МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ,
ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ ТА РАДІОЕЛЕКТРОННИХ СИСТЕМ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Одарченко Р.С.
“_____”_____2021 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Тема: Інформаційно-комунікаційне забезпечення концепції «Розумна лікарня»

Виконавець: _____ Андреєва А. І.
(підпис)

Керівник: _____ Соловйов Д. О.
(підпис)

Нормоконтролер: _____ Бахтіяров Д. І.
(підпис)

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра телекомунікаційних та радіоелектронних систем
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Одарченко Р.С.
“_____”_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

Андреєвої Аліни Ігорівни
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту): Інформаційно-комунікаційне забезпечення концепції «Розумна лікарня»
затверджена наказом ректора від «06» квітня 2021 р. №559/ст
2. Термін виконання роботи: з 17.05.2021 р. по 20.06.2021 р.
3. Вихідні дані до роботи: дослідження та проєктування інформаційно-комунікаційного забезпечення концепції «Розумна лікарня»
4. Зміст пояснювальної записки: Залучення інструментів інформаційних систем та аналітики, формування поняття Е-медицини, Створення ситуаційного диспетчерського центру розумної лікарні з використанням новітніх телекомунікаційних інструментів в умовах урбаністики: Big Data, IP-телефонія та розпізнавання мови, Види атак на IP-АТС Asterisk та методи протистояння їм.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Концептуальна архітектура інформаційно-аналітичної системи Е-медицина, Схема налаштування моніторингу, Схема роботи Corezoid.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів диплому	17.05.2021-25.05.2021	Виконано
2	Вступ	26.05.2021-27.05.2021	Виконано
3	Залучення інструментів інформаційних систем та аналітики, формування поняття Е-медицини	28.05.2021-30.05.2021	Виконано
4	Створення ситуаційного диспетчерського центру розумної лікарні з використанням новітніх телекомунікаційних інструментів в умовах урбаністики: Big Data, IP-телефонія та розпізнавання мови	31.05.2021-02.06.2021	Виконано
5	Види атак на IP-АТС Asterisk та методи протистояння їм	03.06.2021-04.06.2021	Виконано
6	Усунення недоліків дипломної роботи	05.06.2021-20.06.2021	Виконано

7. Дата видачі завдання: “26" квітня 2021 р.

Керівник дипломної роботи _____ Соловйов Д. О.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Андреєва А. І.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Дипломна робота «Інформаційно-комунікаційне забезпечення концепції «Розумна лікарня»» містить 51 сторінку, 26 рисунків, 13 використаних джерел.

Ключові слова: Е-медицина, IP-телефонія, ситуаційний центр.

Об’єкт дослідження – бізнес-процеси лікарні за умови штатних та екстрених ситуацій.

Предмет дослідження – ситуаційний центр лікарні та об’єкти, що можуть перебувати під його контролем, телекомунікаційні системи, що мають вплив на інфраструктурні об'єкти міста.

Мета дипломної роботи – дослідження варіантів вдосконалення та імплементації інформаційних систем на прикладі розумної лікарні.

Методи дослідження – теоретичні: аналіз, синтез, пояснення.

Матеріали дипломної роботи рекомендується використовувати при оптимізації процесу прийому звернень від пацієнтів, обробки та збереження інформації про пацієнтів, наявність препаратів та їх обіг.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

IP (англ. Internet Protocol) – протокол мережевого рівня.

API (англ. Application Programming Interface) – інтерфейс програмування додатків.

DDoS (англ. Distributed Denial of Service) – відмова в обслуговуванні.

FXS (англ. Foreign Exchange Station) – голосовий інтерфейс або порт, що емулює розширення інтерфейсу АТС.

HTTP (англ. HyperText Transfer Protocol) – протокол прикладного рівня.

Hz (герц) – одиниця вимірювання частоти.

ICMP (англ. Internet Control Message Protocol) – міжмережевий протокол керуючих повідомлень.

MFCC (Mel-frequency cepstral coefficients) – Mel-частотні кепстральні коефіцієнти.

SIP (англ. Session Initiation Protocol) – протокол передачі даних.

SQL (англ. Structured Query Language) – мова структурованих запитів.

UCE (англ. Universal Computing Element) – універсальний обчислювальний елемент.

VPN (англ. Virtual Private Network) – віртуальна приватна мережа.

АТС (автоматична телефонна станція) – телефонна станція, що забезпечує автоматичне з'єднання абонентів телефонної мережі.

США (Сполучені Штати Америки) – республіка в Північній Америці.

ВСТУП

Актуальність теми. Сучасна медицина та діагностика захворювань, роботизація медичних приладів надають змогу виконувати якісне та ефективне втручання в організм людини, з вірогідною ймовірністю не завдати ускладнень при плановому лікуванні або обстеженні та врятувати життя у критичних ситуаціях.

Проте, автоматизація лікарень державного та приватного сектору потребує більш радикальних змін у напрямках:

- Не ефективний складський облік препаратів: в це поняття входить відсутність управління ресурсом, так як безліч швидкопсувних препаратів можуть лежати не задіяними в одній лікарні і закуповуватися через брак для іншої лікарні, в обліковій системі не визначаються ліки необхідні на закупівлю, їх беруть загальною стандартною кількістю, що призводить до додаткових витрат.
- Затримка надання невідкладної допомоги збільшує витрату коштів на відновлення пацієнта. При несвоєчасному наданні медичної допомоги витрати на відновлення здоров'я пацієнта можуть виявитися в рази більше. Так несвоєчасне визначення діагнозу або затримка в транспортуванні каретою швидкої допомоги можуть не тільки погіршити стан, а й викликати додаткові витрати на медикаменти, утримання хворого в палаті або покриття страхування в повному обсязі, викликане смертю.

Мета і завдання дослідження.

Мета дипломної роботи – дослідження варіантів вдосконалення та імплементації інформаційних систем на прикладі розумної лікарні.

Для досягнення поставленої мети вирішуються наступні наукові завдання:

1. Дослідження залучення інструментів інформаційних систем та аналітики, формування поняття Е-медицини.

2. Створення ситуаційного диспетчерського центру розумної лікарні з використанням новітніх телекомунікаційних інструментів в умовах урбаністики: Big Data, IP-телефонія та розпізнавання мови, віртуальна реальність, алгоритм “Зеленої хвилі”.

3. Дослідження видів атак інформаційно-аналітичних систем та методів протистояння їм на прикладі IP-телефонії Asterisk.

Об'єктом дослідження є бізнес-процеси лікарні за умови штатних та екстрених ситуацій.

Предметом дослідження є ситуаційний центр лікарні та об'єкти, що можуть перебувати під його контролем, телекомунікаційні системи, що мають вплив на інфраструктурні об'єкти міста.

Методи досліджень. В роботі використовували теоретичні методи досліджень: аналіз, синтез, пояснення.

Практичне значення отриманих результатів.

Матеріали роботи рекомендується використовувати для оптимізації процесу прийому звернень від пацієнтів, обробки та збереження інформації про пацієнтів, наявність препаратів та їх обіг.

Апробація отриманих результатів. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- Науково-практична конференція «Проблеми експлуатації та захисту інформаційно-комунікаційних систем», м. Київ, 2021 р.

РОЗДІЛ 1
ЗАЛУЧЕННЯ ІНСТРУМЕНТІВ ІНФОРМАЦІЙНИХ СИСТЕМ ТА АНАЛІТИКИ, ФОРМУВАННЯ ПОНЯТТЯ Е-МЕДИЦИНИ

Концепція розумної лікарні, перш за все несе у собі мету створення та залучення телекомунікаційних інструментів, їх імплементацію у бізнес-процеси працівників чи користувачів для:

- економії ресурсів (час обробки запитів працівників, економічної складової лікарні);
- оперативного вирішення екстрених ситуацій з підвищенням вірогідності порятунку життя користувачів внаслідок автоматизації та оптимізації процесів, можливості залучення новітніх технологій з коригуванням позитивного досвіду користувача.

Інформаційно-аналітичні системи – системи збору та опрацювання інформації, що дозволяють виконувати ряд важливих функцій:

- збір даних та вивід даних у аналітичні дашборди;
- напівавтоматизоване введення даних при опрацюванні звернень;
- реалізація координаційних додатків, що дозволяють розміщувати у собі оперативну різнорідну медіа інформацію у режимі історичних даних, аналітики, онлайн даних тощо;

Е-медицина – електронна система збору інформації з державних та приватних реєстрів, що дозволить консолідувати у собі необхідну інформацію про об’єкти та суб’єкти, що напряму або дотично відносяться до бізнес-процесів лікарні.

На рис. 1.1 зображено концептуальну архітектуру пропонованої інформаційно-аналітичної системи Е-медицина:

Рис. 1.1. Концептуальна архітектура інформаційно-аналітичної системи Е-медицина

Таким чином система дозволить:

- проводити облік та аудит препаратів, та матеріалів, їх кількість та термін придатності;
- координаторам приймати рішення про державні закупівлі необхідних матеріалів, спираючись на аналітичні дані та підказки системи;
- виконувати ефективні маніпуляції з пацієнтами на виїзді оперативної бригади завдяки медичним карткам онлайн, так як буде відома історія хвороб пацієнта;
- мати єдине вікно координатора, що консолідуватиме у собі контент, дозволить приймати доречні рішення у позаштатних та екстрених ситуаціях.

1.1. Інформаційно-аналітична система Е-медицина. Модуль інфраструктура

Інфраструктура – модуль системи, що дозволяє інтегрувати та організувати збір різнорідних медіа даних до системи та забезпечити безперебійний, захищений процес опрацювання і використання даних.

1.1.1. Моніторинг та управління доступністю

Моніторинг та управління доступністю пропоновано до реалізації за допомогою Zabbix – відкрита система моніторингу та відстеження статусів сервісів комп'ютерної мережі, мережевого обладнання та серверів.

Для збереження даних використовується MySQL, PostgreSQL, SQLite або Oracle Database, веб-інтерфейс написаний на PHP (англ. Hypertext Preprocessor – мова програмування). Підтримує декілька видів моніторингу:

- Simple checks – перевіряє реакцію і доступність стандартних сервісів (наприклад, SMTP (англ. Simple Mail Transfer Protocol – мережевий протокол) або HTTP) без встановлення програмного забезпечення на хості, що спостерігається.
- Zabbix agent – встановлюється на Unix-подібні або Windows-хости для отримання даних про використання мережі, дискового простору, навантаження процесора і т. д.
- External check – виконання зовнішніх програм, також підтримується моніторинг через SNMP (англ. Simple Network Management Protocol – протокол управління).

З точки зору користувача Zabbix ділиться на дві великі частини: сервер і агенти. Сервер розташовується на одній машині, яка збирає і зберігає статистичні дані, а агенти - на тих машинах, дані з яких збираються [1]:

