ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка
до магістерської роботи
на ступінь вищої освіти магістр

на тему: «УДОСКОНАЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ОЦІНКИ ТА ПІДВИЩЕННЯ ФУНКЦІОНАЛЬНОЇ СТІЙКОСТІ МЕРЕЖІ ЗА ДОПОМОГОЮ ТЕОРІЇ ГРАФІВ»

Виконав: студент 6 курсу, групи ПДМ–61
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Балашова Є.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

БАЛАШОВІЙ ЄЛИЗАВЕТИ ОЛЕКСАНДРІВНІ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Удосконалення інформаційної технології для оцінки та підвищення функціональної стійкості мережі за допомогою теорії графів»

Керівник роботи: Жебка В.В., д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом закладу вищої освіти від «11» жовтня 2021 року №170.
2. Строк подання студентом роботи_____
3. Вхідні дані до роботи
Методи обробки зображень;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо розпізнавання тексту з зображень;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Системи розпізнавання та вилучення текстової інформації з зображень.
4.2 Вимоги та оцінка якості системи.
4.3 Опис проектування системи.

4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність проблеми
2. Існуюче програмне забезпечення та методи розпізнавання
3. Принцип роботи інформаційної системи
4. Розпізнавання та групування даних конкретних документів
5. Архітектура бази даних
6. Логічна діаграма компонентів архітектури програмного забезпечення

6. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		Виконано
2	Вимоги до системи		Виконано
3	Створення головного модулю програми		Виконано
4	Розробка візуальної складової додатку		Виконано
5	Концепція та архітектура програмного забезпечення		Виконано
6	Вступ, висновки, реферат		Виконано
7	Розробка обов‘язкових демонстраційних матеріалів		Виконано
8	Попередній захист роботи		Виконано
9	Здача роботи		Виконано

Студент __ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 67 с., 36 рис., 20 джерел.

ГЕОІНФОРМАЦІЙНІ МЕРЕЖІ, СИСТЕМИ МОНІТОРИНГУ, ФУНКЦІОНАЛЬНА СТІЙКІСТЬ МЕРЕЖІ, ZABBIX, GRAFANA, ТЕОРІЯ ГРАФІВ, НАДІЙНІСТЬ МЕРЕЖІ

Об'єкт дослідження – стійкість мережі.

Предмет дослідження – додаток на основі удосконаленої інформаційної технології стійкості мережі.

Мета роботи – моніторинг системи в режимі реального часу за допомогою програмного продукту за домопогою теорії графів.

Методи дослідження – методи розпізнавання та групування інформації, емпіричні методи.

У роботі проведено аналіз існуючих математичних моделей для розрахунку зв'язності шляху з використанням теорії ймовірності, детальний опис моделей представлений у роботі.

Загальною проблемою існуючих моделей полягає в тому, що розрахунки здійснюються за довготривалий час для сучасних телекомунікаційних мереж, що може стати загрозою для постачальників послуг відносно втрати споживачів, та надмірним навантаженням на канали зв'язку через збої.

Удосконалення існуючих методів полягає в використанні для розрахунку функціональної стійкості мережі теорії графів, що значно спрощує складність розрахунків, та оптимізує загальний час на виявлення збою у мережі. Та розробка автоматизованої системи моніторингу телекомунікаційної мережі, з використанням вхідних даних. Для розробки за основу було взято такі програми як Zabbix та Grafana.

Отже, розроблено та описано систему моніторингу, завданням якої є постійне спостереження телекомунікаційної мережі в режимі реального часу, відображення всіх елементів графу та зв'язку між ними, та удосконалено існуючі математичні моделі з використанням теорії графів.

Дана система може бути використана у всіх сферах, де побудова мережі схожа на графоподібну.

Галузь використання – розгалужені інформаційні системи.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

ІС – інформаційна система;

ФС – функціональна стійкість;

МД – модуль модель;

РІМ – розгалужена інформаційна мережа;

ПЗ – програмне забезпечення;

ІТС – інформаційно-телекомунікаційна система.

ВСТУП

Практично будь-яка дія людини супроводжується певною дією, пов’язаною з телекомунікаціями. Часто, здійснивши певну дію, ми навіть не замислюємося, скільки технічних операцій за ними стоїть. Розплатившись на касі за товар, написавши електронний лист, забронювавши квиток на літак або здійснивши звичайний дзвінок по телефону, ми задіяли певні послуги телекомунікацій.

Однією з найбільш успішних та затребуваних телекомунікаційних послуг у наш час є, звичайно, доступ до всесвітньої глобальної мережі Інтернет. Підключення до Інтернету, яке зовсім небагато років тому здавалося привілеєм обраних, сьогодні доступне практично кожному громадянину будь-якої держави, за винятком деяких країн третього світу. Мережа Інтернет практично поглинає всю Землю, використовуючи при цьому Інтернет-провайдерів як знаряддя такого поглинання. Кожен Інтернет-провайдер волею чи неволею стає вирішальним фактором у процесі проникнення інформаційних технологій у наше повсякденне життя.

Інформаційна інфраструктура сучасних підприємств є досить складним механізмом різномаштабних і різнорідних мереж і систем. Для забезпечення їх ефективної та постійної роботи, необхідна ІС корпоративного розміру, що має можливість керувати, з інтегрованими інструментальними засобами.

Функціонування великих інформаційних систем (ІС), призначених для автоматизації управління, відбувається у постійній взаємодії із зовнішнім середовищем. Найважливішою складовою гарної роботи підприємства, що надає послуги телекомунікацій, залежить від стійкості її мережі до будь-яких факторів, що можуть завдати шкоду мережі ззовні або зсередини. Такі (інформаційні) конфлікти призводять до руйнування інформаційних ресурсів, порушення штатних інформаційних процесів, та як наслідок зриву виконання системних та прикладних функцій. Все це визначає наявність в ІС механізмів, які мають забезпечувати нову якість – здатність збереження та/або відновлення даних (стійкість) функцій за умов різноманітних несприятливих впливів.

Дану якість назвемо функціональною стійкістю (ФС) ІС. ФУ є інтегральною властивістю, що включає надійність, живучість та безпеку. Відповідно оцінка показників ФС, необхідна для порівняння різних варіантів проектування, є складним науково-практичним завданням. Ще складніше пошук найкращого варіанти, при якому досягаються оптимальні показники ФС при деяких обмеженнях. У цьому напрямі отримано ряд науково-практичних результатів, але відсутність навчально-методичної літератури з цієї важливої проблеми призводить до певних недоліків у підготовці фахівців, які займаються створенням великих ІС.

На сьогоднішній день існує декілька математичних моделей, які можуть визначати стійкість мережі, проте, для сучасних телекомунікаційних мереж такі моделі не підходять, адже вони не розраховані на велику кількість споживачів, або мають надто довгі алгоритми розрахунку на зв'язність та стійкість мережі, та на сьогоднішній день не мають програмної реалізації.

Основним завданням даної магістерської роботи є оптимізація та удосконалення сучасних математичних моделей теорії стійкості, та розробка системи моніторингу телекомунікаційної мережі.

Було прийнято рішення, що для поставленої мети необхідно виконати наступне:

- проаналізувати сучасні математичні моделі з теорії стійкості;
- удосконалити існуючі методи з використанням теорії графів;
- провести аналіз сучасних систем моніторингу та обрати найбільш відповідну для розробки;
- розробити систему моніторингу з використанням програмного модуля з удосконаленою математичною моделлю на основі теорії графів;
- описати методику розробки системи моніторингу;
- розкрити суть удосконалення математичної моделі.

Об'єкт дослідження є стійкість мережі.

Предмет дослідження це додаток на основі удосконаленої інформаційної технології стійкості мережі.

Метою роботи є моніторинг системи в режимі реального часу за допомогою програмного продукту за домопогою теорії графів.

Завданням роботи є розробка системи моніторингу в режимі реального з удосконаленою інформаційною технологією з використанням теорії графів.

РОЗДІЛ 1. ФУНКЦІОНАЛЬНА СТІЙКІЙСТЬ НА ОСНОВІ ТЕОРІЇ ГРАФІВ

1.1. Теорія графів

Нехай дано множину $V = \{v_1, v_2,\ldots, v_n\}$ і нехай на множині V визначено сімейство $U = \{u_1, u_2,\ldots, u_n\}$ пар елементів $U_k = \{v_i, v_j\}$, $(k = 1, m)$ довільною кратності та впорядкування. Пара $\{V, U\}$ називається графом. Граф, як правило, позначають великими латинськими літерами, наприклад, G, H. Прийнято також писати $G\ (V, U)$ для того, щоб визначити деякий конкретний граф G.

Елементи $\{v_1, v_2,\ldots, v_n\}$ називають вершинами графа, а парами $U_k = \{v_i, v_j\}$, $(k = 1, m)$ – ребра. Графу можна дати таку інтерпретацію. Нехай маємо опис графа:

$G\ (V, U) = \{\{v_1, v_2,\ldots, v_6\}, \{\{v_1, v_3\}, <v_5, v_1>, \{v_3, v_4\}, <v_2, v_3>, \{v_3, v_3\}\}\}$

Цей опис можна відобразити графічно, як показано на рис. 1.1. На цьому рисунку деякі ребра відзначені стрілкою, а відповідні ним пари вершин в описі графа виділені фігурними дужками. Це пов’язано з тим, що з деякого довільного ребра можна брати чи не брати до уваги порядок розташування його кінців.

Якщо цей порядок не суттєвий, то ребро називається неорієнтованим, в іншому випадку ребро називається орієнтованим. Орієнтовані ребра називаються також дугами.

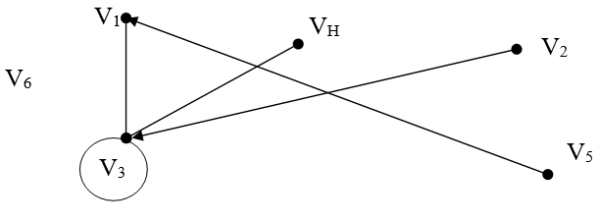


Рисунок 1.1 – Графічне відображення графа

Для орієнтованого ребра визначено поняття початкової та кінцевої вершини. Початкова вершина записується на початку пари вершин, що визначають дугу, а

кінцева – наприкінці. Так, на рис.1.1 ребра <v_5, v_1> та <v_2, v_3> є дугами. Вони представлені впорядкованими парами вершин і у зв'язку з цим позначені в такий же спосіб, як позначаються впорядковані множини.

Ребра $\{v_1, v_3\}$, $\{v_3, v_4\}$ неорієнтовані. Для будь-якого неорієнтованого ребра вважають, що $\{v_i, v_j\} = \{v_j, v_i\}$. Ребро $\{v_i, v_i\}$ називається петлею. Петлі зазвичай неорієнтовані.

Граф, у якого ребра неорієнтовані, називається неорієнтованим.

Граф, у якого ребра орієнтовані, називається орієнтованим чи орграфом.

Графи і орграфи — суттєво різні об'єкти, у певних випадках графи можна як орграфи, у яких кожному ребру відповідають дві протилежно орієнтовані дуги.

На рис.1 наведено змішаний граф, тобто, він містить як орієнтовані, так і неорієнтовані ребра, та ще містить у собі петлю.

Вершина ступеня 0 називається ізольованою вершиною, на рисунку 1 це вершина v6, вершина ступеня 1 називається висячою (або кінцевою) вершиною. Граф, у якого всі вершини ізольовані, називається нуль-графом. Багато ребер, що належать до такого графа вважають порожніми.

Антипод нуль-графа є повний граф. Ребрами повного графа є всі можливі пари його вершин.

Як у разі орієнтованого графа, так і у разі неорієнтованого, кажуть, що ребро інцидентне парі вершин, які його визначають.

Одній і тій же парі вершин можна поставити відповідно безліч інцидентних ребер. Такі ребра називаються кратними. Так, на рис. 1.2 представлені різні пари вершин (vi, vj), інцидентні трьом різним ребрам u1, u2, u3. Одне з них направлене, а два – ні.

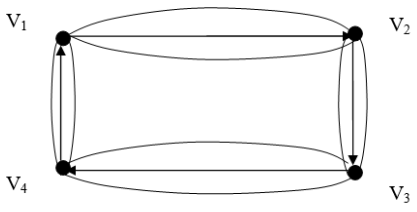


Рисунок 1.2 – Інцидентний граф

Якщо у зв’язному графі не має циклів, то цей граф вважають деревом. Тобото, дерево у своєму складі не має петель та кратних ребер. Граф у якому не має циклів називають графом, у складі якого зв’язні складові дерева; також можна зустріти назву такого графу як ліс. Якщо ланцюг у графі не має циклів, то такий граф простий. Та якщо обрати будь-яку частину цього графу, то вона теж вважається графом без циклів.

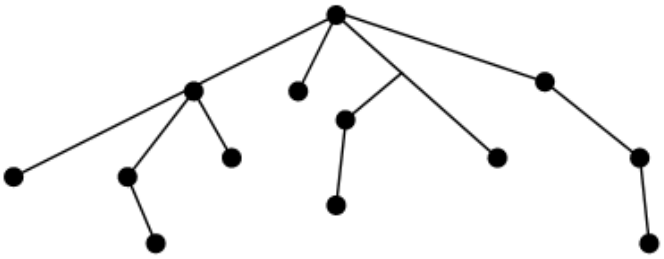


Рисунок 1.3 – Дерево

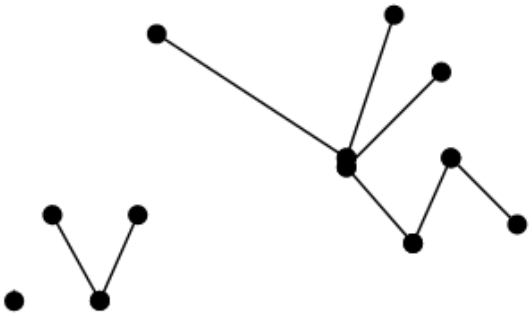


Рисунок 1.4 – Ліс

Приклад дерева показаний на рис. 1.3, також відображені висячі вершини, на малюнку вони виділені зафарбованими точками (вершина дерева, ступінь якої дорівнює одиниці, називається висячою вершиною). Приклад лісу показано на рис. 1.4.

Якщо побудувати довільне дерево з, наприклад, п’ятьма вершинами і порахувати кількість ребер в отриманому графі, то в результаті отримаємо чотири ребра. Виявляється, що в будь-якому дереві з п'ятьма вершинами завжди буде чотири ребра.

Теорема: дерево з n вершинами має $n-1$ ребро.

Для того, щоб з будь-якого дерева Р, що не вважається ізольованою вершиною, отримати ще два дерева, у складі якого будуть ті ж самі вершини, необхідно видалити з Р одне ребро. Для утворення трьох дерев необхідно видалити з G два будь-яких ребра. Найбільше з дерева G з n вершинами можна отримати n

дерев, кожне з яких є ізольованою вершиною. Для цього необхідно видалити $n-1$ ребро з дерева G. Отже, дійсно, у дереві з n вершинами $n-1$ ребро.

На сьогоднішній день самим популярним способом представлення графу полягає в графічному зображенні точок (вершин) та ліній (ребер). Але якщо треба зробити розрахунки програмно, то таке відображення не підходить для обчислювальних машин, в такому разі граф задається дискретним способом, проте дискретним способом також можна представити у багатьох видах. Проте простота та легкість використання відображення графа, як і ефективність алгоритму, в основі якого він знаходиться, напряму залежить від того, в який спосіб він представлений. Один із найпопулярніших напрямів, який використовується в теорії графів, це матричне відображення. Існують різні види матриць, що визивають асоціацію з графами. Ці форми алгебри використовуються для вирішення багатьох завдань теорії графів.

Матрицею суміжності графа G з будь-якою кількістю вершин {v1, ..., vn} (відповідної даної нумерації вершин) називається матриця $A = (a_{ij})$, в якій елемент aij дорівнює числу ребер G, що з'єднують v_i і v_j. Наприклад, на рис. 1.5 дано матрицю суміжності деякого графа. Можна отримати кілька різних матриць суміжності заданого графа, змінюючи позначення його вершин; це приведе до того, що у матриці буде змінено порядок рядків і стовпців. Але в результаті завжди вийде симетрична матриця з невід'ємних цілих чисел, що мають властивість, що сума чисел у будь-якому рядку або стовпці дорівнює ступеню відповідної вершини (тут кожна петля враховується в ступеня вершини один раз).

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Рисунок 1.5 – Матриця суміжності

Назад, за будь-якою заданою симетричною матрицею з невід’ємних цілих чисел легко побудувати граф (єдиний з точністю до ізоморфізму), що має дану матрицю своєї матриці суміжності. Звідси випливає, що теорію графів можна звести до вивчення матриць особливого типу.

1.2. Вершинна та реберна зв’язність графа

Вершинна та реберна зв’язність графа — перші кількісні показники, за допомогою яких почали вимірювати цілісність графів.

Число вершинної зв'язності $\varkappa(G)$ графа $G = (V, E)$ визначається наступним чином:

$$\varkappa(G) = \min_{S \subset V} \{|S| : k(G - S) > 1 \text{ або } G - S = O_1\}, \quad (1.1)$$

тобто це найменша кількість вершин, видалення яких призводить до незв'язного або тривіального графу. Інакше кажучи, число $\varkappa(G)$ визначає розмір меншого сепа ратора графа G. Виняток становить лише граф Kn, у якому немає сепараторів загалом, хоча $\varkappa(Kn) = n - 1$. Граф l-зв'язний, якщо $\varkappa(G) > l$. Відповідно до теореми Менгера, $\varkappa(G) > l$ тоді й тільки тоді, коли будь-яка пара незбігаючих вершин графа G з'єднана принаймні l вершинно-непересічними простими ланцюгами.

Аналогічним чином визначається число реберної зв'язності $\mu(G)$ графа $G = (V, E)$:

$$\mu(G) = \min_{S \subset E} \{|S| : k(G - S) > 1\}, \quad (1.2)$$

тобто $\mu(G)$ — це найменша кількість ребер, видалення яких із G призводить до незв’язного графа. За визначенням число $\mu(G)$ визначає розмір найменшого розрізу графіка G. Граф реберно-зв’язний, якщо $\mu(G)$> l. З реберного варіанта теореми Менгера випливає, що $\mu(G)$> l тоді й тільки тоді, коли будь-яка пара

несхожих вершин графа G пов’язана не менш ніж l реберно-непересічними простими ланцюгами.

Значення числових параметрів $\varkappa(G)$ і $\mu(G)$стосовно проблеми цілісності графа можна інтерпретувати так: це обсяг пошкоджень (число елементів або зв’язків між елементами графа), після видалення яких граф втрачає здатність підтримувати зв'язок між деякими парами вершин. Тому що вище значення $\varkappa(G)$ та $\mu(G)$, тим більш цілісним є граф G. Слід зазначити, що в даному випадку не враховується, скільки компонент містить граф $G - S$ і які розміри цих компонентів. При цьому для будь-якого незв’язного графа $\varkappa(G) = \mu(G) = 0$.

Заходи зв’язності $\varkappa(G)$ і $\mu(G)$ добре вивчені теорії графів. Зокрема, встановлено, що для будь-якого графа $G = (V, E)$, $n = |V| > 1$, вірне відношення

$$0 \leq \varkappa(G) \leq \mu(G) \leq \delta(G) \leq n - 1 \qquad (1.3)$$

Наприклад, $\varkappa(Kn) = \mu(Kn) = \delta(Kn) = n - 1$ і $\varkappa(On) = \mu(On) = \delta(On) = 0$. Можна побачити, що завдання обчислення $\varkappa(G)$ та μ(G) для довільного графа G зводиться до задачі про максимальний поток та мінімальний розріз і є розв’язаною за поліноміальний час. Проте вершинна та реберна зв'язність знаходять лише обмежене застосування в аналізі стійкості до відмови систем, оскільки в них враховується лише обсяг пошкоджень і ігноруються багато важливих факторів, пов'язаних з наслідками пошкоджень. В даний час для обліку подібних факторів запропоновано та вивчено інші числові параметри графа.

Для кількісної оцінки цілісності графів вже запропоновано та вивчено багато заходів, виражених через числові параметри графа. Завдання обчислення відомих на сьогоднішній день детермінованих заходів цілісності та відповідних їм екстремальних множин графа носять комбінаторний характер і є, як правило, NP-важкі. Дослідження проблеми цілісності графів ведуться за такими основними напрямками:

- вдосконалення існуючих та розробка нових моделей ушкодження, та як наслідок, створення та вивчення нових заходів цілісності;

- введення зважених версій відомих заходів;
- аналіз завдань обчислення заходів цілісності та відповідних їм екстремальних множин графа в рамках класичної та параметричної теорії складності обчислень, виявлення поліноміально розв'язних випадків для цих завдань;
- встановлення нижніх і верхніх оцінок заходів цілісності через традиційні чисельні параметри графа, такі, як число вершинного покриття, хроматичне число та інші;
- характеризація класів графів, що задовольняють заданому рівню цілісності;
- отримання формул обчислення заходів цілісності для вузьких класів графів.

