**Національний технічний університет України**

**«Київський політехнічний інститут»**

факультет інформатики та обчислювальної техніки

кафедра обчислювальної техніки

Курсовий проект

*з дисципліни:* **«Проектування комп’ютерних систем»**

|  |  |
| --- | --- |
| **Виконав**: | Мазур Роман Федорович |
| **Група:** | ІО-52 |
| **Перевірили**: | Луцький Г. М. \_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис керівника)  Кулаков Ю. А. \_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис керівника) |

2009 р.

ТЕХНІЧНЕ ЗАВДАННЯ

1. Синтезувати оригінальну масштабовану топологічну організацію. Представити рисунок організації.
2. Визначити топологічні характеристики синтезованої організації. Проаналізувати такі параметри, як: ступінь, діаметр, середній діаметр, вартість, трафік, зв’язність.
3. Порівняти синтезовану топологію з гіперкубом.
4. Відобразити топологію в гіперкуб.
5. Розробити відмовостійкі алгоритми для чотирьох видів маршрутизації для синтезованої топології:
   1. один до одного,
   2. усі до всіх з персональним призначенням,
   3. один до всіх,
   4. multicast.
6. Розглянути розв’язання прикладної задачі на синтезованій топологічній організації.

ПОЯСНЮВАЛЬНА ЗАПИСКА

**Зміст**

[1 Синтез топології 6](#_Toc250904874)

[1.1 Принципи побудови 6](#_Toc250904875)

[1.2 Розмір та нумерація топології 8](#_Toc250904876)

[1.3 Закони формування зв’язків у топології 9](#_Toc250904877)

[1.4 Базові функції для роботи з топологією 9](#_Toc250904878)

[2 Аналіз топології 11](#_Toc250904879)

[2.1 Визначення топологічних характеристик 11](#_Toc250904880)

[2.2 Порівняння топології з гіперкубом 12](#_Toc250904881)

[3 Відображення топології у гіперкуб 13](#_Toc250904882)

[4 Маршрутизація 14](#_Toc250904883)

[4.1 Один до одного 14](#_Toc250904884)

[4.2 Усі до всіх з персональним призначенням 14](#_Toc250904885)

[4.3 Один до всіх 14](#_Toc250904886)

[4.4 Multicast 14](#_Toc250904887)

[5 Завантаження прикладної задачі 15](#_Toc250904888)

1 Синтез топології

1.1 Принципи побудови

В основу структурної організації топології було вирішено покласти гіперкуб у зв’язку з тим, що його побудова дає можливість завантажувати дуже великий спектр прикладних задач. Для того, щоб збільшити відмовостійкість топології основою його системи числення була вибрана тріскова замість двійкової, класичної. Цікавою особливістю такої топології є те, що трійковий гіперкуб 2-го порядку вироджується у меш-топологію 3х3 (рис. 1.1).



Рис. 1.1 Трійковий гіперкуб 2-го порядку

Основною вадою гіперкубічних топологій є їхній високий степінь, який визначається алгебраїчно степеневою функцією (*аN*). У нашому випадку ріст значення степеня буде стрімкішим за класичний через більше значення основи системи числення. Це, у свою чергу, спричиняє швидке зростання вартості системи. Таким чином, необхідно зупинити зростання степеня топології.

Для мінімізації вартості системи вершиною гіперкуба виступає не окремий вузол, а кластер з кількістю вузлів рівною подвоєному значенню порядку системи. Оскільки, в такому випадку кількість вузлів у кластері буде рівною степеню вершини 3-вого гіперкуба, то кожен з вузлів матиме один зв’язок з іншим кластером. Найпростіший варіант зв’язків у кластері – кільце (рис. 1.2).

|  |  |
| --- | --- |
| а) | б) |
| Рисунок 1.2 а) вершина трійкового гіперкуба 3-го порядку,  б) зменшення степеня вершини | |

Зі збільшенням порядку топології все більше значення матимуть параметри зв’язності внутрішнього кластера, а не зовнішнього гіперкуба. Тому, щоб збільшити зв’язність у кластері, введено додаткове ребро з «протилежною» вершиною. У результаті отримано топологічну організацію, показану на рисунках 1.3 та 1.4.

