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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ланецький В.С.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *125 "*Кібербезпека*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ланецькому Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки IP-телефонії спеціального призначення*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Ланецький В.С.
(прізвище та ініціали)

АНОТАЦІЯ

Ланецький В.С. Програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки IP-телефонії спеціального призначення.

Метою розробки є програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення.

Результат роботи – програмна реалізація системи кібербезпеки IP-телефонії спеціального призначення.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: кібербезпека, IP-телефонія

ABSTRACT

Lanetskyi V.S. Software of the special-purpose IR-telephony cyber security system. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the special purpose IR-telephony cyber security system.

The purpose of the development is the software of the cyber security system of special purpose IR-telephony.

The result of the work is the software implementation of the special-purpose IR-telephony cyber security system.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: cybersecurity, IP telephony

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АТМУ – автоматизована телефона мережа установи
БД – база даних
КМЗ – корпоративна мережа зв’язку
ЛОМ – локальна обчислювальна мережа
ОЗП – оперативно-запам’ятовувальний пристрій
ПАТМ – пристрій автоматизованої телефонної мережі
ПЗ – програмне забезпечення
ПП – програмний продукт
СПД – система передачі даних
СУБД – система управління БД
ТфОП – телефонний оператор
AVVID – архітектурна модель фирмы Cisco Systems
CNG – Comfort Noise Generator. Генератор комфортного шуму
DSP – Digital Signal Processor. Процесор цифрової обробки сигналів
DTMF – Dual Tone Multi-Freqency. Багаточастотна система кодування цифр номера
GK – Gatekeeper. Гейткіпер . Виконує функції керування зоною мережі H.323
GW – Gateway. Шлюз. Апаратно-програмний комплекс, що забезпечує обмін даними між мережами різних типів
H.323 – Рекомендація ITU-T, що визначає системи мультимедійного зв'язку в мережах з пакетною комутацією
H.248 – Протокол керування транспортним шлюзом
IP – Internet Protocol. Протокол міжмережної взаємодії
LPC – Linear Prediction Coding. Кодування з лінійним пророкуванням
MCU – Multipoint Control Unit. Пристрій керування конференцією

MG – Media Gateway. Транспортний шлюз

MGCP – Media Gateway Control Protocol. Протокол керування шлюзами

MP – Multipoint processor. Процесор для обробки інформації користувачів при централізованих конференціях

OSI – Open System Interconnection. Взаємодія відкритих систем

PPP – Point-to-Point Protocol. Протокол двостороннього зв'язку

RADIUS – Remote Authentication Dial-In User Service. Протокол автентифікації й авторизації абонентів, а також обліку обсягу наданих їм послуг

RAS – Registration Admission and Status. Протокол взаємодії термінального встаткування з gatekeeper. Входить у сімейство протоколів H.323

RSVP – Resource Reservation Protocol. Протокол резервування ресурсів

RTCP – Real-time Transport Control Protocol. Протокол контролю транспортування інформації в реальному часі

SIP – Session Initiation Protocol. Протокол ініціювання сеансів зв'язку

TAPI – Telephony Applications Programming Interface. Інтерфейс для програмування телефонних додатків

TCP – Transmission Control Protocol. Протокол керування передачею (даних) Основний транспортний протокол у стеці протоколів TCP/IP.

TCP/IP – Transmission Control Protocol/Internet Protocol. Стек протоколів, що забезпечують організацію зв'язку між комп'ютерами в мережі Інтернет

UDP – User Datagram Protocol. Протокол передачі дейтаграмм користувача.

VoIP – Voice over Internet Protocol. Технологія, що дозволяє використовувати IP-мережу для передачі мовної інформації

ВСТУП

Актуальність теми. Державна система урядового зв’язку України (ДСУЗ) являє собою систему спеціального зв’язку, яка забезпечує передачу інформації, що містить державну таємницю, і функціонує в інтересах управління державою в мирний та воєнний час. На Державну службу спеціального зв'язку та захисту інформації України покладено завдання із забезпечення в установленому порядку урядовим зв'язком Президента України, Голови Верховної Ради України, Прем'єр-міністра України, інших посадових осіб органів державної влади, органів місцевого самоврядування, органів військового управління, керівників підприємств, установ і організацій у мирний час, в умовах надзвичайного та воєнного стану, а також у разі виникнення надзвичайної ситуації. На відміну від існуючих і створюваних в Україні спеціальних систем зв’язку ДСУЗ має тільки їй притаманну властивість: забезпечувати гарантоване засекречування інформації яка містить державну таємницю, що передається каналами та лініями зв’язку. Відповідно до свого призначення ДСУЗ забезпечує послугами зв’язку абонентів у стаціонарних умовах, рухомих об’єктах та непідготовлених з питань зв’язку районах. В основному ДСУЗ базується на стаціонарних об’єктах (станціях) урядового зв'язку, які забезпечують функціонування і взаємодію побудованих в усіх регіональних та крупних промислових центрах України мереж і комплексів урядового зв’язку. Забезпечення урядовим зв’язком у місцях, необладнаних стаціонарними засобами зв’язку, здійснюється рухомими вузлами урядового зв’язку.

Національна система конфіденційного зв'язку (НСКЗ) це сукупність спеціальних систем (мереж) зв'язку подвійного призначення, які за допомогою криптографічних та/або технічних засобів забезпечують обмін конфіденційною інформацією в інтересах органів державної влади та органів місцевого

самоврядування, створюють належні умови для їх взаємодії в мирний час та у разі введення надзвичайного і воєнного стану.

Абонентами цієї системи можуть бути як державні, так і комерційні структури, юридичні та фізичні особи.

Однією з особливостей даної системи є те, що вона створюється як система подвійного призначення. Цей принцип полягає у тому, що у мирний час мережі даної системи можуть використовуватися організаціями для передачі сучасними видами зв'язку конфіденційної інформації в інтересах як органів державної влади, так і інших юридичних осіб, у тому числі суб'єктів фінансово-економічної сфери. В особливий період та у разі виникнення надзвичайних ситуацій ресурс даної системи буде задіяний для передачі конфіденційної інформації в інтересах національної безпеки та оборони держави.

Створення та розвиток НСКЗ дозволяє вирішувати наступні стратегічні питання, а саме:

– забезпечити надійний захист конфіденційної інформації, що є власністю держави, відповідно до вимог законодавства;

– створити передумови інтеграції розподілених інформаційних ресурсів та інформаційно-аналітичних систем органів державної влади різного рівня державного управління, в яких циркулює конфіденційна інформація, що є власністю держави;

– забезпечити можливість інформаційної взаємодії між інформаційно-аналітичними системами органів державної влади різного рівня державного управління;

– забезпечити надійний обмін конфіденційною інформацією, що є власністю держави між абонентами НСКЗ. Одним з видів зв'язку є IP-телефонія.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення.

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

– Огляд існуючих систем IP-телефонії спеціального призначення.

– Дослідження системи кібербезпеки IP-телефонії спеціального призначення.

– Програмна реалізація системи кібербезпеки IP-телефонії спеціального призначення.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі IP-телефонії спеціального призначення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

IP-телефонія – це технологія, що здійснює злиття двох напрямків: телефонного й комп'ютерного. CTI (computer telephony integration) – це надання як термінал для спілкування з комп'ютером звичайного телефонного апарата. CTI дозволяє використовувати всі переваги комп'ютеризації (зроблені стандарти, гнучкість, сумісність, зручний і звичний інтерфейс і т.д.) для керування телефонними з'єднаннями.

Якщо телефон для підприємства – один з головних корпоративних інструментів, а його співробітникам доводиться багато спілкуватися із клієнтами, має сенс інтегрувати телефонну систему з мережею передачі даних. Шлях до цього відкриває один з напрямків CTI– LAN-телефонія, що дозволяє використовувати мережу передачі даних для передачі голосового трафіку, а також для керування вхідними й вихідними дзвінками й передачі інформації про дзвінки в комп'ютерні додатки.

У стандартних автоматизованих телефонних мережах (АТМ) для установ (private branch exchange, PBX) можливостей комп'ютерної телефонії, як правило, недостатньо. Системи PBX виконують функції по керуванню дзвінками, обліку й дотриманню пріоритетів, а системи LAN PBX поширюють ці функції на включені в локальну мережу ПК і дозволяють зв'язувати телефонні системи з комп'ютерними програмами. LAN PBX дешевше стандартних телефонних систем, і пропонують унікальні функції й гнучкість, створюючи основу для нових можливостей і додатків. Переміщення, додавання й зміна внутрішніх номерів в LAN PBX простіше й дешевше, ніж у традиційних PBX, до того ж у більшості випадків ними можуть управляти ті ж самі адміністратори локальної мережі.

IP-телефонія – це загальний термін, що позначає передачу голосу й факсу, а також пов'язані із цим сервіси, частково або повністю через пакетні мережі на основі протоколу IP (Internet Protocol – протокол міжмережної взаємодії).

Термін IP-телефонія еквівалентна терміну VoIP (Voice over IP). Internet-телефонія – більше вузьке поняття, коли в ролі транспортного середовища виступає мережа Internet.

У технології комп'ютерної телефонії існує кілька понять. Їх поєднує загальне визначення, як передача голосу в пакетному режимі, але, проте, їх прийнято розділяти на два напрямки:

– CTI (computer telephony integration) – технологія, у якій інтелектуальні комп'ютерні ресурси (апаратура й програмне забезпечення) застосовуються для здійснення вихідних і прийому вхідних дзвінків і для керування телефонним з'єднанням". У це визначення входять, як і безпосередньо програмно-апаратне забезпечення «чистої» комп'ютерної телефонії, так і IP-ATM таких відомих виробників, як 3COM, Cisco і ін.

– VoIP (Voice over IP) – технологія передачі звичайного комутируваного голосового трафіку поверх мережі на основі протоколів TCP/IP. Ця мережа не обов'язково повинна бути пов'язаною з Internet. Майже всі сучасні корпоративні мережі підтримують протоколи TCP/IP, використовуючи Ethernet або Fast Ethernet у локальних мережах, або послуги Frame Relay і ATM – у глобальних мережах.

Як видно, поняття IP-телефонії входить у це визначення, але можна дати й більше докладний опис:

– IP-телефонія – це самостійна послуга з передачі голосу, що представляє собою більше дешеву альтернативу традиційної телефонії;

– IP-телефонія – найбільш проста для реалізації послуга з пакета послуг, включаючи передачу даних і відео по протоколі IP. Більше того, передача голосу – не сама значна складова цього пакета. IP-телефонія сприяти повсюдному

поширенню електронної торгівлі, і додавати в інтерактивні мережні ігри або chat елемент живого спілкування.

Сьогодні комп'ютерна телефонія надає цілий спектр нових послуг. Маючи гнучкість і масштабованістю, КТ-системи вирішують широке коло завдань, як для великих корпорацій, так і для невеликих фірм і навіть часток осіб. CTI-технології забезпечують наступне:

– Голосова пошта.

– Уніфікований обмін повідомленнями.

– Факсимільні системи.

– Електронний секретар.

– Довідково-інформаційні системи.

– Системи оповіщення.

– Call-центри.

– Банківські системи (телебанкинг).

– Системи запису переговорів.

Головне достоїнство CTI – так звана відкритість систем, тобто вся комп'ютерна телефонія заснована на чіткій системі стандартів і, отже, системи CTI легко модифікуються й розширюються. При цьому досягається максимальна сумісність систем і їхніх компонентів. Дані переваги виявилися настільки очевидними, що не могли не позначитися на швидкому розвитку й широкому застосуванні систем CTI.

1.2 Область застосування

Областю застосування розроблювальної системи є IP-телефонія спеціального призначення застосовувана в державних структурах.

Компанії, що надають послуги IP-телефонії, працюють за наступною схемою: у спеціальний пристрій (шлюз) з однієї сторони підключаються телефонні лінії, а з іншого боку – IP-мережа (мережа Інтернет). Дзвінок, що

приходить із телефонної мережі міста А на шлюз у місті А, оцифровується, стискується за допомогою певного алгоритму, і у вигляді IP пакетів передається в мережу Інтернет. У заголовках пакетів утримується інформація про те, на який шлюз в IP-мережі повинні приходити ці пакети. Приходячі на шлюз у місті В IP-пакети перетворюються назад у телефонний сигнал і абонент у місті В піднімає трубку й розмовляє з абонентом А.

Кінцеві споживачі послуги можуть навіть не догадуватися, яким образом здійснюється цей дзвінок.

Оскільки при IP-телефонному дзвінку ніяк не задіяний міжнародний (міжміський) телефонний оператор, вартість цього дзвінка на порядок менше вартості традиційного телефонного з'єднання.

Однак дзвінок «Телефон-телефон» є самим очевидним, але далеко не єдиним сервісом, що може надавати оператор IP-телефонії. Рішення IP-телефонії комбінують голос і дані в одній мережі й пропонують не тільки дешеві міжнародні й міжміські дзвінки, але й цілий набір зовсім нових комунікаційних послуг будь-якому користувачеві – наприклад, дзвінок «Компьютер-телефон», «Интернет–Телефон», «Net-to-Fax».

Для того щоб працював вузол IP-телефонії, необхідний шлюз IP-телефонії, канал зв'язку до встаткування головного оператора мережі і телефонні лінії для прив'язки до місцевої телефонної мережі загального користування.

На початковому етапі необхідно визначитися, по якому каналі будете передавати трафік. Існує кілька можливих варіантів для роботи:

1. Ідеальним варіантом є передача трафіку до встаткування по виділеному каналу, орендованому в якого-небудь канального оператора. При цьому якість передачі мови буде високою.

2. Робота з публічним каналом мережі Інтернет можлива, але канал вашого інтернет-провайдеру повинен відповідати певним умовам (див. таблицю 1.1). При цьому швидкість підключення до провайдеру по публічних каналах мережі Інтернет на всьому каналі не гарантована.

3. При роботі через супутникові канали дуже велика ймовірність затримок, що не може не позначитися на якості послуг.

Необхідний IP-канал з так званим dedicated bandwidth (тобто з гарантованою смугою пропущення) мінімум 64 Кб/с і, відповідно, як мінімум 4 телефонні лінії. Кількість ліній означає кількість абонентів, що розмовляють одночасно. Ємність IP-каналу на одну лінію становить порядку 15 Кб/с, таким чином, на 4-х лінійний сервер необхідний виділений канал 64 Кб/с, на 8-ми лінійний – 128 Кб/с. При збільшенні кількості ліній вимоги до ємності каналу на 1 лінію зменшується (приміром, на 30-лінійний шлюз досить каналу 256 Кб/с).

Таблиця 1.1 – Характеристики, пропоновані до якості IP-каналу

Оцінка якості розмови по 5-ти бальній системі	Затримка, мсек*	Втрата пакетів, %*
Відмінно	150-200	0-3
Добре	200-350	4-6
Задовільно	350-1000	7-10
Неприйнятно	>1000	>10

* Параметр "втрата пакетів" впливає на якість розмови більше, ніж інші параметри.

Оренда телефонних ліній. У першу чергу необхідно визначитися з передбачуваною кількістю телефонних ліній для роботи вузла. Ми пропонуємо починаючим провайдерам почати свій бізнес із установки шлюзу на 4 або 8 аналогових телефонних ліній з наступним розширенням до одного або декількох цифрових потоків E1 (30 ліній). Далі уточнити у свого місцевого оператора або операторів доступу до телефонної мережі загального користування (ТФОП) можливості й вартість оренди аналогових телефонних ліній або цифрових каналів із серійним номером. Варто враховувати, що при використанні цифрових телефонних каналів потрібен телефонний інтерфейс ISDN PRI, R2 або E&M. У випадку, якщо телефонний оператор не може надати дані інтерфейси, але є

можливість надати іншу сигналізацію (наприклад R1,5), то можно використовувати конвертор сигналізацій українського виробництва.

Одне з найважливіших завдань для оператора зв'язку – це правильний вибір Автоматизованої Системи Розрахунків (АСР) або біллінгу. Система повинна бути незалежною від апаратної платформи, легко масштабованою, використовувати стандартні протоколи й мати простий і доступний користувальницький інтерфейс.

IP-телефонія в силу ряду технологічних і споживчих особливостей відрізняється від традиційного телефонного зв'язку. Ці відмінності враховувалися при рішенні питання ліцензування діяльності операторів по наданню послуг інтернет-телефонії. Саме тому інтернет-телефонія класифікується як різновид послуг телематичних служб.

Компаніям, які планують займатися IP-телефонією й уже володіють якою-небудь ліцензією на будь-яку іншу телематичну службу, необхідно подати заявку на розширення наявної ліцензії. При відсутності ліцензії подається заявка на нову ліцензію. Вимоги, пропоновані до компанії, що планує займатися наданням послуг IP-телефонії, перераховані в пакеті документів, які видаються по письмовому запиті в Мінзв'язку.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо дослідження поточного програмного забезпечення з теми бакалаврського проектування. До популярних програм IP-телефонії відносяться наступні:

– Skype (скайп).

– TiVi 602.

– Gizmo Project.

– Google Talk.

– WinLive Messenger.

– Yahoo Messenger.

Проведемо дослідження цих програм.

Skype (скайп)

Skype (вимовляється «скайп») – безкоштовне пропрієтарне програмне забезпечення із закритим кодом, що забезпечує шифрований голосовий зв'язок через інтернет між комп'ютерами (VoIP), а також платні послуги для зв'язку з абонентами звичайної телефонної мережі.

Програма також дозволяє робити конференц-дзвінки (до 25 голосових абонентів, включаючи ініціатора), відеодзвінки (у т.ч. відеоконференції до 10 абонентів), а також забезпечує передачу текстових повідомлень і файлів.

Програмні клієнти Skype випущені для операційних систем: Windows, Mac OS X, Linux, iOS, Windows Mobile, Google Android, PSP, Symbian.

На відміну від багатьох інших програм IP-телефонії, для передачі даних Skype використовує P2P-архітектуру. Каталог користувачів Skype розподілений

по комп'ютерах користувачів мережі Skype, що дозволяє мережі легко масштабуватися до дуже великих розмірів (у цей момент більше 100 мільйонів користувачів, 15-20 мільйонів онлайн) без дорогої інфраструктури централізованих серверів.

Крім того, Skype можуть маршрутизувати дзвінки через комп'ютери інших користувачів. Це дозволяє з'єднуватися один з одним користувачам, що перебувають за NAT або брандмауером, однак створює додаткове навантаження на комп'ютери й канали користувачів, підключених до інтернету прямо.

Єдиним центральним елементом для Skype є сервер ідентифікації, на якому зберігаються облікові записи користувачів і резервні копії їхніх списків контактів. Центральний сервер потрібний тільки для установки зв'язку. Після того як зв'язок установлений, комп'ютери пересилають голосові дані прямо один одному (якщо між ними є прямий зв'язок) або через Skype-посередник (супервузол – комп'ютер, у якого є зовнішня IP-адреса й відкритий TCP-порт для Skype). Зокрема, якщо два комп'ютери, що перебувають усередині однієї локальної мережі, установили між собою Skype-з'єднання, то зв'язок з інтернетом можна перервати й розмова буде тривати аж до його завершення користувачами або яким-небудь збоєм зв'язку усередині локальної мережі.

Завдяки використовуваним Skype кодекам (алгоритмам стиску даних) SVOPC (16 кГц), AMR-WB (16 кГц), G.729 (8 кГц) і G.711 (раніше використовувалися також ILBC і ISAC) і при достатній швидкості інтернет-з'єднання (30-60 Кбіт/с) у більшості випадків якість звуку порівнянна з якістю звичайного телефонного зв'язку.

При установці з'єднання між ПК дані шифруються за допомогою AES-256, для передачі ключа якого, у свою чергу, використовується 1024-бітний ключ RSA. Відкриті ключі користувачів сертифікуються центральним сервером Skype при вході в систему з використанням 1536– або 2048-бітних сертифікатів RSA.

VoIP-протокол Skype закритий і використовується тільки оригінальним програмним забезпеченням Skype. За допомогою API до його функцій можуть одержувати доступ програми сторонніх розроблювачів.

Офіційно підтверджених розроблювачем випадків розшифровки й/або перехоплення даних в Skype не зафіксовано.

Для стабільного використання відеозв'язку необхідна швидкість інтернет-з'єднання більше 200 Кбіт/с і бажана тактова частота процесора не менш 1 Ггц.

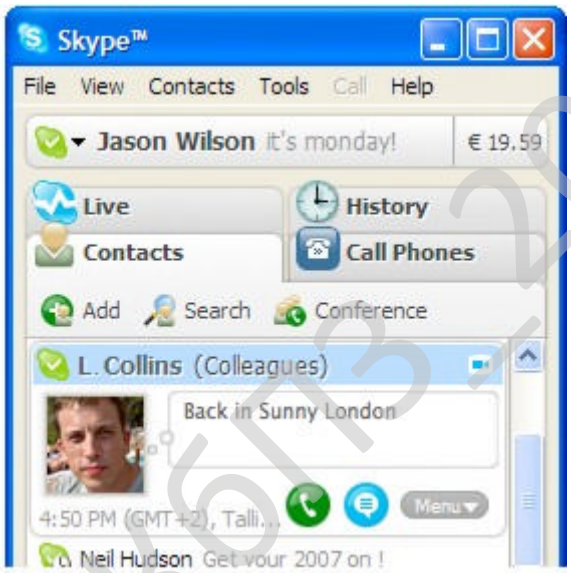

Рисунок 2.1 – Інтерфейс користувача Skype

Для установки цієї програми вам знадобитися скачати потрібний файл (скачати Skype), після чого почати процедуру інсталяції. На початку інсталяції вибирається мова інтерфейсу й запитується згода користувача з умовами ліцензування, а все інше відбувається автоматично. При першому запуску Skype запропоновано зареєструватися, що дуже зручно й не вимагає додаткових зусиль, тому що реєстрація закладена в саму програму. Вибирається ім'я користувача (skypename), що надалі буде використовуватися для вашої ідентифікації в мережі Skype. Після вибору ім'я створюється Skype-аккаунт, і можна починати працювати із програмою.

Skype простий не тільки в установці, але й у використанні. Щоб додати новий контакт, досить увести повне ім'я користувача або його частина, після чого програма автоматично запускає пошук і показує всі результати. Вам залишається тільки вибрати потрібний варіант. Пошук доступний також по містах, країнам, віку, адресі й іншим параметрам. В основному вікні Skype перебуває логічне й зрозуміле меню, виконане у вигляді закладок – заголовків. Якщо клацнути правою кнопкою миші на обраному контакті, з'являється перелік можливостей. У нижній частині вікна розташований символ дзвінкий, нажавши на який, можна подзвонити обраному адресатові. Під час зв'язку відкривається новий заголовок з актуальною інформацією про час і абонента, а також кнопки керування «Відкласти» і «Перервати дзвінок». Меню й розташування кнопок зрозумілі інтуїтивно. Єдиною проблемою іноді стає настроювання певної установки – оскільки можливості Skype досить широкі, перелік настроювань теж немалий.

Skype – дуже багата можливостями VoIP програма. Подзвонити з її допомогою можна не тільки тим користувачам, на комп'ютерах яких установлений Skype, але й абонентам фіксованих і мобільних мереж зв'язку по усьому світі. Крім того можна одержати так званий номер Skypeln, по якому зможуть додзвонитися абоненти мобільних і фіксованих мереж. Через Skype особливо зручно дзвонити за рубіж, тому що дзвінки в багато держав будуть коштувати дешевше чим дзвінки із простого мобільника. Крім іншого, Skype дозволяє здійснювати аудіо- і відеоконференції. Для цього потрібні всього лише мікрофон, навушники або динаміки й веб-камера (для відеоконференцій). Програма надає широкі можливості обміну повідомленнями, можна навіть створити, Mass chat – розмову, у якому візьмуть участь кілька співрозмовників.

Skype пропонує широкий вибір емотиконів (або як їх називають "смайлики"), за допомогою яких можна виражати свої відчуття під час бесіди. Правда, вони убудовані в програму, і додавання нових символів неможливо. Кожний користувач Skype може внести у свій профіль не тільки особисту інформацію (адреса, номер телефону, ім'я й т.д.), але й свою фотографію й

забавний текст. За допомогою програми можна пересилати файли, переказувати гроші, використовуючи свій рахунок PayPal або Skype, переглядати архів розмов, шукати різну інформацію й навіть завести голосову пошту. Опція Skype Extras дозволяє розширити функції програми. Наприклад, додати додаток, що буде записувати всі розмови й зберігати їх у форматі MP3, або ж установити гру, у яку можна буде грати зі своїми співрозмовниками.

В основному, Skype працює без проблем, однак дзвінки в далекі країни, наприклад, у США, можуть іти із затримкою. Іноді трапляється так, що тлом розмови чуєш власний голос. Переважно це трапляється при неправильному конфігуруванні (наприклад, занадто голосно настроєному звуці) або ж низької пропускної здатності інтернет-каналу. Якість звуку не викликає дорікань, так само як і швидкість передачі файлів (залежно від швидкості інтернет-трафіку). Однак слід зазначити, що програма вимагає досить більших ресурсів комп'ютера. Використання навіть для простого зв'язку старого комп'ютера може стати проблематичним.

Виводи: Популярна, багатофункціональна й дуже зручна у використанні програма, доступна навіть новачкам. Хотілося б, щоб Skype споживав небагато менше ресурсів, а тарифи дзвінків у мобільні мережі були нижче.

TiVi 602

Відразу впадає в око, що займає вона набагато менше місця, ніж інші представники VoIP програм. Перевага цієї програми складається й у тому, що не потрібно інсталювати – досить просто відкрити завантажений файл. На жаль, сама програма не включає реєстраційні можливості й посилання на реєстраційну сторінку. Тому доводиться відправлятися на сайт розроблювачів і реєструватися там, після чого можна приступати до роботи. Якщо ви хочете просто зв'язатися з іншим комп'ютером, а не дзвонити абонентам фіксованого зв'язку, реєстрація взагалі не потрібна. Ця маленька особливість істотно спрощує перший запуск програми. Варто додати, що приймати дзвінки можна, не реєструючи ім'я

користувача TiVi, тому що додзвонитися до іншого комп'ютера можна просто використовуючи IP адресу.

TiVi – дуже компактна програма, з якою дуже легко працювати. Щоб подзвонити, досить набрати номер, адресу IP або ім'я користувача TiVi, і нажати відповідну кнопку. Програмне вікно дуже маленьке, але на його правій стороні розташовані символи, за допомогою яких можна набрати потрібний номер або ім'я користувача TiVi, з яким ви хочете зв'язатися. У нижній частині вікна перебувають три кнопки – «Дзвонити», «Перервати дзвінок», «Почати передачу повідомлення». Зв'язатися з будь-яким користувачем можна без його згоди, що дуже зручно, однак іноді може й перешкодити. Наприклад, якщо дзвінок або повідомлення надходять від небажаного абонента. Єдиний мінус – маленькі й майже непомітні іконки установок, телефонної книги й іншої інформації, розташовані на одній панелі з полем введення адресата. Але до цьому можна швидко звикнути, тому подальша робота з невеликими іконками особливих проблем не відносить.