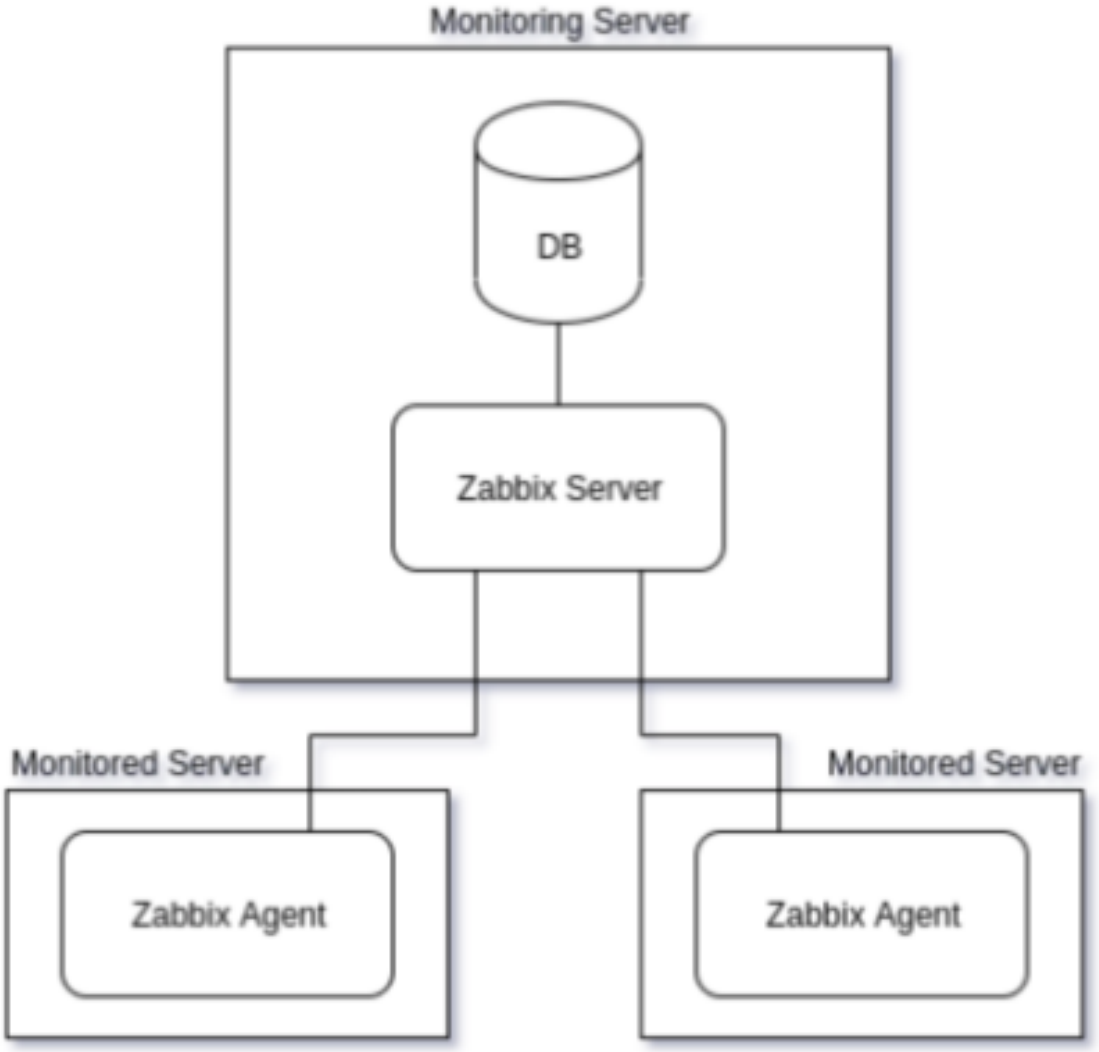

Рис. 1.2. Схема налаштування моніторингу

1.1.2. Збір контента

Внаслідок відсутності єдиного стандарту передачі контенту до інформаційних систем, при реалізації необхідно використати інструмент, що матиме можливість приєднатися до будь-якої точки доступу, отримувати та координувати інформацію за допомогою різнорідних API (DB (англ. DataBase – база даних), JSON (англ. JavaScript Object Notation – текстовий формат обміну даними між комп'ютерами), тощо).

Внаслідок виявлення проблематики до реалізації пропоновано інструмент інтелектуальної побудови бізнес-процесів – Corezoid.

Corezoid – хмарна ОС для створення IT-рішень (IT – інформаційні технології) з використанням методів автоматичного програмування з явним виділенням станів.

Corezoid дозволяє створювати і виконувати алгоритми і процеси, а також керувати технікою і програмним забезпеченням будь-якої складності.

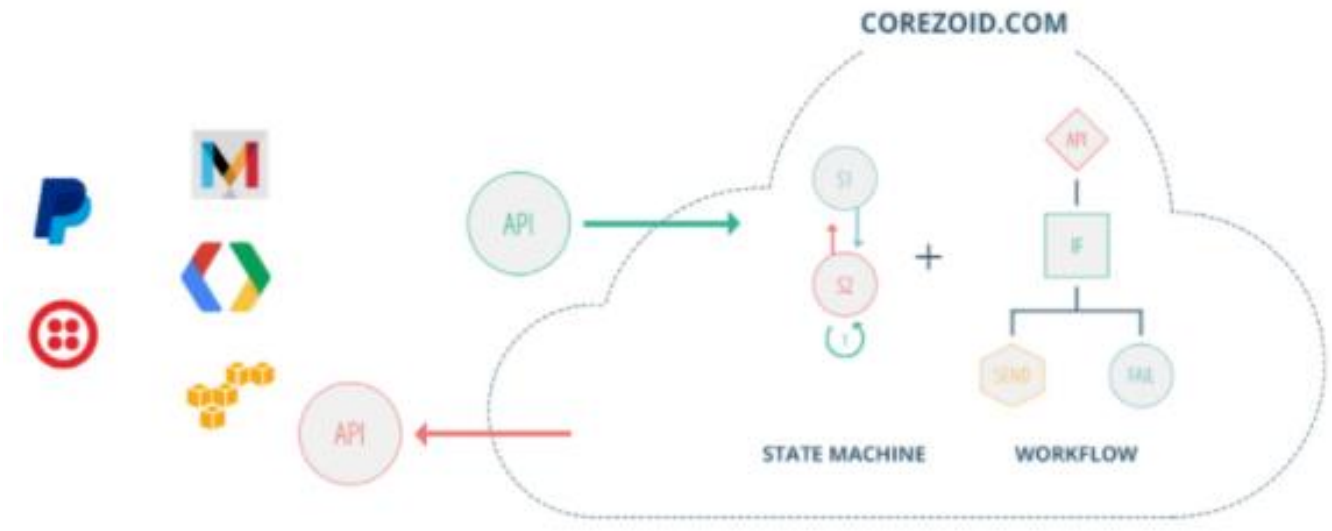

Рис. 1.3. Схема роботи Corezoid

Кожен Corezoid процес складається з вузлів. Кожен вузол містить свою унікальну логіку. Заявки рухаються по процесу від вузла до вузла, слідуючи заданій логіці (рис. 1.4).

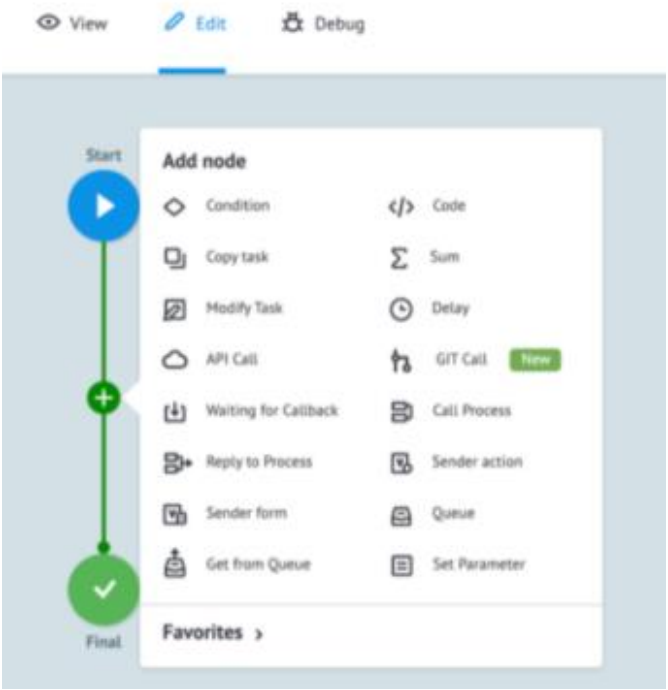

Рис. 1.4. Задання логіки процесу

Процес – алгоритм або програма, реалізована за допомогою послідовності, набору або безлічі вузлів (рис. 1.5).

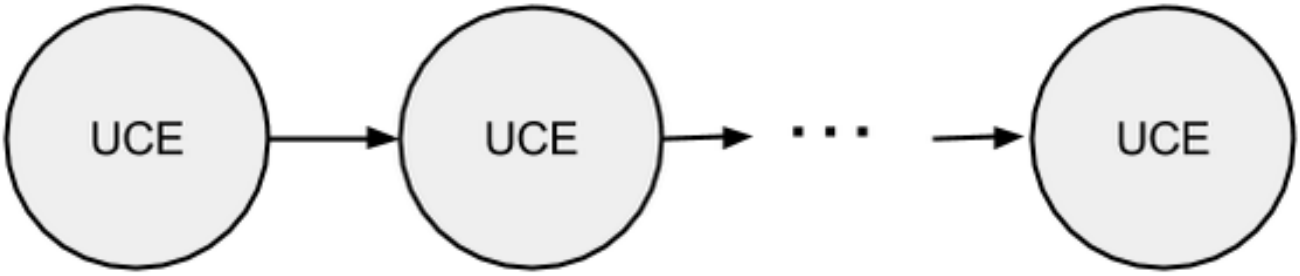

Рис. 1.5. Схема процесу

Вузол – універсальний обчислювальний елемент, який описує стан об'єктів (рис. 1.6). Вузол має наступні характеристики: логіка, функції, лічильники.

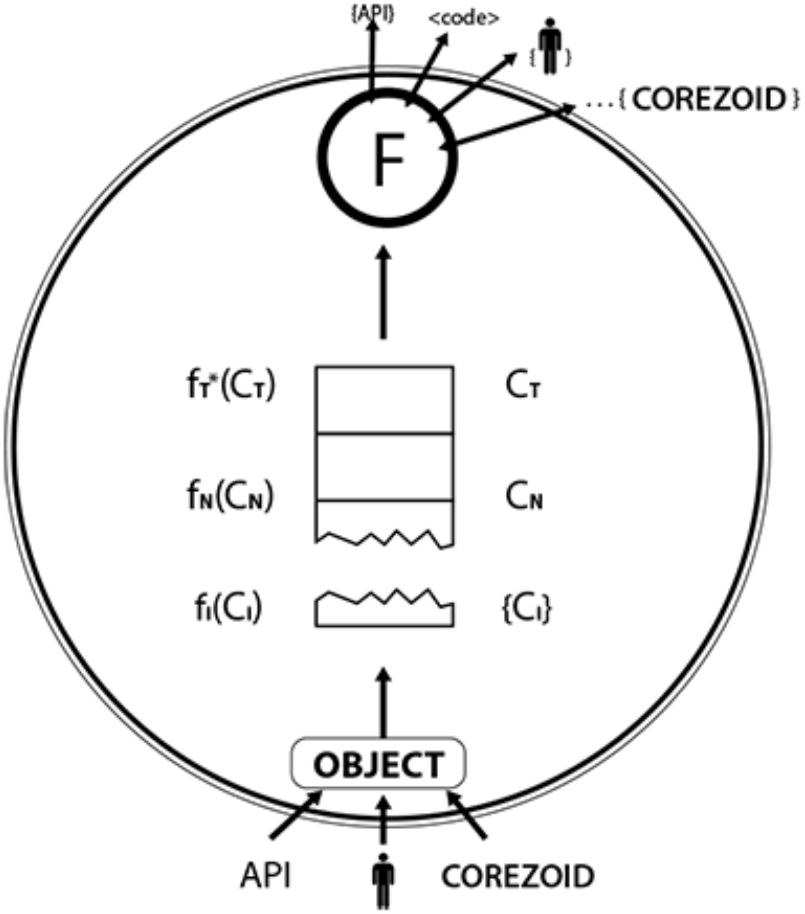

Рис. 1.6. Схема вузла

f – виклик відповідної функції;

F – кожному вузлу ставиться у відповідність функція, яка може бути реалізована через людину, API, код, інший вузол;

Ст – лічильник, час T найстарішого об'єкта в черзі;

Cn – кількість об'єктів у черзі;

{Ci} – кастомні лічильники.

Черга – черга з об'єктів у вузлі.

Об'єкт – набір параметрів, що характеризують об'єкт.

Відповідно до визначених правил у вузлі проводиться обробка інформації про об'єкт – застосовується функція об'єкта.

Функції – дії, які необхідно виконати над об'єктом з черги у вузлі. Функції можуть бути у вигляді API, програмного коду, іншого процесу і т.д.

Логіка – інструменти управління логікою у вузлі. Бувають системні і призначені для користувача.

Системна логіка T – заданий час, протягом якого може перебувати об'єкт у вузлі.

Системна логіка N – задана кількість об'єктів, які можуть перебувати у черзі у вузлі.

Лічильники – показують тільки значення без будь-яких дій (в цьому відмінність від логіки). Також бувають системними і призначеними для користувача [2].

Таким чином, даний інструмент повністю забезпечує втілення концептуальної ідеї збору та використання інформації у системі.

1.2. Інформаційно-аналітична система Е-медицина. Модуль Бекенд

Програмне забезпечення Інформаційно-аналітичної системи призначене для контроля та координації здійснення заходів із надання допомоги та опрацювання тривожних сигналів, що надходять до пункту централізованого спостереження та забезпечує функціонування з даними характеристиками:

- Можливість забезпечення в цілодобовому режимі моніторинг, прийом, реєстрацію та обробку тривожних сигналів, що надходять до пункту централізованого спостереження;
- Можливість введення в базу даних, накопичення та надійне зберігання тривожних сигналів, що надходять до пункту централізованого спостереження;
- Можливість ідентифікації та пошуку клієнтів, в залежності від визначених ознак, перегляду поточного стану в медичній картці;
- Перегляд статистичної інформації про ресурси лікарні та підказки стосовно об’ємів та типів ліків та апаратів до закупівлі.

