ВІДЗИВ НАУКОВОГО КЕРІВНИКА НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студента (ки) **Михайлова В. І.**
(прізвище та ініціали)
представлену на здобуття освітнього ступеня «бакалавр»
зі спеціальності 123 «Комп'ютерна інженерія»

1. Робота виконана на тему Навчально-тренувальна програма для розрахунку метрик динамічних протоколів маршрутизації

2. Актуальність. Постійний розвиток телекомунікаційних мереж вимагає належного рівня підготовки все більшої кількості користувачів, зростає кількість виданих навчальних посібників та наукових праць, проте більшість з них стосується розробниками протоколів маршрутизації або мережевого обладнання. Тому є необхідність в незалежному аналізі наявних протоколів маршрутизації телекомунікаційних мереж. Отож, головна ціль даної бакалаврської кваліфікаційної роботи є створення навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації. Дане програмне рішення буде чудовим інструментом для покращення знать та навичок з комп'ютерних мереж для студентів вищих навчальних закладів..

(зазначити, які питання теоретичного і практичного характеру визначають актуальність роботи)

3. Основний зміст роботи викладено на 62 сторінках тексту, містить 20 ілюстрацій. Список використаних літературних джерел налічує 26 найменувань.

4. Зміст розділів роботи *(зазначити за кожним розділом, які питання досліджені та з якою метою)*

РОЗДІЛ 1. Теоретичний аналіз протоколів маршрутизації. У даному розділі описано особливості протоколу OSPF, здійснено аналіз аналогів програмного продукту, наведено класифікацію протоколів маршрутизації за типом застосованого алгоритму маршрутизації, проаналізовано концептуальні основи маршрутизації, метрики маршрутів.

РОЗДІЛ 2. Проектування навчально-тренувальної програми для розрахунку метрик динамічної маршрутизації. У розділі здійснено опис статичного та динамічного типу маршрутизації, наведено алгоритм метрик маршрутизації, здійснено вибір та обґрунтування засобів вирішення проблеми.

РОЗДІЛ 3. Інтерфейс програми та робота з нею. Описано та реалізовано поставлене завдання, здійснено аналіз отриманих результатів.

5. Висновки обґрунтовані, повністю відповідають поставленим завданням, мають теоретичне та практичне значення. Практичне значення отриманих результатів полягає у тому, що проведено аналіз протоколів маршрутизації та створено алгоритм для побудови навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації, що дало б змогу полегшити навчальний процес у закладах вищої освіти.

(обґрунтованість, теоретичне і практичне значення)

6. Список використаних літературних джерел. Джерела опрацьовані у повному обсязі, відповідають змісту роботи.

(обсяг і повнота опрацювання за змістом роботи)

7. Додатки

(перелік додатків та повнота їх опрацювання за змістом роботи)

8. Таблиці та ілюстрації Ілюстрації носять інформаційний характер, є доповненням до змісту роботи.

(призначення таблиць та їх використання для підтвердження досліджень в роботі)

9. Якість оформлення роботи на достатньому рівні

10. Ставлення студента (ки) до виконання випускної кваліфікаційної роботи відповідально

11. Висновок

В цілому при написанні випускної кваліфікаційної роботи студент (ка) **Михайлов В. І.**

(прізвище та ініціали)

вміння /невміння/ підібрати необхідну літературу, інформаційну базу, проаналізувати їх, зробити відповідні висновки і оформити свої дослідження у випускній кваліфікаційній роботі.

Випускна кваліфікаційна робота студента (ки) **Михайлова В. І.**

(прізвище та ініціали)

написана на достатньо /недостатньо/ високому рівні та може /не може/ бути допущена до захисту в Екзаменаційній комісії.

Науковий керівник д. п.н., доцент, професор кафедри інженерії програмного забезпечення

(вчений ступінь, звання, посада)

(підпис) Вакалюк Т.А.

(прізвище та ініціали)

«17» 06 2020 р.

РЕЦЕНЗІЯ
на кваліфікаційну роботу
студента 4 курсу групи КІ-2 спеціальності 123 «Комп'ютерна інженерія»
факультету інформаційно-комп'ютерних технологій
Державного університету «Житомирська політехніка»
Михайлова Вячеслава Ігоровича
на тему: „Навчально-тренувальна програма для розрахунку метрик протоколів динамічної маршрутизації"

Кваліфікаційна робота студента Михайлова В.І. „Навчально-тренувальна програма для розрахунку метрик протоколів динамічної маршрутизації" присвячена досить актуальній тематиці – розробці навчальних програмних засобів для вивчення спеціалізованих тем у сфері комп'ютерних мереж.

Робота містить Web-орієнтований програмний продукт та пояснювальну записку наступного складу: вступ, три розділи основного тексту, висновки, список використаних джерел та літератури, додатки з програмним кодом.

У першому розділі пояснювальної записки зроблено аналіз теоретичних аспектів маршрутизації, розглянуто основні поняття маршрутизації, класифікацію протоколів маршрутизації, розглянуто принципи функціонування протоколу OSPF. У розділі також наведено аналіз аналогів розроблюваного програмного продукту.

У другому розділі пояснювальної записки здійснено проектування навчально-тренувальної програми, наведено огляд статичної і динамічної маршрутизації, зроблено аналіз параметрів для формування метрик протоколів RIP та OSPF. Також у розділі виконано вибір та обгрунтування засобів розробки програмного продукту.

У третьому розділі пояснювальної записки наведено опис інтерфейсу розробленої програми, зокрема, наведено опис та реалізації завдання.

Зміст роботи в цілому відповідає поставленому завданню. Кваліфікаційна робота виконана на відносно низькому рівні. Тема роботи розкрита не повністю. Студентом продемонстровані достатні фахові компетентності, отримані під час навчання та у процесі виконання роботи. Матеріал пояснювальної записки викладено не зовсім логічно і грамотно. Ілюстративний матеріал (рисунки, таблиці), що наведений у роботі, оформлений відносно якісно. Оформлення роботи у цілому відповідає вимогам.

Недоліками роботи є наступні:

– неналежна структуризація теоретичного матеріалу роботи;

– неповний розгляд питань формування та розрахунку метрик (у роботі не наведені і не проаналізовані формули та алгоритми розрахунку метрик основних протоколів маршрутизації);

– неналежно проведений аналіз аналогів програмного продукту;

– у роботі не наведений опис структури програмного продукту;

– реалізація програмного продукту є досить примітивною;

– у пояснювальній записці наявні відхилення від вимог до оформлення (розміщення розділу 1.1; описи протоколів стор. 28, 29, 30; вільне місце на стор. 44 тощо), граматичні та стилістичні помилки.

У цілому кваліфікаційна робота студента Михайлова В.І. „Навчально-тренувальна програма для розрахунку метрик протоколів динамічної маршрутизації" умовно відповідає вимогам рівня вищої освіти «бакалавр» спеціальності 123 «Комп'ютерна інженерія» і може бути допущена до захисту перед ЕК

Пропонована оцінка роботи – «задовільно» (за умови успішного захисту).

Рецензент,
завідувач кафедри КІ та КБ,
к.т.н
19 червня 2020 р.



А. А. Єфіменко

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему:

«Навчально-тренувальна програма для розрахунку метрик динамічних протоколів маршрутизації»

Виконав студент 4-го курсу, групи КІ-2
спеціальності 123 «Комп’ютерна інженерія»
_____ В.І. Михайлов
Керівник, доктор педагогічних наук,
доцент, професор кафедри інженерії
програмного забезпечення
_____ Вакалюк Т. А.
Рецензент, завідувач кафедри КІ та КБ.,
к.т.н., доцент
_____ Єфіменко А. А.

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
_____ Єфіменко А. А.
«06» квітня 2020 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу

Студента **Михайлова Вячеслава Ігоровича**

Тема роботи: **Навчально-тренувальна програма для розрахунку метрик динамічних протоколів маршрутизації**
затверджена Наказом по університету від **«06» квітня 2020 р. № 98с**
Термін здачі студентом закінченої роботи – **1 червня 2020 р.**
Вихідні дані для роботи:

1. Аналіз проблематики, методів та засобів вирішення задач
2. Проектування та розробка програмного забезпечення
3. Опис роботи з програмним додатком та його тестування

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1,2,3	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	06.04.2020 р.	06.04.2020 р.
2	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	27.04.2020 р.	27.04.2020 р.
3	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	18.05.2020 р.	18.05.2020 р.

Керівник _____ Вакалюк Т.А.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	6 квітня 2020 – 7 квітня 2020	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	8 квітня 2020 – 11 квітня 2020	Виконано
3	Формулювання технічного завдання	12 квітня 2020 – 16 квітня 2020	Виконано
4	Опрацювання літературних джерел	17 квітня 2020 – 19 квітня 2020	Виконано
5	Проектування структури	20 квітня 2020 – 24 квітня 2020	Виконано
6	Написання програмного коду	25 квітня 2020 – 30 квітня 2020	Виконано
7	Відлагодження	1 травня 2020 – 4 травня 2020	Виконано
8	Тестування роботи системи	5 травня 2020 – 9 травня 2020	Виконано
9	Аналіз результатів тестування	10 травня 2020 – 13травня 2020	Виконано
10	Написання пояснювальної записки	25 травня 2020 – 29 травня 2020	Виконано
11	Підготовка презентації та виступу	30 травня 2020 – 31 травня 2020	Виконано
12	Отримання відгуку керівника та рецензії на кваліфікаційну роботу	4 червня 2020 – 8 червня 2020	Виконано
13	Захист	21 червня 2020	Виконано

Студент _____ Михайлов В.

Керівник_____ Вакалюк Т.А

РЕФЕРАТ

Кваліфікаційна робота складається з навчально-тренувальної прогр для розрахунку метрик протоколів динамічної маршрутизації пояснювальної записки до нього. Пояснювальна записка до роботи містит сторінки та 20 ілюстрацій.

Метою роботи є розробка навчально-тренувальної програми розрахунку метрик протоколів динамічної маршрутизації.

В роботі визначено основні завдання на розробку підсистеми, забезпечить полегшення навчального процесу у закладах вищої освіти а та допоможе закріпити навички та знання у протоколах динамі маршрутизації. В роботі проаналізовано засоби для створення програмн забезпечення, розглянуто існуючі аналоги програмного проду обґрунтовано актуальність даної системи та описана сфера її використа Створено графічні матеріали з метриками маршрутизацій та внесено у б запитань для використання у програмному продукті. Створено навчаль тренувальну програму для розрахунку метрик протоколів динаміч маршрутизації та про проведено тестування даного програмного засобу.

КЛЮЧОВІ СЛОВА: ЗАХИСТ, МЕРЕЖА, ПЕРИМЕТР МЕРЕ CISCO, МАРШРУТИЗАТОР, ПРОТОКОЛ, МЕТРИКИ МАРШРУТІВ.

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА. 20.123.18.000 -	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Михайлов В.І.		17.06	Навчально-тренувальна програма для розрахунку метрик динамічних протоколів маршрутизації	Літ. / Арк. 4 / Ар
Керівник		Вакалюк Т.А..		17.06		
Н.контроль		Єфіменко А.А.		19.06.20		
Рецензент		Єфіменко А.А.		19.06.20		ФІКТ, група КІ-
Зав. каф.		Єфіменко А.А.		19.06.20		

ABSTRACT

The qualification work consists of a training program for calculating the metrics of dynamic routing protocols and an explanatory note to it. The explanatory note to the work contains 62 pages and 20 illustrations. The aim of the work is to develop a training program for calculating the metrics of dynamic routing protocols. The paper identifies the main tasks for the development of a subsystem that will facilitate the learning process in higher education institutions and also help to consolidate skills and knowledge in the protocols of dynamic routing. The paper analyzes the tools for creating software, considers the existing analogues of the software product, substantiates the relevance of this system and describes the scope of its use. Graphic materials with routing metrics were created and queries were entered for use in the software product. An educational and training program was created to calculate the metrics of dynamic routing protocols and to test this software.

KEY WORDS: PROTECTION, NETWORK, NETWORK PERIMETER, CISCO, ROUTER, PROTOCOL, ROUTE METRICS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

EGP –Exterior Gateway Protocol

RIP – Routing Information Protocol

OSI – Open Systems Interconnection

TCP – Transmission Control Protocol

WAN – Wide Area Network

IS-IS – Intermediate System to Intermediate System

OSPF – Open Shortest Path First

IGP – Interior Gateway Protocol

ДМЗ – демілітаризована зона

NAT – Network Address Translation

ASA – Adaptive Security Appliances

EIGRP – Enhanced Interior Gateway Routing Protocol

UDP – User Datagram Protocol

МЕ – міжмережний екран

IP – Internet Protocol

LAN – Local Area Network

MAC – Media Access Control

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 7
РОЗДІЛ 1. Теоретичний аналіз протоколів маршрутизації 9
1.1. Протокол OSPF 10
1.2. Аналіз аналогів програмного продукту 12
1.3. Класифікація протоколів маршрутизації за типом застосованого алгоритму маршрутизації 15
1.4. Концептуальні основи маршрутизації, метрики маршрутів 17
ВИСНОВОК ДО РОЗДІЛУ 1 20
РОЗДІЛ 2. Проектування навчально-тренувальної програми для розрахунку метрик динамічної маршрутизації 21
2.1. Статичний та динамічний тип маршрутизації 21
2.2. Алгоритм метрик маршрутизації 25
2.3. Вибір та обґрунтування засобів вирішення проблеми 34
ВИСНОВОК ДО РОЗДІЛУ 2 38
РОЗДІЛ 3. Інтерфейс програми та робота з нею 39
3.1. Опис та реалізація завдання 39
3.2. Аналіз отриманих результатів 42
ВИСНОВОК ДО РОЗДІЛУ 3 46
ВИСНОВКИ 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ 48
ДОДАТКИ 50

ВСТУП

Актуальність. Концепція Інтернету Речей все більш і більш визначає розвиток мереж зв'язку як в даний час, так і в перспективі. Найбільш масштабним додатком концепції Інтернету Речей є безпровідні мережі. Внаслідок потенційно широкого застосування безпровідних мереж у всіх областях життєдіяльності людини їх ще називають всепроникні мережі. Безпровідні мережі відносяться до класу систем, що самоорганізуються, для яких в порівнянні з традиційними інфраструктурними мережами істотно видозмінюються принципи побудови, протоколи маршрутизації, параметри якості обслуговування, моделі і характеристики трафіку тощо. При створенні безпровідних мереж широке використання знайшли методи кластеризації, безліч досліджень присвячується проблемам вибору головного вузла кластера. З'являються все нові протоколи маршрутизації, що пов'язано з широким впровадженням безпровідних мереж і безліччю вирішуваних ними різноманітних завдань.