Програма компанії "TiltsVisiem" дуже проста не тільки у використанні, але й по своїй функціональності. Її головне завдання – забезпечити зв'язок з користувачем іншого комп'ютера через інтернет, подзвонивши або відіславши йому повідомлення. У програму інтегрована елементарна телефонна книга, що дозволяє створювати групи контактів, приєднувати або видаляти їх. Подвійне клацання миші на поле контакту не тільки активізує дзвінок або відправляє повідомлення, але й показує в поле адресата вартість дзвінка в мережі TiVi. Доступна й версія програми, що дозволяє здійснювати відеодзвінки, але поки тільки в тестовому режимі. Будемо сподіватися, що незабаром і цей продукт буде доопрацьований в остаточній версії, що буде значним функціональним доповненням. Програма TiVi працює без проблем.

Якість передачі голосу досить висока, відправлення повідомлень відбувається без затримок. З'єднання виробляється швидше, ніж в інших програм. При цьому утиліта не тільки невелика, але й вимагає мінімальних ресурсів. Ця

особливість порадує власників застарілих комп'ютерів і систем з невеликим обсягом оперативної пам'яті.

Виводи: TiVi виразно не розчарує тих, що хоче просто й зручно спілкуватися за допомогою протоколу IP. Програма невелика, вимагає мінімальних ресурсів і дуже проста у використанні. Однак аматори поспілкуватися в чаті будуть розстроєні бідним вибором емотиконів (смайликів) і недоліком додаткових функцій, характерних для такого роду програм.

Gizmo Project

Gizmo5 (раніше Gizmo Project) – безкоштовне програмне забезпечення й однойменний сервіс для VoIP. Версії програми-клієнта доступні для Microsoft Windows, Linux і Mac OS X. Аналогічний по можливостях Skype, але заснований на відкритому протоколі SIP і має додаткові можливості, відсутні в Skype або пропоновані за додаткову плату.

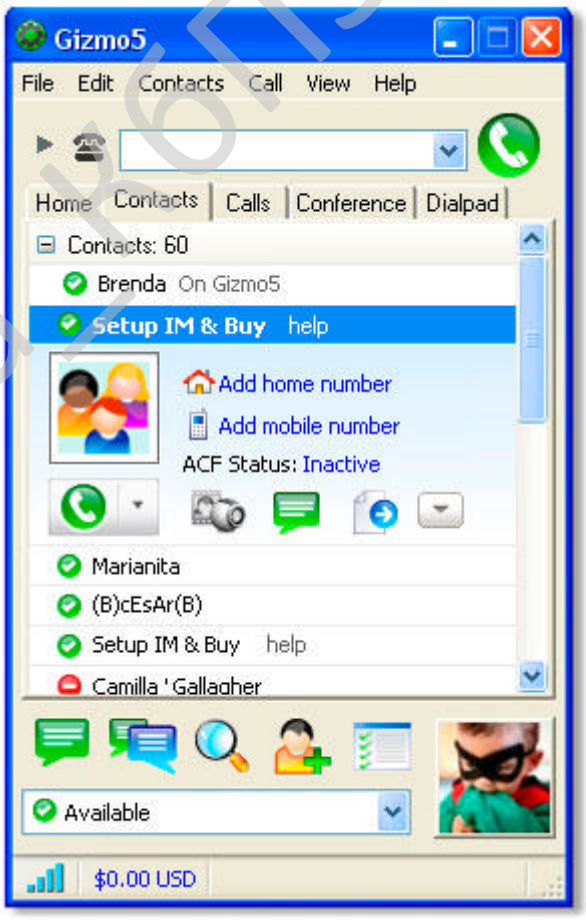

Рисунок 2.2 – Інтерфейс користувача Gizmo5

Можливості:

– Безкоштовна голосова пошта.

– Передача текстових повідомлень у режимі чату.

– Безкоштовні дзвінки усередині мережі користувачів Gizmo5.

– Одночасна реєстрація в мережу Gizmo5 і в інших мережах: наприклад, на внутрішній SIP IP-PBX компанії.

– Можливість спілкуватися (як голосові дзвінки, так і текстові повідомлення) з учасниками Skype за допомогою OpenSky.

Модуль обміну текстовими повідомленнями використовує протокол XMPP.

– Підтримка спілкування з іншими XMPP мережами.

– Підтримка спільних кімнат спілкування XMPP (наприклад, conference.jabber.ru).

У листопаді 2009 Gizmo5 був придбаний компанією Google. У даний момент прийом нових користувачів зупинений, програмою можуть користуватися тільки раніше зареєстровані користувачі.

Gizmo Project це невелика безкоштовна програма для забезпечення передачі голосового сигналу в мережі інтернет, тобто працююча по системі VoIP. Популярність такого роду програм була визначена приходом у маси високошвидкісного інтернету, тому що сьогодні оцифрувати голосове повідомлення й передати його по мережі для багатьох користувачів не є проблемою. Іншим істотним плюсом цих програм є мала ціна переговорів у порівнянні зі стандартними видами телефонного зв'язку, причому тарифи практично однакові для дзвінків у будь-які країни миру. При цьому дзвонити можливо не тільки абонентам, що мають вихід в інтернет і відповідне програмне забезпечення, але й на стаціонарні й мобільні телефони.

Gizmo Project у ряді подібних програм займає особливе місце, тому що відрізняється використанням відкритого коду, і тому має купу додаткових можливостей, які в тому же Skype не передбачені або підключаються за окрему

плату. Властиво, можливості Gizmo Project аналогічні його «старшому братові» Skype, але все з тим же застереженням на відкритий код. У своїх можливостях Gizmo блищить кроссплатфоменістю, його дистрибутив поширюється як для Microsoft Windows, так і для Mac OS X і Linux.

Можливості для голосового спілкування в Gizmo Project такі ж, як і в інших програм із цього класу. Можливо безкоштовно спілкуватися з людьми з будь-якої точки планети, у яких на комп'ютері встановлений дистрибутив цієї програми. Якщо такого не є, то також легко за допомогою Gizmo можна дзвонити на міські й стільникові телефони, правда вже відповідно до тарифних планів. Крім цього, можливо й самі приймати за допомогою Gizmo Project дзвінки зі звичайних телефонів, для цього потрібно придбати власний номер, оренда якого коштує 5 доларів на місяць. Особливості використання програмою Gizmo відкритого програмного забезпечення SIP (Session Initiation Protocol), дозволяє їй бути сумісною з іншими клієнтами, що працюють на основі протоколу VoIP. Про це зараз відкрито заявляють розроблювачі сервісу, готові спільно працювати з будь-якими VoIP-мережами. Має Gizmo Project крім передачі голосових повідомлень безліч додаткових функцій. Для користувачів цієї програми працює безкоштовна голосова пошта, що дає можливість залишати адресатові голосове повідомлення, що він може прослухати пізніше. Це дуже зручний сервіс, тому що якщо мобільний телефон ми носимо із собою практично скрізь, а от апаратуру для голосового зв'язку через інтернет поки немає.

Для тих випадків, коли необхідно передати текстове повідомлення, Gizmo Project має свій сервіс, побудований на протоколі Jabber. Тут можливо передавати повідомлення як окремим користувачам цієї мережі, так і вступати в конференції в режимі спільних кімнат спілкування.

До додаткових можливостей роботи з голосовими повідомленнями в Gizmo Project можна віднести організацію сполучення конференц-зв'язку й запису розмов на жорсткий диск одним клацанням мишки. Крім цього, є ще автовідповідач, функція розпізнавання голосу й утиліта для визначення на карті

місцезнаходження вашого співрозмовника. Можно випробувати функцію варіювання якості переданого оцифрованного мовного сигналу за допомогою зміни ширини пропущення інтернет-каналу. Примітно, що випускається дистрибутив Gizmo Project не тільки для основних операційних систем, але й для різних моделей «розумних» телефонів.

У підсумку потрібно сказати, що Gizmo Project непогано заявила про те, яким може бути гарне безкоштовне програмне забезпечення. Його функціонал дорівнює, а подекуди й перевершує такого монстра як Skype, і може скласти йому реальну конкуренцію. Крім того, якщо вірити обіцянкам розроблювачів Gizmo Project, найближчим часом у багатьох країнах будуть введені безкоштовні дзвінки на звичайні телефони в тому випадку, якщо цей номер зареєстрований у профільних даних в іншого користувача програми. Так що розвиток триває, і ми сподіваємося в майбутньому побачити ще багато нових корисних інструментів від творців Gizmo.

Google Talk

Google Talk – пропрієтарна програма миттєвого обміну повідомленнями із закритим вихідним кодом, розроблена компанією Google.

Google Talk дозволяє спілкуватися за допомогою голосового чату й текстових повідомлень. Особливістю Google Talk є тісна інтеграція з поштовою службою Gmail (наприклад, по Google Talk приходять повідомлення про нові повідомлення). Для використання Google Talk обов'язкова наявність облікового запису Gmail.

Як клієнт можливий використання сторонніх додатків таких, як Psi, Miranda IM, iChat і інших. Користувачі Google Talk можуть спілкуватися з користувачами інших XMPP-серверів відповідно до загальної архітектури протоколу XMPP.

Як заявляють власники компанії Google, ціль їхньої роботи це організація всієї світової інформації для того, щоб зробити її більше доступній і корисної кожній людині. Однак, дивлячись на те, якими кроками розвивається ця

компанія, здається, що щирі її мети куди більше масштабні, Google скуповує багато популярних мережних сервісів, створює свої різні служби й додатки, не боїться вступати в конфронтації з такими гігантами як Microsoft і навіть запускає на космічну орбіту власні супутники.

Один з успішних проектів компанії в області комунікації це мессенджер за назвою Google Talk. Він дає можливість спілкуватися людям як за допомогою коротких текстових повідомлень, так і встановлюючи голосовий зв'язок. Дана програма працює через протокол Jabber/XMPP, не самий популярний у Росії, але досить солідний приріст, що показує останнім часом, користувачів. Тому що технологія Jabber відкрита, можна використовувати для обміну повідомленнями й клієнт Google Talk, і будь-який сторонній додаток, що працює на Jabber.

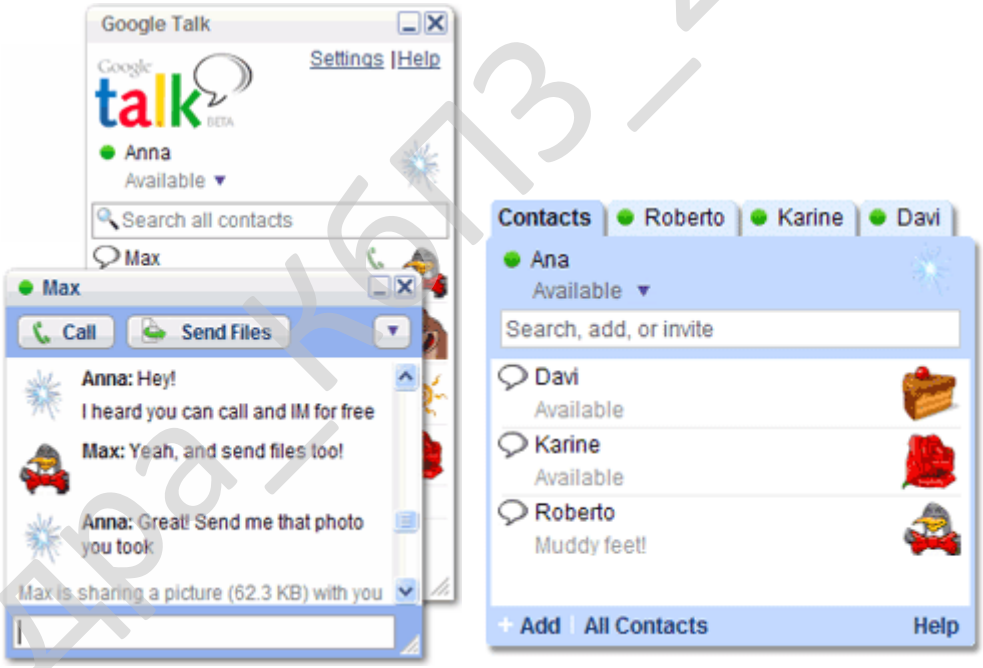

Рисунок 234 – Інтерфейс користувача Google Talk

Google Talk тісно пов'язаний з іншим популярним сервісів від компанії Google, з поштовим клієнтом GMail. При реєстрації аккаунта в GMail заводиться й обліковий запис в Google Talk, причому в останній включена функція автоматичної перевірки поштової скриньки. Крім цього, реалізована можливість

синхронізації контактів в Google Talk з адресною книгою в GMail. Також з особливостей хочеться відзначити зберігання історії переписки на віддаленому сервері, чого так не вистачає користувачам ICQ. Адже іноді при некоректній переустановці ICQ-клієнта ми можемо втратити настільки коштовні «щоденники», та й, користуючись для доступу в мережу різними комп'ютерами, хочеться завжди мати під рукою повну історію переписки, а не тільки її окремі клаптики.

Цікава й дуже зручна реалізація роботи клієнта Google Talk. Вам не потрібно завантажувати і встановлювати дистрибутив програми, досить лише зайти на сторінку Google і нажати кнопку «Установити». Клієнт працює як звичайне Flash-додаток, приліплене до будь-якої сторінки браузера. Незвичайно, але дуже зручно, особливо тим, хто не любить установлювати на свій комп'ютер «зайві» програми й запускати непотрібні процеси. До того ж, можливо розмістити віконце додатка в будь-якій частині сторінки, наприклад, аж унизу свого блогу. У цьому випадку люди зможуть спілкуватися прямо звідти. Одним з головних переваг Google Talk є можливість безкоштовного голосового спілкування з кожним з контактів з вашої адресної книги, також можна дзвонити за гроші на звичайні телефони, як і в інших програмах, що забезпечують з'єднання за допомогою VoIP.

Програма Google Talk абсолютно безкоштовна, вона коректно працює на системах Windows старше версії 2000, з'єднання з інтернетом необхідно хоча б на швидкості 56 Кбіт/с. Шанувальники Mac OS і Linux поки обділені розроблювачами від Google, але в них досить інших додатків, які працюють по тім же протоколі, що й Google Talk. Підтримує «інтернет-звонилку» від Google функцію голосової пошти, тобто вхідний дзвінок при вашій відсутності зберігається в поштовій скриньці, про що неодмінно відправляється повідомлення. Загалом, всі як у пристойних будинках, необхідні функції для голосового зв'язку присутні, однак є й недоліки, характерні для всіх подібних сервісів, такі як проблеми з якістю переданого звуку.

Як уже було сказано, Google Talk працює разом з поштовиком GMail, тому якщо у вас немає зареєстрованого аккаунта, то прийдеться його завести. Однак цей не весь сервіси, з якими може синхронно працювати програма. Як відомо, компанії Google належить площадка для розміщення відеороликів YouTube, і за допомогою нашого мессенджеру можна переглядати ролики безпосередньо в його діалоговому вікні. Крім того, Google Talk має можливість перегляду слайдшоу з веб-сервісів Flickr і Picasa.

Судячи з активної політики завоювання різних ринків в інтернеті, мессенджер від Google буде й далі нарощувати потужності й одержувати всі нових користувачів. Це й не дивно, коли функції програми спрямовані, насамперед, на задоволення потреб користувачів, і розроблювачі не бояться освоювати нові цікаві додатки, то успіх напевно буде гарантований.

WinLive Messenger

Windows Live Messenger – програма миттєвого обміну повідомленнями для Windows. Є спадкоємцем програми MSN Messenger і випущена під новим ім'ям компанією Microsoft 13 грудня 2005 року. Є одним з компонентів Windows Live – набору мережних служб від Microsoft.

Клієнт підключається до Microsoft .NET Messenger Service. Корпорації також можуть інтегрувати власний Live Communication Server і Active Directory у робочу мережу для своїх клієнтів. Головні клієнти з мультипротоколами також можуть підключитися до служби.

Програма, раніше відома як MSN Messenger, знайшла нове ім'я й приєдналася до сімейства Windows Live. Додаток не тільки перемінив зовнішній вигляд, але й стало більше функціональним. При інсталяції пропонується встановити не тільки саму програму, але й вибрати msn.com у якості стартової інтернет-сторінки, завантажити Windows Live Toolbar та інші дріб'язки. Це за замовчуванням відзначено галочками, і неуважний користувач може ненавмисно їх активізувати, сам того не бажаючи. Для використання Live Messenger необхідний аккаунт Windows Live, який можна відкрити за адресою get.live.com.

Доступ на цей ресурс можливий із самої програми, але краще було б, якби реєстраційна система вже була в неї інтегрована.

Після завершення реєстрації на екрані монітора з'являється список контактів, упорядкований по групах або ж за абеткою. Правий «клік» на контакті викликає меню з комунікаційними й іншими можливостями. При спілкуванні в режимі чату у верхній частині вікна доступні всі опції зв'язку, а в нижньої – вікно для введення тексту й вибір емотиконів (смайликів). Недосвідчений користувач, можливо, стикнеться із труднощами при пошуку звичного меню у верхній частині вікна (File, Tools і т.д.). Підкоряючись задуму дизайнера, це меню ховається за кнопками, розташованими поруч із символом мінімізації.

Цей засіб спілкування має дуже широкі можливості персоніфікації. У кожного вікна можна змінити схему кольорів (щоб, наприклад, при діалозі з колегами букви були синіми, а з дітьми – червоними). Точно так само для кожного вікна можна встановити своє тло. Програма має у своєму розпорядженні досить широкий вибір емотиконів (смайликів), однак на відміну від Skype, Windows Live Messenger дозволяє приєднати до їхнього списку будь-яке зображення, що будуть бачити всі співрозмовники. Можна робити й більш барвисті знаки уваги, відсилаючи невеликі flash-анімації (winks), доповнені звуковими ефектами. Голосове спілкування в Live Messenger відбувається в тому же вікні, де й чат. Складається таке враження, що програма більше орієнтована на комунікації за допомогою букв, ніж голосу. На своїй домашній сторінці творці продукту запевняють, що за допомогою Live Messenger можна дзвонити абонентам мобільної й фіксованої мережі. Однак ретельно дослідивши програму й інструкцію виробника, нам так і не вдалося знайти відповідне меню й здійснити телефонні дзвінки. Можливо, ця послуга просто недоступна користувачам певних країн, тому що Windows Live Messenger передбачений, головним чином, для ринку США.

За допомогою веб-камери програма підтримує й відеорозмови. Вони здійснюються в тому же вікні, що й чат, хіба що фотографія в контактах

заміняється на передане камерою відеозображення. Користувачам надається також можливість пересилати файли, грати в спільні ігри й відсилати SMS на незареєстрованні в контактах номера мобільних телефонів.

Одна з найбільш корисних функцій Windows Live Messenger – Remote assistance, або можливість віддаленої допомоги. Вона дозволяє звернутися по допомогу до іншого користувача, з його згоди одержати віддалений доступ до його комп'ютера й працювати з ним, як зі своїм власним. Безумовно, Remote assistance буде корисна у випадках, коли потрібен доступ до файлів або документів, які перебувають на відстані. Варто додати, що Live Messenger дозволяє спілкуватися й з користувачами Yahoo! Messenger.

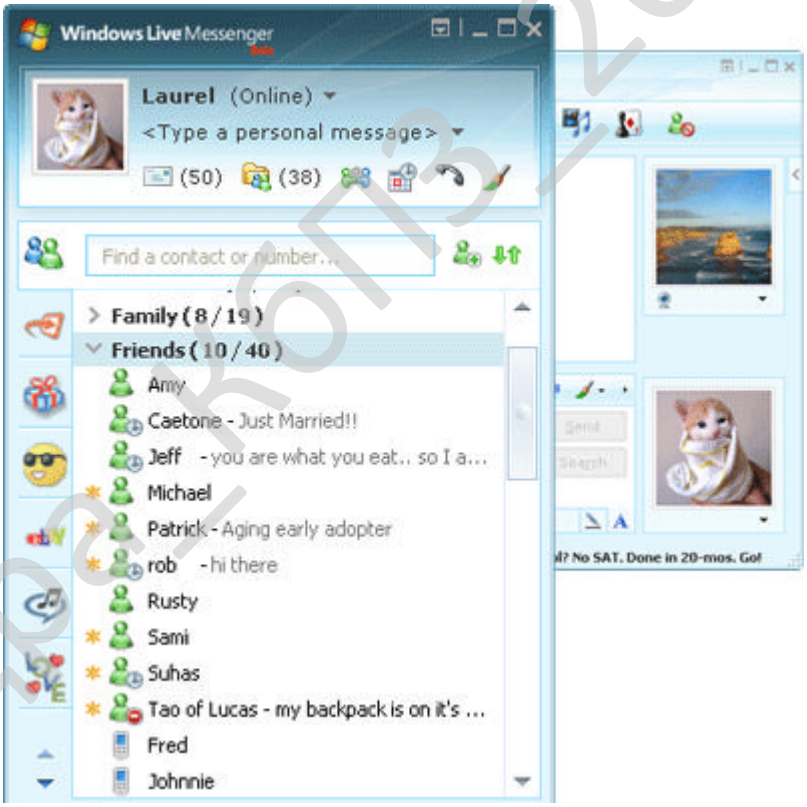

Рисунок 2.4 – Інтерфейс користувача Windows Live Messenger

На жаль, при використанні Windows Live Messenger ми кілька разів зіштовхувалися із проблемами при спробі комусь подзвонити. Програма може «підвиснути» і, оскільки володіє дуже візуально насиченим інтерфейсом, дуже

вимоглива до технічних ресурсів комп'ютера. Однак з обміном повідомленнями все в порядку – послання завжди знаходить свого адресата. Якість звуку теж гарне, якщо ви володієте досить високошвидкісним інтернет-каналом.

Виводи: Windows Live Messenger призначений в основному для чату – складається враження, що функція «Подзвонити на інший комп'ютер» є всього лише факультативної. Програма візуально багата й оснащена безліччю доповнень, пов'язаних з текстовим спілкуванням. На жаль, їй не вистачає можливості просто подзвонити абонентові мобільної або фіксованої мережі.

Yahoo Messenger

Yahoo! Messenger (скорочено Y!M) – програма миттєвого обміну повідомленнями компанії Yahoo!, що використовує власний протокол. Існує для платформ Windows, Mac, Linux (Unix). Поширюється безкоштовно. Для використання потрібна попередня реєстрація на порталі Yahoo!

Програма призначена для спілкування в режимі реального часу з іншими користувачами інтернету. Передбачено наступні можливості: текстове спілкування, голосове спілкування, зокрема багатокористувальницький голосовий чат, відеоконференції, дзвінки на мобільні й стаціонарні телефони, обмін файлами, онлайнові гри, що набудовуються онлайнові трансляції музики, що набудовується інтегрований доступ до сервісів порталу Yahoo! Не має версії з русифікованим інтерфейсом, але розробки в цьому напрямку вже ведуться.

Використовуючи новітню версію, її користувач може зв'язатися й із власником Windows Live Messenger, що істотно розширює комунікаційні можливості. Тому що ця комунікаційна програма передбачена, в основному, для ринку США, то російською мовою меню програми недоступно. На жаль, творець Yahoo! Messenger – компанія Yahoo – хоче, щоб користувач не тільки встановив програму, але й зробив Yahoo! стартовою сторінкою й пошуковою системою за замовчуванням. Тому треба бути уважним і відзначити, що хочеш від цього відмовитися. При виборі типового пакета автоматично інсталюється й Yahoo! Toolbar, і Yahoo! Popup Blocker. Це дуже дратує, тому як більшість користувачів

вибирають не Typical, а Custom інсталяцію. Сам процес установки займає набагато більше часу, ніж у випадку з іншими VoIP програмами. Варто додати, що інсталяційна програма повідомляє про неможливість працювати разом з пошуковою системою Firefox, що теж треба враховувати. Після неймовірно тривалого процесу установки можна, нарешті, приступитися до звичної реєстрації серед користувачів. Для роботи з Yahoo! Messenger необхідний Yahoo ID. Якщо ви вже зареєстрували свою поштову адресу на Yahoo.com, тої же ідентифікатор можна використовувати, реєструючись у системі. Якщо ж ідентифікатора ні, його можна одержати на вищезгаданому ресурсі. При першому запуску програми пропонується імпортувати контакти з інших місць, наприклад, з адресної книги електронної пошти. Після цього можна починати роботу. Відразу стає помітно, що програма перезавантажена, тому орієнтуватися в меню досить важко. У нижній частині вікна перебувають установлені доповнення (plug-ins), вікно пошуку Yahoo і, на жаль, реклама. У верхній частині програмного вікна розташоване звичне меню – Contacts, Actions і т.д. Як і в інших програм, всі дії з Yahoo Messenger можна здійснювати через правий клік на обраному ім'ї в списку контактів. Клацнувши на Send an instant message, відкриваємо вікно обміну повідомленнями. Це просто й зрозуміло – меню у верхній частині, відображення розмови посередині, а поле введення повідомлення, вибір емотикона (смайликів) і різні настроювання – у нижній частині.

Як і будь-яка VoIP програма, Yahoo Messenger дозволяє дзвонити на комп'ютер іншого користувача за допомогою списку контактів, а також на мобільні й фіксовані телефони всього миру. Те саме що Windows Live Messenger, ця програма запитує згоди на контакт того абонента, з яким ви хочете зв'язатися.

Тільки після одержання дозволу можна дзвонити й посилати повідомлення. Для голосових дзвінків потрібно абонувати Yahoo Voice. Подзвонити дуже просто, треба всього лише вибрати відповідний номер і нажати кнопку «Дзвонити». Приблизно так само можна подзвонити на інший комп'ютер, тільки для цього треба вибрати ім'я користувача в контактному аркуші. У добавок

до звичних функцій, Yahoo Messenger пропонує й різні додаткові. Багато хто з них доступні за замовчуванням – наприклад, можна слухати інтернет-радіо за допомогою Lounchcast, дізнаватися прогноз погоди по усьому світі на Yahoo Weather і навіть створити свою сторінку на Yahoo360, де можна публікувати свої особисті мемуари, фотографії або розповідати про себе. Варто відзначити й широкі можливості персоніфікації Yahoo Messenger – можна встановити різні теми, поміняти тло й т.д. Програма укомплектована додатковими можливостями, такими як відеодзвінок, відеоконференція, одержання й відсилання файлів, мережні ігри, послуги електронної пошти й SMS.

Рисунок 2.5 – Інтерфейс користувача Yahoo! Messenger

Yahoo Messenger дозволяє без проблем дзвонити на інший комп'ютер або передавати повідомлення іншим способом. Хіба що при контактах з користувачами Windows Live Messenger повідомлення іноді відсилаються досить довго, а в контактному аркуші присутній онлайн користувач відображався, як

відсутній. Радує те, що при рівних можливостях Yahoo займає набагато менше ресурсів, чим Live Messenger або Skype.

Виводи: Yahoo Messenger – багатофункціональна програма, що споживає набагато менше ресурсів, ніж аналогічні продукти конкурентів. Однак досить неприємна необхідність установки зовсім непотрібних програм і неймовірно тривалий процес інсталяції.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium,

використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4

забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні

64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

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

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки. Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки IP-телефонії спеціального призначення.