Загальний високорівневий процес реагування на звернення:

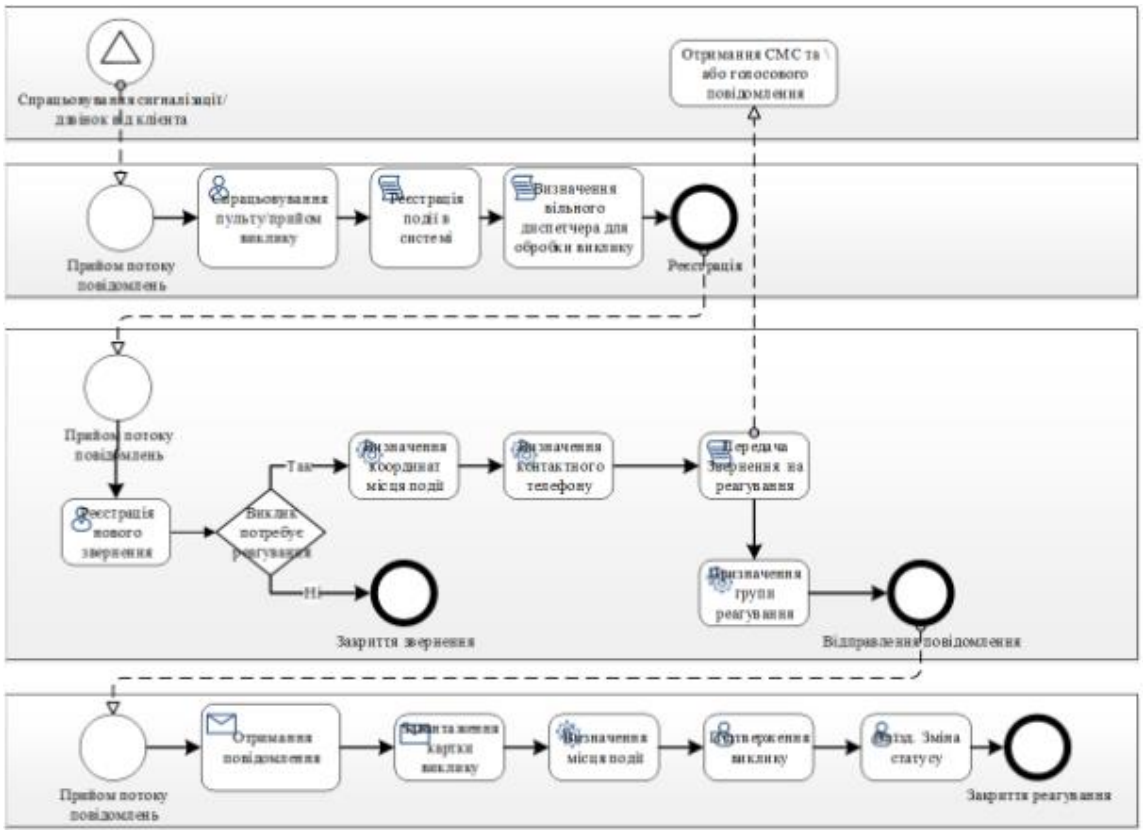

Рис. 1.7. Загальний процес реагування на звернення

Реалізація електронної взаємодії дозволяє підтримувати довідники об’єктів в актуальному стані. Крім того, всі нові об’єкти реєструються та ведуться з прив’язкою до об’єктів, що оцифровані та ведуться в системі управління в реєстрах інших лікарень та державних підприємств.

РОЗДІЛ 2
СТВОРЕННЯ СИТУАЦІЙНОГО ДИСПЕТЧЕРСЬКОГО ЦЕНТРУ РОЗУМНОЇ ЛІКАРНІ З ВИКОРИСТАННЯМ НОВІТНІХ ТЕЛЕКОМУНІКАЦІЙНИХ ІНСТРУМЕНТІВ В УМОВАХ УРБАНІСТИКИ: BIG DATA, IP-ТЕЛЕФОНІЯ ТА РОЗПІЗНАВАННЯ МОВИ

Ситуаційний диспетчерский центр відіграє неабияку важливу роль у концептуальній системі розумної лікарні. Програмно-апаратний комплекс дозволяє побудувати систему, що надасть змогу приймати ефективні рішення у позаштатних та надзвичайних ситуаціях.

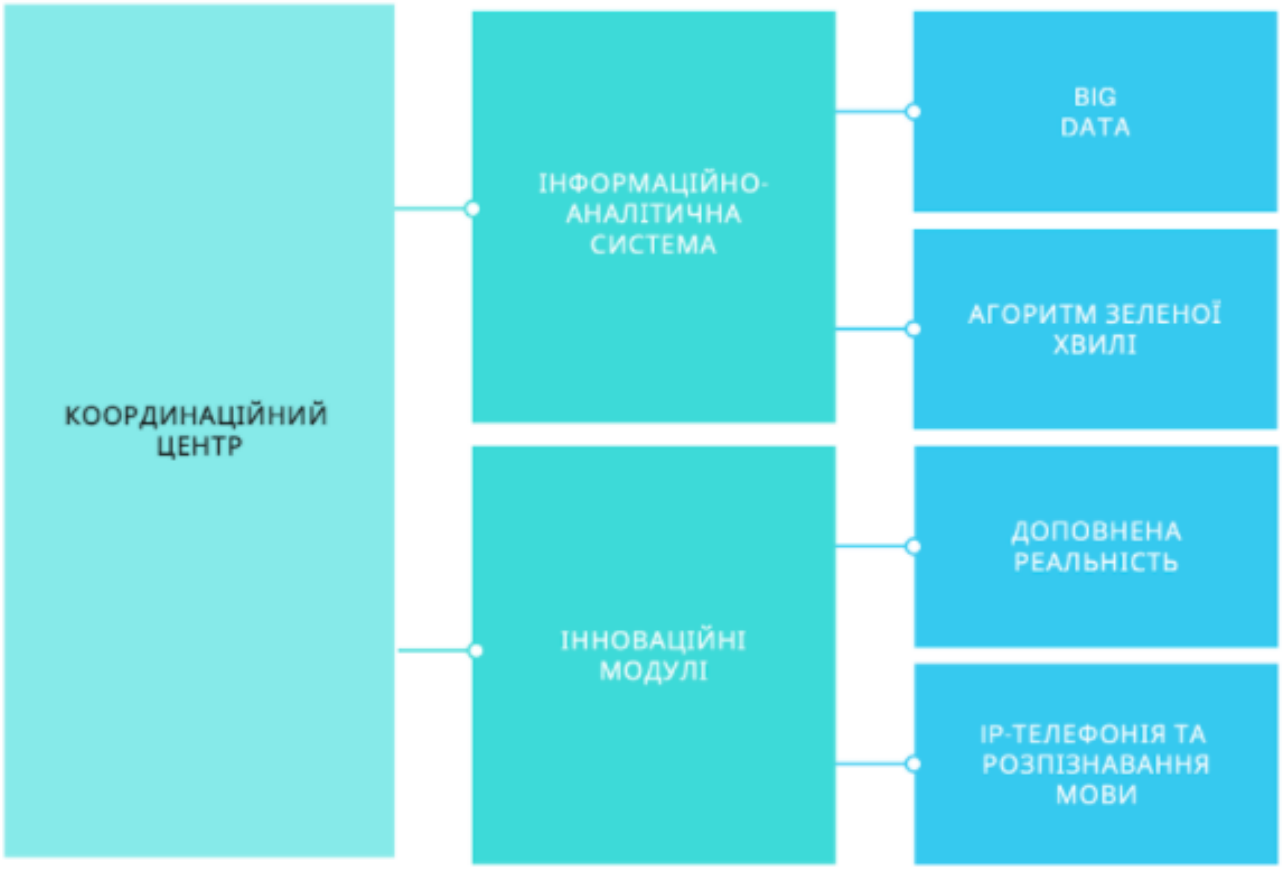

Рис. 2.1. Загальна схема координаційного центру

2.1. Big Data

Великі дані – серія інструментів, підходів і методів обробки неструктурованих і структурованих даних великих обсягів і різного типу для отримання результатів, які будуть легше сприйматися людиною. Методи обробки є ефективними в умовах безперервного приросту і розподілу по численним вузлам обчислювальної мережі, що сформувалися в кінці 2000-х років, і альтернативою традиційним системам управління базами даних і рішень класу Business Intelligence [3].

Щоб отримати робочу гіпотезу про причини виникнення конкретних ситуацій, зокрема, як пов'язані відмови устаткування з умовами подачі напруги, або спрогнозувати майбутнє, наприклад, ймовірність своєчасного прибуття карети швидкої допомоги, аналіз великих обсягів структурованої і неструктурованої інформації виконується в кілька етапів:

- чистка даних (data cleaning) – пошук і виправлення помилок в первинному наборі інформації, наприклад, помилки ручного введення, некоректні значення з вимірювальних приладів через короткочасні збої і т.д .;
- генерація предикторів (feature engineering) – змінних для побудови аналітичних моделей, наприклад, освіта, стаж роботи, стать і вік потенційного позичальника;
- побудова і навчання аналітичної моделі (model selection) для передбачення цільової (таргетної) змінної. Так перевіряються гіпотези про залежність таргетної змінної від предикторів. Наприклад, скільки днів становить прострочення по кредиту для позичальника з середньою освітою і стажем роботи менше 3-х місяців.

Зараз аналітика великих даних використовується в більш ніж 50% компаній по всьому світу. При тому, що в 2015 році цей показник становив усього лише 17%. Big Data найактивніше використовується компаніями, які працюють в сфері телекомунікацій і фінансових послуг. Потім йдуть компанії, які спеціалізуються на технологіях в охороні здоров'я. Мінімальне використання аналітики Big Data в освітніх компаніях:

в більшості випадків представники цієї сфери заявляли про намір використовувати технології в найближчому майбутньому.

У США аналітика Big Data використовується найбільш активно: понад 55% компаній з різних сфер працюють з цією технологією. У Європі та Азії затребуваність аналітики великих даних набагато нижче - близько 53%.

Для прикладу, у США сепсис займає 10 місце в рейтингу причин смерті серед хвороб. Щорічно сепсис виникає приблизно у 1 млн американців, від 28% до 50% з них помирає. На лікування сепсису щорічно витрачається близько 20 млрд $.

При цьому основна причина смертельних випадків – недостатній лікарський контроль. Пацієнти виписуються з госпіталю або отримують першу допомогу, і після цього за ними не спостерігають. Однак після цього великий ризик розвитку сепсису, симптоми якого – жар, озноб, прискорені дихання і пульс, висип, розгубленість і дезорієнтація – схожі на симптоми інших поширених захворювань. Часто пацієнти надто пізно звертаються до лікаря або захворювання не вдається правильно діагностувати на ранніх стадіях. В результаті швидко розвивається септичний шок і відбувається часто необоротна поразка безлічі органів.

Для контролю стану пацієнтів пропонується використовувати сертифікований пристрій HealthPatch від компанії Vital Connect, який буде збирати основні показники стану пацієнтів, включаючи навіть пози і рух (при сепсисі вони змінюються). Далі інформація надходить на сервери ClearStory Data, де об'єднується з іншими медичними даними про пацієнтів і аналізується в реальному часі за допомогою рішення на базі Apache Spark. У перспективі такі пристрої будуть отримувати всі пацієнти, які вийшли із госпіталів і отримали першу допомогу, за якими може слідувати сепсис. Подібна система, але з меншим рівнем аналізу даних, вже успішно реалізована в Сінгапурі [4].

Результат: створено рішення, яке дозволить системі охорони здоров'я США значно знизити смертність від сепсису (загального зараження крові).

Таким чином, використання технології у концепції розумної лікарні є невід'ємною частиною її реалізації.

2.2. IP-телефонія та розпізнавання мови

Використання IP-телефонії у контактних центрах наразі є традиційним способом отримання якісних телекомунікаційних послуг, проте, вона відіграє важливу роль у залученні інтелектуальних систем, таких як розпізнавання та аналіз мови.

Перш за все, наша мова – це послідовність звуків, а звук – суперпозиція (накладення) звукових коливань різних частот. Хвиля (звукове коливання) характеризується двома атрибутами – амплітудою і частотою.

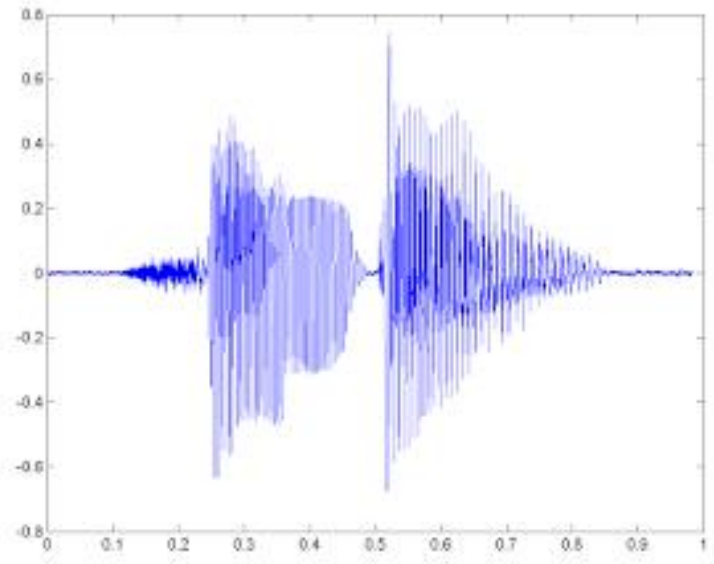

Рис. 2.2. Амплітудно-часова характеристика людської мови

Для того, щоб зберегти на цифровому носії звуковий сигнал, потрібно розбити його на велику кількість проміжків і визначити певне «усереднене» значення на кожному з них.