Існує велика кількість телекомунікаційних мереж, для надійної роботи яких використовуються різноманітні протоколи маршрутизації, кожен з яких має свої особливості. Існують додаткові протоколи, що є мостами між окремими мережами, що дозволяють в одній цілісній системі використовувати несумісних між собою протоколи. Зважаючи на всі зусилля не створено протоколу, що б задовольняв багатьом умовам використання, тому вибір оптимального протоколу – це досить складне та актуальне завдання сьогодення.

Постійний розвиток телекомунікаційних мереж вимагає належного рівня підготовки все більшої кількості користувачів, зростає кількість виданих навчальних посібників та наукових праць, проте більшість з них стосується розробниками протоколів маршрутизації або мережевого обладнання [2 – 12]. Тому є необхідність в незалежному аналізі наявних протоколів маршрутизації телекомунікаційних мереж.

Отож головна ціль даної бакалаврської кваліфікаційної роботи є створення навчально-тренувальної програми для розрахунку метрик динамічних

протоколів маршрутизації. Дане програмне рішення буде чудовим інструментом для покращення знать та навичок з комп'ютерних мереж для студентів вищих навчальних закладів.

Метою атестаційної бакалаврської роботи є створення навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації.

Для виконання поставленого завдання необхідно вирішити такі задачі:

- провести аналіз відомих систем для створення тестування, проаналізувати технічні реалізації та алгоритми побудови;
- провести аналіз протоколів маршрутизації та створити алгоритм для побудови навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації, що дало б змогу полегшити навчальний процес у закладах вищої освіти;
- здійснити вибір методів та засобів для реалізації системи;
- спроектувати Інтернет ресурс у вигляді веб додатку.

Об'єктом дослідження є навчально-тренувальна програма для розрахунку метрик динамічних протоколів маршрутизації.

Предмет дослідження –засоби розробки навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації.

Результатом атестаційної бакалаврської роботи є завершена програма для навчання та розрахунків метрик динамічних протоколів маршрутизації.

РОЗДІЛ 1. Теоретичний аналіз протоколів маршрутизації

До телекомунікаційних мереж у наш час відносять : телефонні мережі; радіомережі; телевізійні мережі; комп'ютерні мережі. У всіх видах мереж клієнтам пропонується сучасний "ресурс" – інформація. Телефонні мережі надають інтерактивні послуги (interactive services), так два абонента, що беруть участь в розмові (або декілька абонентів, як що це конференція), по черзі "проявляють активність". Радіомережі та телевізійні мережі надають широко трансляційні послуги (broadcast services), під час трансляції інформація розповсюджується тільки в одному напрямку – від мережі до абонентів, за схемою "один до багатьох" (point-to-multipoint) [1]. Не дивлячись на те, що різноманітності між телефонними, радіо, телевізійними та комп'ютерними мережами, безумовно, суттєві, на достатньо високому рівні абстракції мають подібні структури. На даний час по багатьом напрямкам відбувається конвергенція різноманітних видів телекомунікаційних мереж.

Статична маршрутизація не підходить для великих, складних мереж тому, що зазвичай мережі включають надлишкові зв'язку, багато протоколів і змішані топології. Маршрутизатор у складних мережах повинен швидко адаптуватися до змін топології і вибирати кращий маршрут з багатьох кандидатів, які мережі мають ієрархічну структуру [2]. З точки зору маршрутизації мережа розглядається як сукупність автономних систем. В автономних підсистемах великих мереж для маршрутизації на інші автономні системи широко використовують маршрути за замовчуванням.

Динамічна маршрутизація може бути здійснена з використанням одного і більше протоколів. Ці протоколи часто групуються відповідно до того, де вони використовуються. Протоколи для роботи всередині автономних систем називають внутрішніми протоколами шлюзів (interior gateway protocols (IGP)), а протоколи для роботи між автономними системами називають зовнішніми протоколами шлюзів (exterior gateway protocols (EGP)). До протоколів IGP відносяться RIP, RIP v2, IGRP, EIGRP, OSPF і IS-IS. Протоколи EGP3 і BGP4

відносяться до EGP [3]. Всі ці протоколи можуть бути розділені на два класи: дистанційно-векторні протоколи і протоколи стану зв'язку.

Маршрутизатор використовують метрики для оцінки або вимірювання маршрутів. Коли від маршрутизатора до мережі призначення існує багато маршрутів, і всі вони використовують один протокол маршрутизації, то маршрут з найменшою метрикою розглядається як кращий. EIGRP використовує складну комбінацію чинників, що включає смугу пропускання каналу і його надійність.

Результати роботи маршрутизуючих протоколів заносяться в таблицю маршрутів, яка постійно змінюється при зміні ситуації в мережі.

1.1. Протокол OSPF

Протокол маршрутизації по стану каналів OSPF (Open Shortest Path First) описаний в документі RFC 2328 [4].Протокол OSPF використовує алгоритм SPF і тому може здійснювати більш інтелектуальний вибір маршруту в порівнянні з дистанційно-векторними протоколами маршрутизації. Існує кілька версій протоколу OSPF, в даний час широке поширення набула друга версія протоколу - OSPF v2.

Всі маршрутизатори підтримують OSPF, мережі і підмережі логічно об'єднані в зони. Мережі передачі даних, в яких застосовується протокол OSPF, можуть становити одну зону або включати безліч зон, організованих за ієрархічним ознакою. Об'єднана мережа передачі даних, що використовує протокол OSPF, незалежно від того, чи перебуває вона з однієї зони або включає безліч зон, являє собою один домен маршрутизації, або іншими словами одну автономну систему. Така ієрархічна структура дозволяє локалізувати зміни маршрутів і трафік маршрутних оновлень в межах кожної зони. Відповідно, це зменшує навантаження на канали зв'язку, пов'язані з підтримкою великих таблиць маршрутизації і перерахунком цих таблиць в разі зміни маршрутів.

Протокол OSPF має такі властивості:

- Групова розсилка оновлень. У протоколі OSPF розсилка топологічної інформації про стан каналів зв'язку здійснюється по груповій адресі 224.0.0.5 для всіх маршрутизаторів OSPF і за адресою 224.0.0.6 для призначеного і резервного призначеного маршрутизатора.

- Безкласова маршрутизація. Протоколом OSPF підтримується технологія VLSM.

- Аутентифікація. Маршрутизатор OSPF мають можливість використовувати кілька методів аутентифікації, таких як аутентифікація по паролю або за допомогою MD5 [5].

- Швидкість поширення змін в топології. Завдяки відсутності періодичної розсилки оновлень маршрутної інформації маршрутизатор, який знайшов зміни в топології мережі, негайно сповіщає про це всі сусідні маршрутизатори.

- Економія пропускної здатності каналів зв'язку. Протокол OSPF виробляє періодичну розсилку інформації бази даних топології мережі передачі даних через тривалі проміжки часу, 30 хвилин.

- Ієрархічна поділ мережі передачі даних. Протокол OSPF дозволяє зробити ієрархічний поділ мережі передачі даних на кілька зон, з метою зменшення навантаження на маршрутизатори всередині кожної зони [6].

Всі маршрутизатори OSPF створюють і підтримують у своїй базі даних дві основні таблиці:

- Таблиця сусідства. Всі маршрутизатори OSPF ведуть таблицю сусідства, в якій зберігається список і вся необхідна інформація про сусідніх OSPF маршрутизатори.

- Таблиця топології. Кожен маршрутизатор OSPF веде таблицю топології, яка містить необхідну інформацію про стан усіх мереж, підмереж і маршрутизаторів в межах зони OSPF. Якщо маршрутизатор OSPF має підключення до двох і більше зонах, то він веде окрему таблицю топології для кожної із зон OSPF, до якої він підключений.

1.2. Аналіз аналогів програмного продукту

Для полегшення перевірки знань студентів сьогодні існує велика кількість програмних рішень. Починаючи від тестування на сайті університету, для цього досить часто використовують LMS Moodle або окремі додатки, що дозволяють створювати тести та оцінювати якість знань студентів або ж працівників в тій чи іншій галузі.

Для того, щоб створити якісний програмний продукт необхідно проаналізувати ринок та оцінити аналоги, що вже наявні.

The Class Maker

Цей веб-додаток, який допомагає створювати тести для викладачів, а також працівників підприємств. У даному додатку можна створювати вікторини з обмеженням у час, публічним та приватним тестовим доступом, декількома варіантами, короткими відповідями які можуть включати в себе кілька зображень, файлів, аудіо та відео, вбудованих у запитання та відповіді. Тести можуть включати в себе обрані запитання або ж випадкові запитання з банку питань і бути упорядкованими за категоріями.

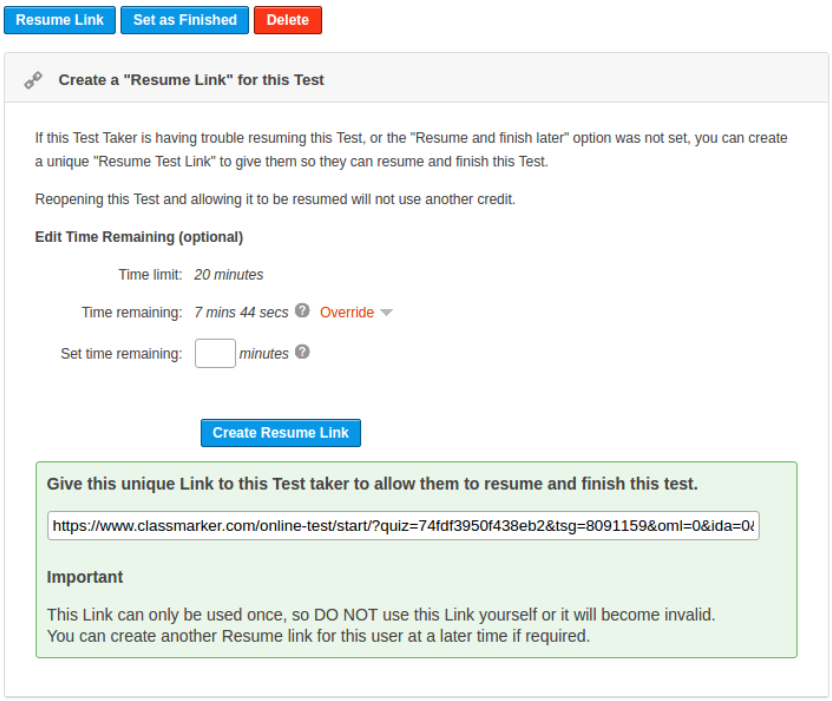


Рис. 1.1. Додаток «The ClassMaker»

Недоліком даного додатку, в першу чергу, є його ціна, так як цільова аудиторія додатку - це викладачі та студенти.

Online Quiz Creator

Дана програма працює як онлайн додаток, створений у вигляді сайту. За допомогою цього додатку також можна створювати вікторини з розширеними стилями та додатковими функціями, а також виводити статистику правильних відповідей та кількості спроб проходження тесту. За допомогою головного алгоритму, програма визначає рівень складності запитання, використовуючи фактичні дані відтворення користувачів у режимі реального часу. Також у програмі є можливість перегляду результатів власного тесту а також запитань інших користувачів.

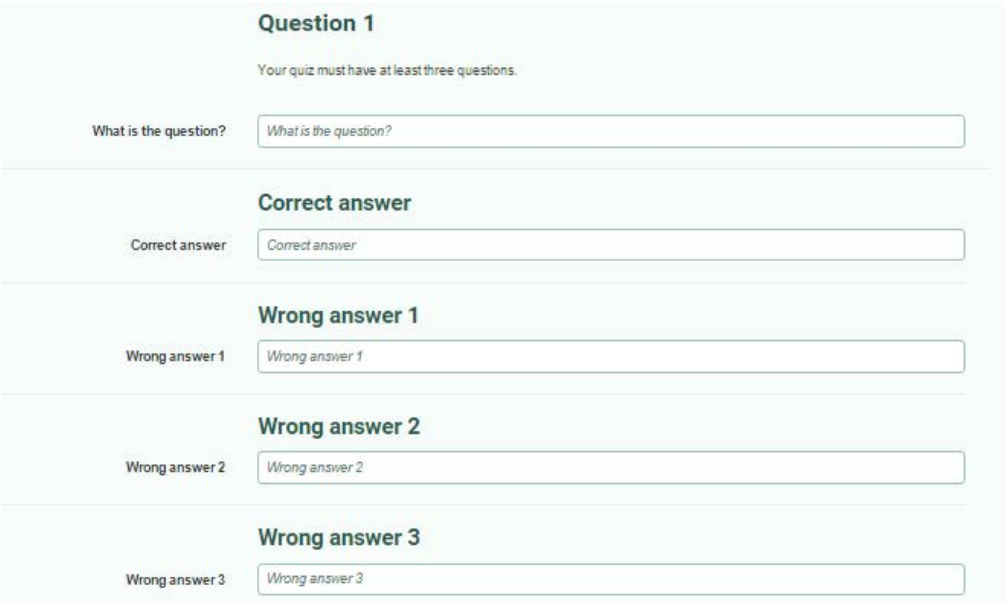


Рис. 1.2. Додаток «Online Quiz Creator»

Не зважаючи на те, що програма надає можливість змінювати колір тексту, фону, в цьому додатку не можна створювати графічні тести (тести з картинками), що не завжди зручно.