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

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

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

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

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

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Інтернет фундаментально змінює наші подання й про телефонію й про способи комунікації. Хоча телефонні мережі й мережі передачі даних співіснували протягом десятиліть, вони розвивалися незалежно друг від друга. IP-телефонія поєднує їх у єдину комунікаційну мережу, що пропонує потужний і економічний засіб зв'язку. Десятки компаній по усьому світі пропонують комерційні рішення для IP-телефонії. Всі великі телекомунікаційні компанії почали дослідження, з метою краще зрозуміти перспективи, що відкриваються. Рішення IP-телефонії комбінують голос і дані в одній мережі й пропонують дешеві міжнародні й міжміські дзвінки й цілий набір комунікаційних послуг будь-якому користувачеві.

Загальний принцип дії телефонних серверів IP-телефонії такий: з одного боку, сервер пов'язаний з телефонними лініями й може з'єднатися з будь-яким телефоном миру. З іншого боку, сервер пов'язаний з Інтернетом і може зв'язатися з будь-яким комп'ютером у світі. Сервер приймає стандартний телефонний сигнал, оцифровує його (якщо він вихідно не цифровий), значно стискає, розбиває на пакети й відправляє через Інтернет по призначенню з використанням протоколу Інтернет (TCP/IP). Для пакетів, що приходять із мережі на телефонний сервер і, що йдуть у телефонну лінію, операція відбувається у зворотному порядку. Обидві складові операції (вхід сигналу в телефонну мережу і його вихід з телефонної мережі) відбуваються практично одночасно, що дозволяє забезпечити повнодуплексну розмову. На основі цих базових операцій можна побудувати багато різних конфігурацій. Допустимо, дзвінок телефон-комп'ютер або комп'ютер-телефон може забезпечувати один телефонний сервер. Для організації зв'язку телефон(факс)-телефон(факс) потрібно два сервери.

Технології IP-телефонії

Мережа IP-телефонії (відповідно до рекомендацій ITU-T H.323) являє собою набір наступних пристроїв, з'єднаних по IP-мережі:

– шлюз (gateway);

– диспетчер (gatekeeper);

– монітор (administration manager).

Архітектура мережі IP-телефонії являє собою з'єднані по IP-мережі Шлюзи у телефонну мережу, які надають безпосередній інтерфейс абонентові й здійснюють кодування, стиск і пакетизацию голосу/факсу і їхнє відновлення. Весь механізм взаємодії шлюзів і облік виробляється Диспетчерами. Диспетчерами віддаленого конфігурування й адміністрування мережі може бути використаний Монітор. Ці три компоненти в різних виробників можуть називатися по-різному, але всі вони виконують функції, узагальнені вище.

Шлюз – необхідний пристрій, підключений до IP-мережі й до телефонної мережі (PBX/PSTN). Функції:

– відповідь на виклик викликаємого абонента PBX/PSTN;

– установлення з'єднання з віддаленим шлюзом;

– установлення з'єднання з викликуваним абонентом PBX/PSTN;

– стиск, пакетування й відновлення голосу (або факс-сигналу).

У такий спосіб шлюз, або Gateway, – це основна й невід'ємна частина архітектури IP-телефонії, безпосередньо з'єднуюча телефонна мережа з мережею IP. Шлюзи різних виробників відрізняються способом підключення до телефонної мережі, ємністю, апаратною платформою, реалізованими кодеками, інтерфейсом і іншими характеристиками. Але всі вони виконують перераховані вище функції, що є базовими для технології IP-телефонії.

Диспетчер, або GateKeeper, – це додатковий пристрій, підключений тільки до IP-мережі й несуче в собі всю логіку роботи мережі IP-телефонії. Його основні функції:

– автентифікація й авторизація абонента;

– розподіл викликів між шлюзами;

– біллінг.

Як правило диспетчер не містить у собі закінченої біллінгової програми, а тільки заснований на стандартах інтерфейс до професійних систем біллінгу третіх виробників, а також API для розробки оператором власної біллінгової програми.

Диспетчер необхідний у будь-якій мережі IP-телефонії, що містить більше двох шлюзів. У перших шлюзах (у перших host-based версіях VocalTec, Vienna і ін.) функції диспетчера в їхньому примітивному виді виконувалися самим шлюзом. З розвитком технології й ростом мереж IP-телефонії, функції диспетчера були винесені в окремий модуль. Хоча в деяких виробників диспетчер може фізично перебувати на одній системі зі шлюзом, логічно це самостійний модуль.

Монітор – необов'язковий додатковий модуль мережі IP-телефонії, що підключається тільки до IP-мережі, використовуваний для віддаленого конфігурування й підтримки інших пристроїв мережі – шлюзів і диспетчерів. Функції: інтерфейс для віддаленого настроювання через IP-мережу параметрів шлюзів і диспетчерів мережі IP-телефонії.

Монітор є зручним засобом конфігурування й адміністрування мережі. У перших шлюзах для цього просто використовувалися стандартні мережні додатки, такі як pcAnywhere. Пізніше з метою оптимізації роботи виробники встаткування IP-телефонії стали випускати власні додатки для цих цілей.

Стандарти

Стандарти є критичним чинником для миру IP-телефонії. Одна з найбільш важливих областей стандартизації – протокол обміну повідомленнями в IP-телефонії. Ранні рішення IP-телефонії використовували для зв'язку один з одним закриті протоколи. Обоє учасника бесіди повинні були мати аналогічні продукти. Intel і Microsoft очолили напрямок на розробку стандартів на основі H.323, рекомендованого International Telecommunications Union (ITU). Цей стандарт

формулює технічні вимоги для передачі аудіо- і відеоданих по мережах передачі даних. H.323 містить у собі:

1. Стандарти на відео кодер/декодери.
2. Стандарти на голосові кодер/декодери.
3. Стандарти на загальнодоступні додатки.
4. Стандарти на керування викликами.
5. Стандарти на керування системою.

Стандарти на відео кодер-декодери не потрібні для обробки телефонних дзвінків, але існують усередині тої ж системи стандартів.

Технічні вимоги до голосових кодерів включають наступні:

- мала смуга пропущення (8 kbit/s або менше);
- висока якість голосу;
- невеликі затримки;
- можливість реконструкції загублених пакетів.

При передачі в режимі реального часу до 30% пакетів можуть втратитися або спізнитися (що в режимі реального часу те саме). Гарний додаток IP-телефонії повинне відшкодувати недостачу пакетів, відновивши загублені дані. Сам алгоритм кодування також впливає на відновлення даних. Складні алгоритми збільшують вартість необхідного встаткування. Найбільш популярним алгоритмом кодування є G.723.1.

Ще одна особливість полягає в тому, що системи IP-телефонії повинні мати можливість підтримувати різні кодери й додавати нові по необхідності. H.323 був спочатку розроблений для локальних обчислювальних мереж, так що змінна ширина смуги частот і час затримки Інтернет зменшують корисність деяких елементів H.323. За замовчуванням голосовим кодер-декодером у стандарті H.323 є G.711. Однак ширина смуги частот в 64 kbps, необхідна в G.711, неприйнятна при використанні в Інтернет, тому що більшість користувачів Інтернету має канал свідомо меншої ширини. Але навіть у цьому випадку багато чого зі стандарту є корисним.

Крім G.711 стандарт H.323 визначає звукові кодер-декодери G.722, G.723,G.723.1, MPEG1, G.728, і G.729. Кодери з низькою шириною смуги частот – G.729 в 8 kbps і G.723 в 5.3/6.3 kbps – цілком підходять для використання в Інтернет. Зокрема, G.723 є одним з декількох "стандартних" кодерів для IP-телефонії, особливо після того, як Intel, Microsoft і Netscape оголосили про підтримку цього кодеру. Основний недолік G.723 полягає в тому, що він вимагає досить великих ресурсів процесора.

Протоколи сімейства H.32x

17 листопада 2001 року була схвалена четверта версія стандарту H.323 . Зараз H.323 – один з найважливіших стандартів із цієї серії. H.323 – це рекомендації ITU-T для мультимедійних додатків в обчислювальних мережах, що не забезпечують гарантовану якість обслуговування (Qo). Такі мережі містять у собі мережі пакетної комутації IP і IPX на базі Ethernet, Fast Ethernet і Token Ring.

Рекомендації H.323 передбачають:

– Керування смугою пропущення.

– Можливість взаємодії мереж.

– Платформну незалежність.

– Підтримку багатоточечних конференцій.

– Підтримку багатоадресної передачі.

– Стандарти для кодеків.

– Підтримку групової адресації .

Передача аудіо- і відеоінформації досить інтенсивно навантажує канали зв'язку, і, якщо не стежити за ростом цього навантаження, працездатність критично важливих мережних сервісів може бути порушена. Тому рекомендації H.323 передбачають керування смугою пропущення. Можна обмежити як число одночасних з'єднань, так і сумарну смугу пропущення для всіх додатків H.323. Ці обмеження допомагають зберегти необхідні ресурси для роботи інших мережних додатків. Кожний термінал H.323 може управляти своєю смугою пропущення в конкретній сесії конференції.

Рекомендації H.323 пропонують засіб з'єднання учасників відеоконференції в різнорідних мережах (наприклад, IP і ISDN, IP і PSTN). H.323 не прив'язаний ні до яких технологічних рішень, пов'язаним з устаткуванням або програмним забезпеченням. Взаємодіючі між собою додатки можуть створюватися на основі різних платформ, з різними операційними системами. Рекомендації H.323 дозволяють організовувати конференцію із трьома або більше учасниками. Багатоточечні конференції можуть проводитися як з використанням центрального MCU (пристрою багатоточечної конференції), так і без нього. H.323 підтримує багатоадресну передачу в багатоточечній конференції, якщо мережа підтримує протокол керування груповою адресацією (такий, як IGMP). При багатоадресній передачі один пакет інформації відправляється всім необхідним адресатам без зайвого дублювання. Багатоадресна передача використовує смугу пропущення набагато більш ефективно, оскільки всім адресатам – учасникам списку розсилання відправляється рівно один потік.

H.323 установлює стандарти для кодування й декодування аудіо- і відеопотоків з метою забезпечення сумісності встаткування різних виробників. Разом з тим стандарт досить гнучкий. Існують вимоги, виконання яких обов'язково, і існують опціональні можливості, у випадку використання яких також необхідно строго додержуватися стандарту. Крім цього, виробник може включати в мультимедійні продукти й додатки додаткові можливості, якщо вони не суперечать обов'язковим і опціональним вимогам стандарту.

Учасники конференції хочуть спілкуватися один з одним, не піклуючись про питання сумісності між собою. Рекомендації H.323 підтримують з'ясування загальних можливостей устаткування кінцевих користувачів і встановлюють найкращі із загальних для учасників конференції протоколів кодування, виклику й керування. H.323 конференція може включати учасників, кінцеве встаткування яких має різні можливості. Наприклад, один з учасників може використовувати термінал як тільки з аудіо- можливостями, у той час як інші учасники конференції можуть мати можливості передачі/прийому також відео й даних.

Таблиця 3.1 – Зведена таблиця протоколів сімейства H.32x

Рекомендація	H.320	H.321	H.322	H.323 V1/V2	H.324
Рік прийняття	1990	1995	1995	1996/1998	1996
Мережа	Вузько-полосна ISDN	Широко-полосна ISDN, ATM LAN	Мережа з комутацією пакетів і гаранторованою якістю обслуговування (isoEthernet)	Мережа з комутацією пакетів і негарантованою якістю обслуговування (Ethernet)	Аналогові телефонні мережі загального призначення (PSTN або POTS)
Відео	H.261 H.263	H.261 H.263	H.261 H.263	H.261 H.263	H.261 H.263
Аудіо	G.711 G.722 G.728	G.711 G.722 G.728	G.711 G.722 G.728	G.711 G.722 G.728 G.723 G.729	G.723
Мульти-плексування	H.221	H.221	H.221	H.225.0	H.223
Керування	H.230 H.242	H.242	H.242 H.230	H.245	H.245
Підтримка багатоточечних конференцій	H.231 H.243	H.231 H.243	H.231 H.243	H.323	–
Обмін даними	T.120	T.120	T.120	T.120	T.120
Мережний інтерфейс	I.400	AAL I.363 AJM I.361 PHY I.400	I.400 & TCP/IP	TCP/IP	V.34 Модем

Таблиця 3.2 – Зведена таблиця кодеків сімейства H.323

Кодек	Тип кодеку	Швидкість кодування	Затримка при кодуванні
G.711	ІКМ	64 Кбіт/с	0,75 мс
G.726	АДІКМ	32 Кбіт/с	1 мс
G.728	LD – CELP	16 Кбіт/с	Від 3 до 5 мс
G.729	CS – ACELP	8 Кбіт/с	10 мс
G.726 a	CS – ACELP	8 Кбіт/с	10 мс
G.723.1	MP – MLQ	6,3 Кбіт/с	30 мс
G.723.1	ACELP	5,3 Кбіт/с	30 мс

Базова архітектура стандарту H.323

У число "об'єктів" H.323, як вони названі в стандарті, включаються термінали, мультимедіа шлюзи, пристрої керування багатоточечними конференціями й контролери зони (Gatekeeper).

Термінал (Terminal) – окінечний мультимедійний (голос, відео, дані) пристрій, призначений для участі в конференції.

Мультимедіа шлюз (Gateway) – пристрій, призначений для перетворення мультимедійної й керуючої інформації при сполученні різнорідних мереж.

Пристрій управління багатоточечними конференціями (Multipoint Control Unit – MCU) – призначено для організації конференцій за участю трьох і більше учасників

Контролер зони (Gatekeeper, Привратник, Конференц-менеджер) – пристрій, що рекомендується, але не обов'язковий, що забезпечує мережне керування й виконує роль віртуальної телефонної станції.

Термінали H.323

Під терміналом стандарт розуміє встаткування кінцевих точок мережі, що дозволяє користувачам спілкуватися один з одним у реальному часі.

Термінал H.323 може являти собою ПК або автономний пристрій, здатне виконувати мультимедіа-додаток. Він зобов'язаний забезпечувати звуковий

зв'язок і може додатково підтримувати передачу відео або даних. Внаслідок того, що основною функцією термінала H.323 є передача звуку, він відіграє ключову роль у наданні сервісу IP-телефонії. H.323-термінал повинен підтримувати протоколи: H.245 – узгодження параметрів з'єднання, Q.931 – для встановлення з'єднання й узгодження параметрів цього з'єднання, RAS (Registration/Admission/Status) – взаємодії з контролером зони (Gatekeeper), RTP/RTCP – для роботи з потоками аудіо й відео пакетів і сімейство протоколів H.450, а також містити в собі аудіокодек G.711 для стиску аудіопотока. Його додатковими компонентами можуть бути інші аудіокодеки й відеокодеки H.261 і/або H.263. Необов'язковою є підтримка протоколу спільної роботи над документами T.120.

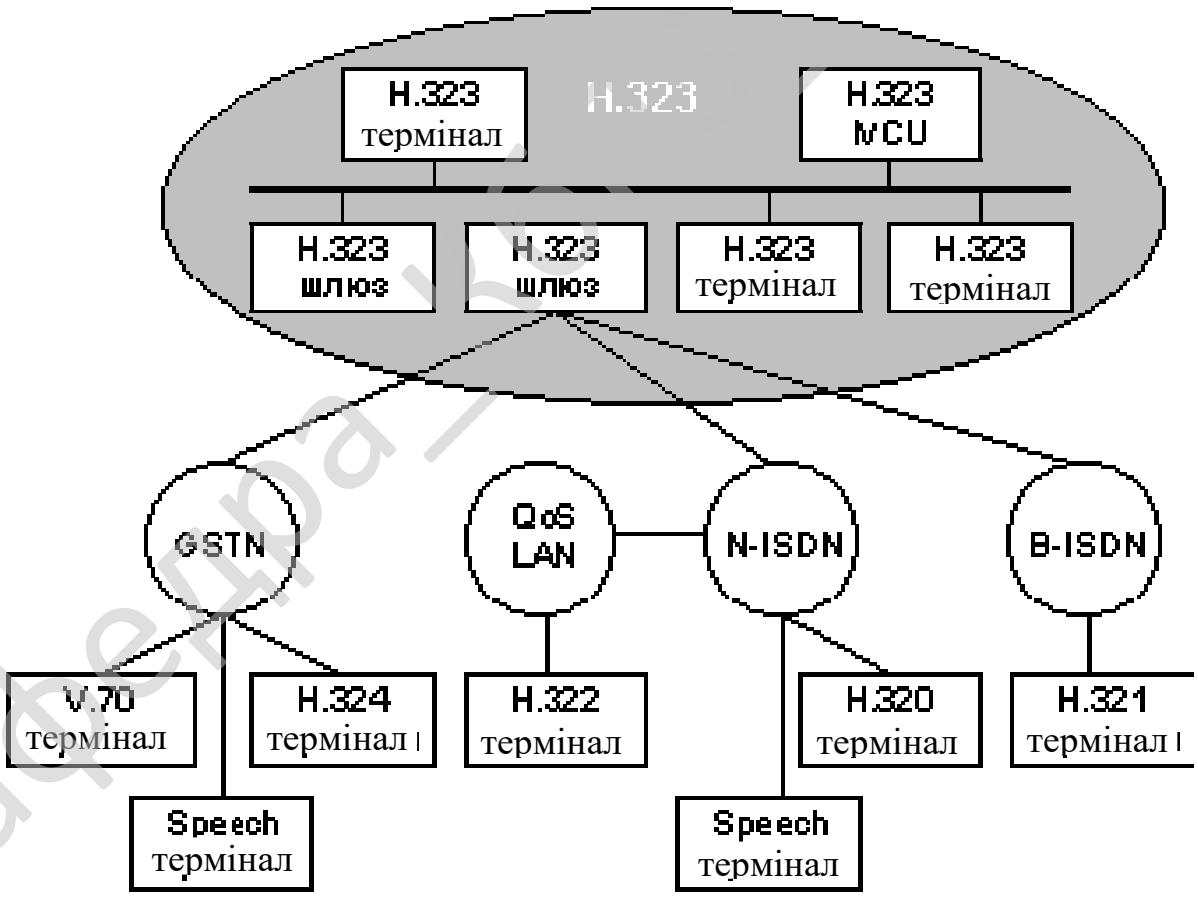

Рисунок 3.1 – Базова архітектура стандарту H.323

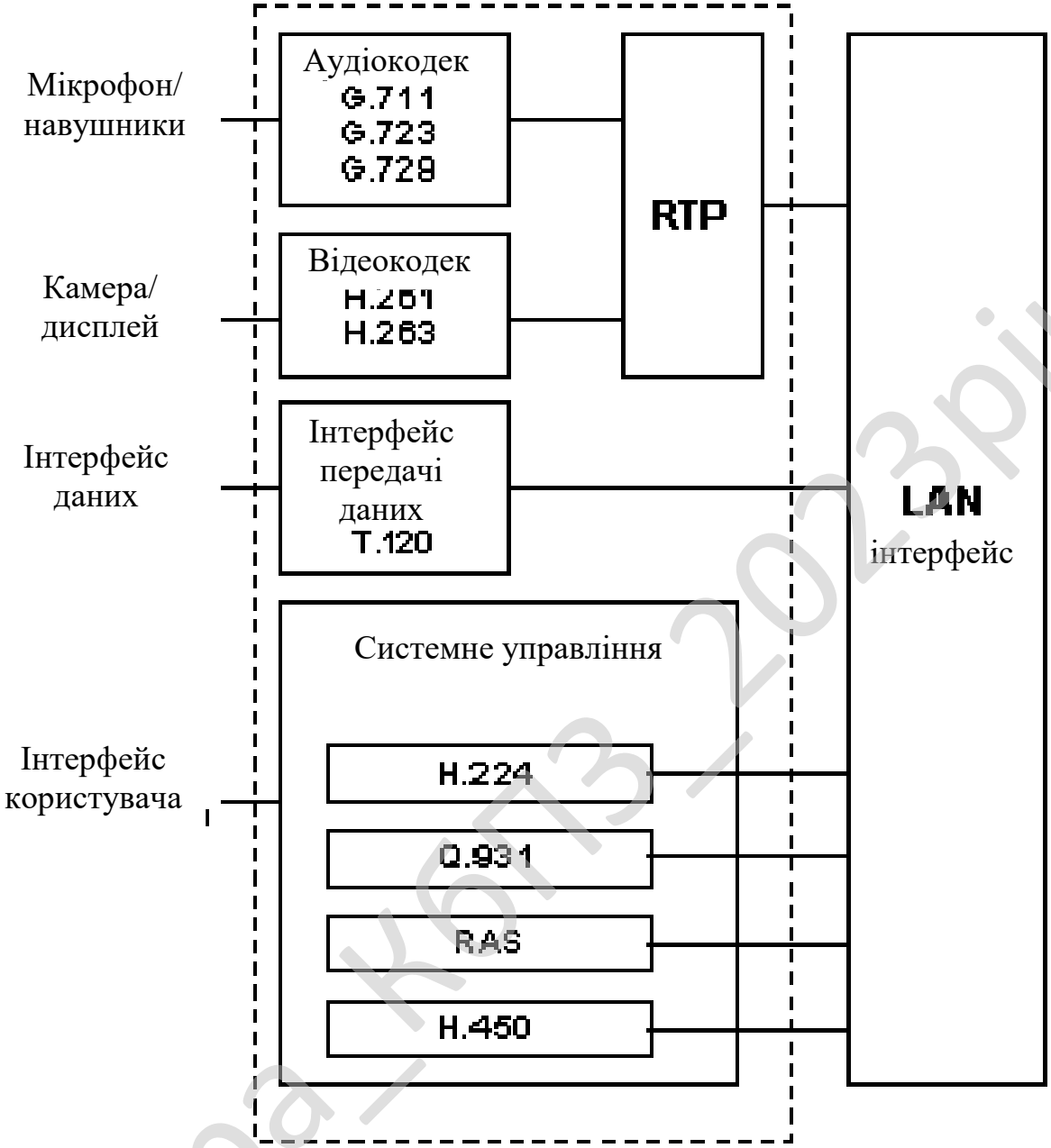

Рисунок 3.2 – Структура термінала H.323.

Аудіокодек призначений для оцифровки аналогового звукового сигналу й стиску отриманого цифрового сигналу, а також проведення зворотної операції. Стандартом H.323 передбачена можливість використання п'яти кодеків – G.711 (перетворення 3,1 кГц аналогові сигнали для передачі в цифровій формі на швидкостях 48, 56 або 64 Кбіт/с), G.722 (7 кГц; 48, 56 або 64 Кбіт/с), G.723 (3,1

кГц; 5,3 або 6,3 Кбіт/с), G.728 (3,1 кГц; 16 Кбіт/с) і G.729 (3,1 кГц; 8 Кбіт/с). Кожний термінал повинен підтримувати принаймні один аудіокодек.

Протокол сигналізації RAS (реєстрації, підтвердження й стани) застосовується для передачі службових повідомлень між терміналами й контролером зони. RAS-повідомлення служать для реєстрації терміналів, допуску їх до сеансу зв'язку, зміни використовуваної смуги пропущення, інформування про стан сеансу і його припиненні. У відсутності контролера зони протокол RAS не задіється.

Протокол сигналізації Q.931 використовується для встановлення й розриву з'єднань між двома терміналами H.323, а також між терміналом і шлюзом. Службові повідомлення цього протоколу передаються поверх TCP.

Протокол керування мультимедійною конференцією H.245 забезпечує:

– узгодження можливостей компонентів;

– установлення й розрив логічних каналів;

– передачу запитів на встановлення пріоритету;

– керування потоком (завантаженням каналу);

– передачу загальних команд і індикаторів.

Повідомлення протоколу H.245 передаються по H.245-каналу керування. Це логічний канал «0», що, на відміну від каналів обміну мультимедіа-потоками, постійно відкритий. Міжтермінальний обмін параметрами дозволяє погоджувати режими роботи й формати кодування інформації, що забезпечує взаємодію терміналів від різних виробників. У процесі обміну повідомленнями про параметри уточнюються можливості терміналів приймати й передавати різні види трафіку.

Протокол RTP (RFC 1889) забезпечує в IP-мережах доставку адресатам аудіо- і відеопотоків у масштабі реального часу. Відповідно до стандарту H.323, у мережах з негарантованою смугою пропущення з метою мінімізації затримок і максимального використання наявної смуги пропущення для передачі аудіо- і відеопотоків, а також сигналізації RAS застосовується протокол User Datagram

Protocol (UDP). Цей протокол задіє механізм багатоадресної розсилання (IP Multicast) для негарантованої доставки звуку й відео певному числу користувачів. Поверх IP Multicast працює RTP, що створює необхідні умови для нормального відтворення отриманих потоків на абонентських терміналах.

RTP ідентифікує тип і номер пакета, установлює в нього мітку синхронізації. На основі цієї інформації прийомний термінал синхронізує звук, відео й дані, здійснює їх послідовне й безперервне відтворення. Коректне функціонування RTP можливо при наявності в абонентських терміналах механізмів буферизації прийнятої інформації.

Транспортний протокол керування передачею в режимі реального часу RTCP (RFC 1889) контролює реалізацію функцій RTP. Він також відслідковує якість обслуговування й постачає відповідною інформацією компонента, що беруть участь у конференції.

Додаткові послуги в мережах H.323 визначає сімейство рекомендацій H.450. Так, 450.1 описує протокол сигналізації між двома компонентами мережі, що дозволяє надавати додаткові послуги, а H.450.2 – механізми послуги трансформації виклику (Call Transfer), завдяки якій з'єднання між терміналами А и Б перетвориться в з'єднання між Б и В. Додаткова послуга Call Diversion, що визначає рекомендація H.450.3, надає можливість переадресувати виклик у тих випадках, коли викликуваний абонент зайнятий, не відповідає або коли попередньо встановлений відповідний параметр.

Відеоможливості терміналів H.323

Незважаючи на те, що стандарт вважає функції відео необов'язковими, всі термінали з відеоможливостями повинні підтримувати кодек H.261, опціонально можлива підтримка H.263. H.263 є розвитком кодека H.261, відеокартинка, отримана за допомогою кодека H.263 має кращу якість, оскільки використовується полупіксельна технологія пророкування руху. Крім того, використовуване кодування по Хаффману оптимізовано для роботи з більше низькими швидкостями передачі. Визначено п'ять стандартних форматів кадрів.

Таблиця 3.3 – Формати кадрів H.261 і H.263

Формат кадру	Розмір у пікселях	H.261	H.263
sub-QCIF	128x96	не визначено	обов'язково
QCIF	176x44	обов'язково	обов'язково
CIF	352x288	можливо	можливо
4CIF	702x576	не визначено	можливо
16CIF	1408x1152	не визначено	можливо

Мультимедіа шлюз (Gateway) H.323

Шлюз не входить до числа обов'язкових компонентів мережі H.323. Він необхідний тільки в тому випадку, коли потрібно встановити з'єднання з терміналом іншого стандарту. Цей зв'язок забезпечується трансляцією протоколів установки й розриву з'єднань, а також форматів передачі даних. Згідно H.323, мультимедіа шлюз – це опціональний елемент у конференції H.323. Він може виконувати багато різних функцій. Типовою його функцією є завдання перетворення форматів протоколів передачі (наприклад, H.225.0 і H.221). Шлюзи H.323 широко застосовуються в IP-телефонії для сполучення IP-мереж і цифрових або аналогових телефонних мереж, що комутируються, (ISDN або PSTN).