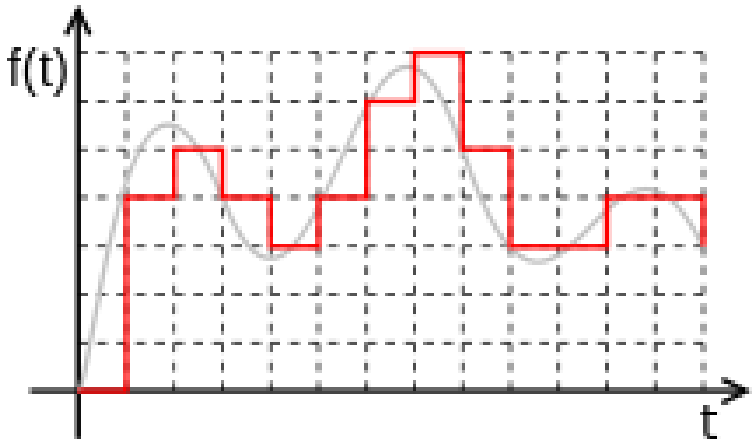

Рис. 2.3. Частотно-часова характеристика людської мови

Таким чином, механічні коливання перетворюються в набір чисел, який в свою чергу є придатним для обробки на сучасних електронно-обчислювальних машинах.

Тобто, розпізнавання мови зводиться до «порівняння» чисельних значень (цифрового сигналу) і слів з словника (української мови, наприклад).

Припустимо, що у нас є певний файл з аудіо даними. Спочатку нам необхідно зрозуміти, як він влаштований і як його прочитати. Розглянемо найпростіший варіант – WAV (англ. waveform audio format – формат аудіофайлу) файл.

Формат має у файлі два блоки. Перший блок – це заголовок, який має інформацію про аудіопоток: бітрейт, частоту, кількість каналів, довжину файлу і т.д. Другий блок складається з «сірих» даних – того самого цифрового сигналу, набору значень амплітуд.

Прочитаємо заголовок, перевіряємо певні обмеження (наприклад, відсутність стиснення) та зберігаємо в спеціально виділений масив дані:

```
/**
 * Represents WAV file data
 *
 * Currently supports only PCM format.
 *
 * @see http://en.wikipedia.org/wiki/WAV
 * @see http://en.wikipedia.org/wiki/Linear_pulse-code_modulation
 * @see https://ccrma.stanford.edu/courses/422/projects/WaveFormat/
 */
#ifndef WAV_DATA_H_
#define WAV_DATA_H_

#include <audio.h>
#include <cstdint>
#include <iostream>
#include <string>

namespace yazz {
namespace audio {

/**
 * WAV header
 */
struct WavHeader {
    char               riff[4];        // RIFF Header
    unsigned long     chunkSize;      // RIFF Chunk Size
    char               wave[4];        // WAVE Header

    char               fmt[4];         // FMT header
    unsigned long     subchunk1Size;  // Size of the fmt chunk
```

Рис. 2.4. Логіка читання даних

```
31      unsigned short    audioFormat;    // Audio format 1=PCM (Other formats are unsi
32      unsigned short    numOfChan;      // Number of channels 1=Mono, 2=Stereo
33      unsigned long     samplesPerSec;  // Sampling Frequency in Hz
34      unsigned long     bytesPerSec;    // bytes per second
35      unsigned short    blockAlign;     // 2=16-bit mono, 4=16-bit stereo
36      unsigned short    bitsPerSample;  // Number of bits per sample
37
38      // The data below depends on audioFormat, but we work only with PCM cases
39      char               data[4];        // DATA header
40      unsigned long     subchunk2Size;  // Sampled data length
41  };
42
43  class WavData;
44
45  /**
46   * WAV data
47   */
48  class WavData {
49  public:
50
51          ~WavData() {
52                  if (NULL != this->rawData) {
53                          delete [] this->rawData;
54                  }
55                  if (NULL != this->normalizedData) {
56                          delete [] this->normalizedData;
57                  }
58          }
59
60          static WavData* readFromFile(const std::string& file);
```

Рис. 2.5. Логіка читання даних (продовження коду)

```
62          uint32_t getNumberOfSamples() const { return numberOfSamples; }
63          void setNumberOfSamples(uint32_t numberOfSamples) { this->numberOfSamples = numberOfSamples; }
64
65          raw_t getMaxVal() const { return maxVal; }
66          void setMaxVal(raw_t maxVal) { this->maxVal = maxVal; }
67
68          raw_t getMinVal() const { return minVal; }
69          void setMinVal(raw_t minVal) { this->minVal = minVal; }
70
71          const WavHeader& getHeader() const { return header; }
72          const raw_t* getRawData() const { return rawData; }
73          const double* getNormalizedData() const { return normalizedData; }
74
75  private:
76          WavHeader               header;
77          raw_t*                  rawData;
78          double*                 normalizedData;
79
80          raw_t                   maxVal;
81          raw_t                   minVal;
82          uint32_t                numberOfSamples;
83
84          WavData(WavHeader header) {
85                  this->header = header;
86                  this->rawData = NULL;
87                  this->normalizedData = NULL;
88
89                  this->maxVal = 0;
90                  this->minVal = 0;
```

Рис. 2.6. Логіка читання даних (продовження коду)

```
91                  this->numberOfSamples = 0;
92          }
93
94          static bool checkHeader(const WavHeader& wavHeader);
95          static void readData(std::fstream& fs, const WavHeader& wavHeader, WavData& wavFile);
96  };
97
98  } // namespace audio
99  } // namespace yazz
100
101 #endif /* WAV_DATA_H_ */
```

Рис. 2.7. Логіка читання даних (кінець коду)

Наш підхід повинен бути стійкий до зміни тембру голосу, гучності і швидкості вимови. Поелементне порівняння двох аудіосигналів не дасть відповідного результату.

Насамперед розіб'ємо наші дані по невеликим тимчасовим проміжкам – фреймам. Причому фрейми повинні йти не строго один за одним, а "внапуск". Тобто кінець одного проміжку повинен перетинатися з початком наступного.

Фрейми – більш придатна одиниця аналізу даних, ніж конкретні значення сигналу. Аналізувати хвилі набагато зручніше на проміжку, ніж в конкретних точках. Розташування ж фреймів "внапуск" дозволяє згладити результати аналізу фреймів, перетворюючи ідею фреймів в певне "вікно", яке рухається вздовж значень сигналу.

Оптимальна довжина фрейму (встановлено дослідним шляхом) повинна відповідати проміжку в 10 мс (мілісекунд), «напуск» – 50%. З урахуванням того, що середня довжина слова становить близько 500 мс, такий крок дасть нам 500/(10*0.5) = 100 фреймів на слово.

Першим завданням, що потрібно вирішити при розпізнаванні мови, є розбиття цієї мови на окремі слова. Щоб було легше, припустимо, що в цьому випадку мова містить в собі певні паузи (проміжки тиші), які можна вважати "роздільниками" слів.

У цьому випадку необхідно знайти таке значення, поріг, значення вище якого є словом, а нижче – тишею. Є декілька варіантів:

- задати константою (спрацює тільки тоді, якщо вихідний сигнал завжди генерується одним і тим же способом, при одних і тих же умовах);
- кластеризувати значення сигналу, явно виділити велику кількість значень, що відповідають тиші (спрацює, якщо тиша займає більшу частину вихідного сигналу);
- проаналізувати ентропію;

У нашому випадку ентропія означає, як сильно в рамках заданого фрейму "коливається" сигнал.

Для того, щоб підрахувати ентропію конкретного фрейму, виконаємо наступні дії:

- припустимо, що сигнал пронормований і всі його значення лежать в діапазоні [-1; 1];

- побудуємо щільність розподілу (гістограму) значень сигналу фрейму: розрахуємо ентропію, наступним чином:

$$E = \sum_{i=0}^{N-1} P[i] * log_2(P[i]) \quad (2.1)$$

Отримали значення ентропії. Це всього лиш одна характеристика фрейму, і для того, щоб відокремити звук від тиші, нам все ще потрібно її з чимось порівнювати. У деяких випадках рекомендують брати поріг ентропії рівним середньому між її мінімальним і максимальним значеннями (серед всіх фреймів).

Ентропія по середині слова може просідати (на голосних) і може раптово схоплюватися через невеликий шум. Для того, щоб боротися з першою проблемою, доводиться вводити поняття "мінімальної відстані між словами" і "склеювати" набори фреймів, які лежать поблизу, але розділені через просідання. Ще одна проблема вирішується, якщо використовувати "мінімальну довжину слова" і відсікати всіх кандидатів, які не пройшли відбір (і не використані в першому пункті). Якщо ж мова взагалі не є "членороздільною", можна спробувати вихідний набір фреймів розбити на певним чином підготовлені послідовності, кожна з яких буде піддана процедурі розпізнавання.

Маємо набір фреймів, відповідних певному слову. Можна піти по шляху найменшого опору і в якості чисельної характеристики фрейму використовувати середній квадрат всіх його значень. Однак, така метрика несе в собі вкрай мало інформації придатної для подальшого аналізу.

Тому ми будемо використовувати Mel-частотні кепстральні коефіцієнти. MFCC – це певне уявлення енергії спектра сигналу. Переваги його використання полягають у наступному:

- Використовується спектр сигналу (тобто розкладання по базису ортогональних [до] синусоїдальних функцій), що дозволяє враховувати хвильову "природу" сигналу при подальшому аналізі;
- Спектр проєктується на спеціальну Mel-шкалу, дозволяючи виділити найбільш значущі для сприйняття людиною частоти;

• Кількість обчислюваних коефіцієнтів може бути обмежена будь-яким значенням. Це дозволяє "стиснути" фрейм і кількість оброблюваної інформації, як наслідок;

Розглянемо процес обчислення MFCC коефіцієнтів для певного фрейму.

Уявімо фрейм у вигляді вектору $x[k], 0 \leq k < N$, де N – розмір кадру.

Насамперед розрахуємо спектр сигналу за допомогою дискретного перетворення Фур'є:

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{\frac{-2*\pi*i*k*n}{N}}, 0 \leq k < N \quad (2.2)$$

Також до отриманих значень рекомендується застосувати віконну функцію Хеммінга, щоб "згладити" значення на кордонах фреймів:

$$H[k] = 0.54 - 0.46 * \cos\left(\frac{2 * \pi * k}{N - 1}\right) \quad (2.3)$$

Тобто результатом буде вектор вигляду:

$$X^*[k] = X[k] * H[k], 0 \leq k < N \quad (2.4)$$

Важливо пам’ятати, що після виконаного перетворення по осі X ми матимемо частоту сигналу (Hz), а по осі Y – магнітуду (як спосіб уникнути комплексних значень):

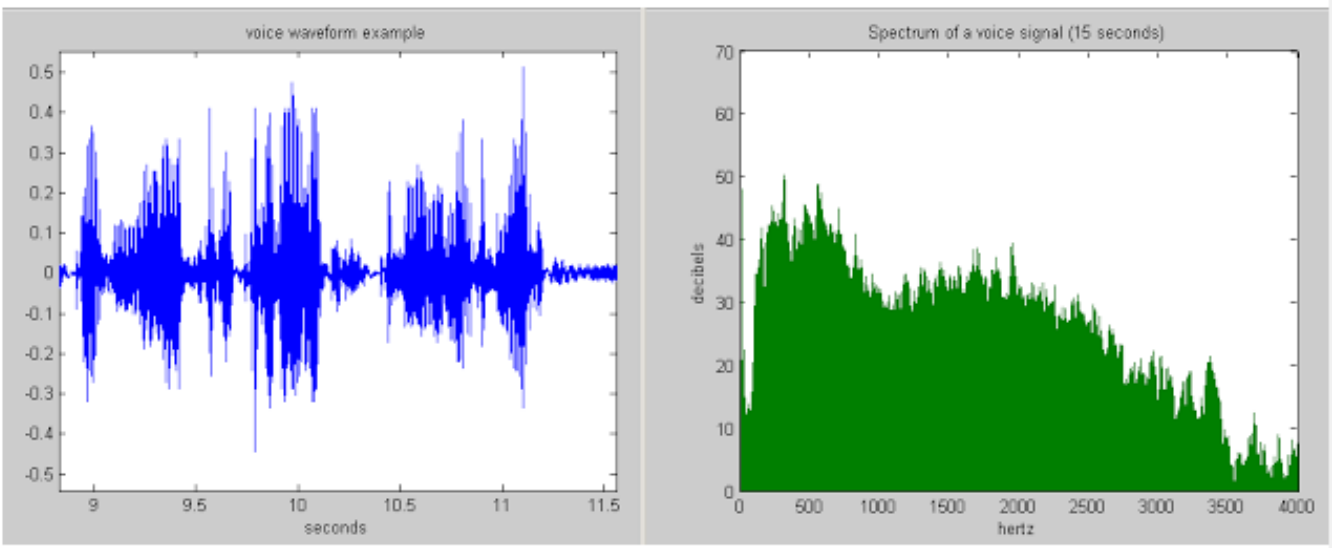

Рис. 2.8. Амплітудно-часова характеристика голосу та спектр голосового сигналу

Mel - це "психофізична одиниця висоти звуку". Вона заснована на суб'єктивному сприйнятті звуку середньостатистичними людьми. В першу чергу величина залежить від частоти звуку (а також від гучності і тембру). Іншими словами, це величина, що показує, наскільки звук певної частоти "значущий" для нас.