Quiz Revolution

Як і всі додатки такого типу, ця програма має простий інтерфейс та дає можливість створювати тести всіх типів. На відміну від попередньої програми Quiz Revolution дозволяє створювати тести з зображеннями та відео. Програма є

безкоштовною, але з наявністю реклами. Перевагами даного додатку є: залучення інтерактивного контенту, підтримка зображень та медіа, автоматичне оцінювання та виставлення оцінки.

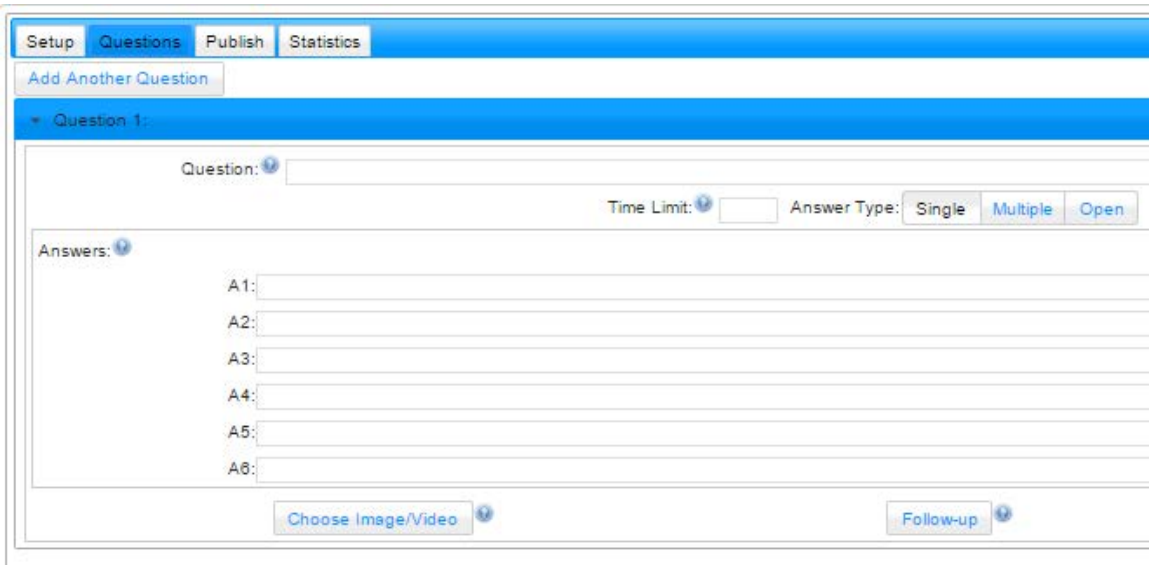


Рис. 1.3. Додаток «Quiz Revolution»

Недоліком даного додатку, в першу чергу, є проблеми зі створенням тестів, які виникають після його запуску. Отже, надійність даної програми викликає сумніви для її використання для серйозних рішень. Також для відключення реклами у додатку необхідно здійснювати оплату.

iSpringQuizMaker

Мобільний додаток, який має гарний інтерфейс та різноманітні варіанти налаштувань. За відмінні результати додаток дає нагороди, що мотивує користувачів виковувати тести якомога краще. В разі неправильних відповідей, додаток демонструє помилки та надає ресурси для їх усунення (статті, книги, відео уроки). Даний додаток створений для покращення процесу дистанційного навчання, що є актуальним на сьогоднішній день.

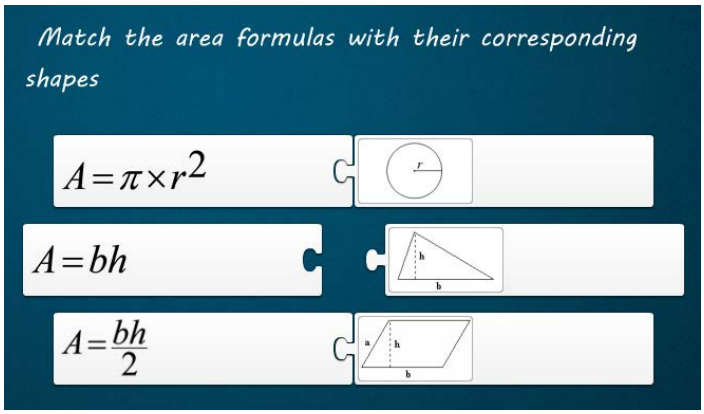


Рис. 1.4. Мобільний додаток iSpring Quiz Maker

Недоліком даного додатку є відносно велика вартість, а також відсутність української та російської локалізації. В додатку відсутній алгоритм випадкових запитань, через що порядок запитань повторюється щоразу після нового запуску тесту.

1.3. Класифікація протоколів маршрутизації за типом застосованого алгоритму маршрутизації

Алгоритми, що використовують статичні маршрути, нескладні для розробки і відмінно працюють в середовищах, де трафік мережі порівняно передбачуваний, а схема мережі є порівняно проста. Так як статичні системи маршрутизації не реагують на зміни в мережі, вони, як правило, вважаються непридатними для сучасних великих мереж, що постійно змінюються, але у свою чергу можуть легко використовуватися у:

- статичних системах маршрутизації, які не змінюються в часі;
- статичних системах маршрутизації, що змінюються за розкладом;
- статичних системах маршрутизації, що змінюються по ситуації – адміністративно в момент виникнення стандартної ситуації [7].

Більшість домінуючих алгоритмів маршрутизації з 1990рр. – динамічні. Динамічні алгоритми маршрутизації підлаштовуються до обставин, що змінюються мережі в масштабі реального часу. Вони це виконують після аналізу

вхідних повідомлень про оновлення маршрутизації. Якщо в повідомленні вказується, що мала місце зміна мережі, програми маршрутизації перераховують маршрути і розсилають нові повідомлення про коригування маршрутизації. Такі повідомлення пронизують мережу, стимулюючи роутери заново проганяти свої алгоритми і відповідним чином змінювати таблиці маршрутизації. Динамічні алгоритми маршрутизації можуть доповнювати статичні маршрути там, де це доречно. Наприклад, можна розробити “роутер останнього звернення” (тобто роутер, в який надсилаються всі невідправлені за певним маршрутом пакети) [8]. Такий роутер виконує роль сховища невідправлених пакетів, гарантуючи, що всі повідомлення будуть хоча б певним чином оброблені. Деякі складні протоколи маршрутизації забезпечують безліч маршрутів до одного й того ж пункту призначення.

Такі багатомаршрутні алгоритми роблять можливою мультиплексну передачу трафіку по численних лініях; одномаршрутні алгоритми не можуть робити цього. Переваги багатомаршрутних алгоритмів очевидні – вони можуть забезпечити значно більшу пропускну спроможність і надійність процесу передачі трафіку. Проте, багаторівневі алгоритми маршрутизації набагато складніше реалізувати.

Деякі алгоритми маршрутизації оперують в плоскому просторі, в той час як інші використовують ієрархії маршрутизації. У однорівневої системи маршрутизації всі роутери рівні по відношенню один до одного. В ієрархічній системі маршрутизації роутери формують те, що становить основу (backbone - базу) маршрутизації. Пакети з небазових роутерів переміщуються до роутера та пропускаються через них до тих пір, поки не досягнуть спільної області пункту призначення. Починаючи з цього моменту, вони переміщаються від останнього базового роутера через один або кілька небазових роутерів до кінцевого пункту призначення. Системи маршрутизації часто встановлюють логічні групи вузлів, званих доменами, або автономними системами (AS), або областями. В ієрархічних системах одні роутери будь-якого домену можуть єднатися з роутерами інших доменів, в той час як інші роутери цього домену можуть

підтримувати зв'язок з роутери тільки в межах свого домену. У дуже великих мережах можуть існувати додаткові ієрархічні рівні. Роутери найвищого ієрархічного рівня утворюють базу маршрутизації [9].

Основною перевагою ієрархічної маршрутизації є те, що вона імітує організацію більшості компаній а отже, дуже добре підтримує їх схеми трафіку. Велика частина мережевого зв'язку має місце в межах груп невеликих компаній (доменів). Внутрішньодоменним роутерам необхідно знати тільки про інші роутери в межах свого домену, тому їх алгоритми маршрутизації можуть бути спрощеними. Відповідно може бути зменшений і трафік поновлення маршрутизації, що залежить від використаного алгоритму маршрутизації.

1.4. Концептуальні основи маршрутизації, метрики маршрутів

Алгоритми роботи маршрутизаторів можуть бути як статичними, так і динамічними. При статичній маршрутизації конфігурація проводитися вручну. При динамічній маршрутизації, обміном інформації управляють протоколи маршрутизації, завдяки їм маршрутизатори можуть відстежувати топологію мережі і коригувати маршрути. І статична, і динамічна конфігурації, а також їх комбінування переслідують одну і ту ж саму мету - забезпечити обмін інформацією між віддаленими вузлами.

Всі маршрутизатори повинні мати локальні таблиці маршрутизації. Вони використовуються маршрутизатором при передачі інформації для визначення найкращого шляху від джерела до пункту призначення.

В процесі маршрутизації проводитися вибір оптимального маршруту до мереж одержувачам. Так як одночасно на маршрутизаторі може бути запущено відразу декілька протоколів маршрутизації, необхідний метод вибору між маршрутами, отриманими від різних протоколів маршрутизації. У маршрутизаторах для вибору маршрутів отриманих від різних протоколів маршрутизації використовується концепція адміністративного відстані [10].

Адміністративна відстань розглядається як міра достовірності джерела інформації про маршрут. Це має сенс тоді, коли маршрутизатор має інформацію про маршрут до мережі одержувача від декількох протоколів маршрутизації.

Малі значення величини адміністративної відстані краще великих значень. Стандартні значення адміністративної відстані встановлюються такими, щоб значення, що вводяться вручну, були краще, значень отриманих автоматично, і протоколи маршрутизації з більш складними метриками переважали над протоколами маршрутизації, що мають прості метрики.

Визначення того, який власне маршрут є найкращим шляхом до мережі одержувачу, є особливістю властивою для будь-якого протоколу. Кожен протокол має свою міру того, що є кращим. Маршрутизатори характеризують маршрут до мережі за допомогою метрики маршруту.

Процес маршрутизації обирає маршрут, що володіє найменшим значенням метрики.

Метрики можуть бути обчислені на основі однієї або кількох характеристик. Найбільш часто в алгоритмах маршрутизації використовуються наступні параметри:

- Ширина смуги пропускання є засіб оцінки обсягу інформації, який може бути переданий по каналу зв'язку;

- Затримка – проміжок часу, необхідний для переміщення пакета по кожному з каналів зв'язку від відправника до одержувача. Затримка залежить від пропускної здатності проміжних каналів, розміру черг в портах маршрутизаторів, завантаження мережі і фізичного відстані;

- Завантаження – середня завантаженість каналу зв'язку в одиницю часу;

- Надійність – відносна кількість помилок на каналі зв'язку;

- Кількість переходів – кількість маршрутизаторів, які необхідно подолати пакету, перш ніж він досягне пункту призначення;

- Вартість – значення, зазвичай обчислюється на основі пропускної здатності, грошової вартості або інших одиниць виміру, що призначаються адміністратором [11].

До пункту призначення може існувати безліч шляхів, і всі вони можуть відображатися в таблиці маршрутизації. Якщо існує більш ніж один шлях до вузла одержувачу, протокол маршрутизації повинен вибрати один шлях як найкращий і помістити його в таблицю маршрутизації. Однак багато протоколів маршрутизації підтримують механізм балансування навантаження, при якому в таблицю маршрутизації можуть бути записані декілька можливих маршрутів до вузла одержувачу, і передача трафіку буде здійснюватися по кожному з маршрутів.

Однією з основних завдань маршрутизаторів є побудова таблиці маршрутизації на основі даних отриманих від протоколів маршрутизації і настройках введених вручну.

Вибір маршруту для занесення в таблицю маршрутизації повинен ґрунтуватися на таких критеріях:

- Доступність IP адреси переходу. Процес маршрутизації полягає в послідовній передачі трафіку від відправника до одержувача. Маршрутизатор повинен знати IP адресу наступного маршрутизатора в ланцюжку передачі трафіку.

- Метрика маршруту. Якщо перехід можливий, то протокол маршрутизації обирає найкращий можливий маршрут передачі. Критерієм вибору маршруту є мінімальна метрика маршруту.

- Префікс. Маршрутизатор розглядає довжину префіксу (маска підмережі), якщо є кілька маршрутів до мережі одержувача, але з різними префіксами, то в таблицю маршрутизації заносяться всі маршрути.

- Адміністративне відстань маршруту. Якщо маршрутизатор має більше одного маршруту до одержувача, критерієм вибору для занесення в таблицю маршрутизації є мінімальне адміністративне відстань.

Після створення таблиці маршрутизації, необхідно підтримувати її точну відповідність щодо топології мережі. Підтримка таблиць маршрутизації здійснюється або адміністратором мережі вручну, або за допомогою динамічних протоколів маршрутизації. Незалежно від того, конфігуруються чи маршрути

вручну або за допомогою протоколів маршрутизації, точність відображення маршрутів є ключовим фактором в здатності маршрутизатора забезпечувати пересилання даних її одержувачам [12].

Існує кілька методів маршрутизації, для побудови і підтримки таблиці в актуальному стані. У загальному випадку при побудові таблиці маршрутизації маршрутизатор застосовує комбінацію наступних методів:

- Пряме з'єднання;
- Статична маршрутизація;
- Маршрутизація за умовчанням;
- Динамічна маршрутизація.

