Міністерство освіти і науки України
Національний університет кораблебудування імені адмірала Макарова

В.М. Рябенький, О.М. Юрченко, О.О. Ушкаренко, М.Ф. Сопель

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ РОЗПОДІЛЕНИХ СИСТЕМ КЕРУВАННЯ АВТОНОМНИМИ ЕЛЕКТРОЕНЕРГЕТИЧНИМИ УСТАНОВКАМИ

Монографія

Київ 2012

УДК621.3:004.75:681.324
ББК 621.38
В 22

В монографії розглянуті питання інформаційного забезпечення розподілених систем керування автономними електроенергетичними установками.

Визначені проблеми керування інформаційними потоками в мережах розподілених систем керування, виконано аналіз характеристик мереж та шляхів покращення їх функціонування, запропоновані алгоритми керування інформаційними потоками з визначеними критеріями оптимізації. Розглянуті особливості використання уніфікованої мови моделювання при проектуванні розподілених систем керування, сформовані вимоги до програмного забезпечення автоматизованих електроенергетичних систем.

Для наукових та інженерно-технічних працівників, які виконують дослідження в галузі інформаційних технологій, телекомунікацій й займаються питаннями проектування й експлуатації розподілених систем керуавння.

Затверджено до друку Вченою радою Інституту електродинаміки НАН України (протокол №5 від 26.04.2012 р.)
Вченою радою Національного університету кораблебудування ім. адмірала Макарова (протокол №3 від 22.03.2012)

Рецензенти: О.В. Кириленко, акад.. НАН України
В.Я. Жуйков, докт. техн. наук

Автори: В.М. Рябенький, докт. техн. наук, проф.,
О.М. Юрченко, докт. техн.. наук,
О.О. Ушкаренко, канд. техн. наук,
М.Ф. Сопель, канд. техн. наук

Рябенький В.М., Юрченко О.М., Ушкаренко О.О., Сопель М.Ф.
Інформаційне забезпечення розподілених систем керування автономними електроенергетичними установками: монографія.

В монографии рассмотрены вопросы информационного обеспечения распределенных систем управления автономными электроэнергетическими установками.

Определены проблемы управления информационными потоками в сетях распределенных систем управления, выполнен анализ характеристик сетей и путей улучшения их функционирования, предложены алгоритмы управления информационными потоками с определенными критериями оптимизации. Рассмотрены особенности использования унифицированного языка моделирования при проектировании распределенных систем управления, сформированы требования к программному обеспечению автоматизированных электроэнергетических систем.

Для научных и инженерно-технических работников, которые выполняют исследования в области информационных технологий, телекоммуникаций и занимаются вопросами проектирования и эксплуатации распределенных систем управления.

ISBN 978-966-02-6439-7

ЗМІСТ

ВСТУП

Розділ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ТА ХАРАКТЕРИСТИКИ СИСТЕМИ КЕРУВАННЯ АВТОНОМНИМИ ЕЛЕКТРОСТАНЦІЯМИ

1.1. Структура та функції системи керування автономною електростанцією
1.2. Ієрархічна модель системи керування АЕЕС
1.3. Вибір топології мережі для реалізації функцій керування
1.4. Аналітичний огляд підходів до аналізу інформаційних потоків
1.5. Використання UML для моделювання цифрових систем керування
1.6. Використання автоматного підходу в проектуванні засобів автоматизації АЕЕС
1.7. Визначення проблеми та вимог до математичного апарату

Розділ 2. МЕТОДИКА АНАЛІЗУ ПРОДУКТИВНОСТІ МЕРЕЖІ ОБРОБКИ ІНФОРМАЦІЇ РОЗПОДІЛЕНОЇ СИСТЕМИ КЕРУВАННЯ АЕЕС

2.1. Вимоги до якості обслуговування
2.2. Класифікація інформаційних потоків з використанням нечіткої логіки
2.3. Формування матриці вимог до передачі даних
2.4. Пріоритетне обслуговування та зважені черги
2.5. Визначення параметрів мережі обробки інформації аналітичними засобами
2.6. Аналіз кореляційних характеристик інформаційних потоків
2.7. Стаціонарні характеристики системи масового обслуговування
2.8. Вплив зміни параметрів інформаційних потоків на ефективність роботи мережі

Розділ 3. МОДЕЛІ ТА АЛГОРИТМИ КЕРУВАННЯ ІНФОРМАЦІЙНИМИ ПОТОКАМИ

3.1. Мінімізація та упорядкування зв’язків по рівням завантаженності інформаційних каналів
3.2. Оптимальна стратегія доступу до каналу зв’язку
3.3. Мінімізація колізій інформаційних потоків в мережах розподілених систем керування
3.4. Алгоритм обробки інформаційних пакетів в маршрутизаторі
3.5. Рішення задачі оптимальної маршрутизації інформаційних потоків
    3.5.1. Алгоритм обробки змін маршрутизації
    3.5.2. Алгоритм генерації змін маршрутизації
    3.5.3. Процедура розрахунку метрики шляху
    3.5.4. Алгоритм розподілу пакетів по різних шляхах до вузла призначення
3.6. Мінімізація затримки передачі даних по мережі
3.6. Розробка протоколу мережевого рівня

3.7. Завадостійке кодування інформації з використанням SWITCH–технології
3.8. Модельні дослідження роботи систем керування автономними електроенергетичними установками
3.8.1. Моделювання мікропроцесорної системи керування АЕЕС
3.8.2. Аналіз впливу динаміки Ethernet-мережі на якість керування

Розділ 4. АВТОМАТИЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ РОЗПОДІЛЕНИХ СИСТЕМ КЕРУВАННЯ ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ UML

4.1. Статична UML – модель системи керування
4.2. Розробка UML – моделі інформаційних потоків
4.3. Використання UML-FB в проектуванні та моделюванні систем керування автономними електростанціями
4.4. Вимоги до автоматизованих робочих місць оператора та програмного забезпечення
4.5. Методика створення програмного забезпечення для систем автоматизації електроенергетичних установок
4.6. Статична модель інтерфейсу користувача
4.7. Динамічна модель та аналіз повідомлень
4.8. Розробка програмних засобів керування електростанцією

Розділ 5. МЕТОД СИНТЕЗУ МАТЕМАТИЧНИХ МОДЕЛЕЙ ЛОГІКО–ДИНАМІЧНИХ ПРОЦЕСІВ КОНТРОЛЮ Й КЕРУВАННЯ

5.1. Аналіз стану інформаційних технологій при синтезі математичних моделей логіко–динамічних процесів контролю й керування
5.2. Формування аналітичної форми запису функціональних структур з елементами інформаційних технологій
5.2.1. Особливість структури розрахункової формули в основах математичного апарату
5.2.2. Логіка формування аналітичного запису функціональних аналогових і функціональних цифрових структур
5.3. Синтез математичної моделі логіко–динамічного процесу перетворення сигналів в електроенергетиці
5.4. Приклад застосування математичних моделей

Перелік літератури

ВСТУП

Ефективність сучасних автономних електроенергетичних систем в значній мірі залежить від якості систем контролю і керування, збору і обробки інформації, систем прийняття рішень, основою яких є мікропроцесорні керуючі системи. Функціонування сучасних засобів автоматизації пов'язане з введенням, обробкою, зберіганням, видачею великої кількості інформації. Вони характеризуються паралельною і розподіленою обробкою інформації, роботою в реальному часі. Розвиток мікропроцесорних мереж змушує використовувати складніші способи керування інформаційними потоками. Необхідно контролювати такі параметри, як затримка передачі пакетів, коливання затримки, втрати пакетів в мережі. Основна тенденція в розвитку систем керування автономними електроенергетичними системами полягає в переході від централізованих систем до розподілених інтелектуальних систем, заснованих на мережевих технологіях. Підходи до синтезу засобів керування та автоматизації електроенергетичних систем та промислових виробництв в цілому описані в [77, 20, 70, 92, 99, 132].

Оскільки в процесі керування відбувається обмін інформацією між окремими блоками і агрегатами, то актуальними є питання розв’язання задачі маршрутизації інформаційних пакетів, упорядковування зв'язків по рівнях завантаженості каналів. Дослідженню інформаційних систем та потоків в обчислювальних мережах присвячена велика кількість робіт, в яких розглянуті проблеми керування у вузлах мережі [52, 78], досліджені процеси в швидкісних інформаційних системах [15, 17, 56, 57, 75, 76].

Потоки інформації в сучасній системі керування можуть мати різні вимоги до затримки, швидкості, надійності передачі. Якщо вимоги одних підсистем можуть бути невисокими, вимоги інших, що працюють в режимі реального часу, можуть бути критичними до швидкості передачі і затримок. У зв'язку з цим актуальним є завдання розділення потоків на різні класи обслуговування, і рішення задачі маршрутизації для кожного з класу потоків

з метою зменшення затримок при передачі пакетів і зменшення навантаження на мережу. Розв’язання цієї задачі вимагає залучення апарату нечіткої логіки. Використання апарату вейвлетів, який може бути використаний для рішення задач компресії даних з метою мінімізації об’єму потоків розглядається в роботах [27, 41].

У сучасних мережах обробки інформації розподілених систем керування помітна тенденція збільшення числа ресурсів, що розподіляються, кількості паралельно реалізованих процесів, зростання потужностей процесорів, числа каналів. У цих умовах досить важливим є розробка оптимальних методів розподілу ресурсів мережі й диспетчеризації процесів з погляду підвищення продуктивності мережі, що полягає в зменшенні числа конфліктів. Перспективним для керування автономними електроенергетичними системами є GRID-технології [137].

При побудові мереж велике практичне значення має рішення задачі оптимізації функціонування мереж з точки зору найбільш економічного розподілу потоку даних по каналах зв'язку. Є можливість знайти оптимальні шляхи проходження потоку інформації за різними критеріями – мінімізація часу доставки пакетів, вартості проходження пакета, середньої затримки, або мінімізація пропускної здатності. Аналіз якості функціонування та оптимізації характеристик інформаційних потоків для підвищення ефективності функціонування комп’ютерних мереж розглянуті в роботах [55, 58, 133, 134, 154].

Актуальною залишається задача розробки спеціалізованого протоколу, у якому була б врахована можливість збору даних про стан мережі й на основі цих даних виконувати оптимальну маршрутизацію. Все більші вимоги пред'являються до вірогідності повідомлень, які передаються. Це приводить до необхідності використовувати спеціальні засоби заходу щодо зниження кількості помилок.

При рішенні задач оптимізації інформаційних потоків необхідно виділити критерії оптимізації. До критеріїв оптимізації можна віднести

величину затримки передачі пакета між двома вузлами мережі, швидкість передачі, обсяг переданих даних, завантаженість інформаційного каналу, надійність передачі даних, питомувартість передачі пакета інформації.

Як правило, оптимізація одного параметра може привести до погіршення якості керування технічними засобами автоматизації АЕЕС. Наприклад, оптимізація часу передачі пакетів по мережі вимагає збільшення швидкості передачі й, як наслідок, застосування каналів із широкою смугою пропускання, а також швидкодіючих маршрутизаторів. Крім того, зростають економічні витрати. Разом з тим, коефіцієнт завантаженості каналу передачі при цьому буде невеликим. Зменшення обсягу переданих даних, у свою чергу, може привести до погіршення якості керування. Отже при рішенні задач оптимізації необхідно враховувати всі перераховані фактори.

Все більша увага приділяється проблемам керування та створення систем сбору та обробки інформації з використанням радіоканалу. Однак у такому випадку постає задача організації доступу до радіоканалу. При одночасній передачі даних від двох і більше вузлів виникає колізія й робота мережі стане неможливою. В якості пристроїв, що виконують передачу й прийом даних, використовуються пакетні радіомодеми. Особливість таких модемів полягає в тому, що можлива передача на різних частотах, які встановлюються програмно, тобто мережа володіє декількома каналами. Для таких мереж актуальне завдання знаходження оптимальної стратегії доступу до каналів.

У процесі створення систем автоматизації електроенергетичних установок на базі промислових контролерів розробники вирішують задачі вибору або власної розробки як апаратної частини системи (мікроконтролер, різноманітне периферійне устаткування, датчики, мережне устаткування), так і програмного забезпечення. Програмне забезпечення для автоматизованих систем, як правило представляє собою автоматизоване робоче місце. Комплексні програмні продукти створені фірмами, які спеціалізуються на розробці прикладного програмного забезпечення, є закритими, і це не

дозволяє адаптувати їх до реальної області застосування. Такі системи мають високу вартість. Введення їх в експлуатацію й супровід вимагають значного часу й додаткових засобів, а також залучення представників фірм-розроблювачів.

Відсутність необхідного програмного забезпечення спонукує розробляти інтелектуальні програми з потрібними властивостями самостійно, виходячи зі своїх уявлень про дану проблему. Однак і в цьому питанні відсутні єдині думки про вимоги й властивості таких систем, зокрема, про способи їхньої реалізації. Для створення програмного забезпечення мікроконтролерів широко використовується автоматний підхід, або SWITCH-технологія [144–147].

На даний момент актуальні питання розробки ефективних засобів для сумісного проектування програмного і апаратного забезпечення. Перспективним напрямом в розробці систем автоматики є використання сучасних технологій інженерії програмних систем, до яких відноситься об'єктно-орієнтований аналіз і проектування на основі уніфікованої мови моделювання.

На основі викладеного можна сформулювати основні наукові задачі, що потребують вирішення:

- визначення структури системи керування автономною електроенергетичною установкою та склад вирішуваних нею завдань залежно від умов функціонування;
- формулювання й теоретичне обґрунтування нових науково-технічних положень організації процесу керування інформаційними потоками при проектуванні розподілених систем керування;
- оцінка продуктивності мережі системи керування реального часу з метою пошуку шляхів підвищення її ефективності;
- оцінка інтенсивності сумарного потоку й побудова автокореляційної функції з метою аналізу впливу інтервалу часу недоступності, протягом якого пакети не обслуговуються;

– формулювання критерію оптимізації для пошуку оптимального розподілу потоків по різних каналах мережі;
– розробка алгоритму обробки пакетів у маршрутизаторі, що виконує пошук оптимального маршруту;
– розробка алгоритму диспетчеризації інформаційних потоків з метою підвищення якості функціонування мережі шляхом зменшення числа колізій;
– визначення характеристик обчислювальних мереж аналітичними засобами;
– дослідження аналітичної моделі мережі з метою визначення оптимальних значень продуктивності та пропускної здатності при зміні таких параметрів, як довжина кадрів, завантаження каналу, інтенсивність повторення, інтенсивність генерування нових кадрів;
– розробка цифрового скінченного автомата для завадостійкого кодування повідомлень, переданих по мережі, а також алгоритму й програми для кодування повідомлень з використанням SWITCH-технології;
– моделювання електроенергетичної системи з використанням уніфікованої мови моделювання UML;
– розробка метода синтезу математичних моделей логіко–динамічних процесів контролю й керування, який дозволить перейти від структурних схем до математичних моделей для подальшого аналізу та оптимізації.

У першому розділі монографії розглядаються теоретичні аспекти розробки методики та алгоритмів керування інформаційними потоками в мережах. Розглянуто принципи побудови розподілених систем керування електроенергетичними системами; визначені рівні ієрархії системи керування; сформульовані вимоги щодо вибору математичного апарату.

У другому розділі дослідження характеристик мережі обробки інформації розподіленої системи керування. Розглянуті питання класифікації

інформаційних потоків в мережі розподіленої системи керування з використанням нечіткої логіки. Досліджено вплив зміни параметрів інформаційних потоків на ефективність роботи мережі та сформовані вимоги до покращення якості функціонування мережі.

У третьому розділі розглянуті алгоритми керування інформаційними потоками мережі керування. Ключове завдання керування потоками включає мінімізацію втрат пакетів і затримок, оптимізацію пропускної спроможності. Розроблена методика вирішення оптимізаційних задач з використанням теорії масового обслуговування та апарату нечіткої логіки при керуванні інформаційними потоками.

Четвертий розділ присвячений розробці моделі системи керування з використанням уніфікованої мови моделювання UML та UML-FB, з метою синтезу програмного коду для керуючих мікроконтролерів. Запропонована методика розробки програмного забезпечення для автоматизованих систем керування автономними електроенергетичними установками.

В п’ятому розділі розглядається розробка нового методу формалізованого запису різних логіко–динамічних процесів і сигналів у системах керування й контролю. Виконано синтез математичної моделі логіко–динамічного процесу перетворення сигналів в електроенергетиці.

ПЕРЕЛІК СКОРОЧЕНЬ

АД – асинхронний двигун

АЕЕС – автономна електроенергетична система

АСУ ТП – автоматизована система управління технологічним процесом

ВВС – взаємодія відкритих систем

ГА – генераторний агрегат

ГРЩ – головний розподілюючий щит

ДГ – дизель-генератор

ЕОМ – електронно-обчислювальна машина

ЕРП – електророзподільні пристрої

МП – мікропроцесор

ОК – об'єкт керування

ПЗ – програмне забезпечення

ПІД – пропорційно-інтегрально-диференціальний регулятор

СГ – синхронний генератор

СЕЕС – суднова електроенергетична система

СК – система керування

СМО – система масового обслуговування

ССГ – система синхронізації генераторів

ФБ – функціональний блок

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ТА ХАРАКТЕРИСТИКИ СИСТЕМИ КЕРУВАННЯ АВТОНОМНИМИ ЕЛЕКТРОСТАНЦІЯМИ

1.1. Структура та функції системи керування автономною електростанцією

Автономна електроенергетична система (АЕЕС) – це складна багатоелементна структура [10], що включає в себе агрегати генерування електроенергії, вузли її розподілу, елементи контролю показників якості, забезпечення надійності, захисту.

На рис. 1.1. наведена узагальнена функціональна схема силової частини АЕЕС, що може розглядатися з точки зору енергетичних показників розподілу електричної енергії. Вона містить у собі генераторні агрегати ($Г_1$ – $Г_3$), головний розподільний щит (ГРЩ), ряд проміжних електророзподільних пристроїв (ЕРП), споживачі у вигляді асинхронних двигунів (АД).

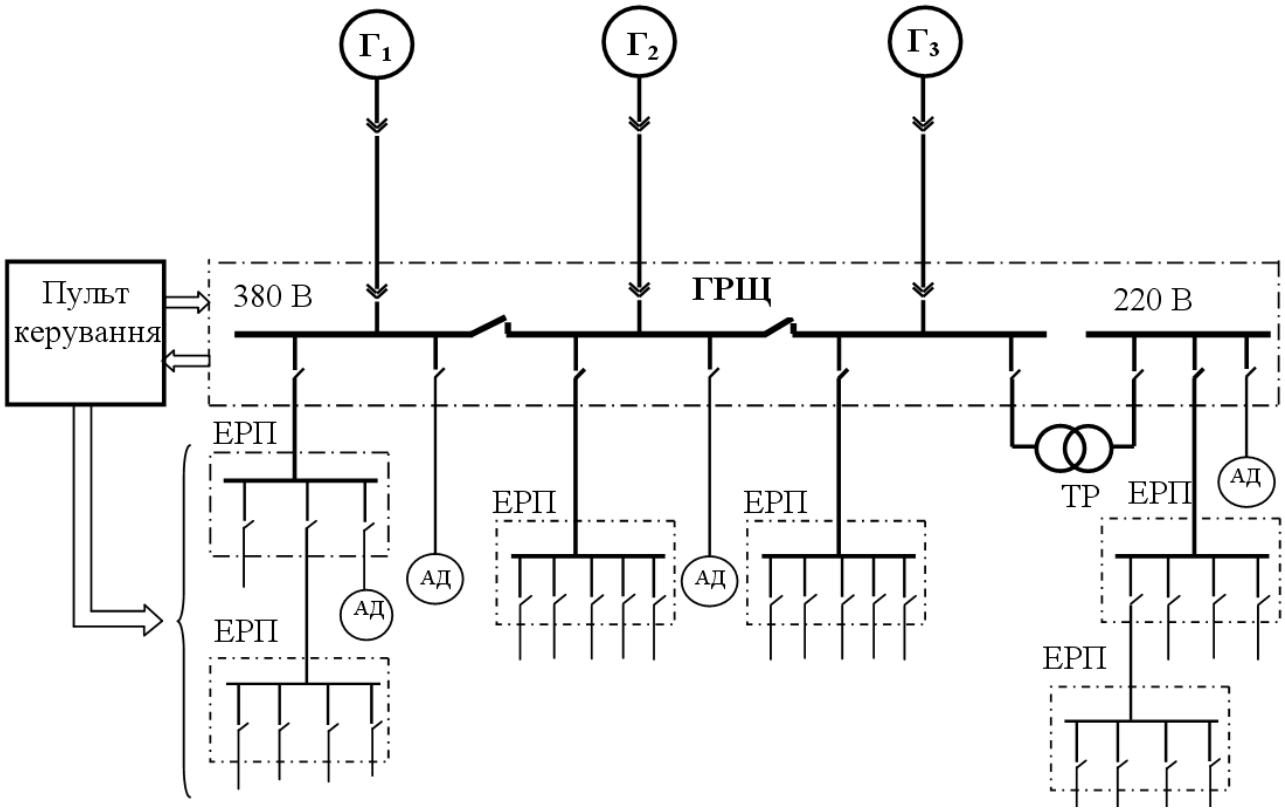


Рис. 1.1. Схема силової частини АЕЕС

Якісні показники електроенергії й роботи всієї системи забезпечуються системою керування й автоматики, елементи якої більш детально зображені на рис. 1.2.