Перетворення частоти в Mel:

$$M = 1127 * \log\left(1 + \frac{F}{700}\right) \tag{2.5}$$

Графік залежності Mel від частоти:

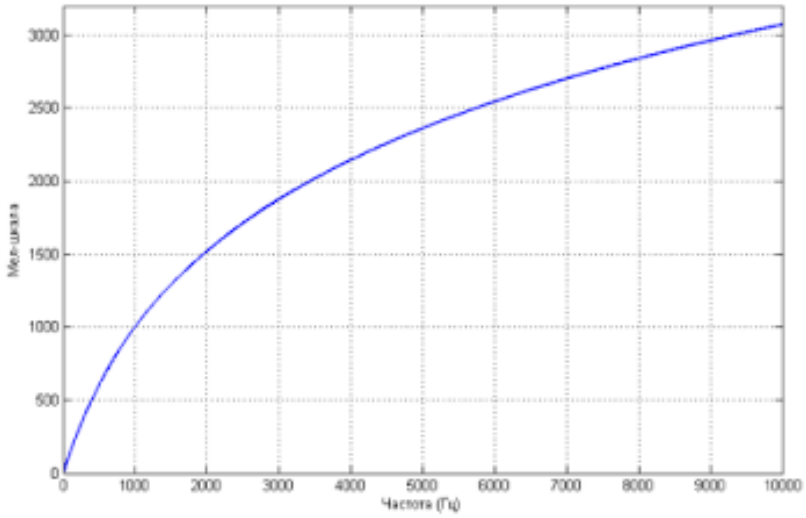

Рис. 2.9. Графік залежності Mel від частоти

Припустимо, є фрейм розміром 256 елементів. Відомо (з даних про аудіоформат), що в даному фреймі частота звуку 16000 Hz. Уявімо, що людська мова лежить в діапазоні від [300; 8000] Hz. Кількість шуканих Mel-коефіцієнтів припустимо M = 10 (рекомендоване значення).

Для того, щоб розкласти за Mel-шкалою отриманий вище спектр, потрібно створити "гребінку" фільтрів. По суті, кожен Mel-фільтр – це трикутна віконна функція, що дозволяє підсумувати кількість енергії на певному діапазоні частот і тим самим отримати Mel-коефіцієнт. Знаючи аналізований діапазон частот і кількість Mel-коефіцієнтів, можна побудувати набір таких фільтрів:

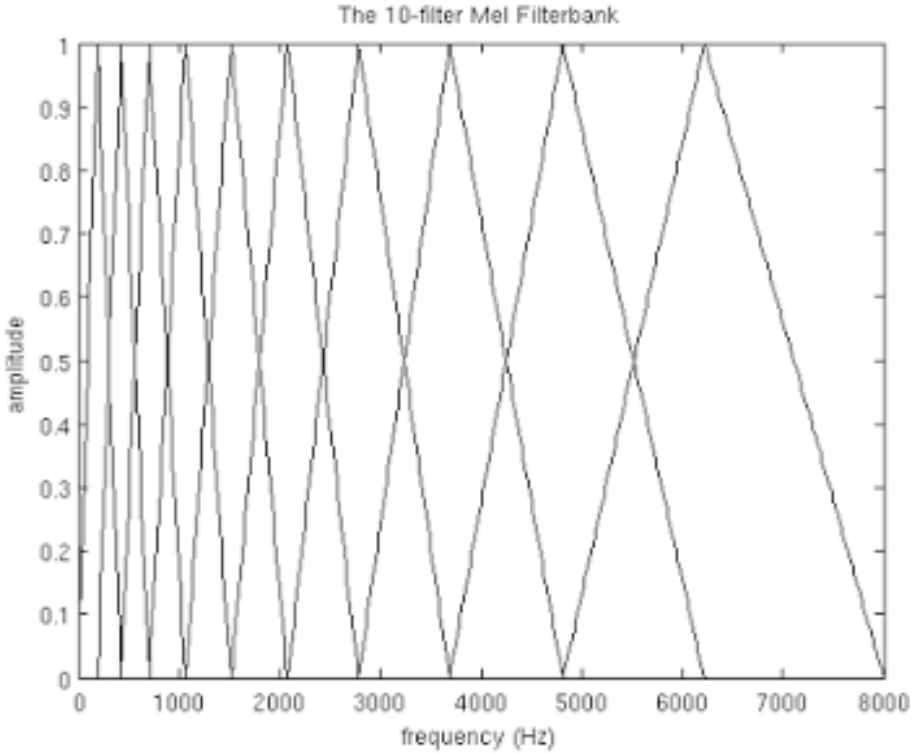

Рис. 2.10. Набір фільтрів

Чим більший порядковий номер Mel-коефіцієнта, тим ширша основа фільтра. Це пов'язано з тим, що розбиття діапазону частот на оброблювані фільтрами діапазони відбувається на Mel-шкалі. Діапазон частот дорівнює [300, 8000]. Відповідно до формули (5) цей діапазон перетворюється в [401.25; 2834.99] на Mel-шкалі.

Далі для того, щоб побудувати 10 трикутних фільтрів, потрібно 12 опорних точок:

m[i] = [401.25, 622.50, 843.75, 1065.00, 1286.25, 1507.50, 1728.74, 1949.99, 2171.24, 2392.49, 2613.74, 2834.99]

На Mel-шкалі точки розташовані рівномірно. Переведемо за допомогою формули (5) шкалу назад в Hz:

h[i] = [300, 517.33, 781.90, 1103.97, 1496.04, 1973.32, 2554.33, 3261.62, 4122.63, 5170.76, 6446.70, 8000]

Тепер шкала стала поступово розтягуватися, тим самим вирівнюючи динаміку зростання "значущості" на високих і низьких частотах.

Тепер потрібно накласти отриману шкалу на спектр фрейму. Довжина спектра – 256 елементів, при цьому в нього вміщається 16 kHz. Вирішивши пропорцію, можна отримати наступну формулу:

$$f(i) = floor((frameSize + 1) * \frac{h(i)}{sampleRate}) \quad (2.6)$$

f(i) = 4, 8, 12, 17, 23, 31, 40, 52, 66, 82, 103, 128

Знаючи опорні точки на осі X нашого спектра, можна легко побудувати необхідні нам фільтри за наступною формулою:

$$H_m(k) = \begin{cases} 0, k < f(m-1); k > f(m+1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, f(m) \le k \le f(m+1) \end{cases} \quad (2.7)$$

Застосування фільтру полягає в попарному перемножуванні його значень зі значеннями спектру. Результатом цієї операції є Mel-коефіцієнт. Оскільки фільтрів M, коефіцієнтів буде стільки ж.

$$S[m] = \log(\sum_{k=0}^{N-1} |X[k]|^2 * H_m[k]), 0 \le m < M \quad (2.8)$$

Проте, потрібно застосувати Mel-фільтри до його енергії, а не до значень спектра. Після цього потрібно прологарифмувати отримані результати. Вважається, що таким чином знижується чутливість коефіцієнтів до шумів.

Дискретне косинусне перетворення використовується для того, щоб отримати ті самі "кепстральні" коефіцієнти. Сенс його в тому, щоб "стиснути" отримані результати, підвищивши значимість перших коефіцієнтів і зменшивши значимість останніх.

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos(\pi * l * \frac{m + 0{,}5}{M})), 0 \le l < M \quad (2.9)$$

Тепер для кожного фрейму маємо набір з M MFCC-коефіцієнтів, які можуть використовуватись для подальшого аналізу:

```
#include <complex.h>
#include <MFCC.h>
#include <cmath>
#include <cstring>
#include <limits>
#include <assert.h>
#include "../common.h"

using namespace std;

namespace yazz {
namespace math {

double* MFCC::transform(const double* source, uint32_t start, uint32_t finish, uint8_t mfccSize,
                uint32_t frequency, uint32_t freqMin, uint32_t freqMax) {
        uint32_t sampleLength = finish - start + 1;

        // Calc
        double* fourierRaw = fourierTransform(source, start, sampleLength, true);
        double** melFilters = getMelFilters(mfccSize, sampleLength, frequency, freqMin, freqMax);
        double* logPower = calcPower(fourierRaw, sampleLength, melFilters, mfccSize);
        double* dctRaw = dctTransform(logPower, mfccSize);

        // Clean up
        delete [] logPower;
        delete [] fourierRaw;

        for (unsigned short m = 0; m < mfccSize; m++) {
                delete [] melFilters[m];
        }
        delete [] melFilters;

        return dctRaw;
}

/**
 * Preemphasis digital filtration
 */
double* MFCC::filter(const double* source, uint32_t start, uint32_t finish) {
        UNUSED(source);
        UNUSED(start);
        UNUSED(finish);
```

Рис. 2.11. Порівняння

```
        return NULL;
}

/**
 * Compute singnal's spectrum and its magnitudes (short-time Fourier transform with Hamming window)
 */
double* MFCC::fourierTransform(const double* source, uint32_t start, uint32_t length,
                bool useWindow) {

        complex<double>* fourierCmplxRaw = new complex<double>[length];
        double* fourierRaw = new double[length];


        for (uint32_t k = 0; k < length; k++) {
                fourierCmplxRaw[k] = complex<double>(0, 0);

                for (uint32_t n = 0; n < length; n++) {
                        double sample = source[start + n];

                        // According Euler's formula: e^(ix) = cos(x) + i*sin(x)
                        double x = -2. * M_PI * k * n / (double) length;
                        complex<double> f = sample * complex<double>(cos(x), sin(x));

                        double w = 1.;
                        if (useWindow) {
                                // Hamming window
                                w = 0.54 - 0.46 * cos(2 * M_PI * n / (length - 1));
                        }

                        fourierCmplxRaw[k] += f * w;
                }

                // As for magnitude, let's use Euclid's distance for its calculation
                fourierRaw[k] = sqrt(norm(fourierCmplxRaw[k]));
        }

        delete [] fourierCmplxRaw;

        return fourierRaw;
}
```

Рис. 2.12. Порівняння (продовження коду)

```
90  double** MFCC::getMelFilters(uint8_t mfccSize, uint32_t filterLength, uint32_t frequency,
91                  uint32_t freqMin, uint32_t freqMax) {
92
93          // Create points for filter banks
94          double* fb = new double[mfccSize + 2];
95          fb[0] = convertToMel(freqMin);
96          fb[mfccSize + 1] = convertToMel(freqMax);
97
98          // Create mel bin
99          for (unsigned short m = 1; m < mfccSize + 1; m++) {
100                 fb[m] = fb[0] + m * (fb[mfccSize + 1] - fb[0]) / (mfccSize + 1);
101         }
102
103         //frequency = 0.5 * frequency;
104         for (unsigned short m = 0; m < mfccSize + 2; m++) {
105
106                 // Convert them from mel to frequency
107                 fb[m] = convertFromMel(fb[m]);
108
109                 // Map those frequencies to the nearest FT bin
110                 fb[m] = floor((filterLength + 1) * fb[m] / (double) frequency);
111
112                 assert("FT bin too small" &&
113                                 !(m > 0 && (fb[m] - fb[m-1]) < numeric_limits<double>::epsilon()));
114         }
115
116         // Calc filter banks
117         double** filterBanks = new double*[mfccSize];
118         for (unsigned short m = 0; m < mfccSize; m++) {
119                 filterBanks[m] =  new double[filterLength];
120         }
121
122         for (unsigned short m = 1; m < mfccSize + 1; m++) {
123                 for (uint32_t k = 0; k < filterLength; k++) {
124
125                         if (fb[m - 1] <= k && k <= fb[m]) {
126                                 filterBanks[m - 1][k] = (k - fb[m - 1]) / (fb[m] - fb[m - 1]);
127
128                         } else if (fb[m] < k && k <= fb[m + 1]) {
129                                 filterBanks[m - 1][k] = (fb[m + 1] - k) / (fb[m + 1] - fb[m]);
130
131                         } else {
132                                 filterBanks[m - 1][k] = 0;
133                         }
```

Рис. 2.13. Порівняння (продовження коду)

```
134                 }
135         }
136
137         delete [] fb;
138
139         return filterBanks;
140 }
141
142 /**
143  * Apply mel filters to spectrum's magnitudes, take the logs of the powers
144  */
145 double* MFCC::calcPower(const double* fourierRaw, uint32_t fourierLength,
146                 double** melFilters, uint8_t mfccCount) {
147
148         double* logPower = new double[mfccCount];
149
150         for (unsigned short m = 0; m < mfccCount; m++) {
151                 logPower[m] = 0.;
152
153                 for (uint32_t k = 0; k < fourierLength; k++) {
154                         logPower[m] += melFilters[m][k] * pow(fourierRaw[k], 2);
155                 }
156
157                 assert("Spectrum power is less than zero" &&
158                                 !(logPower[m] < numeric_limits<double>::epsilon()));
159
160                 // NOTE I'm not sure that we need to take logs since we normalized the input data
161                 logPower[m] = log(logPower[m]);
162         }
163
164         return logPower;
165 }
166
167 /**
168  * Take the discrete cosine transform of the list of mel log powers
169  */
170 double* MFCC::dctTransform(const double* data, uint32_t length) {
171
172         double* dctTransform = new double[length];
173
174         for (unsigned short n = 0; n < length; n++) {
175                 dctTransform[n] = 0;
```

Рис. 2.14. Порівняння (продовження коду)

```
177                 for (unsigned short m = 0; m < length; m++) {
178                         dctTransform[n] += data[m] * cos(M_PI * n * (m + 1./2.) / length);
179                 }
180         }
181
182         return dctTransform;
183 }
184
185 } /* namespace math */
186 } /* namespace yazz */
```

Рис. 2.15. Порівняння (кінець коду)

Тепер наше завдання зводиться до підбору найбільш "близької" моделі для деякого набору MFCC-коефіцієнтів (розпізнавання слова). На перший погляд завдання можна вирішити наступним чином:

- для кожної моделі знайдемо середню (евклідову) відстань між ідентифікованим MFCC-вектором і векторами моделі;
- вибираємо в якості вірної ту модель, середня відстань до якої буде найменшою [13].