Контролер зони (Gatekeeper, Воротар, Конференц-менеджер)

Це рекомендований, але не обов'язковий пристрій, що забезпечує мережне керування й виконує роль віртуальної телефонної станції. Контролер зони (gatekeeper) – найважливіший компонент H.323-мережі й центральна точка для всіх обігів усередині однієї зони. H.323-зона – це сукупність терміналів, шлюзів і серверів MCU, керованих одним контролером. У зоні присутня принаймні один термінал, а крім того, вона може включати сегменти ЛОМ, об'єднані маршрутизаторами. Контролер зони – необов'язковий компонент мережі H.323, однак якщо він присутній у мережі, то термінали й шлюзи повинні

використовувати його послуги. Відзначимо, що він може бути виконаний як частина шлюзу або сервера MCU.

Стандарт H.323 визначає основні (обов'язкові) і додаткові функції контролера зони. До першої групи відносяться трансляція адрес, контроль за встановленням з'єднань між терміналами, а також останніх зі шлюзами й серверами MCU, керування смугою пропущення й ін. У другу групу входить, зокрема, така важлива функція, як маршрутизація викликів. Вона дозволяє підвищити ефективність роботи мережі, оскільки контролер здатний вибирати маршрут з'єднання на основі, наприклад, даних про завантаження шлюзів своєї зони. Ця функція може служити й для переадресації виклику при відсутності можливості встановити з'єднання з викликуваним абонентом.

Пристрій керування багатоточечної конференцією (Multipoint Control Units (MCU))

Пристрій MCU призначений для підтримки конференції між трьома й більше учасниками. У цьому пристрої повинен бути присутнім контролер Multipoint Controller (MC), і, можливо, процесори Multipoint Processors (MP). Контролер MC підтримує протокол H.245 і призначений для узгодження параметрів обробки аудіо- і відеопотоків між терміналами. Процесори займаються комутуванням, мікшуванням і обробкою цих потоків.

Конфігурація багатоточечної конференції може бути централізованою, децентралізованою, гібридною й змішаною.

Централізована багатоточечна конференція вимагає наявності пристрою MCU. Кожний термінал обмінюється з MCU потоками аудіо, відео, даними й командами керування за схемою "точка-точка". Контролер MC, використовуючи протокол H.245, визначає можливості кожного термінала. Процесор MP формує необхідні для кожного термінала мультимедійні потоки й розсилає їх. Крім того, процесор може забезпечувати перетворення потоків від різних кодеків з різними швидкостями даних.

Таблиця 3.4 – Функції контролера зони

Функції	Опис
Основні	
Трансляція адрес	Перетворення внутрішніх адрес ЛОМ і телефонних номерів формату E.164 (застосовуються в мережах ISDN) у транспортні адреси протоколів IP або IPX
Керування доступом	Авторизація доступу в H.323-мережі шляхом обміну RAS-повідомленнями «запит реєстрації» (ARQ), «задоволення запиту» (ACF) і «відхилення запиту» (ARJ). Наприклад, якщо мережний адміністратор установив ліміт числа одночасних з'єднань, то при досягненні цього порога контролер зони буде відхиляти нові запити на доступ. Параметру даної функції може бути привласнене значення «0», що означає допуск всіх кінцевих точок в H.323-мережа
Керування смугою пропущення	Використовуються RAS-повідомлення «запит ширини смуги пропущення» (BRQ), «задоволення запиту» (BCF) і «відхилення запиту» (BRJ). Параметру даної функції може бути привласнене значення «0», що означає автоматичне задоволення всіх запитів на зміну смуги пропущення
Додаткові	
Керування процесом установлення з'єднань	При двосторонній конференції контролер здатний обробляти службові повідомлення протоколу сигналізації Q.931. Контролер може служити й простим ретранслятором таких повідомлень від кінцевих точок
Авторизація з'єднання	У відповідності зі специфікаціями Q.931 допускається відхилення контролером запиту на встановлення з'єднання. Серед підстав – обмеження прав або часу доступу, а також інші критерії, що перебувають поза рамками стандарту H.323
Керування викликами	Контролер зони може відслідковувати стан всіх активних з'єднань, що дозволяє управляти викликами, забезпечуючи виділення необхідної смуги пропущення й баланс завантаження мережних ресурсів за рахунок переадресації викликів на інші термінали й шлюзи

Децентралізована багатоточечна конференція використовує технологію групової адресації. Ті термінали, які беруть участь в конференції H.323 здійснюють багатоадресну передачу мультимедіа потоку іншим учасникам без посилки на MCU. Передача контрольної й керуючої інформації здійснюється за схемою "точка-точка" між терміналами й MCU. У цьому випадку контроль багатоточечного розсилання здійснюється контролером MC.

Гібридна схема організації конференцзв'язку є комбінацією двох попередніх. Термінали, що беруть участь в конференції H.323 здійснюють багатоадресну передачу тільки аудіо- або тільки відеопотоку іншим учасникам без посилки на MCU. Передача інших потоків здійснюється за схемою "точка-точка" між терміналами й MCU. У цьому випадку задіються як контролер, так і процесор MCU.

У змішаній схемі організації конференцзв'язку одна група терміналів може працювати за централізованою схемою, а інша група – по децентралізованій.

Принципи протоколу SIP

Протокол ініціювання сеансів – Session Initiation Protocol (SIP)є протоколом прикладного рівня й призначається для організації, модифікації й завершення сеансів зв'язку: мультимедійних конференцій, телефонних з'єднань і розподілу мультимедійної інформації. Користувачі можуть брати участь в існуючих сеансах зв'язку, запрошувати інших користувачів і бути запрошеними ними до нового сеансу зв'язку. Запрошення можуть бути адресовані певному користувачеві, групі користувачів або всіх користувачів.

Протокол SIP розроблений групою MMUSIC (Multiparty Multimedia Session Control) комітету IETF (Internet Engineering Task Force),а специфікації протоколу представлені в документі RFC 2543. В основу протоколу робоча група MMUSIC заклала наступні принципи:

– Персональна мобільність користувачів. Користувачі можуть переміщатися без обмежень у межах мережі, тому послуги зв'язку повинні

надаватися їм у будь-якому місці цієї мережі. Користувачеві привласнюється унікальний ідентифікатор, а мережа надає йому послуги зв'язку поза залежністю від того, де він перебуває. Для цього користувач за допомогою спеціального повідомлення – REGISTER – інформує про свої переміщення сервер визначення місця розташування.

– Масштабованість мережі. Вона характеризується, у першу чергу, можливістю збільшення кількості елементів мережі при її розширенні. Серверна структура мережі, побудованої на базі протоколу SIP, повною мірою відповідає цій вимозі.

– Розширюваність протоколу. Вона характеризується можливістю доповнення протоколу новими функціями при введенні нових послуг і його адаптації до роботи з різними додатками.

Як приклад можна привести ситуацію, коли протокол SIP використовується для встановлення з'єднання між шлюзами, взаємодіючими із ТфОП за допомогою сигналізації ОКС7 або DSS1. У цей час SIP не підтримує прозору передачу сигнальної інформації телефонних систем сигналізації. Внаслідок цього додаткові послуги ISDN виявляються недоступними для користувачів IP мереж.

Розширення функцій протоколу SIP може бути зроблене за рахунок введення нових заголовків повідомлень, які повинні бути зареєстровані у вже згадуваній раніше організації IANA. При цьому, якщо SIP сервер приймає повідомлення з невідомими йому полями, то він просто ігнорує їх і обробляє лише ті поля, які він знає.

Для розширення можливостей протоколу SIP можуть бути також додані й нові типи повідомлень.

Інтеграція в стек існуючих протоколів Інтернет, розроблених IETF. Протокол SIP є частиною глобальної архітектури мультимедіа, розробленої комітетом Internet Engineering TaskForce (IETF). Ця архітектура містить у собі також протокол резервування ресурсів (Resource Reservation Protocol – RSVP,

RFC 2205), транспортний протокол реального часу (Real Time Transport Protocol – RTP, RFC 1889), протокол передачі потокової інформації в реальному часі (Real Time Streaming Protocol – RTSP, RFC 2326), протокол опису параметрів зв'язку (Session Description Protocol – SDP, RFC 2327). Однак функції протоколу SIP не залежать від жодного із цих протоколів.

Взаємодія з іншими протоколами сигналізації. Протокол SIP може бути використаний разом із протоколом H.323. Можливо також взаємодія протоколу SIP із системами сигналізації ТфОП – DSS1 і ОКС7. Для спрощення такої взаємодії сигнальні повідомлення протоколу SIP можуть переносити не тільки специфічну SIP адресу, але й телефонний номер формату E.164 або будь-якого іншого формату. Крім того, протокол SIP, нарівні із протоколами H.323 і ISUP/IP, може застосовуватися для синхронізації роботи пристроїв керування шлюзами; у цьому випадку він повинен взаємодіяти із протоколом MGCP. Іншою важливою особливістю протоколу SIP є те, що він пристосований до організації доступу користувачів мереж IP телефонії до послуг інтелектуальних мереж, і існує думка, що саме цей протокол стане основним при організації зв'язку між зазначеними мережами.

3.2 Розробка структурної схеми

Розглянемо структурну схему системи. Почнемо з розгляду пристроїв складових систему IP-телефонії спеціального зв'язку.

Гібридні мережні пристрої. Як правило, використовують маршрутизатори, які містять як порти для передачі даних (Ethernet), так і голосові порти (FXS або FXO) – фактично вони є гібридом маршрутизатора й адаптера або шлюзу. Гібридний мережний пристрій може застосовуватися для створення корпоративної IP-телефонної мережі, з'єднуючи офісні УПАТМ через мережу Інтернет.

При такій схемі державній організації прийдеться мати втрати на придбання шлюзів, а також на створення служб експлуатації. При цьому звичайний провайдер послуг мережі Інтернет не зможе гарантувати надання необхідної смуги пропущення, тому що по каналу передаються й дані, і мова. Щоб уникнути подібних проблем доцільніше скористатися послугами ITSP-провайдеру. Провайдери IP-телефонії гарантують не тільки доступ до шлюзового встаткування, але й необхідну смугу пропущення.

ITSP надають можливості для викликів наступних типів сценаріїв:

– "Від телефону до телефону". Виклик іде зі звичайного телефонного апарата до комплекту встаткування ITSP і через мережу Інтернет доходить до іншого ITSP, що здійснює зворотні перетворення.

– "Від телефону до комп'ютера". Для державної організації переважніше встановити на кожне автоматизоване робоче місце локальної мережі програмне забезпечення IP-телефонії й мікрофон (більшість сучасних ПК уже мають звукові адаптери й акустичні системи) і підключити локальну мережу до ITSP.

– "Від комп'ютера до телефону". Шлюз в IP-системі виконує значне функціональне навантаження. Шлюз призначений для перетворення аналогових мовних і службових сигналів у цифрову послідовність, організації із цієї послідовності пакетів глобальної мережі Інтернет і передачі її в мережу, прийому пакетів і відновлення кодової послідовності цифрових, мовних і службових сигналів і їхнього перетворення в аналогову форму, а також рішення інших завдань, пов'язаних з організацією інтерфейсів, генеруванням і детектуванням сигналів абонентської сигналізації, керуванням режимами телефонних переговорів і деяких інших.

Шлюзи можуть установлюватися на серверах інтернет-провайдерів, міських телефонних станціях, АТМ для установ, серверах локальних обчислювальних мереж, веб-серверах компаній, що потребують в організації голосових гарячих ліній, служб технічної підтримки, діалогових довідкових служб і т.д. Також шлюзи можуть входити до складу маршрутизаторів.

Залежно від схеми організації зв'язку архітектура шлюзу може мінятися, можуть додаватися й інтегруватися деякі функції, виконувані шлюзом, додаватися або мінятися інтерфейси. Однак головні завдання шлюзу – забезпечення якісного дуплексного телефонного спілкування абонентів у режимі пакетної передачі й комутації цифрових сигналів – зберігаються поза залежністю від варіанта.

Варто помітити, що розглянуті вище базові схеми можуть комбінуватися. Можливі різноманітні способи організації IP-телефонного зв'язку з використанням шлюзів, розміщених у функціонально різних точках мережі. Однак у кожному разі шлюз у кожному з варіантів з'єднання буде ключовим елементом.

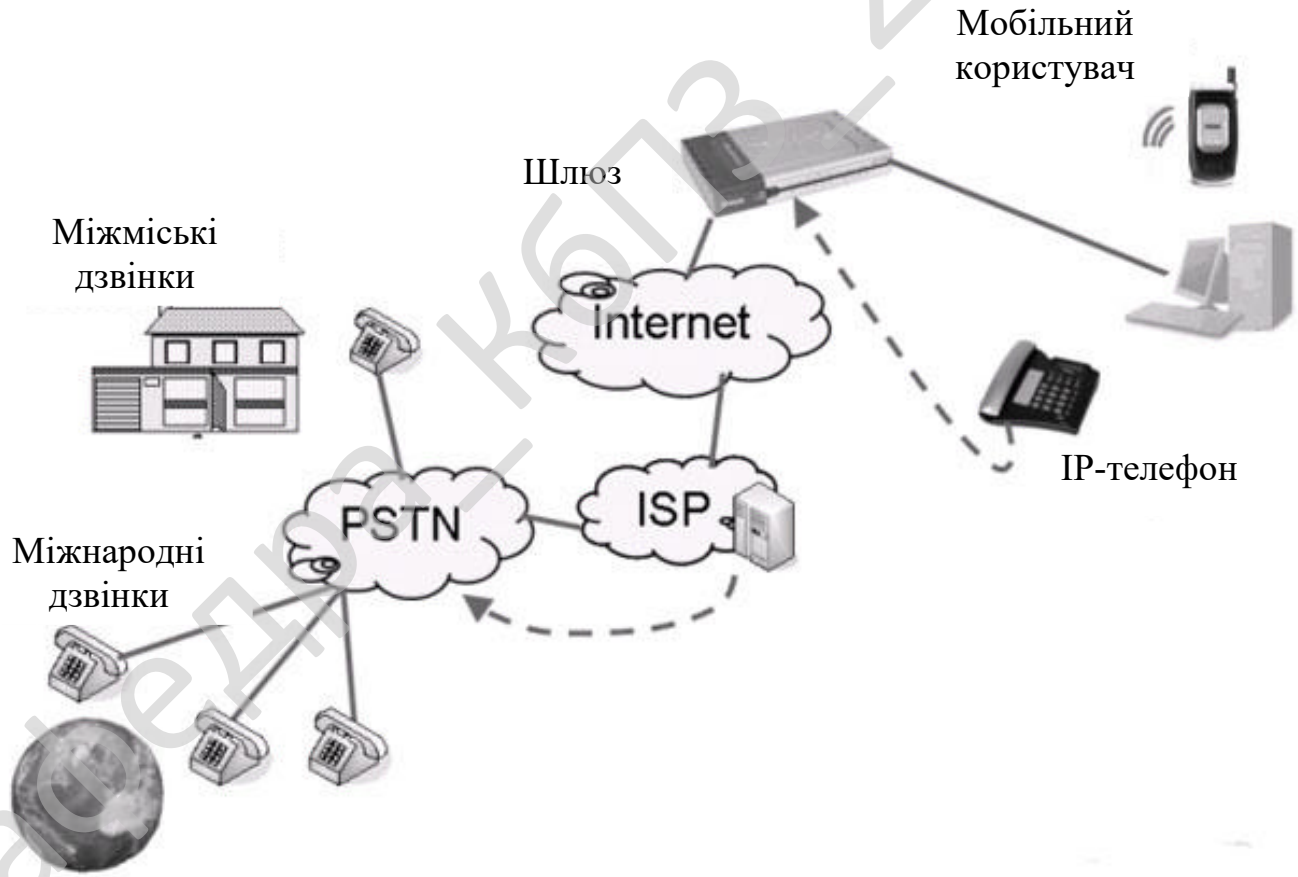

Рисунок 3.3 – Структурна схема системи

У схемі використовуються наступні позначення:

– PSTN – Телефонна мережа загального користування, ТМЗК, ТфОП (Public Switched Telephone Network) – це мережа, для доступу до якої використовуються звичайні провідні телефонні апарати, міні-АТМ і встаткування передачі даних. PSTN – секція телефонної інфраструктури, що веде від Class-5 офісів і здійснювана IXC (interexchange carriers). В PSTN передача сигналів (у тому числі й настроювання з'єднання) і сама розмова здійснюється через ту саму універсальну лінію зв'язку (магістраль) від системи комутації (СК) джерела до СК адресата. Цей процес займає канали зв'язку всіх задіяних при з'єднанні СК. Тобто, якщо викликуваний адресат зайнятий, всі ці з'єднання виявляться даремними. Звичайно PSTN використовують зірковидну топологію (головний елемент з'єднаний з безліччю другорядних). Але це не єдиний метод. Приміром, CATV використовують деревоподібну топологію.

– ISP – Інтернет-провайдер, іноді просто Провайдер, (Internet Service Provider, ISP, буквально "постачальник Інтернет-послуги") – організація, що надає послуги доступу до Інтернету й інші пов'язані з Інтернетом послуги.

Крім терміналів користувачів, які підключаються до мереж аналогової телефонії, існують також і сервери керування. До серверів керування відносяться:

– SIP-сервер (він же SIP-Proxy). Веде список підключених до нього й зареєстрованих клієнтів для того, щоб брати участь у пошуку абонента для з'єднання й керування сеансом з'єднання.

– Проксі для вихідних з'єднань (outbound proxy). Потрібний для обходу клієнтом обмежень, що накладаються використанням трансляції адрес (NAT).

– STUN-сервер. Спеціальний сервер, що дозволяє клієнтові визначити використовуваний тип трансляції адрес для того, щоб спробувати обійти його обмеження без використання проксі для вихідних з'єднань.

– DNS (Domain Name Service). Відомий протокол, що в SIP застосовується для знаходження SIP-сервера для заданого домену шляхом публікації спеціальних DNS-записів у зоні цього домену.

Окремим типом сервера є SIP-агент. Це програма, що безпосередньо займається прийомом і здійсненням дзвінків. Вона містить у собі кодеки й взаємодіє з кінцевим користувачем. SIP-агент убудований у кожний SIP-термінал – будь то програмний телефон, шлюз або апаратний VoIP-телефон. SIP-агенти взаємодіють між собою прямо (у випадку використання технології SIP Peer to Peer) або через SIP-сервери й проксі для вихідних з'єднань.

Помітимо, що сам протокол SIP голос не передає. Але він використовується для установки сеансу зв'язку й керування ім. Самі голосові дані передає протокол RTP. Причому працюють вони паралельно, але по різних портах: SIP координує сесію, а RTP передає голос.

Без використання RTP SIP не може "передавати" голос, а без SIP – RTP не зможе встановити сеанс зв'язку.

Типи погроз у мережах IP-телефонії

Існує кілька основних типів погроз, що представляють небезпеку в мережах IP-телефонії:

– Прослуховування. У момент передачі конфіденційної інформації про користувачів (ідентифікаторів, паролів) або конфіденційних даних по незахищених каналах існує можливість прослуховування й зловживання ними в корисливих цілях зловмисником.

– Маніпулювання даними. Дані, які передаються по каналах зв'язку, у принципі можна змінити.

– Підміна даних про користувача відбувається у випадку спроби видачі одного користувача мережі за інший. При цьому виникає ймовірність несанкціонованого доступу до важливих функцій системи.

– Відмова в обслуговуванні (denial of service – DoS) є одніє. з різновидів атак порушників, у результаті якої відбувається вивід з ладу деяких вузлів або всієї мережі. Вона здійснюється шляхом переповнення системи непотрібним трафіком, на обробку якого йдуть всі системні ресурси. Для запобігання даної

погрози необхідно використовувати засіб для розпізнавання подібних атак і обмеження їхнього впливу на мережу.

Базовими елементами в області безпеки є:

– автентифікація;

– цілісність;

– активна перевірка.

Застосування розширених засобів автентифікації допомагає зберегти в недоторканності вашу ідентифікаційну інформацію й дані. Такі засоби можуть ґрунтуватися на інформації, що користувач знає (пароль).

Цілісність інформації – це здатність засобу обчислювальної техніки або автоматизованої системи забезпечувати незмінність інформації в умовах випадкового й (або) навмисного перекручування (руйнування). Під погрозою порушення цілісності розуміється будь-яка навмисна зміна інформації, що зберігається в обчислювальній системі або передаваємій з однієї системи в іншу. Коли зловмисники навмисно змінюють інформацію, говориться, що цілісність інформації порушена. Цілісність також буде порушена, якщо до несанкціонованої зміни приводить випадкова помилка програмного або апаратного забезпечення.

І, нарешті, активна перевірка означає перевірку правильності реалізації елементів технології безпеки й допомагає виявляти несанкціоноване проникнення в мережу й атаки типу DoS. Активна перевірка даних діє як система раннього оповіщення про різні типи неполадок і, отже, дозволяє вжити попереджуючі заходи, поки не нанесений серйозний збиток.

Особливості системи безпеки в IP-телефонії

У системі IP-телефонії повинні забезпечуватися два рівні безпеки: системний і викличний.

Для забезпечення системної безпеки використовуються наступні функції:

– Запобігання неавторизованого доступу до мережі шляхом застосування поділюваного кодового слова. Кодове слово одночасно обчислюється по стандартних алгоритмах на ініціюючій і кінцевій системах, і отримані результати

порівнюються. При встановленні з'єднання кожна із двох систем IP-телефонії спочатку ідентифікує іншу систему; у випадку принаймні одного негативного результату зв'язок переривається.

– Списки доступу, у які вносяться всі відомі шлюзи IP-телефонії.

– Запис відмов у доступі.

– Функції безпеки інтерфейсу доступу, включаючи перевірку ідентифікатора й пароля користувача з обмеженням доступу по читанню/запису, перевірку прав доступу до спеціального WEB-серверу для адміністрування.

– Функції забезпечення безпеки виклику, включаючи перевірку ідентифікатора й пароля користувача (необов'язково), статус користувача, профіль абонента.

При встановленні зв'язку шлюзу з іншим шлюзом своєї зони виробляється необов'язкова перевірка ідентифікатора й пароля користувача. Користувач у будь-який час може бути позбавлений права доступу.

Дійсно, при розробці протоколу IP не приділялося належної уваги питанням інформаційної безпеки, однак згодом ситуація мінялася, і сучасні додатки, що базуються на IP, містять досить захисних механізмів. А рішення в області IP-телефонії не можуть існувати без реалізації стандартних технологій автентифікації й авторизації, контролю цілісності й шифрування й т.д. Для наочності розглянемо ці механізми в міру того, як вони задіються на різних стадіях організації телефонної розмови, починаючи з підняття слухавки й закінчуючи сигналом відбою:

– Телефонний апарат. В IP-телефонії, перш ніж телефон пошле сигнал на встановлення з'єднання, абонент повинен увести свій ідентифікатор і пароль на доступ до апарата і його функцій. Така автентифікація дозволяє блокувати будь-які дії сторонніх і не турбуватися, що чужі користувачі будуть дзвонити в інше місто або країну за ваш рахунок.

– Установлення з'єднання. Після набору номера сигнал на встановлення з'єднання надходить на відповідний сервер керування дзвінками, де здійснюється

цілий ряд перевірок з погляду безпеки. У першу чергу засвідчує дійсність самого телефону – як шляхом використання протоколу 802.1x, так і за допомогою сертифікатів на базі відкритих ключів, інтегрованих в інфраструктуру IP-телефонії. Така перевірка дозволяє ізолювати несанкціонованно встановлені в мережі IP-телефони, особливо в мережі з динамічною адресацією. Однак автентифікацією телефону справа не обмежується – необхідно з'ясувати, чи надано абонентові право дзвонити по набраному їм номеру. Це не стільки механізм захисту, скільки міра запобігання шахрайства. Якщо інженерові державній організації не можна користуватися міжміським зв'язком, то відповідне правило відразу записується в систему керування дзвінками, і з якого би телефону не здійснювалася така спроба, вона буде негайно припинена. Крім того, можна вказувати маски або діапазони телефонних номерів, на які має право дзвонити той або інший користувач. У випадку ж з IP-телефонією проблеми зі зв'язком, подібні до перевантажень ліній в аналоговій телефонії, неможливі: при грамотному проектуванні мережі з резервними з'єднаннями або дублюванням сервера керування дзвінками відмова елементів інфраструктури IP-телефонії або їхнє перевантаження не робить негативного впливу на функціонування мережі.

– Телефонна розмова. В IP-телефонії рішення проблеми захисту від прослуховування передбачалося із самого початку. Високий рівень конфіденційності телефонного зв'язку забезпечують перевірені алгоритми й протоколи (DES, 3DES, AES, IPSec і т.п.) при практично повній відсутності витрат на організацію такого захисту – всі необхідні механізми (шифрування, контролю цілісності, хешування, обміну ключами й ін.) уже реалізовані в інфраструктурних елементах, починаючи від IP-телефону й закінчуючи системою керування дзвінками. При цьому захист може з однаковим успіхом застосовуватися як для внутрішніх переговорів, так і для зовнішніх (в останньому випадку всі абоненти повинні користуватися IP-телефонами). Однак із шифруванням зв'язаний ряд моментів, про які необхідно пам'ятати, впроваджуючи інфраструктуру VoIP. По-перше, з'являється додаткова затримка

внаслідок шифрування/дешифрування, а по-друге, ростуть накладні витрати в результаті збільшення довжини переданих пакетів.

– Невидимий функціонал. Дотепер ми розглядали тільки ті небезпеки, яким піддана традиційна телефонія і які можуть бути усунуті впровадженням IP-телефонії. Але перехід на протокол IP несе із собою ряд нових погроз, які не можна не враховувати. На щастя, для захисту від цих погроз уже існують рішення, технології й підходи, які добре зарекомендували себе. Більшість із них не вимагає ніяких фінансових інвестицій, будучи вже реалізованими в мережному встаткуванні, що і лежить в основі будь-якої інфраструктури IP-телефонії. Найпростіше, що можна зробити для підвищення захищеності телефонних переговорів, коли вони передаються по тій же кабельній системі, що й звичайні дані, – це сегментувати мережу за допомогою технології VLAN для усунення можливості прослуховування переговорів звичайними користувачами. Гарні результати дає використання для сегментів IP-телефонії окремого адресного простору. І, звичайно ж, не варто скидати з рахунків правила контролю доступу на маршрутизаторах (Access Control List, ACL) або міжмережних екранах (firewall), застосування яких ускладнює зловмисникам завдання підключення до голосових сегментів.

– Спілкування із зовнішнім миром. Якої би переваги IP-телефонія не надавала в рамках внутрішньої мережі, вони будуть неповними без можливості здійснення й прийому дзвінків на міські номери. При цьому, як правило, виникає завдання конвертації IP-трафіка в сигнал, переданий по телефонній мережі загального користування (ТфОП). Вона вирішується за рахунок застосування спеціальних голосових шлюзів (voice gateway), що реалізують і деякі захисні функції, а сама головна з них – блокування всіх протоколів IP-телефонії (H.323, SIP і ін.), якщо їхнього повідомлення надходять із неголосового сегмента. Для захисту елементів голосової інфраструктури від можливих несанкціонованих впливів можуть застосовуватися спеціалізовані рішення – міжмережні екрани (MME), шлюзи прикладного рівня (Application Layer Gateway, ALG) і

прикордонні контролери сеансів (Session Border Controller). Зокрема, протокол RTP використовує динамічні порти UDP, відкриття яких на міжмережному екрані приводить до появи зяючої діри в захисті. Отже, міжмережний екран повинен динамічно визначати використовувані для зв'язку порти, відкривати їх у момент з'єднання й закривати по його завершенню. Інша особливість полягає в тому, що ряд протоколів, наприклад, SIP, інформацію про параметри з'єднання розміщає не в заголовку пакета, а в тілі даних. Тому пристрій захисту повинне бути здатне аналізувати не тільки заголовок, але й тіло даних пакета, отримуючи з нього всі необхідні для організації голосового з'єднання відомості. Ще одним обмеженням є складність спільного застосування динамічних портів і NAT.