І хоча кожен з цих методів має свої переваги і недоліки, вони не є взаємовиключними.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі було розглянуто та проаналізовано літературні джерела. Розглянуто статичну та динамічну маршрутизації на основі протоколів RIP, RIP v2, IGRP, EIGRP, OSPF і IS-IS. Було розглянуто протокол OSPF та детально описано його функції (Групова розсилка оновлень, автентифікація, швидкість поширення змін, ієрархічний поділ мережі передачі даних та інші).Розглянуто класифікацію протоколів маршрутизації. Також, було проведено дослідження та аналіз аналогів програмного продукту. Було розглянуто концептуальні основи маршрутизації, метрики маршрутів.

РОЗДІЛ 2. Проектування навчально-тренувальної програми для розрахунку метрик динамічної маршрутизації

2.1. Статичний та динамічний тип маршрутизації

Статичні маршрути - це такі маршрути до мереж одержувачам, які адміністратор мережі вручну вносить в таблицю маршрутизації. Статичний маршрут визначає IP адреса наступного сусіднього маршрутизатора або локальний вихідний інтерфейс, який використовується для направлення трафіку до певної мережі одержувачу.

Статичний маршрут не може бути автоматично адаптований до змін в топології мережі. Якщо певний маршрутизатор або ж інтерфейс стають недоступними, то маршрут до мережі одержувача буде також недоступним [13].

Перевагою цього способу маршрутизації є те, що він виключає весь службовий трафік, пов'язаний з підтримкою і коригуванням маршрутів. Статична маршрутизація може бути використана в наступних ситуаціях:

- Коли адміністратор потребує повний контроль маршрутів використовуваних маршрутизатором;
- Коли необхідно резервування динамічних маршрутів;
- Коли є мережі досяжні єдино можливим шляхом;
- Коли небажано мати службовий трафік необхідний для оновлення таблиць маршрутизації, наприклад при використанні комутованих каналів зв'язку.
- Коли використовуються застарілі маршрутизатори не мають необхідного рівня обчислювальних можливостей для підтримки динамічних протоколів маршрутизації.

Найкращою топологією для використання статичної маршрутизації є топологія «зірка». При даній топології маршрутизатори, підключені до центральної точки мережі, мають тільки один маршрут для всього трафіку, який буде проходити через центральний вузол мережі. І один або два маршрутизатора в центральній частині мережі мають статичні маршрути до всіх віддалених вузлів.

Однак з часом така мережа може вирости до десятків і сотень маршрутизаторів з будь-якою кількістю підключених до них підмереж. Кількість статичних маршрутів в таблицях маршрутизації буде збільшуватися пропорційно збільшенню кількості маршрутизаторів в мережі. Кожен раз при додаванні нової підмережі або маршрутизатора, адміністратор повинен буде додавати нові маршрути в таблиці маршрутизації на всіх необхідних маршрутизаторах. При такому підході може наступити момент, коли більшу частину свого робочого часу адміністратор буде займатися підтримкою таблиць маршрутизації в мережі. У цьому випадку необхідно зробити вибір в сторону використання динамічних протоколів маршрутизації [14].

Інший недолік статичної маршрутизації проявляється при зміні топології корпоративної мережі. При цьому адміністратор повинен вручну вносити всі зміни в таблиці маршрутизації маршрутизаторів, на які вплинули зміни в топології мережі.

Протоколи динамічної маршрутизації можуть автоматично відстежувати зміни в топології мережі.

При використанні протоколів динамічної маршрутизації, адміністратор мережі конфігурує обраний протокол на кожному маршрутизаторі в мережі. Після цього маршрутизатори починають обмін інформацією про відомі їм мережі та стани. Причому маршрутизатори обмінюються інформацією тільки з тими маршрутизаторами, де запущений той же протокол динамічної маршрутизації. Коли відбувається зміна топології мережі, інформація про ці зміни автоматично поширюється по всьому маршрутизаторів, і кожен маршрутизатор вносить необхідні зміни в свою таблицю маршрутизації.

Показана на рис. 2.1 мережа по-різному адаптується до змін топології, в залежності від того, який тип маршрутизації використовується: динамічна або статична.

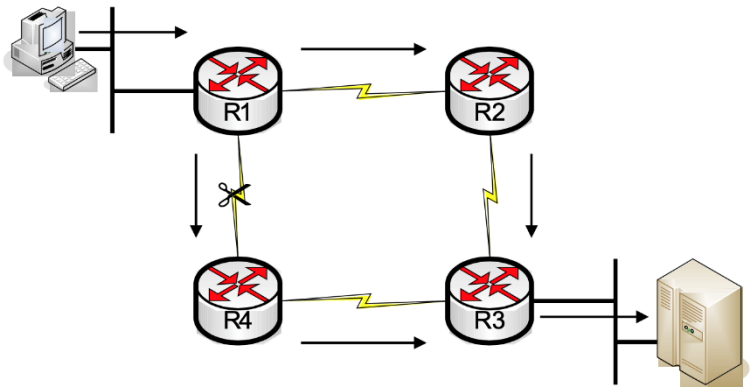


Рис 2.1. Динамічний маршрут

Статична маршрутизація дозволяє переслати пакет з однієї мережі в іншу на основі вручну заданих маршрутів. В даному прикладі маршрутизатор R1 завжди пересилає потоки даних, призначені маршрутизатора R3, через маршрутизатор R4. Маршрутизатор звертається до своєї таблиці маршрутизації і відповідно розташованої в ній інформацією про статичному маршруті направляє пакет на вузол одержувач.

Якщо маршрут від маршрутизатора R1 до маршрутизатора R4 по якійсь причині ставати недоступним, то маршрутизатор R1 не може передавати пакет маршрутизатора R4 по ньому. Відповідно, до повторного ручного конфігурування маршрутизатора R1 на передачу пакетів через маршрутизатор R2 зв'язок з мережею одержувачем буде неможлива [15].

Динамічна маршрутизація забезпечує більшу гнучкість. Відповідно до таблиці маршрутизації, створеної на маршрутизаторі R1, пакет може бути доставлений до пункту призначення по більш кращого маршруту через маршрутизатор R4. Однак при цьому залишається доступним і другий шлях до пункту призначення - через маршрутизатор R2. Коли маршрутизатор R1 дізнається про те, що канал до маршрутизатора R4 вийшов з ладу, він відновить свою таблицю маршрутизації, роблячи маршрут через маршрутизатор R2 кращим маршрутом до пункту призначення. В цьому випадку маршрутизатори продовжують пересилання пакетів з резервного каналу.

Після того, як маршрут між маршрутизаторами R1 і R4 відновиться, маршрутизатор R1 знову оновлює свою таблицю маршрутизації, віддаючи перевагу основному маршруту через маршрутизатор R4.

Протоколи динамічної маршрутизації можуть також для підвищення ефективності роботи мережі застосовувати механізм балансування навантаження по декількох маршрутах.

Одним з поширених протоколів на основі стану каналу є протокол OSPF. Це безкласовий протокол маршрутизації. Технологія роботи протоколу полягає у відстеженні стану каналів і пошуку найкоротших шляхів (Shortest Path First - SPF), використовуючи алгоритм Дейкстри. Протокол підтримує складну топологічну базу даних. Якщо протоколи на базі вектору відстаней не містять інформацію про віддалені мережі, то протоколи на основі стану каналу підтримують всю інформацію про віддалені маршрутизатори та їх з'єднаннях [16].

Стан каналу в цьому протоколі має на увазі опис інтерфейсу (наприклад, IP-адреса, маска, тип мережі і т.п.) і його відношення з сусідніми маршрутизаторами. На основі вище зазначених описів інтерфейсів формується база даних стану каналів. База даних заповнюється завдяки отриманню повідомлень про стан каналу (Link-State Advertisement - LSA), які розсилаються регулярно або ж відразу після зміни топології мережі, або при будь-яких змінах на маршрутизаторах. Ці повідомлення є невеликі пакети. В LSA міститься інформація про підключених інтерфейсах, метриках і інших параметрах.

На основі отриманих повідомлень LSA маршрутизатор використовує алгоритм SPF, який будує дерево найкоротших маршрутів. Алгоритм проводить розрахунок над базою даних топології мережі, видаляючи зайві гілки (Гілки - всі можливі шляхи). Отримані маршрути заносяться в таблицю маршрутизації.

Протокол OSPF - внутрішній протокол маршрутизації і працює всередині однієї автономної системи. Її можна розбити на зони (області), які представляють собою логічні розділи автономної системи. Розглянемо дворівневу мережеву ієрархію, в якій у всіх маршрутизаторів свій принцип роботи.

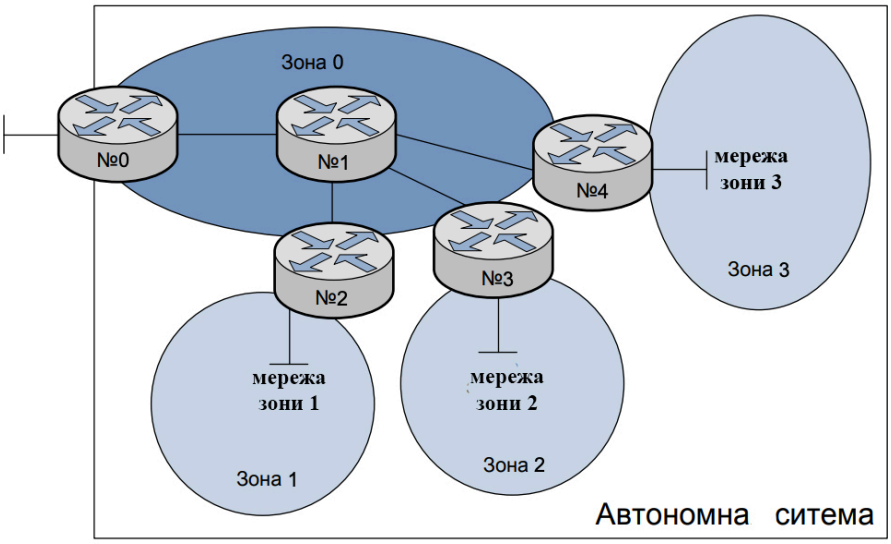


Рис 2.2. Дворівнева мережева ієрархія

На рис. 2.2 представлена одна автономна система, яка має один прикордонний маршрутизатор (марш. №0), служба для зв'язку із зовнішньою мережею (наприклад, з іншого автономною системою).

Автономна область включає в себе:

- зону 0 - область, що відповідає за з'єднання всіх областей, іноді її називають як магістральна область. Маршрутизатор 1 в цій області називають магістральним;
- зони 1-3 - маршрутизатори в цих областях мають назви не магістральні. Це маршрутизатори, які знають топологію області, в яких вони знаходяться, і ведуть базу даних станів каналів тільки своїх областей [17];
- на стику зони 0 і зон 1-3 знаходяться прикордонні маршрутизатори №2-4. Ці маршрутизатори ведуть базу даних станів каналів всіх областей, до яких вони підключені;
- маршрутизатор №0 так само є прикордонним, але вже для автономної системи.

2.2. Алгоритм метрик маршрутизації

Протоколи маршрутизації вибираються, виходячи з характеристик, перерахованих нижче.

Оптимальність визначає можливості протоколу та алгоритму щодо вибору найбільш оптимального маршруту на підставі метрик і їх вагових значень, що

використовуються при розрахунках. Наприклад, якийсь протокол може використовувати лічильник вузлів і затримки для визначення метрик; затримки мають більш високий вагу при обліку остаточного значення, але зате їх складніше розрахувати.

Простота і низькі накладні витрати. Ідеальна ефективність роботи алгоритму маршрутизації може бути досягнута, коли завантаження процесора і пам'яті маршрутизатора мінімальні. Ця характеристика важлива для масштабованості мережі, яка в граничному випадку може бути розширена до розмірів мережі Internet.

Стійкість і надійність. Алгоритм маршрутизації повинен коректно функціонувати навіть при наявності нестандартних і непередбачених обставин, таких, як збій обладнання, високе завантаження і помилки експлуатації.

Швидка конвергенція. Конвергенцією називається процес встановлення домовленості між усіма маршрутизаторами про наявні маршрути. Алгоритми маршрутизації, що не володіють швидкою конвергенцією, можуть викликати збій або значну затримку при доставці інформації.

Гнучкість. Алгоритм і протокол маршрутизації повинні швидко адаптуватися до різноманітних змін у мережі. Змінами в мережі вважаються зміни в стані пристроїв, зокрема, маршрутизаторів, зміна пропускної здатності каналів, зміна розміру черг або мережевий затримки.

Масштабованість. Деякі протоколи розроблені таким чином, що можуть бути масштабуються краще за інших. Важливо пам'ятати, що якщо планується розширення мережі (або така можливість в принципі передбачається), слід віддати перевагу протоколу EIGRP, ніж RIP. Першочергове завдання алгоритму маршрутизації при оновленні таблиці маршрутизації полягає у визначенні кращої інформації, яка повинна бути внесена в таблицю. Алгоритми маршрутизації використовують різні метрики для визначення найкращого маршруту, але кожен алгоритм інтерпретує вибір кращого варіанту шляху по-своєму. Алгоритм маршрутизації розраховує число, зване метрикою, для кожного мережевого маршруту. Складні алгоритми маршрутизації можуть

засновувати вибір маршруту на основі декількох параметрів, об'єднуючи їх в одну загальну метрику. Чим менше метрика, тим краще обраний маршрут. Метрики можуть бути обчислені на основі однієї або кількох характеристик. Найбільш часто в алгоритмах маршрутизації використовуються параметри метрики, які перераховані нижче.

Ширина смуги пропускання є засіб оцінки обсягу інформації, який може бути переданий по каналу зв'язку (канал Ethernet зі швидкістю 10 Мбіт / с більш кращий, ніж виділена лінія зі швидкістю 64 Кбіт / с).

Затримка - проміжок часу, необхідний для переміщення пакета по кожному з каналів зв'язку від відправника одержувачу. Затримка залежить від пропускної здатності проміжних каналів, розміру черг в портах маршрутизаторів, завантаження мережі і фізичного відстані [18].