Але розмір MFCC-вектора може бути різний для одного і того ж слова.

Це дослідження дає змогу показати, що кожен з нас має індивідуальні голосові параметри, але застосування моделей розпізнавання мови з залученням моделей Big Data матиме змогу розпізнати фізичний та психологічний стан пацієнта під час прийому звернення (наприклад, передінфарктний стан) та скоординувати бригаду у найшвидші терміни.

РОЗДІЛ 3
ВИДИ АТАК НА IP-АТС ASTERISK ТА МЕТОДИ ПРОТИСТОЯННЯ ЇМ

3.1. Будова мережі IP-телефонії на базі IP-АТС Asterisk

IP-телефонія - це технологія, яка дозволяє використовувати Інтернет або будь-яку іншу IP-мережу для проведення міжнародних і міжміських телефонних розмов та передачі факсу в режимі реального часу.

Будь-яка мережа IP-телефонії використовує мережу передачі даних, найчастіше це комп'ютерна мережа. Голос в таких мережах передається в цифровому вигляді за допомогою RTP (англ. Real-time Transport Protocol – протокол транспортування інформації в реальному часі) пакетів.

Під IP-телефонією мається на увазі голосовий зв'язок, який здійснюється по мережах передачі даних, зокрема по IP-мережам. На сьогоднішній день IP-телефонія все більше витісняє традиційні телефонні мережі за рахунок легкості реалізації, низькою вартістю дзвінка, простоти конфігурації, високої якості зв'язку та відносній безпеці з'єднання [5].

При проектування і реалізації системи необхідно звернути увагу на дизайн мережі.

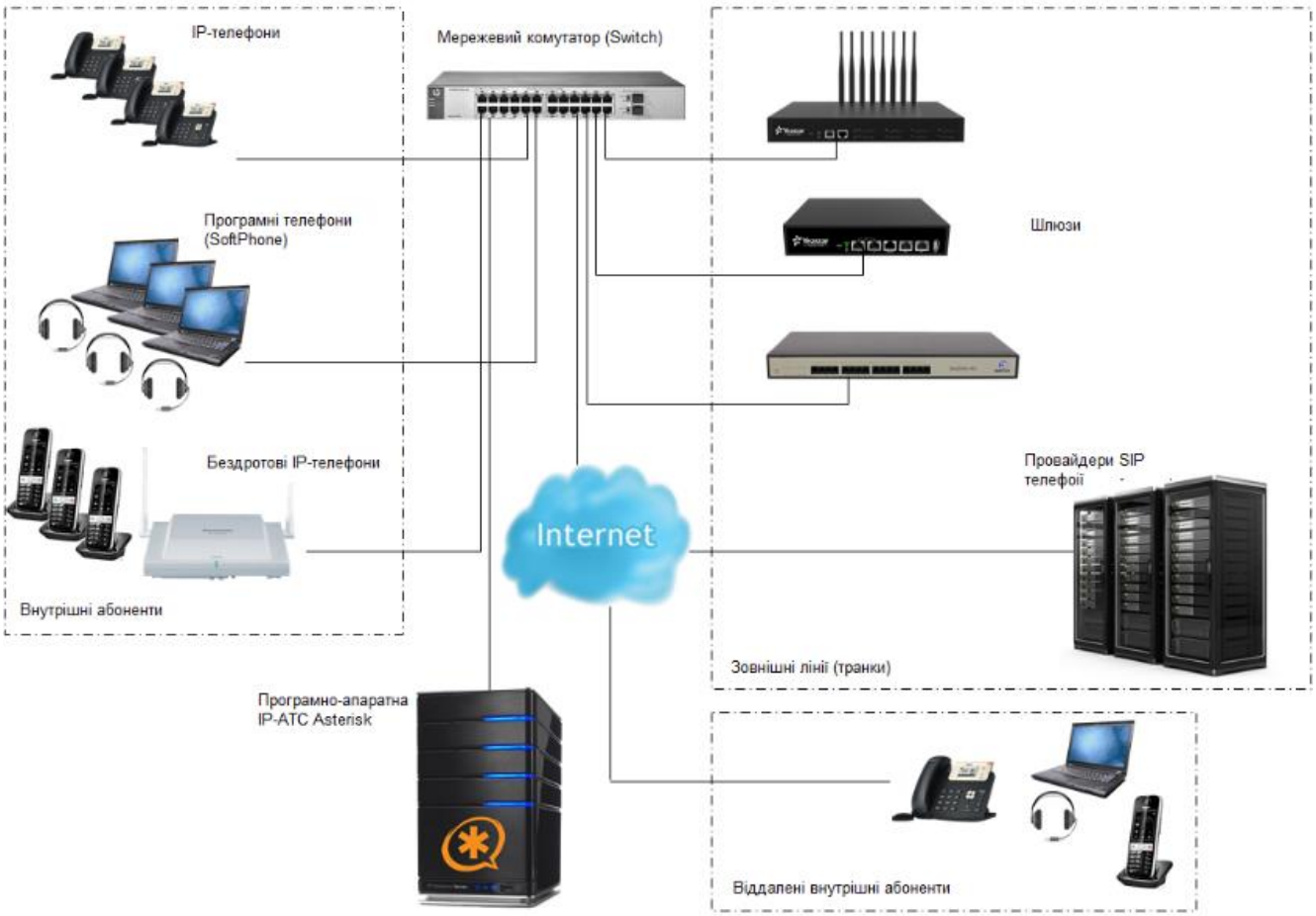

Рис. 3.1. Мережа IP телефонії [6]

Устаткування IP телефонії для офісу може включати в себе цілий комплекс пристроїв (рис. 3.1), кожен з яких виконує певні функції:

- АТС для IP телефонії. Це основний комутаційний пристрій, який не тільки забезпечує підключення до традиційних послуг зв'язку, а й створює локальну мережу передачі даних всередині компанії, розвантажуючи зовнішню лінію. Такий комплекс обладнання володіє широким вибором додаткових функцій, який гідно оцінили власники бізнесу - автоматична або ручна переадресація, внутрішній безкоштовний зв'язок, конференц-зв'язок, утримання виклику і заборона (обмеження) вхідних дзвінків з певних номерів і багато іншого;
- сервера для IP телефонії. Серверні станції забезпечують контроль локальної мережі і об'єднують АТС і кінцевого користувача. Усередині кожного сервера можуть перебувати цифрові і аналогові пристрої для перетворення (кодування) сигналу, а та-

кож спеціальні плати для транслювання інформації на різні зовнішні пристрої. Завдяки уніфікованим елементам, при необхідності можна істотно розширити абонентську внутрішню мережу або оснастити сервер GSM (англ. Global System for Mobile Communications – міжнародний стандарт для мобільного цифрового стільникового зв'язку) модулем для повноцінного мобільного зв'язку;

- телефони для IP телефонії. Це кінцеві термінали для здійснення обміну інформацією між абонентами. Зовні вони нічим не відрізняються від стандартних телефонних апаратів, але мають спеціальний роз'єм, завдяки якому вони підключаються до мережі. Також багато сучасних апаратів підтримують функцію відеозв'язку, що істотно розширює їх функціональність;
- гарнітура для IP телефонії. Це допоміжні пристрої, які є незамінними для роботи call-центрів. На відміну від звичайної телефонної трубки, гарнітура кріпиться на голові оператора, звільняючи руки, що дозволяє співробітникові компанії одночасно вести розмову і працювати за комп'ютером. Крім того, гарнітура істотно підвищує якість сигналу і усуває ефект еха, в порівнянні з підключенням мікрофона через зовнішні динаміки;
- програми для IP телефонії. Нормальну функціональність зв'язку неможливо забезпечити без спеціального програмного забезпечення, яке призначене для кодування сигналу, його перетворення, стиснення пакетів даних, контролю за трафіком і для інших функцій [6].

Часто без будь-якої потреби Asterisk підключають до мережі за найбільш вразливою схемою, зображеної на рис. 3.2.

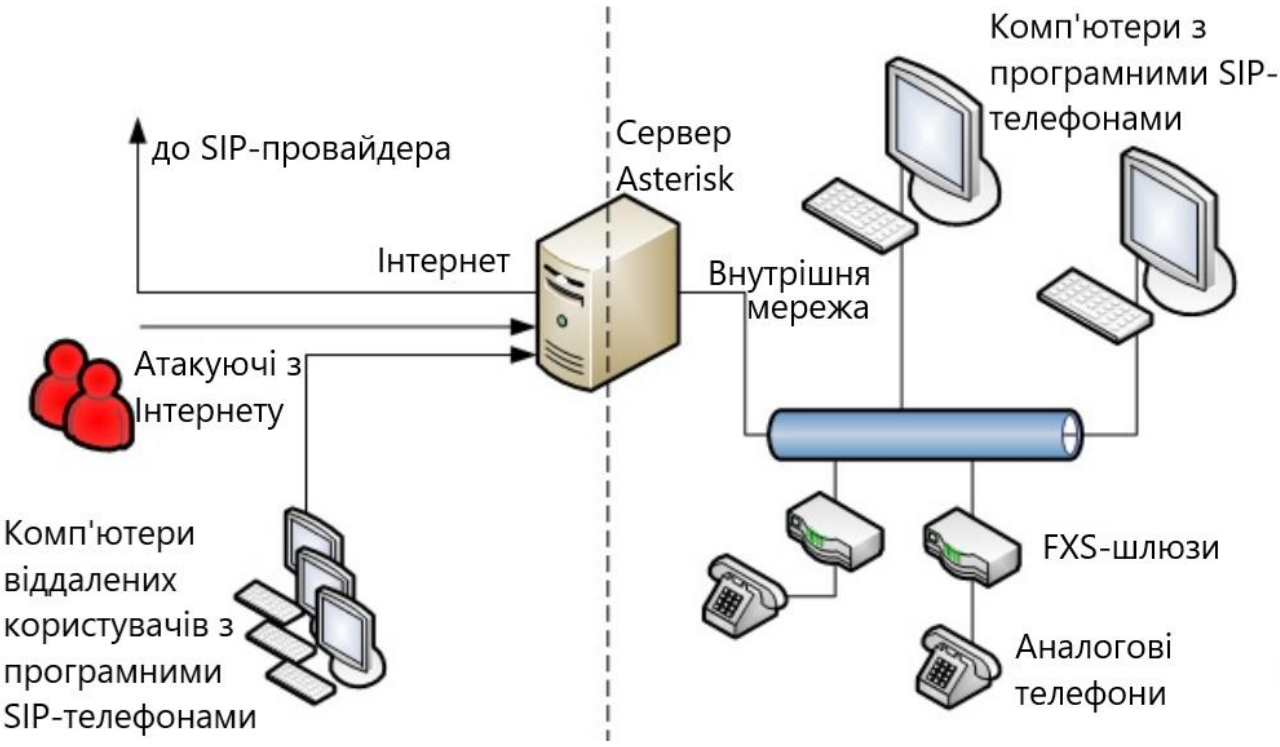

Рис. 3.2. Не рекомендована схема підключення Asterisk через зовнішню адресу

Факт доступності Asterisk з Інтернету – головна загроза безпеці всієї системи. Будь-який слабкий пароль або будь-яка вразливість у вихідному коді Asterisk можуть бути використані атакуючими, щоб отримати несанкціонований доступ до АТС і дзвонити за рахунок компанії-власника сервера Asterisk. У кращому випадку метою атакуючих може стати відмова в обслуговуванні.