3.3 Розробка функціональної схеми

На рисунку 3.4зображена функціональна схема розробленого програмного забезпечення.

З функціональної схеми ми бачимо, що основний функціональний модуль включає в себе наступні функціональні підблоки:

– IP-телефонія.

– Селективний зв'язок.

– Відеоконференція.

– Формування вхідної інформації.

– База даних абонентів мережі спеціального зв’язку.

– Біллінг.

– Виведення статусу ПЗ.

– Довідник.

Дані, які поступають з основного функціонального модуля шифруються алгоритмом шифрування ДСТ 28147:2009, та перетворюються згідно протоколу H.323, після чого передаються до IP-мережі.

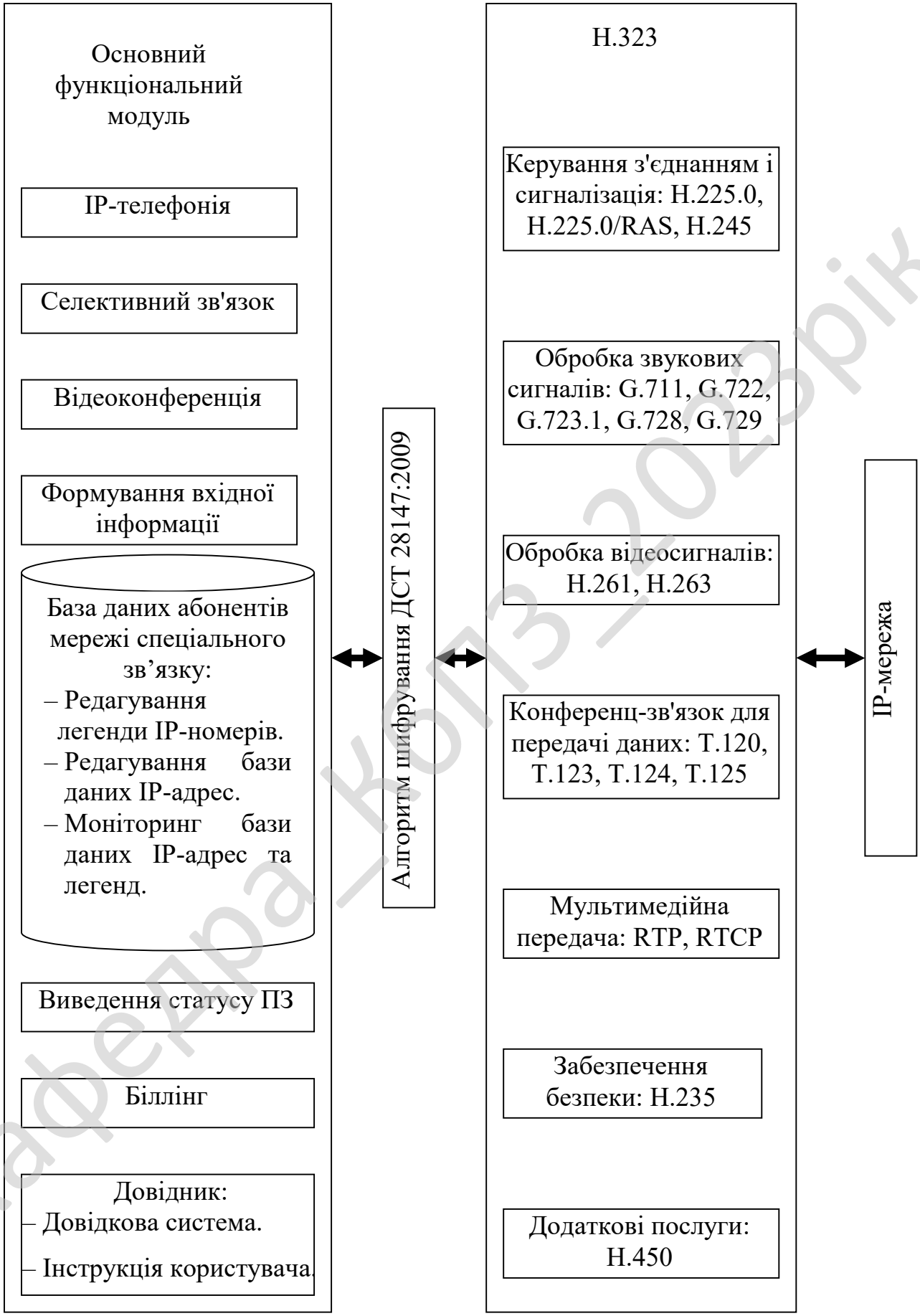

Рисунок 3.4– Функціональна схема розробленої системи

Робота з програмою починається з введення інформаційного вікна й активізації системи меню. Робота програми здійснюється по діалоговому і подійному режиму, при цьому по діалогом розуміється надання користувачу декількох альтернатив і обробка його вибору. У діалогову систему входять головне меню з відповідними спливаючими підменю а також діалогові вікна. Під подіями розуміються процеси, що активізуються користувачем (наприклад – натискання функціональних клавіш), а також програмні події – одержання з'єднання з абонентом або закінчення з'єднання з абонентом. На підставі даних подій активізуються процедури контролю допустимості даних.

Головне вікно програми призначене для запуску основних процедур програми і завершення роботи з програмою.

Модуль роботи з довідниками містить у собі два довідники:

– Довідник – Довідкова система.

– Довідник – Інструкція користувача.

Модуль роботи з базою даних абонентів мережі спеціального зв'язку включає в себе наступні підмодулі:

– Редагування легенди IP-номерів.

– Редагування бази даних IP-адрес.

– Моніторинг бази даних IP-адрес та легенд.

Призначення даного модуля є пошук і перегляд інформації з телефонних даних адрес абонентів корпорації, а також їх легенд.

Інформаційною базою даного модуля є таблиці: Значення IP-номерів та Легенда IP-номерів. Дані в інформаційну базу заносяться за допомогою спеціальних форм, що викликаються з головного меню програми.

Модуль «Формування вхідної інформації» призначений для введення первинних даних і перегляду раніше занесених. Даний модуль реалізує задачі обліку телефонних номерів, забезпечуючи введення номерів та їх легенд, ранжування за важливістю та їх знищення.

У комп'ютерних системах користувачі для введення, перегляду та редагування інформації бази даних (IP-адрес та відповідних легенд) можуть застосовувати форми. Основні переваги використання форм наступні:

– При введенні даних у поля-форми, додаток може зчитувати словник даних сервера й автоматично перевірити допустимість даних відповідно до правил цілісності.

– Поле введення у формі може представляти список допустимих значень, з яких користувачі можуть легко вибрати потрібне.

– Область форми може виводити шаблон, що відповідає поточної виведеної у формі запису.

– Командні кнопки у формі можуть виконувати дії, зв'язані з виведеної у формі поточною записом.

Біллінгова система – автоматизована система розрахунків з абонентами за надані послуги. Керування й статистика може бути доступна з будь-якої точки мережі через Веб-інтерфейс.

Для захисту даних у системі IP-телефонії запропоновано використовувати вітчизняний алгоритм ДСТ 28147:2009, що є класичним алгоритмом симетричного шифрування на основі мережі Фейстеля (рисунок 3.5).

Даний алгоритм шифрує інформацію блоками по 64 біта (такі алгоритми називаються "блоковими"). Зміст мережі Фейстеля полягає в тому, що блок шифруемої інформації розбивається на два або більше субблоків, частина яких обробляється за певним законом, після чого результат цієї обробки накладається (операцією побітового додавання за модулем 2) на необроблювані субблоки. Потім субблоки міняються місцями, після чого обробляються знову й т.д. певне для кожного алгоритму число раз – раундів.

Основна відмінність алгоритмів симетричного шифрування друг від друга складається саме в різних функціях обробки субблоків. Дана функція часто називається "основним криптографічним перетворенням", оскільки саме вона несе основне навантаження при шифруванні інформації. Основне перетворення

алгоритму ДСТ 28147:2009 є досить простим, що забезпечує високу швидкодію алгоритму; у ньому виконуються наступні операції (рисунок 3.6):

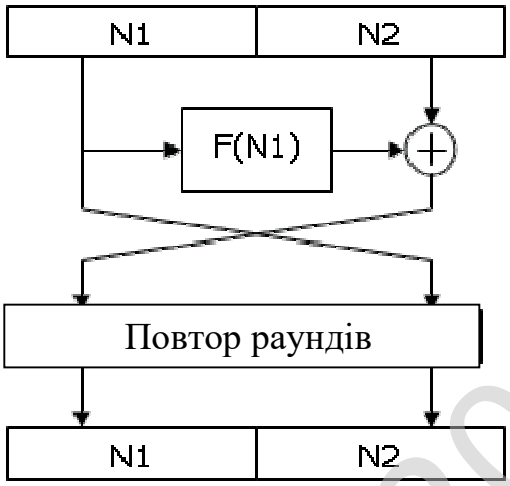

Рисунок 3.5– Мережа Фейстеля

Рисунок 3.6– Основне перетворення алгоритму ДСТ 28147:2009

1. Додавання субблоку з певним фрагменом ключа шифрування за модулем 2^{32}. Кх – це 32-бітна частина ("підключ") 256-бітного ключа шифрування, якому можна представити як конкатенацію 8 підключей: $К = К_0К_1К_2К_3К_4К_5К_6\ К_7$. Залежно від номера раунду й режиму роботи алгоритму (про їх – нижче), для даної операції вибирається один з підключей.

2. Таблична заміна. Для її виконання субблок розбивається на 8 4-бітних фрагментів, кожний з яких прогоняється через свою таблицю заміни. Таблиця заміни містить у певній послідовності значення від 0 до 15 (тобто всі варіанти значень 4-бітні фрагменти даних); на вхід таблиці подається блок даних, числове подання якого визначає номер вихідного значення. Наприклад, подається

значення 5 на вхід наступної таблиці: "13 0 11 74 91 10 143 5 122 15 8 6". У результаті на виході виходить значення 9 (оскільки 0 заміняється на 13, 1 – на 0, 2 – на 11 і т.д.).

3. Побітове циклічне зрушення даних усередині субблока на 11 біт уліво.

Стандарт H.323 визначає широкі вимоги для багатьох різних протоколів, які становлять повний стек протоколів H.323.

Стек H.323 складають 7 груп протоколів:

– керування й сигналізація;

– обробка звукових сигналів;

– обробка відеосигналів;

– конференц-зв'язок;

– передача мультимедійної інформації;

– забезпечення інформаційної безпеки;

– додаткові послуги;

1. Керування з'єднанням і сигналізація:

– H.225.0: протоколи сигналізації й пакетування мультимедійного потоку (використовує підмножину протоколу сигналізації Q.931).

– H.225.0/RAS: процедури реєстрації, допуску й стану.

– H.245: протокол керування для мультимедіа.

2. Обробка звукових сигналів:

– G.711: імпульсно-кодова модуляція тональних частот.

– G.722: кодування звукового сигналу 7 кГц в 64 кбіт/с.

– G.723.1: мовні кодери на дві швидкості передачі для організації мультимедійного зв'язку зі швидкістю передачі 5.3 і 6.3 кбіт/с.

– G.728: кодування мовних сигналів 16 кбіт/с за допомогою лінійного пророкування з кодуванням сигналу порушення з малою затримкою.

– G.729: кодування мовних сигналів 8 кбіт/с за допомогою лінійного пророкування з алгебраїчним кодуванням сигналу порушення сполученої структури.

3. Обробка відеосигналів:

– H.261: відеокодеки для аудіовізуальних послуг зі швидкістю 64 кбіт/с.

– H.263: кодування відеосигналу для передачі з малою швидкістю.

4. Конференц-зв'язок для передачі даних:

– T.120: це стек протоколів (який включає T.123, T.124, T.125) для передачі даних між окінцевими пунктами. Він може використовуватися для різних додатків в області спільної роботи (Collaboration Work), такий як колективне редагування растрових зображень, спільне використання додатків і спільна організація документів. В T.120 застосовується багаторівнева архітектура, подібна моделі OSI.

5. Мультимедійна передача:

– RTP: транспортний протокол реального часу.

– RTCP: протокол керування передачею в реальному часі.

6. Забезпечення безпеки:

– H.235: забезпечення безпеки й шифрування для мультимедійних терміналів мережі H.323.

7. Додаткові послуги:

– H.450.1: узагальнені функції для керування додатковими послугами в H.323.

– H.450.2: переклад з'єднання на телефонний номер третього абонента.

– H.450.3: переадресація виклику.

– H.450.4: утримання виклику.

– H.450.5: паркування виклику (park) і відповідь на виклик (pick up).

– H.450.6: повідомлення про виклик, що надійшов, у стані розмови.

– H.450.7: індикація повідомлення, що очікує.

– H.450.8: служба ідентифікації імен.

– H.450.9: служба завершення з'єднання для мереж H.323.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.7.

З діаграми, ми бачимо, що спершу запускається процес авторизації. Він взаємодіє з наступними процесами:

– Процес зміни параметрів системи.

– Процес підключення до мережі.

Останній процес взаємодіє з наступними процесами:

– Процес виведення списку підключених користувачів, та їх IP-адрес.

– Процес створення та обміну ключами шифрування.

– Процес виклику абоненту.

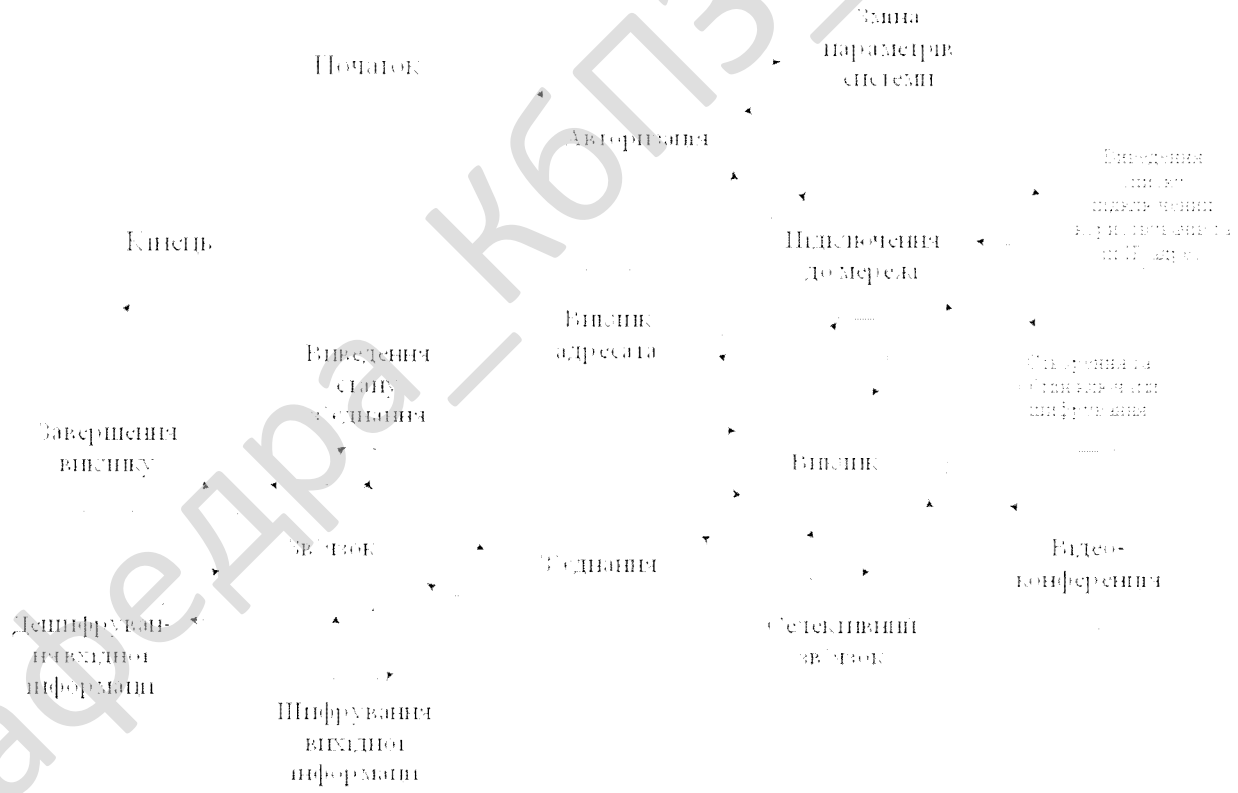

Рисунок 3.7 – Діаграма взаємодії процесів

Процес виклику абоненту взаємодіє з наступними процесами:

– Процес відеоконференції.

– Процес селективного зв'язку.

– Процес виклику адресата.

– Процес з’єднання.

Останній процес взаємодіє з процесом реалізації зв’язку.

Процесом реалізації зв’язку взаємодіє з наступними процесами:

– Процес шифрування вхідної інформації.

– Процес дешифрування вхідної інформації.

– Процес виведення стану з’єднання.

– Процес завершення виклику.

Процес завершення виклику є завершальним у системі.

Розглянувши діаграму взаємодії процесів, які відбуваються у системі, перейдемо до опису алгоритмів функціонування системи та їх блок-схем.

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

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається ініціалізація програмного забезпечення та підключення основних модулів програми. Наступним кроком є виведення вікна авторизації.

Після цього відбувається авторизація користувача, з наданням йому відповідних прав.

Якщо авторизація не пройшла успішно, тоді видається повідомлення про помилку, й знову виводиться вікно авторизації.

Якщо ж авторизація пройшла успішно, відбувається виконання наступних ітерацій:

– Виводиться головне вікно програми.

– Відбувається встановлення параметрів інтерфейсу користувача.

– Відбувається встановлення параметрів з’єднання та шифрування.

– Виводиться список користувачів, та їх IP-адрес.

Якщо є запит на обмін ключами то відбувається обмін ключами шифрування.

Якщо здійснюється виклик, то виконується наступна послідовність дій:

– Вибирається тип з’єднання, відкритий зв'язок, або захищений.

– Відбувається виклик.

– Запускається підпрограма зв’язку.

Після цього користувач обирає працювати йому далі з системою, або ні.

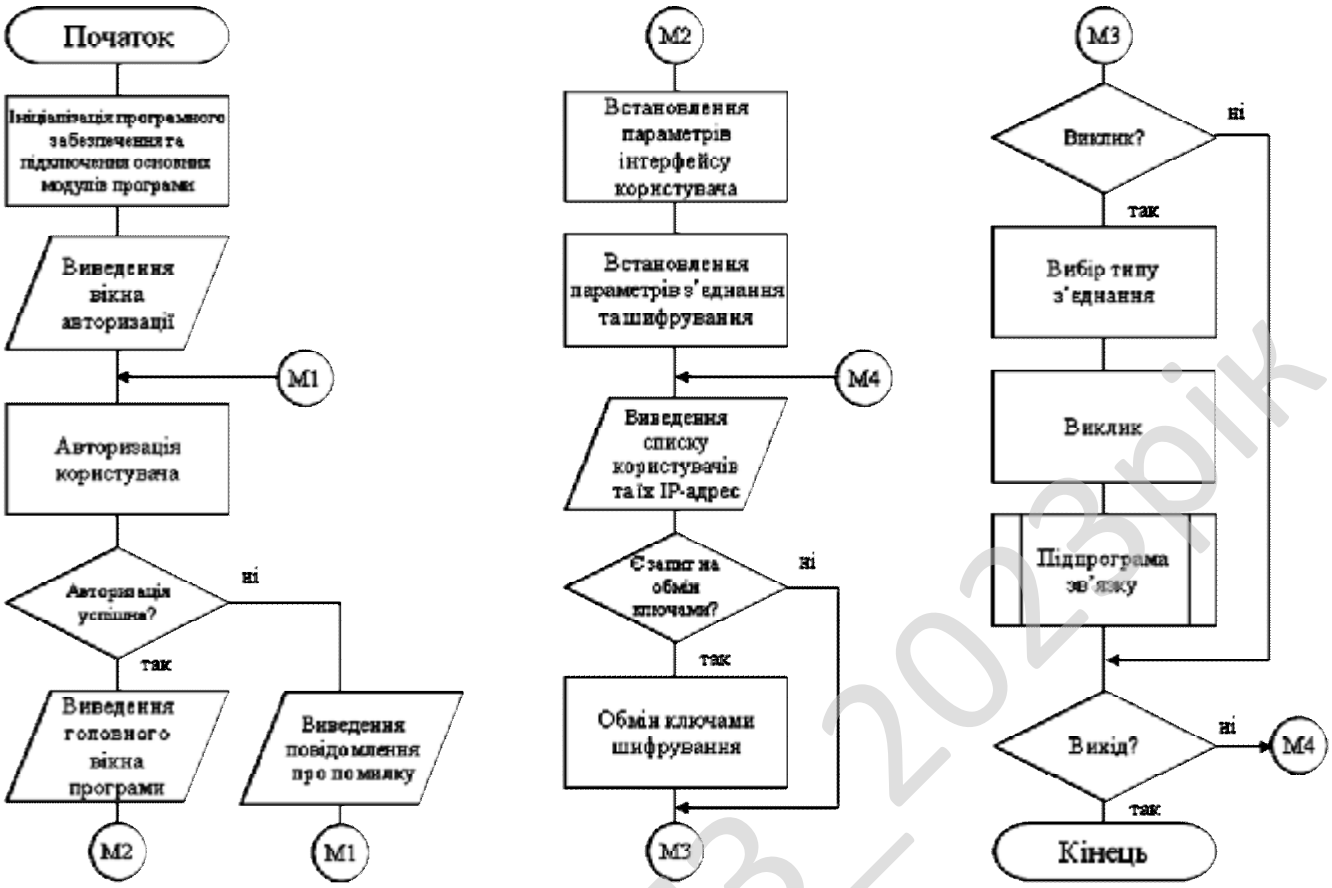

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 приведено блок-схему підпрограми зв'язку. З неї ми бачимо, що підпрограма працює наступним чином.

Відбувається спроба доступу до звукової карти.

Якщо спроба не є успішною, тоді виводиться повідомлення про помилку, й підпрограма завершує роботу.

Якщо ж спроба успішна, тоді виконуються наступні етапи роботи системи:

– Налаштовується звук.

– Відбувається спроба встановлення захищеного зв’язку.

Якщо спроба є успішною, тоді відбувається спроба підключення до віддаленого адресату.

Якщо зєднання встановлено, тоді виконуються наступні кроки:

– Забезпечується надійність сеансу зв’язку.

– Відбувається шифрування вихідної інформації алгоритмом ДСТ 28147:2009.

– Відбувається дешифрування вихідної інформації алгоритмом ДСТ 28147:2009.

– Виводиться стан з’єднання.

Після цього з’являється запит, чи потрібно змінити параметри з’єднання. Якщо потрібно, то параметри змінюються.

Якщо потрібно розірвати з’єднання, то програма закінчує свою роботу.

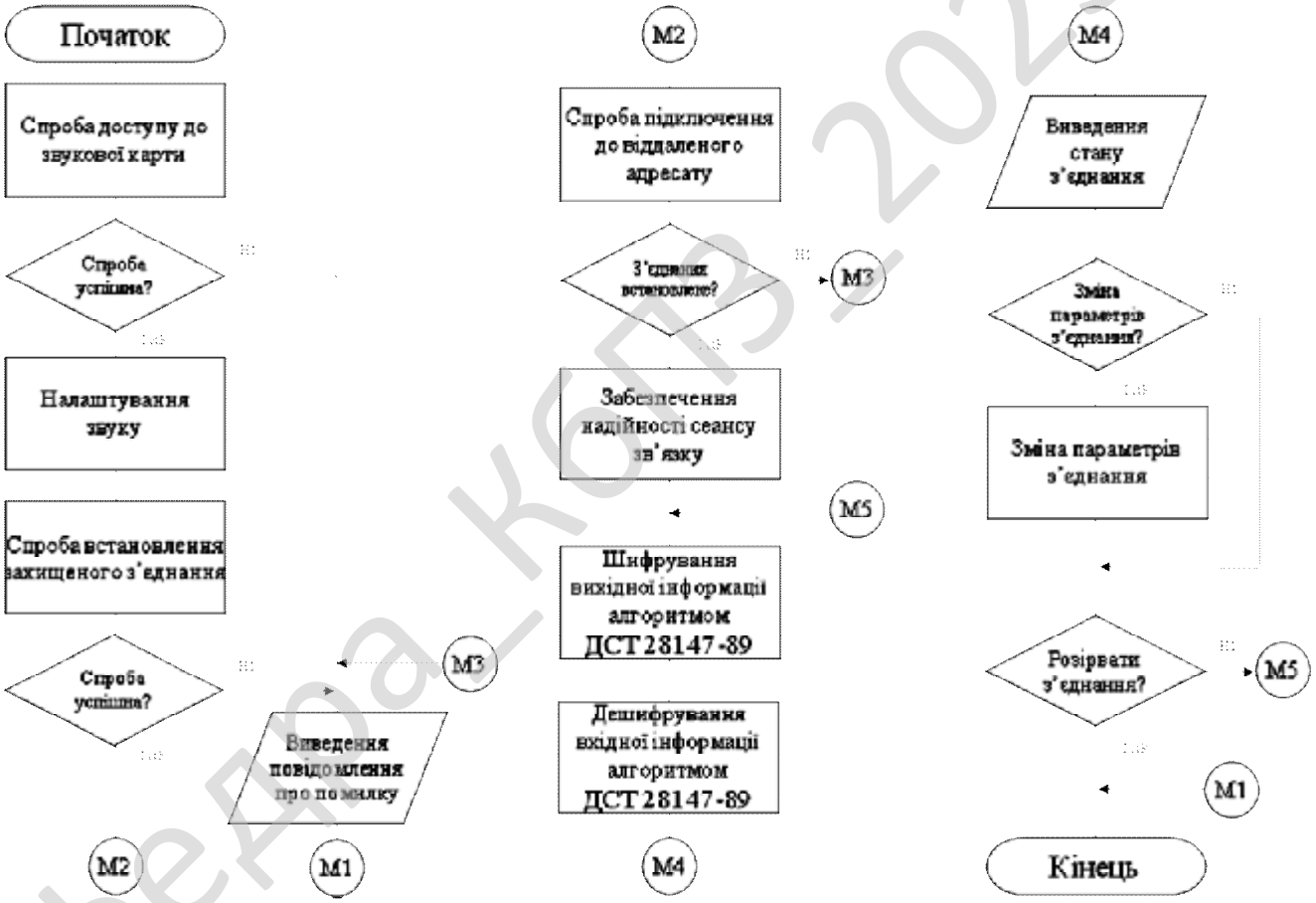

Рисунок 4.2 – Блок-схема підпрограми зв'язку

Опис математичного забезпечення

Для передачі мови по IP-каналам необхідно, як було відмічено раніше, перетворити аналоговий сигнал у цифровий, тобто безперервний в дискретний.