Завантаження - обсяг операцій, виконуваних мережевим пристроєм, таким, як маршрутизатор, або середня завантаженість каналу зв'язку.

Надійність зазвичай позначає відносне значення кількості помилок для кожного з каналів зв'язку.

Лічильник транзитних вузлів - кількість маршрутизаторів, через які повинен пройти пакет, перш ніж досягне пункту призначення. Коли пакет проходить через маршрутизатор, значення лічильника вузлів збільшується на одиницю. Шлях, для якого значення лічильника вузлів дорівнює чотирьом, означає, що дані, відправлені по цьому маршруту, пройдуть через чотири маршрутизатора, перш ніж будуть отримані адресатом. Якщо існує кілька шляхів, маршрутизатор вибирає той, для якого значення лічильника вузлів найменше.

Вартість - значення, зазвичай обчислюється на основі пропускної здатності, грошової вартості або інших одиниць виміру, що призначаються адміністратором.

Протокол RIP

Протокол маршрутної інформації (Routing Informational Protocol - RIP) був спочатку визначений у документі RFC 1058. Найбільш істотні наступні його характеристики:

- RIP є дистанційно-векторного протоколу маршрутизації;
- Як метрики при виборі маршруту використовується кількість переходів;
- Максимальна довжина маршруту дорівнює 15 переходам;
- За замовчуванням поновлення маршрутної інформації розсилаються широкомовною способом.

При роботі протоколу RIP використовується транспортний протокол UDP. Всі пристрої, що підтримують RIP, прослуховують UDP порт 520 і здійснюють передачу через цей же порт. У мережах загального доступу, таких як Ethernet, ці широкомовні дейтаграми отримують всі пристрої широкомовного домену [19].

Протокол RIP використовує відстань як єдину метрику для визначення найкращого маршруту, тобто чим коротше маршрут, тим він кращий. Якщо до пункту призначення існує безліч маршрутів, то маршрутизатори підтримують протокол RIP, вибирають з них найкоротший і записують його в таблицю маршрутизації.

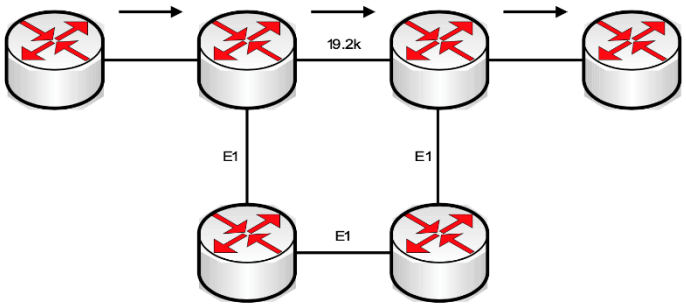


Рис. 2.3. Метрика маршруту в протоколі RIP

Варто звернути увагу на те, що на рис. 2.3. маршрут з пропускною здатністю рівною 19,2Кбіт/с включає в себе три переходи. Нижній альтернативний маршрут по каналах зв'язку E1 та включає п'ять переходів. Оскільки, вибір маршруту в протоколі RIP ґрунтується виключно на кількості

переходів, то в даному випадку в таблицю маршрутизації буде записаний маршрут з пропускною спроможністю 19,2 Кбіт/с замість набагато більш швидких каналів E1.

Протокол RIP запобігає появі петель в маршрутизації, встановлюючи максимальну кількість переходів на маршруті від відправника до одержувача. Стандартне максимальне значення кількості переходів дорівнює 15. При отриманні маршрутизатором поновлення маршрутної інформації, що містить нову або змінену запис, він збільшує значення метрики на одиницю. Якщо при цьому значення метрики перевищує 15, то метрика вважається нескінченно великий, а маршрут до мережі одержувача недосяжним. Крім цього щоб підвищити ефективність роботи протокол RIP використовує механізми розщеплення горизонту і таймери утримання інформації.

Налаштування протоколу RIP

RIP-поширений протокол на основі вектору відстані, підтримуваний більшістю маршрутизаторів. Він найбільш доцільний для невеликих мереж з декількома маршрутизаторами. Перед налаштуванням RIP необхідно привласнити IP-адреса і активувати всі фізичні інтерфейси, які будуть брати участь в маршрутизації. На послідовних каналах необхідно встановити тактову частоту головного маршрутизатора [20]. Потім можна перейти до налаштування RIP.

Налаштування RIP в самому базовому вигляді потребує знання лише трьох команд:

- Router(config)#router rip;
- Router(config-router)#version 2;
- Router(config-router)#network [номер мережі].

Остання команда вводить адреси всіх безпосередньо підключених мереж, яким потрібно повідомлення протоколом RIP. При виконанні налаштування протоколу RIP можна скористатися візуальними засобами PacketTracer. Для введення таблиці маршрутизації необхідно на вкладці Config вибрати ROUTING-

RIP. На рис. 2.4. показаний приклад введення адрес всіх безпосередньо підключених мереж для маршрутизатора Router1.

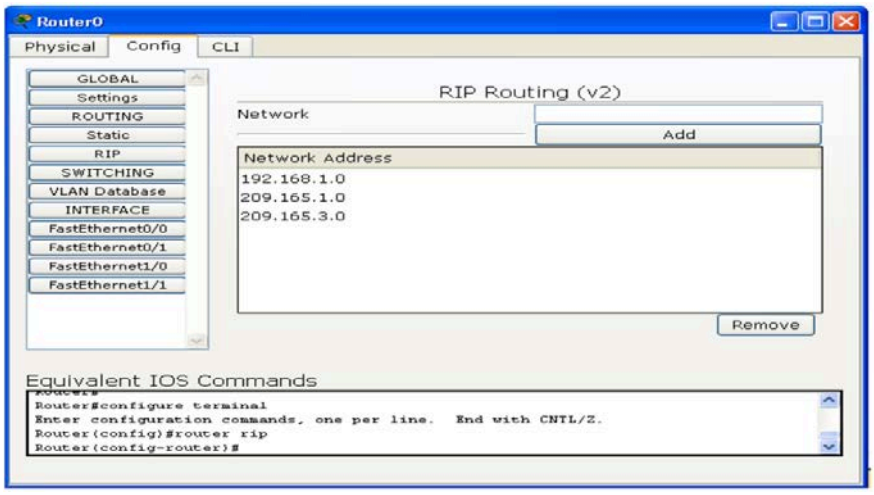


Рис. 2.4. Введення адрес всіх підключених на пряму мереж дляRouter1

Якщо конфігурація правильна, то для перевірки працездатності маршрутизації можна відправити ехо-запити командою ping на пристрої у віддалених мережах. Успішне виконання команди ping буде свідченням працездатності маршрутизації.

Протокол EIGRP

Протокол EIGRP (Enhanced Interior Getaway Routing Protocol) - розширений протокол маршрутизації внутрішнього шлюзу є розширеною версією протоколу IGRP. Він є фірмовим протоколом корпорації Cisco представленим в 1994 році, проте в даний час ряд сторонніх виробників ліцензував EIGRP для реалізації в своєму телекомунікаційному обладнанні.

Протокол EIGRP бере свій початок від дистанційно-векторної маршрутизації. Як і його попередник, протокол IGRP, EIGRP простий в налаштуванні і знайшов застосування в великому спектрі мережевих топологій [21].

За способом подання маршрутної інформації протокол EIGRP все ж є дистанційно-векторного протоколу. Мережі одержувачі в маршрутних оновлення супроводжуються метриками і адресами «наступних» маршрутизаторів. Подібно до інших дистанційно-векторного протоколам,

протокол EIGRP не володіє повними знаннями про топології мережі передачі даних.

Однак в основі протоколу EIGRP лежить не алгоритм Беллмана-Форда, а алгоритм DUAL (Diffusing Update Algorithm), алгоритм дифузійного поновлення. Сам алгоритм веде свій початок від роботи Едсгер Дейкстри і К. С. Шольтен (Edsger Dijkstra, C. S. Sholten) «Termination Detection for Diffus-ing Computations» виданої в 1980 році.

Алгоритм DUAL підкріплений багатьма математичними роботами, основною з яких є «Loop-Free Routing Using Diffusing Computations», написана Х. Х. Гарсія-Луна-Асевесом (J. J. Garcia-Luna-Aceves) .

Алгоритм DUAL лежить в основі протоколу EIGRP. Тому розглянемо його першим. Почнемо з класичного алгоритму DUAL. А потім розглянемо необхідні зміни для адаптації алгоритму для протоколу маршрутизації EIGRP [22].
Протоколи маршрутизації на основі дистанційно-векторного алгоритму Беллмана-Форда, мають дві фундаментальні проблеми:

- Маршрутні петлі;

- Рахунок до нескінченності.

Обидві ці проблеми обумовлені тим, що маршрутизатори нічого не знають про дійсну топології мережі передачі даних. Все що вони знають, це якісь мережі одержувачі доступні через кожного з сусідів. Коли в мережі відбуваються зміни, маршрутизатори намагаються перемикати маршрути на підставі нових метрик, які їм стають відомі.

До досягнення збіжності мережі, метрики можуть неточно відображати дійсну вартість маршрутів, що може привести до маршрутним петель [23].

Основним методом боротьби з виникненням маршрутних петель є введення метрики рівній нескінченності, при досягненні якої маршрут вважається недоступним. Зазвичай маршрутизаторів необхідно виконати певну кількість ітерацій, перед тим як встановити метрику в нескінченність, отже, час збіжності мережі з використанням дистанційно-векторних алгоритмів значно збільшується. Щоб прискорити сходження мережі, доводиться застосовувати

додаткові методи: правило розщеплення горизонту, метод зворотного поновлення, миттєві оновлення і таймери утримання інформації.

Згадаймо, яку інформацію маршрутизатори отримують один від одного і як вони її використовують. Маршрутизатор в разі використання алгоритму Беллмана-Форда, обмінюються маршрутними оновленнями, що містять відомі їм мережі одержувачі. Кожна мережа одержувач, супроводжується метрикою, що відбиває, наскільки далеко маршрутизатор, що відправляє це маршрутне оновлення, на його думку, перебувати від даної мережі.

Маршрутизатор, який одержує маршрутне оновлення, використовує метрику з цього оновлення для розрахунку своєї власної метрики. У більшості випадків нова метрика включає в себе вартість інтерфейсу, через який було отримано маршрутне оновлення. Якщо отримана метрика є найменшою для даної мережі одержувача, маршрутизатор створює маршрут до цієї мережі. Маршрут вказує на маршрутизатор, який відправив маршрутне оновлення, і містить тільки що розраховану метрику. Ця метрика потім використовується маршрутизатором в своїх власних маршрутних оновленнях про цю мережі одержувача [24].

У такій ситуації маршрутизатор, який одержує маршрутне оновлення, відкидає фрагмент дуже важливої інформації - метрику маршруту яку він отримав від свого сусіда. Маршрутизатор використовує її для рас подружжя своєї метрики, а потім ця метрика забувається.

Виникає питання: що такого цінного в заявленому відстані? На рисунку 2.4 зображена топологія мережі, з маршрутизатором R1 в центрі. У маршрутизатора R1 є маршрут до деякої мережі одержувачу (МО) через маршрутизатор R2.

Стрілки з числами вказують, з якою метрикою маршрутизатор, з якого вказує стрілка, оголошує маршрут до МО на відповідному каналі зв'язку. Числа в колах вказують вартість каналів зв'язку. Маршрутизатор обчислюють свої метрики складанням метрики з маршрутного поновлення з вартістю каналу зв'язку, через який оновлення було отримано.

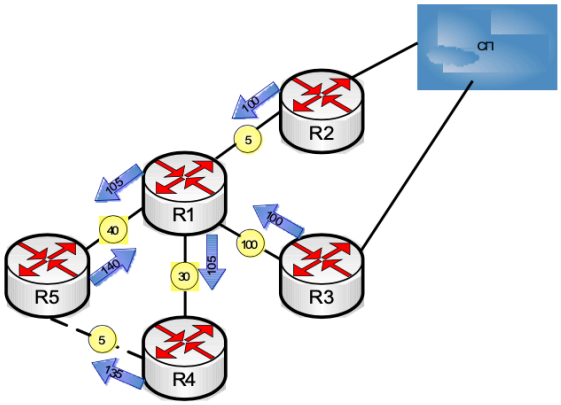


Рис. 2.5. Обробка топології мережі алгоритмом Беллмана-Форда

Маршрутизатор R1 спочатку вибрав маршрутизатор R2 в якості наступного маршрутизатора до МО, оскільки оголошена метрика маршрутизатора R2 (100) в сумі з вартістю каналу зв'язку до нього (5) дала найменше значення (100 + 5 = 105).

Розташування маршрутизаторів R4 і R5 таке, що незалежно від того, який шлях вони оберуть для передачі трафіку, спрямованого до МО, цей шлях пролягає через маршрутизатор R1. Наприклад, маршрутизатор R5 може відправити трафік безпосередньо на маршрутизатор R1 через канал зв'язку між двома цими маршрутизаторами. Або ж він може вибрати альтернативний маршрут, показаний на рисунку 2.5 в вигляді пунктирною лінії. Цей шлях спочатку призводить трафік на маршрутизатор R4, який потім передає його на маршрутизатор R1 [24].

Як показано на малюнку метрика шляху за альтернативним маршрутом менше, ніж метрика прямого маршруту через маршрутизатор R1, отже, маршрутизатор R5 вибирає альтернативний маршрут. Незважаючи на те, що оголошується маршрутизатором R1 метрика (105) менше, маршрутизатор R5 вибирає маршрут через іншого сусіда, оскільки оголошується сусідом метрика (135) в сумі з вартістю сегмента, дає менше значення (135 + 5 = 140), ніж метрика маршруту через маршрутизатор R1 (105 + 40 = 145) [25].