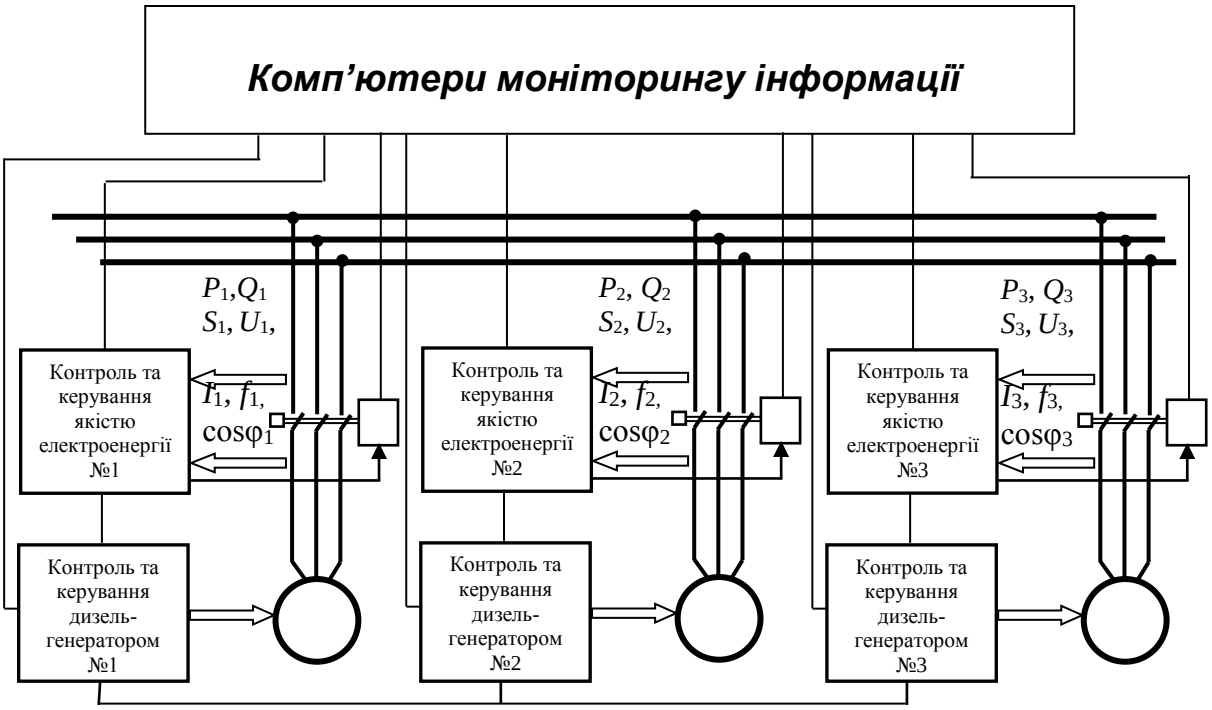


Рис. 1.2. Схема керування й автоматики.

Таким чином, АЕЕС може розглядатися як система, що складається із двох підсистем – енергетичної та інформаційної. Елементами енергетичної підсистеми є генеруючі агрегати, комутаційна апаратура, захисні пристрої, кабельні траси [20]. Основними властивостями енергетичної підсистеми – це здатність забезпечувати споживачів електроенергією із заданою надійністю й ефективністю.

Введемо позначення елемента системи як M_k, $k=\overline{1,l}$, а їхню сукупність $\{M\}$. Тоді система описується множиною:

$$M=\{M_1, M_2, ...M_k\}.$$

Зв'язку між елементами позначимо через L_{ij} від i – го ($i=\overline{1,n}$) джерела до j – му ($j=\overline{1,m}$) споживачу, $\{L(M)\}$ – сукупність зв'язків у системі, а $F(M,L)$ – функція системи, в цьому випадку це багатомірна функція логічних зв'язків, що описуються простими змінними:

$$L \in \{0,1\}.$$

Тоді система може бути описана наступним чином:

$$C : \left[\{M\}\ \{x(M)\}\ \{F(M,L)\}\right],$$

де $x\{M\}$ – впливи на систему;

C – виводи системи

Оскільки АЕЕС має велику кількість однотипних елементів й однотипних зв'язків, то можемо вважати, що вона відноситься до великої системи [66].

Якщо розглядати в системі інформаційні потоки, то це надає право розглядати систему складною, оскільки вона має різнорідні зв'язки між елементами.

Всі сучасні АЕЕС оснащуються системами автоматизації технічних засобів. Визначилася загальна тенденція автоматизації на основі мікропроцесорних обчислювальних засобів [5, 73, 87]. Система керування виконує збір, перетворення, передачу й відображення інформації, формування, передачу й реалізацію керуючих впливів на керовані об'єкти. Система керування виконує функції автоматичного контролю, регулювання, керування, захисту, діагностування й прогнозування стану технічних засобів, керування якими повинне забезпечуватися автоматично або дистанційно.

Система керування АЕЕС мають у своєму составі наступні функціональні пристрої [66]:

– синхронізації;

– розподілу навантаження між паралельно працюючими генераторами;

– розвантаження генераторів;

– включення резервного генераторного агрегату;

– перемикання живлення споживачів;

– контролю ізоляції;

– захисту від обриву фази й зниження напруги;

– звукової й світлової сигналізації.

Кожен функціональний пристрій виконаний у вигляді окремого модуля. Модульний принцип побудови системи передбачає розробку й використання в системі обмеженого ряду типових уніфікованих блоків і дозволяє вирішити проблему уніфікації. Необхідність знання інформаційної моделі полягає в тому, що при синтезі енергетичної підсистеми конструкторові необхідно закладати в проект і передбачати при замовленні устаткування ряд додаткових пристроїв, контактів, допоміжних реле, датчиків струму, напруги, показників якості електроенергії.

На рис. 1.3. наведена спрощена граф–схема підсистеми автоматики й керування для випадку АЕЕС з двома генераторами, з рядом спрощуючих передумов.

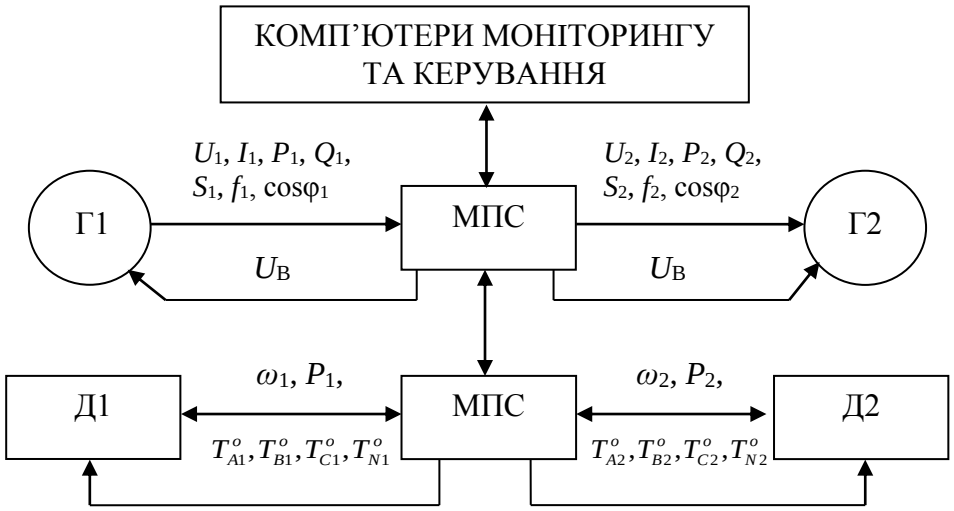


Рис. 1.3. Схема автоматики й керування

Для забезпечення якості генерування електроенергії система керування АЕЕС повинна виконувати наступні функції [5, 10, 66]:

- Автоматичний пуск і включення резервного ДГ при завантаженні працюючих ГА вище 80–90% їхньої номінальної потужності;
- Автоматичний пуск і включення резервного ДГ при знеструмленні шин ГРЩ;
- Автоматичне включення резервного ДГ при падінні тиску масла в працюючому ГА;

- Автоматичне відключення резервного ДГ при розвантаженні працюючих ГА до 30% їхньої номінальної потужності;
- Автоматичне включення резервного ДГ при падінні частоти в мережі й частоти обертання працюючого ГА нижче припустимих меж;
- Автоматичне включення іншого ГА, якщо у резервного ДГ не піднімається тиск масла;
- Автоматична й дистанційна точна синхронізація генераторів із шинами ГРЩ і між собою;
- Автоматичне регулювання частоти обертів первинного двигуна й пропорційний розподіл активного навантаження між паралельно працюючими генераторами;
- Автоматичне регулювання напруги й пропорційний розподіл реактивного навантаження між паралельно працюючими генераторами (виконується системою збудження);
- Автоматичний захист генераторів від перевантажень;
- Дистанційні автоматизовані пуск і зупинка (із щита керування системи) первинних двигунів генераторів і сигналізація їхньої роботи;
- Автоматична перевірка резерву потужності при пуску потужних споживачів електроенергії;
- Автоматичний безперервний контроль опору ізоляції на секціях шин ГРЩ і на шинах аварійного ДГ;
- Автоматична світлова сигналізація про розвантаження паралельно працюючих генераторів до 30–35% їхньої номінальної потужності;
- Аварійно–попереджувальна сигналізація про вихід електричних параметрів за припустимі межі;
- Автоматична обробка аварійно–попереджувальної сигналізації, видача узагальненої інформації в систему централізованого контролю;
- Дистанційний контроль параметрів електроенергії;
- Автоматичний контроль параметрів електроенергії;

- Сигналізація про втрату живлення в системі керування АЕЕС;
- Діагностування устаткування АЕЕС та її системи керування;
- Автоматичний вивід параметрів електроенергії на засоби відображення інформації;
- Автоматична індивідуальна й узагальнена сигналізація про вибіги параметрів і несправності в системі керування АЕЕС;
- Використання в керуванні результатів діагностування встаткування АЕЕС;
- Автоматичний вивід з паралельної роботи ГА із плавним перекладом активного навантаження.
- Автоматична реєстрація складу працюючого устаткування, вибігу параметрів і несправностей устаткування АЕЕС та апаратури її системи керування;
- Автоматична видача рекомендацій оператору про способи усунення несправностей у системі керування (система підтримки прийняття рішень).

1.2. Ієрархічна модель системи керування АЕЕС

Системи керування АЕЕС будуються по ієрархічному принципу. У випадку відмови елементів верхнього рівня забезпечується працездатність АЕЕС за допомогою елементів нижніх рівнів системи.

На нижньому (нульовому) рівні ієрархії перебувають керовані елементи АЕЕС із датчиками й виконавчими органами. Цей рівень призначений для керування виконавчими пристроями. Тут же реалізовані різні види блокувань. При виникненні несправностей інформація передається на вищестоящий рівень.

Перший рівень ієрархії представляє собою сукупність локальних засобів автоматизації, що забезпечують стабілізацію напруги й частоти генераторів, їхній захист від перевантаження й к.з., від зворотної потужності. Вони

дозволяють формувати керуючі впливи з метою збереження працездатності АЕЕС при відмові її елементів.

Другий рівень ієрархії представляє собою рівень локальних систем керування, включаючи системи керування генераторними агрегатами, на якому виконуються наступні функції: керування приводними двигунами генераторів за сигналом від автоматичного пристрою або оператора, контроль і захист генераторів, синхронізація з наступним розподілом навантаження, регулювання частоти. Також на цьому рівні перебувають пристрої місцевого керування генераторними агрегатами.

Третій рівень ієрархії представляє собою рівень систем керування функціональними комплексами. На цьому рівні виконуються наступні функції: контроль навантаження працюючих генераторів, контроль резерву потужності для забезпечення пуску потужних споживачів енергії, вибір послідовності пуску резервних генераторних агрегатів, контроль технічного стану АЕЕС, захист від струмів короткого замикання.

Четвертий рівень ієрархії – рівень комплексної системи керування технічними засобами, на якому забезпечується керування електростанцією у цілому оператором. На цьому рівні виконується завдання режимів використання АЕЕС залежно від режиму експлуатації, від якого залежить склад елементів АЕЕС.

Час, необхідний для виконання операцій керування, на різних рівнях ієрархії неоднаковий. Найбільш швидко виконуються операції 1–го рівня за допомогою систем автоматичного регулювання, контролю й захисту окремих елементів. Операції 2–го рівня вимагають більше тривалого часу, що визначається видом процесу керування приводними двигунами, порушенням генераторів. На цьому рівні є ряд програмних автоматів, які видають сигнал у систему керування АЕЕС про їхню готовність. Операції 3–го рівня вимагають швидкості керуючого пристрою. При виконанні операцій зміни структури АЕЕС потрібен проміжок часу, необхідний для виконання операцій 2–м рівнем. Час для виконання операцій 4–го рівня визначається

припустимими швидкостями зміни режимів експлуатації енергетичної установки.

Побудова складних систем керування повинна виконуватися із застосуванням досить простих структур, які також підтримують прості в реалізації програмно–апаратні засоби. Це стає можливим на основі побудови багаторівневої апаратної системи, що має у своєму складі обчислювальні прилади й модулі, спроектовані з використанням контролерів.

Мережа передачі даних може представляти собою локальну або загальну обчислювальну мережу, організовану по магістральному, радіальному, комбінованому або іншому принципу. Інтерфейс між вузлами системи на різних ієрархічних рівнях може бути різним. На рис. 1.4 представлена структура електроенергетичної системи, що складається із трьох дизель–генераторів.

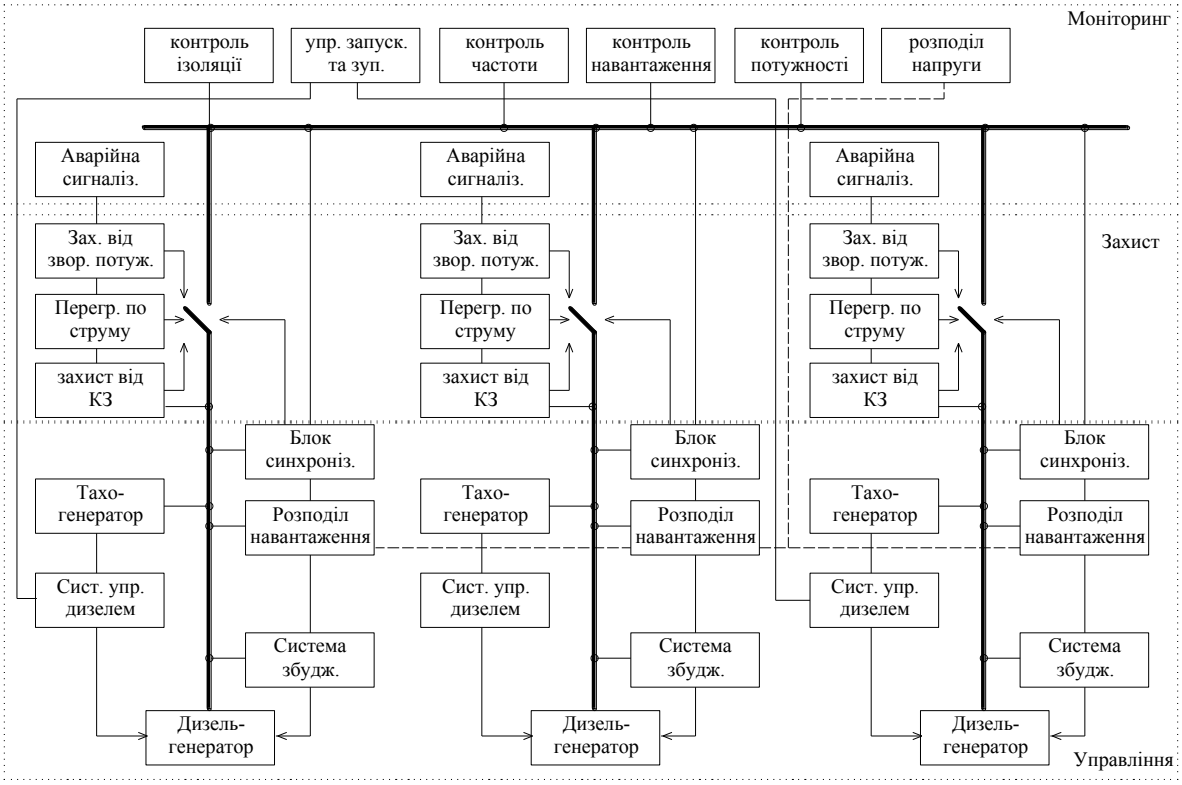


Рис. 1.4. Загальна структура енергетичної системи

Система керування представляє собою мережу, вузли якої зв'язані каналами передачі даних. Вибір топології з'єднань вузлів мережі

визначається структурою АЕЕС, вимогами надійності, припустимим часом очікування пакета даних від одного рівня до іншого. У загальному випадку, конфігурація взаємозв'язків мережі може залишатися незмінною в процесі експлуатації (статична), або ж оперативно змінюватися за допомогою програмних засобів (динамічна).

Використання динамічної топології дозволяє створити резервні шляхи, завантаженість яких низька, і використати ці шляхи для передачі команд керування. У мережах з комутацією з'єднань формується канал від вузла–джерела до вузла–приймача, що зберігається поки весь пакет, що доставляється, не досягне пункту призначення. Час існування такого каналу визначається довжиною пакета. З іншого боку, у мережі може використовуватися стратегія комутації пакетів. У цьому випадку передбачається, що повідомлення самостійно знаходить свій шлях до місця призначення, який щоразу може бути іншим. Пакет послідовно проходить через вузли мережі. Кожний вузол запам'ятовує пакет у своєму буфері, аналізує його й ухвалює рішення щодо подальших дій. Залежно від завантаженості мережі приймається рішення про можливості негайного пересилання пакета до наступного вузла й про подальший маршрут проходження пакета на шляху до мети. Якщо всі можливі канали зайняті, то в буфері вузла формується черга пакетів, що зменшується в міру звільнення ліній зв'язку між вузлами. Таке явище неприйнятно для системи керування, оскільки час очікування передачі пакета визначається завантаженістю каналів передачі, що у свою чергу визначається режимом роботи АЕЕС. Крім того, якщо черга пакетів перевищує розміри буфера, відбувається втрата даних, що також неприйнятно в умовах аварійної ситуації. Рішенням проблеми може служити збільшення кількості каналів зв'язку. При цьому пропорційно зменшується навантаження на кожний з них. Крім того, можна створити окремий канал зв'язку, що не використовується при нормальному режимі роботи АЕЕС. При виникненні аварійної ситуації резервний канал зв'язку дозволить з максимальною швидкістю передати відповідні команди

вузлам системи. Кожний з вузлів мережі може бути як термінальним, так і комутатором (але не одночасно). Між термінальними вузлами повідомлення відправляються напряму. Збільшення числа комутаторів приводить до збільшення затримки. Оскільки система керування виконує також задачі прогнозування, навіть при відсутності аварійного режиму, при якому може бути використаний резервний канал, затримка передачі команди повинна бути мінімальною.

1.3. Вибір топології мережі для реалізації функцій керування

Для забезпечення високої надійності системи керування мережа повинна мати високу зв’язність. При відмові окремих компонентів загальна працездатність системи керування повинна зберігатися. Вибір топології мережі багато в чому визначається наявністю додаткових зв'язків між вузлами. Так, лінійна топологія, що представляє собою одновимірний масив, об'єднаний у ланцюжок, має ряд недоліків: час пересилання даних між двома вузлами залежить від відстані між ними, а відмова одного з вузлів приводить до неможливості посилки повідомлення. Використання кільцевої топології також недоцільне, оскільки вона не має можливість нарощування (додавання нового вузла вимагає демонтажу мережі), навантаження на канал передачі між різними вузлами буде різною, відсутні резервні зв'язки [18, 26, 29, 100, 110, 174].

Електрична енергія в автономних електростанціях, у тому числі й на судах, виробляється синхронними генераторами. При виборі структури електроенергетичної системи враховуються такі фактори, як розмір корабля, його призначення, потужність навантаження. На контейнеровозах основним споживачем енергії є оснащення по транспортуванню й фіксації контейнерів. На танкерах основним споживачем електроенергії будуть насоси й компресори. На пасажирських судах більше навантаження створюють системи вентиляції, освітлення, а також підрулюючі пристрої при

маневруванні в портах. На судах з електрорухом двигуни є головним споживачем енергії.

При виборі структури електроенергетичної системи враховуються такі положення:

– генерування максимальної потужності;

– тип і потужність навантаження, обмеження максимальної потужності споживачів;

– кількість і потужність генераторів;

– частота й величина напруги;

– розрахунок струмів короткого замикання.