Виявляється, що навіть такі на перший погляд зовсім різні сигнали, як безперервні й дискретизовані мають дуже багато загального й зв'язані твердою функціональною залежністю, встановленою теоремою дискретизації, або теоремою Котельникова [3].

Реальний безперервний сигнал володіє спектром, основна частина енергії якого зосереджена в обмеженій смузі частот. Це зумовлене тим, що прилади, що формують і перетворюють повідомлення і сигнали, а також канали зв'язку мають кінцеву смугу пропускання. Функція часу з обмеженим по ширині спектром повністю визначається своїми миттєвими значеннями, відрахованими через інтервали часу:

$$\Delta t = 1/2F_m, \tag{4.1}$$

де F_m – найвища частота спектру сигналу.

Це положення складає зміст теореми Котельникова.

Теорема дискретизації, або, як її ще називають, теорема Котельникова, теорема Уїтекера, формулюється в такий спосіб: безперервна функція $X(t)$ з обмеженим спектром, тобто не має у своєму спектрі:

$$F\{X(t)\} = \int_{-\infty}^{\infty} X(t) \cdot e^{-j2\pi ft}\, dt, \tag{4.2}$$

складових із частотами, що лежать за межами смуги $f \in (-F_m, F_m)$, повністю визначається послідовністю своїх обчислень у дискретні моменти часу $X(t_i)$, що виходять із кроком $\Delta t < 1/F_m$. Іншими словами безперервна детермінована функція часу $X(t)$, що має обмежений спектр, може бути розкладена в ряд по ортогональним функціям часу виду:

$$\Psi_i(t) = \frac{\sin 2F_m \pi (t - i\Delta t)}{2\pi F_m (t - i\Delta t)}, \tag{4.3}$$

з коефіцієнтами, рівними значенням функції $X(i\Delta t)$. Цей розклад, що називається рядом Котельникова, має наступний вигляд: $X(t) = \sum_{i=-\infty}^{\infty} X(i\Delta t)\Psi_i(t)$.

Функція відрахунків має наступні властивості:

1. Вона дорівнює 1 при $t = i\Delta t$ і рівна 0 в усіх інших точках $t = j\Delta t; j \neq i$.

2. Її спектр рівномірний в межах смуги частот від $-F_{min}$ до F_{max} але рівний 0 поза цією смугою.

3.Вона має властивості ортогональності:

$$\int_{-\infty}^{\infty} \Psi_i(t)\Psi_j(t)\,dt = \begin{cases} 0 & \text{при } i \neq j \\ \dfrac{1}{2F_{max}} & \text{при } i = j \end{cases}.$$

Якщо функція f(t) з обмеженим спектром розглядається на кінцевому інтервалі часу T, то число дискрет, що передаються, буде дорівнювати: $T/\Delta t = 2F_m T$. Величину $F_m T$ називають базою сигналу.

Доказ сформульованої теореми ґрунтується на однозначній відповідності між сигналами й відповідними їм спектрами. Іншими словами, якщо сигнали однакові, те й відповідні їм спектри також однакові. І, навпаки, якщо спектри двох сигналів однакові, той і відповідний сигнали також однакові.

Опис технологічного процесу обробки даних

Технологічний процес складається з двох основних етапів – збір і облік даних по IP-адресам й формування відповідної легенди та власне сама робота програми у режимі IP-телефонії та селективного зв'язку. Вони можуть виконуватися в будь-який календарний момент часу і включають операції введення, з'єднання і ін. Операції мають програмне виконання, підлегле єдиній алгоритмічній схемі. Програма реалізована в середовищі Delphi. Робота з програмою починається з виведення інформаційного вікна й активізації системи меню й здійснюється в діалоговому і по-дійному режиму. При цьому під діалогом розуміється надання користувачу декількох альтернатив і обробка його вибору. У діалогову систему входять головне меню з відповідними спливаючими підменю а також діалогові вікна. Під подіями розуміються процеси, що активізуються користувачем (наприклад – натискання функціональних клавіш), а також програмні події – з'єднання по IP-технології.

Програма складається з наступних основних модулів.

Основна процедура – конфігурація середовища оточення, формування основного екрана програми, створення системи головного меню і відповідних підміню, активізація меню.

Процедура обробки головного меню – запуск відповідної процедури. Процедура введення даних – забезпечення введення інформації у бази даних IP-адрес та легенд до них, контроль за допустимістю значень, забезпечення введення даних шляхом вибору зі списку.

Допоміжні процедури і функції – реалізація запитів, повідомлень, формування списків вибору IP-адрес та легенд, а також контроль за даними, що вводяться.

Усі модулі в програмі зв'язані між собою за даними, що аналізуються на вході і виробляються на виході. Дані в модулі надходять через діалог з користувачем, параметри і документи інформаційної бази. Передача даних від одного модуля до іншого здійснюється тільки через збережені документи.

Дані через діалог можуть бути отримані прямим і непрямим способом. Прямий спосіб реалізується шляхом їхнього введення за шаблоном чи по запиту конкретних значень реквізитів. Непрямий спосіб – шляхом чи меню логічних (альтернативних) запитів – «так», «ні». При непрямому способі дані, що надходять у модуль, заздалегідь передбачені алгоритмом, але зовні виглядають в обліку відомими фразами.

Параметри (мова) – вхідні дані, отримані у виді конкретних значень, переданих в оперативній пам'яті суміжним модулям (функціям).

Обґрунтування інформаційного забезпечення

Перелік первісних даних

Під вхідною інформацією розуміється вся інформація, необхідна для вирішення задачі і розташована на різних носіях: первинних документах, машинних носіях, у пам'яті персонального комп'ютера. Вхідною інформацією для

розроблювальної в бакалаврському проекті корпоративної системи зв’язку з використанням IP-технологій є мова або відеозображення.

Щоб передати мову через телефонну мережу, мовну інформацію потрібно перетворити в аналоговий електричний сигнал. При переході до цифрових мереж зв'язку виникла необхідність перетворити аналоговий електричний сигнал у цифровий формат на передавальній стороні, тобто закодувати, і перевести назад в аналогову форму, тобто декодувати, на прийомній стороні [5-8].

Процес перетворення аналогового мовного сигналу в цифрову форму називають аналізом або цифровим кодуванням мови, а зворотний процес відновлення аналогової форми мовного сигналу – синтезом або декодуванням мови.

Ціль будь-якої схеми кодування – одержати таку цифрову послідовність, що вимагає мінімальної швидкості передачі й з якої декодер може відновити вихідний мовний сигнал з мінімальними змінами.

При перетворенні мовного сигналу в цифрову форму, мають місце два процеси – дискретизація, тобто формування дискретних у часі відрахунків амплітуди сигналу, і квантування, тобто дискретизація отриманих значень за амплітудою. Ці дві функції виконуються т.зв. аналого-цифровими перетворювачами (АЦП), які розміщаються в сучасних АТС на платі абонентських комплектів, а у випадку передачі мови по IP-мережах – у терміналі користувача (комп'ютері або IP-телефоні).

Процес аналого-цифрового перетворення одержав, стосовно до систем зв'язку, назву імпульсно-кодової модуляції (ІКМ).

Щоб знизити необхідну швидкість передачі біт, застосовують нелінійний (логарифмічний) закон квантування, тобто квантуванню піддається не амплітуда сигналу, а її логарифм. У цьому випадку має місце процес «стиску» динамічного діапазону сигналу, а при відновленні сигналу відбувається зворотний процес.

Сьогодні застосовуються два основні різновиди ІКМ: з кодуванням по m-закону й по А-закону. У результаті стиску сигнал з амплітудою, що кодується 12-

13 бітами, описується всього вісьма бітами. Розрізняються ці різновиди ІКМ деталями процесу стиску (m-закон кодування переважніше використовувати при малій амплітуді сигналу й при малому відношенні сигнал/шум). У Північній Америці використовується кодування по m-закону, а в Європі – по А-Закону. Тому при міжнародному зв'язку в багатьох випадках потрібне перетворення m-закону в А-закон, відповідальність за яке несе країна, у якій використовується m-закон кодування. В обох випадках кожний відлік кодується 8 бітами, або одним байтом, який можна вважати звуковим фрагментом. Для передачі послідовності таких фрагментів необхідна пропускна здатність каналу, рівна 64 Кбіт/с. Оскільки ІКМ була першою стандартною технологією, що одержала широке застосування в цифрових системах передачі, пропускна здатність каналу, рівна 64 Кбіт/с, стала всесвітнім стандартом для цифрових мереж всіх видів, причому – стандартом, що забезпечує передачу мови з дуже гарною якістю. Однак така висока якість передачі мовного сигналу (що є еталоном при оцінці якості інших схем кодування) досягнута в системах ІКМ за рахунок явно надлишкова, при сучасному рівні технології, швидкості передачі інформації [3].

Щоб зменшити властиву ІКМ надмірність і знизити вимоги до смуги пропущення, послідовність чисел, отримана в результаті перетворення мовного аналогового сигналу в цифрову форму, піддається математичним перетворенням, що дозволяють зменшити необхідну швидкість передачі. Ці перетворення «сирого» цифрового потоку в потік меншої швидкості називають «стиском» (а часто – кодуванням, розглядаючи ІКМ як якусь відправну точку для подальшої обробки інформації). Існує безліч підходів до «стиску» мовної інформації; всі їх можна розділити на три категорії: кодування форми сигналу (waveform coding), кодування вихідної інформації (source coding) і гібридне кодування, що представляє собою сполучення двох підходів.

Кодування форми сигналу. Імпульсно-кодова модуляція, по суті, і являє собою схему кодування форми сигналу. Однак цікавлять більш складні алгоритми, що дозволяють знизити вимоги до смуги пропущення. Розглянуті

методи кодування форми сигналу використовують ту обставину, що між випадковими значеннями декількох послідовних обчислень існує деяка залежність. Це дозволяє з досить високою точністю пророчити значення будь-якого відліку на основі значень декількох попередніх йому обчислень.

При побудові алгоритмів кодування названа закономірність використовується двома способами. По-перше, є можливість змінювати параметри квантування залежно від характеру сигналу. У цьому випадку крок квантування може змінюватися, що дозволяє певною мірою згладити протиріччя між зменшенням числа біт, необхідних для кодування величини відліку при збільшенні кроку квантування, і звуженням динамічного діапазону кодера, неминучим без адаптації. Деякі алгоритми передбачають зміну параметрів квантування приблизно в рамках вимовних складів, а деякі змінюють крок квантування на основі аналізу статистичних даних про амплітуду сигналу, отриманих за відносно короткий проміжок часу. По-друге, існує підхід, називаний диференціальним кодуванням або лінійним пророкуванням. Замість того, щоб кодувати вхідний сигнал безпосередньо, кодують різницю між вхідним сигналом і «передвіщеною» величиною, обчисленою на основі декількох попередніх значень сигналу. Описаний метод називається лінійним пророкуванням, тому що він використовує тільки лінійні функції попередніх обчислень. Найпростішою реалізацією останнього підходу є так звана дельта-модуляція (ДМ), алгоритм якої передбачає кодування різниці між сусідніми обчисленнями сигналу тільки одним інформаційним бітом, забезпечуючи передачу, по суті, тільки знака різниці.

Алгоритмом, побудованим на описані вище принципах, є алгоритм адаптивної диференціальної імпульсно-кодової модуляції (АДІКМ) (G.726). Алгоритм передбачає формування сигналу помилки пророкування і його наступне адаптивне квантування. При досить гарних характеристиках алгоритму, АДІКМ практично не застосовується для передачі мови по мережах з комутацією пакетів, тому що цей алгоритм дуже чутливий до втрат цілих блоків відліку, що

відбуваються при втратах пакетів у мережі. У таких випадках порушується синхронізація кодера й декодера, що приводить до катастрофічного погіршення якості відтворення мови навіть при малій імовірності втрат [1-7].

Перелік вихідних даних

У ході розробки корпоративної системи зв’язку з використанням IP-технологій визначено, що вихідною інформацією є мова або відеозображення на пункті отримання інформації. Як може здатися на перший погляд, вузькополосне кодування мови, що вимагає обчислювальної потужності, є самим складним завданням, виконуваної устаткуванням IP-телефонії. Однак це не так: алгоритми кодування мови стандартизовані й відмінно документовані, більше того, на ринку доступні досить ефективні їхні реалізації для всіх популярних DSP-платформ. З іншого боку, в устаткуванні IP-телефонії повинні бути реалізовані багато інших функцій, спосіб реалізації яких не є об'єктом стандартизації.

На передавальній стороні устаткування IP-телефонії працює за принципом «закодував, передав і забув». На прийомній стороні все набагато складніше. Пакети приходять із мережі із затримкою, що міняється за випадковим законом. Більше того, пакети можуть прийти не в тій послідовності, у якій були передані, а деякі пакети можуть взагалі бути загублені. Приймач повинен справлятися з усіма цими труднощами, забезпечуючи на виході нормальний звуковий потік з тактовою синхронізацією, або генерованим на основі прийнятого потоку даних, або одержуваним із ТфОП по каналах E1. Прив'язка мовних потоків до місцевого тактового сінхросигналу здійснюється шляхом непомітної на слух деформації періодів мовчання у відтвореному сигналі. До цього залишається додати необхідність передачі факсимільної інформації в реальному часі з автоматичним розпізнаванням сигналів факсимільних апаратів і передачу DTMF-сигналів з коректним їхнім відновленням у приймачі.

Сигнали багаточастотного набору номера (DTMF) – просто звукові сигнали, передані по телефонному каналі. При передачі їх по цифровій телефонній мережі не виникає ніяких проблем, тому що кодування за допомогою

алгоритму G.711 не накладає ніяких обмежень на вид звукових сигналів – це може бути мова, сигнали модему, або тональні сигнали – всі будуть успішно відтворені на приймачі [1-5].

Вузькополосні кодеки, щоб досягти низьких швидкостей передачі, використовують той факт, що сигнал, який вони кодують, представляє саме мову. Сигнали DTMF при проходженні через такі кодеки спотворюються й не можуть бути успішно розпізнані приймачем на прийомній стороні [7].

Коли користувачеві ТфОП потрібно ввести якусь додаткову інформацію у віддалену систему при вже встановленому з'єднанні, необхідно забезпечити можливість надійної передачі DTMF-сигналів через мережу IP-телефонії. У випадках, коли система, взаємодіюча з користувачем, просто ставить запитання й чекає введення, тривалість і момент передачі сигналу не важливі. В інших випадках система видає користувачеві список і просить його нажати, наприклад, кнопку «#», як тільки він почує потрібну інформацію; тут ситуація більше складна, і необхідна більше точна прив'язка вчасно.

Існуючі методи передачі сигналів DTMF по мережах IP-телефонії [4, 6].

– Обов'язковий метод. Спеціальне повідомлення протоколу H.245 може містити символи цифр і «*», «#». У цьому випадку використовується надійне TCP-з'єднання, так що інформація не може бути загублена. Однак через особливості TCP можуть мати місце значні затримки [4];

– Нестандартний метод. Він може бути застосований у терміналах H.323v2 при використанні процедури fastStart і відсутності каналу H.245. Для передачі сигналів DTMF відкривається спеціальна RTP-сесія, у якій передаються кодовані значення прийнятих цифр, а також дані про амплітуду й тривалість сигналів. Може бути використана та ж сесія, що й для мови, але зі спеціальним типом корисного навантаження. Використання RTP дозволяє прив'язати DTMF- сигнали до реального часу, що є важливою перевагою даного методу [6].

У принципі, перший метод може бути більше кращим, однак у випадку міжнародних викликів і при використанні віддалених систем, що вимагають

твердої прив'язки введення користувача до часу, може виявитися необхідним застосувати другий метод. Шлюзи IP-телефонії повинні обов'язково придушувати перекручені сигнали DTMF, що пройшли через основний мовний канал. У противному випадку, при відновленні сигналів, про які була прийнята інформація, можуть виникнути неприємні ефекти накладення й розмноження сигналів.

На основі даного огляду функцій устаткування IP-телефонії можна зробити вивід, про тім що, незважаючи на існування стандартних алгоритмів кодування мови, у розроблювачів є величезний простір для діяльності, спрямованої на подальше вдосконалювання технології IP-телефонії.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа і-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою:

оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Встановлення програмного додатку на ПК здійснюється простим копіюванням програмної папки IP_TEL_NET на жорсткий диск ПК. Нижче наведено вміст вказаної папки:

– **IpTel.exe** – виконавчий файл програми;

– **abc.ttf** – файл зі шрифтом.

Налаштування зв'язку між кінцевими абонентами

1. Після запуску програми на екран виводиться вікно завантаження програмного забезпечення та інформації про автора, яке спливає.

2. Після цього необхідно пересвідчитися у наявності доступу до мережі. Для цього необхідно перевірити наявність мережної карти у персональному комп'ютері та підключеного шнура до мережі. Перевірити системні параметри мережної карти. Зайти в меню **Пуск→Налаштування→Мережні підключення→Налаштування мережних підключень**: вибрати**: IP-адрес, шлюз, маску під мережі, використовуваний протокол (TCP/IP).**

На мережній карті повинна горіти зелена лампочка. Це каже про те, що фізично мережна карта працездатна та готова до роботи з мережею.

3. Потім необхідно перевірити параметри установки файрволу:

– системи дозволу IP-портів;

– привілеї користувача;

– поточний статус файрволу та перевірити наявність на одночасну роботу декількох файрволів на персональному комп'ютері.

4. Перевірити на наявність вірусів антивірусом та файрволом, або включити максимальний рівень захисту від вірусів. Антивіруси забезпечують найвищий рівень захисту. Виявляються й видаляються всі типи вірусів і троянських програм, ворожі об'єкти Java/Active, блокується доступ до

небезпечних WEB-ресурсів.Захист від вірусів охоплює всі операційні системи й групові додатки, які використовуються в сучасних корпоративних мережах: клієнтські ОС, Macintosh; серверні ОС Windows NT, Novell Netware, FreeBSD, Linux, HP-UX, AIX, SCO, Solaris; групові додатки MS Exchange і Lotus Notes/Domino; інтернет-шлюзи Windows і Sun Solaris; мережні пристрої зберігання даних NetApp; ОС мікрокомп'ютерів (PDA), EPOC (Psion).

5. Якщо є IP-телефон, то підключити його згідно інструкції користувача, яка до нього додається.

6. Запустити розроблене у ході виконання магістерської роботи програмне забезпечення на кінцевих персональних комп’ютерах.

7. Визначити IP-адресу кінцевих машин. Для цього необхідно виконати наступні дії:

– Отримати інформаційні дані від адміністратора локальної мережі.

– У разі неможливості виконання попереднього пункту необхідно виконати наступні дії: зайти на закладку: **Пуск→Налаштування→Мережні підключення** вибрати ярлик мережного підключення й нажати на ньому правою кнопкою миші. У контекстному меню, яке з’явиться вибрати: **Властивість→Вибір пункта TCP/IP→в закладці IP-адрес**. Там буде відображено IP-адресу персонального комп’ютера у локальній мережі.

8. Заповнити базу даних IP-адрес та легенду до кожної з неї.

9. Після виконання усіх операцій, перерахованих вище, працюємо з розробленим програмним забезпеченням, згідно інструкції користувача.

На рисунку 5.1 зображено вікно авторизації. Система захищена програмним чином і її активізація здійснюється через введення пароля у відповідне поле та натисненням кнопки. По замовчуванню встановлено пароль «123». За умови коректного введення пароля система надає користувачеві відповідний доступ для входу в систему, в іншому випадку програма закінчує свою роботу.

Існує можливість зміни паролю. Для цього у вікні зміни паролю необхідно ввести старий пароль, ввести новий пароль та продублювати новий пароль. Фіксація паролю здійснюється кнопкою “**Ок**”.

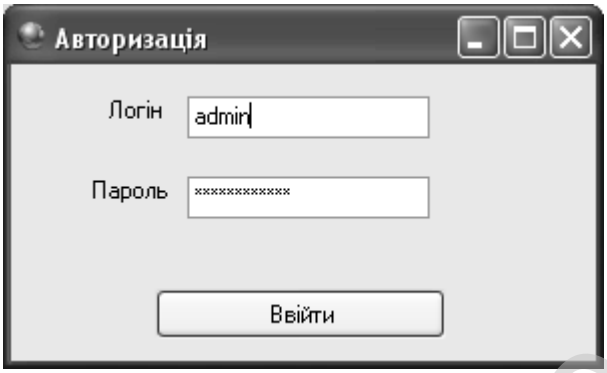

Рисунок 5.1 – Вікно авторизації

Організація бази клієнтів

В додатку є можливість редагування IP-адрес та легенди до них. Зміни вносяться безпосередньо в сітку відповідних таблиць, їх фіксація здійснюється за допомогою кнопки **ОК**.

Організація селективного зв’язку

Для організації селективного зв’язку необхідно за нажати клавішу Ctrl та за допомогою лівої кнопки миші виділити ті номера з якими потрібно організувати зв’язок (рисунок 5.2).

Можливі проблеми та їх усунення

Причинами неможливості доступу мережі можуть бути:

– проблеми з мережею;

– проблеми з мережною картою;

– неактивність сервера БД або некоректність його встановлення.

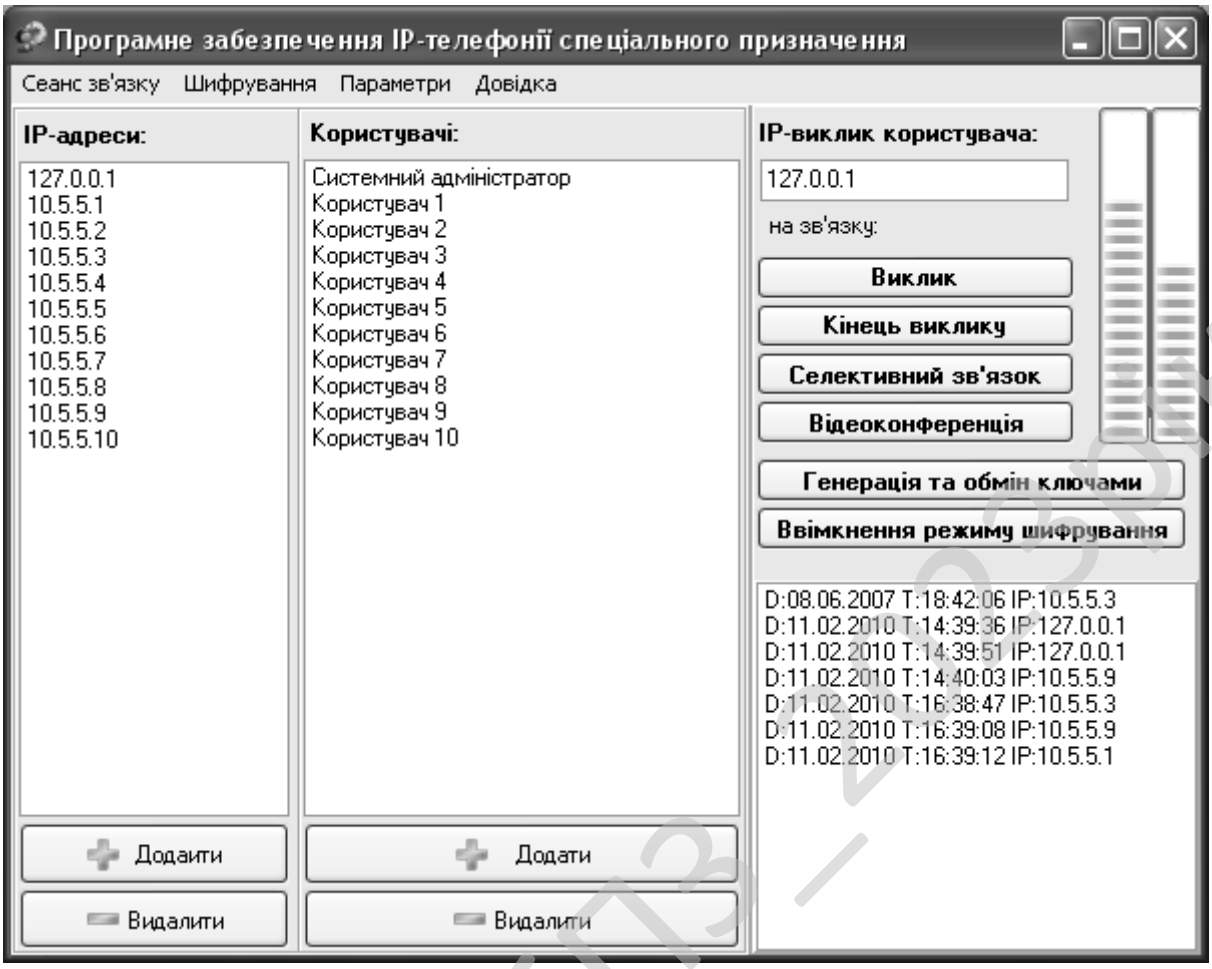

Рисунок 5.2 – Головне вікно програми

Перевірка запуску сервера БД здійснюється через запуск менеджера сервера БД із панелі управління Windows

При цьому на ПК не повинно бути файрволів або проксі, які не дають йому приймати з'єднання на порту 3050 по прокотолу tcp/ip (або файрвол/проксі потрібно налаштувати, щоб не заважали СУБД Interbase).

Після цього потрібно переконатись, що якщо у вас в мережі немає серверів Netware, то не один клієнт або сервер не має встановленої підтримки протоколів IPX/SPX (NWLink і т.п). Цей протокол не потрібний для роботи.

Взагалі, якщо ОС встановлена коректно, і мережа правильно налаштована, то InterBase або Firebird повинні також функціонувати нормально. З приводу

налаштування мережі слід звертатись до її адміністратора або програмної документації ОС.

Причинами помилки можуть бути наступні:

– клієнтська частина (gds32.dll, fbclient.dll) не відповідає версії сервера;

– клієнтська частина не підтримує локальний протокол взагалі (наприклад, в Firebird 1.5.1 for Windows, Classic). Також локальний протокол не працює, якщо на Win2003 або WinXP активізовані служби Terminal Services;

– особливості конкретного логіна або версії операційної системи.

У всіх випадках проблем з локальним протоколом рекомендується перевірити всі вище перераховані варіанти, і якщо їх не вдалось усунути – використовувати локальний мережевий протокол для з'єднання з БД.

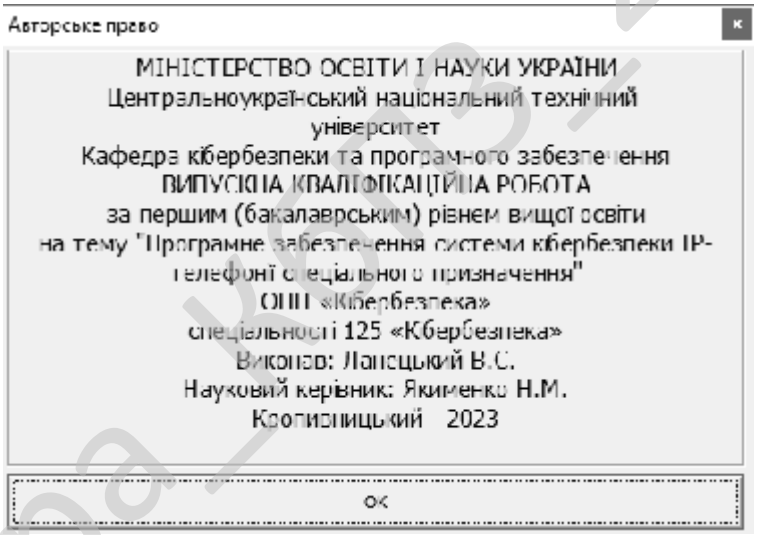

Рисунок 5.3 – Вікно довідки

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки IP-телефонії спеціального призначення.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем IP-телефонії спеціального призначення.