Стійкі паролі і регулярні оновлення, звичайно ж, знижують ризик. Ще сильніше його можна знизити спеціалізованими засобами захисту, а саме – міжмережевими екранами (МЕ). Якщо підключати Asterisk так, як це показано на рис. 3.3, то сервер буде безпечно закритий за периметром локальної мережі.

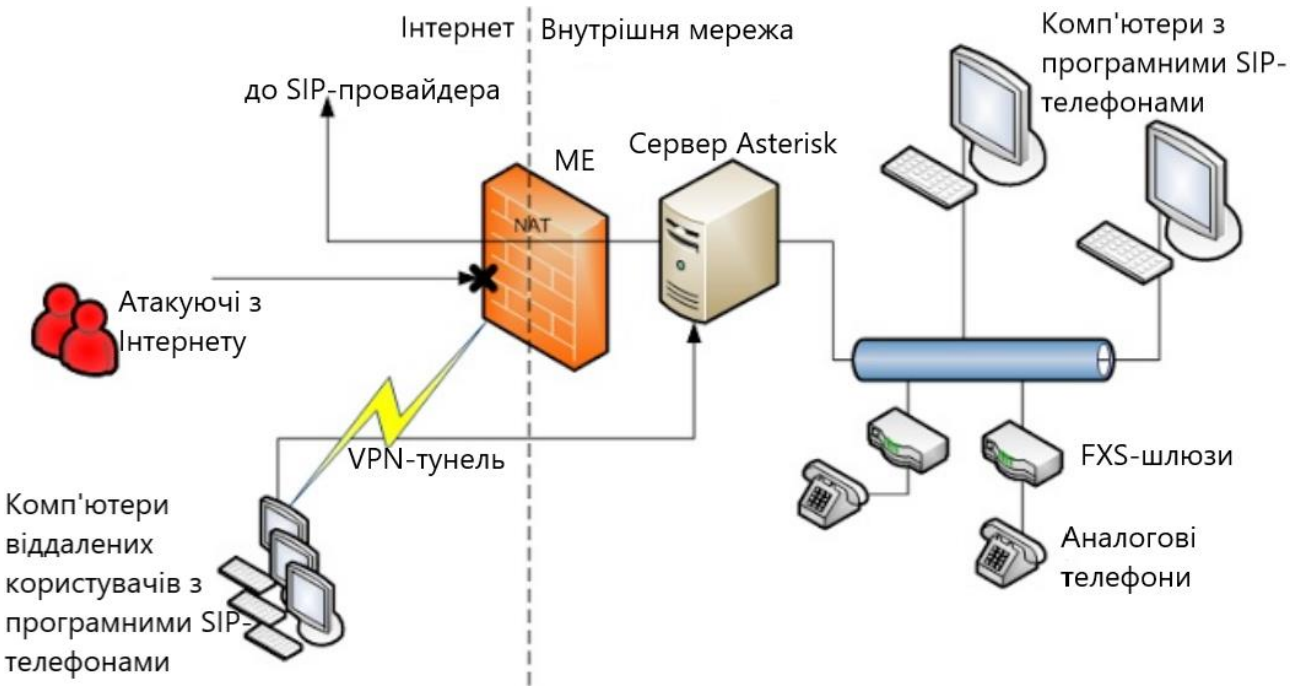

Рис. 3.3. Рекомендована схема підключення Asterisk з використанням ME

ME пропускає вихідний трафік від сервера Asterisk до SIP провайдера і назад через динамічні правила NAT (англ. Network Address Translation – технологія перетворення приватних IP-адрес в зовнішні в IPv4). А можливість підключення користувачів віддалених офісів забезпечується через VPN-тунель. Користувач спочатку підключається по VPN до мережі підприємства, і вже потім, по віртуальному каналу – до сервера Asterisk. Asterisk більше не видно із зовнішньої мережі, недоступний атакуючим [7].

3.2. Аналіз видів можливих атак на сервер Asterisk

3.2.1. Перепродаж трафіку (Tollfroud)

Перепродаж трафіку – один з найбільш зручних способів заробляння грошей хакерами. Зламавши станцію і направивши дзвінки на дорогі міжнародні напрямки, хакер отримує якусь винагороду на свої електронні гаманці, які далі переводяться в готівку і виходять реальні гроші. Наочна схема зображена на рис. 3.4.

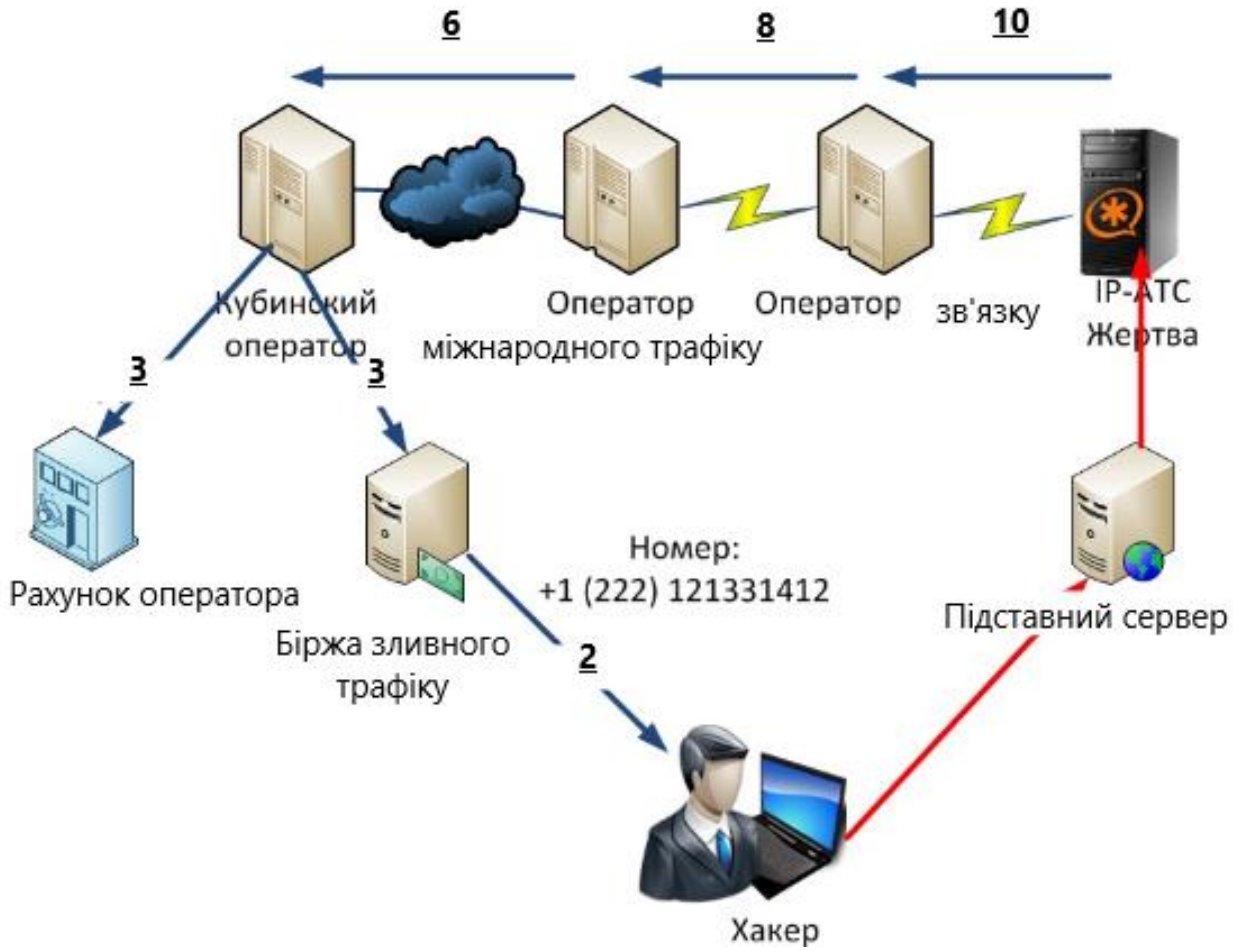

Рис. 3.4. Схема перепродажу трафіку

Розглянемо схему детальніше. Є хакер, який хоче заробити грошей шляхом перепродажу трафіку. Він йде на біржу зливного трафіку, реєструється і отримує якийсь пул номерів, які можуть його ідентифікувати надалі. Далі через якісь підставні сервера знаходяться і, якщо виходить, зламуються сервера IP-АТС. Далі на номери, які були надані хакеру на біржі зливного трафіку, починають йти дзвінки. Дзвінки платні. Припустимо, дзвінок коштує 10 грн (оплата жертви оператору). Дзвінок направляється, як правило за кордон (як правило це Куба, Палестина, Румунія, Латвія і т.д.), в країни, де не приділяється належної уваги кібер-злочинності, тобто де є можливість займатися такого характеру бізнесом. І так жертва платить Оператор зв'язку (ОЗ) 10 грн за хвилину. ОЗ виходить на міжнародні лінії через оператор міжнародного трафіку (ОМТ), якому він платить 8 грн за хвилину, далі ОМТ направляє дзвінок на місцевого оператора (Куба, Палестина, і ін.), який займається «відмиванням» голосового трафіку і платить йому вже 6 грн за хвилину, даний оператор розподіляє гроші в такий

спосіб – частина грошей йде йому в оплату за послуги (так скажемо, заробляє цю суму – залишає у себе на рахунку), а частину грошей направляє на біржу зливного трафіку, де зареєстрований хакер. Далі біржа, залишивши собі 1 грн, дві платить хакеру [8].

3.2.2. DDoS-атаки

DDoS (Distributed Denial of Service attack) – це вторгнення, яке задовольняє співвідношення «багато порушників – одна жертва». Вторгнення вважається розподіленим, якщо різні його етапи виконуються від імені різних джерел в мережі. Події, що становлять розподілене вторгнення, є скоординованими. Джерела, від імені яких виконується розподілене вторгнення (атака), взаємопов'язані.

Простий трафік – це HTTP-запити. Основа запиту – HTTP-заголовок. Запитуюча сторона може використовувати скільки завгодно заголовків, надаючи їм потрібні властивості. Зловмисники, які проводять DDoS, можуть змінювати ці заголовки, тому їх важко розпізнати як атаки.

HTTP(S) GET-запит – спосіб, що запитує дані на сервері. Цей запит може "попросити" у сервера передати якийсь файл, зображення, сторінку або скрипт для відображення в веб-браузері. HTTP(S) GET-флуд – DDoS атака прикладного рівня (7) моделі OSI. Зловмисник посилає потужний потік запитів на сервер для переповнення його ресурсів. У цьому випадку сервер перестає відповідати на запити реальних користувачів.

HTTP(S) POST-запит – метод, суть якого лежить в тому, що дані розміщуються в тілі запиту для подальшої обробки на сервері. HTTP POST-запит кодує передану інформацію і поміщає на форму, а потім відправляє цей контент на сервер. Цей метод використовують, коли необхідно передавати великі обсяги даних. HTTP(S) POST-флуд – тип DDoS-атаки, при якому кількість POST-запитів переповнюють сервер, в результаті він не може відповісти на них. Це тягне за собою аварійну зупинку сервера з відповідними наслідками.

Всі перераховані запити також передаються по HTTPS, дані, що надсилаються, в такому випадку шифруються. І подібний захист грає на користь хакерам. Адже, щоб виявити такий запит, сервер повинен спочатку розшифрувати його. А розшифрувати

потік запитів під час такої атаки дуже складно і це створює додаткове навантаження на сервер.

ICMP-флуд (Smurf-атака). Досить небезпечний тип атаки. Хакер відправляє підроблений ICMP-пакет, в якому адреса того, хто атакує, змінюється на адресу жертви. Всі вузли надсилають відповідь на отриманий ping-запит. Для цього в більшості випадків використовують велику мережу, щоб у комп'ютера-жертви не було ніяких шансів.

UDP-флуд (англ. User Datagram Protocol – протокол транспортного рівня моделі OSI) (або атака Fraggle). За своїм типом аналогічний ICMP флуду, але в даному випадку застосовуються UDP пакети. Через насичення смуг пропускання відбувається відмова в обслуговуванні сервера жертви.

SYN-флуд (англ. Synchronize sequence numbers – синхронізація номерів послідовності). В основі такої атаки лежить запуск великої кількості одночасних TCP-з'єднань (англ. Transmission Control Protocol – протокол управління передачею) через посилання SYN-пакету з неіснуючою зворотною адресою.

Відправлення «важких пакетів». При подібному виді атаки зловмисник відсилає пакети сервера, що витрачають його процесорний час, але не насичують смугу пропускання. У підсумку відбувається збій в системі, і користувачі не можуть отримати свої ресурси.