Аварійні дизель–генератори використовуються для забезпечення живлення у випадку виходу з ладу основних генераторів із причин поломки або короткого замикання (спрацьовування захисту). Аварійний дизель–генератор не повинен залежати від інших силових агрегатів у машинному відділенні і має власні системи автоматичного запуску, системи охолодження й підтримки в гарячому резерві. АДГ повинен забезпечувати електроенергією такі споживачі, як аварійне освітлення, системи навігації й зв'язку, систему протипожежної безпеки (насоси й компресори), приводи дверей і ліфти.

При зникненні напруги на шинах ГРЩ автоматично повинен виконуватися пуск аварійного дизель–генератора й підключення до шин ГРЩ. Час від моменту зникнення напруги на шинах ГРЩ і до моменту включення в роботу аварійного дизель–генератора й забезпечення відповідальних споживачів живленням не повинно перевищувати 45 секунд.

Електростанція повинна містити джерело безперебійного живлення, що забезпечує живленням відповідальних споживачів, поки не буде зроблений запуск аварійного дизель–генератора. ДБЖ повинно забезпечувати роботу споживачів не менш 30 хвилин.

На рис. 1.5 представлена схема електроенергетичної системи судна. Напруга становить 690 В, частота 50 Гц.

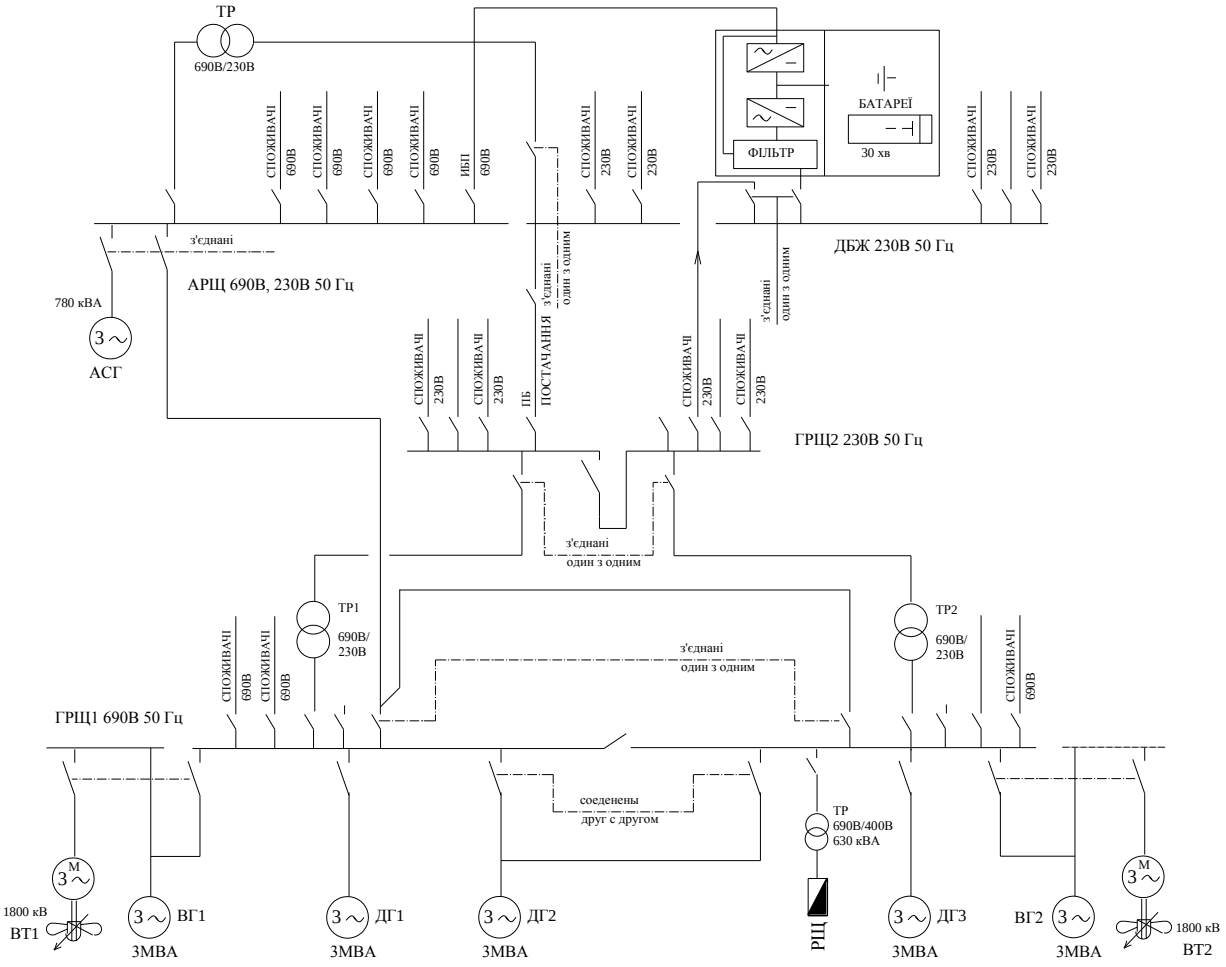


Рис. 1.5. Схема електроенергетичної системи

Через свою гнучкість і універсальність широке застосування знайшли дизель–генераторні установки. Чотири або шість дизель–генераторів підключаються до шин головного розподільного щита (ГРЩ). Генератори вмикаються симетрично щодо автоматичного вимикача шини. Крім того, кожна з генераторних секцій має своє навантаження. З використанням трансформаторів забезпечується живлення споживачів з напругами 400 В и 220 В.

Крім основних дизель–генераторів, електростанція має у своєму складі валогенератори з приводом від головних двигунів. На одних судах валогенератори можуть використовуватись для живлення систем, пов'язаних з маневруванням судна. В інших випадках вони можуть бути підключені до основної енергосистеми судна. Тому потрібно враховувати додаткові

з'єднання для запобігання перевантажень і ушкодження мережі. У випадку підключення валогенератора до основної мережі, повинні бути обрані фіксовані оберти головного двигуна.

Наявність навантажень різної потужності, а також широкий спектр можливих операцій, вироблених на судні часто приводить до того, що встановлюються кілька генераторів різної потужності. Таким чином, у заданих режимах роботи електростанції можливо вибрати оптимальну структуру й мінімізувати фінансові витрати на генерування електроенергії. Однак при такій структурі електростанції необхідна наявність систем розподілу активної/реактивної потужності, які працюють при паралельному включенні генераторів, а також системи захисту.

Кожен з дизелів оснащений системою керування обертами, як правило електронної, але іноді й механічної. Електронна система дозволяє більш стабільно підтримувати оберти дизеля, і як наслідок, частоту напруги генераторів. Крім того, використання електронних систем керування обертами дозволяє виконувати автоматичний розподіл активної потужності між генераторами при їхній паралельній роботі.

Електростанція має в складі блок контролю й керування споживанням потужності, основною функцією якого є контроль за потужністю, яка генерується й забезпечення резерву потужності у випадку включення потужних споживачів електроенергії. Залежно від типу й потужності судна, цей блок може бути встановлений окремо або інтегрований у розподілену систему керування.

На рис. 1.6 представлена схема електроенергетичної системи танкера з напругою 6,6 кВ.

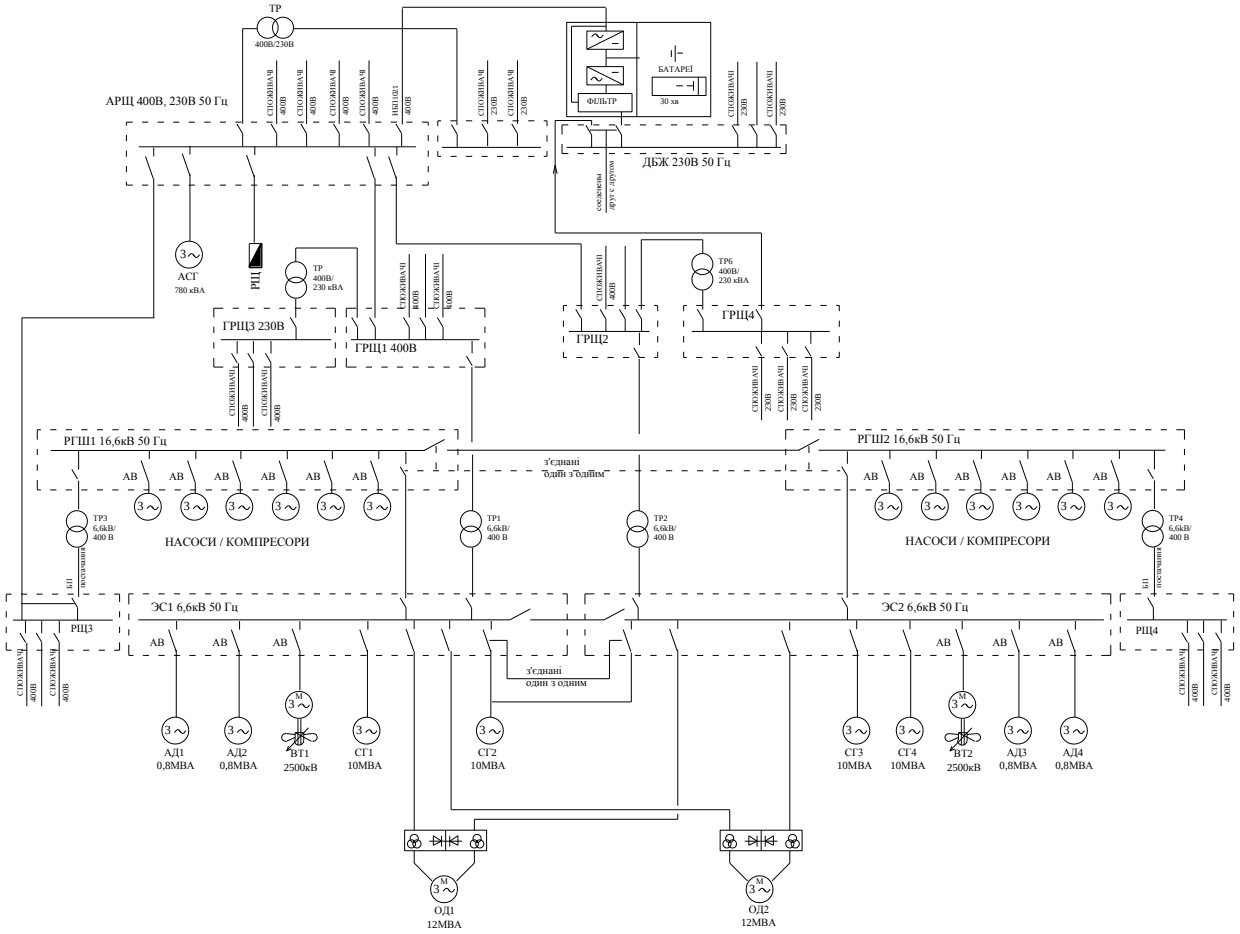


Рис. 1.6. Схема електроенергетичної системи танкера

Основні функції системи керування полягають у наступному:

– автоматичний запуск і зупинка дизель–генераторних агрегатів залежно від режиму роботи електростанції;

– контроль за станом дизельних агрегатів;

– автоматична точна синхронізація генераторів і вивід з паралельної роботи;

– розподіл активної й реактивної потужностей;

– автоматичний пуск/зупинка дизель–генераторів залежно від споживаної потужності й наявності необхідного резерву;

– контроль за збільшенням споживаної потужності;

– захист від коротких замикань;

– забезпечення резерву потужності для пуску потужних споживачів;

– контроль частоти напруги;

– робота з певної програми залежно від режиму роботи;

– відбір енергії від валогенераторів.

Функція запуску й зупинки дизель–генераторів може виконуватися автоматично або вручну. Автоматична синхронізація виконується спеціальною системою синхронізації, що контролює параметри працюючого генератора й генератора, який підключають, і формує сигнал на замикання автоматичного вимикача. Час від моменту появи сигналу на запуск дизель–генератора до моменту підключення його на паралельну роботу звичайно не перевищує 45 секунд.

Система контролю за резервом потужності забезпечує запуск і зупинку дизель–генераторів залежно від навантаження на електростанцію. У випадку, якщо відсутній резерв потужності для запуску потужного споживача, а також якщо навантаження на генератори перевищує їхнє номінальне значення або струм генератора вище заданої межі протягом тривалого проміжку часу, виконується запуск дизель–генератора.

Якщо ж енергії, яка генерується, цілком достатньо для включення потужного навантаження, навантаження генераторів нижче деякого значення, то виконується автоматичний вивід генератора з паралельної роботи й зупинка дизеля.

У випадку зникнення напруги на шинах ГРЩ виконується автоматичний пуск всіх доступних дизель–генераторних агрегатів. До шин ГРЩ підключається генератор, що першим запустився, і на шинах якого з'явилася напруга. Потім по черзі підключаються на паралельну роботу інші генератори в міру їхньої готовності.

Перспективними напрямками в розвитку інформаційного забезпечення автономних електроенергетичних систем можна вважати:

– побудову систем моніторингу за станом об'єктів автономної електроенергетики з метою вибору оптимальних режимів роботи;

– використання систем контролю й забезпечення живучості суден;

– використання систем з оптичними каналами зв'язку, що забезпечують високу пропускну здатність;

– використання систем бездротової передачі даних;

– використання стандартних протоколів передачі даних і відкритих архітектур.

Система керування АЕЕС розглядається як сукупність мікропроцесорних систем, розосереджених по виробничих приміщеннях і об'єднаних по магістральній, кільцевій або іншій ознаці, яка забезпечує централізацію контролю (збір, обробку, подання інформації персоналу на постах керування) про технічний стан устаткування і якість протікання технологічних процесів, керування якими повинне забезпечуватися автоматично й дистанційно. Таким чином, по топології й організації керування система є розподіленою.

Структура ліній комутації між вузлами обчислювальної системи (топологія мережі передачі даних) визначається, як правило, з урахуванням можливостей ефективної технічної реалізації. Важливу роль також при виборі структури мережі грає й аналіз інтенсивності інформаційних потоків при рішенні завдань керування.

До числа типових топологій зазвичай відносять наступні схеми комунікації:

– повний граф (completely–connected graph або clique) – система, у якій між будь–якою парою процесорів існує пряма лінія зв'язку. Така топологія забезпечує мінімальні витрати при передачі даних, однак складно реалізується при великій кількості процесорів;

– лінійка (linear array або farm) – система, у якій всі процесори перенумеровані по порядку й кожний процесор, крім першого і останнього, має лінії зв'язку тільки із двома сусідніми (з попереднім і наступним) процесорами. Така схема, з одного боку, просто реалізується, з іншого боку, відповідає структурі передачі даних при рішенні багатьох обчислювальних задач (наприклад, при організації конвеєрних обчислень);

– кільце (ring) – дана топологія отримується із лінійки процесорів з'єднанням першого й останнього процесорів лінійки;

– зірка (star) – система, у якій всі процесори мають лінії зв'язку з деяким керуючим процесором. Дана топологія є ефективною, наприклад, при організації централізованих схем паралельних обчислень;

– гратки (mesh) – система, у якій граф ліній зв'язку утворює прямокутну сітку (звичайно двох–або тривимірну). Подібна топологія може бути досить просто реалізована й, крім того, ефективно використана при паралельному виконанні багатьох чисельних алгоритмів (наприклад, при реалізації методів аналізу математичних моделей, описуваних диференціальними рівняннями в частинних похідних);

– гіперкуб (hypercube) – дана топологія являє собою окремий випадок структури гратки, коли по кожній розмірності сітки є тільки два процесори (тобто гіперкуб містить $2N$ процесорів при розмірності N). Такий варіант організації мережі передачі даних досить широко розповсюджений на практиці.

В якості основних характеристик топології мережі передачі даних найбільш широко використовується наступний ряд показників:

діаметр – показник, що визначається як максимальна відстань між двома процесорами мережі (під відстанню звичайно розуміється величина найкоротшого шляху між процесорами). Ця величина може характеризувати максимально необхідний час для передачі даних між вузлами, оскільки час передачі зазвичай прямо пропорційний довжині шляху;

з'вязність (connectivity) – показник, що характеризує наявність різних маршрутів передачі даних між процесорами мережі. Конкретний вид даного показника може бути визначений, наприклад, як мінімальна кількість дуг, яку треба видалити для поділу мережі передачі даних на дві незв'язні області;

ширина бінарного розподілу (bisection width) – показник, обумовлений як мінімальна кількість дуг, яку треба видалити для поділу мережі передачі даних на дві незв'язні області однакового розміру;

вартість – показник, що може бути визначений, наприклад, як загальна кількість ліній передачі даних системі.

Для порівняння в табл. 1.1 приводяться значення перерахованих показників для різних топологій мережі передачі даних.

Таблиця 1.1. Характеристики топологій мережі передачі даних

Топологія	Діаметр	Ширина	Зв'язність бісекції	Вартість
Повний граф	1	$P^2/4$	p–1	$p(p-1)/2$
Зірка	2	1	1	p–1
Повне двійкове дерево	$2\log((p+1)/2)$	1	1	p–1
Лінійка	p–1	1	1	p–1
Кільце	$[p/2]$	2	2	p
Гратка N=2	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
Гратка–тор N=2	$2[\sqrt{p}/2]$	$2\sqrt{p}/2$	4	$2p$
Гіперкуб	$\log p$	$p/2$	$\log p$	$(p \log p)/2$

де p – кількість вузлів.

Оскільки система керування АЕЕС будується по ієрархічному принципу, одним з варіантів топології може бути використана деревоподібна топологія [174].

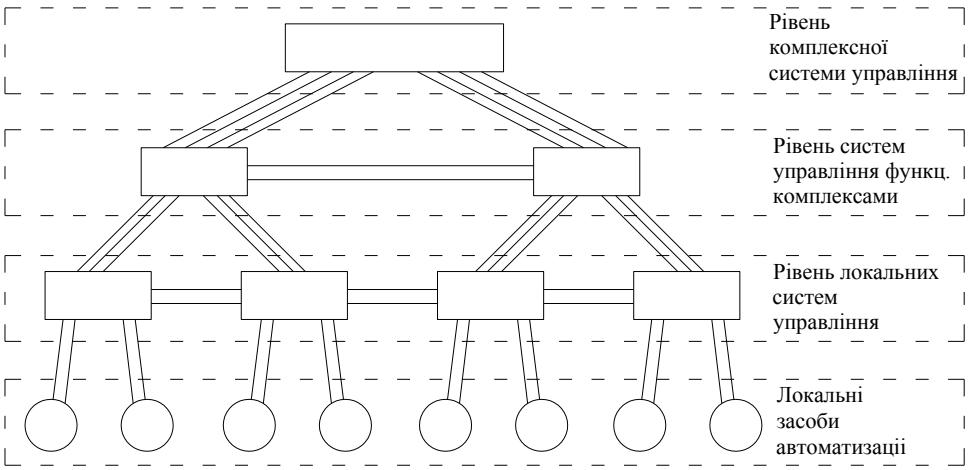


Рис. 1.7. Деревоподібна топологія: «товсте дерево»

Стандартне двійкове дерево має той недолік, що в міру просування від нижніх рівнів до верхнього значно збільшується навантаження на

інформаційні канали [174], що приводить до збільшення часу очікування. Повідомлення повинне проходити через один або кілька проміжних ланок. На більш високих рівнях мережі ймовірність затору через недостатню пропускну здатність лінії вища. Тому доцільніше використати «товсте дерево». Ідея товстого дерева полягає в збільшенні пропускної здатності комутаційних ліній на прикореневих рівнях мережі. Із цією метою на верхніх рівнях мережі батьківські й дочірні вузли зв'язують не одним, а декількома каналами, причому, чим рівень вище, тим більше каналів. Розмір мережі визначається кількістю вузлів мережі й залежить від рівня автоматизації енергосистеми. Діаметр мережі – мінімальний шлях, по якому проходить повідомлення між двома найбільш віддаленими один від одного вузлами, визначається виразом:

$$D_1=2(h-1),$$

де h – висота дерева.

На рис. 1.7 представлена топологія мережі «товсте дерево», для якого діаметр дорівнює:

$$D_2=2(h-1)-1.$$

Але це окремий випадок. Це обумовлено введенням додаткових зв'язків між вузлами одного рівня. Однак, при збільшенні дерева завширшки, тобто збільшення кількості вузлів на кожному з рівнів, діаметр мережі визначається по виразу D_1. Порядок вузла – сума числа вхідних і вихідних каналів вузла, у розглянутій топології міняється від 1 до 5 при переході від нижнього рівня до верхніх рівнів. Підвищення порядку вузлів дозволяє забезпечити менший діаметр мережі. Однак, оскільки порядок вузлів міняється, це ускладнює реалізацію системи по модульному принципу. Введення додаткових з'єднань між вузлами одного рівня дозволило збільшити зв’язність мережі. Цей параметр можна визначити як число вузлів або ліній зв'язку, які повинні вийти з ладу, щоб мережа розпалася на дві непересічні мережі. Так, для топології «класичного товстого дерева» цей параметр дорівнює одному, якщо розглядати нижні рівні, і двом при розгляді верхніх рівнів. При введенні

додаткових зв'язків зв’язність мережі збільшується до трьох. Затримка передачі повідомлення залежить від маршруту, по якому воно буде передаватися.