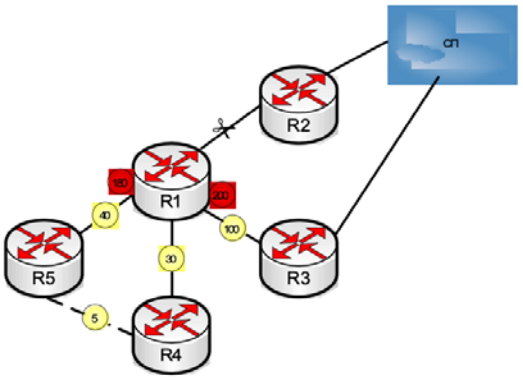


Рис 2.6. Пошук альтернативного маршруту алгоритмом Беллмана-Форда

На рисунку 2.6 перед маршрутизатором R1 стоїть дилема: два маршрутизатора які недавно оголошували MO, це маршрутизатори R3 і R5. Ті менш, незважаючи на меншу результуючу метрику маршруту до MO через маршрутизатор R5, маршрутизатор не може покластися на цю метрику, оскільки вона може бути застарілою. Тому маршрутизатор R1 повинен використовувати нормальні процедури досягнення стабільності, такі, як таймери утримання інформації, миттєві оновлення, а також інші описані раніше методи, щоб переконатися, що він не має справу із застарілою інформацією про метриці [26].

2.3. Вибір та обґрунтування засобів вирішення проблеми

Після проведення аналізу відомих систем, для вирішення проблеми було вирішено створити веб-додаток, який би допомагав у процесі навчання щодо динамічних протоколів маршрутизації.

Для реалізації поставленої задачі, потрібно обрати гнучкі засоби розробки, доступний хостинг та інтерфейс для адміністрування баз даних. В якості операційної системи було обрано Microsoft Windows, у зв’язку з розповсюдженістю даної ОС та великою кількістю доступних додатків, також перевагою є стабільність та практичність в роботі.

Розроблювана система повинна бути відображеною у браузері, який підтримується даною операційною системою. Оскільки для написання інтерфейсу веб-додатків, який буде відображатися у браузері, потрібно обрати

гіпертекстову мову розмітки, то найбільш популярним і вдалим рішенням є нова версія HTML та CSS: HTML5, CSS3.

Веб-програмування – це галузь веб-розробки та різновид дизайну, завданням яких є проектування користувальницьких веб-інтерфейсів для сайтів чи веб-додатків. В ході веб-дизайну здійснюється проектування логічної структури веб-сторінок, пошук більш зручних рішень подачі інформації, також відбувається оформленням веб-проекту.

Мова HTML (HyperText Markup Language) – це мова розмітки гіпертексту, яка призначена для написання гіпертекстових документів, що публікуються в World Wide Web. Щоб створити веб-документ, потрібно дотримуватися правил, за якими працює браузер. Для вірного відображення на екрані відповідної веб-сторінки, він має подаватись у форматі HTML. Документ у форматі HTML є текстом у форматі ASCII, що містить спеціальні помітки для потрібного відображення різних елементів браузером. Важливо виділити основні переваги нової версії – HTML5: створено нові елементи й атрибути, за допомогою яких стало значно легше просувати сайти; розроблено теги, що дозволять розбивати сторінки сайту на інтуїтивно зрозумілі блоки (header, footer, section, nav, article, aside, figure); з'явилися нові графічні можливості та додана підтримка мультимедійного контенту; завантаження сторінок сайту відбувається значно швидше; не потрібно підключати додаткові розширення до браузера, щоб відтворити аудіо чи відео; cookie, які вже морально застаріли і не всюди годяться, тепер можуть бути замінені на локальне сховище на стороні клієнта. Інформацію можна зберігати на пристрої клієнта в набагато більшому розмірі і з ширшими можливостями, ніж у cookie.

CSS це мова стилів, що визначає відображення HTML-документів. Вона використовується для визначення кольорів, шрифтів, верстки та інших аспектів вигляду веб-сторінки. Одна з головних переваг – можливість розділити зміст сторінки від вигляду документу (що описується в CSS). Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш

структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення.

До основних переваг CSS3 можна віднести: можливість одночасно використовувати фонове зображення і фоновий колір; додавання тіней до блоків та тексту; додавання прозорості для блоків; створення градієнтів – переливання кольору; час затримки зміни властивості; для написання коду на HTML чи CSS використовуються текстові редактори. Це є основний інструмент для верстання веб-сторінки. Він повинен бути зручним, функціональним та швидким, допомагати автоматизувати написання коду, оскільки в ньому проводиться основна частина робочого часу. На сьогодні, найбільш популярними текстовими редакторами на ринку є Sublime Text, Atom, Visual Studio Code. Розглянемо переваги і недоліки цих редакторів.

Sublime Text 3 – це швидкий крос платформний редактор початкових текстів програм. Підтримує плагіни, розроблені за допомогою мови програмування Python. До переваг даного редактора можна віднести: автоматичне збереження; закладки в файлах; мінімапа: попередній перегляд повного початкового тексту; можливість мультивибору тексту; редагування на кількох панелях; вбудована підтримка 27 мов програмування; підсвітка синтаксису, що повністю налаштовується; відповідність дужок, автодоповнення; користувацькі гарячі клавіші; функція швидкого переходу до певної ділянки коду; надання швидкого доступу до команд (Палітра команд); індексування файлів для пошуку оголошення змінної, функції або класу як в сучасних IDE; переміщення плагінів в окремий процес для запобігання затримок завантаження і завершення роботи Sublime Text; перехід до попередніх позицій курсору, відсутність підтримки синтаксів Rust, Kotlin и GLSL; нема історії буфера обміну; не можливо переглянути список закладок та давати їм імена.

Текстовий редактор Atom — це вільний текстовий редактор і редактор коду, який може використовуватися як самодостатнє рішення, так і у ролі технологічного стека для побудови різних спеціалізованих рішень. Зокрема, на платформі Atom побудовані середовище розробки «Visual Studio Code» від

компанії «Microsoft» і «Nuclide» від «Facebook». Переваги даного редактора [44]: вбудований пакетний менеджер і інтерфейс навігації файловою системою, надає засоби для одночасної спільної роботи з кодом; має інтелектуальну систему автоматичного доповнення вводу; надає режими сумісності з Vim і Emacs, підтримує API для розробки розширень; кілька файлів можуть бути відкриті в різних вкладках і одночасно відображені з використанням вертикального або горизонтального розбиття панелей; інтерфейс може налаштовуватися через теми оформлення, підтримуються вкладки, закладки, розумний контекстний пошук коду, схлопування блоків коду; одночасне використання декількох курсорів і областей виділення; автоматичне доповнення та перевірка коду для різних мов (Ruby, Python, SQL, PHP, Perl, Objective-C, C/C++, JavaScript, Java, Go тощо.

React - це декларативна, ефективна та гнучка JavaScript бібліотека для створення користувальницьких інтерфейсів. Вона дозволяє збагатити складний інтерфейс з маленьких виділених кусочків кодів, названих «компонентами». Дозволяє створювати різних розмірів програми, що мають великий обсяг даних який змінюються без перезавантаження. Використовується для рендерингу HTML. Важливою особливістю ReactJs є використання JSX. Це надбудова на JS, що дозволяє використовувати про-XML синтаксис в Javascript коді.

JSX - це поєднання javascript і html, які в зв’язці є незвичним синтаксисом для більшості розробників. JSX дозволяє бачити всі процеси в одному місці, не відволікаючись на складності грамотного і валідованого коду.

PHP (Hypertext Preprocessor) – є універсальною мовою сценаріїв з відкритим кодом, яка спочатку розроблялася як препроцесор гіпертекстових сторінок, тому PHP може бути легко інтегрована в HTML код. Синтаксис PHP дуже схожий з синтаксисом мов C або Perl, але сама мова набагато простіша у вивченні. У PHP реалізовані механізми безпеки, які при правильному налаштуванні адміністраторами забезпечують максимальну свободу дій і безпеку. PHP може працювати в так званому безпечному режимі (safe mode), який обмежує можливості застосування мови користувачами по ряду важливих показників. У стандартний набір функцій PHP входить ряд надійних механізмів

шифрування. PHP використовують для написання серверних скриптів, які виконують завдання надання сайтам максимальної динаміки і інтерактивності. Зокрема: для генерації динамічних веб-сторінок; опрацювання даних різних форм; відсилання і прийом cookies. Недоліком PHP можна вважати складності при написанні об’ємних додатків і велика тривалість опрацювання такої програми.

MVC - це патерн проектування веб-додатків, який включає в себе декілька більш менших шаблонів. При використанні MVC модель даних програми, призначена для користувача, інтерфейс і взаємодія з користувачем розділені на три окремих компонента, завдяки чому модифікація одного з них надає мінімальний вплив на інші або не робить його зовсім.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі проаналізовано та описано статичний та динамічний тип маршрутизації, розглянуто характеристики протоколів маршрутизації. Детально розглянуто роботу стичної та динамічної маршрутизації з використанням графічних матеріалі метрик маршрутизації. Було розглянуто алгоритми маршрутизації та описано протоколи RIPта EIGRP. Було визначено проблеми алгоритму Беллмана-Форда та описано пошук альтернативного маршруту алгоритмом даним алгоритмом. Вибрано та обґрунтовано засоби вирішення проблеми для створення навчально-тренувальної програми розрахунку метрик протоколів динамічної маршрутизації (HTML, CSS, JavaScript, ReactJS).

РОЗДІЛ 3. Інтерфейс програми та робота з нею

3.1. Опис та реалізація завдання

Принцип роботи навчально-тренувальної програми для розрахунку метрик динамічних протоколів маршрутизації полягає в складанні та проходженні тестів користувачами для покращення та закріплення знань. Програма створена максимально простою та зручною будь якому користувачеві. Додаток написаний засобами JavaScript на базі фреймворку (бібліотеки) React Js, що дозволяє зробити додаток максимально зручним ефективним. В подальшому завдяки фреймворку React Native даний веб додадок можна імпортувати як мобільний додаток на платформи Android та iOS, що буде ще зручнішим для потенційних користувачів.

Для того, щоб відображати тести необхідно здійснювати їх імпорт з JSON файлу або з масиву програми. В нашому випадку пакети тестів описані у окремих масивах.

```
render() {
  const { quiz, screen } = this.props;
  if (!quiz) return null;
  const challenges = quiz.get('challenges');
  return (
    <div className='studyWrapper reviewContainer'>
            <div className='studyContainer'>
        <div className='quizHeader'>
          <div className='quizTitle'>
            <span>{quiz.get('title')} Overview</span>
            <span style={{ marginLeft: 10 }}>🔥♂</span>
          </div>
          <h3 className='quizMeta'>
            {challenges.size > 1 ? `${challenges.size} total questions` : ''}
          </h3>
          {screen.isDesktop && <span id="return">
            <Link to='/'>
              <i className="fa fa-times-circle" aria-hidden="true"></i>
            </Link>
          </span>}
        </div>
        {challenges.map(renderQuestion)}
            </div>
          </div>
```

Рис.3.1. Імпорт та налаштування тестів

Під час створення, імпортування чи копіювання запитань у банк запитань, можуть траплятись однакові запитання, відповіді чи описи у тестах. Через те, було вирішено створити алгоритм, для верифікації тестів. Завдяки такому

алгоритму, тести та відповіді у тестах не будуть повторюватись та не виникатимуть проблеми з їх розв’язанням.

```
const verifiedQuestions = category.challenges.filter(q => {
    if (!q.title || !q.subtitle || !q.solution) {
        return false;
    }
    if (q.choices.length < 2) return false;
    if (noRepeatTitle[q.title]) {
        rejectedTitles.push(q.title);
        return false;
    }
    if (noRepeatSubtitle[q.subtitle]) {
        rejectedTitles.push(q.subtitle);
        return false;
    }
    noRepeatTitle[q.title] = true;
    noRepeatSubtitle[q.subtitle] = true;
    if (!q.explanation) noExplanation++;
    return true;
});
if (verifiedQuestions.length > 0) {
    category.challenges = verifiedQuestions;
    return verified.concat(category);
} else {
    rejectedCategories.push(category.title);
    return verified;
}
```

Рис. 3.2 Алгоритм перевірки (верифікації) запитань

```
render() {
  const { quiz, screen } = this.props;
  if (!quiz) return null;
  const challenges = quiz.get('challenges');
  return (
    <div className='studyWrapper reviewContainer'>
            <div className='studyContainer'>
        <div className='quizHeader'>
          <div className='quizTitle'>
            <span>{quiz.get('title')} Overview</span>
            <span style={{ marginLeft: 10 }}></span>
          </div>
          <h3 className='quizMeta'>
            {challenges.size > 1 ? `${challenges.size} total questions` : ''}
          </h3>
          {screen.isDesktop && <span id="return">
            <Link to='/'>
              <i className="fa fa-times-circle" aria-hidden="true"></i>
            </Link>
          </span>}
        </div>
        {challenges.map(renderQuestion)}
            </div>
          </div>
```

Рис. 3.3. Алгоритм відображення тестів

Для перевірки та виводу правильної відповіді використовуємо наступний код (рис. 3.2.).

```
} else if (answer) {
    if (solution === idx) {
        return (
            <div
                key={key}
                className='choice' id='correctWinner'>
                <p>{this.renderMarkup(choice)}</p>
            </div>
        )
    } else {
        return (
            <div
                key={key}
                className='choice' id='wrongWinner'>
                <p>{this.renderMarkup(choice)}</p>
            </div>
        )
    }
```

Рис. 3.3. Вивід правильної відповіді

Для того, щоб отримати результат, по замовчуванню було створено алгоритм, завдяки якому визначатиметься чи успішно пройдений тест. У разі 75% правильних відповідей, програма сповістить користувача про хорошу

роботу, в іншому випадку, спроба не буде зарахована і потрібно повторити тест ще раз.

```
{complete &&
    <div>
        <h1 className='scoreMessage'>
            Твій результат {score} правильних відповідей серед {numberOfQuestions} запитань! { percentage > 0.75 ? 'Хороша робота!
        </h1>
        <Link className='finishBtn' to='/' onClick={() => this.props.finishQuiz()}>
            <button>Повернутись на Головну</button>
        </Link>

    </div>}

    {!isMobile && <div id='infoBox'>
        <p>Use <i className='fa fa-long-arrow-up'></i> <i className='fa fa-long-arrow-down'></i> space and esc</p>
    </div>}

</div>
</div>
```

Рис. 3.4. Алгоритм повідомлення про результат

3.2. Аналіз отриманих результатів

Для того, щоб переконатись у працездатності створеної системи, необхідно представити результати реалізації створеного програмного продукту. Для початку роботи зі системою, необхідно запустити React додаток. В подальшому даний проект передбачається перенесень на хостинг для глобального використання.