– Досліджена система IP-телефонії спеціального призначення.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки IP-телефонії спеціального призначення.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання IP-телефонії спеціального призначення.

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки IP-телефонії спеціального призначення. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MISTY1.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бакланов И.Г. ISDN и IP-телефония / Вестник связи, 1999, №4.
2. Будников В.Ю., Пономарев Б.А. Технологии обеспечения качества обслуживания в мультисервісних сетях / Вестник связи, 2000. №9.
3. Варламова Е. IP-телефония в России / Connect! Мир связи, 1999, №9.
4. Гольдштейн Б.С. Сигнализация в сетях связи. Том 1. М.: Радио и зв'язок, 1998.
5. Гольдштейн Б.С. Протоколы сети доступа. Том 2. М.: Радио и зв'язок, 1999.
6. Гольдштейн Б.С., Ехриель И.М., Рерле Р.Д. Интеллектуальні сети. М.: Радио и зв'язок, 2000.
7. Кузнецов А.Е., Пинчук А. В., Суховицкий А.Л. Построение сетей IP-телефонии / Компьютерная телефония, 2000, №6.
8. Кульгин М. Технологии корпоративних сетей. Изд. «Питер», 1999.
9. Ломакин Д. Технические решения IP-телефонии / Мобильні системы, 1999 №8.
10. Мюнх Б., Скворцова С. Сигнализация в сетях IP-телефонии. -Часть I, II/Сети и системы связи, 1999. – №13(47), 14(48).
11. Cisco Voice Over IP. Student Guide – Cisco Systems Inc, 2003.
12. Cisco IP Telephony. Student Guide. Version 3.3. – Cisco Systems Inc, 2002.
13. Robert Padjen – Cisco AVVID and IP Telephony. Design & Implementation – Syngress Publishing Inc, 2001.
14. Paul J. Fong – Configuring Cisco Voice Over IP. Second Edition – Syngress Publishing Inc, 2001.
15. ITU-T Recommendation G.723.1. Dual Rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kit / sec. – 1996.

16. ITU-T Recommendation G.729. Speech codec for multimedia telecommunications transmitting at 8 / 13 kbit / s. – 1996.

17. ITU-T Recommendation H.225.0. Call signaling protocols and media stream packetization for packet-based multimedia communication systems. -Geneva, 1998.

18. ITU-T Recommendation H.245. Control protocol for multimedia communication. -Geneva, 1998.

19. ITU-T Recommendation H.248. Gateway control protocol. – Geneva, 2000.

20. ITU-T Recommendation H.320. Narrow-band Visual Telephone Systems and Terminal Equipment. – 1996.

21. ITU-T Recommendation H.321. Adaptation of H.320 Visual Telephone Terminals to B-ISDN Environments. – 1996.

22. ITU-T Recommendation H.322. Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Guaranteed Quality of Service. – 1996.

23. ITU-T Recommendation H.323. Packet based multimedia communication systems. – Geneva, 1998.

24. ITU-T Recommendation H.324. Terminal for Low Bit Rate Multimedia Communications. -1996.

25. ITU-T Recommendation Q.931. ISDN User-Network Interface Layer 3 Specification for Basic Call Control. – 1993.

26. RFC 2705. Media Gateway Control Protocol (MGCP) Version 1.0. M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett. October 1999.

27. RFC 2865. Remote Authentication Dial In User Service (RADIUS). C.Rigney, S. Willens, A. Rubens, W. Simpson. June 2000.

28. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.

29. Смирнов А.А. Разработка математической GERT-модели технологии

распространения компьютерных вирусов в информационно-телекоммуникационных сетях / А.А.Смирнов, Мохамад Гани Абу Таам // Информационные системы в управлении, образовании, промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – 498 с.

30. Смирнов А.А. Метод управления доступом в интеллектуальных узлах коммутации / Мохамад Гани Абу Таам, А.А.Смирнов // Информационные технологии и защита информации в информационно-коммуникационных системах: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – 486 с.

31. Смирнов А.А. Математическая GERT-модель технологии передачи метаданных в облачные антивирусные системы / В.В.Босько, А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Збірник наукових праць "Системи обробки інформації". – Випуск 1(117). – Х.: ХУПС – 2014. – С. 137-141.

32. Смирнов А.А. Структурно-логическая GERT-модель технологии распространения компьютерных вирусов / А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Системи управління, навігації та зв’язку. – Випуск 1(29). – П.: ПНТУ. – 2014. – С. 120-125.

33. Смирнов А.А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Гани Абу Таам, А.А. Смирнов, А.В. Коваленко, С.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 9(125). – Х.: ХУПС – 2014. – С. 105-110.

34. Смирнов А.А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 4 (41). – Харків: ХУПС. – 2014. – С. 48-52.

35. Смирнов А.А. Исследование показателей качества функционирования

интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 4(17). – Харків: ХУПС. – 2014. – С.90-95.

36. Смирнов А.А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 1(126). – Х.: ХУПС – 2015. – С. 150-153.

37. Смирнов А.А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Системи озброєння і військова техніка. – Випуск 3(43) – Х.: ХУПС – 2015. – С. 100-107.

38. Смирнов А.А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 3(19). – Х.: ХУПС. – 2015. – С. 134-141.

39. Mohamad Abou Taam Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

40. Смирнов А.А. GERT-модель технологии передачи данных в облачные антивирусные системы / А.А. Смирнов, В.В. Босько, Мохамад Гани Абу Таам // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 18-19.

41. Смирнов А.А. Математическое моделирование технологии передачи сигнатур в облачные антивирусные системы / Мохамад Гани Абу Таам, А.А. Смирнов // Збірник тез VI міжнародної науково-практичної конференції

“Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 260.

42. Смирнов А.А. Анализ требований к качеству обслуживания в информационно-телекоммуникационных системах / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVI міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 11-12 квітня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 124-126.

43. Смирнов А.А. Дослідження та реалізація GERT-моделі технології розповсюдження комп’ютерних вірусів для захисту телекомунікаційних систем / Мохамад Гані Абу Таам, С.А. Смирнов // Збірник тез науково-практичної конференції «Інформаційні технології та комп’ютерна інженерія». м. Кіровоград. 4 грудня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 168.

44. Смирнов А.А. Исследование математических моделей технологии распространения компьютерных вирусов / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник наукових праць міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 25-28 лютого 2015 р. – Київ: Європейський університет. – 2015. – С. 90-91.

45. Смирнов А.А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез всеукраїнської науково-практичної конференції «Інформаційна безпека держави, суспільства та особистості». м. Кіровоград. 16 квітня 2015. – Кіровоград: КНТУ. – 2015. – С. 50-52.

46. Смирнов А.А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник тез VII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2015 р. – Харків: ХНЕУ. – 2015. – С. 14.

47. Смирнов А.А. Реализация метода управления доступом в

интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

48. Смирнов А.А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез II Міжнародної науково-практичної Інтернет-конференції «Інформаційна та економічна безпека» (INFECO-2015)». м. Харків. 21-22 травня 2015 р. – Харків: ХІБС УБС НБУ. – 2015. – С. 20-24.

49. Смирнов А.А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Сборник тезисов XI международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 01 – 06 июня 2015 г – Варна. ТУВ. – 2015. – С. 488-491

50. Смирнов А.А. Метод управления доступом к облачным телекоммуникационным ресурсам для обеспечения защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Комп'ютерні технології та інформаційна безпека». м. Кіровоград. 2-3 липня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 4-5.

51. Смирнов А.А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації». м. Затока. 7-9 вересня 2015 р. – Одеса: ОНАЗ. – 2015. – С. 90-94.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки IP-телефонії спеціального призначення.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки IP-телефонії спеціального призначення.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки IP-телефонії спеціального призначення;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 101 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи кібербезпеки IP-телефонії спеціального призначення

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 34

Літера: РП

Кропивницький – 2023 року

Підпрограма запуску завантажувального вікна

```
unit U_SPLASH;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, jpeg, ExtCtrls, StdCtrls, Gauges;

type
  TForm_SPLASH = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Gauge1: TGauge;
    Timer1: TTimer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_SPLASH: TForm_SPLASH;

implementation

{$R *.dfm}

end.
```

Основна програма – Програма організації зв`язку

```
program iptel;

uses
  Forms,
  Dialogs,
  Windows,
  Sysutils,
  Messages,
  usimple in 'usimple.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2},
  Unit3 in 'Unit3.pas' {Form3},
  Unit1 in 'Unit1.pas' {Form0},
  Unit4 in 'Unit4.pas' {Form4};
{$R simple.RES}
var
  i:integer;
{        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;}
begin
Application.Initialize;
{
if FileExists('main.dat')=false then
 begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
 end else
 begin
   AssignFile(F, 'main.dat');
   Reset(F);
   Readln(F, S);
   CloseFile(F);
      if Crypt(InputBox('Увага!','Введіть пароль',Y),KEY,false)=S then
      begin
           MessageDlg('Дякую, пароль вірний!', mtInformation,[mbOK],0);}
           Try
                Form2:=TForm2.Create(Application);
                Form2.Show;
                Form2.Update;
                for i:=1 to 10 do
                begin
                  sleep(200);
                  Form2.ProgressBar1.Position:=i*10;
                  Form2.Update;
                end;
              Application.Title := 'IpTel';
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm0, Form0);
  Application.CreateForm(TForm4, Form4);
  Finally
                Form2.Free;
   End;
     Application.Run;
{      end
      else
      begin
      MessageDlg('Невірний пароль!',mtInformation,[mbOK],0);
```

```
        Application.Terminate;
        end;
    end;}
end.
```

Головне вікно програми

```
unit usimple;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  telefoon, StdCtrls, ComCtrls, ExtCtrls, jpeg, Menus, Buttons;

type
  TForm1 = class(TForm)
    Telefoon1: TTelefoon;
    StatusBar1: TStatusBar;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Button2: TButton;
    Button1: TButton;
    Panel5: TPanel;
    TabSheet2: TTabSheet;
    Panel6: TPanel;
    Label2: TLabel;
    Panel7: TPanel;
    MainMenu1: TMainMenu;
    Panel8: TPanel;
    ListBox1: TListBox;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    ListBox2: TListBox;
    Panel9: TPanel;
    Label3: TLabel;
    Panel10: TPanel;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    N1: TMenuItem;
    IP1: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    BitBtn5: TBitBtn;
    Memo1: TMemo;
    TabSheet3: TTabSheet;
    Panel11: TPanel;
    Image3: TImage;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    N11: TMenuItem;
    N12: TMenuItem;
    ListBox3: TListBox;
    Label9: TLabel;
    Label10: TLabel;
    N2: TMenuItem;
    PB1: TProgressBar;
    PB2: TProgressBar;
    T1: TTimer;
    Image1: TImage;
    Timer1: TTimer;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ListBox2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N3Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure T1Timer(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit3, Unit2, Unit4, Unit1;
var
  KEY:string='2%#$T%&(*qwrdas@@@#45';
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
 Telefoon1.placecall(Edit1.text);
 ListBox3.items.add('D:'+DateToStr(now)+'                T:'+TimeToStr(now)+'
IP:'+Edit1.text);
  T1.Enabled:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
 Telefoon1.calling:=false;
 T1.Enabled:=false;
 PB1.Position:=0;
 PB2.Position:=0;
end;
procedure TForm1.ListBox1Click(Sender: TObject);
begin
 Edit1.text:=ListBox1.Items.Strings[ListBox1.ItemIndex];
end;
procedure TForm1.ListBox2Click(Sender: TObject);
begin
 Label4.Caption:=ListBox2.Items.Strings[ListBox2.ItemIndex];
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  InputString: string;
begin
  InputString:=  InputBox('Додавання  ip  адреси',  'Прошу',  'Введення  IP  не
здійснено');
  if (InputString<>'Введення IP не здійснено') then
  begin
  Listbox1.Items.add(InputString);
  end;
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
var
```

```
  InputString: string;
begin
 InputString:=  InputBox('Додавання легенди ip адреси', 'Прошу', 'Введення
легенди не здійснено');
 if (InputString<>'Введення легенди не здійснено') then
 begin
 Listbox2.Items.add(InputString);
 end;
end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 ListBox1.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox1.Items.Delete(ListBox1.ItemIndex);
end;

procedure TForm1.BitBtn4Click(Sender: TObject);
begin
 ListBox2.Items.Strings[ListBox1.ItemIndex]:='';
 ListBox2.Items.Delete(ListBox1.ItemIndex);
end;
procedure TForm1.N10Click(Sender: TObject);
begin
 Application.Terminate;
end;
procedure TForm1.BitBtn5Click(Sender: TObject);
var
 i:integer;
 G:boolean;
begin
 Form3.ListBox1.Clear;
 g:=false;
 for i:=0 to (ListBox1.Items.Count - 1) do
 begin
  if ListBox1.Selected[i] then
    begin
        Form3.ListBox1.Items.add(ListBox1.Items.Strings[i]);
        g:=true;
    end;
    if g then Form3.show;
 end;
end;
procedure TForm1.N12Click(Sender: TObject);
var
  F: TextFile;
  H,S:string;
  OLD:string;
        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;
begin
if FileExists('main.dat') then
begin
    AssignFile(F, 'main.dat');
    Reset(F);
    Readln(F, S);
     CloseFile(F);
  if Crypt(InputBox('Увага!','Введіть старий пароль',''),KEY,false)=S then
    begin
        H:=Crypt(InputBox('Увага!','Введіть новий пароль',''),KEY,false);
        if H<>'' then
        begin
```

```
          MessageDlg('Дякую пароль змінено',mtInformation,[mbOK],0);
          AssignFile(F, 'main.dat');
          Rewrite(F);
          Writeln(F,H);
          CloseFile(F);
        end else  MessageDlg('Введіть значення!',mtInformation,[mbOK],0);
    end
    else
     begin
        MessageDlg('Файл main.dat не знайдено чи пароль невірний! завершення
програми',mtInformation,[mbOK],0);
        Application.Terminate;
      end;
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
randomize;
PB1.Position:=0;
PB2.Position:=0;
T1.Enabled:=false;
Timer1.Enabled:=true;
if FileExists('MainLegend.dat')=false then
 begin
    MessageDlg('Файл MainLegend.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox2.Items.LoadFromFile('MainLegend.dat');
  end;
if FileExists('MainIP.dat')=false then
 begin
    MessageDlg('Файл MainIP.dat не знайдено!',mtInformation,[mbOK],0);
 end else
 begin
    ListBox1.Items.LoadFromFile('MainIP.dat');
  end;
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.LoadFromFile('MainHISTORY.dat');
if  FileExists('mainHelp.dat')=false  then  MessageDlg('Файл  mainHelp.dat  не
знайдено!',mtInformation,[mbOK],0)
else Memo1.Lines.LoadFromFile('mainHelp.dat');
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if FileExists('MainLegend.dat')=false then MessageDlg('Файл MainLegend.dat не
знайдено!',mtInformation,[mbOK],0)
else ListBox2.Items.SaveToFile('MainLegend.dat');
if   FileExists('MainIP.dat')=false   then   MessageDlg('Файл   MainIP.dat   не
знайдено!',mtInformation,[mbOK],0)
else ListBox1.Items.SaveToFile('MainIP.dat');
if  FileExists('MainHISTORY.dat')=false  then  MessageDlg('Файл  MainHISTORY.dat
не знайдено!',mtInformation,[mbOK],0)
else ListBox3.Items.SaveToFile('MainHISTORY.dat');
end;
procedure TForm1.N3Click(Sender: TObject);
begin
TabSheet2.Visible:=true;
end;
procedure TForm1.N6Click(Sender: TObject);
begin
 Telefoon1.calling:=false
end;
procedure TForm1.N2Click(Sender: TObject);
begin
Form0.show;
end;
procedure TForm1.FormShow(Sender: TObject);
begin
```

```
PB1.Position:=0;
PB2.Position:=0;
end;
procedure TForm1.T1Timer(Sender: TObject);
var
i1,i2:integer;
begin
PB1.Position:=random(100);
PB2.Position:=PB1.Position;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Form4.show;
Form1.hide;
Timer1.Enabled:=false;
end;
end.
```

Підпрограма налагодження звукової карти AudioSettings

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, AMixer, MMSystem;
type
  TForm0 = class(TForm)
    ComboBox1: TComboBox;
    ComboBox2: TComboBox;
    TrackBar: TTrackBar;
    CheckBox: TCheckBox;
    Label1: TLabel;
    Label2: TLabel;
    Mixer: TAudioMixer;
    Label3: TLabel;
    Label4: TLabel;
    ComboBox3: TComboBox;
    LabelStereo: TLabel;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure MixerControlChange(Sender: TObject; MixerH, ID: Integer);
    procedure TrackBarChange(Sender: TObject);
    procedure CheckBoxClick(Sender: TObject);
    procedure ComboBox3Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    Setting:Boolean;
  public
    { Public declarations }
  end;
var
  Form0: TForm0;
implementation
uses usimple;
{$R *.DFM}
procedure TForm0.FormCreate(Sender: TObject);
var A:Integer;
begin
  For A := 0 to Mixer.MixerCount - 1 do
    ComboBox3.Items.Add ('Mixer '+IntToStr(A));
  If (ComboBox3.Items.Count > 0) then
    ComboBox3.ItemIndex := 0;
  ComboBox3Change (Sender);
end;
procedure TForm0.ComboBox1Change(Sender: TObject);
var A:Integer;
begin
  ComboBox2.Items.Clear;                                    ComboBox2.Items.Add
(Mixer.Destinations[ComboBox1.ItemIndex].Data.szName);
  For A:=0 to Mixer.Destinations[ComboBox1.ItemIndex].Connections.Count-1 do
ComboBox2.Items.Add(Mixer.Destinations[ComboBox1.ItemIndex].Connections[A].Dat
a.szName);
  If ComboBox2.Items.Count>0 then
  begin
    ComboBox2.ItemIndex:=0;     ComboBox2Change (Self);
  end;
end;
procedure TForm0.ComboBox2Change(Sender: TObject);
var L,R,M:Integer;
    VD,MD:Boolean;
    Stereo:Boolean;
    IsSelect:Boolean;
begin
```

```
  Mixer.GetVolume                    (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,L,R,M,Stereo,VD,MD,IsSelect);
  Setting:=True;
  TrackBar.Visible:=not VD;
  Label1.Visible:=not VD;
  Label3.Visible:=VD;
  If TrackBar.Visible then
    TrackBar.Position:=L;
  CheckBox.Visible:=not MD;
  Label2.Visible:=not MD;
  Label4.Visible:=MD;
  If CheckBox.Visible then
    CheckBox.Checked:=M<>0;
  If (Stereo) then
    LabelStereo.Caption := '- stereo -'
  else
    LabelStereo.Caption := '- mono -';
  Setting:=False;
end;
procedure TForm0.MixerControlChange(Sender: TObject; MixerH, ID: Integer);
begin
  ComboBox2Change (Self);
end;
procedure TForm0.TrackBarChange(Sender: TObject);
begin
  If (not Setting) then
  begin
    Setting:=True;
    Mixer.SetVolume                  (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.CheckBoxClick(Sender: TObject);
begin
  If not Setting then
  begin
    Setting:=True;
    Mixer.SetVolume                  (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;
procedure TForm0.ComboBox3Change(Sender: TObject);
var A:Integer;
begin
  If (ComboBox3.ItemIndex >= 0) AND (ComboBox3.ItemIndex < Mixer.MixerCount)
then
    Mixer.MixerId := ComboBox3.ItemIndex;
  ComboBox1.Items.Clear;
  If Mixer.MixerCount>0 then
  begin
    For A:=0 to Mixer.Destinations.Count-1 do
      ComboBox1.Items.Add (Mixer.Destinations[A].Data.szName);
    If ComboBox1.Items.Count>0 then
    begin
      ComboBox1.ItemIndex:=0;    ComboBox1Change (Self);
    end;
  end
  else
   begin
    ComboBox1.OnChange:=nil;     ComboBox2.OnChange:=nil;
    TrackBar.OnChange:=nil;       CheckBox.OnClick:=nil;
    MessageDlg ('No mixer present in the system !',mtError,[mbOK],0);
  end;
  Setting:=False;
end;
procedure TForm0.Button1Click(Sender: TObject);
begin
```

```
Form0.hide;  Form1.show;
end;
procedure TForm0.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form0.hide; Form1.show;
end;
end.
```

Підпрограма виклику вікна налагодження звукової карти AudioSettings

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  jpeg, ExtCtrls, StdCtrls, ComCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    RichEdit1: TRichEdit;
    ProgressBar1: TProgressBar;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

end.
```

Підпрограма селективного зв`язку

```
unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, jpeg, ComCtrls;

type
  TForm3 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    ListBox1: TListBox;
    Animate1: TAnimate;
    Panel4: TPanel;
    BitBtn2: TBitBtn;
    BitBtn1: TBitBtn;
    BitBtn3: TBitBtn;
    Image1: TImage;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses usimple;

{$R *.DFM}

procedure TForm3.BitBtn2Click(Sender: TObject);
begin
form3.hide;
end;

procedure TForm3.BitBtn1Click(Sender: TObject);
begin
Animate1.Active:=true;
form1.telefoon1.placecall(form1.edit1.text);
end;

procedure TForm3.BitBtn3Click(Sender: TObject);
begin
Form1.Telefoon1.calling:=false;
Animate1.Active:=false;
end;

end.
```

Підпрограма парольного захисту

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Mask, ExtCtrls;
type
  TForm4 = class(TForm)
    Panel1: TPanel;
    M: TMaskEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form4: TForm4;
implementation
uses usimple;
VAR
  KEY:string='2%#$T%&(*qwrdas@@@#45';
  DAT:string='VUW';
{$R *.DFM}
        function Crypt(Text,Key: String; Encode: boolean): String;
        var
          i, KeyLength: integer;
          Sign: ShortInt;
        begin
         KeyLength:=Length(Key);
         if Encode then Sign :=-1 else Sign:=1;
         for i:=1 to Length(Text) do
          Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
          Result:=Text;
        end;
procedure TForm4.BitBtn1Click(Sender: TObject);
var
  F: TextFile;
  i:integer;
  s:string;
  Y:string;
  UUU:string;
begin
if FileExists('main.dat')=TRUE then
 begin
   AssignFile(F, 'main.dat');
   Reset(F);
   Readln(F, S);
   CloseFile(F);
UUU:=Crypt(M.text,KEY,false);
if UUU=S then
      begin
           Form4.hide;
           Form1.show;
           MessageDlg('Дякую, пароль вірний!',mtInformation,[mbOK],0);
      end
      else MessageDlg('Введіть пароль!',mtInformation,[mbOK],0);
 end
      else       begin
    MessageDlg('Файл       main.dat       не       знайдено!       завершення
програми',mtInformation,[mbOK],0);
    Application.Terminate;
      end;
end;
end.
```

Розроблений компонент налагодженя звука

```
unit AMixer;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  MMSystem;

type
  TAudioMixer=class;

  TPListFreeItemNotify=procedure (Pntr:Pointer) of object;
  TMixerChange=procedure (Sender:TObject;MixerH:HMixer;ID:Integer) of object;
    {MixerH is handle of mixer, which sent this message.
     ID is ID of changed item (line or control).}

  TPointerList=class(TObject)
  private
    FOnFreeItem:TPListFreeItemNotify;
    Items:Tlist;
  protected
    function GetPointer (Ind:Integer):Pointer;
    function GetCount :integer;
  public
    constructor Create;
    destructor Destroy; override;
    procedure Clear;
    procedure Add (Pntr:Pointer);
    property Count:Integer read GetCount;
    property Pointer[Ind:Integer]:Pointer read GetPointer; default;
    property   OnFreeItem:TPListFreeItemNotify   read   FOnFreeItem   write
FOnFreeItem;
  end;

  TMixerControls=class(TObject)
  private
    heap:pointer;
    FControls:TPointerList;
  protected
    function GetControl (Ind:Integer):PMixerControl;
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Control[Ind:Integer]:PMixerControl read GetControl; default;
    property Count:Integer read GetCount;
  end;

  TMixerConnection=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;

  TMixerConnections=class(TObject)
  private
    XMixer:TAudioMixer;
    FConnections:TPointerList;
  protected
    procedure DoFreeItem (Pntr:Pointer);
    function GetConnection (Ind:Integer):TMixerConnection;
```

```
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property  Connection[Ind:Integer]:TMixerConnection  read  GetConnection;
default;
    property Count:Integer read GetCount;
  end;

  TMixerDestination=class(TObject)
  private
    XMixer:TAudioMixer;
    FData:TMixerLine;
    FControls:TMixerControls;
    FConnections:TMixerConnections;
  public
    constructor Create (AMixer:TAudioMixer; AData:TMixerLine);
    destructor Destroy; override;
    property Connections:TMixerConnections read FConnections;
    property Controls:TMixerControls read FControls;
    property Data:TMixerLine read FData;
  end;

  TMixerDestinations=class(TObject)
  private
    FDestinations:TPointerList;
  protected
    function GetDestination (Ind:Integer):TMixerDestination;
    procedure DoFreeItem (Pntr:Pointer);
    function GetCount:Integer;
  public
    constructor Create (AMixer:TAudioMixer);
    destructor Destroy; override;
    property Count:Integer read GetCount;
    property  Destination[Ind:Integer]:TMixerDestination  read  GetDestination;
default;
  end;

  TAudioMixer = class(TComponent)
  private
    XWndHandle:HWnd;

    FDestinations:TMixerDestinations;
    FMixersCount:Integer;
    FMixerHandle:HMixer;
    FMixerId:Integer;
    FMixerCaps:TMixerCaps;
    FDriverVersion: MMVERSION;
    FManufacturer: String;
    FProductId: Word;
    FNumberOfLine: Integer;
    FProductName: String;
    FOnLineChange:TMixerChange;
    FOnControlChange:TMixerChange;
  protected
    procedure SetMixerId (Value:Integer);
    procedure MixerCallBack (var Msg:TMessage);
    procedure CloseMixer;
  published
    constructor Create (AOwner:TComponent); override;
    destructor Destroy; override;
    property DriverVersion: MMVERSION read FDriverVersion;
    property ProductId: WORD read FProductId;
    property NumberOfLine: Integer read FNumberOfLine;
    property Manufacturer: string read FManufacturer;
    property ProductName: string read FProductName;
    property MixerId:Integer read FMixerId write SetMixerId;
      {Opened mixer - value must be in range 0..MixersCount-1
       If no mixer is opened this value is  -1}
```