На рис. 1.7 це відображено у вигляді множинних ліній між вузлами верхніх рівнів. Нижній рівень містить сукупність локальних засобів автоматизації, які управляють об'єктом. При відмові якого–небудь із вузлів цього рівня, виконується автоматичне включення резерву.

1.4. Аналітичний огляд підходів до аналізу інформаційних потоків

Для знаходження характеристик мереж розподілених систем керування застосовуються два походи [6]: експериментальний й аналітичний. При використанні експериментального походу необхідні характеристики системи визначаються із результатів досвіду. Існуючі системи, що моделюють функціонування АЕЕС, розробляються з метою їхнього аналізу й синтезу.

При аналізі задаються наступні вихідні дані:

1. Структура мережі, число каналів та їхня пропускна здатність;
2. Показники надійності елементів;
3. Характеристика інформації;
4. Склад та основні характеристики технічних засобів, що використовуються у мережі;
5. Алгоритми маршрутизації, спосіб комутації і т.п.
6. Вартісні параметри мережі.

Завдання аналізу мережі полягає в розрахунку параметрів мережі в цілому, або окремих її фрагментах, з метою визначення ступеня відповідності характеристик мережі заданим нормативам.

При синтезі мережі вихідні дані:

1. Структура й характеристики первинної мережі.
2. Множина вузлів, між якими повинен бути організований обмін повідомленнями різних категорій терміновості.

3. Обмеження, що накладають на структуру синтезованої мережі з боку системи керування.
4. Характеристика інформації;
5. Показники надійності елементів мережі;
6. Вартісні характеристики елементів мережі.

Завдання синтезу мережі розподіленої системи керування полягає у виборі структури, алгоритмів, розробці технічних комплексів мережі, що забезпечують виконання заданих вимог до обміну даними між вузлами мережі та оптимізуючих деяку цільову функцію.

Як вимоги до обміну повідомленнями задаються вимоги часу доставки по категоріях терміновості, норми вірогідності, надійності та ін. Оптимізується найчастіше цільова функція у вигляді вартісної характеристики [19].

Серед найбільш відомих методів аналізу потоків для розгляду, виділені наступні:

1. Накопичення статистики по роботі реальної системи.
2. Тестування мережі шляхом введення генераторів потоку.
3. Засновані на правилах системи.
4. Графові потокові алгоритми;
5. Системи масового обслуговування.
6. Мережі Петрі.

Одним з найбільш ефективних є аналіз взаємодій між вузлами в мережі шляхом накопичекння й аналізу статистики звернень до пристроїв. Зберігаючи в базі даних структуровану інформацію з взаємодії в мережі, яку можна отримати із заголовків переданих по мережі пакетів, і обробляючи накопичені дані за допомогою автоматизованих алгоритмів аналізу, можна скласти чітку схему використання інформаційних ресурсів мережі, необхідну для формування збалансованої обробки потоків.

Задача про максимальний потік – це задача пошуку максимального значення функції потоку [42, 67], заданої на ребрах графа G з певними

пропускними здатностями ребер, джерелом *s* і стоком *t*. Функція потоку повинна задовольняти умовам обмеження пропускної здатності ребер й умовам збереження потоку у всіх вершинах, за винятком джерела й стоку.

Прийнято розрізняти завдання пошуку орієнтованого й неорієнтованого потоку. У стандартному формулюванні задачі пропускні здатності – будь–які дійсні числа.

Використання потокових алгоритмів широко застосовується при формуванні маршрутних таблиць маршрутизаторів [65, 83, 187, 189].

Найбільш розповсюджені протоколи маршрутизації – RIP (Routing Information Protocol) і OSPF (Open Shortest Path First). Метод RIP інакше називається методом рельєфів [100, 101]. Він заснований на алгоритмі Белмана – Форда [100]. OSPF – алгоритм динамічної маршрутизації, у якому інформація про кожну зміну у мережі розсилається лавиноподібно.

Хоча алгоритм Беллмана – Форда сходиться повільно, для мереж порівняно невеликих масштабів він цілком прийнятний. У більших мережах краще себе зарекомендував алгоритм OSPF. Він заснований на використанні в кожному маршрутизаторі інформації про стан всієї мережі. В основі OSPF лежить алгоритм Дейкстри пошуку найкоротшого шляху в графах. При цьому мережа моделюється графом, у якому вузли відповідають маршрутизаторам, а ребра – каналам зв'язку. Ваги ребер – оцінки (відстані) між сусідніми вузлами.

Графові потокові алгоритми використовуються при функціонуванні пристроїв обчислювальної мережі [31], тому доцільно використовувати їх і при моделюванні. Засоби аналізу за допомогою графових потокових алгоритмів описані в роботах L. R. Ford'a [164, 165], L. Kleinrock'a [59, 60, 166], A. V. Goldberg'a [169, 170] та ін.

Будь–яка інформаційна система може бути представлена сукупністю приладів масового обслуговування. Для складних систем, таких як розподілена система керування, кількість таких приладів може бути досить великою. Імітаційне моделювання таких систем за допомогою мереж

масового обслуговування з аналізом й оцінкою їх функціонування стає ускладненим, тому що виникає необхідність у застосуванні якісного аналізу властивостей інформаційних систем [26].

Важливим етапом у застосуванні теорії масового обслуговування для дослідження реального об'єкта є формальний опис функціонування цього об'єкта в термінах тієї або іншої системи масового обслуговування (СМО). СМО вважається заданою, якщо повністю описані наступні її компоненти [44, 60, 139]:

1. Вхідний потік запитів (заявок, вимог, повідомлень, викликів);

2. кількість і типи обслуговуючих пристроїв;

3. ємності накопичувачів (буферів), де запити, що застали всі прилади зайнятими, очікують початку обслуговування;

4. час обслуговування запитів в приладах;

5. дисципліна обслуговування (визначає порядок обробки запиту в системі, починаючи з моменту його надходження в систему, і до моменту, коли він залишає СМО). Сукупність взаємозалежних СМО називається мережею масового обслуговування. У мережах масового обслуговування розрізняють експонентні й неекспонентні моделі обчислювальних мереж [44].

Експонентні моделі засновані на припущенні про те, що потоки заявок, що надходять у ВР, є пуасонівськими, а час обслуговування у вузлах має експонентний розподіл. Для таких мереж отримані точні методи для визначення їхніх характеристик. Трудомісткість одержання рішення залежить в основному від розмірності мережі. Повідомлення, що прийшло у вузол, чекає деякий час до того, як воно буде оброблено. При цьому може утворитися черга повідомлень, що очікують обробки. Час передачі, або повний час затримки повідомлення d дорівнює:

$$D = T_{\text{р}} + S + W,$$

де $T_{\text{р}}$ – час поширення,

S – час обслуговування,

W – час очікування.

При великих завантаженнях основний внесок дає очікування обслуговування W. Середнє значення довжини черги Q при заданій середній вхідній частоті повідомлень λ і середньому часі очікування W визначається виразом (теорема Литла):

$$\overline{Q} = \lambda \overline{W} .$$

Для вхідного процесу, що характеризується розподілом Пуассона зі швидкістю надходження повідомлень λ, імовірність надходження k повідомлень на вхід за час t дорівнює:

$$p(k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, k = 1,2,3...$$

Знаючи ймовірність надходження даних за період, можна одержати розподіл інтервалу τ між сусідніми подіями:

$$p(\tau) = \lambda e^{-\lambda t}.$$

Пуасонівский потік характеризується експонентним розподілом інтервалів між подіями. Середній час очікування W може бути визначено з використанням формули Поллажека–Хинчина:

$$\overline{W} = \frac{\overline{Q}}{\lambda} = \frac{\rho\tau}{2(1-\rho)}(1+\frac{\sigma^2}{\tau^2}),$$

де σ – середньоквадратичне відхилення для розподілу часу обслуговування,

τ – час обслуговування,

$$\rho = \lambda\tau .$$

Способи математичного моделювання мереж масового обслуговування в існуючий час достатньо добре вивчені й часто застосовуються на практиці. Є аналітичні формули для оцінки ефективності обслуговування в системах з найпростішими (Пуасонівськими) потоками заявок. Вони названі по імені їхнього автора формулами Ерланга [40].

Моделі СМО зручні для опису окремих підсистем сучасних обчислювальних мереж. Обчислювальна мережа в цілому представляє собою

сукупність взаємозалежних підсистем. Сукупність взаємозалежних СМО, взаємодія яких носить імовірнісний характер, називають Марковськими мережами масового обслуговування, а взаємодія яких носить інший характер – Немарковскими мережами масового обслуговування [44].

Мережі Петрі є гнучким апаратом моделювання складних динамічних систем [80], якими і є обчислювальні мережі з якістю обслуговування. Є безліч модифікацій мереж Петрі, у яких реалізуються механізми обліку імовірнісних характеристик систем, що моделюються, урахування часових параметрів, одночасного аналізу різних видів потоків. Розроблено алгебраїчні методи аналізу мереж Петрі, як їх статичних, так і динамічних властивостей [28].

Засоби аналізу й верифікації моделей на мережах Петрі описані в роботах Герхардта Х.Д. [28], Coves C. [157], Murata T. [182, 183], А.А. Суконщикова [2, 127], Зайцева Д.А. [47–51].

1.5. Використання UML для моделювання цифрових систем керування

У сучасних умовах безперервний ріст складності й ефективності елементного базису, що використовується, призводить до ускладнення процесів розробки, проектування та реалізації цифрових систем. Спостерігається стійка тенденція до зближення стратегій проектування апаратних і програмних засобів [30, 39].

Уніфікована мова моделювання (UML) призначена для проектування як програмних, так і апаратних систем, зокрема цифрових систем керування [158, 159, 177]. UML надає широкий набір інструментів (представлень, діаграм і засобів розширення), які вимагають відповідного аналізу й адаптації при використанні в конкретній області. Адаптація уніфікованої мови моделювання для цифрових систем керування передбачає:

– використання існуючих стандартизованих профілів для вбудованих цифрових систем;

– розробку й використання нових профілів різних моделей керуючих автоматів, а також профілів верифікації;

– аналіз, визначення призначення, взаємодії, та подальше використання підмножини діаграм уніфікованої мови моделювання.

Статично моделюються компоненти та сигнали системи керування. Сигнали використовуються для опису статичних зв'язків між окремими компонентами.

На етапі динамічного моделювання розглядаються динамічні, або поведінкові аспекти системи. Динамічна модель є одночасно межоб’єктною та внутрішньооб'єктною (характеризує, як залежить об'єкт від стану, який визначається скінченним автоматом, і зображується на діаграмі станів). Внутрішньооб'єктна сторона динамічного моделювання має відношення до скінченних автоматів і діаграм станів.

Взаємодія між об'єктами описується за допомогою UML–діаграм кооперацій. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але й всі структурні відносини між об'єктами, що беруть участь у цій взаємодії. Повідомлення на діаграмі кооперації специфікує комунікацію між двома об'єктами, один із яких передає іншому визначену інформацію. Таким чином, повідомлення є причиною, або стимулом для початку виконання операцій, відправлення сигналів, створення й знищення окремих об'єктів. Зв'язок забезпечує канал для спрямованої передачі повідомлень між об'єктами від об'єкта–джерела до об'єкта–одержувача.

Уніфікована мова моделювання базується на графічній нотації. На основі побудованої моделі можливий синтез системи керування (у тому числі й розподіленої) у заданому апаратному базисі. Використання уніфікованої мови моделювання дозволяє виконати аналіз всіх повідомлень, якими обмінюються об'єкти в процесі взаємодії. На підставі цих даних пред'являються вимоги до граничних пропускних здатностей різних

фрагментів мережі, що впливає на вибір топології й протоколу обміну даними.

1.6. Використання автоматного підходу в проектуванні засобів автоматизації АЕЕС

Сучасні системи керування будуються на основі інтелектуальних і спеціалізованих мікроконтролерних модулів, адаптованих по архітектурі, продуктивності й набору інтегрованих на кристал периферійних пристроїв до завдань прямого цифрового керування устаткуванням [77, 81]. При цьому спеціалізована периферія (аналого–цифрові перетворювачі, таймери–лічильники, квадратурні декодери та ін.) працюють практично автономно, відволікаючи центральний процесор по перериваннях тільки для одержання нового завдання або повернення результату. Характерною рисою таких систем є необхідність твердої ієрархії рішення множини завдань із різною частотою [98].

Побудова систем керування з використанням технології автоматів при невеликому числі внутрішніх станів має ряд переваг [144–147]. З одного боку, мікропроцесор представляє собою складний автомат [123], зміна станів якого визначається програмою, що виконується. Велика кількість його станів і переходів між ними, а також асинхронна робота мікропроцесора зменшують надійність [54] функціонування пристрою і не гарантують своєчасної доставки інформації. З іншого боку, реалізація систем керування у вигляді набору скінченних синхронних автоматів дозволяє організувати обробку паралельних потоків інформації в режимі реального часу й забезпечити строго детерміновану затримку при доставці інформації до пунктів призначення.

Керуючий автомат можна розглядати як пристрій, що реалізує алгоритм функціонування систем керування, що задає послідовність виконання тих або інших операцій по керуванню деяким об'єктом [11].

Методи композиції вирішують завдання аналізу мереж, оскільки дослідження поведінки мереж зручно проводити при наявності автоматного опису (графа переходів, таблиці переходів, системи канонічних рівнянь).

Найбільш практично важливою й добре вивченою задачею декомпозиції є задача реалізації автомата в заданому автоматному базисі, тобто завдання побудови мережі, що реалізує даний автомат, із заданого набору автоматів (автоматного базису), що називається в цьому випадку елементарними автоматами [73].

Як правило, таблиці переходів містять ряд збиткових внутрішніх станів, усунення яких не порушить алгоритму роботи автомата, але, у той же час, дозволить спростити його функціональну схему. У ряді випадків наявність надлишкових станів не є очевидним фактом, тому для їх виявлення необхідно проводити детальний аналіз таблиць переходів.

Скорочена частина станів асинхронного автомата базується на виявленні еквівалентних і псевдоеквівалентних станів, а також на виявленні й об'єднанні спільних внутрішніх станів автомата (рядків початкової таблиці переходів).

На рис. 1.8 представлений алгоритм мінімізації кількості станів автомата. Принцип роботи алгоритму полягає в наступному. У вигляді таблиці переходів задається формалізований опис закону функціонування автомата. У циклі виконується порівняння станів автомата для кожного із вхідних значень сигналу (відбувається пошук однакових станів у кожному стовпці). Якщо є однакові стани, то в багатомірний масив arr записується значення «1». Індекси масиву (col, row, index) визначають, у яких комірках таблиці є однакові стани.

На наступному етапі роботи алгоритму виконується пошук однакових вихідних сигналів для кожної з пар еквівалентних станів. Для цього значення елемента багатомірного масиву перевіряється на рівність «1» і виконується порівняння вихідних сигналів для відповідних станів.

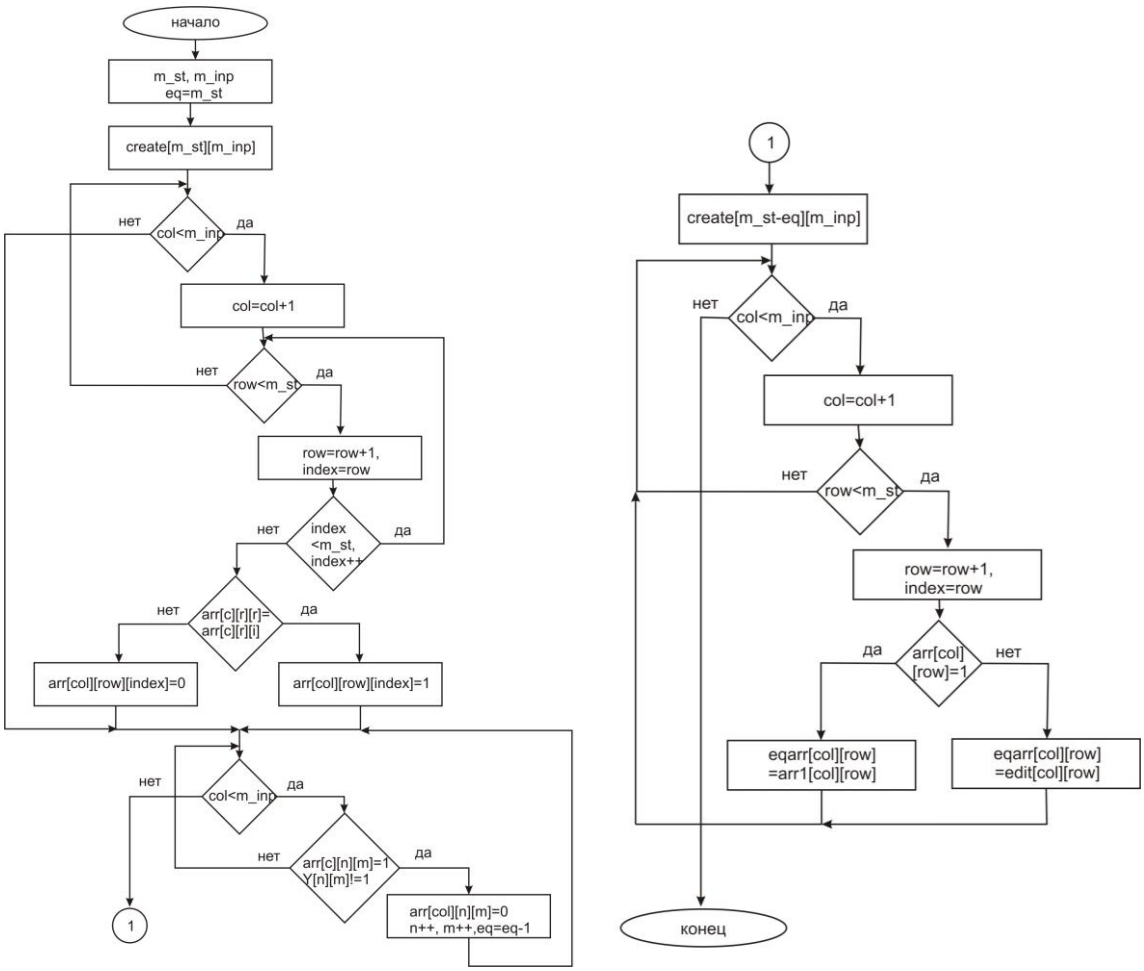


Рис. 1.8. Блок–схема алгоритму мінімізації кількості станів автомата

Після того, як знайдені еквівалентні стани, створюється нова таблиця, у якій еквівалентні стани поєднуються. Інші елементи таблиці залишаються без змін.

На підставі отриманої таблиці з мінімальним числом станів автомата будується граф переходів.

За допомогою скінченних автоматів можна побудувати модель всієї системи керування автономною електроенергетичною системою. При описі поведінки системи використовується не один автомат, а система взаємозалежних автоматів, що дозволяє, зокрема, описувати паралельні процеси.

Перехід електроенергетичної системи з одного стану в інший може відбуватися по команді оператора, при зміні режиму роботи електростанції внаслідок зміни навантаження, виникнення аварійної ситуації.

Методологія алгоритмізації й побудови програм логічного керування припускає:

– системне (архітектурне) проектування;

– розробку формальних специфікацій автоматів;

– побудова алгоритмічних моделей автоматів;

– програмування.

На першому етапі описуються об'єкти керування. Виділяються джерела інформації й виконавчі механізми. Надалі на цій стадії визначається склад додаткових джерел інформації (органів керування) і засобів представлення інформації. Якщо автомати розглядати як пристрої із входами й виходами, то приєднання виходів одних автоматів до входів інших створює мережу з автоматів. На рис. 1.9 представлена UML–діаграма станів автомата при виході АЭЭС зі знеструмленого стану. У кожному зі станів системи функції виконуються певною системою, алгоритм якої також може бути описаний з використанням автоматного підходу. Взаємодія між об'єктами здійснюється по мережі.