Розробка та застосування обчислювальних систем у критичних областях, а також стрімкий розвиток різних видів зв'язку стимулюють пошук нових моделей, методів і засобів створення відмовостійких систем. Актуальні моделі з пошкоджень графа, що відображають ієрархічність побудови систем та управління ними.

Для деяких відомих заходів цілісності, як і раніше, відкриті питання обчислювальної складності стосовно деяких класів графів. Затребувані алгоритми обчислення точних і наближених значень для існуючих заходів цілісності.

1.3. Обчислювальна складність

Схематично введемо основні поняття відповідно до робіт (Гері, Джонсон, 1982; Пападимитру, Стайгліц, 1995) [9], опускаючи деякі формальні деталі.

Під масовим завданням розуміється деяке спільне питання, на яке потрібно дати певну відповідь. Наприклад, знайти вершинне k-розширення графа. Також завдання містить певні параметри, наприклад граф, для якого потрібно знайти вершинне k-розширення. Індивідуальне завдання виходить із масової, якщо всім параметрам задачі вказано конкретні значення. У нашому разі це означає завдання конкретного графа.

Під алгоритмом розуміємо покрокове розв’язання задачі. Говорять, що алгоритм вирішує масове завдання $П$, якщо він застосовуємо до будь-якої індивідуальної задачі з $П$ і дає її розв'язання. Ефективність різних алгоритмів, що вирішують це завдання, зазвичай оцінюється з погляду витрати часових та ємнісних ресурсів. Часові витрати є важливішими з практичної точки зору, і тому переважно розглядається саме тимчасова ефективність алгоритму.

Час роботи алгоритму зручно виражати функцією від змінних, що характеризують розмірність індивідуального завдання. Вибирається деяка схема кодування вхідних даних для завдання, обсяг цих вхідних даних використовується як характеристика розміру індивідуального завдання.

Часова складність алгоритму характеризує час, потрібний його роботи. Це функція, яка кожному значенню розмірності індивідуального завдання ставить у відповідність максимальний час, що витрачається алгоритмом на вирішення індивідуальних задач розмірності n.

Зазвичай часова складність описується спеціальними термінами, що вказують швидкість зростання.

Говорять, що $f(n) = O(g(n))$, якщо знайдеться така константа $c > 0$ і таке число n0, що $0 \leq f(n) \leq cg(n)$ для всіх $n \geq n_0$.

Поліноміальним алгоритмом (або поліноміальним алгоритмом тимчасової складності) називається алгоритм, у якого тимчасова складність дорівнює $O(p(n))$, де $p(n)$ – певна поліноміальна функція, а n – араметр, що визначає розмірність задачі. Алгоритми, часова складність яких не піддається такій оцінці, називаються експонентними. Більшість експоненційних алгоритмів представляє собою перебір всіх можливих варіантів розв'язання. Кажуть, що завдання можна розв’язати за поліноміальний час, якщо існує поліноміальний алгоритм для її вирішення. Традиційно поліноміальні алгоритми вважаються хорошими або ефективними, а завдання, для яких не існує поліноміальних алгоритмів розв’язання, називаються важкорозв’язними.

Для графів як характеристика розмірності індивідуальної завдання зазвичай використовуються або кількість вершин n, або кількість ребер m. Для кодування

графа зазвичай використовується матриця суміжності, що вимагає n^2 біт, або список ребер, що вимагає $m \log_2 n$ біт. При «розумних обмеженнях» схеми кодування для одного завдання не відрізняються більш ніж «поліноміальним чином», тобто будь-який алгоритм, що має поліноміальну часову складність при одній схемі кодування, буде також володіти поліноміальною тимчасовою складністю при інших схемах кодування. Таким чином, з'ясування питання, чи це завдання важко вирішуване, не залежить від вибору конкретної схеми кодування.

Серед труднорозв'язних завдань можна виділити два аспекти. Перший аспект полягає в тому, що для пошуку рішення потрібен експонентний час. Другий у тому, що довжина самого рішення неспроможна бути представлена поліномом від розмірності задачі. Наприклад, така ситуація може мати місце при пошуку всіх вершинних розширень для довільного графа.

Основи теорії NP-повних завдань заклав С. Кук у роботі 1971 р., де довів, що завдання здійсненності булевої формули є NP-повною. Потім Карп довів NP-повноту 21 завдання. У книзі Гері та Джонсона (Гері, Джонсон, 1982) [1] міститься вже понад 300 NP-повних завдань.

Так, серед завдань теорії графів можна відзначити такі NP-повні завдання: завдання про клік, про гамільтоновий цикл, про вершинне покриття. Теоретично складності розглядаються завдання розпізнавання властивостей або завдання прийняття рішень, тобто такі завдання, рішеннями у яких є відповідь «так» чи «ні». Клас P[26] визначається як клас завдань прийняття рішень, дозволених за поліноміальний час. Неформально клас NP[27] можна визначити як клас завдань ухвалення рішень, для яких за поліноміальний час можна перевірити правильність рішення за наявності сертифіката рішення. Клас P очевидно входить до класу NP:

P ⊆NP. Якщо P ≠ NP, то завдання з NP \ P є розв'язними.

Особливий інтерес становить клас NP-повних завдань. Завдання A називається NP-повним, якщо A належить класу NP та будь-яке інше завдання NP може бути поліноміально зведена до A.

1.4 Булевий гіперкуб

Булевий простір M можна представити у вигляді графа, вершини якого відповідають елементам простору, а ребра представляють відношення між елементами простору. Два вектори є сусідніми, якщо вони відрізняються один від одного значенням однієї компоненти. Наприклад, вектори (10 0 1) і (11 0 1), значення однойменних компонент яких, крім однієї другої компоненти, збігаються, є сусідніми. Даний граф, що представляє n-мірне булевий простір, має 2^n вершин і $n2^{n-1}$ ребер. Він називається повним булевим графом, або розмірним гіперкубом. Розглянемо побудову такого гіперкуба для різних значень розмірності простору.

Одномірний гіперкуб складається із двох вершин, що пов'язані ребром. Однією з цих вершин приписується константа 0, а іншій – константа 1, які є кодами даних вершин. Щоб отримати двовимірний гіперкуб, треба продублювати одномірний гіперкуб і кожну вершину вихідного гіперкуба з’єднати ребром із дублем. Коди вершин побудованого двовимірного гіперкуба виходять додаванням нулів праворуч до кодів вершин вихідного гіперкуба та одиниць – до кодів дублів вершин. Аналогічно виходять тривимірний гіперкуб, чотиривимірний гіперкуб і т. д. Послідовність гіперкубів від одновимірного до чотиривимірного представлена на рис. 1.6.

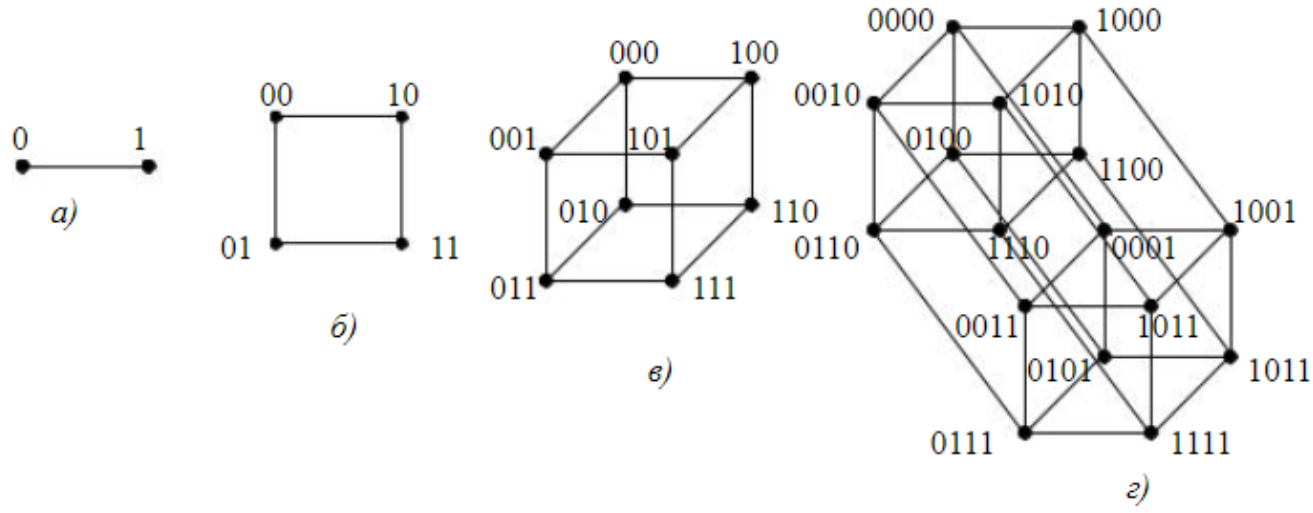


Рисунок 1.6 – Графічне представлення булевого простору: *а)* одномірне; *б)* двумірне; *в)* тримірне; *г)* чотиривимірне

Сформулюємо загальне правило збільшення розмірності гіперкуба: для переходу від m-мірного гіперкуба до $(m + 1)$-мірного треба вихідний m-мірний гіперкуб продублювати та кожну вершину вихідного гіперкуба з'єднати ребром з її дублем. В отриманому гіперкубі до кодів вершин вихідного розмірного гіперкубу додаються справа нулі, а до кодів їх дублів – одиниці.

У гіперкубі виділяються гіперграні, які є породженими підграфами, що є гіперкубами меншої розмірності, ніж гіперкуб, який розглядається. Це може бути окреме ребро, двовимірна грань, тривимірний куб і т. п. Підграф, що представляє гіперграні, породжується безліччю вершин, що становлять інтервал булевого простору.

Булев n-куб є одним із центральних об'єктів дослідження у теорії графів, теорії кодування, дискретному аналізі та ряді інших областей математики Інтерпретація безлічі вершин цього графа як множини всіх можливих підмножин робить граф Qn та його дослідження актуальними для багатьох областей знань. Останнім часом інтерес до вивчення даного графа стрімко зростає у зв'язку з появою паралельних суперкомп'ютерів, структура мережі яких представляє собою гіперкуби різної розмірності. Це не тільки забезпечує нову потенційну область програми для результатів на цю тему, але і також ставить нові завдання, які потребують дослідження. Основні властивості даного графа наведено у огляді Ф. Харарі зі співавторами.

Через актуальність і широку сферу застосування даного графа, особливий інтерес представляють різноманітні фундаментальні властивості Qn, серед яких крім різних інваріантів теорії графів є менш очевидні. До таких фундаментальних властивостей, наприклад, може відноситися існування в Qn шляхів, що володіють різними локальними властивостями.

1.5 Гамільтонові графи

Граф називається гамільтоновим, якщо в ньому є гамільтонів цикл, тобто простий цикл, що містить всі вершини цього графа. До проблеми гамільтоновості

графів, а також до її виваженого аналога – задачі про комівояжер – проявляється стійкий інтерес протягох багатьох років, а дослідження цих завдань є одним із магістральних напрямків теорії графів та комбінаторної оптимізації. Завдання про гамільтоновий цикл полягає у відповіді на питання, чи є граф гамільтоновим. Це завдання NP-повне в загальному випадку і залишається NP-повним у багатьох вузьких класах графів. З іншого боку, відомий ряд достатніх умов, коли завдання можна вирішити за поліноміальний час. Дослідження «областей ефективності» завдання про гамільтоновий цикл - класів графів, для яких вона може бути вирішена за поліноміальний час, - має не тільки теоретичний, а й практичний інтерес.

Любий граф G можна зробити гамільтованим, якщо додати до нього визначену кількість вершин. Для цього, для прикладу, достатньо до вершин $v1, \dots, vp$ графа G додати вершини $u1, \dots, up$ та ребера $\{(vi, ui)\}\,\{(ui, vi+1)\}$.

Ступінь вершини v це число ребер $d(v)$, інцидентних їй, у петлях враховується два рази. Якщо граф орієнтований, то в нього буде ступінь $d_0(v)$ по дугах, що виходять, і $d_i(v)$ — по вхідних.

Якщо порівнювати з ейлеровими графами, де є конкретна умова для графа бути ейлеровим, для гамільтонових графів такого критерію немає. Проте, завдання перевірка на існування гамільтонового циклу виявляється NP-повною. Більшість відомих теорем приймають вигляд: «якщо граф G має достатньо ребер, то граф є гамільтоновим».

1.6. Відмовостійкість

У техніці велике значення мають високонадійні системи, які можуть зберігати працездатність, навіть якщо деякі компоненти виходять з ладу. Перші комп'ютери, які з'явилися торік у 1940-х роках, відрізнялися своєю ненадійністю (Carter, Bouricius, 1971). Наприклад, ENIAC12 – перший електронний цифровий комп'ютер був побудований у 1946 і складався з майже 18000 ламп. Його середній час напрацювання на відмову становив 7 хвилин, а за 4 роки роботи завдання

вирішувалися правильно у 54% випадків. У наступних комп'ютерах надійність значно зросла. Розробка EDVAC13 була завершена в 1949 р. за участю фон Неймана. На відміну від ENIAC, це був комп'ютер, заснований на двійковій системі числення, а не на десятковій. В EDVAC використовувалося 2 АЛУ14 результати роботи яких порівнювалися для виявлення відмови. UNIVAC I15 виник 1951 р. У ньому також було реалізовано дублювання елементів, крім цього використовувалися коди з корекцією помилок та контролем парності для передачі та зберігання інформації.

Пройшло понад 60 років з моменту появи комп’ютерів. Сучасні процесори будуються на значно надійнішій основі, інші складові частини комп'ютера також є надійнішими, ніж раніше.

Наприклад, час напрацювання на відмову сучасних процесорів вимірюється роками. Проте найпотужніші суперкомп’ютери містять десятки тисяч процесорів, уникнути збоїв та відмов компонентів таких систем неможливо. Авіженіс (Avižienis, 1975) запропонував два підходи підвищення надійності реальних обчислювальних систем: запобігання помилкам та відмовостійкість. Перший напрямок пов'язаний із зменшенням ймовірності виникнення помилки та полягає у розробці високонадійних компонентів системи. У другому напрямку використовується введення в систему надлишкових структур для надання їй додаткових властивостей відмовостійкості.

У наступні роки дослідження у сфері надійності обчислювальних систем проводилися дуже інтенсивно. Деякий час тому (Laprie, 1985; Dependability …, 1992; Avižienis et al., 2004; Харченко, 2006; Теслер, 2008; Харченко, 2009) стали розглядати більш загальне поняття, ніж надійність (reliability), - гарантоспроможність (dependability). У роботі (Avižienis et al., 2004) надається докладна таксономія для гарантоспроможності та супутніх понять.

Під системою розуміється сутність, яка взаємодіє з іншими сутностями, зокрема з іншими системами. Обчислювальні системи характеризуються своїми фундаментальними властивостями: функціональністю, продуктивністю, гарантоспроможністю, безпекою та вартістю. Функціональна специфікація описує

функції системи з погляду функціональності та продуктивності. Під поведінкою (behavior) системи розуміється те, як вона реалізує свої функції. Послуги, що поставляються системою (service) – це поведінка системи з точки зору її користувачів, тобто споживачів послуг. Обслуговування вважається правильним, якщо надані послуги відповідають функцій системи.

Під гарантоспроможністю (dependability) розуміється комплексна властивість системи надавати послуги, яким можна виправдано довіряти. Властивістю гарантоспроможності є:

- готовність (availability) - готовність до правильного обслуговування;
- надійність (reliability) - безперервність правильного обслуговування;
- безпека (safety) - відсутність катастрофічних наслідків для користувачів та навколишнього середовища;
- цілісність (integrity) - відсутність некоректних змін системи;
- обслуговуваність (maintainability) - здатність піддаватися модифікаціям та ремонту.

Загрозами гарантоспроможності є збої, помилки та відмови.

Відмова (failure) означає, що послуги, що надаються системою, відрізняються від правильних. Відмова означає, що поведінка системи в одному або більше станів відрізняється від очікуваної поведінки при правильному обслуговуванні. Ця відмінність називається помилкою (error). Встановлена або гіпотетична причина помилки називається збоєм (fault). Збій системи може бути внутрішнім та зовнішнім. Внутрішній збій може не призводити до помилки та відмови системи.

Відмовостійкість (fault tolerance) означає можливість уникнути відмови системи у присутності збою. Очевидно, що стійкість до відмови є найважливішою передумовою для гарантоспроможності великих систем. Як інші засоби досягнення гарантоспроможності відзначаються: запобігання збоям (fault prevention), усунення збоїв (fault removal) та передбачення збоїв (fault forecasting).

Вважається, що перша теоретична робота з дослідження відмовостійкості належить фон Нейману (von Neumann, 1956). Він запропонував і досліджував концепцію потрійної модульної надмірності, яка й досі не втратила актуальності.

Поняття відмовостійкості було введено Авіженісом (Avižienis, 1967; Авіженіс, 1978) як забезпечення системи здатністю протистояти помилці та можливістю продовжувати роботу в присутності цієї помилки. Виділяють два рівні відмовостійкості:

- повна стійкість до відмови – система продовжує працювати в присутності помилок без суттєвої втрати функціональних властивостей;
- амортизація відмов – система продовжує працювати у присутності помилок із частковою деградацією функціональних можливостей.

Якщо говорити про обчислювальні системи, то відмови можуть виникати як в апаратній (апаратурній), так і в програмній частині. У разі повної відмови стійкості елемент, що відмовив, повинен бути замінений справним, який візьме він виконання відповідних функцій. Це означає, що відмовостійка система повинна мати певну надмірність. Загалом, можна виділити кілька рівнів, де можуть виникати помилки і може знадобитися надмірність (Hayes, 1998):

- апаратна надмірність (hardware redundancy) - додаються копії критичних компонентів системи;
- програмна надмірність (software redundancy) - різні версії програм для виконання критичних операцій;

Іноді виділяють інші рівні. Так, у роботі (Теслер, 2008) автор наводить 13 видів надмірності, окремо виділяючи комунікаційну, алгоритмічну, архітектурну надмірність. Кожному із зазначених вище рівнів присвячені великі дослідження. Більше того, в рамках кожного рівня існує безліч різних моделей для дослідження та побудови відмовостійких систем. Так, у рамках апаратної надмірності виділяють статичні та динамічні моделі.

Статична надмірність означає, що надлишкові компоненти використовуються як стала система. Найпростішим прикладом є така схема (n-модульна надмірність): припустимо, що деякий модуль генерує сигнал X. Розмістимо $n \geq 3$ копій цього модуля, налаштувавши систему так, щоб усі модулі працювали паралельно. Виходи модулів подаються на вхід спеціального пристрою «голосування», виходом з якого є сигнал, що найчастіше зустрівся на вході. Таким

чином, помилка, породжена модулем, що відмовив, маскуватиметься модулем голосування. За $n = 3$ виходить потрійна модульна надмірність. На рис. 1.7 зображено 3 модулі M. З виходів X_1, X_2 та X_3 цих модулів вибирається такий, що зустрічається частіше. Пристрій голосування Voter у двійковому випадку описується добре відомою формулою:

$$X = X_1X_2 + X_1X_3 + X_1X_2 \quad (1.4)$$

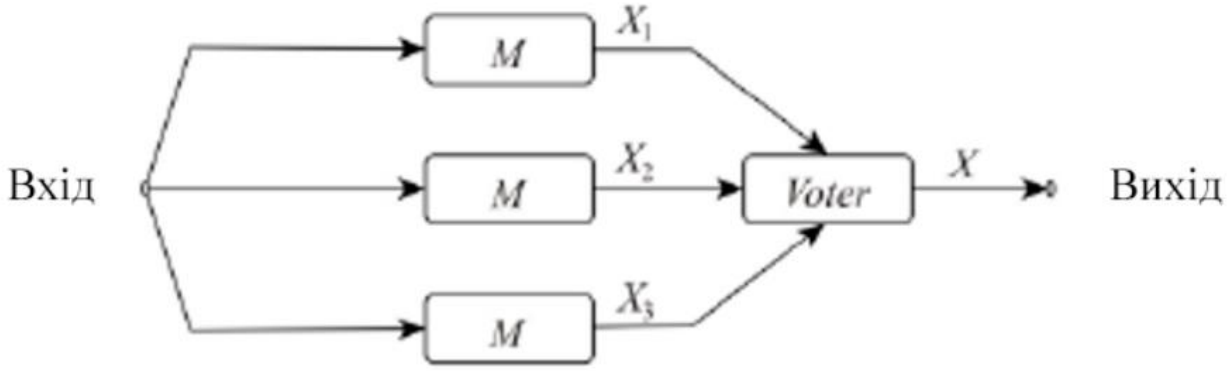


Рисунок 1.7 – Потрійна модульна надмірність

При динамічній надмірності система перебудовується так, що функції відмовив елемента передаються справному елементу. При цьому виділяються додаткові завдання:

1. Виявлення відмови – діагностичні процедури повинні дозволяти виявляти елемент, що відмовив.

2. Виняток відмови – елемент, що відмовив, виключається з системи і або замінюється справним, або система реконфігурується, щоб уникнути використання елемента, що відмовив.

3. Відновлення – виконується приведення системи до робочого стану.