```
    property OnLineChange:TMixerChange read FOnLineChange write FOnLineChange;
    property OnControlChange:TMixerChange read FOnControlChange write
FOnControlChange;
  public
    function GetVolume (ADestination, AConnection:Integer; var LeftVol,
RightVol, Mute:Integer; var Stereo, VolDisabled, MuteDisabled,
MuteIsSelect:Boolean):Boolean;
    function SetVolume (ADestination, AConnection:Integer; LeftVol, RightVol,
Mute:Integer):Boolean;
    function GetPeak(ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer):Boolean;
    function GetMute(ADestination, AConnection:Integer; var
Mute:Boolean):Boolean;
    function SetMute(ADestination, AConnection:Integer; Mute:Boolean):Boolean;

    property Destinations:TMixerDestinations read FDestinations;
      {Ind must be in range 0..DestinationsCount-1}
    property MixerCaps:TMixerCaps read FMixerCaps;
    property MixerCount:Integer read FMixersCount;
      {Number of mixers present in system; mostly 1}
    property MixerHandle:HMixer read FMixerHandle;
      {Handle of opened mixer}
  end;

procedure Register;

implementation

{------------}
{TPointerList}
{------------}

constructor TPointerList.Create;
begin
  Items := TList.Create;
end;

destructor TPointerList.Destroy;
begin
  Clear;
  Items.Free;
end;

procedure TPointerList.Add (Pntr:Pointer);
begin
  Items.Add (Pntr);
end;

function TPointerList.GetPointer (Ind:Integer):Pointer;
begin
  Result := nil;
  If (Ind < Count) then
    Result := Items[Ind];
end;

procedure TPointerList.Clear;
var I:Integer;
begin
  for I := 0 to Items.Count-1 do begin
    If Assigned (FOnFreeItem) then
      FOnFreeItem (Items[I])
  end;
  Items.Clear;
end;

function TPointerList.GetCount:Integer;
begin
  Result := Items.Count;
end;
```

```
{--------------}
{TMixerControls}
{--------------}
constructor TMixerControls.Create (AMixer:TAudioMixer; AData:TMixerLine);
var MLC:TMixerLineControls;
    A,B:Integer;
    P:PMixerControl;
begin
  FControls := TPointerList.Create;
  GetMem (P, SizeOf(TMixerControl)*AData.cControls);
  heap := P;
  MLC.cbStruct := SizeOf(MLC);
  MLC.dwLineID := AData.dwLineID;
  MLC.cbmxctrl := SizeOf(TMixerControl);
  MLC.cControls := AData.cControls;
  MLC.pamxctrl := P;
  A          :=          MixerGetLineControls(AMixer.MixerHandle,          @MLC,
MIXER_GETLINECONTROLSF_ALL);
  If A = MMSYSERR_NOERROR then
  begin
    For B := 0 to AData.cControls-1 do
    begin
      FControls.Add (P);
      P := PMixerControl (DWORD(P) + sizeof (TMixerControl));
    end;
  end;
end;

destructor TMixerControls.Destroy;
begin
  FControls.free;
  freemem(heap);
  inherited;
end;

function TMixerControls.GetControl (Ind:Integer):PMixerControl;
begin
  Result := FControls.Pointer[Ind];
end;

function TMixerControls.GetCount:Integer;
begin
  Result := FControls.Count;
end;

{---------------}
{TMixerConnection}
{---------------}

constructor TMixerConnection.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FControls := TMixerControls.Create (AMixer, AData);
end;

destructor TMixerConnection.Destroy;
begin
  FControls.Free;
  inherited;
end;

{-----------------}
{TMixerConnections}
{-----------------}

constructor TMixerConnections.Create (AMixer:TAudioMixer; AData:TMixerLine);
var A,B:Integer;
```

```
    ML:TMixerLine;
begin
  XMixer := AMixer;
  FConnections := TPointerList.Create;
  FConnections.OnFreeItem := Dofreeitem;
  ML.cbStruct := SizeOf(TMixerLine);
  ML.dwDestination := AData.dwDestination;
  For A := 0 to AData.cConnections-1 do
  begin
    ML.dwSource := A;
    B        :=        MixerGetLineInfo        (AMixer.MixerHandle,        @ML,
MIXER_GETLINEINFOF_SOURCE);
    If B = MMSYSERR_NOERROR then
      FConnections.Add (Pointer(TMixerConnection.Create (XMixer, ML)));
  end;
end;

destructor TMixerConnections.Destroy;
begin
  FConnections.Free;
  inherited;
end;

procedure TMixerConnections.DoFreeItem (Pntr:Pointer);
begin
  TMixerConnection(Pntr).Free;
end;

function TMixerConnections.GetConnection (Ind:Integer):TMixerConnection;
begin
  Result := FConnections.Pointer[Ind];
end;

function TMixerConnections.GetCount:Integer;
begin
  Result := FConnections.Count;
end;

{-----------------}
{TMixerDestination}
{-----------------}

constructor TMixerDestination.Create (AMixer:TAudioMixer; AData:TMixerLine);
begin
  FData := AData;
  XMixer := AMixer;
  FConnections := TMixerConnections.Create (XMixer, FData);
  FControls := TMixerControls.Create (XMixer, AData);
end;

destructor TMixerDestination.Destroy;
begin
  Fcontrols.Free;
  FConnections.Free;
  inherited;
end;

{-----------------}
{TMixerDestinations}
{-----------------}

constructor TMixerDestinations.Create (AMixer:TAudioMixer);
var A,B:Integer;
    ML:TMixerLine;
begin
  FDestinations := TPointerList.Create;
  FDestinations.OnFreeItem := DoFreeItem;
  if (AMixer = nil) then
    Exit;
```

```
  For A := 0 to AMixer.MixerCaps.cDestinations-1 do
  begin
    ML.cbStruct := SizeOf(TMixerLine);
    ML.dwDestination := A;
    B        :=        MixerGetLineInfo        (AMixer.MixerHandle,        @ML,
MIXER_GETLINEINFOF_DESTINATION);
    If B = MMSYSERR_NOERROR then
      FDestinations.Add (Pointer(TMixerDestination.Create (AMixer, ML)));
  end;
end;

procedure TMixerDestinations.DoFreeItem (Pntr:Pointer);
begin
  TMixerDestination(Pntr).Free;
end;

destructor TMixerDestinations.Destroy;
begin
  FDestinations.Free;
  inherited;
end;

function TMixerDestinations.GetDestination (Ind:Integer):TMixerDestination;
begin
  Result := nil;
  If (Assigned (FDestinations)) then
    Result := FDestinations.Pointer[Ind];
end;

function TMixerDestinations.GetCount:Integer;
begin
  Result := FDestinations.Count;
end;

{-----------}
{TAudioMixer}
{-----------}

constructor TAudioMixer.Create (AOwner:TComponent);
begin
  inherited Create (AOwner);
  XWndHandle := AllocateHWnd (MixerCallBack);
  FMixersCount := mixerGetNumDevs;
  FMixerId := -1;
  if (FMixersCount = 0) then
    FDestinations := TMixerDestinations.Create (nil)
  else
  begin
    FDestinations := nil;
    SetMixerId (0);
  end;
end;

destructor TAudioMixer.Destroy;
begin
  CloseMixer;
  if XWndHandle <> 0 then
    DeAllocateHwnd (XWndHandle);
  inherited;
end;

procedure TAudioMixer.CloseMixer;
begin
  If FMixerId >= 0 then
  begin
    mixerClose (FMixerHandle);
    FMixerId := -1;
  end;
  FDestinations.Free;
```

```
  FDestinations := nil;
end;

procedure TAudioMixer.SetMixerId (Value:Integer);
label AllOK;
begin
  If (Value < 0) OR (Value >= FMixersCount) then
    Exit;
  CloseMixer;

  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW OR
MIXER_OBJECTF_MIXER) = MMSYSERR_NOERROR then
    goto AllOK;


  If mixerOpen (@FMixerHandle, Value, XWndHandle, 0, CALLBACK_WINDOW) =
MMSYSERR_NOERROR then
    goto AllOK;
  If mixerOpen (@FMixerHandle, Value, 0, 0, 0) = MMSYSERR_NOERROR then
    goto AllOK;

  // відбулася помилка
  FMixerId := -1;
  FDestinations := TMixerDestinations.Create (nil);

  Exit;

AllOK:
  FMixerId := Value;
  mixerGetDevCaps (MixerId, @FMixerCaps, SizeOf (TMixerCaps));

  if FMixerCaps.wMid = MM_MICROSOFT then
    FManufacturer := 'Microsoft'
  else
    FManufacturer := IntToStr(FMixerCaps.wMid) + ' = Unknown';
  FDriverVersion := FMixerCaps.vDriverVersion;
  FProductId := FMixerCaps.wPid;
  FProductName := StrPas(FMixerCaps.szPName);
  FNumberOfLine := FMixerCaps.cDestinations;

  FDestinations := TMixerDestinations.Create (Self);
end;

procedure TAudioMixer.MixerCallBack (var Msg:TMessage);
begin
  case Msg.Msg of
    MM_MIXM_LINE_CHANGE:
        If Assigned (OnLineChange) then
          OnLineChange (Self, Msg.wParam, Msg.lParam);
    MM_MIXM_CONTROL_CHANGE:
        If Assigned (OnControlChange) then
          OnControlChange (Self, Msg.wParam, Msg.lParam);
    else
      Msg.Result   :=   DefWindowProc   (XWndHandle,   Msg.Msg,   Msg.WParam,
Msg.LParam);
  end;
end;

const MIXER_LONG_NAME_CHARS = 64;

type MIXERCONTROLDETAILS_LISTTEXT = record
       dwParam1:DWORD;
       dwParam2:DWORD;
       szName:Array [0..MIXER_LONG_NAME_CHARS-1] of Char;
     end;

type ListTextArray = array [0..1000] of MIXERCONTROLDETAILS_LISTTEXT;
```

```
function TAudioMixer.GetVolume (ADestination,AConnection:Integer;var LeftVol,
RightVol,     Mute:Integer;var     Stereo,     VolDisabled,     MuteDisabled,
MuteIsSelect:Boolean):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;

begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  Stereo := False;
  MuteDisabled := True;
  VolDisabled := True;
  LeftVol := -1;
  RightVol := -1;
  Mute := -1;
  MuteIsSelect := False;
  MD := Destinations[ADestination];
  MC := nil;
  If AConnection <> -1 then
    MC := MD.Connections[AConnection];

  If MD <> nil then
  begin

    Result := True;

//      If Mute = -1 then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Mute = -1) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
//              If (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MIXER =
MIXERCONTROL_CONTROLTYPE_MIXER) OR
//                  (Cntrl.dwControlType AND MIXERCONTROL_CONTROLTYPE_MUX =
MIXERCONTROL_CONTROLTYPE_MUX) then

//                If (Cntrl.dwControlType AND MIXERCONTROL_CT_CLASS_MASK =
MIXERCONTROL_CT_CLASS_LIST) then

            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                    begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
              MCD.cbDetails := 4;
```

```
              MCD.paDetails := @Details;
              B                    :=                    mixerGetControlDetails
(FMixerHandle,@MCD,MIXER_GETCONTROLDETAILSF_VALUE);
              If B <> MMSYSERR_NOERROR then
              begin
                Inc (A);
                continue;
              end;

              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCDText.cChannels := 1
              else
                MCDText.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCDText.cMultipleItems := Cntrl.cMultipleItems
              else
                MCDText.cMultipleItems := 0;
              GetMem  (ltext,  MCDText.cChannels  *  MCDText.cMultipleItems  *
sizeof (MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              B    :=    mixerGetControlDetails   (FMixerHandle,    @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              If B <> MMSYSERR_NOERROR then
              begin
                FreeMem (ltext);
                Inc (A);
                continue;
              end;
              B := MCD.cChannels - 1;
              while (B < integer(MCD.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, MCD.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (MCD.cMultipleItems)) then
              begin
                Mute := Details[B];
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                MuteIsSelect := True;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;

    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
```

```
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin
      A := 0;
      while ((LeftVol = -1) OR (Mute = -1)) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
          If ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) AND (Mute
= -1))) AND
             (Cntrl.fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE    <>
MIXERCONTROL_CONTROLF_MULTIPLE)
             then
          begin
            if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) then
              MCD.cbStruct := SizeOf(TMixerControlDetails)
            else
              MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @details;
            B     :=     mixerGetControlDetails     (FMixerHandle,     @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
            If B = MMSYSERR_NOERROR then
            begin
              If  (Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_VOLUME)  AND
(LeftVol = -1) then
              begin
                VolDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                LeftVol := details[0];
                If MCD.cChannels > 1 then
                begin
                  RightVol := Details[1];
                  Stereo := True;
                end
                else
                  RightVol := LeftVol;
              end
              else  If  (Cntrl.dwControlType  =  MIXERCONTROL_CONTROLTYPE_MUTE)
AND (Mute = -1) then
              begin
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
              end
// NEW ->
(*              else If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF)
AND (Mute = -1) then
              begin
                MuteDisabled          :=          Cntrl.fdwControl          AND
MIXERCONTROL_CONTROLF_DISABLED > 0;
                If Details[0] <> 0 then
                  Mute := 1
                else
                  Mute := 0;
                MuteIsSelect := True;
              end;*)
```

```
// <- NEW
            end;
          end;
        end;
        Inc (A);
      end;

{      If LeftVol = -1 then
        VolDisabled := True;
      If Mute = -1 then
        MuteDisabled := True;}
    end;
  end;
end;

function  TAudioMixer.SetVolume  (ADestination,  AConnection:Integer;  LeftVol,
RightVol, Mute:Integer):Boolean;
var MD:TMixerDestination;
    MC:TMixerConnection;
    Cntrls:TMixerControls;
    MCD:TMixerControlDetails;
    Cntrl:PMixerControl;
    A,B:Integer;
    ML:TMixerLine;
    details:array [0..100] of Integer;
    VolSet,MuteSet:Boolean;
    ltext:^ListTextArray;
    MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  MC := nil;
  MD := Destinations[ADestination];
  If MD <> nil then
  begin
    If AConnection <> -1 then
      MC := MD.Connections[AConnection];

    VolSet := LeftVol = -1;
    MuteSet := Mute = -1;
    Result := True;

    If not MuteSet then
    begin
      If AConnection <> -1 then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while not MuteSet AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (*(Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR*)
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              MCD.cbStruct := SizeOf(TMixerControlDetails);
              MCD.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                MCD.cChannels := 1
              else
                MCD.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                MCD.cMultipleItems := Cntrl.cMultipleItems
              else
                MCD.cMultipleItems := 0;
```

```
              MCD.cbDetails := 4;
              MCD.paDetails := @Details;
              MuteSet := True;
              mixerGetControlDetails           (FMixerHandle,           @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;

              GetMem  (ltext,  MCD.cChannels  *  MCD.cMultipleItems  *  sizeof
(MIXERCONTROLDETAILS_LISTTEXT));
              MCDText.cbStruct := sizeof (MCDText);
              MCDText.dwControlID := Cntrl.dwControlID;
              MCDText.cChannels := MCD.cChannels;
              MCDText.cMultipleItems := MCD.cMultipleItems;
              MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
              MCDText.paDetails := ltext;
              mixerGetControlDetails         (FMixerHandle,         @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
              B := MCD.cChannels - 1;
              while (B < integer (MCD.cMultipleItems)) do
              begin
                if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                  break;
                Inc (B, MCD.cChannels);
              end;
              FreeMem (ltext);

              If (B < integer (MCD.cMultipleItems)) then
              begin
                Details[B] := Mute;
                mixerSetControlDetails         (FMixerHandle,           @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;
    end;



    If AConnection = -1 then
    begin
      Cntrls := MD.Controls;
      ML := MD.Data;
    end
    else
    begin
      If MC <> nil then
      begin
        Cntrls := MC.Controls;
        ML := MC.Data;
      end
      else
        Cntrls := nil;
    end;
    If Cntrls <> nil then
    begin

      A := 0;
      while (not VolSet OR not MuteSet) AND (A < Cntrls.Count) do
      begin
        Cntrl := Cntrls[A];
        If Cntrl <> nil then
        begin
```

```
          If (((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) AND not
VolSet) OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) AND not
MuteSet) (* NEW -> *) (*OR
              ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF) AND not
MuteSet)*) (* <- NEW *)) AND
             (Cntrl.fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE    <>
MIXERCONTROL_CONTROLF_MULTIPLE)
             then
          begin
            MCD.cbStruct := SizeOf(TMixerControlDetails);
            MCD.dwControlID := Cntrl.dwControlID;
            If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
              MCD.cChannels := 1
            else
              MCD.cChannels := ML.cChannels;
            MCD.cMultipleItems := 0;
            MCD.cbDetails := SizeOf(Integer);
            MCD.paDetails := @Details;
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_VOLUME) then
            begin
              Details[0] := LeftVol;
              If RightVol = -1 then
                Details[1] := LeftVol
              else
                Details[1] := RightVol;
              VolSet := True;
            end
            else if ((Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUTE) (*
NEW -> *) (* OR
                     (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_ONOFF) *)
(* <- NEW *)) then
            begin
              For B := 0 to MCD.cChannels - 1 do
                Details[B] := Mute;
              MuteSet := True;
            end;
            mixerSetControlDetails            (FMixerHandle,            @MCD,
MIXER_GETCONTROLDETAILSF_VALUE);
          end;
        end;
        Inc (A);
      end;

    end;
  end;
end;

function TAudioMixer.GetMute(ADestination, AConnection: Integer; var Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
```

```
  MC := nil;
  Mute := False;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 1;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;

    if      (mixerGetLineControls(FMixerHandle,     @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
      with mcdMixerDataMute do
      begin
        cbStruct := SizeOf(TMIXERCONTROLDETAILS);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if        mixerGetControlDetails(FMixerHandle,        @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR then
      begin
        Mute := pmcdsMixerDataUnsignedMute^.fValue = 1;
        Result := True;
      end;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
              mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
                mcdMixerDataMute.cChannels := 1
              else
```

```
              mcdMixerDataMute.cChannels := ML.cChannels;
            If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
              mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
            else
              mcdMixerDataMute.cMultipleItems := 0;
            mcdMixerDataMute.cbDetails := 4;
            mcdMixerDataMute.paDetails := @Details;
            mixerGetControlDetails     (FMixerHandle,     @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE);

            GetMem        (ltext,        mcdMixerDataMute.cChannels        *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
            MCDText.cbStruct := sizeof (MCDText);
            MCDText.dwControlID := Cntrl.dwControlID;
            MCDText.cChannels := mcdMixerDataMute.cChannels;
            MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
            MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
            MCDText.paDetails := ltext;
            mixerGetControlDetails          (FMixerHandle,          @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
            B := mcdMixerDataMute.cChannels - 1;
            while (B < integer (mcdMixerDataMute.cMultipleItems)) do
            begin
              if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                break;
              Inc (B, mcdMixerDataMute.cChannels);
            end;
            FreeMem (ltext);

            If (B < integer (mcdMixerDataMute.cMultipleItems)) then
            begin
              Result := True;
              Mute := Details[B] <> 0;
              break;
            end;
          end;
          Inc (A);
        end;
      end;
    end;

    FreeMem(pmcdsMixerDataUnsignedMute);
    FreeMem(pmcMixerControlMute);
  end;
end;

function  TAudioMixer.SetMute(ADestination,  AConnection:  Integer;  Mute:
Boolean):Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mlcMixerLineControlsMute : TMIXERLINECONTROLS;
  mcdMixerDataMute : TMIXERCONTROLDETAILS;
  pmcMixerControlMute : PMIXERCONTROL;
  pmcdsMixerDataUnsignedMute : PMIXERCONTROLDETAILSBOOLEAN;
  mlMixerLine : TMixerLine;
  Cntrl:PMixerControl;
  Cntrls:TMixerControls;
  ML:TMixerLine;
  A,B:Integer;
  details:array [0..100] of Integer;
  ltext:^ListTextArray;
  MCDText:TMixerControlDetails;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
```

```
  MC := nil;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
      mlMixerLine := MD.Data
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
        mlMixerLine := MC.Data
      else
        Exit;
    end;

    GetMem(pmcMixerControlMute, SizeOf(TMIXERCONTROL));
    GetMem(pmcdsMixerDataUnsignedMute, SizeOf(TMIXERCONTROLDETAILSBOOLEAN));

    with mlcMixerLineControlsMute do
    begin
      cbStruct := SizeOf(TMIXERLINECONTROLS);
      dwLineID := mlMixerLine.dwLineID;
      dwControlType := MIXERCONTROL_CONTROLTYPE_MUTE;
      cControls := 0;
      cbmxctrl := SizeOf(TMIXERCONTROL);
      pamxctrl := pmcMixerControlMute;
    end;

    if      (mixerGetLineControls(FMixerHandle,      @mlcMixerLineControlsMute,
MIXER_GETLINECONTROLSF_ONEBYTYPE) = MMSYSERR_NOERROR) then
    begin
      with mcdMixerDataMute do
      begin
        cbStruct := SizeOf(TMixerControlDetails);
        dwControlID := pmcMixerControlMute^.dwControlID;
        cChannels := 1;
        cMultipleItems := 0;
        cbDetails := SizeOf(TMIXERCONTROLDETAILSBOOLEAN);
        paDetails := pmcdsMixerDataUnsignedMute;
      end;

      if Mute then
        pmcdsMixerDataUnsignedMute^.fValue := 1
      else
        pmcdsMixerDataUnsignedMute^.fValue := 0;

      if
(mixerSetControlDetails(FMixerHandle,@mcdMixerDataMute,MIXER_SETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
        Result := True;
    end
    else
    begin
      If (AConnection <> -1) then
      begin
        Cntrls := MD.Controls;
        ML := MD.Data;
        If (MC <> nil) AND (Cntrls <> nil) then
        begin
          A := 0;
          while (Result = False) AND (A < Cntrls.Count) do
          begin
            Cntrl := Cntrls[A];
            If (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MIXER) OR
               (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
            begin
              mcdMixerDataMute.cbStruct := SizeOf(TMixerControlDetails);
              mcdMixerDataMute.dwControlID := Cntrl.dwControlID;
              If Cntrl.fdwControl AND MIXERCONTROL_CONTROLF_UNIFORM > 0 then
```

```
                mcdMixerDataMute.cChannels := 1
              else
                mcdMixerDataMute.cChannels := ML.cChannels;
              If   Cntrl.fdwControl   AND   MIXERCONTROL_CONTROLF_MULTIPLE   =
MIXERCONTROL_CONTROLF_MULTIPLE then
                mcdMixerDataMute.cMultipleItems := Cntrl.cMultipleItems
              else
                mcdMixerDataMute.cMultipleItems := 0;
              mcdMixerDataMute.cbDetails := 4;
              mcdMixerDataMute.paDetails := @Details;
              if   (mixerGetControlDetails   (FMixerHandle,   @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) <> MMSYSERR_NOERROR) then
              begin
                Inc (A);
                continue;
              end;
              if (Cntrl.dwControlType = MIXERCONTROL_CONTROLTYPE_MUX) then
                For B := 0 to Cntrl.cMultipleItems-1 do
                  Details[B] := 0;
              If Mute then
              begin
                GetMem       (ltext,       mcdMixerDataMute.cChannels       *
mcdMixerDataMute.cMultipleItems * sizeof (MIXERCONTROLDETAILS_LISTTEXT));
                MCDText.cbStruct := sizeof (MCDText);
                MCDText.dwControlID := Cntrl.dwControlID;
                MCDText.cChannels := mcdMixerDataMute.cChannels;
                MCDText.cMultipleItems := mcdMixerDataMute.cMultipleItems;
                MCDText.cbDetails := sizeof (MIXERCONTROLDETAILS_LISTTEXT);
                MCDText.paDetails := ltext;
                mixerGetControlDetails       (FMixerHandle,       @MCDText,
MIXER_GETCONTROLDETAILSF_LISTTEXT);
                B := mcdMixerDataMute.cChannels - 1;
                while (B < integer (mcdMixerDataMute.cMultipleItems)) do
                begin
                  if (ltext[B].dwParam1 = MC.Data.dwLineID) then
                    break;
                  Inc (B, mcdMixerDataMute.cChannels);
                end;
                FreeMem (ltext);

                If (B < integer (mcdMixerDataMute.cMultipleItems)) then
                  Details[B] := 1;
              end;
              if   (mixerSetControlDetails   (FMixerHandle,   @mcdMixerDataMute,
MIXER_GETCONTROLDETAILSF_VALUE) = MMSYSERR_NOERROR) then
              begin
                Result := True;
                break;
              end;
            end;
            Inc (A);
          end;
        end;
      end;

    FreeMem(pmcdsMixerDataUnsignedMute);
    FreeMem(pmcMixerControlMute);
  end;
end;

function TAudioMixer.GetPeak (ADestination, AConnection:Integer; var LeftPeak,
RightPeak:Integer): Boolean;
var
  MD : TMixerDestination;
  MC : TMixerConnection;
  mcdMixerDataPeak : TMIXERCONTROLDETAILS;
  pmcMixerControlPeak : PMIXERCONTROL;
{  pmcdsMixerDataSignedPeak : PMIXERCONTROLDETAILSSIGNED;}
```

```
  mlMixerLine : TMixerLine;
  A:Integer;
  Cntrls:TMixerControls;
  Details:Array [1..100] of Integer;
begin
  Result := False;
  If (not Assigned (FDestinations)) then
    Exit;
  LeftPeak := 0;
  RightPeak := 0;
  MD := Destinations[ADestination];
  if MD <> nil then
  begin
    if AConnection = -1 then
    begin
      mlMixerLine := MD.Data;
      Cntrls := MD.Controls;
    end
    else
    begin
      MC := MD.Connections[AConnection];
      if MC <> nil then
      begin
        mlMixerLine := MC.Data;
        Cntrls := MC.Controls;
      end
      else
        Exit;
    end;
    GetMem(pmcMixerControlPeak, SizeOf(TMIXERCONTROL));

    A := 0;
    while (A < Cntrls.Count) do
    begin
      If    (Cntrls[A].dwControlType    AND    MIXERCONTROL_CT_CLASS_MASK)    =
MIXERCONTROL_CT_CLASS_METER then
        break;
      Inc (A);
    end;
    If A = Cntrls.Count then
    begin
      FreeMem(pmcMixerControlPeak);
      Exit;
    end;

    with mcdMixerDataPeak do
    begin
      cbStruct := SizeOf(TMIXERCONTROLDETAILS);
      dwControlID := Cntrls[A].dwControlID;
      cChannels := mlMixerLine.cChannels;
      If    (Cntrls[A].fdwControl    AND    MIXERCONTROL_CONTROLF_MULTIPLE)    =
MIXERCONTROL_CONTROLF_MULTIPLE then
        cMultipleItems:=Cntrls[A].cMultipleItems
      else
        cMultipleItems:=0;
      cbDetails := SizeOf(TMIXERCONTROLDETAILSSIGNED);
      paDetails := @Details;
    end;
    if
(mixerGetControlDetails(FMixerHandle,@mcdMixerDataPeak,MIXER_GETCONTROLDETAILS
F_VALUE) = MMSYSERR_NOERROR) then
    begin
      LeftPeak := Details[1];
      if mlMixerLine.cChannels = 2 then
        RightPeak := Details[2]
      else
        RightPeak := LeftPeak;
      Result := True;
    end;
```

```
    FreeMem(pmcMixerControlPeak);
   end;
end;
procedure Register;
begin
  RegisterComponents('Samples', [TAudioMixer]);
end;
end.
```