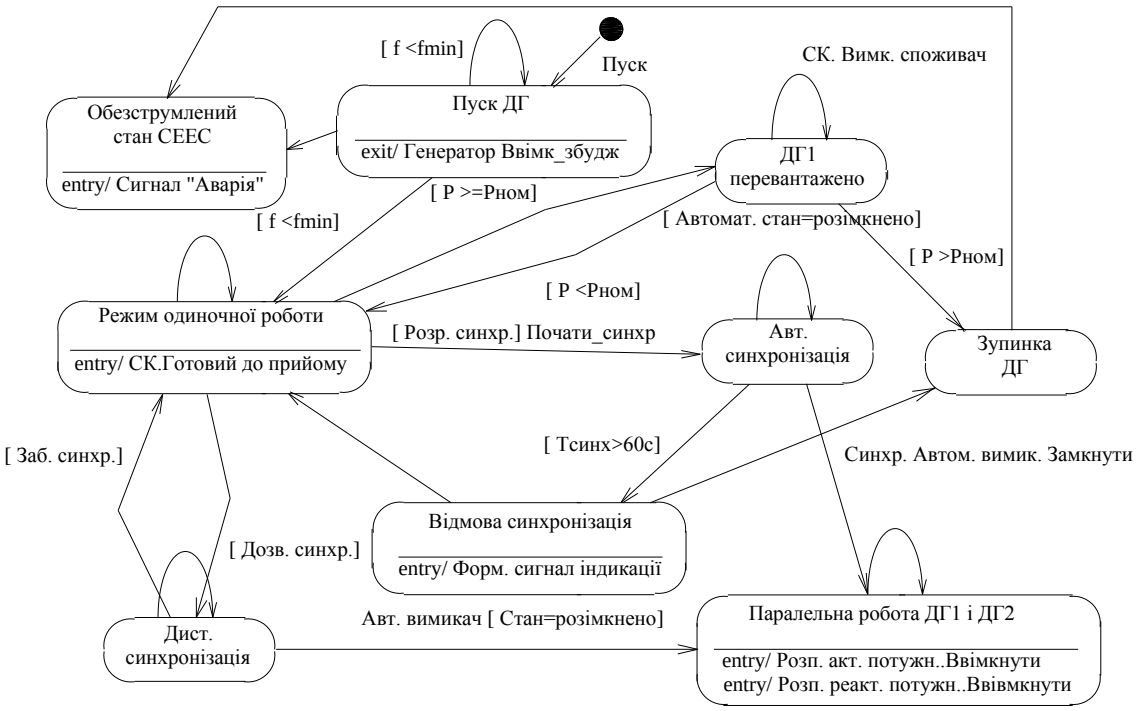


Рис. 1.9. UML–діаграма станів автомата

На рис. 1.10. представлено UML–діаграму станів автомата, що виконує екстрений пуск дизель–генератора.

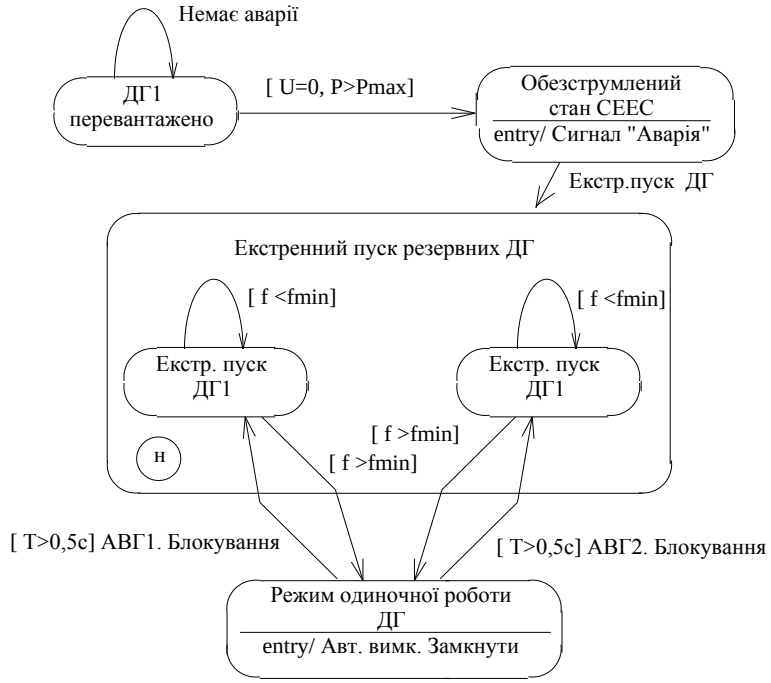


Рис. 1.10. Діаграма станів автомата при екстреному пуску дизель–генератора

З використанням автоматного підходу можуть бути побудовані ряд засобів автоматизації АЭЭС.

Система синхронізації генераторів (ССГ) є однією з найважливіших систем автономної електроенергетики для забезпечення стабільної роботи електричної мережі. ССГ призначена для видачі сигналів на включення автоматичного вимикача при точній синхронізації. Вона підключається до генераторів, які необхідно синхронізувати, за допомогою типових вимірювальних трансформаторів. ССГ видає сигнал на включення автоматичного вимикача при виконанні умов синхронізації: при знаходженні кута неузгодженості напруг φ у припустимих межах, при припустимій різниці напруг ΔU генераторів і при задовільній різниці частот Δf генераторів. При цьому також враховується час спрацьовування автоматичного вимикача T_a, що безпосередньо підключає фази генератора до мережі (або іншого генератора). Тому сигнал на включення автоматичного

вимикача видається з деяким випередженням. Блок синхронізації повинен видавати сигнал на збільшення частоти генератора, що підключається, якщо:

$$-5Гц \le \Delta f < 0{,}2Гц,$$

і на зниження частоти при

$$-0{,}6Гц \le \Delta f < 5Гц.$$

Якщо різниця частот перевищує 1 Гц, видається безперервний сигнал на зміну частоти генератора, що підключається. Інакше, видається імпульсний сигнал. Максимальна різниця частот, при якій виробляється сигнал на збільшення або на зменшення частоти – не менше 5 Гц.

Вхідними сигналами для ССГ, реалізованого на базі мікроконтролера є імпульси, які формуються при переході напруги кожного з генераторів (або напруги мережі і генератора) через нуль. Ці сигнали подаються на входи зовнішніх переривань *Int0* та *Int1* мікроконтролера.

Алгоритм роботи системи синхронізації є циклічним, тому його можна представити у вигляді графа переходів скінченного автомата. На рис. 1.11 представлений автоматний алгоритм блоку синхронізації.

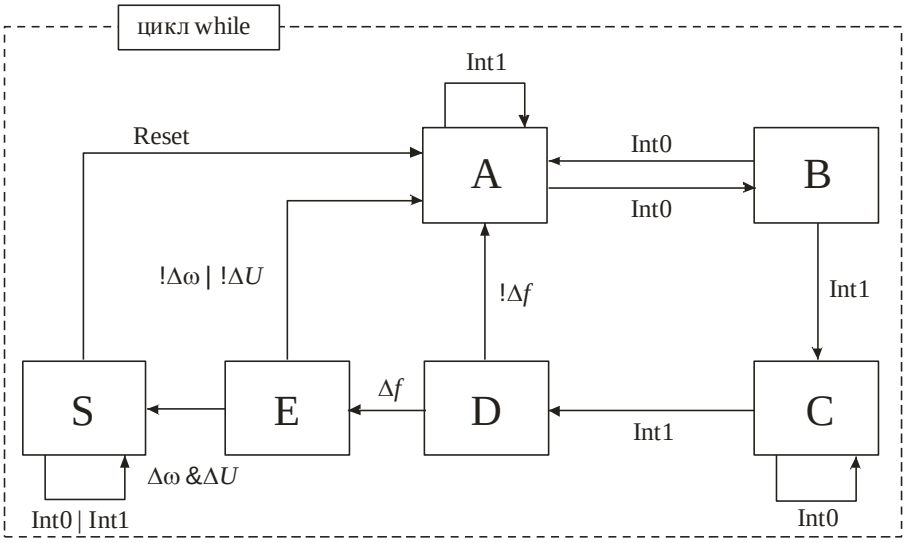


Рис. 1.11. Автоматний алгоритм системи синхронізації

Кожна наступна подія починає оброблятися тільки після завершення обробки попередньої. Зміна стану автомата можлива по одній з наступних подій:

– зовнішнє переривання *Int*0, викликане фронтом імпульсного сигналу з компаратора 1;

– зовнішнє переривання *Int1*, викликане фронтом імпульсного сигналу з компаратора 2;

– завершення роботи функції розрахунку різниці частот;

– завершення роботи функцій визначення припустимої різниці фаз і значень напруг;

– натискання на кнопку скидання (Reset).

Для відстеження поточного стану автомата вводиться спеціальна змінна state. Початковим станом є стан 'A', у який переходить система при включенні живлення. Прямокутниками, позначеними буквами A – E та S, позначені вершини графа, що визначають його шість станів.

Коли автомат перебуває в стані 'B', очікується поява одного із зовнішніх переривань. При виникненні переривання 0, автомат переходить у початковий стан 'A'. Цей перехід можливий, якщо за час періоду напруги працюючого генератора, напруга генератора, що підключається, не перетинала значення 0, наприклад, через значну різницю частот. Якщо в стані 'B' автомата виникло переривання 1, автомат переходить у стан 'C'. При цьому виконується зупинка таймера 1, за допомогою якого виміряються фазовий зсув між напругами, і запускається таймер 2 (TIMER2), що використовується для виміру періоду напруги генератора, що підключається.

У стані автомата 'C' виникнення події зовнішнього переривання 0 викличе зупинку таймера 0 (TIMER0), при цьому стан автомата не зміниться. Дані про період (отже, і частоту) напруги працюючого генератора будуть отримані. Перехід зі стану 'C' в 'D' може відбутися тільки при повторному виникненні переривання 1. Це означає, що значення періоду напруги генератора, що підключається, було виміряне.

У стані ‘D’ виконується обчислення різниці частот напруг працюючого ігенератора і генератора, що підключається. Це виконує функція CalcDeltaFreq(), що повертає значення істина (TRUE), якщо різниця частот

лежить у припустимих межах, або лож (FALSE), якщо різниця частот велика. У цій же функції виконується формування сигналу на збільшення/зменшення частоти генератора, що підключається, залежно від значення різниці частот Δf. Таким чином, виконання функції підготовлює функціонування автоматної програми в новому стані. Відпрацьовування автоматом однієї події – досить короткочасний процес. Змінні, у яких зберігаються значення періодів сигналів і різниці фаз, є глобальними і доступні з будь–якої функції. Завершення процесу обчислення різниці частот є локальною подією, що приводить до зміни стану автомата. Напрямок переходу визначається значенням різниці частот. Якщо різниця частот велика, і не задовольняє вимогам синхронізації, автомат переходить у стан ‘A’, оскільки подальше визначення припустимості різниці фаз і напруг не має смислу. Якщо різниця частот мала, то автомат переходить у стан ‘E’, у якому виконується розрахунок різниці фаз і різниці напруг.

У стані 'S' автомат перебуває до повторного включення блоку живлення системи синхронізації, або подачі сигналу скидання Reset, по якому автомат переходить у стан 'A' і процес синхронізації починається заново.

1.7. Визначення проблеми та вимог до математичного апарату

При проектуванні систем автоматизованого керування необхідно вибрати устаткування, яке буде задовільняти по своїм вартісним і технічним характеристикам, оцінити навантаження на мережу й характеристики цього навантаження. Складність ситуації полягає в тому, що багато параметрів системи задаються в нечіткій формі, і комплексне рішення цих завдань є досить непростою справою.

Однією з актуальних проблем загальної теорії систем є розробка методів дослідження систем в умовах невизначеності [37, 38]. Труднощі, пов'язані з рішенням цієї проблеми, обумовлені цілим рядом факторів, до яких можна віднести наступні:

1. наявність великої кількості взаємозалежних підсистем (елементів) зі складними структурними відносинами між ними (підсистеми керування, аварійного захисту, діагностики);
2. функціонування окремих підсистем (елементів) відбувається асинхронно, правила взаємодії їх описуються логічними умовами;
3. функціонування окремих підсистем не носить самостійний характер й обумовлено її місцем у системі в цілому;
4. невизначеність поводження окремих підсистем може носити як імовірнісний, так і нечіткий характер.

На підставі вивчених підходів [3, 4, 8, 26, 59, 155, 156, 157] визначені попередні вимоги до математичного апарату:

1. Відсутність необхідності безпосереднього доступу до реально діючої мережі. У складній системі керування доступ до окремих елементів може бути обмежений, або може знадобитися оцінити функціонування системи в критичних умовах.
2. Відсутність необхідності знання будови всіх елементів мережі. Для аналізу всієї системи в цілому можна зневажити роботою деяких окремих її частин.
3. Одночасний аналіз великого комплексу механізмів обробки потоків. Робота механізму обробки потоків повинна бути показана у взаємодії.
4. Аналіз потоків у мережі повинен проводитися протягом малого інтервалу часу і використовувати для обчислень обмежені ресурси.
5. Можливість використання на етапі проектування. При модернізації існуючої мережі або проектуванні нової може знадобитися оцінити її функціонування ще до закупівлі устаткування і прокладки кабельних систем.
6. Розвинений аналітичний апарат. Наявність розвиненого аналітичного апарату дозволяє більш ефективно проводити обробку даних, швидше й точніше одержувати результати.
7. Можливість роботи в умовах нечітких даних. У такій області, як

система керування автономною електроенергетикою, деякі оцінки можуть бути виражені тільки через лінгвістичне завдання. Обробкою такого роду оцінок займається нечітка логіка [7, 34, 79].

РОЗДІЛ 2
МЕТОДИКА АНАЛІЗУ ПРОДУКТИВНОСТІ МЕРЕЖІ ОБРОБКИ ІНФОРМАЦІЇ РОЗПОДІЛЕНОЇ СИСТЕМИ КЕРУВАННЯ АЕЕС

2.1. Вимоги до якості обслуговування

Як основні вимоги до якості обслуговування найчастіше вибирають такі характеристики [100, 101, 103, 105]:

1. Відносна передбачуваність швидкості передачі даних.

2. Чутливість до затримок пакетів.

3. Чутливість до втрат і перекручувань пакетів.

За критерієм передбачуваності швидкості потоки можна розділити на два класи:

Потоки з рівномірним трафиком – навантаження надходить у мережу з постійною бітовою швидкістю. Швидкість потоку може змінюватися, але вона має верхню границю, що легко обчислюється.

Потоки з пульсуючим трафиком – інтенсивність навантаження відрізняється високим ступенем непередбачуваності, коли періоди очікування замінюються передачею великих блоків даних. Для потоку характерна бітова швидкість, що змінюється. Так, у випадку запиту даних (наприклад, діагностична інформація), інтенсивність потоку може збільшуватися від нуля, коли дані не передаються, до повного насичення смуги пропущення каналу.

За критерієм чутливості потоку до затримок розрізняються потоки:

Асинхронні – практично немає обмеження на час затримки.

Синхронні – чутливі до затримок, але допускають їх (діагностична інформація).

Ізохронні – при перевищенні порога чутливості до затримок функціональність різко знижується (керування деякими технологічними об'єктами).

Надчуттєві до затримок – затримки доставки даних зводять функціональність до нуля (керування об'єктом у реальному часі).

За критерієм чутливості до втрат пакетів та їх спотворень потоки розділяються на 2 групи:

– чутливі до втрат і перекручувань, коли втрата одного пакета знецінює інформацію (керування об'єктом);

– стійкі до втрат. Відсоток допустимості втрат оцінюється кількісно (наприклад, 1%).

Трьом критеріям класифікації (передбачуваність швидкості, чутливість до втрат і затримок) відповідають три групи параметрів:

– параметри пропускної здатності, що включають середню, максимальну і мінімальну швидкості передачі даних;

– параметри затримок – середню й максимальну величину затримок, а також середньє й максимальнє значення варіацій затримок;

– параметри надійності передачі – відсоток загублених пакетів, а також відсоток спотворених пакетів.

Вимір зазначених параметрів виконується на визначеному інтервалі часу. Чим менше цей часовий інтервал, тим більш жорсткі вимоги пред'являються до мережі, а отже до всіх її елементів, оскільки забезпечення якості обслуговування вимагає взаємодії всіх вузлів на шляху пакета і визначається надійністю, функціональністю та продуктивністю самої слабкої ланки.

2.2. Класифікація інформаційних потоків з використанням нечіткої логіки

Розподілені системи керування, що працюють у реальному масштабі часу, мають потребу у твердих і точних гарантіях якості обслуговування. Коректна побудова мережі й класифікація потоків з розбивкою на невелике число пріоритетних класів, можуть виявитися достатніми для забезпечення нормальної роботи системи керування. Отже, правильне виділення ресурсів

на періоди пікового навантаження або аварійної ситуації, разом із захистом від потоків з більш низьким пріоритетом, забезпечить необхідний рівень обслуговування. Перевагою даного рішення є більш висока загальна ефективність, тому що вона дозволяє передавати більше число потоків більш простим чином і за допомогою простих механізмів організації шляхів передачі даних [160]. Такого роду задачі при багатокритеріальному нечіткому розподілі пропонується вирішувати за допомогою апарата нечіткої логіки.

Математичний апарат нечіткої логіки досить давно використовується в керуванні, системах штучного інтелекту, системах прийняття рішень [41, 90, 96, 97].

Розподіл на класи виконується виходячи з очікуваної інтенсивності потоків, вимог до якості обслуговування потоків. Ухвалення рішення про приміщення найпростішого потоку в конкретний клас обслуговування виробляється на підставі близькості нечітких оцінок їхніх обсягів, фізичного розташування приймачів і генераторів пакетів.

Суть методу полягає в наступному [34, 35, 46]:

1. Фазифікація – внесення нечіткості:

   – виділяються лінгвістичні змінні, необхідні для оцінки якості обслуговування потоку;

   – для кожної лінгвістичної змінної вибирається кількість термів, і визначаються фізичні значення граничних для них умов.

2. Приймається рішення про кількість класів обслуговування.
3. Розробляються нечіткі правила обробки нечітких оцінок потоків.
4. Розробляються правила дефазифікації – для усунення нечіткості результату при декількох результуючих термах.

Для оцінки якості обслуговування потоку за методами нечіткої логіки виділимо лінгвістичні змінні:

– чутливість трафіка до затримок;

– передбачуваність швидкості;

– чутливість до втрат пакетів.

Крім цього для оцінки потрібно визначити такі лінгвістичні змінні:

– швидкість передачі;

– розподіл інтенсивності потоку в часі.

Терми для лінгвістичних змінних можна задати відповідно до класифікації з [102].

1. Чутливість трафіка до затримок *Fz* виражається через час. Змінна буде приймати значення $X(Fz) \in \{0 \div 5c\}$. Лінгвістична змінна «чутливість трафіка» приймає наступні значення (рис. 2.1).

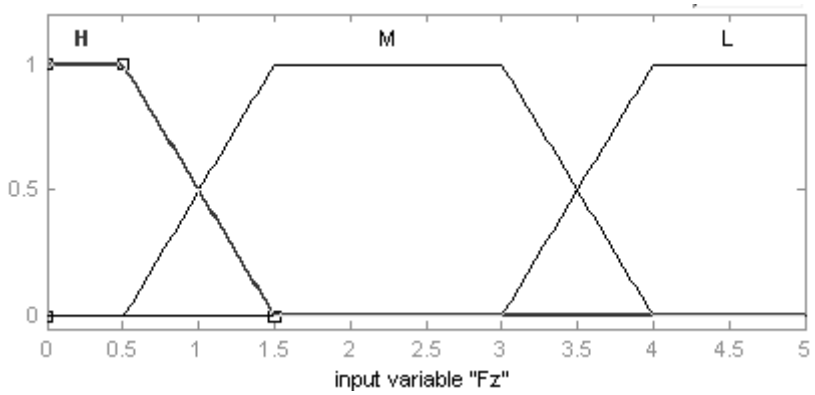


Рис. 2.1. Завдання функції нечіткого розподілу

Використання цієї лінгвістичної змінної дозволяє вивести розподіл параметрів затримок: мінімальну, середню, максимальну. Розрізняють потоки асинхронні – практично немає обмеження на час затримки, синхронні – чутливі до затримок, але допускають їх, надчуттєві – затримки доставки даних зводять функціональність до нуля.

2. Передбачуваність швидкості *W* можна виразити через відношення максимальної швидкості до мінімальної. Змінна буде приймати значення $X(W) \in \{1 \div \infty\}$. За допомогою лінгвістичної змінної «передбачуваність швидкості» можна вивести максимальну (пікову) і мінімальну швидкості передачі при заданій середній. За критерієм передбачуваності потоки можна розділити на два більших класи: потоки з рівномірним трафіком (швидкість потоку може мінятися, але вона має верхню границю; потоки з пульсуючим трафіком

(інтенсивність навантаження відрізняється високим ступенем непередбачуваності). Наприклад, у випадку запиту даних інтенсивність потоку може змінюватись від нуля, коли дані не передаються, до повного насичення смуги пропускання каналу.

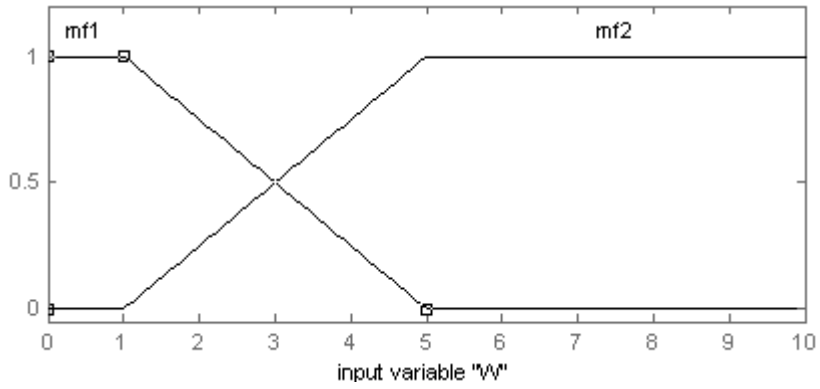


Рис. 2.2. Графіки функцій приналежності для термів вхідної лінгвістичної змінної «передбачуваність швидкості»

3. Оцінка чутливості до втрат пакетів *Y* виражається через відсоток припустимої кількості пакетів, що втрачаються. Змінна буде приймати значення $X(Y) \in \{0 \div 100\%\}$. Її використання дозволяє вивести припустимий відсоток загублених пакетів, а також відсоток викривлених пакетів.