РОЗДІЛ 2. ОПИС МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Аналіз існуючих моделей до оцінювання показників функціональної стійкості ІС за допомогою теорії графів

В основних роботах з даної теми увага приділяється основам теорії надійності: поняттям, визначенням та постулатам, докладної класифікації відмов, характеристик надійності при раптових та поступових відмовах. У зазначених джерелах виявлено базові характеристики надійності як показники безвідмовності, ремонтопридатності, довговічності, збереження. Розглянуто загальні методи розрахунку надійності технічних систем різного призначення як нерезервованих, і резервованих.

В наукових роботах Барабаша О.В. [2] ймовірність зв'язності мережі показано, як один з основних показників функціональної стійкості РІМ. Метод спирається на теорію ймовірностей, яка використовується для ймовірності зв'язності конкретної підмножини вершин $v1, \dots, vk$, тобто алгоритм перевіряє ймовірність того, що в заданий момент часу між кожною заданою парою вершин буде існувати хоча б один маршрут при заданих ймовірностях наявності ребер. Модель має два варіанти: $k = 2$ і $k = n$, де n – кількість вершин мережі. Деякі автори проводили дослідження, де розглядали в яких в якості показника середню ймовірність на існування хоча б одного маршруту серед всіх пар ребер. Існує поліном зв'язності, за допомогою якого можна побачити ймовірність зв'язністі та функціональну стійкість мережі, зробити певні висновки при порівнянні ФС у різних структурах. Зазвичай цей поліном використовують у вигляді:

$$P_{\text{зв.}} = p^m + \sum_{i=1}^{m-n+1} a_i (1-p)^i \, p^{m-i} \qquad (2.1)$$

що відповідає розкладу за зв'язними суграфами із заданим числом ребер.

Багато іноземних та вітчизняних науковців досліджували та продовжують досліджувати зв'язність мережі за допомогою теорії ймовірностей. Найбільш відомі з них це Артамонова Г.Т., Вишневського В.М., Єпіхіна В.В., Кауля С.Б., Кельманса А.К., Литвака В.І., Ломоносова М.В., Нечепуренко М.І., Поллеского В.П., Скоробагатова В.А., Толчана А.Я., Харкевича А.Д., Мура Е. [Moor E.], Шеннона К. [Shennon K.], Ван Слайка Р. [Van Slyke R.], Шумана А. [Shooman A.].

В роботі Кауля С.Б. було проведено і узагальнено деякі роботи, та виведено оптимальні графи для перевірки різних показників надійності. Для аналізу зупинимося на цій моделі.

Математична модель. У роботі за основу взяті неорієнтовані графи Розглядаються неорієнтовані графи з рівнонадійними ребрами, тобто будь-яке ребро, що належить до графа G, може бути видалено з графу з ймовірністю p. Будемо вважати, що $G = (V, E)$ – граф з n вершинами V(G)={v1, v2, ..., vn} і m ребрами $E(G) = \{e1, e2, \ldots, em\}$, то через $G \backslash E'$ позначимо суграф графа G, $E' \subset E(G)$.

Тоді ймовірність зв'язності $P_{\text{зв.}}(G, p)$ графа G обчислюється за формулою:

$$P_{\text{зв.}}(G, p) = \sum_{E' \subset E(g)} f(G \backslash E') p^{|E'|} (1-p)^{m-|E'|} \qquad (2.2)$$

де $f(G \backslash E') = 1$, якщо суграф вважається зв'язним, та $f(G \backslash E') = 0$, якщо суграф не зв'язний.

Можна побачити, що ймовірність зв'язності мережі була досліджена багатьма науковцями та авторами. Найбільш розгорнуто розглянуто проблему в роботах Дудника Б.Я, Кельманса А.К., Полеського В.П. і Шумана А. В роботі Кельманса А.К. розглядаються три способи способи визначення величини $P_{\text{зв.}}$, які є чисельними, проте вони не дають змогу оцінити $P_{\text{зв.}}$ для заданої множини графів, або провести нормальну оцінку, якщо структура графу буде приймати якусь зміну. Один з методів дає змогу отримати асимптотичну оцінку ймовірність, що

досліджується, але якщо кількість вершин буде збільшуватись, то метод буде досить трудомістким.

Припустимо, що будь-яка мережа приймає вигляд графа $G = (V, E)$, де $V = n$ – це кількість вершин (вузлів зв'язку), $|E| = m$ - число ребер (каналів зв'язку). Використаймо метод декомпозиції, його суть полягає в тому, що граф розбивається на складові, тобто підграфи. Далі, у вихідному графі кожна складова міняється на підграф, який отримали раніше, до складу якого входить менше ребер та вузлів, ніж у підграфа, який замінили. Коли ми отримали новий граф G', необхідно знайти рішення, що буде оптимальним, або близьким йому. Такий підхід дає змогу швидко оцінити зв'язність великих графів. Подібний підхід дозволяє ефективно і швидко отримати оцінку (або навіть точне значення) для великих графів. Запропонований метод вважається хорошим, проте, не завжди вдається зробити заміну графа.

Другий підхід, який використовується у роботі Мура Э. та Шеннона К. допускає, що до кожного ребра є відповідна незалежна ймовірність пошкоджень p тоді $q = 1 - p,$ а $P_{\text{зв.}}(p)$ це ймовірність незв'язності мережі. З заданого графа необхідно видалити $m - k$ ребер. $\mathrm{C}(k)$ виражає позначає кількість незв'язних підграфів, що складаються з k ребер, яке буде отримано після видалення. Тоді формула приймає наступний вигляд:

$$\bar{P}_{\text{зв}}(p) = \sum_{k=0}^{m} C(k) p^{m-k} q^{k} \qquad (2.3)$$

Для обчислення $\bar{P}_{\text{зв}}(p)$ обираємо метод моделювання з такою логікою:

- спочатку буде створено послідовність, що складається з рівномірних випадкових величин, за допомогою яких буде видно , які ребра випали з мережі, або пошкоджені;
- перевірка отриманого графа на зв'язність;
- повторення попередніх двох етапів, обчислення середньої частки зв'язних графів. За допомогою цього методу можна отримати найбільш наближену ймовірність зв'язності мережі.

Третій підхід, що розглянуто у роботі Франка «Аналіз надійності мережі» вважається самим ефективним. Цей метод полягає у тому, що задається граф G з n вершинами. Тоді підграф вважається незв'язним , якщо він має менш ніж $n-1$ ребр.

Отже, отримаємо $\mathrm{C}(k) = C_m^k$ при $k = 0,1,\ldots,n-2$. Якщо величина мінімального перерізу буде дорівнювати u, то будь-який підграф, в складі якого кількість ребер менше u, буде зв'язним. Таким чином, $\mathrm{C}(m-k) = 0$ при $k = 0,1,\ldots,u-1$. Якщо ймовірність працездатності ребра близька до нуля, то поведінка функції $\bar{P}_{\text{зв}}(p)$ буде визначатися останнім ненульовим членом $\mathrm{C}(m-k)$ з (10). В такий же спосіб, коли p буде наближатись до одиниці, поведінка певної величини буде визначатися першим ненульовим членом $\mathrm{C}(n-1)$ з (10). Далі $\mathrm{C}(m-u)$дорівнює числу різних мінімальних перерізів, що належать графу G, а $\mathrm{C}(n-1)$ буде дорівнювати C_m^{n-1} мінус кількість дерев графа.

Отже, за допомогою цих трьох методів можна проводити аналіз на функціональну стійкість на основі теорії ймовірностей.

Проте дані методи були актуальними років сім назад, коли кількість споживачів була обмежена.

2.2. Програмна надійність

Визначимо надійність програмного забезпечення ІС як можливості комплексу програм виконувати задані функції, зберігаючи у часі значення встановлених показників у заданих межах. При експлуатації зміна надійності програмного забезпечення з часом істотно відрізняється від зміни надійності технічних засобів.

ПЗ не схильне до зносу, практично відсутні помилки виробництва, так як вони зустрічаються рідко і легко можуть бути виправлені.

Виділимо роботу В. Р. Матвєєвського [10], присвячену математичним моделям у теорії надійності. У ній наведено аналітичні залежності інтенсивності відмов від часу для розподілів Вейбулла, Релея та ін. Так, нормальний розподіл та

розподіл Вейбулла використовуються для розрахунку надійності «старіючих» елементів, а експоненційний розподіл, будучи приватним випадком розподілу Вейбулла – для розрахунку надійності «нестаріючих» елементів. Крім цього розглядаються заходи щодо формування показників надійності на різних стадіях проектування, методи підвищення надійності.

У навчальному посібнику С. М. Віхарьова велика увага приділена структурно-логічному аналізу технічних систем. Такий аналіз проводиться з метою оцінки впливу кожного елемента на працездатність системи загалом. Структурно-логічний аналіз передбачає уявлення технічної системи як схеми з послідовно і паралельно з'єднаних елементів. При цьому підході простежується певна аналогія з ланцюгом, складеним з провідних елементів (справний елемент пропускає струм, що відмовив – не пропускає): працездатний стан технічної системи відповідає можливість протікання струму від входу до виходу ланцюга.

У статті С. Г. Романюка розглядається імовірнісний підхід до проблеми надійності, що при вивченні надійності полягає в аналізі досліджуваного об'єкта (літака, системи охорони, комп'ютерної програми і т.д.), побудові, виходячи з "фізичних" міркувань про його природу, просторів елементарних подій, у введенні на них ймовірнісної міри та розгляді випадкових величин. У роботі висунуті цікаві міркування щодо взаємозв'язку надійності програми та наявності в ній помилок:

- число помилок у програмі – величина "неспостерігаюча", спостерігаються самі помилки, а результат їх прояви;
- неправильне спрацювання програми може бути наслідком не

однієї, а відразу кількох помилок;

- помилки можуть компенсувати один одного, тому після виправлення якоїсь однієї помилки програма може почати "працювати гірше";
- надійність характеризує частоту прояву помилок, але не

їхня кількість. У той же час добре відомо, що помилки виявляються з різною частотою: деякі залишаються невиявленими після багатьох місяців і навіть років експлуатації, проте, неважко навести приклади, коли одна помилка призводить до

неправильного спрацьовування програми за будь-яких вихідних даних, тобто. до нульової надійності.

У книзі І. А. Рябініна викладено основи логіко-імовірнісного обчислення, необхідні для дослідження надійності та безпеки структурно-складних систем. Розглянуто проблеми вихідних даних про безвідмовність елементів при малих обсягах статистичної інформації, довірчі та допустимі інтервали

оцінки надійності Наведено аналітичні та графічні форми подання умов працездатності та небезпечного стану системи, викладено логіко-імовірнісні методи дослідження надійності та безпеки на великій кількості прикладів.

У дисертації С. В. Гурова запропоновано універсальну математичну модель функціонування невідновлюваних та відновлюваних систем з довільними розподілами відмов та відновлень елементів та параметрів. На її основі дано базисне математичний опис стаціонарного та нестаціонарного режимів функціонування, придатний для побудови математичних моделей різних класів систем. Ці моделі можуть застосовуватись для оцінки надійності систем.

2.3. Постановка наукового завдання

Сьогодні кількість користувачів мережею з кожним днем зростає, що робить мережу вразливою та функціонально не стійкою. Запропоновані вище методи мають досить складний алгоритм розрахунку зв’язності мережі, та сама перевірка на функціональну стійкість досить довготривала. Сучасні телекомунікаційні мережі за час простою можуть понести витрати у розмірах мільйонів, або навіть мільярдів гривень.

У досліджуваних роботах є наведені алгоритми реалізації ПЗ, проте реально реалізованих продуктів на сьогоднішній день не існує, що дає підгрунтя для огляду та удосконалення існуючих математичних моделей та програмних алгоритмів.

Загальною проблемою існуючих моделей полягає в тому, що розрахунки здійснюються за довготривалий час для сучасних телекомунікаційних мереж, що

може стати загрозою для постачальників послуг відносно втрати споживачів, та надмірним навантаженням на канали зв’язку через збої.

Метою дипломної роботи є удосконалення існуючих інформаційних систем, зменшення часу на обчислення зв’язності шляху, та забезпечення функціональної стійкості телекомунікаційної мережі з урахуванням відмов у системі.

Методи дослідження – методи розпізнавання та групування інформації, емпіричні методи.

Для досягнення мети було вирішено такі наукові завдання:

- удосконалено методику розрахунку зв’язності шляху при відмовах на лініях зв’язку;
- удосконалено методику управління та моніторингу мережі зі змінною структурою;
- розглянуто понятійний апарат ФС мережі та проблема надійності програмного забезпечення;
- удосконалено та реалізовано сумісність обладнання з програмним забезпеченням;
- удосконалено методику управління мережею зі змінною структурою;
- реалізовано систему моніторингу в реальному часі зі своєчасним сповіщенням про відмови.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ НА ОСНОВІ УДОСКОНАЛЕНОЇ МОДЕЛІ ФУНКЦІОНАЛЬНОЇ СТІЙКОСТІ

3.1. Вибір та обгрунтування елементів та технологій

Головна умова до ПЗ проста і банальна: мати засіб, за допомогою якого максимально малою кількістю рухів за мінімальний час можна по можливості повно оцінити стан усієї мережі, а також мати можливість сповіщати окремих зацікавлених осіб про ту чи іншу подію. Для реалізації головного модулю для відстеження стану мережу було обрано вільну систему моніторингу Zabbix. Нижче на табл. 1 можна побачити порівняльну характеристику даної системи з аналогом Prometheus.

Таблиця 3.1 – Порівняльна характеристика Zabbix та Prometheus

Параметри	Zabbix	Prometheus
Збір метрик	Zabbix поділено на дві великі частини: сервер та агенти. Сервер розташований на одній машині, яка збирає та зберігає статистичні дані. Агенти розташовані на машинах, з яких збираються дані.	Отримує метрики із цільової системи, можна також збирати метрики за допомогою механізму push. (у випадках, коли служби моніторингу захищені брандмауером)
Сховище даних	Використовує зовнішню базу даних для зберігання. Надає базову функціональність для зберігання деяких текстових значень, їх аналізу та налаштування тригерів.	Зберігає дані у власній базі даних тимчасових рядів (TSDB). Зберігає лише значення часових рядів. Не підходить для тексту, логів чи журналів подій. Для журналів найкраще використовувати спеціалізовані продукти.

Продовження таблиці 3.1 – Порівняльна характеристика Zabbix та Prometheus

Візуалізація	Має власний веб-інтерфейс, в якому є панель керування із гнучкими налаштуваннями. Ця функція за замовчуванням, вам не потрібно нічого встановлювати чи настроювати. Веб-інтерфейс Zabbix пропонує кілька варіантів: панелі моніторингу на основі віджетів, графіки, мережеві карти, слайд-шоу, деталізовані звіти.	Має Expression Browser, простий, але корисний інструмент візуалізації. У браузера немає функцій повноцінної панелі моніторингу. Використовуйте його для докладних запитів до збережених метриків. Браузер Expression не відображає показники, за якими ви спостерігаєте протягом тривалого часу.
Сповіщення	Є вбудована функція оповіщення. Zabbix інформує відповідального колегу про виникнення проблем, використовуючи різні канали та опції. Система попереджень Zabbix керує подіями по-різному: надсилати повідомлення, виконувати віддалені команди тощо. Також можна настроїти повідомлення залежно від ролі одержувача, вибравши, яку інформацію включати.	Щоб керувати сповіщеннями за допомогою Prometheus, необхідно встановити Alertmanager. Повідомлення надсилаються електронною поштою, через відповідні системи та чат-платформи. Що зручно, Alertmanager класифікує кілька повідомлень схожого характеру на одне, щоб уникнути дублювання.
Довгострокове зберігання даних	Розрахований на довгострокове зберігання	Розрахований на короткострокове (тиждень-два) зберігання метрик та роботи з ними.

Продовження таблиці 3.1 – Порівняльна характеристика Zabbix та Prometheus

Вартість	Zabbix безкоштовний продукт з відкритим вихідним кодом. Zabbix Enterprise доступний через саме ПЗ.	У Prometheus є відкритий вихідний код, продукт безкоштовний. Готовий для бізнесу Prometheus (послуга від MetricFire), включаючи Hosted Prometheus, Hosted Graphite та Grafana, коштує від 85 доларів на місяць.

З порівняльної таблиці 3.1 можна побачити, що Zabbix має ряд переваг над Prometheus, а саме: має довгострокове зберігання даних, що для великих підприємств телекомунікаційних мереж важливо; має вбудовані елементи для моніторингу; має більш зручну реалізацію для сповіщення; відкритий вихідний код із серверною частиною C та клієнтською частиною PHP; можливість підключення сторонніх плагінів для більш красивого відображення даних; підтримка відомих реляційних систем управління БД як MySQL, MariaDB, PostgreSQL, SQLite, Oracle або IBM DB2.

Як вище було зазначено у табл. 3.1 Zabbix поділяється на дві великі частини: сервер та агенти. Сервер розташовується на одній машині, яка збирає та зберігає статистичні дані, а агенти – на тих машинах, дані з яких збираються, схематичне зображено такої побудови системи можна побачити на рис. 3.1

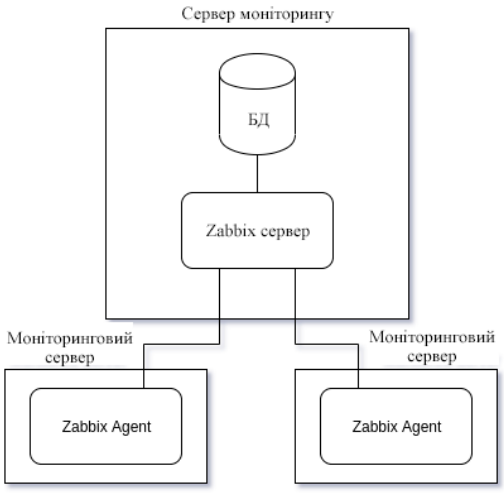


Рисунок 3.1 – Складові системи моніторингу Zabbix

Zabbix дає можливість налаштувати під себе правила моніторингу для окремого хоста, це дозволяє створити, наприклад, єдиний шаблон-тригер, який буде відпрацьовувати на конкретні сигнали, що будуть поступати з системи, та виводити якесь конкретне повідомлення рис.3.2.

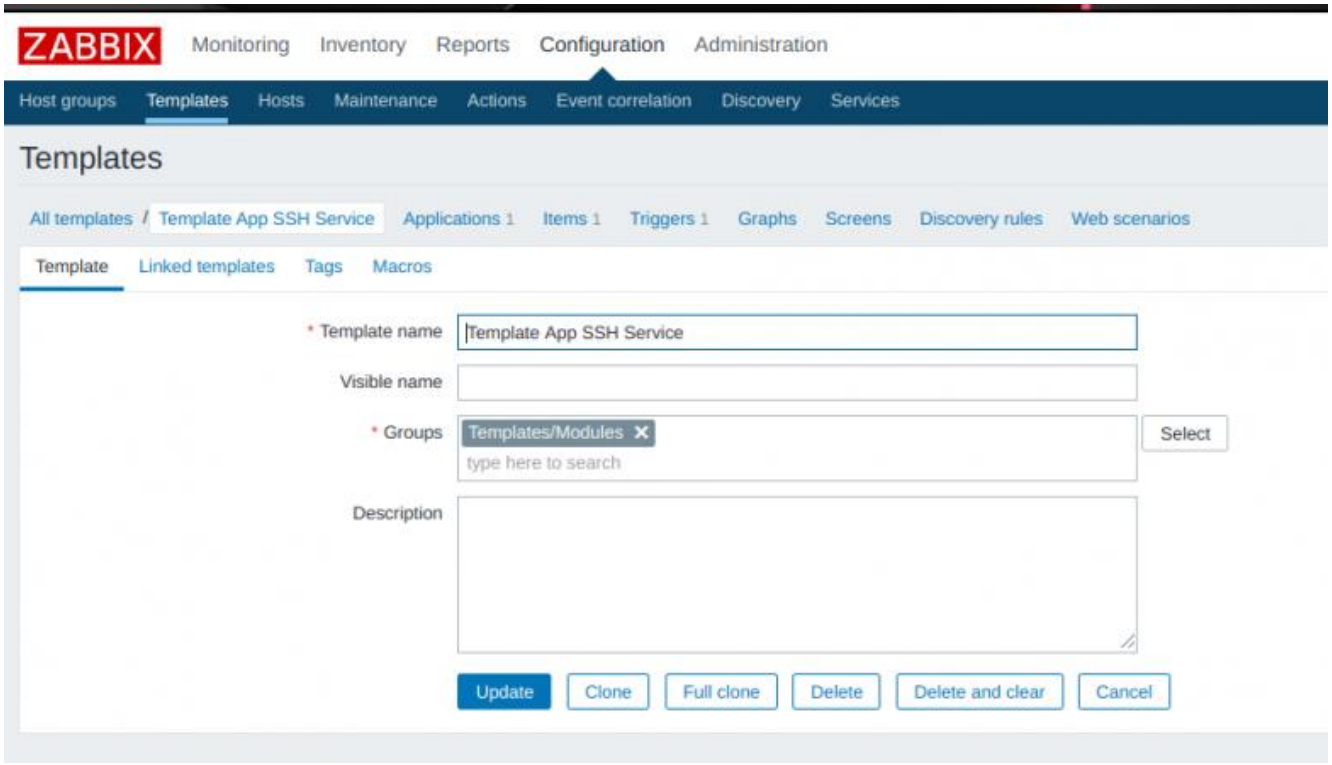


Рисунок 3.2 – створення шаблону у Zabbix

У Zabbix дає змогу збирати інформацію у 17 способів. Нижче вказано найпопулярніші моделі збору інформації:

- Zabbix agent (Zabbix-агент) - сервер збирає інформацію в агента самостійно, підключаючись за певним інтервалом;
- Simple check (Прості перевірки) - прості операції, у тому числі пінг;
- Zabbix trapper (Zabbix-траппер) — збір інформації з траперів, що є мостами між використовуваними сервісами і системою;
- Zabbix aggregate (Zabbix-комплекс) - процес, що передбачає збирання сукупної інформації з бази даних;
- SSH agent (SSH-агент) - система підключається по SSH, використовує зазначені команди;

- Calculate (Обчислення) — перевірки, які система здійснює, зіставляючи наявні дані, зокрема після попередніх зборів.