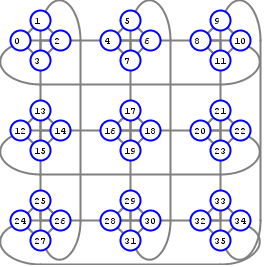


Рисунок 1.3 Топологічна організація 2-го порядку

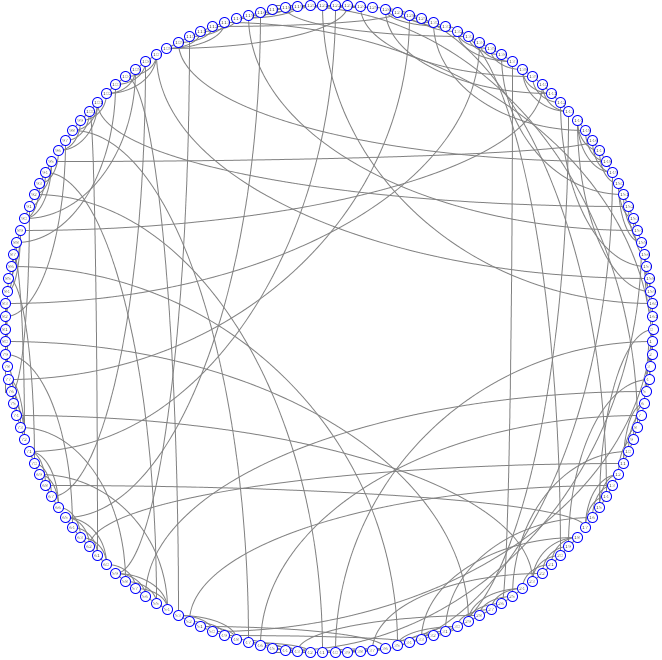


Рисунок 1.4 Топологічна організація 3-го порядку

1.2 Розмір та нумерація топології

Топологічна організація формується з кластерів. Їхня кількість визначається кількістю вершин трійкового гіперкуба відповідного порядку. Тобто, для організації з порядком *d* матимемо 3*d* кластерів. Кожен кластер має розмір (кількість вузлів, що його формують) рівний степеню вершини трійкового гіперкуба, тобто .

Кожен вузол у топології має наскрізний номер, однак в алгоритмах, звісно, часто використовуються поняття номера кластера та номера вузла у кластері. Перший може бути визначений, як частка від ділення наскрізного номера на розмір кластера, а другий, як остача цього ділення.

1.3 Закони формування зв’язків у топології

Як уже було сказано, кожен вузол у синтезованій топології має 4 зв’язки: три з них – зв’язки всередині кластера, і ще один – ребро зовнішнього трійкового гіперкуба. Нехай, *d* – порядок топології, *cs* – розмір кластера, *і* – номер вузла (наскрізний), *N* – кількість вершин у топології. Тоді чотири зв’язки *і*-го вузла можуть бути описані наступним чином ().

1. – зв’язок усередині кластера, за годинниковою стрілкою, рисунок 1.5 а).
2. – зв’язок усередині кластера, проти годинникової стрілки, рисунок 1.5 б).
3. – зв’язок усередині кластера з «протилежною» вершиною, рисунок 1.5 в).
4. – зв’язок з відповідною вершиною сусіднього кластера, рисунок 1.5 г).



Рисунок 1.5 Чотири зв’язки між вузлами

1.4 Базові функції для роботи з топологією

Далі наведемо основні функції, що можуть бути корисними при роботі з синтезованою топологічною організацією. В усіх формулах *d* – порядок топології, *і* та *j* – номера вузлів.

|  |  |
| --- | --- |
| Таблиця 1.1 Базові функції | |
| Кількість вузлів |  |
| Розмір кластера |  |
| Номер кластера |  |
| Номер вузла у кластері |  |
| Номер відповідної вершини сусіднього кластера | **int** axis = ni % d;  **int** sourceConnector = ni / d;  **int** nearDigit = *getDigit*(ci, axis);  nearDigit += sourceConnector == 0 ? -1 : 1;  nearDigit %= 3;  **if** (nearDigit < 0) { nearDigit=3+nearDigit; }  **int** nearCluster = *setDigit*(ci,axis,nearDigit);  **return** nearCluster \* cs + (ni + d) % cs; |
| Отримання масиву сусідніх вершин | **int** startNode = i / cs \* cs;  **return** **new** Integer[] {  startNode + (i + 1) % cs,  startNode + (i > 0 ? (i - 1) % cs : cs - 1),  startNode + (i + d) % cs,  *getNearClusterConnection*(i, d)  }; |

2 Аналіз топології

2.1 Визначення топологічних характеристик

Розглянемо наступні топологічні характеристики:

* N – кількість вузлів;
* S – степінь;
* D – діаметр;
* Dсер – середній діаметр;
* C – вартість;
* T – трафік.