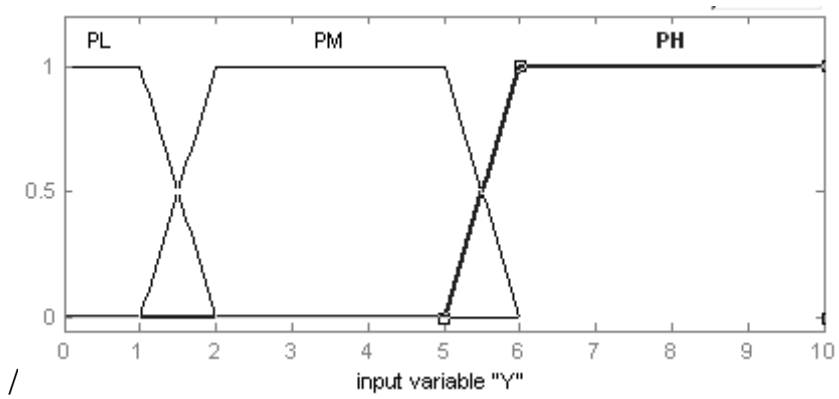


Рис. 2.3. Графіки функцій приналежності для термів вхідної лінгвістичної змінної «оцінка чутливості до втрат»

4. Швидкість передачі *V* виразимо через Біт/с. Змінна буде приймати значення:

$$X(V) \in \{0 \div 115 кБіт/с; 115 кБіт/с \div 1 Мбіт/с; 1 МБіт/с \div 10 МБіт/с\}.$$

Дозволяє вивести середню швидкість передачі даних.

Розподіл за часом *p* дозволяє вивести розклад функціонування потоків для кожного інтервалу протягом розглянутого періоду часу.

Таблиця 2.1

Множина параметрів оцінки потоку

Вимоги якості обслуговування	Лінгвістичні змінні	Нечіткі змінні
Вимоги до швидкості передачі	Швидкість передачі (*V*)	Швидкість передачі (*V*)
	Спрямованість потоку	Спрямованість потоку (*D*)
	Передбачуваність швидкості (*W*)	Передбачуваність швидкості (*W*)
	Розподіл за часом	Розподіл за часом – відсоток часу (*Q*)
		Розподіл за часом – періодичність генерації (T_Q)
Вимоги по чутливості до втрат пакетів	Чутливість до втрат пакетів (*Y*)	Чутливість до втрат пакетів (*Y1*)
		Чутливість до спотворень (*Y2*)
Вимоги по чутливості до затримок	Чутливість до затримок (*Fz*)	Чутливість до затримок (*Fz*)

Потоки в мережі розглядаються в рамках обмежених часових відрізків. В якості нечіткої функції розподілу пропонується використовувати трапецієподібні нечіткі інтервали. Кількість пакетів, що генерується у рамках обмеженого часового відрізка обчислюється на підставі виразу:

$$N = f(V, r, T_N, Q),$$

де *r* – розмір пакета,

V – швидкість передачі потоку,

T_N – довжина часового відрізка,

Q – відсоток часу від усього розглянутого відрізка, коли поток генерується.

5. Розподіл за часом *p* дозволяє вивести розклад функціонування потоків для кожного інтервалу протягом розглянутого періоду часу.

На рис. 2.4 представлені поверхні нечіткого виводу, які дозволяють установити значення вихідної змінної від значень вхідних змінних нечіткої моделі.

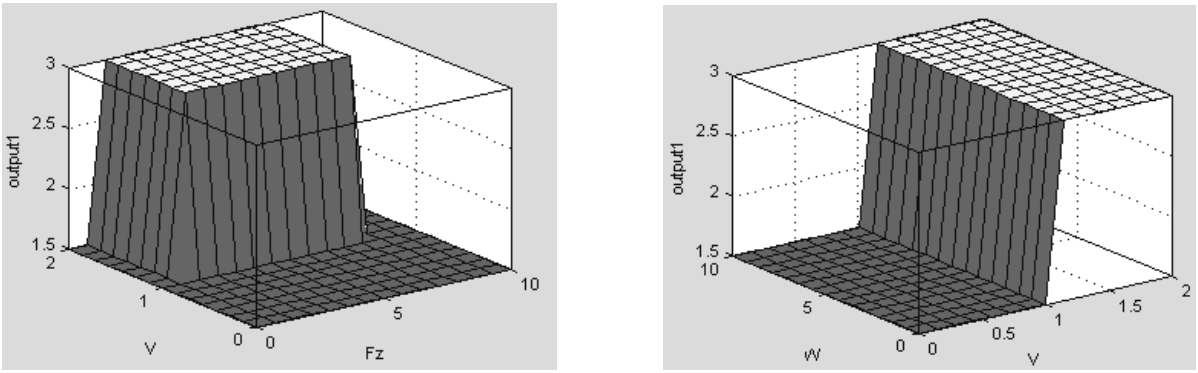


Рис. 2.4. Поверхні нечіткого виводу

Результатом нечіткої продукції правил стає віднесення розглянутого потоку до конкретного класу або класів обслуговування з нечітким значенням приналежності до класів. У наведеному прикладі використані три класи із пріоритетами 1, 2 і 3 (рис. 2.5).

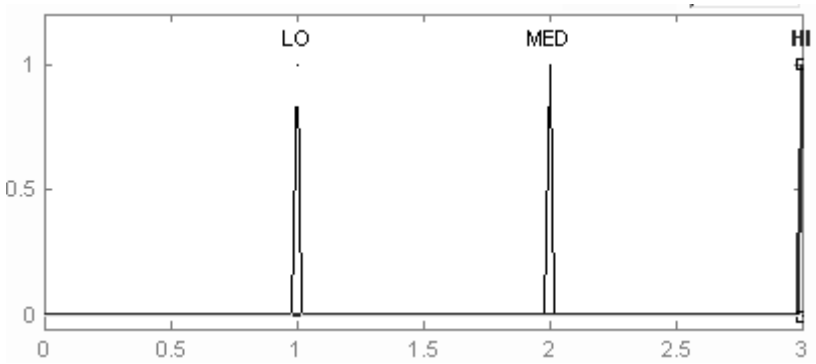


Рис. 2.5. Функції приналежності вихідної змінної

Для характеристики необхідного класу обслуговування використовуються три групи параметрів:

– параметри пропускної здатності, включають середню, максимальну й мінімальну швидкості передачі;

– параметри затримок – середня й максимальна величина затримок;

– параметри надійності передачі – відсоток загублених пакетом, а також викривлених.

Вимір зазначених параметрів виробляється на певному інтервалі часу. Чим менше цей часовий інтервал, тим більш жорсткі вимоги пред'являються до мережі, а отже й до всіх елементів, оскільки передача пакетів вимагає взаємодії всіх вузлів на шляху потоку й визначається надійністю, продуктивністю самого повільного вузла.

Після розділення потоків на класи й визначення оцінок, що їх очікують можна одержати загальну картину потоків у мережі. Виробляється об'єднання потоків приналежних до одного класу, спрощення структури мережі шляхом виключення незначущих для розрахунку елементів методами структурної композиції.

Використання нечіткої логіки можливо для зменшення перевантажень у мережі. Перевантаження часто відбувається, коли:

1. Найкоротші шляхи декількох потоків трафіка поєднуються в якихось каналах або інтерфейсах маршрутизатора;

2. Даний потік маршрутизований через канал або інтерфейс маршрутизатора, що не має достатньої смуги пропускання.

Пропоновані можливості містять у собі:

1. Набір атрибутів, пов'язаних з каналами передачі даних, сукупність яких характеризує робочий стан мережі.

2. Набір атрибутів, пов'язаних з ресурсами, які обмежують розміщення шляху інформаційних потоків. Вони можуть розглядатися також як топологічні обмеження.

3. "Маршрутизація на основі обмежень", що використовується для вибору шляху для каналу передачі даних відповідно до набору параметрів пунктів 1 і 2, наведених вище.

Представлена загальна схема механізму оцінки параметрів потоків у мережі ґрунтується на нечіткій логіці. Відповідно до механізму нечіткої продукції правил, пропонується розбивати потоки на класи. Вхідними позиціями приймаються терми таких лінгвістичних змінних, як «швидкість передачі», «передбачуваність швидкості», «оцінка чутливості до втрат

пакетів», «чутливість трафіка до затримок». Вихідними позиціями нечіткої продукції правил є класи обслуговування.

2.3. Формування матриці вимог до передачі даних

Після розбивки потоків на класи й визначення їх очікуваних оцінок, можна одержати загальну картину потоків у мережі. Виконується об'єднання потоків, приналежних до одного класу, спрощення структури мережі шляхом виключення несуттєвих для розрахунку елементів методами структурної композиції.

Використання нечіткої логіки можливо для зменшення перевантажень у мережі. Перевантаження часто відбувається, коли:

1. Найкоротші шляхи декількох потоків поєднуються в каналах або інтерфейсах маршрутизатора.

2. Даний потік спрямований через канал або інтерфейс маршрутизатора, що не має достатньої смуги пропущення.

Пропоновані можливості містять у собі:

1. Набір атрибутів, пов'язаних з каналами передачі даних, сукупність яких характеризує робочий стан мережі.

2. Набір атрибутів, пов'язаних з ресурсами, які обмежують розміщення шляху інформаційних потоків. Вони можуть розглядатися також як топологічні обмеження.

3. "Маршрутизація на основі обмежень", що використовується для вибору шляху для каналу передачі даних відповідно до набору параметрів пунктів 1 та 2, наведених вище.

Кількість пакетів, що генерується, є нечіткою величиною, яку можна обчислити по формулі:

$$T_N < T_Q : N = (V \times T_N) / r ,$$

$$T_N \geq T_Q : N = (V \times T_N \times Q) / r .$$

Функцію розподілу генерації за часом для кожного наступного пакета можна розрахувати по формулі:

$$\tau_0 = f(N, T_N),$$

$$\tau_0 = (T_N / N).$$

Класифікація потоку – виділення із загального потоку пакетів з однаковими вимогами до якості обслуговування. Класифікація виконується на основі різних принципів: адрес, пріоритетів, спеціальних позначок і т.п. [79]. Маркування пакетів дозволяє їх обслуговувати, але з більш низькою якістю, наприклад, зі збільшеним часом затримки.

Матриця вимог до передачі інформації для конкретного процесу в конкретний інтервал $T_N - H_k$ приводиться нижче.

H_k=	*i/j*	1	2	3	…	*n*
	1	*x*	h_{k12}	h_{k13}	…	h_{k1n}
	2	h_{k21}	*x*	h_{k23}	…	h_{k1n}
	3	h_{k31}	h_{k32}	*x*	…	h_{k3n}
	…	…	…	…	*x*	…
	m	h_{km1}	h_{km2}	h_{km3}	…	*x*

де h_{kmn} – вимога передачі пакета від *i*-го вузла *j* для *k*-го процесу.

Матриця вимог у передачі інформації для всіх процесів у конкретний інтервал $T_N - H$ знаходиться як сумарна матриця всіх матриць вимог до передачі потоків, що генеруються у мережі:

$$H = \sum_{k=1}^{n} H_k,$$

де *n* – кількість вузлів, які генерують потоки у мережі.

2.4. Пріоритетне обслуговування та зважені черги

Механізм пріоритетного обслуговування черг припускає наявність декількох вхідних підчерг: підчерги з високим, середнім, звичайним і низьким пріоритетом обслуговування. Дослідженню систем масового обслуговування та їх характеристик присвячені роботи [22, 24, 32, 53, 60, 71].

Пакети, що відносяться до черги з високим пріоритетом обслуговування, передаються першими. Коли високопріоритетна черга виявиться порожньою, почнеться передача пакетів наступної по пріоритету обслуговування черги і т.д. Передача пакетів черги із середнім пріоритетом обслуговування не почнеться доти, поки не будуть обслужені всі пакети високопріоритетної черги. Пріоритетне обслуговування необхідне для рішення критично важливих завдань, коли передача пакетів повинна бути здійснена навіть за умови повного домінування високопріорітетного трафика в моменти перевантаження мережі.

Механізм пріоритетного обслуговування пакетів підтримує класифікацію пакетів залежно від вхідного інтерфейсу, адреси, розміру пакета та об'єкта, що згенерувавцей пакет.

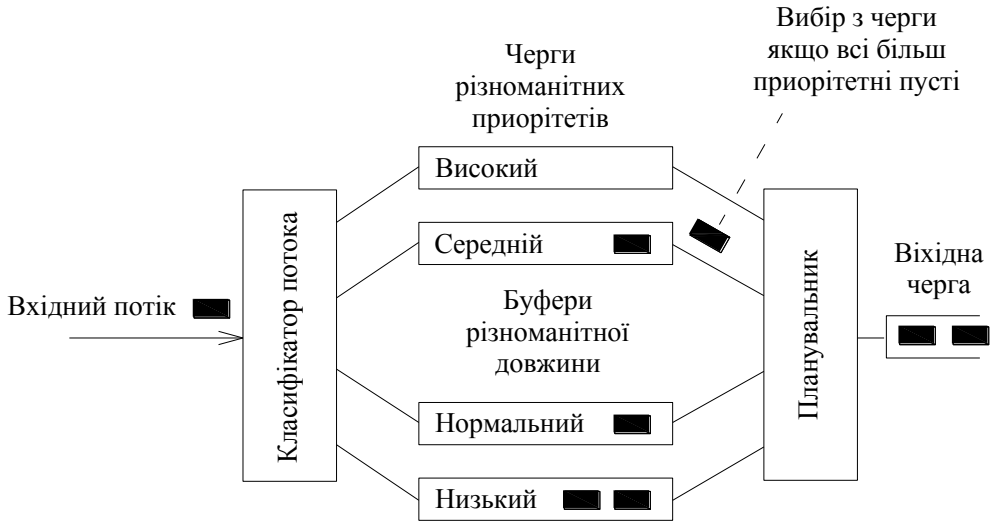


Рис. 2.6. Механізм функціонування моделі пріоритетного обслуговування

У межах черги пакети обробляються відповідно до механізму FIFO. Цілком можливо, що під час перевантаження мережі для передачі пакетів, необхідних для рішення критично важливих завдань, буде виділено 100 відсотків доступної смуги пропускання. Якщо інтенсивність високопріорітетного трафика зрівняється на певний проміжок часу з лінійною інтенсивністю або перевищить її, механізм пріоритетного

обслуговування черг однаково забезпечить передачу високопріорітетної інформації.

Алгоритм зважених черг розроблений для того, щоб можна було надати всім класам потоків певний мінімум пропускної здатності, або гарантувати деякі вимоги до затримок.

Під вагою даного класу розуміється відсоток пропускної здатності від повної пропускної здатності вхідного інтерфейсу, що надається класу потоку.

Як і при пріоритетному обслуговуванні, потоки діляться на кілька класів, для кожного з яких ведеться окрема черга пакетів. Але з кожною чергою зв'язується не пріоритет, а відсоток пропускної здатності вихідного інтерфейсу, що гарантовано надається даному класу потоку при перевантаженні цього інтерфейсу.

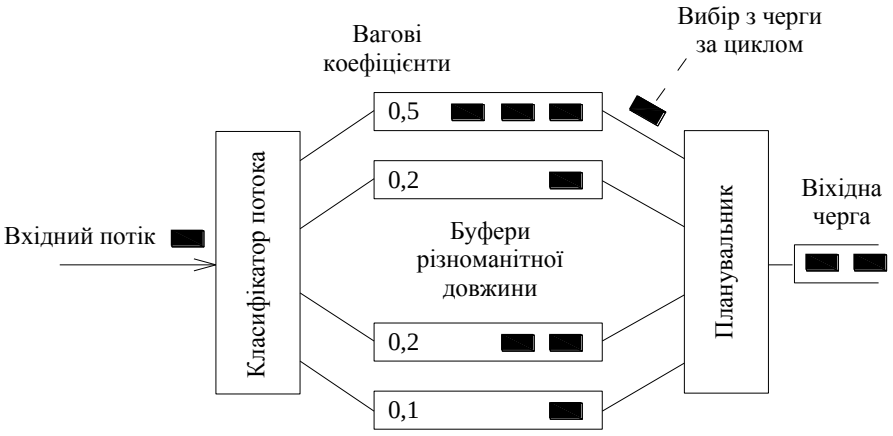


Рис. 2.7. Механізм функціонування зваженої черги, що налаштовується

Механізм FIFO не має засобів диференціювання пакетів, і, отже, не може виділити пріоритетні потоки.

2.5. Визначення параметрів мережі обробки інформації аналітичними засобами

Для оцінки якості функціонування й оптимізації мережі застосовують аналітичні (математичні) і імітаційні (програмні) методи й моделі експериментального дослідження [1].

Перевага аналітичних рішень полягає в тому, що вони дозволяють швидко й з мінімальними витратами одержати значення показників якості в широкому діапазоні зміни вхідних параметрів моделі. При фізичному моделюванні обчислювальної мережі практично неможливо перевірити її роботу для варіантів з використанням різних типів комунікаційних пристроїв – маршрутизаторів, комутаторів і т.п. Перевірка на практиці біля десятка різних типів маршрутизаторів зв'язана не тільки з великими зусиллями й часовими витратами, але й із чималими матеріальними витратами [190]. Тому, при оптимізації мереж у багатьох випадках кращим виявляється використання математичного моделювання.

Аналіз проводиться для мережі, що складається з рівноприорітетних вузлів, підімкнених до загального каналу Ethernet. Вузол, що має повідомлення для передачі, є активним, у противному випадку вона буде пасивним. У пасивному стані кожний вузол перебуває випадковий час, розподілений експоненціально з параметром λ. Інтенсивність сумарного потоку нових повідомлень, що генеруються дорівнює

$$\lambda_n = \lambda(M - n),$$

де M – число станцій,

n – число активних станцій,

$$n = \overline{0, M}$$

Якщо в момент виникнення повідомлення загальний канал вільний, то вузол починає передачу повідомлення. Якщо при виникненні повідомлення у вузлі канал зайнятий передачею іншого повідомлення, то повідомлення, яке з'явилось, очікує звільнення каналу. Після його звільнення, при числі повідомлень, що очікували, і $\geq$ 2, відбувається конфлікт. Виявлення конфлікту триває випадковий час τ_2. Вузли, що мають повідомлення для передачі в момент початку виявлення конфлікту, відкладають свою передачу на випадковий час, розподілений за експоненціальним законом з параметром υ. Після виявлення конфлікту починається його дозвіл, що триває випадковий час t_0 з функцією розподілу $B_0(x)$.

У випадку, коли після звільнення каналу очікує передачі тільки один вузол, то відбувається безконфліктна передача.

Середня пропускна здатність каналу $\bar{u}$ визначається як середнє число успішно переданих повідомлень за час, рівний середній тривалості μ^{-1} повідомлень:

$$\bar{u} = \lambda d / \mu = \lambda(M - \bar{n}).$$

Середній час перебування ϖ повідомлення

$$\varpi = \bar{n} / \lambda_D,$$

де середнє число активних вузлів:

$$\bar{n} = \sum_{n-1}^{M} n\left[P(n) + n(n-1)(P_2(n) + P_0(n, j))\right].$$

λ_0 – інтенсивність виходу успішно переданих повідомлень:

$$\lambda_D = \lambda(M - n)$$

Відповідно до [60] середній час затримки пакета в мережі масового обслуговування із груповим надходженням буде:

$$D_i = c_i((d_i / c_i^2 + f_i / b_i - 1)\rho_i / 2(1-\rho_i) + b_i d_i / 2c_i^2 - 1/2) \qquad (2.1)$$

де $c_i = M\{B_i\}$ – момент випадкової величини B_i.

$$d_i = M\left\{B_i^2\right\},$$

$$f_i = \sum_{j-1}^{\infty} f^2 r_{ij},$$

де ρ_i – коефіцієнт завантаження вузла;

B_i – реальний час передачі пакета;

r_{ij} – імовірність, з якою i – повідомлення запаковується в j пакетів:

$$\left(\sum_{j-1}^{\infty} r_{ij} = 1\right).$$

Апроксимація ймовірності безконфліктної передачі пакета вузлом q_i має вигляд:

$$q_i = (1-\rho_i)(1+t_c \sum_{j=1\, j\neq 1}^{N}((1-P_{j0})\nu_j + \rho_{j0}\lambda_j))^{-1}, \quad (2.2)$$

де t_c – тривалість вікна конфлікту;

$P_{i_0} = 1-\rho_i$ – імовірність того, що станція вільна;

ν_j – інтенсивність повторних звертань до каналу:

$$\frac{1}{\nu_j} = t\sum_{n=1}^{m}\frac{q_n q_i (1-q_i)^{n-1}}{1-(1-q_i)^m}, \quad (2.3)$$

$$q_i = M\{R_i\},$$

де R_i – цілочисельні незалежні випадкові величини.