Тригери це логічні вирази зі значеннями FALSE, TRUE та UNKNOWN, що використовуються для обробки даних. Їх можна створити вручну. Перед використанням тригери можна протестувати на довільних значеннях рис. 3.3.

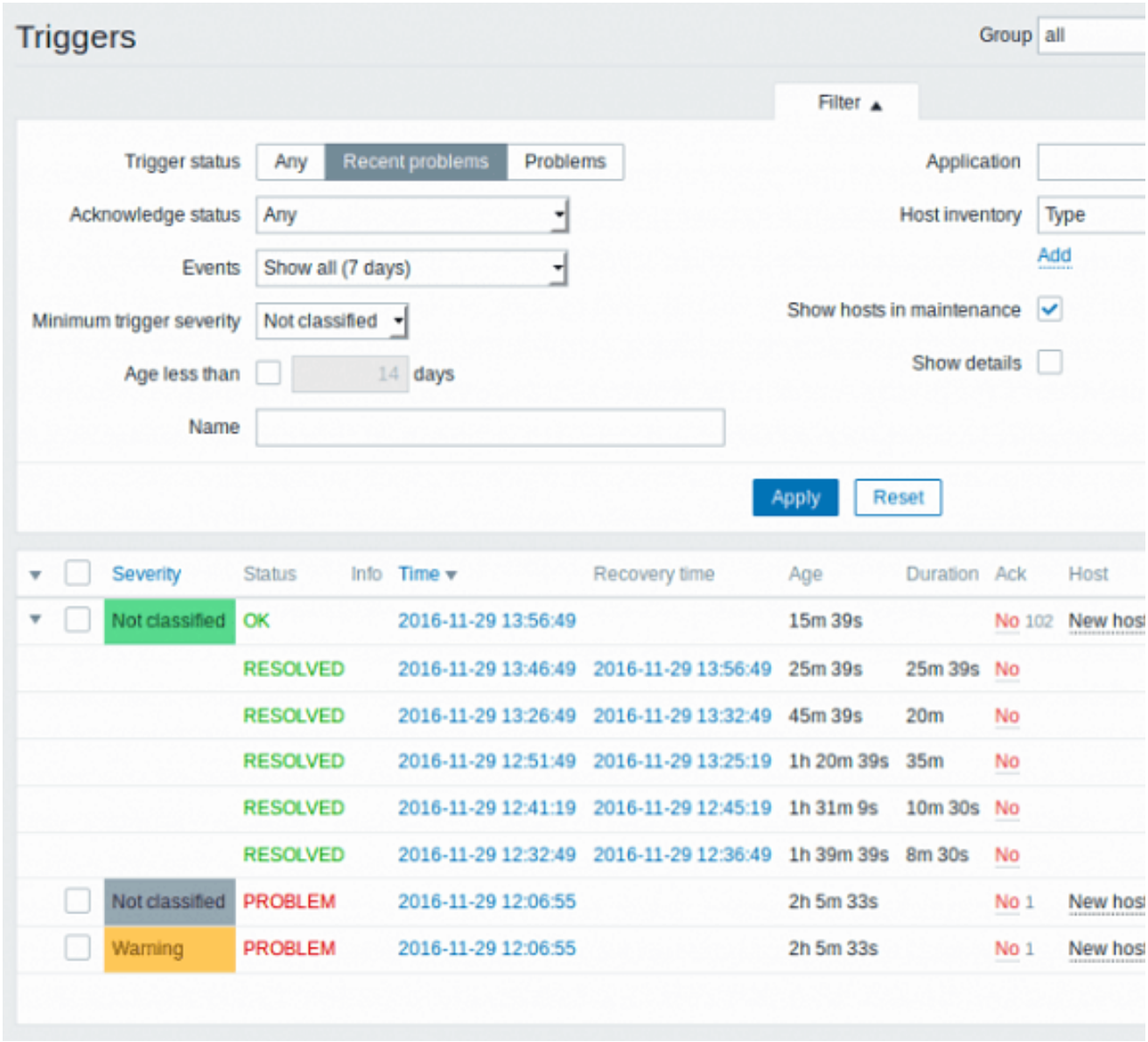


Рисунок 3.3 – Тестування тригерів

Кожен тригер має рівень серйозності загрози, який маркується відповідним кольором і передається звуковим оповіщенням у веб-інтерфейсі рис. 3.3:

- Не класифіковано (Not classified) – сірий;

- Інформація (Information) - світло-синій;
- Попередження (Warning) – жовтий;
- Середня (Average) – помаранчевий;
- Висока (High) – світло-червоний;
- Надзвичайна (Disaster) – червоний.

Тригери мають ще одну важливу функцію для моніторингу – прогнозування. Воно передбачає можливі значення та час їх виникнення. Прогноз складається з урахуванням раніше зібраних даних. Аналізуючи їх, тригер виявляє майбутні проблеми, попереджає адміністратора про ймовірність. Це дає можливість запобігти пікам навантаження на обладнання або місце, що закінчується, на жорсткому диску.

Нажаль, у Zabbix не має можливості нормально реалізувати карту покриття мережі, проте, є можливість до системи моніторингу підключити додаткові плагіни, які відображення покриття зроблять значно зручніше. В якості додаткового графічного відображення у роботі використовується плагін Grafana.

Grafana – це найпотужніший інструмент для аналітики та візуалізації даних. Велика кількість плагінів дозволяють забирати дані з різних джерел (zabbix, clickhouse, influxDB), обробляти їх на льоту (вважати середнє значення, суму, різницю тощо) та малювати різноманітні графіки (від простих ліній, спідометрів, таблиць до складних схем).

Для зберігання, збору та надання даних використовується phpMyAdmin. Простота використання додатоку phpMyAdmin полягає в тому, що вона дозволяє обійтися без введення команд SQL, тому робота з БД є цілком простим завданням навіть для людини, яка вперше бачить програмі, та трохи знайома з MySQL. Активне застосування MySQL в web-програмуванні зробило phpMyAdmin досить актуальним, а інтуїтивно зрозумілий інтерфейс спільно з широкою функціональністю і підтримкою понад 62 мов забезпечило йому неймовірну популярність серед web-розробників.

Вибір операційної системи. З табл. 3.2 можна побачити обгрунтування, чому для реалізації системи моніторингу було обрано Linux.

Таблиця 3.2 – Порівняльна характеристика ОС

Параметри	Linux	Windows
Доступні мови та технології програмування	Кращий та простіший для використання PHP + MySQL	ASP.NET лише під Windows Server
Служби та протоколи, що підтримуються	в Linux наявна служба для роботи через SSH.	Не має служби для роботи через SSH, та серверне ПЗ платне
Цінова політика	Більшість серверних дистрибутивів Linux є безкоштовними, хоча є й платні варіанти	Ліцензію Windows купувати доведеться у будь-якому випадку, хоча в залежності від дистрибутива ціна може змінюватись у широких межах
Робота з базами данных	в Linux зв'язка PHP + MySQL працює ефективніше, забезпечуючи більшу швидкодію. Проте якщо проект спочатку оптимізований для роботи з MS SQL або Access, то Linux не підійде	Дані системи управління базами даних дуже популярні у деяких розробників, тому Windows часто вибирають саме для роботи із цими продуктами
Складність адміністрування	Віртуальний сервер на Linux для новачка буде складнішим у плані віддаленого адміністрування. Але для сайту на PHP краще вибирати саме цей варіант, оскільки він вважається стабільнішим, захищеним і швидким, ніж Windows. Тобто, на власному сервері можна буде запустити додатковий віртуальний сервер (або декілька)	Для новачків набагато зручнішим вибором стане ОС Windows, оскільки практично кожна людина, яка має відношення до комп'ютерів, має багатий досвід роботи з десктопною версією цієї операційної системи

3.2. Налаштування Zabbix

Всі налаштування були виконані у дистрибутиві Ubuntu 20.04. Для установки Zabbix на комп’ютер на офіційному сайті необхідно обрати наступне: Zabbix Version (4.4) -> OS Distribution (Ubuntu) -> OS Version (20.04 Bionic) -> Database (MySQL) -> Веб сервер (Nginx или Apache). Після того, як завантажили программу, необхідно завантажити та додати репозиторій, та встановити необхідні пакети рис. 3.4. Для цього необхідно у консолі виконати наступні команди [14]:

```
# wget https://repo.zabbix.com/zabbix/4.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.4-1+bionic_all.deb
# dpkg -i zabbix-release_4.4-1+bionic_all.deb
# apt update
# apt install zabbix-server-mysql zabbix-frontend-php zabbix-nginx-conf zabbix-agent
```

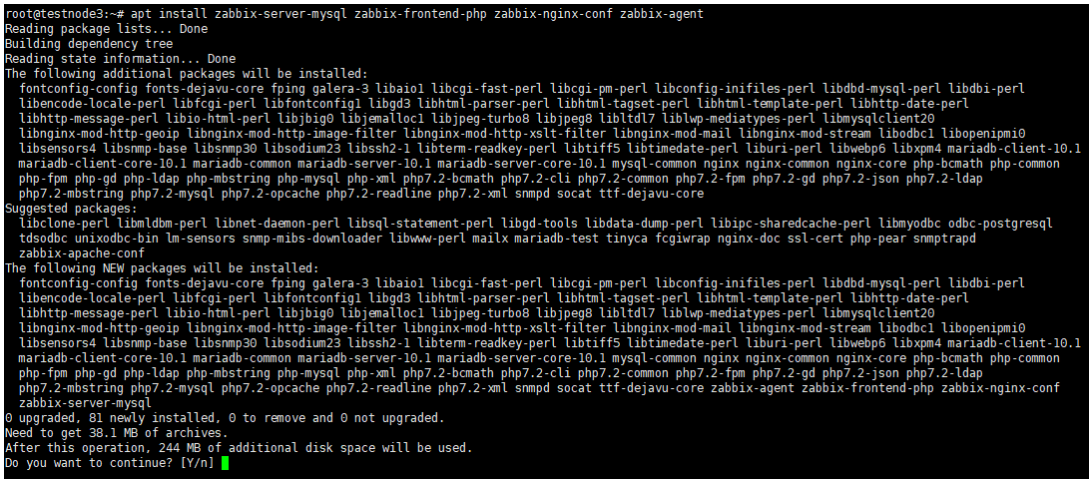


Рисунок 3.4 – Завантаження додаткових пакетів для роботи

Для того, щоб мати доступ до веб-версії програми, необхідно створити та налаштувати початкову БД. Для цього створюємо БД та надаємо їй права користувача, за допомогою яких Zabbix буде звертатись до бази даних, для цього виконуємо команди [14]:

```
mysql -uroot
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> grant all privileges on zabbix.* to zabbix@localhost identified by 'ВашПароль';
mysql> quit;
```

UTF-8 є єдиним кодуванням, яке підтримується у Zabbix. Вона, як відомо, працює без будь-яких проблем із безпекою. Користувачі повинні знати, що існують певні проблеми з безпекою під час використання деяких інших кодувань.

Імпортуємо базу даних Zabbix. Потрібно ввести пароль, який вказали під час створення користувача. Змінюємо файл конфігурації /etc/zabbix/zabbix_server.conf, та вказуємо пароль від нового користувача. «DBPassword» є паролем, який ми задали під час створення початкової бази даних:

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<пароль>
```

Додатково необхідно виставити наступні параметри PHP /etc/php.ini [14]:

```
memory_limit 128M
upload_max_filesize 8M
post_max_size 16M
max_execution_time 300
max_input_time 300
max_input_vars 10000
```

Додаємо службу zabbix-server до автозапуску та запускаємо її. Тепер потрібно настроїти frontend (веб-інтерфейс) Zabbix. У браузері відкриваємо зазначену раніше URL-адресу zabbix сервера, та приписуємо його в hosts файлі або на DNS сервері. Перевіряємо, що в усіх вимогах інсталятора вказано ОК рис.3.5.

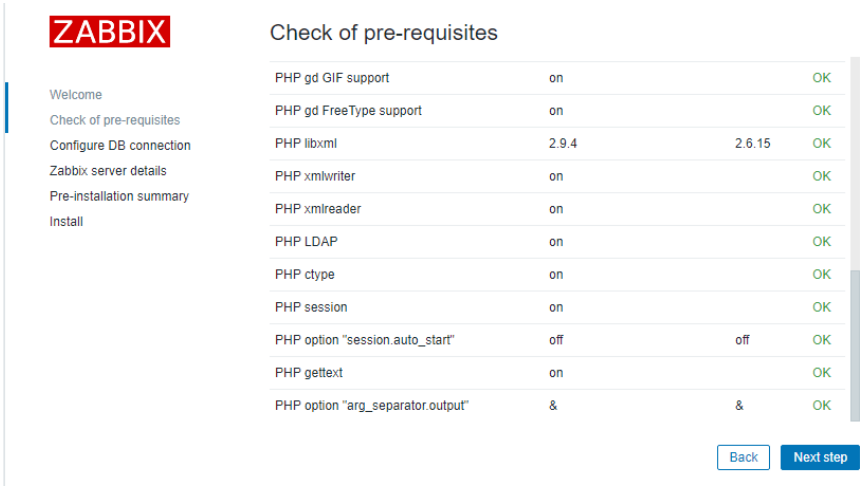


Рисунок 3.5 – Перевірка налаштувань

Вказуємо дані для підключення до БД. Для підключення обираємо користувача та пароль, який створили раніше рис.3.6.

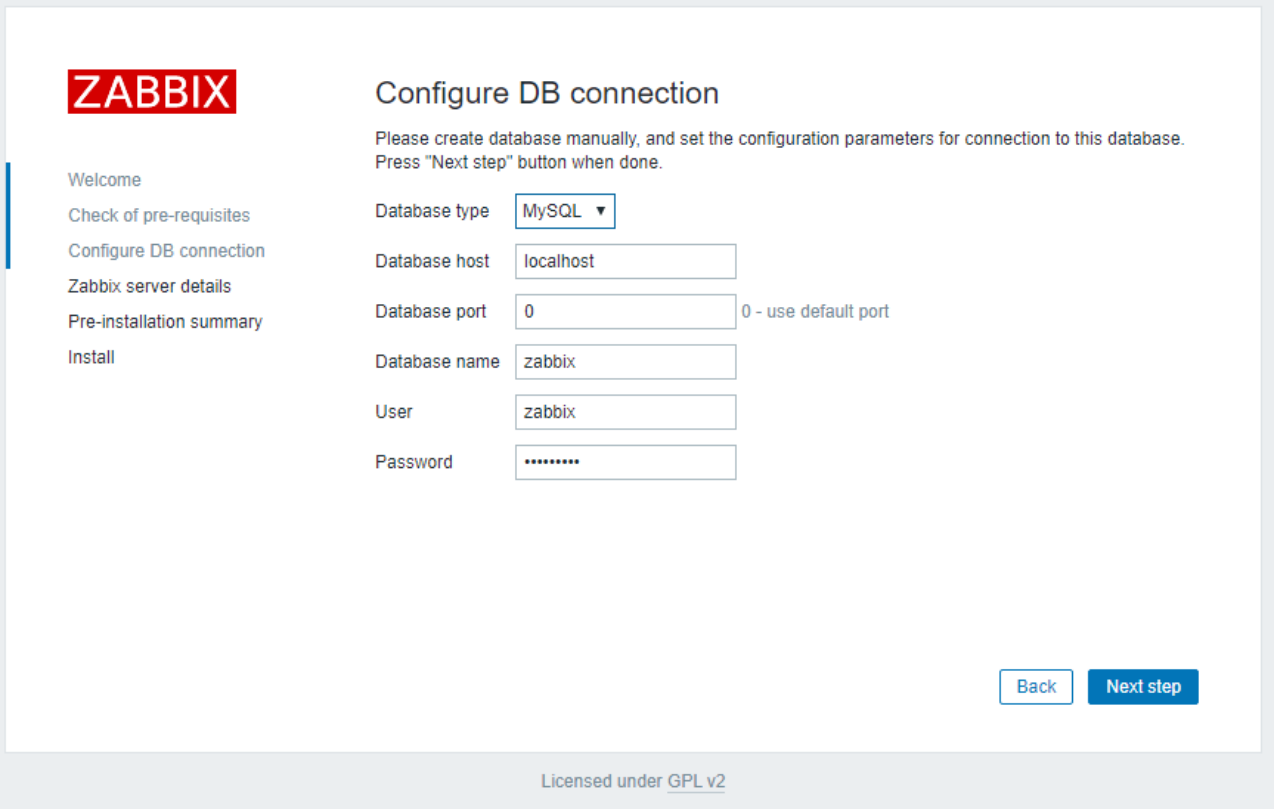


Рисунок 3.6 – Підключення до БД

Вказуємо назву сервера Zabbix. По рекомендації засновників краще залишити стандартний порт TCP 10051 рис. 3.7. Система Zabbix за замовчуванням використовує два порти:

- TCP 10050 - порт пасивного агента, за ним zabbix сервер опитує клієнтів;
- TCP 10051 – порт, на якому zabbix сервер отримує дані від клієнтів (активний агент).

Відкриваємо порти та перезавантажуємо firewall [14]:

```
#firewall-cmd --add-service={http,https} --permanentfirewall-cmd
--add-port={10051/tcp,10050/tcp} –permanent
# firewall-cmd –reload
```

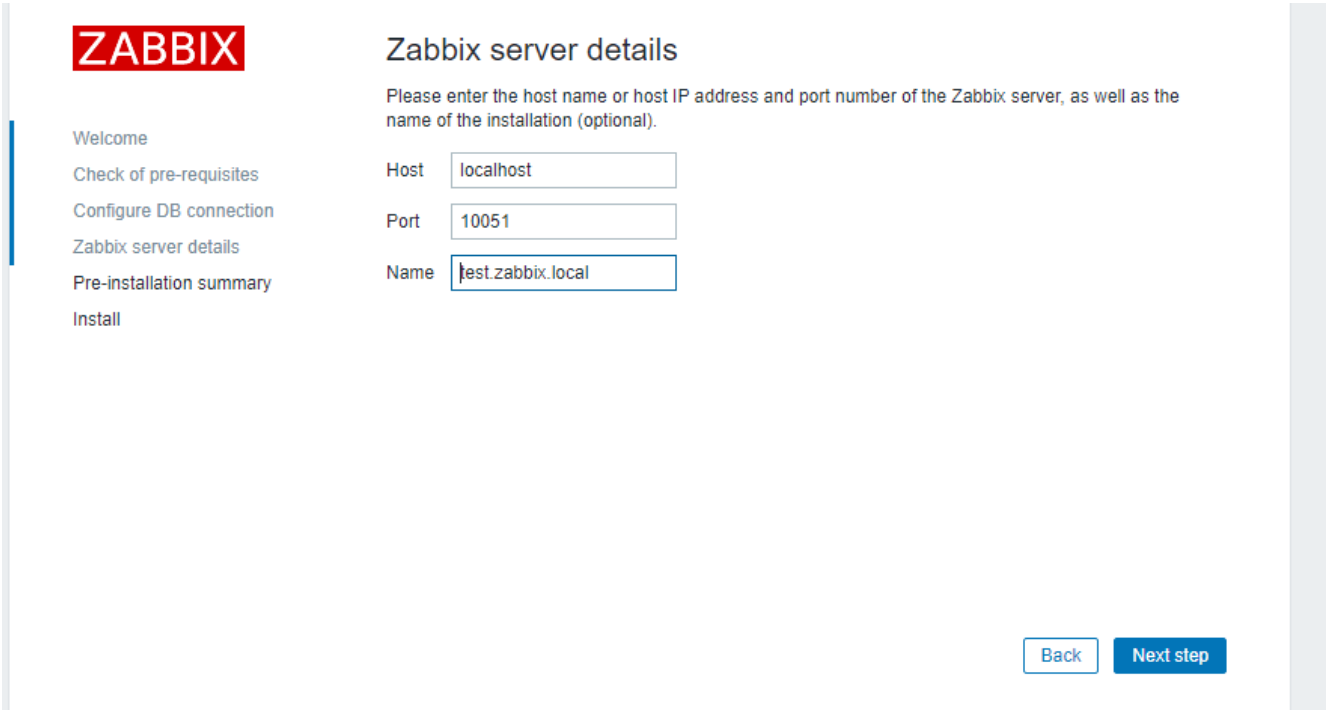


Рисунок 3.7 – Налаштування порту

Після цього натискаємо «Next Step» і «Finish» рис.3.7. Після успішного встановлення заходимо на сервер. За замовчуванням логін Admin, пароль zabbix рис.3.8, для безпеки даних дані у цих полях необхідно замінити на свої.

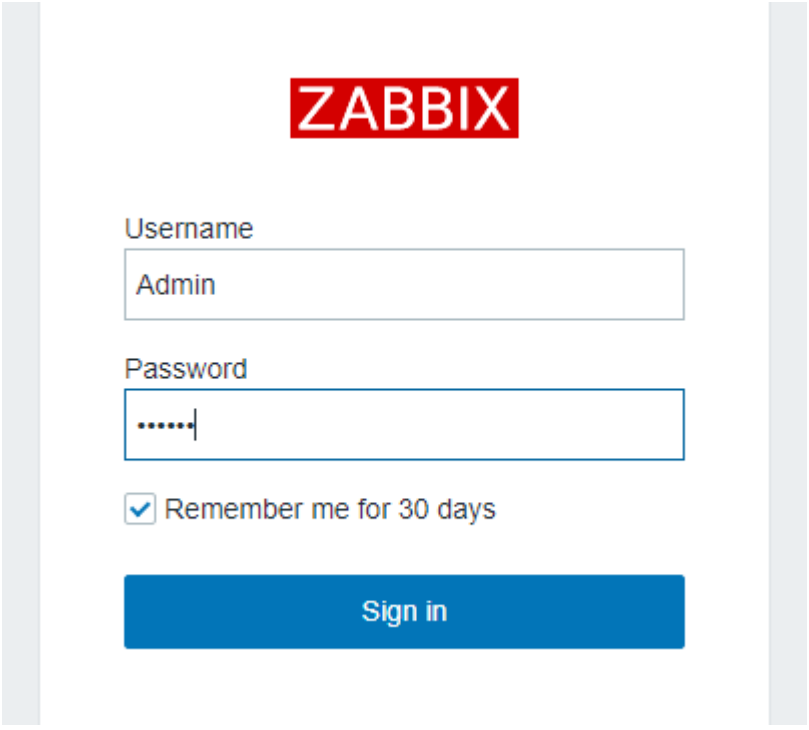


Рисунок 3.8 – Авторизація

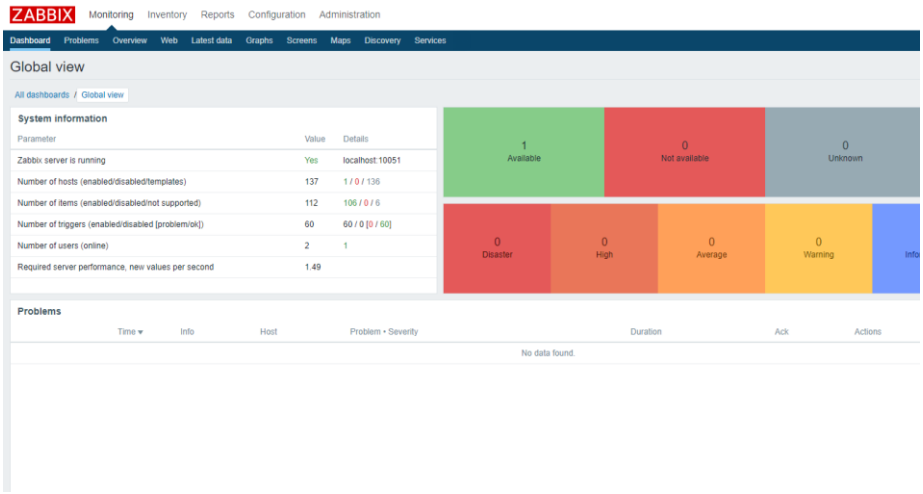


Рисунок 3.9 – Завершення налаштувань

На цьому встановлення сервера Zabbix Server завершено рис.3.9. На офіційному сайті є покрокові інструкції для встановлення Zabbix Server на інші операційні системи, а також інструкції, як зібрати zabbix-server з вихідного коду або запустити його в docker контейнерах.

З установкою агента закінчили, тепер потрібно створити та налаштувати хост у панелі управління zabbix. Переходимо в Configuration->Hosts->Create host рис.3.10. Вкладка «Вузол мережі» містить загальні атрибути вузла мережі.

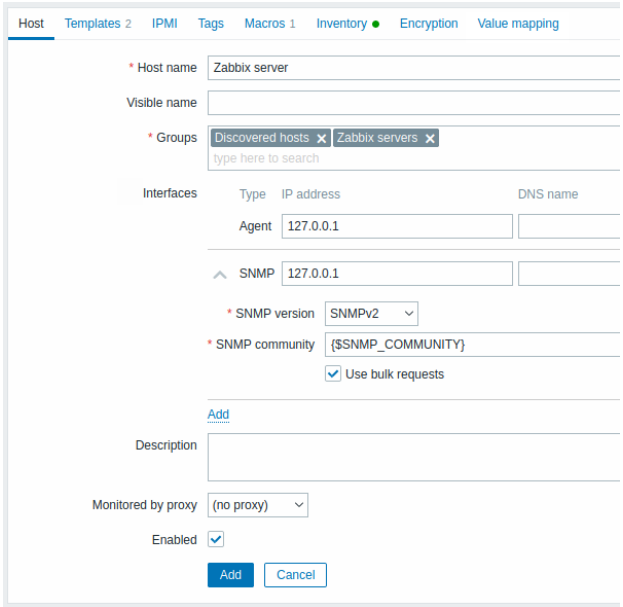


Рисунок 3.10 – Створення хосту

Усі обов’язкові поля введення позначені червоною зірочкою. З табл. 3.3 можна побачити детальний опис пунктів для більш точного налаштування системи.

Таблиця 3.3 – опис полів налаштування хосту

Параметри	Опис
Ім’я вузла мережі	Унікальне ім’я мережі. Дозволено буквено-цифрові символи, пробіли, точки, тире та підкреслення. При працюючому Zabbix агенті на вузлі мережі, що настроюється, параметр Hostname з файлу конфігурації агента повинен мати таке ж значення, як і введене тут ім'я вузла мережі. Ім'я цього параметра необхідне для обробки активних перевірок.
Видиме ім’я	Якщо вказати це ім’я, саме воно буде видимим у списках, картах та інше. Цей атрибут має підтримку UTF-8.
Групи	Обрати групи вузлів мережі, до яких належатиме вузол мережі. Вузол мережі повинен належати принаймні до однієї групи вузлів мережі. Нову групу можна створити та приєднати до вузла мережі, додавши неіснуюче ім’я групи.
IP адрес	IP адреса вузла мережі (опціонально).
DNS ім’я	DNS ім'я вузла мережі (опціонально).
Підключатися через	Натискання на відповідну кнопку підкаже що використовувати для отримання даних від агентів: IP - Підключення до вузла мережі, використовуючи IP-адресу (рекомендується) DNS - Підключення до вузла мережі, використовуючи ім'я DNS
Порт	Номер TCP/UDP порту. Значення за замовчуванням: 10050 для агента Zabbix, 161 для агента SNMP, 12345 для JMX і 623 для IPMI.
За замовчуванням	Позначити, щоб інтерфейс вважався за замовчуванням.
Спостереження через проксі	Вузол мережі може спостерігатися через Zabbix сервер або через один із Zabbix проксі: - (без проксі) – вузол мережі спостерігається Zabbix сервером; - Ім'я проксі - вузол мережі спостерігається через Zabbix проксі "Ім'я проксі".
Активовано	Позначити, щоб зробити вузол мережі активним, готовим до моніторингу. Якщо не зазначено, вузол мережі неактивний, таким чином, не спостерігається.

Вкладка «Шаблони» дозволяє приєднати раніше створені шаблони до мережі. Усі об’єкти (елементи даних, тригери, графіки та групи елементів даних) будуть успадковані із шаблону.

Вкладка «Макроси» дозволяє задати користувацькі макроси на рівні вузла мережі. Є можливість переглянути макроси рівня шаблону та глобальні макроси, якщо обрати опцію Макроси вузла мережі та успадковані. Це те місце, де відображаються всі певні користувацькі макроси для цього вузла мережі зі своїми розкритими значеннями, а також інформація про те, звідки ці макроси.

Для зручності є посилання на налаштування відповідних шаблонів та глобальних макросів. Також можна змінити макрос рівня шаблона/глобальний на рівні вузла мережі, фактично створивши копію цього макросу на вузлі мережі.

Вкладка Інвентарні дані вузла мережі дозволяє вручну ввести інформацію про інвентарні дані вузла мережі. Ви також можете вибрати Автоматично заповнення інвентарних даних або деактивувати заповнення інвентарних даних для цього вузла мережі.

Вкладка «Шифрування» дозволяє вимагати шифрування з’єднань із вузлом мережі табл.3.4.

Таблиця 3.4 – Опис полів шифрування

Параметр	Опис
Підключення до вузла мережі	Zabbix сервер або проксі підключаються до агента Zabbix на хості: без шифрування (за замовчуванням), використовуючи PSK (pre-shared key) або сертифікат.
Зв’язок з вузлом	Обираємо тип підключень з вузлом мережі (тобто з агентом Zabbix і Zabbix sender). Можна вибрати кілька типів з'єднань одночасно.За замовчуванням – "Без шифрування".
Видавець	Дозволений емітент сертифікату. Сертифікат спочатку підтверджується CA (центр сертифікації). Якщо він дійсний, підписаний за допомогою CA, тоді можна використовувати поле Видавець для більш строгого обмеження дозволених CA. Це поле призначене для використання, якщо Zabbix інсталяція використовує сертифікати від кількох CA. Якщо поле не заповнене, тоді приймається будь-яка CA.

Продовження таблиці 3.4 – Опис полів шифрування

Тема	Дозволена тема сертифікату. Сертифікат спочатку підтверджується CA. Якщо він дійсний, підписаний за допомогою CA, тоді можна використати поле Тема, щоб дозволити лише одне значення рядка Тема. Якщо поле порожнє, то приймається будь-який сертифікат, підписаний налаштованим CA.
Ідентифікатор PSK	Рядок ідентифікації pre-shared key.
PSK	Pre-shared key (рядок у шістнадцятирічному форматі). Максимальна довжина: 512 шістнадцятирічних цифр (256-байт PSK), якщо Zabbix використовує бібліотеки GnuTLS або OpenSSL, 64 шістнадцятирічних цифр (32-байт PSK), якщо Zabbix використовує бібліотеку mbed TLS (PolarSSL).

Створення групи вузлів мережі. Для створення групи вузлів мережі у веб-інтерфейсі Zabbix виконуємо:

- переходимо до: Налаштування → Групи вузлів мережі;
- надимаємо «Створити групу вузлів мережі» у верхньому правому куті екрана;
- у діалозі водимо параметри групи.

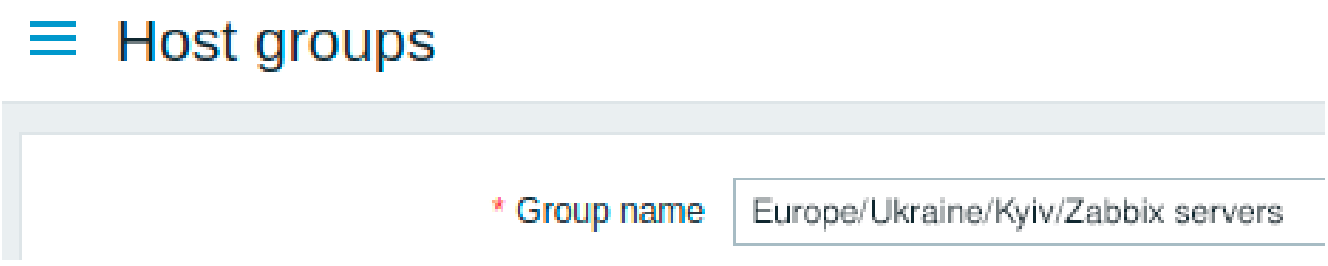


Рисунок 2 – Створення групових хостів

Усі обов’язкові поля введення позначені червоною зірочкою табл.3.5.

Таблиця 3.5 – Опис полів для заповнення

Параметри	Опис
Ім’я групи	Необхідно ввести унікальну назву групи вузлів мережі. Щоб створити вкладену групу вузлів мережі, використовуємо пряму косу лінію '/', наприклад Європа/Україна/Київ/Zabbix сервер. Можна створити цю групу, навіть якщо жодна з батьківських груп вузлів мережі не існує.
Застосувати права доступу до всіх підгруп	Ця опція доступна лише Zabbix Супер Адмін користувачам і лише при редагуванні існуючої групи вузлів мережі. При виборі цієї натиснути кнопку Оновити, щоб застосувати однаковий рівень доступу до всіх вкладених груп вузлів мережі. Для груп користувачів, які могли б мати різні права доступу, призначені на вкладці групи вузлів мережі, буде застосовано рівень прав доступу батьківської групи вузлів мережі до всіх вкладених груп. Ця дія є одномоментною опцією, яка не зберігається у базі даних. Опція підтримується з Zabbix 3.4.0.

Налаштування карти

Налаштування карти мережі в Zabbix вимагає того, що спочатку необхідно створити карту визначивши її загальні параметри, а потім заповнити карту елементами та зв’язками між цими елементами.

Заповнити елементами можна: вузлами мережі, групами вузлів мережі, тригерами, зображеннями та іншими картами мережі.

Для відображення елементів карти використовуються іконки. Можна вказати інформацію, яка буде відображена з іконками та встановити, які нещодавні проблеми будуть відображатися в певний спосіб. Можна зв'язати іконки та задати інформацію, яка відображатиметься у зв’язків. За необхідності можна додати власні URL, які будуть доступні при натисканні на іконках. Таким чином, при

необхідності можна зв’язати іконку вузла мережі з властивостями вузла мережі або іконку картки мережі з іншою карткою.

На картах мережі, які вже готові, можна переглянути у Моніторинг → Карти мережі. На сторінці моніторингу можна натиснути на іконки та скористатися посиланнями на якісь скрипту або URL.

Усі користувачі Zabbix (включаючи користувачів не адміністраторів) можуть створювати карти мережі. Карти мережі мають власника – користувача, який створив їх.

Карти мережі можна зробити публічними або приватними. Публічні карти мережі видимі всім користувачам, однак вони повинні мати принаймні права на читання всіх елементів картки мережі, щоб її побачити. Для додавання елемента на карту мережі користувач також повинен мати права принаймні читати цей елемент.

Карти мережі помітні їх власникам. Власник може надавати спільний доступ до приватних карток мережі іншим користувачам та групам користувачів. Звичайні (не Супер адміністратори) користувачі можуть призначати спільний доступ лише тим групам та користувачам, до яких вони входять самі. Приватні карти мережі будуть видно своїм власникам та користувачам із загальним доступом до цієї картки мережі так довго, поки вони мають права на читання всіх елементів картки мережі. Користувачі рівня Адміністратора, поки вони мають право на читання всіх елементів картки мережі, можуть переглядати та редагувати приватні картки мережі незалежно від того, чи вони є власниками та чи входять до списку користувачів загального доступу.

Створення картки мережі. Для створення карти, зробіть таке:

- перейти в Моніторинг → карта мережі
- перейти до перегляду всіх карт
- натиснути Створити карту мережі

Вкладка Карта мережі містить загальні атрибути карт мереж рис.3.11, з детальним описом полів у табл. 3.6:

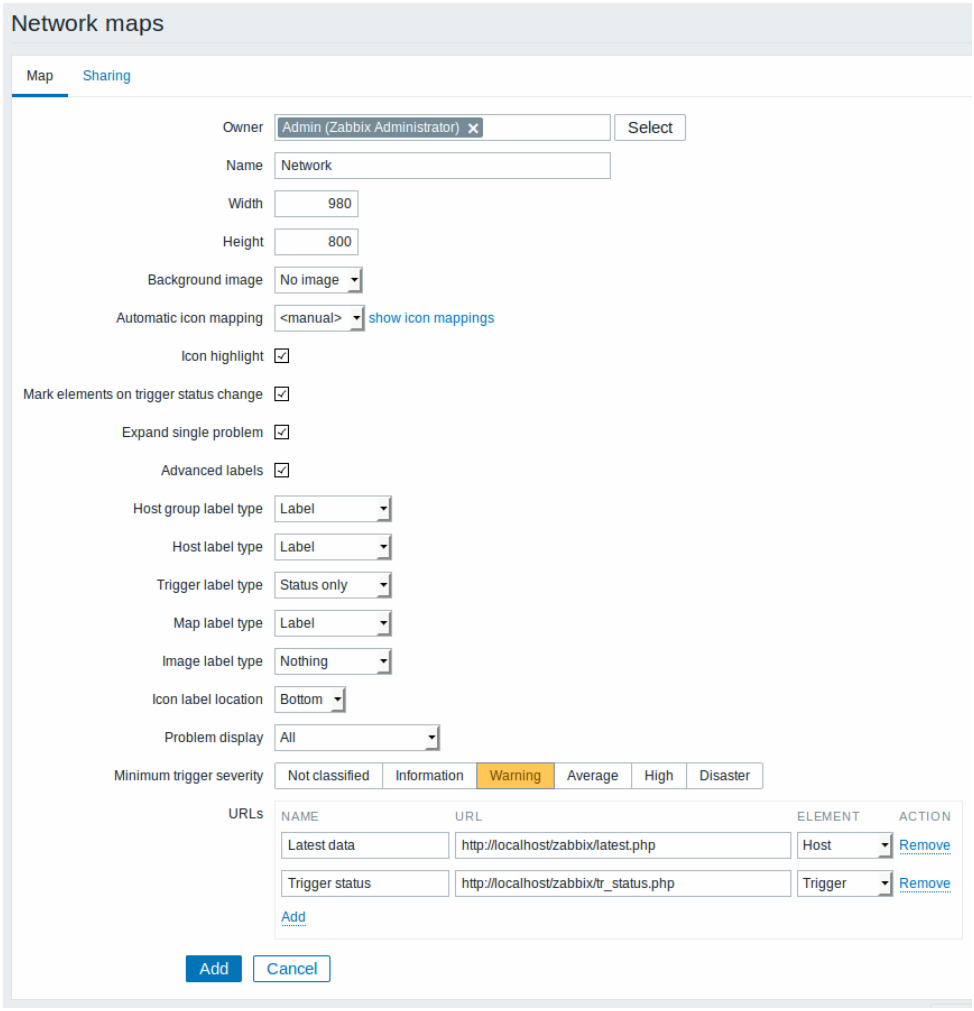


Рисунок 3.11 - Вкладка Карта мережі

Таблиця 3.6 – Опис полів карти мережі

Параметри	Опис
Власник	Ім’я власника карти
Ім’я	Унікальне ім’я карти
Ширина	Ширина карти у пікселях
Висота	Висота карти у пікселях

Продовження таблиці 3.6 – Опис полів карти мережі

Фонове зображення	Використання фонового зображення: - Немає зображення - без фонового зображення - Зображення – обране зображення, яке використовується як фонове зображення. Масштабування не провадиться. Можна використовувати географічну карту або будь-яке інше зображення для покращення вашої карти.
Автоматична відповідність іконок	Можна вказати використання автоматичної відповідності іконок, задані в Адміністрація → Загальні → Відповідність іконок. Відповідність іконок дозволяє відображати деякі іконки відповідно до полів інвентарних даних вузлів мережі.
Підсвічування іконок	Якщо пизначити цю опцію, іконки отримають підсвічування. Елементи з активними тригерами матимуть фон у вигляді кола, такого ж кольору, як і тригер з найвищою важливістю. Крім того, відображатиметься зелена товста лінія навколо кола, якщо всі проблеми підтверджені.Якщо елемент "деактивований" або "в обслуговуванні", буде використовуватися фон у вигляді квадрата, сірий і помаранчевий відповідно.
Позначки елементів при зміні стану тригера	Нещодавня зміна стану тригера (нещодавна проблема або її вирішення) буде підсвічуватися маркерами (червоні трикутники, що вказують всередину) по трьох сторонах іконки елемента, які не зайняті підписом. Маркери відображаються 30 хвилин.
Розгортання одиночної проблеми	Якщо елемент карти (вузол мережі, група вузлів мережі або інша картка) містить одну проблему, ця опція керує, що буде відображено ім’я проблеми (тригера) або кількість проблем. Якщо вибрано, використовується ім’я проблеми.

Продовження таблиці 3.6 – Опис полів карти мережі

Розширені підписи	Якщо ця опція вибрана, ви зможете задати тип підпису для різних типів елементів.
Тип підпису до іконки	Тип підпису, який буде використовуватися для іконок: Підпис - підпис до іконки; IP адреса - IP адреса; Ім’я елемента – ім’я елемента (наприклад, ім'я вузла мережі); Тільки стан - тільки стан (ОК або ПРОБЛЕМА); Нічого - підписи не відображатимуться.
Розташування підпису до іконки	Розташування підпису стосовно іконки: По нижньому краю – нижче іконки; Ліворуч – ліворуч; Праворуч – праворуч; По верхньому краю - вище за іконки.
Відображення проблем	Відображення кількості проблем як: Все – буде відображено повну кількість проблем; Непідтверджені окремо - кількість непідтверджених проблем буде відображено окремим числом із загальної кількості проблем; Тільки непідтверджені – буде відображено лише кількість непідтверджених проблем
Мінімальна важливість тригерів	Проблеми з важливістю нижче заданої в даній опції не відображаються на карті. Наприклад, з обраною важливістю Попередження, зміни станів тригерів з Інформація та Не класифіковано не будуть відображені на карті.
URL	Можна вказати URL для кожного типу елемента (з підписом). Вони будуть відображатися як посилання, якщо користувач натисне елемент у режимі перегляду карти. В URL карт можна

	використовувати макроси: {MAP.ID}, {HOSTGROUP.ID}, {HOST.ID}, {TRIGGER.ID}

Вкладка «Загальний доступ» містить тип карти мережі, так і опції загального доступу (групи користувачів, користувачі) для приватних карт мережі рис.3.12:

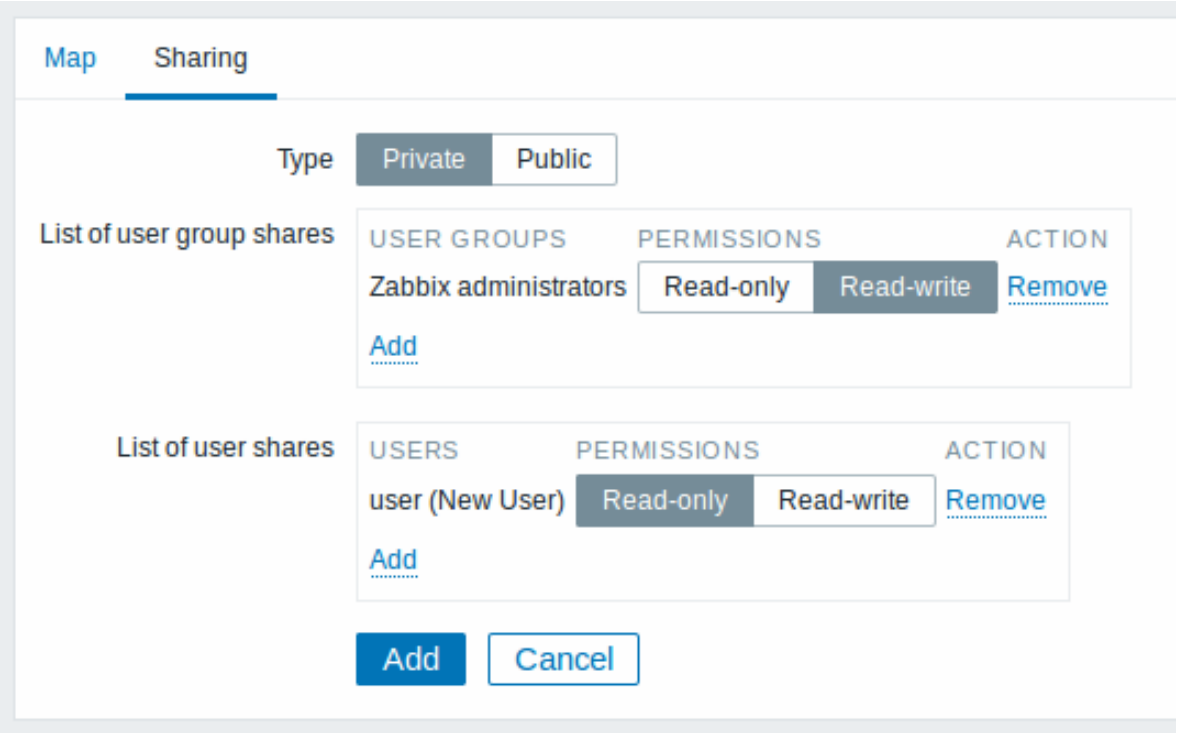


Рисунок 3.12 - Вкладка «Загальний доступ»

Коли натикаємо Додати для збереження картки мережі, створюємо порожню карту мережі з ім’ям, розмірами та певними параметрами. Тепер необхідно додати якісь елементи. Щоб це зробити, обираємо Конструктор у списку карт мереж, щоб відкрити область редагування.

Щоб додати елемент, обираємо «Додати» наступну за підписом Іконка. Новий елемент з’явиться у верхньому лівому куті карти. Переміщаємо у бажане місце. Повторюємо процедуру до тих пір, поки на карті не буде знаходитись стільки іконок, скільки потрібно для роботи.

Маємо декілька розміщених елементів, можна змінити імена та задати параметри якомусь конкретному об’єкту рис.3.13. Натиснувши на елемент, в

випадаючому діалозі задаємо тип елемента, ім’я або обираємо іншу іконку тощо рис.3.13.

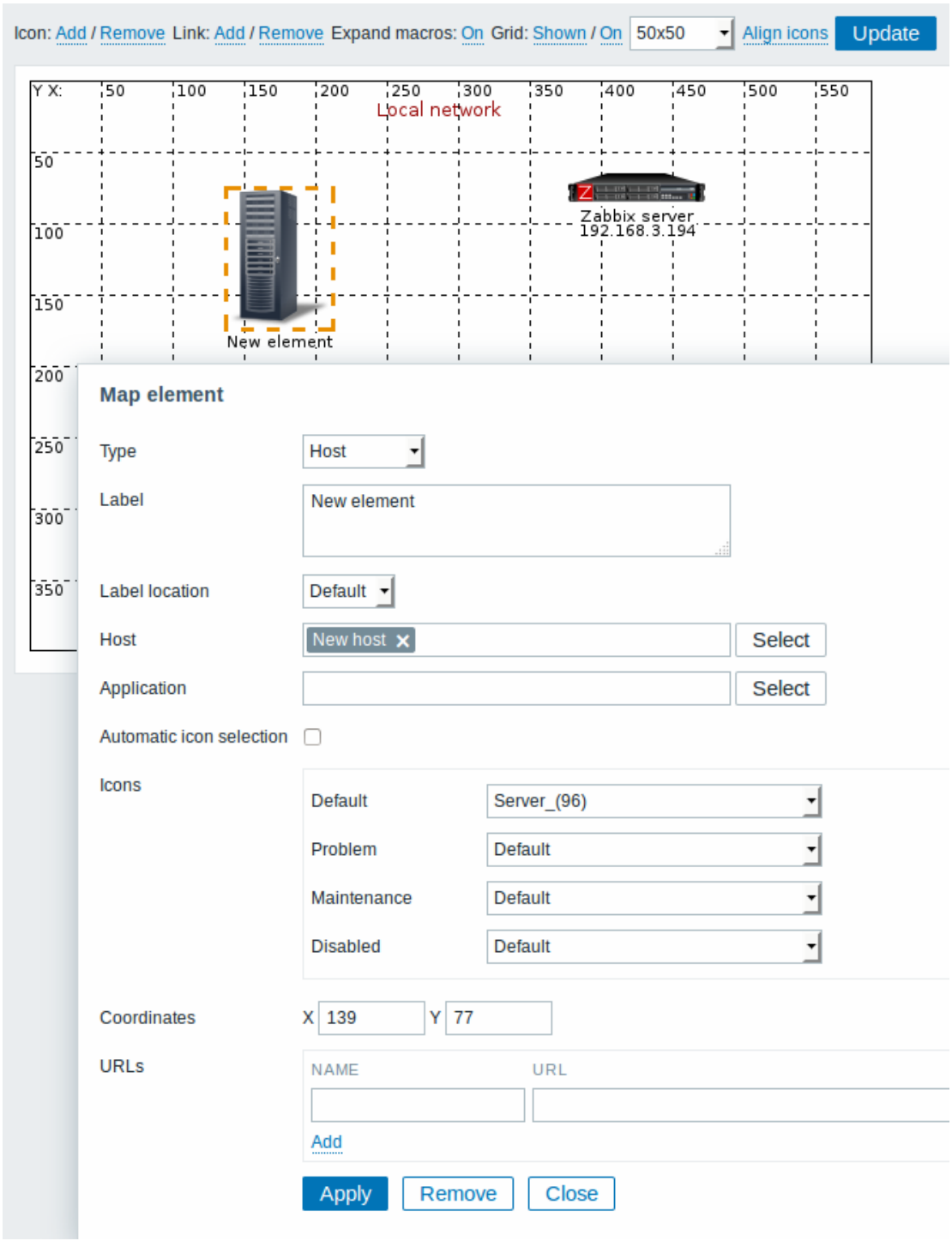


Рисунок 3.13 – Налаштування об’єктів

Вибір елементів

Щоб виділити один або декілька елементів, обираємо спочатку один елемент, потім натискаємо та утримаємо Ctrl, щоб вибрати інші рис.3.14.

Також можна вибрати кілька елементів, перемістивши прямокутник в області редагування і тим самим вибравши всі елементи, що знаходяться в ньому (опція доступна починаючи з Zabbix 2.0).

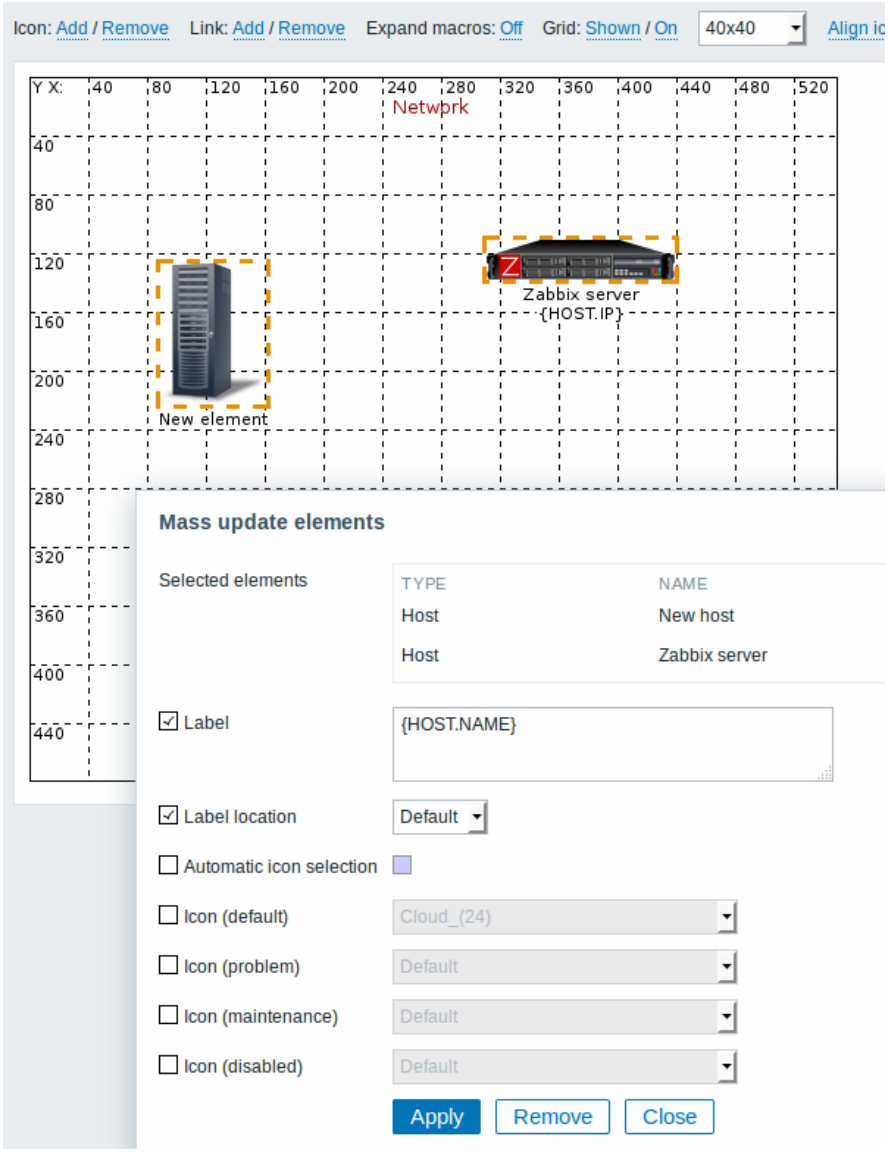


Рисунок 3.14 – Вибір елементів

Як тільки оберемо більше одного елемента, діалог властивостей елемента переключиться в режим масового оновлення, так що з’явиться можливість змінити атрибути вибраних елементів за один раз рис.3.14. Щоб це зробити, позначаємо атрибут, використовуючи прапорець, та вказуємо для нього нове значення. Тут можна використовувати макроси (такі як, скажімо, {HOSTNAME} для підпису елемента).

Зв’язність елементів

Після того, як розставили елементи на карті, саме час з’єднати їх. Для з’єднання двох елементів потрібно спочатку вибрати їх. Коли елементи будуть вибрані, натискаємо «Додати» праворуч від підпису Зв'язок.

За наявності створеного зв’язку діалог одного елемента тепер містить додатковий розділ Зв’язку. Натискаємо кнопку «Змінити», щоб редагувати атрибути зв’язку рис.3.15.

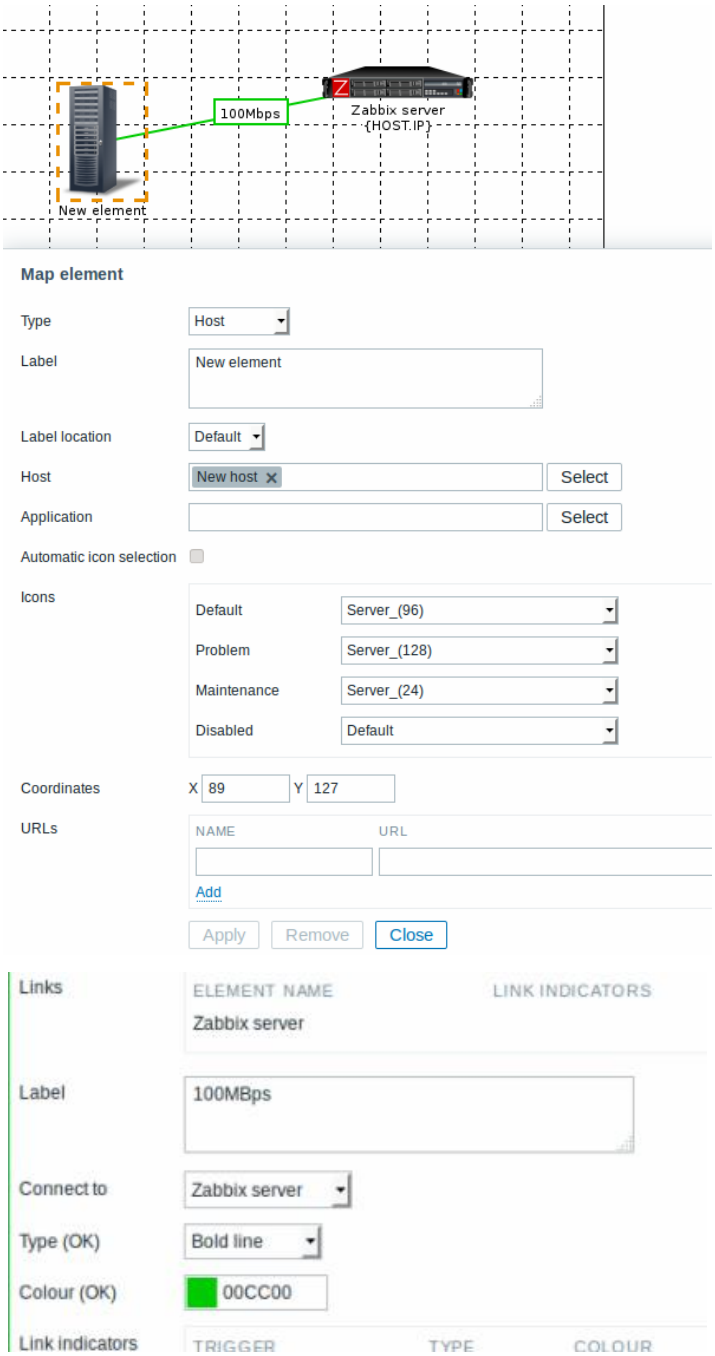


Рисунок 3.15 – Зміна артибутів елемента

Таблиця 3.7 – Атрибути зв’язків:

Парамери	Опис
Підпис	Підпис, який відображається поверх зв’язку. У цьому полі підтримується {узел_сети:ключ.функ(парам)} макрос, але з функціями тригеров avg, last, min і max, з секундами як параметра.
Підключатися через	Елемент, до якого приєднується зв’язок.
Тип (ОК)	Стиль зв'язку за замовчуванням: Лінія – одиночна лінія; Жирна лінія – жирна лінія; Точкова лінія – точки; Пунктирна лінія – пунктирна лінія.
Колір (ОК)	Колір зв'язку за замовчуванням.
Індикатори зв'язку	Список тригерів з’єднаних із зв’язком. У випадку, якщо тригер може ПРОБЛЕМА, його стиль застосовується до зв’язку.

3.3. Налаштування Grafana

На Zabbix сервері створюється "вузол мережі" (host), якому належать "елементи даних" (item) з метриками від наших вузлів. Імена вузлів та елементів даних бажано продумати заздалегідь і зробити максимально структурованими, тому що до них ми звертатимемося з графани через регулярні висловлювання. Такий підхід зручний тим, що можна одним запитом отримувати дані групи елементів.

Для налаштування Grafana потрібно встановити додаткові плагіни:

- Zabbix by Alexander Zobnin (alexanderzobnin-zabbix-app) – інтеграція з zabbix;
- natel-discrete-panel – плагін для дискретної візуалізації на горизонтальному графіку;
- pierosavi-imageit-panel – плагін для відображення даних поверх своєї картинки.

На Ubuntu Grafana встановлюється із репозиторію виробника. Скачаємо GPG-ключ і додамо його до списку надійних [13]:

```
wget -q -O -https://packages.grafana.com/gpg.key | sudo apt-key add-
```

Після цього додамо до системи репозиторій Grafana [13]:

```
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
```

Далі оновлюємо кеш APT та встановимо Grafana:

```
sudo apt update
sudo apt install grafana
```

Встановлення завершено. Grafana на Ubuntu 20.04 готова до використання. Перевіримо поточний статус сервісу [13]:

```
udo systemctl status grafana-server
```

Мережний екран дозволятиме підключення по порту 3000. У цьому можна переконатися, відкривши в браузері інтерфейс Grafana. Логін та пароль за замовчуванням admin рис.3.16.

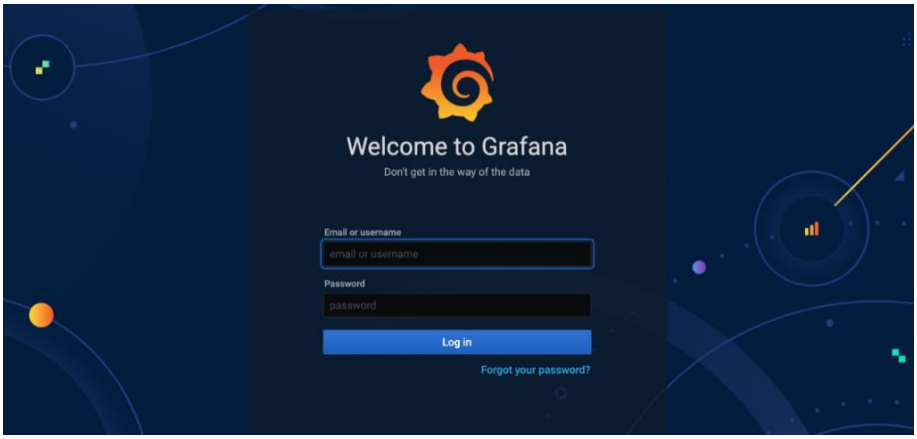


Рисунок 3.16 – Вікно вітання у Grafana

Якщо не вдається увійти до Grafana або змінили пароль і не можете його згадати, через CLI адміністративний пароль Web-інтерфейсу можна скинути [13]:

```
grafana-cli admin reset-admin-password --homepath
"/usr/share/grafana" новий_пароль
```

Після виконання команди вище можна входити з новим паролем.

Grafana підтримує різні джерела даних: Prometheus, Graphite, OpenTSDB, InfluxDB, Elasticsearch та інші рис. 3.17.

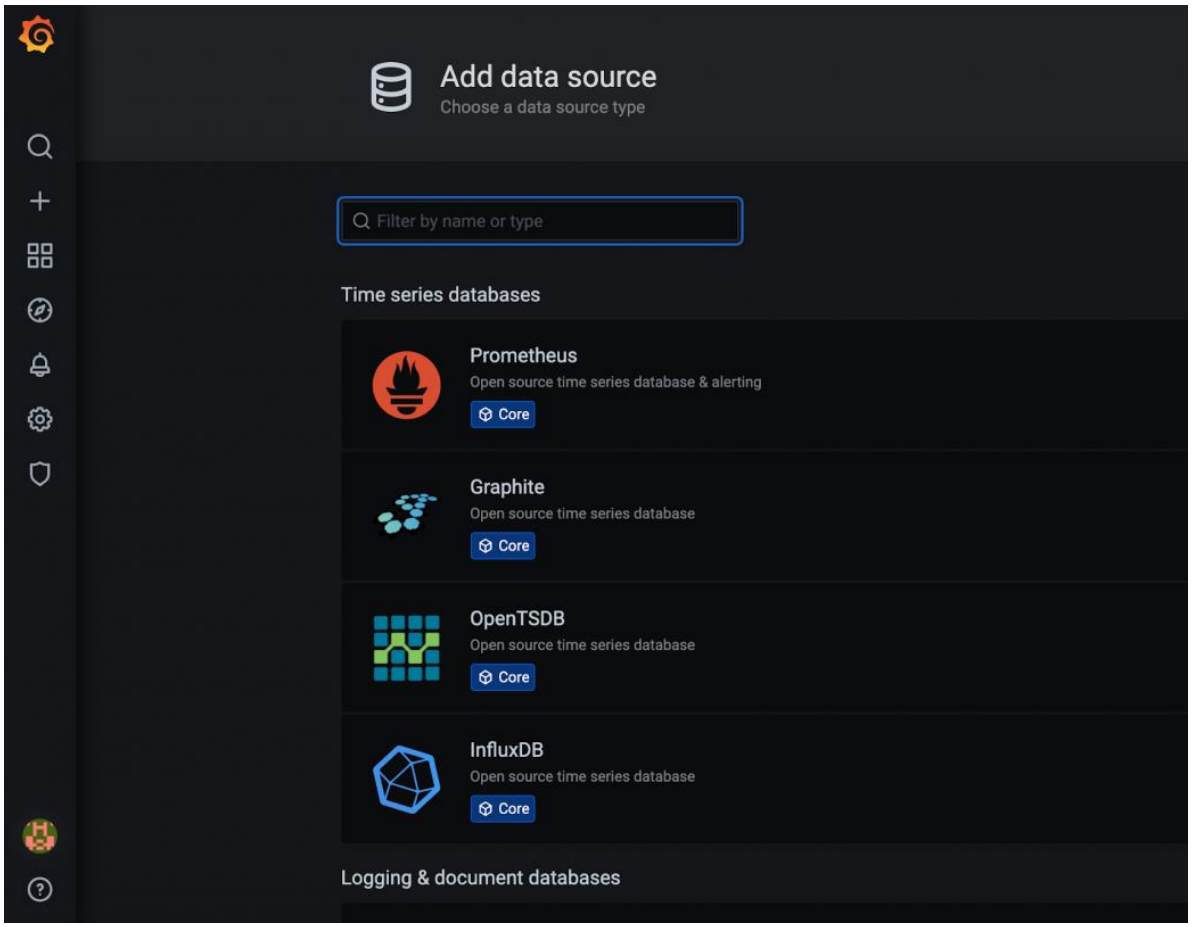


Рисунок 3.17 – Вибір джерела даних

Повний список можна переглянути в документації. Список вбудованих джерел даних може бути розширений за допомогою сторонніх плагінів. Один із таких плагінів — плагін Zabbix для Grafana. Моніторинг – одне з основних призначень використання Grafana. Встановимо цей плагін і додамо його до Grafana:

```
grafana-cli plugins install alexanderzobnin-zabbix-app
```

Тепер необхідно перезавантажити Grafana. В інтерфейсі Grafana перейдемо в Configuration->Plugins і знайдемо тут плагін для Zabbix.

Необхідно перейти до налаштування плагіна і натиснути кнопку Enable рис.3.18.

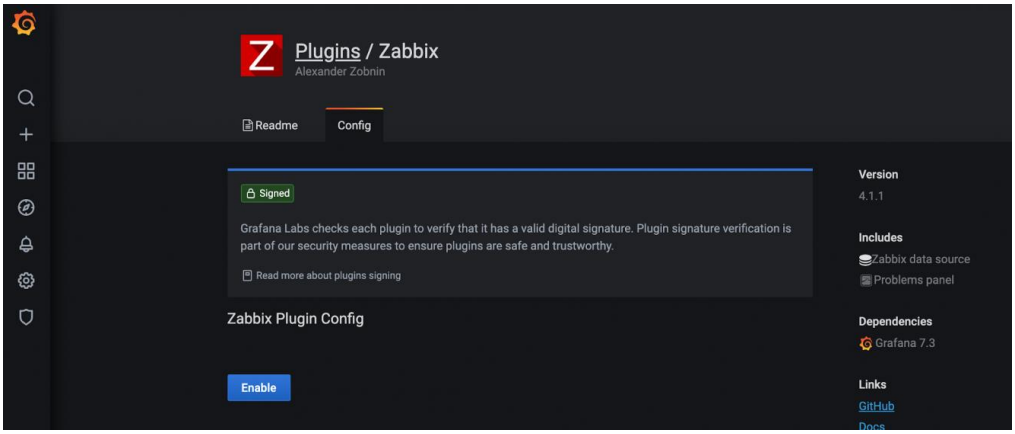


Рисунок 3.18 – Встановлення плагіна Zabbix

З моменту включення плагіна, в меню Data Sources з’явиться нове джерело даних Zabbix. Натисніть кнопку Select і налаштуємо джерело даних рис.3.18.

У налаштуваннях плагіна вкажемо URL-адресу, ім'я користувача та пароль для доступу API до Zabbix рис.3.19.

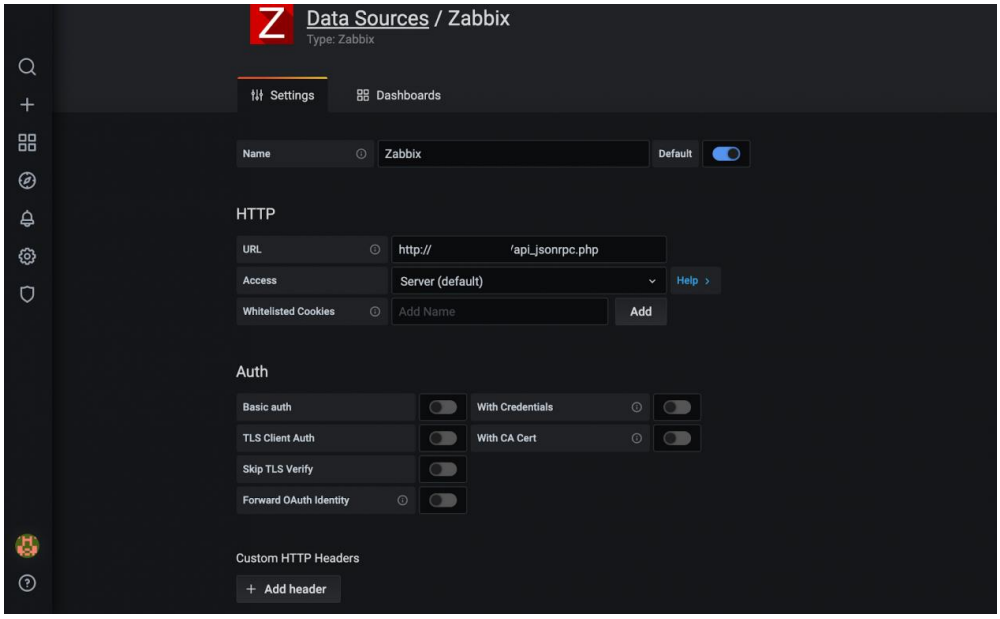


Рисунок 3.19 – З’єднання з системою моніторингу

Натискаємо на кнопку Save & Test та отримуємо підтвердження коректності налаштувань. З цього моменту дані з Zabbix можуть бути використані для створення візуалізацій рис.3.20.

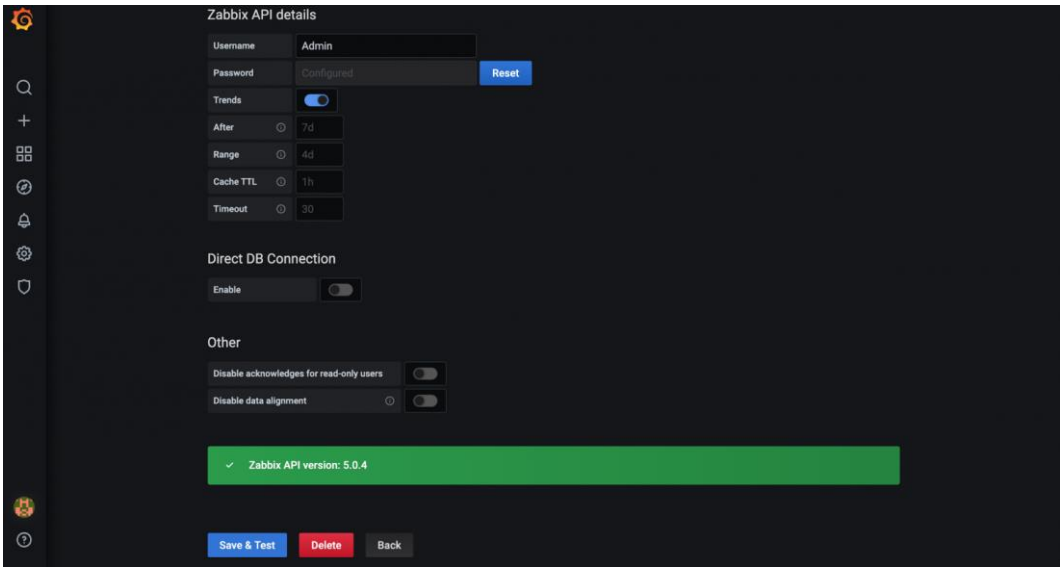


Рисунок 3.20 – Підключення до системи моніторингу

Панелі та дашборди – основні елементи представлення у Grafana. Кожен дашборд складається із набору панелей. Для створення дашбордів перейдемо у графу Dashboards і натиснемо на кнопку New Dashboard рис.3.21.

На наступному етапі буде запропоновано додати на дашборд нову панель.

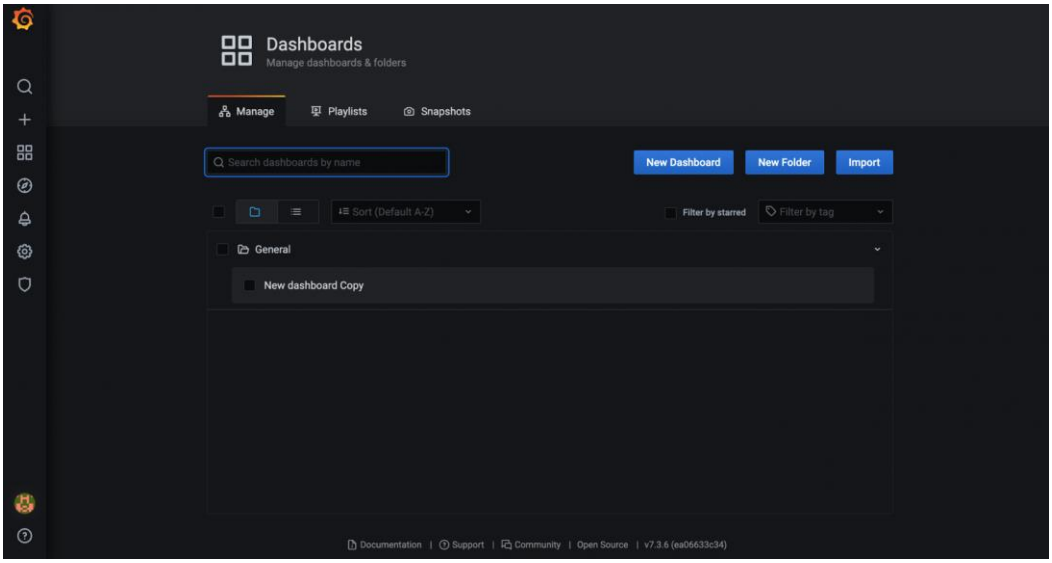


Рисунок 3.21 – Створення дашборду

Grafana має вбудовані панелі інструментів, які можна одразу починати використовувати. За замовчуванням наведено такі типи:

- Graph – панель з графіками з можливістю комбінувати кілька метриків на одній панелі;
- Stat (раніше SingleStat) - панель з одиночним графіком та можливістю відображення моментального значення метрики;
- Gauge - панель у форматі спідометра, можна обмежити верхнє значення на шкалі;
- Bar Gauge – панель з можливістю відображення кількох метриків на вертикальній гістограмі;
- Table — панель з представленням у вигляді таблиці, де можна відображати значення кількох метрик;
- Text — панель, щоб відображати довільний текст (підпис);
- Heatmap – панель для відображення теплової карти значень метрики;
- Alert list – панель для відображення подій із зовнішніх систем;
- Dashboard list – комбінована панель для відображення дашбордів, доданих до обраного;
- News — панель для відображення стрічки новин із зовнішніх джерел;
- Zabbix problems – панель для відображення подій із системи моніторингу Zabbix;
- Logs – панель для відображення рядків лога, які збираються однією із зовнішніх систем.

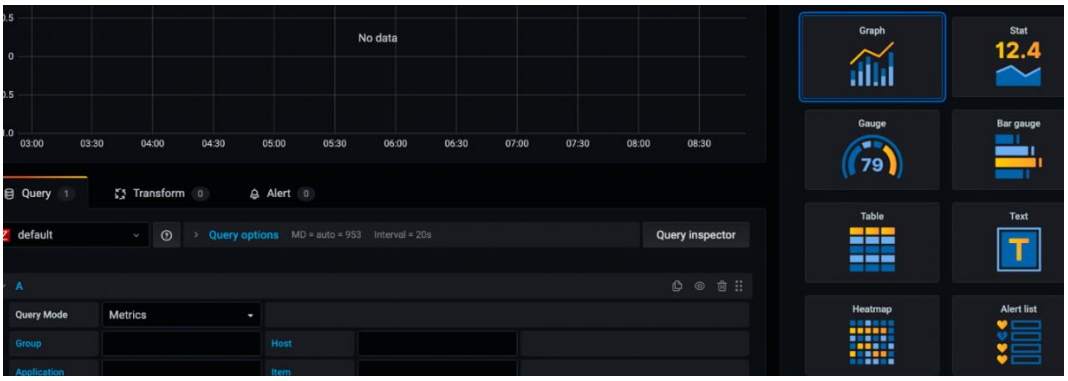


Рисунок 3.22 – Панелі інструментів за замовчуванням

Далі виберемо у випадаючому меню джерело даних Zabbix і вкажемо групу, хост, програму та елемент даних рис.3.22. Якщо все виконано правильно, на графіку з’являться дані рис. 3.23.

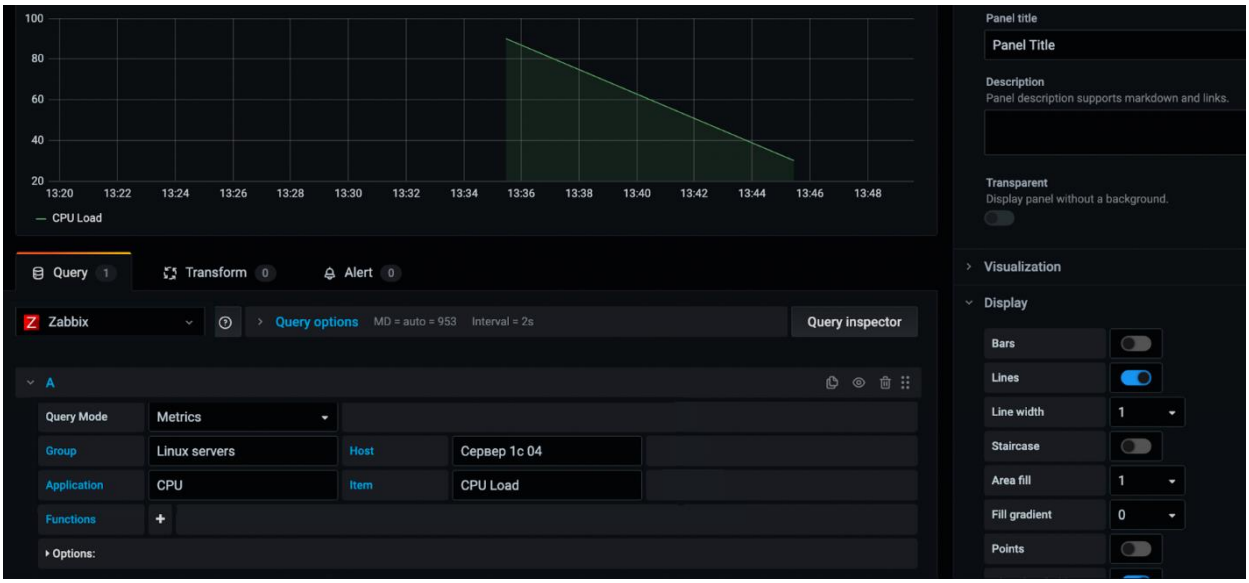


Рисунок 3.23 – Вибір джерела даних

Grafana підтримує різні типи візуалізацій: графічні, табличні, гістограми, теплові карти, карти мережевої взаємодії у Grafana та інші.

При створенні нової панелі, у правій частині екрана у розділі Visualization є можливість вибрати тип візуалізації. Перейдемо до створення SingleStat (відображається як Stat) - панелі з комбінацією чисельного та графічного уявлень. Після введення специфічних даних для Zabbix натисніть кнопку Apply рис.3.24.

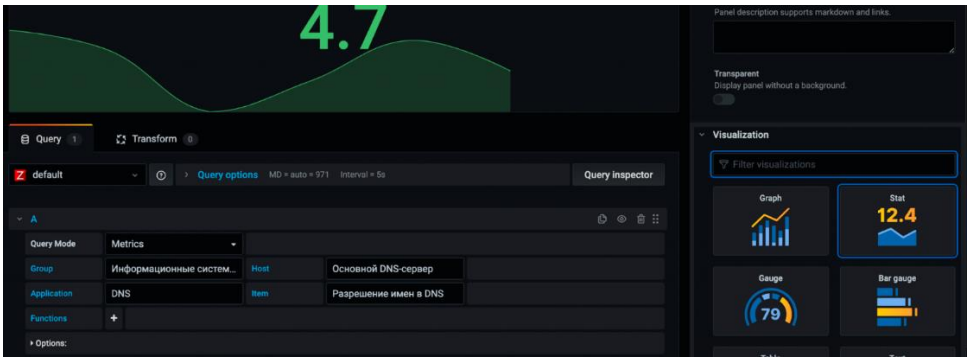


Рисунок 3.24 – Створення графічного відображення

В результаті виконаних дій побачимо на дашборді дві панелі. Кожна панель та кожен дашборд у Grafana мають свій набір налаштувань. Налаштування першої відкриваються через меню панелі, а налаштування другого відкриються після натискання на шестерню у верхньому правому куті екрана рис.3.25.

Рисунок 3.25 – Налаштування дашборду

3.4. Основний програмний модуль

Основним завданням роботи було удосконалити існуючі математичні моделі, та спростити процес розрахунку, щоб зекономити час на перевірку. Головними цілями удосконалення методики є:

- підвищення точності вимірювань каналі зв’язку;
- автоматизація та управління обладнанням зв’язку, та моніторинг системи;
- використання сучасних методів моніторингу з використанням програмних продуктів.

Методика автоматизованого розрахунку зв’язності шляху. Основні змінні для обчислення та зберігання даних у БД: t – час; t_m - час відновлення об'єкта ІТС; t_r – час, протягом якого об'єкт ІТС перебуває в робочому стані

Зв’язність шляху розраховується зі всіх імовірних шляхів від заданої початкової вершини до заданої кінцевої вершини, та справного стану всіх ліній зв’язку та вершин, які утворюють цей ланцюг:

$$k_{i,j}^{l} = \prod_{\forall i \in \mu_{i,j}^{l}} k_i \tag{3.1}$$

Де k_i — коефіцієнт готовності i-го об'єкту (вузлів і ліній зв’язку між ними), що належить шляху $\mu_{i,j}^{l}$.

Коефіцієнт готовності фрагмента, створеного паралельно з’єднаними засобами телекомунікацій, розраховується за формулою:

$$k_{\Sigma} = 1 - \left[\left(1-k_1\right) \times \left(1-k_2\right) \times \ldots \times \left(1-k_i\right) \times \ldots \times \left(1-k_n\right) \right] = 1 - \prod_{i=1}^{n} \left(1-k_i\right) \tag{3.2}$$

Тоді імовірність зв’язності шляху $k_{i,j}$, що був утворений паралельним з’єднанням ланцюгів — це імовірність працездатності хоча б одного ланцюга з усіх можливих ланцюгів:

$$k_{i,j} = k_{i,j}^{\max} = 1 - \prod_{\forall \mu_{i,j}^{l} \in \mu_{i,j}} \left(1 - k_{i,j}^{l}\right) \tag{3.3}$$

Проаналізувавши існуючі телекомунікаційні мережі, можна зробити висновок, що мережі побудованими таким чином, що ланцюги взаємозалежні – мають спільні вершини та лінії зв’язку. Імовірність, яку ми отримуємо при обчислені формули (3.3) має завищене значення. Тому, якщо отримане значення формули (3.3) виходить більше одиниці, то замінюємо це значення на одиницю. Для цього виконуємо дію поглинання E. Тоді після цього формула (3.3) приймає наступний вигляд:

$$k_{i,j} = E\left\{k_{i,j}^{\max} = 1 - \prod_{\forall \mu_{i,j}^{l} \in \mu_{i,j}} \left(1 - k_{i,j}^{l}\right)\right\} \quad (3.4)$$

Після цього розрахунок зв'язності обраного ланцюга закінчено, тоді переходимо до наступного та проводимо ті ж самі розрахунки.

Головний програмний модуль

Програмний модуль було розроблено для ПрАТ «НЕК «Укренерго». Було отримане технічне завдання, в якому було обрано мову розробки Python.