Дані параметри залежатимуть від порядку топологічної організації, тому розглянемо закони, за якими вони можуть бути визначені.

Кількість вузлів має дуже стрімке зростання при масштабуванні та визначається за формулою

Степінь є геометричною характеристикою, що характеризує локальну зв’язність організації та визначається як максимальне число ребер, інцидентних одній вершині. Для синтезованої топологічної організації степінь є константою, що було однією із задач синтезу.

Діаметр топології є мінімальною відстанню між максимально віддаленими вершинами в топології. Його можна визначити через матрицю зв’язності. Для синтезованої організації діаметр визначається формулою

Середній діаметр показує усереднену відстань між вузлами у топології та обчислюється за допомогою матриці відстаней ().



Вартість визначається як мультиплікативний критерій діаметра, кількості вузлів та степеню.

Вартість показує, на скільки раціонально використовувати топологію при заданих параметрах, тобто фактично визначає співвідношення «ціна-якість».

Трафік (або густина трафіку) визначається за формулою

Дана характеристика показує ефективність використання ребер у топології. Чим ближче її значення до 1, тим ефективнішим є потенційне використання. У таблиці 2.1 наведено дані показники для різного порядку топології.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблиця 2.1 Топологічні характеристики | | | | | | |
| d | N | S | D | Dсер | T | C |
| 2 | 36 | 4 | 5 | 3,11 | 2,08 | 540 |
| 3 | 162 | 4 | 9 | 5,27 | 3,52 | 4374 |
| 4 | 648 | 4 | 11 | 6,99 | 4,66 | 21384 |

2.2 Порівняння топології з гіперкубом

Наведемо таблицю з топологічними характеристиками для гіперкуба.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблиця 2.2 Топологічні характеристики гіперкуба | | | | | |
| N | S | D | Dсер | T | C |
| 32 | 4 | 5 | 2,58 | 1,29 | 640 |
| 64 | 5 | 6 | 3,05 | 1,22 | 1920 |
| 128 | 6 | 7 | 3,53 | 1,18 | 5376 |
| 256 | 7 | 8 | 4,02 | 1,15 | 14336 |
| 512 | 8 | 9 | 4,51 | 1,13 | 36864 |
| 1024 | 9 | 10 | 5 | 1,11 | 92160 |

3 Відображення топології у гіперкуб

Для відображення синтезованої топологічної організації у гіперкуб природно представити кожен з кластерів як одну вершину гіперкуба. У результаті отримаємо трійковий гіперкуб (рисунок 3.1). Таким чином задача відображення зводиться до відображення гіперкуба з системою числення 3 до двійкового.

|  |  |
| --- | --- |
|  |  |

Рисунок 3.1 Перехід до трійкового гіперкуба

У трійковому гіперкубі на кожній з осей маємо по три вершини. Це створює великі незручності при відображенні у двійкову систему. Для розв’язання даної проблеми можна збільшити кількість вершин на кожній з осей до чотирьох (рисунок 3.2). У такому випадку ми отримаємо двійкову гіперкубічну організацію.

|  |  |
| --- | --- |
|  |  |

Рисунок 3.2 Перехід до n-вимірної меш-топології

4 Маршрутизація

4.1 Один до одного

Особливості топологічної організації породжують два підтипи маршрутизації: маршрутизація в кластері та маршрутизація в зовнішньому гіперкубі.

Варто відзначити, що інформацію про відмови вузлів містять лише їхні сусіди, і рішення про обхід недосяжного вузла відбувається в динаміці. Такий підхід вимагає механізму уникнення циклів.