В [85] показане формування ітераційної процедури для визначення q_i.

Вихідними даними для обчислення характеристик обчислювальних мереж при цьому є наступні набори величин, що вважаються заданими:

$\{\lambda_i, 1 \le i \le N\}$ – інтенсивність появи повідомлень на станції;

$\{r_{ij}, 1 \le i \le N, 1 \le j \le \infty\}$ – розподіл ймовірностей числа пакетів, що потрібні для пакування *i* повідомлення;

$\{a_i, 1 \le i \le N\}$ – середні часи передачі пакетів, включаючи міжкадрові інтервали;

$\{m, k(n), 1 \le n \le m\}$ – максимальне число спроб передачі пакета;

$k(n)$ – параметр, що визначає тривалість випадкових інтервалів між спробами;

t – тривалість інтервалу урізання кадру, дорівнює сумі тривалостей вікна конфлікту і бітової комбінації "наявність конфлікту" (у протоколі Ethernet вона дорівнює 512 бітових інтервалів)

t_c – тривалість вікна конфлікту.

Ітераційна процедура виглядає наступним чином:

1. Вважаємо $P_{i_0}(0) = 1, 1 \le i \le N$;

2. По формулі (2.2) обчислюємо $q_i(0), 1 \le i \le N$;

3. По формулі

$$c_i = a_i + t\sum_{n=1}^{m} q_i(1-q_i)^{n-1} g_n /(1-(1-q_i)^m)$$

обчислюємо $c_i(0), 1 \le i \le N$;

4. По формулі (2.3) обчислюємо $\nu_i(0), 1 \le i \le N$;

5. Використовуючи $c_i(n)$, по формулі

$$p_{i_0} = 1 - p_i = 1 - \lambda_i b_i c_i$$

обчислюємо

$$P_{j_0}(n+1), 1 \le i \le N, \qquad (2.4)$$

6. Використовуючи $\{P_{j_o}(n+1)\}, \{\nu_i(n)\}$, по формулі (2.4) обчислюємо ймовірність $P_{j_o}(q_i+1), 1 \le i \le N$;

7. Використовуючи $\{q_i(n+1)\}$, по формулах

$$c_i = a_i + t\sum_{n=1}^{m} q_i(1-q_i)^{n-1} g_n /(1-(1-q_i)^m),$$

і (2.3) обчислюємо $\{c_i(n+1)\}, \{\nu_i(n+1)\}$ відповідно;

8. Перевіряємо нерівність:

$$\sum_{i=1}^{N}(c_i(n+1) - c_i(n) + q_i(n+1) - q_i(n) + \nu_i(n+1) - \nu_i(n) + p_{j_0}(n+1) - p_{j_0}(n)) \le \varepsilon .$$

Якщо воно виконується, то процедура обчислення $\{c_i\}, \{q_i\}, \{\nu_i\}$, завершується. Якщо нерівність не виконується, то n присвоюється значення $(n+1)$ й повторюються етапи 5-8 ітераційної процедури.

9. Перевіряються умови існування стаціонарних режимів в мережі та кінцевому вузлі.

Після завершення цієї процедури обчислюються:

– коефіцієнт завантаження вузла:

$$\rho_i = \lambda_i b_i c_i, 1 \le i \le n;$$

– коефіцієнт завантаження загального каналу зв'язку (моноканала):

$$\rho = \sum_{i=1}^{n} \lambda_i b_i (1-(1-q_i)^m)(a_i + \frac{(1-q_i)t}{2q_i});$$

– коефіцієнт корисного завантаження загального каналу зв'язку:

$$\rho = \sum_{i=1}^{n} \lambda_i b_i (1 - (1 - q_i)^m a_i);$$

– середній час затримки пакета:

$$D_i = c_i((d_i / c_i^2 + f_i / b_i - 1)\rho_i / 2(1 - \rho_i) + b_i d_i / 2c_i^2 - 1/2);$$

– імовірність втрати пакета:

$$P_i = (1 - q_i)^m, 1 \le i \le N.$$

Разом з перевагами, аналітичні моделі мають ряд істотних недоліків. Ці недоліки обумовлені спрощеннями моделі, такими як подання потоків заявок як найпростіших, припущення про експоненціальний розподіл тривалостей обслуговування заявок, неможливість обслуговування заявок одночасно декількома приладами. Подібні спрощення, а найчастіше штучне пристосування аналітичних моделей з метою використання добре розробленого математичного апарата для дослідження реальних мереж можуть вносити значну похибку у результати моделювання.

2.6. Аналіз кореляційних характеристик інформаційних потоків

При різкому зростанні рівня потоків у мережі виникають перевантаження, тобто блокування надходження повідомлень у мережу. Однією з причин, що приводять до зниження продуктивності, є зайнятість систем обробки черг, які представляють собою системи масового обслуговування. Заявка, що надійшла на вхід системи “чекає” черги на виконання. Цей час очікування залежить від таких факторів, як інтенсивність потоків, час обробки пакетів у системі масового обслуговування.

У кожний момент часу на кожному з портів вузлів комунікаційних модулів діє визначена одиниця генеруючого потоку з того набору одиниць, що відповідає цьому порту. Частота появи кожної з одиниць потоку залежить від інтенсивності. Таким чином, існують інтервали неприступності, протягом яких потоки не обслуговуються. Будь-який інший інформаційний пакет, що надійшов на вхід системи в цей період, губиться. Це приводить до зниження ефективності обробки інформаційно-керуючих потоків і зниженню

продуктивності мережі розподіленої системи керування. Тому актуальними є завдання визначення імовірнісної структури інформаційних потоків, що впливає на процес їхньої обробки. Крім того, необхідно оцінити інтенсивність сумарного потоку й побудовати автокореляційну функцію з метою аналізу впливу інтервалу часу недоступності, впродовж якого потоки не обслуговуються, на ефективність їхньої обробки.

На рис. 2.8 представлена функціональна схема системи обробки потоків.

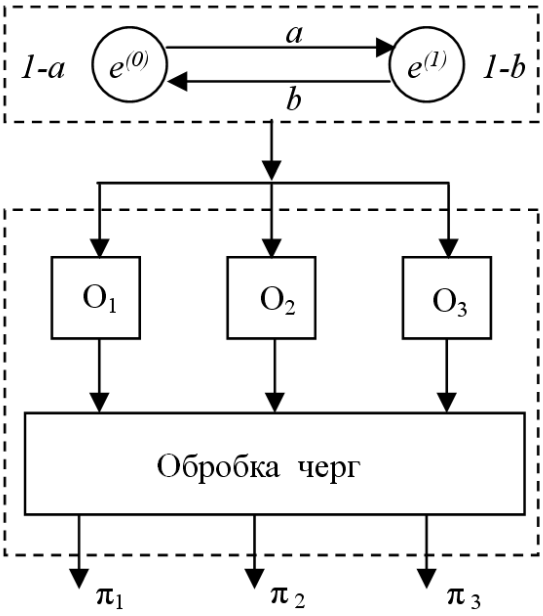


Рис. 2.8. Схема системи обробки потоків.

Вхідні потоки π_1, π_2, π_3 формуються засобами автоматизації автономної електроенергетичної системи, стан і режим роботи якої визначає імовірнісну структуру цих потоків. Заявки вхідних потоків надходять у накопичувачі (черги) O_1, O_2, O_3 з необмеженими ємностями. Можна визначити наступні класи обслуговування потоків [72, 115]:

– потік π_1 є малоінтенсивним інформативним пріоритетним потоком;

– потік π_2 представляє собою малоінтенсивний потік;

– потік π_3 – пріоритетний потік найбільшої інтенсивності.

Якщо середовище перебуває в стані $e^{(0)}$, то вхідні потоки представляють собою потоки типу Пуассона (потоки окремих вимог). При стані

середовища $e^{(1)}$ вхідні потоки не є найпростішими. Далі розглядається випадок, коли система перебуває в стані $e^{(0)}$.

Інформативність потоку π_1 означає, що в динаміці роботи системи обслуговування враховується наявність заявок у накопичувачі O_1 і надходження вимог по цьому потоку. Його пріоритетність – необхідність оперативного обслуговування вступників вимог. Пріоритетність потоку π_3 означає, що при відсутності вимог по потоку π_1 (розрив) буде продовжено обслуговування по потоку π_3.

Довільний вхідний потік π_j описується векторною випадковою послідовністю $\{(\tau_i, \nu_i, \eta_{ji}), i \geq 0\}$, де η_{ji} – число заявок типу, ν_i, що надійшли на проміжку часу $[\tau_i, \tau_{i+1}]$ по цьому потку. Тип заявок визначений міткою ν_i.

Системи масового обслуговування (СМО) можуть бути розділені на два типи, а саме: на системи з очікуванням і системи із втратами. У першому випадку – заявка, що надійшла на вхід системи "чекає" черги на виконання, у другому – вона через зайнятість каналу обслуговування одержує відмову й губиться для СМО.

Для найпростішого потоку, яким можна вважати потік від локальних засобів керування, що надходять на маршрутизатор потоку, число n подій, що попадають на будь-який інтервал часу τ розподілено за законом Пуассона [60, 86]:

$$P(n,\tau) = \frac{(\lambda\tau)^n \exp(-\lambda\tau)}{n!}, n = 0,1,2... \qquad (2.5)$$

Імовірність того, що на інтервалі часу τ не з'явиться жодної події дорівнює:

$$P(0,\tau) = e^{-\lambda\tau}. \qquad (2.6)$$

Тоді ймовірність протилежної події:

$$P(T < \tau) = 1 - e^{-\lambda\tau}. \qquad (2.7)$$

Для моделювання пуассонівського потоку з інтенсивністю $\lambda(t)$, що визначає кількість інформаційних пакетів, прийнятих в одиницю часу, необхідно скористатися системою щільностей:

$$a_i(t_1,...,t_i) = \prod_{j=1}^{i} \lambda(t_j) \exp(-\int_{T_0}^{t_i} \lambda(x)dx) \quad (2.8)$$

.

Умовна щільність імовірності може бути визначена виразом:

$$f_i(t_{i/t_{i-1}},...,t_1) = \frac{a_i(t_1,...,t_i)}{a_{i-1}(t_1,...,t_{i-1})} \quad (2.9)$$

.

Підставивши (2.8) в (2.9) одержимо:

$$d_i = 1 - \exp(-\int_{t_{i-1}}^{t_i} \lambda(t)dt) \quad (2.10)$$

де t_1, …, t_i – моменти настання подій потоку.

Наявні точки t_1, …, t_i становлять реалізацію потоку Пуассона. Були проаналізовані два стаціонарних потоки Пуассона з різною інтенсивністю λ=10 і 20. Графіки інтенсивності цих потоків показані на рис. 2.9, а, б, відповідно.

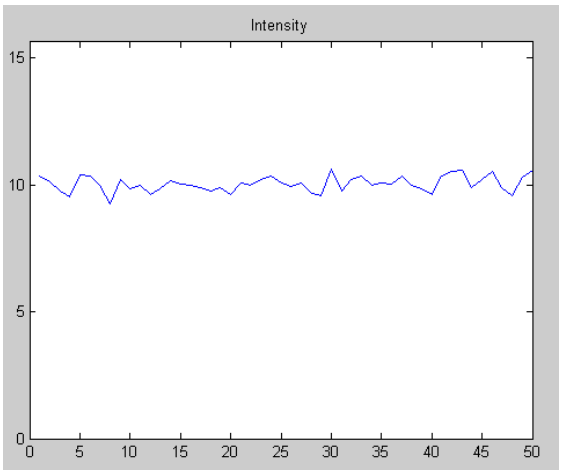
Для реалізації потоку були сгенеровані моменти настання подій t_1, t_2, … потоки з необхідною інтенсивністю. Далі було виконано наступний алгоритм::

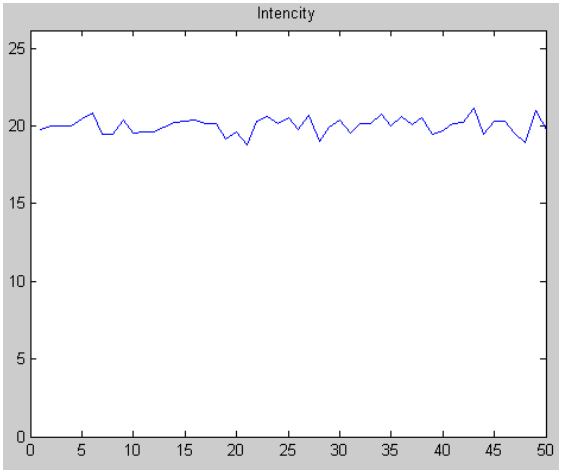
1. Реєструється перша подія: i_m=1, i=1;

2. Зберігається час появи події: $t_{im}=t_i$ і збільшується лічильник числа подій на одиницю: $i_m=i_m+1$;

3. Збільшуємо лічильник числа подій на одиницю: $i=i+1$;

4. Перевіряємо умову $t_i-t_{mim}-1>\tau$, якщо воно виконується, то виконується перехід до пункту 2, інакше до пункту 3.





а) б)

Рис. 2.9. Графіки інтенсивності потоків: а) λ=10; б) λ=20

Кореляційна функція (аналіз) має двояке застосування. З її допомогою визначається ступінь "подібності" двох потоків при різних значеннях зсуву за часом (крос-кореляція). Також вона використовується для визначення ступеня кореляції потоку із самим собою залежно від часу затримки (автокореляція) [9].

Кореляція – це нормована величина зі значенням відрізка [–1,1]. Для оцінки стаціонарного потоку застосовується автокореляційна функція:

$$R_j = \begin{cases} i = 1, \dfrac{1}{NWh^2} \sum\limits_{j=1}^{N} \nu_j (\nu_j - 1) \\ i \neq 1, \dfrac{1}{(N-i+1)Wh^2} \sum\limits_{j=1}^{N-i+1} \nu_j \nu_{j+i-1}) \end{cases} \qquad (2.11)$$

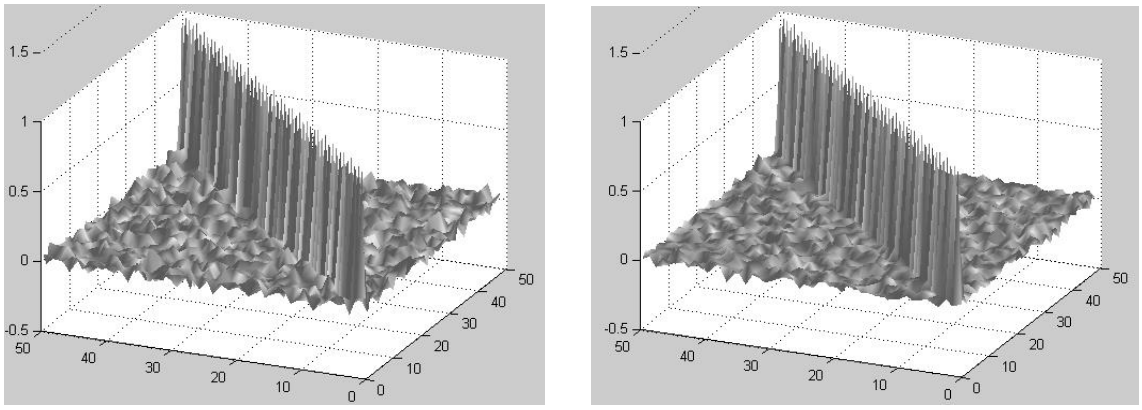
де N – кількість часових відрізків;

ν_{m} – кількість прийнятих пакетів у плині кожного часового відрізка;

W – число повторень процесу одержання реалізацій досліджуваного потоку;

h – крок розбивки часового відрізка.

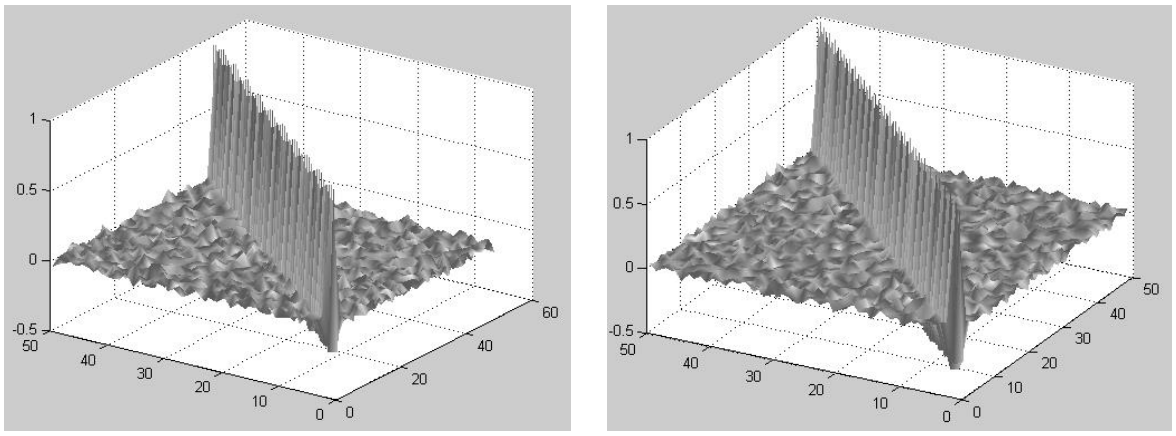
При виводі графіка по осі ординат відкладається значення R_{j} обчислене по формулі (2.11), а по осі абсцис – номера тимчасових відрізків. На рис. 2.10, а, б представлені матриці кореляційних коефіцієнтів для потоків з інтенсивностями λ=10 і 20 відповідно.



а) б)

Рис. 2.10. Кореляційні коефіцієнти при τ=0,01 а) λ=10, б) λ=20

Також був проаналізований вплив мертвого часу з різною величиною (τ=0,01 і 0,1).



а) б)

Рис. 2.11. Кореляційні коефіцієнти при τ=0,1 а) λ=10, б) λ=20

На рисунках видно, як величина «мертвої» зони праворуч і ліворуч від діагоналі збільшується зі збільшенням значення величини інтервалу неприступності системи обслуговування інформаційних потоків.

Одним з підходів рішення таких проблем є оптимізація використання всіх наявних ресурсів мережі. Зокрема, бажано гарантувати, щоб вузли мережі не були перевантажені, у той час як аналогічні вузли на альтернативних маршрутах недовантажені. Можливе використання алгоритмів зі зворотним зв'язком, що враховують наявність і розмір черг, швидкості надходження пакетів, інтервал між послідовними пакетами, у яких використовується тільки інформація про вхідні потоки й потоки насичення. Такий режим керування, у якому обслуговування потоків вимог відбувається строго по заздалегідь визначеному закону, найчастіше застосовується в системах обслуговування з великим завантаженням, коли інтенсивності надходження вимог по різних потоках практично однакові [31]. Проте, у випадку появи в потоках розривів, циклічний спосіб керування є не доцільним: для деякого потоку обслуговуючий пристрій працює в холостому режимі, у той час як по інших потоках є черги заявок на обслуговування. У

таких випадках раціональніше застосовувати інші керуючі алгоритми, що використовують додаткову інформацію про структуру вхідних потоків вимог.

2.7. Стаціонарні характеристики системи масового обслуговування

Для побудови моделей продуктивності обчислювальних систем широко використовується апарат теорії масового обслуговування [59, 60]. Пакети даних при передачі з нижніх рівнів на верхні проходять через комутуючі пристрої. У загальному випадку, потік заявок (пакетів) приходить у випадкові моменти часу. Найбільш характерний момент функціонування систем масового обслуговування – наявність черг, у яких заявки очікують звільнення ресурсу (каналу зв'язку). Аналіз мікропроцесорної мережі розподіленої системи керування як системи масового обслуговування дозволяє визначити кількість пакетів, які перебувають у мережі на різних стадіях обслуговування, час очікування в чергах, час передачі пакета й інші характеристики процесу обробки потоків.

Мережа складається з M ідентичних станцій. Інформація між ними передається кадрами, кожна станція може мати не більше одного кадру. Затримка поширення сигналу по передавальному середовищу дорівнює нулю. Тривалість передачі кадрів у канал має експонентний розподіл з параметром $0<\mu<\infty$.

Розглядається однолінійна система масового обслуговування (СМО). У систему надходить примітивний потік первинних заявок від M джерел з інтенсивністю $G_n=\lambda(M-n)$, що залежить від числа заявок у системі $n, n=\overline{0,M}$, де λ – інтенсивність генерування первинної заявки окремим джерелом. При надходженні в систему первинна заявка поступає на обслуговування, якщо прилад вільний, і стає до нього в чергу – у противному випадку. Заявки, що перебувають у черзі, є готовими. Заявка на приладі обслуговується без переривань протягом експонентного проміжку часу з параметром μ. Після завершення обслуговування заявка залишає систему. Якщо в цей момент у системі немає готових заявок, то прилад звільняється. У

противному випадку кожна з них у момент відходу обслугованої заявки з імовірністю p надходить на прилад миттєво, а з додатковою ймовірністю $(1-p)$ – через час такту. Нехай на прилад надходить j готових заявок. Якщо $j=0$, то прилад залишається вільним, а число готових заявок у системі не міняється. Якщо $j=1$, то прилад починає обслуговувати заявку, що надійшла, а число готових заявок у системі зменшується на одиницю. Якщо $j>1$, то в момент відходу обслуженої заявки прилад миттєво переходить у конфліктний стан, j – готових заявок, що надійшли на прилад, стають повторними, а число готових заявок у системі відповідно зменшується на j. Прилад зайнятий конфліктом протягом експонентного часу з параметром γ.