```
k1ij = []

total=1
for key in paths:
    count = 0
    while count < len(paths.get(key)[2]):
        total*= paths.get(key)[2][count]
        count+=1
    k1ij.append(total)

total = 1
for column in range(len(k1ij)):
    total *= (1 - k1ij[column])
total = 1 - total
```

Рисунок 3.26 – Розрахунок зв'язності шляху

Порівнюючи Python з іншими мовами, має ряд переваг, а саме:

- низький поріг входження;
- логічний, лаконічний та зрозумілий. Якщо порівнювати з іншими мовами, то має легший синтаксис;
- кросплатформовий: підходить для різних операційних систем: і Linux, і Windows;
- є реалізація інтерпретаторів для мобільних пристроїв та непопулярних систем.
- популярність. Використовується для розробки веб-додатків, ігорових додатків, зручний для автоматизації, математичних обчислень;

- висока затребуваність ринку праці;
- багато якісних бібліотек, для навчання є багато тлумачних книг, насамперед англійською мовою, звичайно, але й у перекладі також видано гідну літературу;
- Python відрізняється суворою вимогою до написання коду (вимагає відступів), що є перевагою, мова сприяє писати код організовано та красиво.

Отже, основний програмний модуль було розроблено на мові Python, функція, яка розраховує зв’язність всього ланцюга має наступний вигляд рис.3.26:

Візуалізація карти у Grafana

Було задано граф, який має наступний вигляд рис.3.27:

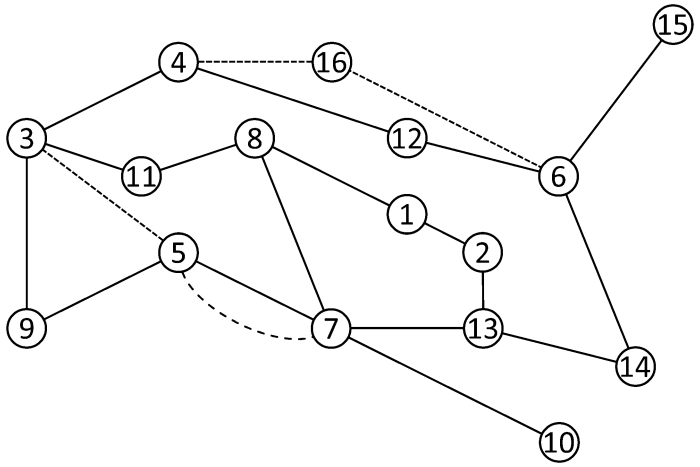


Рисунок 3.27 – Київське кільце мережі ПрАТ «НЕК «Укренерго»

Після отриманого графу було сформовано карту покриття у Grafana, вона включає у себе всі вершини та лінії зв’язку між ними рис.3.28. На самій карті відображення ліній зв’язку не доступне, проте у спеціальному режимі можна переглянути карту покриття у вигляді графу. На самій карті є два варіанти міркувань: червоне та зелене. Червоне відповідає за відмову, а зелене за працездатність шляху.

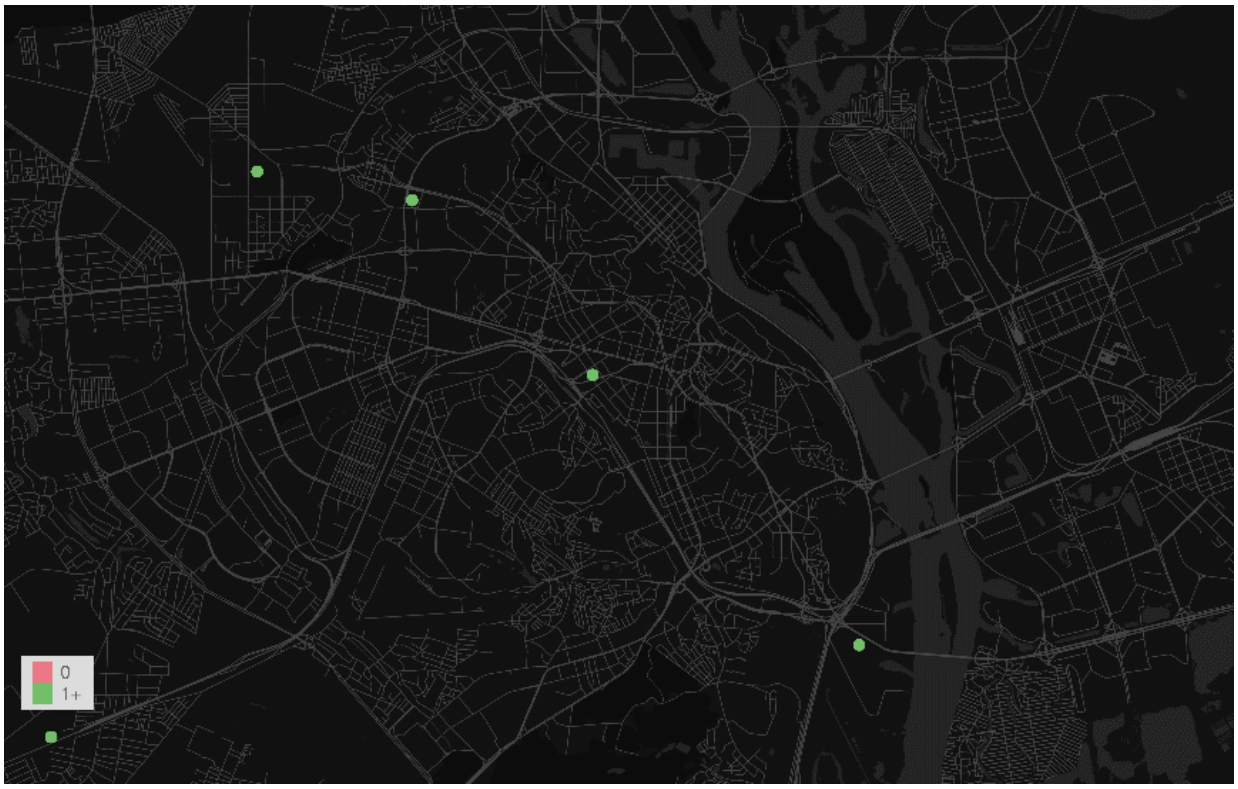


Рисунок 3.28 – Працездатні вершини у місті Києві

Якщо на лінії стався збій, то вершина або ребро буде підсвічуватись червоним кольором. У спеціальній вкладці можна простежити динаміку збою у вигляді графіка. За допомогою часової шкали можна встановити, коли надійшов найбільш сильний сигнал про збій рис.3.29.

Рисунок 3.29 – Відстеження збою у Grafana

ВИСНОВКИ

В результаті досліджень під час магістерської роботи було знайдено та вирішено проблему у сучасних математичних моделях, які відповідають за функціональну стійкість телекомунікаційних мереж. Дане наукове дослідження має ряд переваг серед своїх попередників, які можуть забезпечити потреби сучасних телекомунікаційних мереж. Під час дослідження було виявлено, що на сьогоднішній день не існує продукту, що забезпечує всі проблеми мереж як з теоретичної точки зору, так і з програмної, тому це робить дану розробку унікальною.

В магістерській роботі одержано такі результати:

1. Під час проведення аналізу серед існуючих аналогів, та математичних моделей виявлено проблему у вигляді складності обчислень показників функціональної стійкості мережі. У сучасних моделях науковці пропонують робити обчислення спираючись на теорію ймовірності, проте розроблені розрахунки мають досить складну та довгу реалізацію, що робить їх для сучасних телекомунікаційних мереж не актуальними.
2. Удосконалено інформаційну технологію для зв’язності шляху при наявності відмов ліній зв’язку, або вершин мережі.
3. Проведено аналіз щодо програмної надійності, та розроблено систему з забезпеченням повної сумісності обладнання.
4. Розроблено систему моніторингу в режимі реального часу з використанням удосконалених математичних розрахунків, що дозволяє слідкувати за станом мережі, та сповіщати про несправності на лініях зв’язку.

На основі проведеного аналізу було сформоване чітке технічне завдання щодо розробленого програмного продукту, визначено основні недоліки серед існуючих систем моніторингу, сформовано вимоги щодо стійкості програмного забезпечення.

Після формування технічного завдання було здійснено вибір програмних продуктів, за допомогою яких було реалізовано систему моніторингу. Для

реалізації програми, було обрано операційну систему Linux, адже ця операційна система забезпечує повну надійність та простоту для розгортання серверу. Було проведено та обгрунтовано вибір ОС з операційною системою Windows.

Основний програмний модуль з удосконаленою математичною частиною був розроблений за допомогою мови програмування Python. Підключення даного модулю здійснюється у Zabbix або Grafana, в залежності від того, що саме треба переглянути.

Додаток моніторингу системи розроблений для ПрАТ «НЕК «Укренерго». Від компанії було надано стандарт, щодо існуючого покриття мережі. При ознайомлені зі стандартом було сформовано чіткі вимоги щодо програмного продукті конкретно для їх телекомунікаційної мережі.

Система Zabbix допомагає переглянути всі лінії зв’язку, та вершини у вигляді списку. Також модуль може робити прогнозування відносно збоїв на основі зібраних даних за весь період роботи системи. Проведено аналіз та обгрунтовано вибір системи моніторингу серед аналогів. Zabbix дозволяє зберігати дані за довготривалий час, що для телекомунікаційних мереж є важливою складовою, для перегляду та вивантаження статистичних даних щодо працездатності мережі загалом.

Відображення графічної частини, тобто відображення покриття мережі на території Києва було здійснено у Grafana. У Grafana підключено основний програмний модуль та реалізовано зв’язок з системою Zabbix, де є сформовані тригери для попередження збоїв у системі. Grafana дозволяє переглянути працездатність системи у вигляді карти та певних позначок на ній. Також є можливість отримати дані з бази даних та Zabbix та сформувати їх у вигляді графіків та діаграм, що робить додаток інтуїтивно простим для першого використання, та користування програмою в цілому.

Наведено основні принципи та опис проектування користувацького інтерфейсу. Представлено процес тестування розробленої системи моніторингу.

Таким чином, розроблена система моніторингу є першою програмною реалізацією серед свої аналогів для телекомунікаційних мереж. Вона забезпечує зручний моніторинг системи в режимі реального часу, та на сьогоднішній день має математичну модель, що робить розрахунки на функціональну стійкість за короткий час, що дозволяє у разу відмов швидко зреагувати та перекинути сигнал на іншу лінію зв’язку.

В подальшому розроблену систему можна вдосконалити. Одним з можливих варіантів вдосконалення є доопрацювання функціоналу користувацького інтерфейсу, та у разі виникнення нових потреб, доопрацювання програмного модулю з використанням штучного інтелекту, що дозволить зробити більш швидкі та точні прогнозування щодо можливих збоїв на лініях зв’язку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гері М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. — 416 с.

2. Барабаш О.В. Оцінювання показника функціональної стійкості графа структури розгалуженої інформаційної мережі / О.В. Барабаш, І.П. Саланда // Науково-практичний журнал «Зв'язок». – Київ: ДУТ, 2015. – № 2 (114). – С. 9 – 12.

3. Барабаш О.В. Забезпечення функціональної стійкості телекомуніка-ційних систем / О.В. Барабаш, І.П. Саланда // Тези доповідей П’ятої міжнародної науково-технічної конференції «Проблеми інформатизації», Київ – Полтава – Кіровоград – Орел – Белгород – Харків.11-12 грудня 2015 року, – Київ: ДУТ, 2015. – С. 34.

4. Barabash O. Diagnostic Model of Wireless Sensor Network Based on Mutual Inspection of Network Elements / O. Barabash, N. Lukova-Chuiko, A. Musienko, I. Salanda // Proceedings of 14 International Conference the Experience of Designing and Application of Cad Systems in Microelectronics (CADSM 2017), 21-25 February, 2017, Polyana-Svalyava (Zakarpattya), Ukraine. – Lviv: Lviv Polytechnic National University, 2017. – P. 303 – 305.

5. Кучук Г.А. Управление ресурсами инфо-телекоммуникаций / Г.А. Кучук, Р.П. Гахов, А.А. Пашнев. – М.: Физматлит, 2006. – 220 с.

6. L. Nazir, R. N. Mir (2016) Realization of Efficient High Throughput Buffering Policies for Network on Chip Router. International Journal of Computer Network and Information Security (IJCNIS), Vol. 8, No. 7, pp. 61 – 70.

7. Буданов П.Ф. Экспериментальные исследования пространственно-временной модели информационного пространства для процесса формирования случайного сигнала с признаками аварийности / П.Ф. Буданов, К.Ю. Бровко // Системи обробки інформації. – Харків: ХНУПС. – 2016. – Вип. 3 (140). – С. 227 – 233.

8. Гнатюк С.Є. Аналітична модель надійності програмних засобів комп’ютерних систем і програмно-керованих засобів зв’язку / С.Є. Гнатюк // Наука

9. і техніка Повітряних Сил Збройних сил України. – Харків: ХУПС, 2014. – № 3 (16). – С. 104 – 108.

10. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир. 1985, 524 с.

11. Матвеевский В.Р. Надежность технических систем – Уральский государственный горный университет.

12. Поповский В.В. Математическое моделирование связности телекоммуникационных систем с использованием симплициальных комплексов / В.В. Поповский, А.В. Лемешко // Праці УНДІРТ. – 2000. – № 2 (22). – С. 79 – 82.

13. Зайченко Ю.П. Структурная оптимизация сетей ЭВМ / Ю.П. Зайченко, Ю.В. Гонта. – Киев: Техника, 1986. – 167 с.

14. Grafana.com [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2001-2021 Grafana] – Режим доступа: https://grafana.com/docs/grafana/latest/installation/debian/

15.Zabbix.com [Електронний ресурс] : [Інтернет портал]. – Електронні дані. – [2001-2021 Zabbix] – Режим доступа: https://www.zabbix.com/documentation/current/

16. Жебка В. В. Моніторинг сталості інформаційно-телекомунікаційної системи і опрацювання заходів її захисту від небезпек / В. В. Жебка, П. В. Анахов // Метрологія та прилади. – 2021. – №1(87). – С. 23-29.

17.Andrea Dalle Vacche Mastering Zabbix. Packt Publishing

18.Степанченко И. В. Методы тестирования программного обеспечения; навч. посібник - ВолгГТУ, Волгоград, 2006. – 74 с.

19. Zabbix Overview [Електронний ресурс] – https://www.zabbix.com/documentation/2.4/ru/manual/introduction/overview

20. Обідін Д. М. Ознаки та критерії функціональної стійкості інтелектуалізованої системи автоматичного управління польотом літака. / Д.М. Обідін, О.В. Барабаш // Системи озброєння і військова техніка: Науковий журнал. – Харків: ХУПС, 2012. – № 1 (29). – С. 133 – 136.

21. Мур Э., Шеннон К. Надежные схемы из ненадежных реле // Кибернетический сб. – М.: Иностр. лит., 1960. – Вып. 1. – С. 109 – 148.

ДОДАТОК

Слайд 1

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
«УДОСКОНАЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПІДВИЩЕННЯ ФУНКЦІОНАЛЬНОЇ СТІЙКОСТІ МЕРЕЖІ ЗА ДОПОМОГОЮ ТЕОРІЇ ГРАФІВ»

Виконав: студентка групи ПДМ-61, Балашова Єлизавета Олександрівна

Керівник: , к.т.н., доц. кафедри ІПЗ, Жебка Вікторія Вікторівна

Київ - 2021

Слайд 2

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета роботи: моніторинг системи в режимі реального часу за допомогою програмного продукту за домопогою теорії графів.

Об'єкт дослідження: стійкість мережі.

Предмет дослідження: додаток на основі удосконаленої інформаційної технології стійкості мережі.

Слайд 3

АНАЛІЗ ІСНУЮЧИХ ІТ-РІШЕНЬ ДЛЯ МОДЕЛЮВАННЯ 4

Prometheus – одна з набільш популярних систем для розгортання систем моніторингу.

Проте вона має ряд недоліків, а саме:

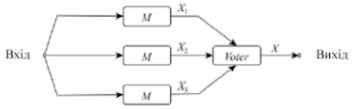
- Зберігає лише значення часових рядів. Не підходить для тексту, логів чи журналів подій.
- У браузера немає функцій повноцінної панелі моніторингу.
- Розрахований на короткострокове (тиждень-два) зберігання метрик та роботи з ними.

Слайд 3

ВІДМОВОСТІЙКІСТЬ СИСТЕМИ

У разі повної відмови стійкості елемент, що відмовив, повинен бути замінений справним, який візьме виконання відповідних функцій. Це означає, що відмовостійка система повинна мати певну надмірність.

Статична надмірність означає, що надлишкові компоненти використовуються як стала система. Пристрій голосування Voter :



При динамічній надмірності система перебудовується так, що функції відмовив елемента передаються справному елементу. При цьому виділяються додаткові завдання:
1. **Виявлення відмови** – діагностичні процедури повинні дозволяти виявляти елемент, що відмовив.
2. **Виняток відмови** – елемент, що відмовив, виключається з системи і або замінюється справним, або система реконфігурується, щоб уникнути використання елемента, що відмовив.
3. **Відновлення** – виконується приведення системи до робочого стану.

Слайд 4

МОДЕЛЬ НА ОСНОВІ НЕОРІЄНТОВАНИХ ГРАФІВ

Неорієнтовані графи з рівнонадійними ребрами, тобто будь-яке ребро, що належить до графа G, може бути видалено з графу з ймовірністю p.
Будемо вважати, що $G = (V, E)$ – граф з n вершинами V(G)={v1, v2, ..., vn} і m ребрами $E(G) = \{e1, e2, \ldots, em\}$, то через $G\backslash E'$ позначимо суграф графа G, $E' \subset E(G)$.

Тоді ймовірність зв'язності $P_{зв.}(G, p)$ графа G обчислюється за формулою:

$$P_{зв.}(G,p) = \sum_{E' \subset E(g)} f(G\backslash E') p^{|E'|}(1-p)^{m-|Er|}$$

де $f(G\backslash E') = 1$, якщо суграф вважається зв'язним,

та $f(G\backslash E') = 0$, якщо суграф не зв'язний.

Слайд 5

НАЛАШТУВАННЯ СИСТЕМИ МОНІТОРИНГУ

Zabbix

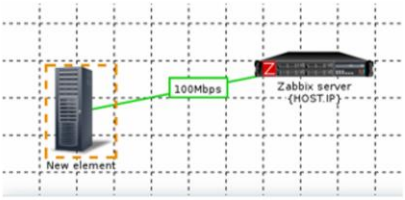
Grafana

Слайд 6

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ 8

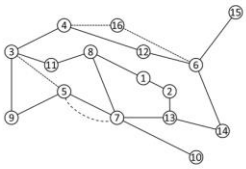
Zabbix – це програмне забезпечення для моніторингу числових параметрів мережі, життєздатності та цілісності серверів. Використовує гнучкий механізм оповіщень.

ZABBIX



Слайд 7

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ 8



Grafana

Слайд 8

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ 4

Науковий результат- розробка системи моніторингу в режимі реального часу.

python™

```
k1ij = []

total=1
for key in paths:
    count = 0
    while count < len(paths.get(key)[2]):
        total*= paths.get(key)[2][count]
        count+=1
    k1ij.append(total)

total = 1
for column in range(len(k1ij)):
    total *= (1 - k1ij[column])
total = 1 - total
```

Слайд 9

ВИСНОВКИ

В результаті роботи було розроблено систему моніторингу, яка відповідає за цілодобове спостереження системи, та своєчасно сповіщає про наявність проблеми у телекомунікаційній мережі.

При виконанні роботи було виконано наступні задачі.

1. Проведено аналіз щодо програмної надійності, та розроблено систему з забезпеченням повної сумісності обладнання.
2. Розроблено систему моніторингу в режимі реального часу з використанням удосконалених математичних розрахунків, що дозволяє слідкувати за станом мережі, та сповіщати про несправності на лініях зв'язку.
3. Удосконалено інформаційну технологію для функціональної стійкості при наявності відмов ліній зв'язку, або вершин мережі.

Слайд 10

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ

Статті:

1. Стаття написана у співавторстві зі студенткою групи ПДМ-61, Балашовою Єлизаветою. Стаття в №3 журналу Телекомунікаційні та інформаційні технології. ДУТ.

Тези доповідей на конференціях:

1. Тези на тему «Удосконалення інформаційної технології для підвищення функціональної стійкості мережі за допомогою теорії графів», XIII науково-технічна конференція ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ», ДУТ

Слайд 11

ДЯКУЮ ЗА УВАГУ!