Переповнення сервера лог-файлами. При некоректній системі ротації лог-файлів шахрай може відправляти об'ємні пакети, які незабаром займуть все вільне місце на жорсткому диску сервера. В результаті – збій в системі.

Помилки програмного коду. Деякі зловмисники з досвідом в даній сфері діяльності розробляють спеціальні програми-експлоїти, які дозволяють атакувати складні системи комерційних організацій. Для цього вони шукають помилки в коді програм, які можуть привести до завершення служби.

Недоліки в програмному коді. Та ж ситуація: хакери шукають помилки в коді програм або ОС (операційна система), при цьому змушують їх обробляти виняткові ситуації, в результаті – програми виходять з ладу [9].

3.3. Багаторівневий захист IP-АТС Asterisk

Говорячи про безпеку рішення IP-АТС в цілому потрібно розуміти, що безпека будується не тільки на безпеці самого Asterisk, так само необхідно забезпечити безпеку і оточенню Asterisk.

Типові помилки, які допускають адміністратори Asterisk.

1. Відсутність мережевого екрану (відсутність iptables, або вимкнений iptables, або налаштований не вірно).

2. Старі дистрибутиви і програмне забезпечення (старе ПЗ (програмне забезпечення) може містити уразливості. Необхідні постійні оновлення системи).

3. Стандартні логіни і паролі (Web-інтерфейс, SQL, обладнання).

4. Дизайн мережі (Asterisk і обладнання, що працює з ним по зовнішній адресі без захисту – це ласий шматочок для хакерів).

5. Помилки в конфігурації.

6. Відсутність контролю за системою (немає контролю логів) [9].

На жаль, проблеми з IP-АТС Asterisk останнім часом стали частими. Тому багато хто вважає, що Asterisk небезпечна система. Однак цю тезу хочеться заперечити. По-перше, якщо говорити про можливості самого Asterisk, то у нього дуже багатий набір інструментів для забезпечення безпеки. Тим не менше він дуже часто піддається зломам. Чому так? Справа все в адміністраторі даної системи. Немає єдиної галочки для активації безпеки Asterisk. Безпека такої мережі - це прийняття ряду комплексних заходів, про які, як правило адміністратори забувають, тим самим піддають Asterisk загрозам.

Розглянемо варіанти захисту від перепродажу трафіку.

За допомогою грамотно написаного dialplan (план маршрутизації дзвінків) можна значно підвищити безпеку. Перше, що рекомендується, - це поділ абонентів на контексти зі своїми правилами маршрутизації.

В sip.conf:

[1000]

context = from_chef

[1001]

context = from_it

[1002]

context = from_fin

Поділ за контекстами дає нам можливість наділити різними правами абонентів на телефонний зв'язок. Це важливо, тому що не всім потрібно, припустимо, дзвонити на міжнародні напрямки. Далі контекстам призначаємо правила.

[allow]

*exten =>_X., n, Dial(SIP/operator/${EXTEN}) ;*Дозволяємо будь-які дзвінки через оператора

*exten => _[12]XXX, n, Dial(SIP/${EXTEN}) ;*Дозволяємо внутрішні з'єднання

[from_chef]

*exten=> _X., n, Goto(allow,${EXTEN},1) ;*Всі дзвінки направляємо в створений контекст [allow]

[from_it]

*exten=> _9810., n, Hungup() ;*Обмежуємо міжнародні напрямки

*exten=> _X., n, Goto(allow,${EXTEN},1) ;*Всі дзвінки направляємо в контекст [allow]

[from_fin]

*exten=> _9810., n, Hungup() ;*Обмежуємо міжнародні напрямки

*exten=> _989., n, Hungup() ;*Обмежуємо направлення на стільникові напрямки

*exten=> _X., n, Goto(allow, ${EXTEN},1) ;*Всі дзвінки направляємо в контекст [allow]

Тим самим фінансовому відділу забороняємо дзвінки на міжнародні напрямки і на стільникові напрямки. IT підрозділу заборонені тільки міжнародні напрямки. Ну а керівнику дозволяємо будь-які з'єднання [10].

Для того, щоб отримувати інформацію про заборонені дзвінки на пошту, навчимо Asterisk фіскалити про порушення. Для цього додамо ще один контекст [alarm]. І направляємо в цей контекст заборонені дзвінки.

[alarm]

*exten => _9810X., 1, playback(zaboroneno) ;*Повідомляємо абоненту, що йому заборонений даний напрям

*exten => _9810X., n, System(echo «To» ${EXTEN} «Ext» ${CALLERID(num)} | mail -s «8-10 ALARM» it_admin@ukr.net) ;*Фіскалить на пошту, якщо будуть міжнародні дзвінки

exten => _9810X., n, Hangup()

Таким чином, не тільки блокуємо заборонені напрямки, а й у разі інциденту отримуємо повідомлення, хто (ext) намагається додзвонитися. Тим самим вчасно зможемо зрозуміти, що сервер зламаний, а не тоді, коли прийдуть величезні рахунки за зв'язок.

Такий план маршрутизації безпечний у тому випадку, якщо доступ до контексту є тільки у керівника, як і передбачається. Але можна захистити і цей контекст від несанкціонованого використання ресурсів компанії. У dialplan ми можемо прописати функцію IVR (англ. Interactive Voice Response – інтерактивне голосове меню), яка буде виконувати роль пароля для доступу до здійснення дзвінків на заборонені напрямки.

Ось чому добре використовувати Asterisk, dialplan дозволяє гнучко налаштувати маршрутизацію. Можемо забороняти напрямки за різними параметрами, можемо розмежовувати права абонентам, можемо налаштувати повідомлення про заборонені дзвінки, так само можна налаштувати обмеження дзвінків по часу, по введенню pin-коду і багато іншого, що дозволяє підвищити безпеку IP-АТС.

Тепер розглянемо варіанти захисту від ще однієї загрози, що може загрожувати роботі серверу IP телефонії, – DdoS-атаки.

Методи боротьби з DDoS можна поділити на два види: активні і пасивні. Пасивні – це заздалегідь підготовлені методи обережності і запобігання атаки. Активні застосовуються в разі, якщо атака вже відбувається на даний момент [11].

Основний пасивний метод – це, звичайно ж, запобігання. Багато хто вважає цей метод несуттєвим, але все-таки в більшості випадків він є основним.

В основі профілактики повинно лежати виключення таких факторів як особиста неприязнь, конкуренція, релігійні або інші розбіжності. Якщо вчасно усунути подібні причини і зробити відповідні висновки, то DDoS не торкнеться вашої мережі. Але цей метод більше стосується управління, а не технічного боку проблеми.

Використання спеціалізованого програмного і апаратного забезпечення.

Сьогодні багато компаній-виробників розробили спеціальні і вже готові рішення для захисту від DDoS-атак. Це програмне забезпечення різних видів для захисту малих і найбільших мереж для різних типів організацій. Це також вважається пасивним методом захисту, так як є превентивним способом.

Фільтрація і блокування трафіку, який виходить від атакуючих машин дає можливість знизити або зовсім погасити атаку. Існує два способи фільтрації: маршрутизація за списками ACL (англ. Access Control List – список правил, що забороняють або дозволяють використання ресурсів мережі) і використання міжмережевих екранів. Використання списків ACL дозволяє фільтрувати другорядні протоколи, не зачіпаючи при цьому протоколи TCP і не уповільнюючи швидкість роботи користувачів з ресурсом. Міжмережеві екрани застосовуються виключно для захисту приватних мереж (рис. 3.3).

Зворотний DDoS - перенаправлення трафіку на атакуючого. Якщо мережа має достатні серверні потужності, то можна не тільки подолати атаку, але і вивести з ладу обладнання атакуючого.

Усунення вразливостей - тип захисту націлений на усунення помилок в системах або службах. На жаль, такий спосіб захисту не працює проти флуд-атак.

Побудова розподільних систем - дозволяє обслуговувати користувачів, навіть якщо деякі вузли стають недоступними через DDoS-атаки. Для цього використовується мережеве або серверне обладнання різних типів, яке розміщується в різних ДЦ (дата-центрах). Також часто встановлюють дублюючу (резервну) систему. Це вигідно робити для великих проектів, які хвилюються за свою репутацію і мають величезну кількість користувачів.

Моніторинг - установка спеціальної системи моніторингу та оповіщення. Вона дасть можливість обчислити DDoS-атаку за певними критеріями. Моніторинг безпосередньо не захищає систему, що атакується, але дозволяє вчасно зреагувати і запобігти збій в системі роботи ресурсу. Наприклад, оповіщення про недостатній залишок місця на дисках сервера, про перезагрузку або повне вимкнення сервера. Це, звичайно ж, пасивний метод захисту.

Придбання сервісу по захисту від DDoS-атак - дає можливість захиститися від багатьох типів DDoS-атак, використовуючи цілий комплекс механізмів фільтрації небажаного трафіку до атакуючих серверів. Правда, коштують такі сервіси недешево [12].

ВИСНОВКИ

У роботі були досліджені проблематики сучасної медицини та шлях їх вирішення через концепцію розумної лікарні. У майбутньому створення ситуаційного диспетчерського центру розумної лікарні на базі запропонованих технологій та розвиток розумного міста нададуть змогу втілювати в життя концепції проєктів:

- “Алгоритм Зелена Хвиля” (управління дорожньою інфраструктурою міста для своєчасного надання допомоги екстрених служб у кризових ситуаціях), та прийняття рішення про активацію алгоритму, адже він є дієвим, проте ресурсозатратним;
- Використання віртуальної реальності у ситуаційних центрах для швидкої координації працівників при вирішенні екстрених ситуацій (теракт, техногенна катастрофа, тощо).

Дані проєкти не підлягають детальному розгляду у роботі, тому що потребують залучення концепцій розумного міста та змін на законодавчому рівні, проте існують, як логічний розвиток описаної концепції.

Таким чином, використання технологій BigData та мовної аналітики у поєднанні з системою Е-медицина допоможуть відображати необхідні дані, що знаходяться в реєстрах інших установ, та підказувати працівникам оптимальні алгоритми взаємодії, будуючи причинно-наслідковий зв’язок, відображаючи картину подій у повному, необхідному розрізі, аналізуючи прийняті рішення та корегуючи їх наступного разу.

Системи класу екстреного реагування, повинні бути високоточними, швидкодіючими та оснащуватися заходами підвищеного рівню безпеки.

Розглянуті приклади можливих атак на систему телефонії досліджувались на основі реальних позаштатних ситуацій інтерв'ювованих працівників телекомунікаційної сфери.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zabbix: мониторим всё подряд (на примере Redis'a) [Електронний ресурс]. – 2020. – Режим доступу: https://habr.com/ru/post/485538/ (дата звернення 06.06.2021 р.) – Назва з екрана.

2. Corezoid process engine. Введение [Електронний ресурс]. – 2019. – Режим доступу: https://doc.corezoid.com/v2/docs/protocol-description-v2 (дата звернення 06.06.2021 р.) – Назва з екрана.

3. Big Data (Великі дані) [Електронний ресурс]. – 2018. – Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/big-data-bolshie-dannye (дата звернення 06.06.2021 р.) – Назва з екрана.

4. Big Data от А до Я. Часть 1: Принципы работы с большими данными, парадигма MapReduce [Електронний ресурс]. – 2015. – Режим доступу: https://habr.com/ru/post/267361/ (дата звернення 06.06.2021 р.) – Назва з екрана.

5. Меггелен Дж. Asterisk: будущее телефонии, 2-е издание / Дж. Меггелен, Л. Мадсен, Дж. Смит – СПб: Символ-Плюс, 2009. – 656 с.

6. Платов М. Asterisk и Linux – миссия IP-телефония / М. Платов – 2005 г. - № 31. – С. 12-19.

7. Платов М. Asterisk и Linux: миссия IP-телефония. Действие 2 / М. Платов – 2005 р. - № 32. – С. 32-38.

8. База знаний Asterisk [Електронний ресурс] – Режим доступу: asterisk.ru/knowledgebase (дата звернення 06.06.2021 р.) – Назва з екрана.

9. ХабрХабр. Безопасность в VoIP сетях [Електронний ресурс] — Режим доступу: habrahabr.ru/post/145206 (дата звернення 06.06.2021 р.) – Назва з екрана.

10. Росляков А.В. IP-телефония / А.В. Росляков, М.Ю. Самсонов, И.В. Шибаева – М.: Эко-Трендз, 2003. -252 с.

11. Гольдштейн Б.С. IP-телефония / Б.С. Гольдштейн, А.В. Пинчук, А.Л. Суховицкий – М.: Радио и связь, 2001. -336 с.

12. CITForum. Безопасность IP-телефонии – полевые зарисовки. А. Веселов [Електронний ресурс] — Режим доступу: citforum.ru/security/articles/ipsec (дата звернення 06.06.2021 р.) – Назва з екрана.

13. Распознавание речи [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/226143/ (дата звернення 06.06.2021 р.) – Назва з екрана.