Для цього зберігається інформація про вже відвідані вузли. Вона може передаватися в заголовку повідомлення, що звісно, збільшує об’єм службової інформації. Однак великий об’єм буде передаватися лише у випадку великої кількості відмов (при збільшення довжини маршруту). Тому у випадках, коли даний момент стає критичним, варто скористатися технікою віртуальної комутації каналів, при якій встановити з’єднання, за допомогою маршрутизації спеціального пакета описаними способами, а потім передавати користувацькі дані за вже встановленим шляхом.

*Маршрутизація у кластері*

Нехай cnIndex – номер поточного вузла в кластері (cnIndex , тут і далі d – порядок гіперкуба).

Тоді зв’язані вузли можуть бути записані масивом

[(cnIndex + 1) % cs, (cnIndex + d) % cs, (cnIndex – 1) % cs]

З цих трьох вузлів вибирається той, який найменш віддалений від вузла призначення, доступний та ще не був відвіданий.

Якщо такого сусіда не знайдено, вузол призначення вважається недосяжним в рамках поточного кластера. У сусідній кластер, при цьому, пакет не передається, оскільки це може призвести до довгого подальшого маршруту, який, ти не менше, може й не досягнути вузла призначення.

Для обчислення відстані між вузлами в кластері використовується наступна функція:

**int** getInClusterDistance(**final** **int** i, **final** **int** j, **final** **int** d) {

**int** dif = j - i;

**if** (dif == 0) { **return** 0; }

**int** sign = dif > 0 ? 1 : -1;

dif = Math.*abs*(dif);

**int** cs = d << 1;

**int** next = 0;

**if** (dif > d) {

next = (i + (dif < d + (d >> 1) ? d : -sign)) % cs;

} **else** {

next = (i + (dif > d >> 1 ? d : sign)) % cs;

}

**if** (next < 0) { next = cs + next; }

**return** *getInClusterDistance*(next, j, d) + 1;

}

*Міжкластерна маршрутизація*

Міжкластерна маршрутизація здійснюється в трійковому гіперкубі. При чому кожен вузол кластера має один лінк даного гіперкуба.

Нехай currentNode – номер поточного вузла, destNode – номер вузла призначення.

Тоді

currentCluster = currentNode / 2d – номер поточного кластера,

destCluster = destNode / 2d – номер кластера, якому належить вузол призначення.

Знаходяться вісі гіперкуба, в яких відрізняються координати у поточного та цільового кластера. Маючи дані вісі, легко знаходяться сусідні кластери, через які може відбуватися передача даних. Сусідніми кластерами є ті, що відрізняються в координатах лише в одній осі.

Далі знаходяться вузли, що належать поточному кластеру та мають лінки до сусідніх кластерів (їх назвемо місцями стрибка). Ці дані можуть бути обчислені при ініціалізації кожного з вузлів системи та записані в локальну пам’ять для уникнення зайвих обчислень.

Знайдений масив вузлів сортується за зростанням відстані до поточного вузла (за допомогою функції getInClusterDistance). До нього додаються вузли, протилежні до знайдених.

Далі отриманий список переглядається у встановленому порядку та вибирається перше місце стрибка, яке є досяжним, а цільова вершина після стрибка ще не була відвідана.

Якщо ми знаходимося у вибраному вузлі, то стрибок відбувається, коли цільова вершина є доступною. Інакше розглядається наступний варіант.

Якщо після розгляду усіх елементів попереднього масиву рішення про перехід не було прийнято, то це означає, що варто спробувати виконати передачу по осі, де немає відмінностей в координатах. Вибираються всі місця стрибків, що не розглядалися раніше, алгоритм повторюється для них.

Якщо рішення про перехід не було прийнято й зараз, то вузол призначення вважається недосяжним.

Розглянемо приклад маршрутизації на топологічній організації третього порядку (рисунок 4.1). Повідомлення передається з 5-го в 100-ий вузол.