Головним вікном додатку є початкове меню, в якому можна обрати, який тест пройти або ж обрати «Змішати всі запитання».

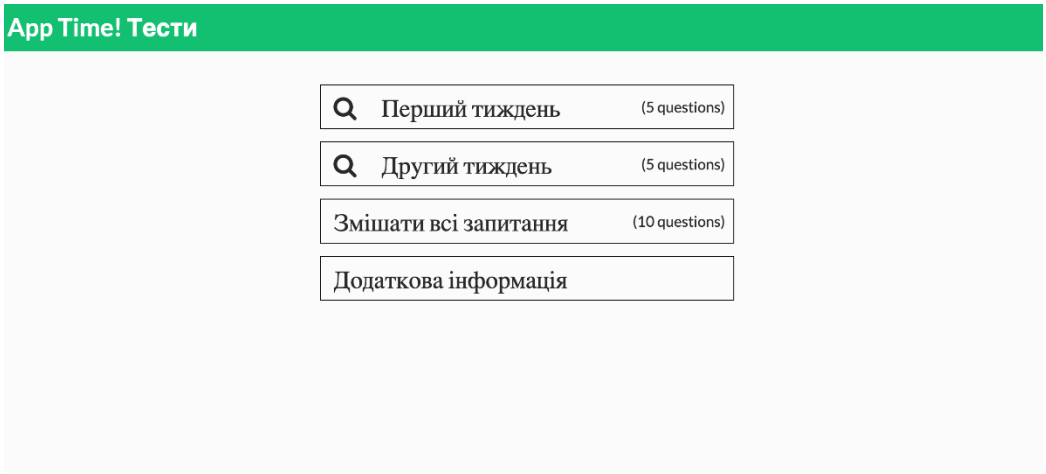


Рис.3.5. Головне вікно програми

У головному меню бачимо два набори тестів, які можна обрати для тренування. Перший тиждень, передбачає загальні запитання, які є легкої складності. Алгоритм передбачає функцію випадкових відповідей, тому місце правильної відповіді кожного разу інше. Інший блок запитань («Другий тиждень») передбачає запитання важкого рівня складності та включає в собі графічні запитання з визначенням метрик маршрутизації. Відповідно для засвоєння всіх запитань можна скористатись варіантом «Змішати запитання».

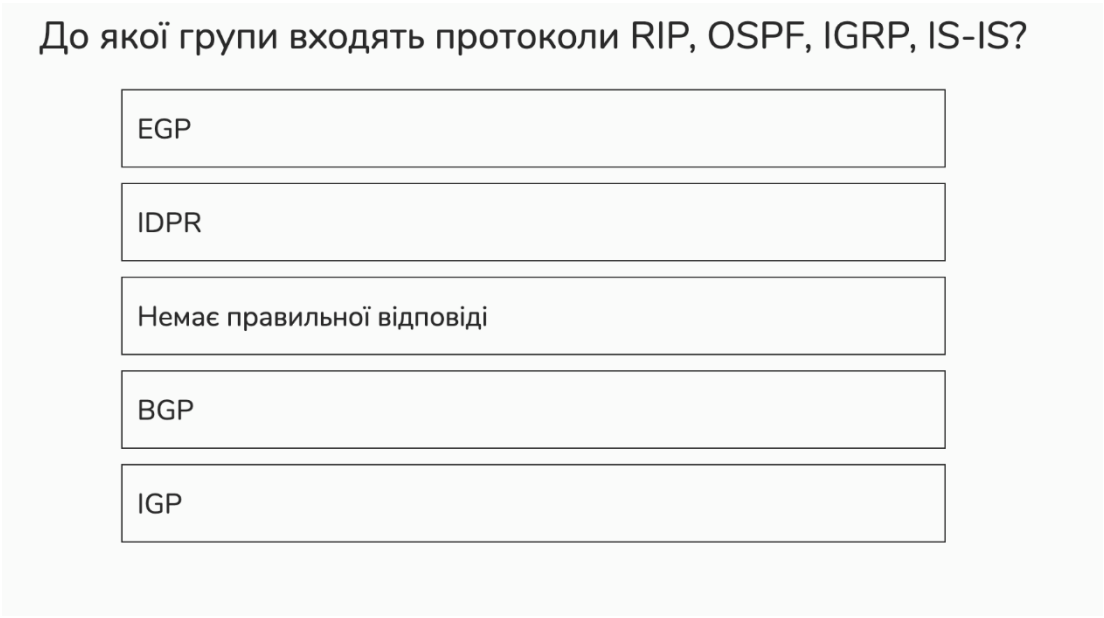


Рис.3.6. Приклад тестового завдання з блоку «Перший тиждень»

Після вибору відповіді, отримаємо сповіщення, яке відразу повідомить про правильно відповідь або ж навпаки.

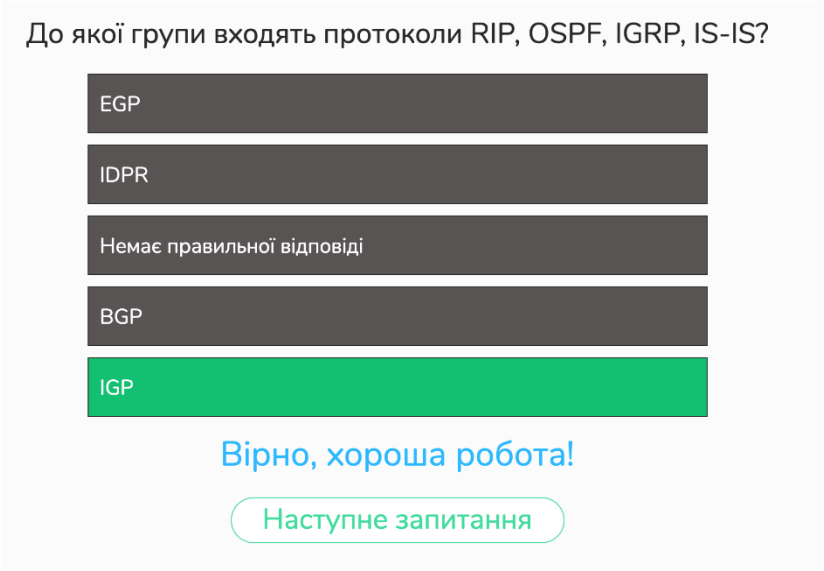


Рис. 3.7. Правильний варіант відповіді

Після правильної відповіді, ми отримаємо зелений фон на обраному варіанті та сповіщення про те, що тест виконано правильно.



Рис. 3.8. Не правильний варіант відповіді

В разі не правильної відповіді, отримаємо сповіщення про не правильно відповідь. Також, знизу з’являється пояснення, де описана правильну відповідь

та відповідну теорію. Завдяки такій функції, користувач, зможе не просто дізнаватись правильну відповідь але й поглиблювати знання та вчитись на помилках.



Рис. 3.9. Тест з блоку «Другий тиждень» з графічними матеріалами

Після завершення відповідного блоку запитань, буде можливість отримати результат тесту. Викладач або ж той хто створює тест, може обрати відсоток правильних запитань, для того щоб успішно пройти тест. За замовчуванням тест вважається спішно пройденим, якщо кількість правильних відповідей є більше, ніж 75%.

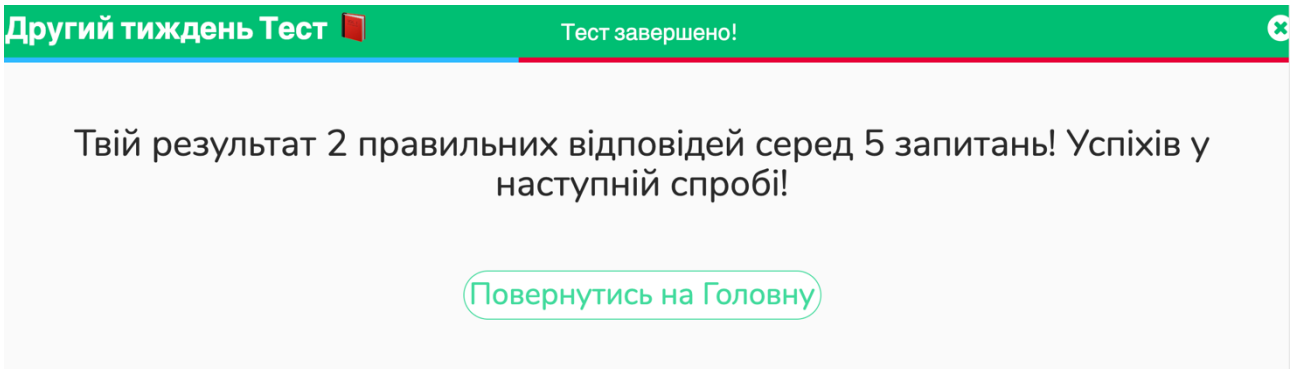


Рис. 3.10. Результат тесту, з негативним сповіщенням

ВИСНОВОК ДО РОЗДІЛУ 3

В третьому розділі атестаційної бакалаврської роботи передбачав розробку інтерфейсу програми та роботу з нею. У даному розділі було розглянуто алгоритми для максимально точної оцінки тестів створеної програми, Також було спроектовано та реалізовано зручний інтерфейс користувача, розроблено банк запитань у вигляді масивів та Json об’єктів, а також програмну частину засобами HTML, CSS, JSта фреймворком (бібліотекою) ReactJS. Також, було розглянуто та протестовано додаток.

ВИСНОВКИ

В даній бакалаврській кваліфікаційні роботі було описано процес розроблення навчально-тренувальної програми для розрахунку метрик протоколів динамічної маршрутизації.

У першому розділі було досліджено актуальність роботи та проаналізовано літературні джерела.

В першому розділі проведено аналітичний огляд існуючих рішень. Для того, щоб показати всі переваги і недоліки розроблюваної системи необхідно проведено аналіз та порівняння з іншими подібними системами. Проаналізувавши переваги та недоліки існуючих програмних засобів для створення тестових завдань, було зведено інформацію про основні функції проектованої системи. Результатом аналізу є перелік особливостей, що необхідно врахувати при проектуванні даної програмної системи.

Дана система дозволяє створювати тестові завдання різних видів складності, полегшує навчальний процес у закладах вищої освіти, допомагає закріпити здобуті знання. Для отримання необхідного результату, було пройдено багато етапів. На етапі здійснення аналітичного огляду, було проведено аналіз методологічних засад та дослідження галузі комп'ютерних мереж.

Для реалізації програмної частини було використано мову розмітки HTML, мову опису стилів CSS, мову програмування JavaScript та застосовано бібліотеку React Js.За допомогою Cisco Packet Tracer а також навчальних матеріалів Cisco Academy було створено блоки завдань та імпортовано їх у додаток у вигляді Json.

Результатом атестаційної бакалаврської роботи є готове програмне рішення, яке можна застосовувати у вищих навчальних закладах, а також школах та коледжах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Али С. Потоковая модель динамической балансировки очередей в MPLS-сети с поддержкой traffic engineering queues / С. Али, А.В. Симоненко // Проблемы телекоммуникаций. — 2010. — № 1. — с. 59–67.

2. Бараш Л. Виртуальные частные сети на базе MPLS / Леонид Бараш // Компьютерное Обозрение. — 2004. — № 17. — с. 14–16.

3. Cisco Systems: документация [Электронный ресурс]. Режим доступа: http://www.cisco.com/en/US/docs .

4. Cisco Systems: конфигурации [Электронный ресурс]. Режим доступа: http://www.cisco-conf.ru/ .

5. Braden, R., and Postel, J., "Requirements for Internet Gateways",USC/Information Sciences Institute, RFC-1009, June 1987.

6. Гольдштейн А.Б. Технология и протоколы MPLS / А.Б. Гольдштейн, Б.С. Гольдштейн. — СПб. : БХВ – Санкт-Петербург, 2005. — 304 с.

7. Олвейн В. Структура и реализация современной технологии MPLS. Руководство Cisco / Вивек Олвейн. — М. : Вильямс, 2004. — 480 с.

8. Олифер В. Искусство оптимизации трафика / В. Олифер, Н. Олифер // Журнал сетевых решений LAN. — 2001. — № 12. — с. 21–26.

9. Rosen, E. Multiprotocol Label Switching Architecture / E. Rosen, A. Viswanathan, R. Callon. — RFC 3031, 2001. — 61 с

10.Томас Т. Структура и реализация сетей на основе протокола OSPF. Руководство Cisco / Том Томас. — М. : Вильямс, 2004. — 816 с.

11. Вивек Олвейн. Структура и реализация современной технологи MPLS. Руководство Cisco = Advanced MPLS Design and Implementation. — М.: Вильямс, 2004. — 480 с. — ISBN 1-58705

12. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях. — М.: КУДИЦ-ОБРАЗ, 2001. — 368 с.

13. Столлингс В. Основы защиты сетей. Приложения и стандарты Network Security Essentials. Applications and Standards. —М.:«Вильямс», 2002. — С. 432. — ISBN 0-13-016093-8

14. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер 2006. -958с.

