriter.flush();
            }catch(Exception e){
              return false;
            }

            //Receive Response
            if(getRTSPResponse() == 200)
              return true;
            else
              return false;
          }
```

```
        public boolean teardown(){
           //Create RTSP Message
           String rtsp_teardown = "TEARDOWN rtsp://" + ip_addr + ":"
+ port + "/" + movie_file + " RTSP/1.0" + "\nCSeq: " + c_seq
+"\n"+"Session: " + id;
           c_seq++;

           //Create and send packet
           try{
             RTSPBufferedWriter.write(rtsp_teardown);
             RTSPBufferedWriter.flush();
           }catch(Exception e){
             return false;
           }

           //Receive Response
           if(getRTSPResponse() == 200)
             return true;
           else
             return false;
        }
        //Gets the return code
        private int getRTSPResponse(){
           int reply = 0;
           try{
             //Get Response
             String status = RTSPBufferedReader.readLine();
             StringTokenizer tokens = new StringTokenizer(status);
             tokens.nextToken();
             reply = Integer.parseInt(tokens.nextToken());
           }catch(Exception e){
             repy = -1;
           }
           //Parse the response to get ID
```

```
        try{
          String seq = RTSPBufferedReader.readLine();
          String session = RTSPBufferedReader.readLine();
          StringTokenizer tokens = new StringTokenizer(session);
          tokens.nextToken();
          id = Integer.parseInt(tokens.nextToken());
        }catch(Exception e){
          reply = -1;
        }
        return reply;
      }
      //Getters and Setters
      public String getMovie_file(){
        return movie_file;
      }
      public void setMovie_file(String movie_file){
        this.movie_file = movie_file;
      }
      public String getIp_addr(){
        return ip_addr;
      }
      public void setIp_addr(String ip_addr){
        this.ip_addr = ip_addr;
      }
      public int getPort(){
        return port;
      }
      public void setPort(int port){
        this.port = port;
      }

    }
```

Додаток 5. XML файл головного екрану

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainClientView" >

    <Button
        android:id="@+id/buttonTeardown"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/buttonPlay"
        android:text="Teardown"
        android:onClick="teardown"/>

    <Button
        android:id="@+id/buttonSetup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/buttonPause"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="Setup"
        android:onClick="setup" />

    <Button
        android:id="@+id/buttonPause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
```

```
    android:layout_toLeftOf="@+id/buttonPlay"
    android:text="Pause"
    android:onClick="pause" />

<Button
    android:id="@+id/buttonPlay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/buttonSetup"
    android:layout_centerHorizontal="true"
    android:text="Play"
    android:onClick="play" />
<EditText
    android:id="@+id/editIP"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/buttonSetup"
    android:layout_alignParentLeft="true"
    android:ems="10"
    android:text="10.0.2.2"
    tools:ignore="TextFields" >
    <requestFous />
</EditText>
<EditText
    android:id="@+id/editPort"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/buttonSetup"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/editIP"
    android:ems="10"
    android:inputType="number"
    android:text="3000" />

</RelativeLayout>
```