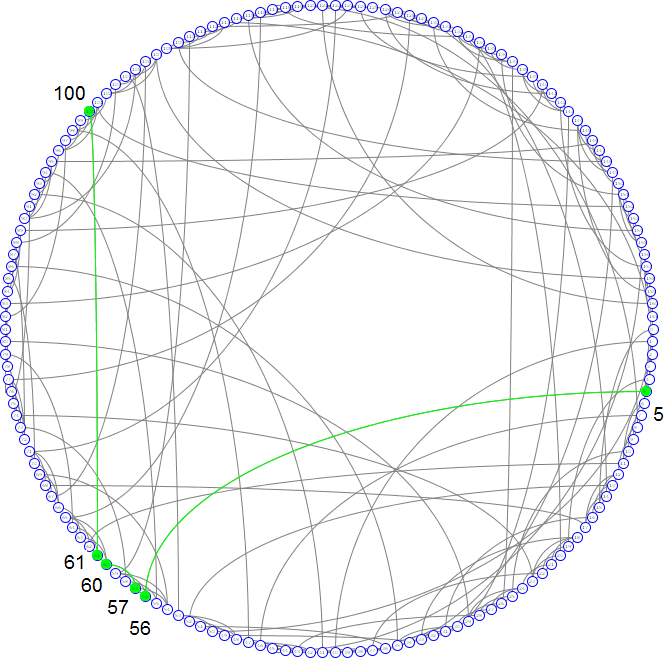


Рисунок 4.1 Приклад маршрутизації один до одного

Лог маршрутизації:

<Step1. [{5 -> 56: Jumping from 5 to 56.}]>

<Step2. [{56 -> 57: Look for 57 to jump.; go to 57}]>

<Step3. [{57 -> 60: Jumping from 57 to 60.}]>

<Step4. [{60 -> 61: Look for 61 to jump.; go to 61}]>

<Step5. [{61 -> 100: Jumping from 61 to 100.}]>

Розглянемо також приклад маршрутизації з вузлами, що відмовили (46 та 56).

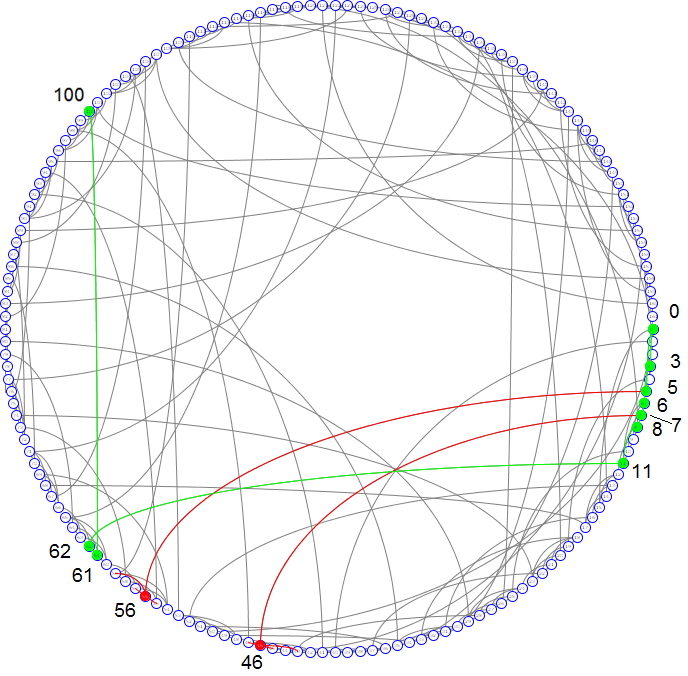


Рисунок 4.2 Приклад маршрутизації один до одного з вузлами, що відмовили

Лог маршрутизації:

<Step1. [{5 -> 0: Jump link to 56 is down.Look for 3 to jump.; go to 0}]>

<Step2. [{0 -> 3: Look for 3 to jump.; go to 3}]>

<Step3. [{3 -> 6: Jumping from 3 to 6.}]>

<Step4. [{6 -> 7: Look for 7 to jump.; go to 7}]>

<Step5. [{7 -> 8: Jump link to 46 is down.Look for 11 to jump.; go to 8}]>

<Step6. [{8 -> 11: Look for 11 to jump.; go to 11}]>

<Step7. [{11 -> 62: Jumping from 11 to 62.}]>

<Step8. [{62 -> 61: Look for 61 to jump.; go to 61}]>

<Step9. [{61 -> 100: Jumping from 61 to 100.}]>

4.2 Усі до всіх з персональним призначенням

4.3 Один до всіх

4.4 Multicast

5 Завантаження прикладної задачі