15. Заика А. Компьютерные сети / А. Заика. – М.: ИД "Олма-Пресс", 2006 – 448 с.

16. Документация компании Cisco [Электронный ресурс]. – Режим доступа к ресурсу:http://www.cisco.com/en/US/docs/ios/12_2/ip/configuration/guide/1cfipadr.html# wp 1001061.

17. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский . – М.: Техносфера, 2003. – 512 с.

18. Куроуз Дж. Компьютерные сети. 2-е изд / Дж. Куроуз, К. Росс. – СПб.: Питер, 2004. – 765 с.

19. Основы компьютерных сетей: учеб. пособие / Под ред. Л.Г. Гагариной. – М.: Форум»: ИНФРА-М, 2007. – 272 с.

20. Савельев А. Современные протоколы маршрутизации. [Электронный ресурс]. – Режим доступа к ресурсу: http://www.citforum.ru/nets/articles/ sovr_marsh.shtml.

21. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. / В.Г. Олифер , Н.А. Олифер. – СПб.: Питер, 2010. – 944 с.

22. Таненбаум Э. Компьютерные сети. 4-е изд. / Таненбаум Э. – СПб.: Питер, 2007. – 992 с.

23. Олифер В.Г. Протокол межсетевого взаимодействия IP [Электронный ресурс] / В.Г. Олифер, Н.А. Олифер. – Режим доступа к ресурсу: http://cylib.iit.nau. edu.ua/Books/NetWork/ ip/contents.htm.

24. Олифер Н.А. Использование моделирования для оптимизации производительности сети [Электронный ресурс] / В.Г. Олифер, Н.А. Олифер.. – Режим доступа к ресурсу: http://www. citforum.ru/ nets/optimize/locnop_09.shtml.

25. Плешаков В. Основы маршрутизации [Электронный ресурс]. – Режим доступа к ресурсу: http://www.citforum.ru/ nets/ito/2.shtml.

26. Остерлох Х. Маршрутизация в IP-сетях. Принципы, протоколы, настройка: Пер. с англ. / В. Плешаков. – СПб.: ООО «ДиаСофтЮП», 2002. – 512 с.

ДОДАТКИ

ДОДАТОК А

Код програми

```
export default (function(challenges) {

    let noExplanation = 0;
    let rejectedTitles = [];
    let rejectedCategories = [];

    const verified = challenges.reduce((verified, category) => {

        const noRepeatTitle = {};
        const noRepeatSubtitle = {};

        if (!category.title || !category.category) {
            rejectedCategories.push(category.title);
            return verified;
        }
        const verifiedQuestions = category.challenges.filter(q => {
            if (!q.title || !q.subtitle || !q.solution) {
                return false;
            }
            if (q.choices.length < 2) return false;
            if (noRepeatTitle[q.title]) {
                rejectedTitles.push(q.title);
                return false;
            }
            if (noRepeatSubtitle[q.subtitle]) {
                rejectedTitles.push(q.subtitle);
                return false;
            }
```

```
                noRepeatTitle[q.title] = true;
                noRepeatSubtitle[q.subtitle] = true;
                if (!q.explanation) noExplanation++;
                return true;
        });
        if (verifiedQuestions.length > 0) {
                category.challenges = verifiedQuestions;
                return verified.concat(category);
        } else {
                rejectedCategories.push(category.title);
                return verified;
        }
}, []);

/* Log notes about any removed challenges/content: */
if (rejectedCategories.length || noExplanation) console.warn('Take Note:');

if (rejectedTitles.length) {
        console.log(`The following question titles are duplicates, titles must be ` +
                `unique: ${rejectedTitles.join(', ')}`
        );
}

if (rejectedCategories.length) {
        console.log(`The following quiz categories could not be added, ` +
                `probably because they are incomplete: ${rejectedCategories.join(', ')}`
        );
}

if (noExplanation) console.log(`${noExplanation} Questions have no explanation. ` +
        `Adding explanations will improve the learning experience.`);
```

```
        return verified;

})(challenges);

////////////////////////////////////////////

import React from 'react';
import Prism from 'prismjs';
import { connect } from 'react-redux';
import { Link } from 'react-router-dom';
import { connectScreenSize } from 'react-screen-size';
import { findQuiz, mapScreenSizeToProps } from '../utils/helpers';

const renderMarkup = (html) => {
  return (
<span dangerouslySetInnerHTML={{__html: html}}></span>
  );
}

const renderQuestion = (question) => {
  const solution = +question.get('solution');
  const title = question.get('title');
  const choices = question.get('choices');
  const explanation = question.get('explanation');
   return (
<div key={title}>
<div className='reviewTitle'>
<h1 className='questionTitle'>{renderMarkup(title)}</h1>
</div>
    {choices.map((choice, index) => (
```

```
<div
        key={choice}
        className={solution === index ? 'choice review solution' : 'choice review'}>
<p>{renderMarkup(choice)}</p>
</div>
    ))}
    {explanation && (
<div className='explanation'>
<h3>Пояснення:</h3>
<p>{renderMarkup(explanation)}</p>
</div>
    )}
</div>
 );
};

/* Review Quiz Questions Component */
class Review extends React.Component {
 componentDidMount() {
   document.addEventListener('keydown', this.handleKeyDown);
   Prism.highlightAll();
 }
 componentWillUnmount() {
   document.removeEventListener('keydown', this.handleKeyDown, false);
 }
 componentDidUpdate() {
   Prism.highlightAll();
 }
 handleKeyDown = ({ code }) => {
                if (code === 'Escape') this.props.history.push('/');
        }
```

```
 render() {
   const { quiz, screen } = this.props;
   if (!quiz) return null;
   const challenges = quiz.get('challenges');
   return (
<div className='studyWrapper reviewContainer'>
                                        <div className='studyContainer'>
<div className='quizHeader'>
<div className='quizTitle'>
<span>{quiz.get('title')} Overview</span>
<span style={{ marginLeft: 10 }}>🏳‍🌈</span>
</div>
<h3 className='quizMeta'>
            {challenges.size > 1 ? `${challenges.size} total questions` : ''}
</h3>
          {screen.isDesktop &&<span id="return">
<Link to='/'>
<i className="fa fa-times-circle" aria-hidden="true"></i>
</Link>
</span>}
</div>
        {challenges.map(renderQuestion)}
                                        </div>
                                </div>
   )
 }
}

const mapStateToProps = (state, props) => {

            const { title } = props.match.params;
```

```
  const quizzes = state.get('quizzes');

            const quiz = findQuiz(title.replace(/-/g, ' '), quizzes);

  if (!quiz) {
    props.history.push('/');
  }

            return { quiz };

};

////////////////////////////////////

export default class Quiz extends React.Component {
   constructor(props) {
super(props);
this.state = {
        complete: false,
        selection: null,
        answer: null,
      }
      document.addEventListener('keydown', this.handleKeyDown);
   }
   componentWillUnmount() {
      document.removeEventListener('keydown', this.handleKeyDown, false);
   }
   componentDidMount() {
const subtitle = this.props.meta.getIn(['currentQuestion', 'subtitle']);
setHtmlMetadata(subtitle);
      Prism.highlightAll();
   }
   componentWillReceiveProps(nextProps) {
if (this.props.meta !== nextProps.meta) {
const subtitle = nextProps.meta.getIn(['currentQuestion', 'subtitle']);
setHtmlMetadata(subtitle);
      }
   }
   componentDidUpdate() {
      Prism.highlightAll();
   }
onHover = () =>this.setState({ selection: null });
handleKeyDown = (event) => {
```

```
const { code } = event;

if (code === 'Space' || code === 'ArrowDown' || code === 'ArrowUp') {
        event.preventDefault();
      }

let {
        answer,
        selection,
        complete
      } = this.state;

const { meta } = this.props;

const question = meta.get('currentQuestion');
const questions = question.get('choices').size;
const solution = +question.get('solution');

switch(code) {
case 'Space':
if (complete) {
this.props.finishQuiz();
this.props.history.push('/');
        } else if (selection !== null) {
this.handleAnswer(selection, solution);
        } else if (answer !== null) {
this.nextQuestion();
        }
break;
case 'ArrowDown':
if (selection === null) {
this.setState({ selection: 0 });
        } else {
          selection++;
if (selection === questions) {
this.setState({ selection: 0 });
          } else {
this.setState({ selection });
          }
        }
break;
case 'ArrowUp':
if (selection === null) {
this.setState({ selection: questions - 1 });
        } else {
          selection--;
if (selection === -1) {
this.setState({ selection: questions - 1 });
          } else {
this.setState({ selection });
          }
        }
break;
case 'Escape':
```

```
this.props.finishQuiz();
this.props.history.push('/');
break;
default:
return;
    }
  }
handleAnswer = (choice, solution) => {
if (choice === solution) {
this.setState({
        answer: true,
        selection: null,
      });
    } else {
this.setState({
        answer: false,
        selection: null
});
    }
  }
nextQuestion = () => {
const { meta, title } = this.props;
const index = meta.get('index');
const length = meta.getIn(['quiz', 'challenges']).size;
if (this.state.answer) this.props.correct();
if (index === length - 1) {
this.props.viewResults();
this.setState({ complete: true });
    } else {
this.props.nextQuestion();
const nextTitle = meta
        .getIn(['quiz', 'challenges'])
        .find((v, k) => k === (index + 1))
        .get('subtitle')
        .replace(/\s/g, '-');
this.props.history.replace(`/practice/${title.replace(/\s/g, '-')}/${nextTitle}`);
this.setState({
        answer: null,
        selection: null
});
    }
  }
renderMarkup = (html) => {
return (
<span dangerouslySetInnerHTML={{__html: html}}></span>
    );
  }
  render() {

const { selection, answer, complete } = this.state;
const { meta, screen } = this.props;
const { isMobile, isDesktop } = screen;

const quiz = meta.get('quiz');
```

```
const score = meta.get('score');
const index = meta.get('index');
const  numberOfQuestions = quiz.get('challenges').size;
const currentQuestion = meta.get('currentQuestion');
const explanation = currentQuestion.get('explanation');
const solution = +currentQuestion.get('solution');
const percentage = score / meta.getIn(['quiz', 'challenges']).size;

const renderClassName = (i) => {
return (selection === i)
        ? `choice selected ${isMobile ? 'mobile' : 'desktop'}`
        : `choice ${isMobile ? 'mobile' : 'desktop'}`;
    };

const tower = createScoreMeter(score, index, numberOfQuestions);
const widthPercentage = (100 / numberOfQuestions);

return (
<div className='studyWrapper'>
<div className='studyContainer'>

<div id='score-tower'>
                {tower.map((type, i) => (
<div
className={type}
key={quiz.get('challenges').find((v,k) => k === i)}
style={{ width: `${widthPercentage}%` }}>
</div>
                ))}
</div>

<div className='quizHeader'>
<div className='quizTitle'>
<span>{quiz.get('title')} Тест</span>
<span style={{ marginLeft: 10 }}></span>
</div>
                {!this.state.complete
                  ? <h3 className='quizMeta'>Запитання {index + 1} з {numberOfQuestions}</h3>
                  : <h3 className='quizMeta'>Тест завершено!</h3>}
                  {isDesktop &&<span id=''return''>
<Link to='/'>
<i className=''fa fa-times-circle'' aria-hidden=''true''></i>
</Link>
</span>}
</div>

              {!complete &&<h1 className='questionTitle'>
                {this.renderMarkup(currentQuestion.get('title'))}
</h1>}

              {!complete && currentQuestion.get('choices').map((choice, idx) => {
const key = (choice + idx);
/* User has not selected an answer yet: */
                if (answer === null) {
```

```
return (
<div
key={key}
className={renderClassName(idx)}
onMouseEnter={this.onHover}
onClick={() =>this.handleAnswer(idx, solution)}>
<p>{this.renderMarkup(choice)}</p>
</div>
                    )
/* User selected the correct answer: */
} else if (answer) {
if (solution === idx) {
return (
<div
key={key}
className='choice' id='correctWinner'>
<p>{this.renderMarkup(choice)}</p>
</div>
                      )
                    } else {
return (
<div
key={key}
className='choice' id='wrongWinner'>
<p>{this.renderMarkup(choice)}</p>
</div>
                      )
                    }
/* User selected the wrong answer: */
} else {
if (solution === idx) {
return (
<div
key={key}
className='choice' id='correctLoser'>
<p>{this.renderMarkup(choice)}</p>
</div>
                      )
                    } else {
return (
<div
key={key}
className='choice' id='wrongLoser'>
<p>{this.renderMarkup(choice)}</p>
</div>
                      )
                    }
                  }
                })}

            {answer !== null && !complete &&
<div className='messageDiv'>
                  {answer
                    ? <h1 className='correctAnswer'>Вірно, хороша робота!</h1>
```

```
                : <h1 className='wrongAnswer'>Нажаль,відповідь не правильна!</h1>}
              {answer !== null && !answer && explanation && (
<div className='explanation'>
<h3>Пояснення:</h3>
<p>{this.renderMarkup(explanation)}</p>
</div>
              )}
              {index + 1 === numberOfQuestions
                ? <button onClick={this.nextQuestion}>Переглянути результат</button>
                : <button onClick={this.nextQuestion}>Наступне запитання</button>}
</div>}

          {complete &&
<div>
<h1 className='scoreMessage'>
                Твій результат {score} правильних відповідей серед {numberOfQuestions} запитань! { percentage >
0.75 ? 'Хороша робота!' : 'Успіхів у наступній спробі!'}
</h1>
<Link className='finishBtn' to='/' onClick={() =>this.props.finishQuiz()}>
<button>Повернутись на Головну</button>
</Link>

</div>}

            {!isMobile &&<div id='infoBox'>
<p>Use <i className='fa fa-long-arrow-up'></i><i className='fa fa-long-arrow-down'></i> space and esc</p>
</div>}

</div>
</div>
    );
  }
};
```