Визначимо стаціонарні характеристики розглянутої системи масового обслуговування.

1. Середнє число заявок :

$$\overline{M}=\sum_{m=0}^{M} n\left(\sum_{n=0}^{n} p_0(n,p)+\delta_n^{-0}\sum_{m=0}^{n-1} p_1(n,p)+\delta_n^{-0}\delta_n^{-1}\sum_{m=0}^{n} p_2(n,p)\right), \qquad (2.12)$$

де $\delta_j^i=0$ якщо $i\neq j$, $\delta_j^i=1$ якщо $i=j$, $\delta_j^{-1}=1-\delta_j$.

2. Ймовірність того, що прилад системи масового обслуговування не зайнятий:

$$P_0=\sum_{n=2}^{M}\sum_{m=0}^{n} p_0(n,p),$$

де $p_0(n,p)$ – стаціонарна ймовірність того, що прилад вільний, у системі n заявок, з низ p готових.

Ймовірність того, що прилад системи масового обслуговування зайнятий обслуговуванням заявки:

$$P_1=\sum_{n=1}^{M}\sum_{m=0}^{n-1} p_1(n,p),$$

де $p_1(n,p)$ – стаціонарна ймовірність того, що прилад зайнятий обслуговуванням, у системі n заявок, з низ p готових.

Ймовірність того, що прилад системи масового обслуговування зайнятий конфліктом:

$$P_2 = \sum_{n=2}^{M} \sum_{m=0}^{n} p_2(n, p),$$

де $p_2(n, p)$ – стаціонарна ймовірність того, що прилад зайнятий конфліктом, у системі n заявок, з низ p вільних.

3. Середня швидкість обслуговування заявок дорівнює середньому числу, обслужених в одиницю часу:

$$\lambda = \mu \sum_{n=1}^{M} \sum_{m=0}^{n-1} p_1(n, p) = \mu P_1 .$$

З урахуванням (2.12) маємо $\overline{\lambda} = \lambda\left(M - \overline{M}\right)$.

4. Середня продуктивність системи масового обслуговування, дорівнює середньому числу обслужених заявок за середній час обслуговування однієї заявки:

$$\overline{c} = \overline{\lambda} / \mu = P_1 .$$

5. Пропускна здатність системи:

$$\overline{c}_{\max} = \max_{\lambda} \overline{c} .$$

6. Середній час перебування заявок у системі:

$$\overline{W} = \overline{M} / \overline{\lambda} ..$$

2.8. Вплив зміни параметрів інформаційних потоків на ефективність роботи мережі

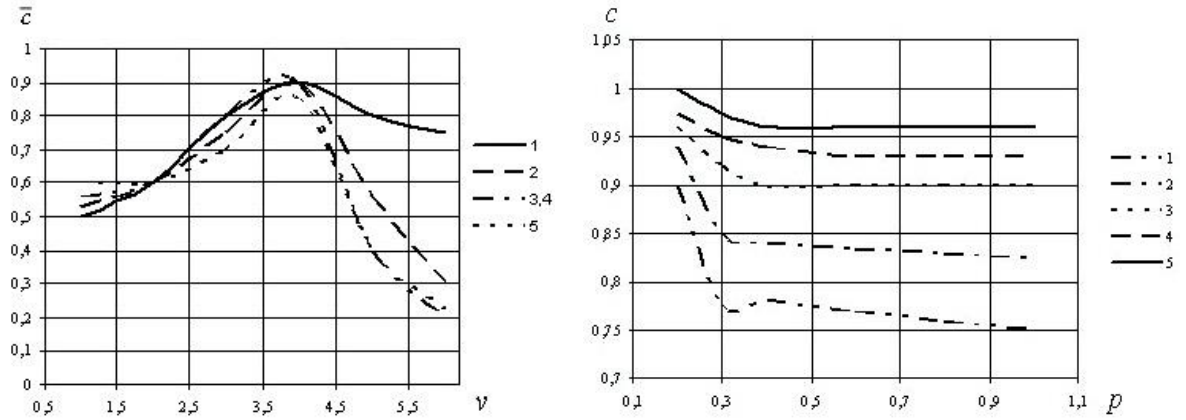
Отриманий вище алгоритм розрахунку стаціонарного розподілу системи масового обслуговування і її характеристик реалізований мовою С++.

Нехай швидкість передачі по каналу дорівнює c=10 Мбіт/с, довжина міжкадрового інтервалу τ_0 =9,6·10^{-6} с, службова інформація в кадрі займає B_0=12 байт, тривалість повідомлення про конфлікт дорівнює передачі B_c=6 байт, тривалість такту τ_1=51,2·10^{-6} с, тривалість затримки повторної передачі кадрів $\tau_1=\tau_2 \cdot R_n$, де R_n – ціле випадкове число, рівномірно розподілене на відрізку [0,2^{k-1}], k=min(n,10), n-порядковий номер конфлікту для даного

кадру. Звідси середня затримка $M_{\tau 2}$ повторної передачі перебуває в межах $\tau_{min}<M_{\tau 2}<\tau_{max}$, $\tau_{min}=2{,}56\cdot10^{-6}$ с, $\tau_{max}=13{,}1\cdot10^{-3}$ с. Тривалістю поширення сигналу по каналу можна зневажити через малу відстань між вузлами мережі (одиниці-десятки метрів).

Нехай дані в кадрі становлять B байт. Тоді середня тривалість обслуговування заявок дорівнює $\mu^{-1}=8(B_0+B)/з+\tau_0$, а середня тривалість конфлікту дорівнює $\gamma^{-1}=8B_c/c+\tau_0$ або $\gamma=7\cdot104$ с$^{-1}$. Середня затримка повторної передачі $\nu^{-1}\in[\tau_{\min},\tau_{\max}]$, середня тривалість такту $\alpha^{-1}=\tau_1$, тобто $\alpha=2\cdot10^4$ с$^{-1}$.

На рис. 2.12, а, показана залежність продуктивності мережі від інтенсивності повторної передачі при різних значеннях імовірності p, довжині кадрів B=64, числі станцій M=10, $\lambda=10^4$ с$^{-1}$. Залежність пропускної здатності від імовірності p показана на рис. 2.12, б. З нього видно, що пропускна здатність каналу збільшується зі зменшення ймовірності p. Це пояснюється тим, що при малих імовірностях p зменшується ймовірність конфлікту, і тому мережа може функціонувати при більших значеннях інтенсивності надходження нових кадрів λ, при яких і досягається висока пропускна здатність мережі.

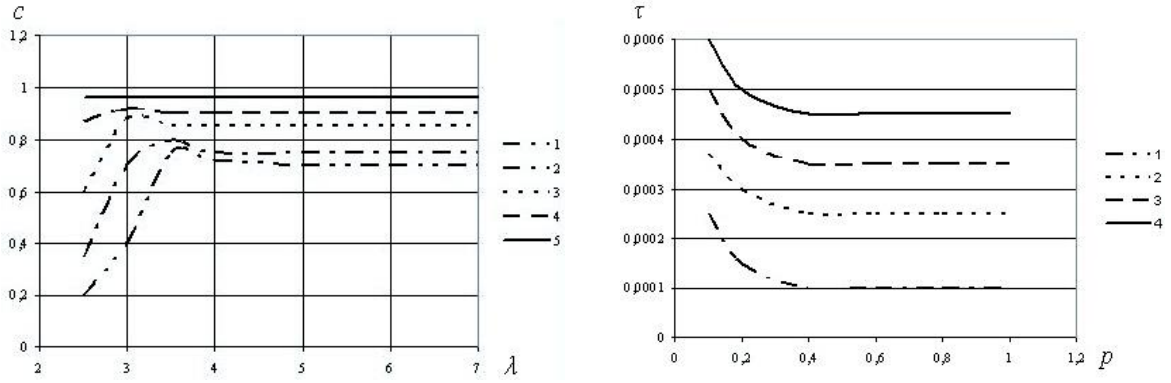


а) б)

Рис. 2.12. Залежність продуктивності: а – від інтенсивності повторення ν при значеннях p: 0,2 (1); 0,4 (2); 0,6 (3); 0,8 (4); 1,0 (5); б – залежність пропускної здатності від імовірності p при різній довжині кадрів B: 64 байт (1); 128 байт (2); 256 байт (3); 512 байт (4); 1024 байт (5)

Як видно з рис. 2.12, а, при малих і більших значеннях інтенсивності повторної передачі ν продуктивність каналу мережі падає. З рис. 2.12, б, видно, що пропускна здатність каналу збільшується зі зменшенням імовірності *p*. Це пояснюється тим, що при малих імовірностях *p* зменшується ймовірність конфліктів і тому мережа може функціонувати при більших інтенсивностях надходження нових кадрів, при яких досягається висока пропускна здатність мережі.

На рис. 2.13, а, показаний вплив на продуктивність мережі довжини кадрів. Тут M=10, p=0,5. На рис. 2.13, б, представлена залежність середньої затримки $\overline{w}$ від імовірності p при різних навантаженнях у каналі $p=\lambda M/\mu$, M=10, B=64 байт.



а) б)

Рис. 2.13. Залежність продуктивності: а – від інтенсивності λ при різній довжині кадрів B: 64 байт (1); 128 байт (2); 256 байт (3); 512 байт (4); 1024 байт (5); б – залежність середньої затримки від імовірності P при різних значеннях завантаження p: 0,5 (1); 1,0 (2); 1,5 (3); 2,0 (4)

При різкому зростанні рівня потоків у мережі виникають перевантаження, тобто блокування надходження повідомлень у мережу. Однією з причин, що приводять до зниження продуктивності, є зайнятість систем обробки черг, які являють собою системи масового обслуговування.

Аналіз отриманих залежностей показує, що при малих і великих значеннях інтенсивності повторної передачі продуктивність каналу мережі

падає й існує таке значення інтенсивності, при якій продуктивність мережі є максимальною.

РОЗДІЛ 3
МОДЕЛІ ТА АЛГОРИТМИ КЕРУВАННЯ ІНФОРМАЦІЙНИМИ ПОТОКАМИ

3.1. Мінімізація та упорядкування зв'язків по рівням завантаженності інформаційних каналів

На рис. 3.1 представлений варіант організації мікропроцесорної мережі розподіленої системи керування електроенергетичною системою. Кожний функціональний пристрій виконаний у вигляді окремого модуля. Організація взаємодії приладів може бути виконана із застосуванням обчислювальної мережі CAN або RS485, Ethernet.

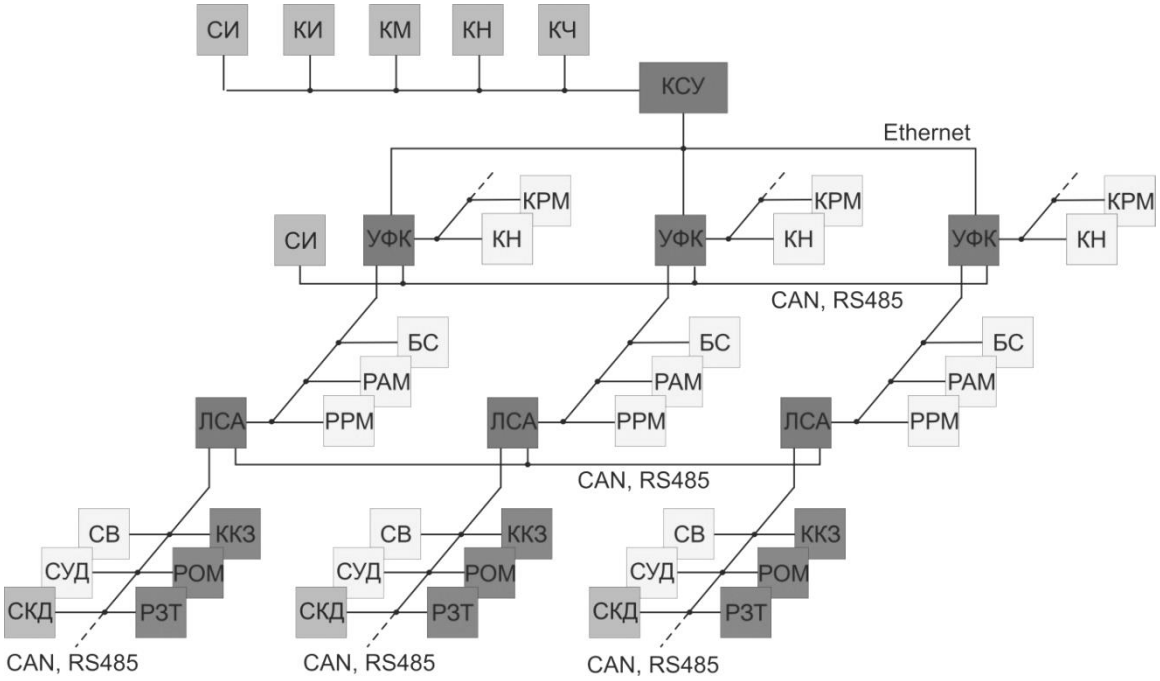


Рис. 3.1. Структура мікропроцесорної мережі системи керування електроенергетичною установкою

На рисунку використані наступні позначення: СКД – система контролю параметрів дизельного агрегату; СКД – система керування дизельним агрегатом; СЗ – система збудження синхронного генератора; РЗС – реле захисту по струму; РЗП – реле зворотної потужності; ЗКЗ – захист від

короткого замикання; ЛЗА – маршрутизатор потоку від локальних засобів автоматизації; РРП, РАП – пристрої розподілу реактивної й активної потужностей; БС – блок синхронізації генераторів; КФК – маршрутизатор потоків від пристроїв керування функціональними комплексами; СИ – засоби аварійної сигналізації; КН – пристрій контролю напруги; КРП – пристрій контролю резерву потужності; КСК – комплексна система керування; КІ – блок контролю ізоляції; КП – блок контролю потужності; КН – блок контролю напруги; КЧ – блок контролю частоти

Для дослідження потоків у мікропроцесорній мережі використано граф, представлений на рис. 3.2.

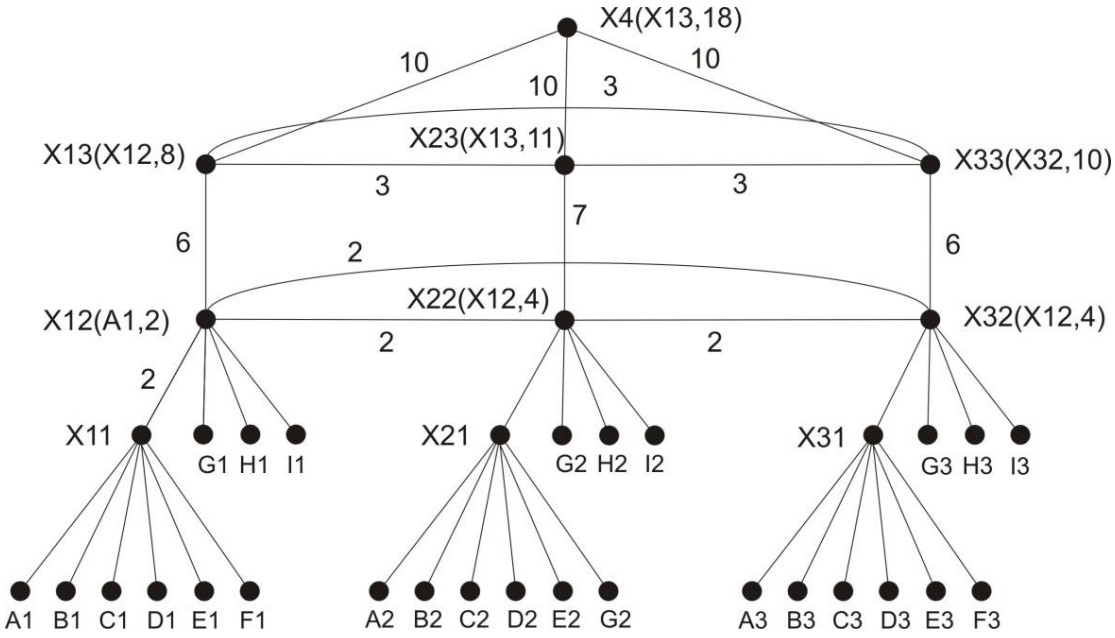


Рис. 3.2. Граф мережі керування

Для побудови моделей продуктивності обчислювальних систем широко використовується апарат теорії масового обслуговування. Пакети даних при передачі з нижніх рівнів на верхні проходять через комутуючі пристрої. Аналіз мікропроцесорної мережі розподіленої системи керування як системи масового обслуговування дозволяє визначити кількість пакетів, які перебувають у мережі на різних стадіях обслуговування, час очікування в чергах, час передачі пакета й інші характеристики процесу обробки потоків.

Система масового обслуговування складається із вхідного потоку заявок *a*, черги *Q*, дисципліни обслуговування *D*, що визначає порядок вибору заявок із черги, і обслуговуючого приладу $П_k$, або однаковим обслуговуючим приладам (каналів) $П_1 ... П_k$. Функціонування системи полягає в постановці вступників заявок у чергу на обслуговування, виборі із черги заявки, що підлягає першочерговому обслуговуванню, і наданні каналу заявці на певний час. Таким чином, система масового обслуговування характеризується наступним набором параметрів:

1) розподілом тривалості інтервалів між заявками вхідного потоку *p(a);*

2) дисципліною обслуговування заявок *D;*

3) числом обслуговуючих приладів (каналів) *K;*

4) розподілом тривалості обслуговування заявок приладами (каналами) *p(b).*

Зазначений набір параметрів повністю визначає порядок функціонування системи. Процес функціонування кількісно оцінюється наступним набором основних характеристик:

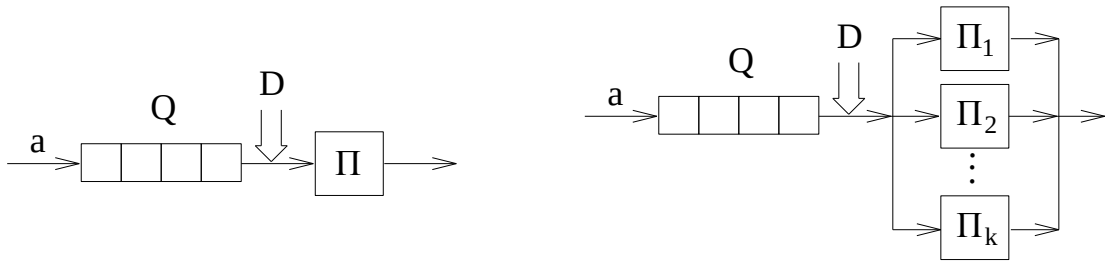
1) завантаженням – середнім числом приладів, зайнятих обслуговуванням (для одноканальної системи завантаження визначає проміжок часу, протягом якого прилад зайнятий обслуговуванням, тобто не простоює);

2) довжиною черги – числом заявок, які очікують обслуговування;

3) числом заявок, які перебувають у системі (у черзі й на обслуговуванні приладами);

4) часом очікування заявки – від моменту надходження заявки в систему до початку обслуговування;

5) часом перебування заявки в системі – від моменту надходження заявки до закінчення її обслуговування, тобто до виходу із системи.



а) б)

Рис. 3.3. Одноканальна (а) і багатоканальна (б) система масового обслуговування

Поруч із основними характеристиками для оцінки функціонування системи використовуються додаткові характеристики: тривалість простою, час безперервної зайнятості приладів та ін [60].

Всі зазначені характеристики, крім завантаження – випадкові величини, які представляють відповідними розподілами: розподілом довжини черги, числа заявок у системі, часі очікування і т.д. На більш низькому рівні деталізації випадкові величини характеризуються середніми значеннями й дисперсіями.

На початковій стадії проектування систем, найбільш важливим є інформаційне обстеження об'єкта керування, основне завдання якого – визначення інформаційних потоків.

Запропонований підхід дозволяє досліджувати потоки інформації, виявити й побудувати інформаційні зв'язки між об'єктами, а також побудувати інформаційну модель вигляду:

$$N = \{I_O, I_P\},$$

де: I_O, I_P – обсяги оперативної й постійної інформації.

На підставі отриманих моделей узагальнюються окремі інформаційні потоки $Z_k(t_i)$; $П_j(t_i)$;…; $A_l(t_i)$у групи:

$$I_g = \sum_{g=1}^{n_1} Z_g(t_i);\ I_j = \sum_{j=1}^{n_2} \Pi_j(t_i);\ I_l = \sum_{l=1}^{n_m} A_l(t_i).$$

Отримані групи інформації класифікуються на вихідні, транзитні й вхідні: X_1, X_2, X_3...X_n. У результаті будується неупорядкована графа-схема інформаційного потоку розподіленої системи керування (рис. 3.4).

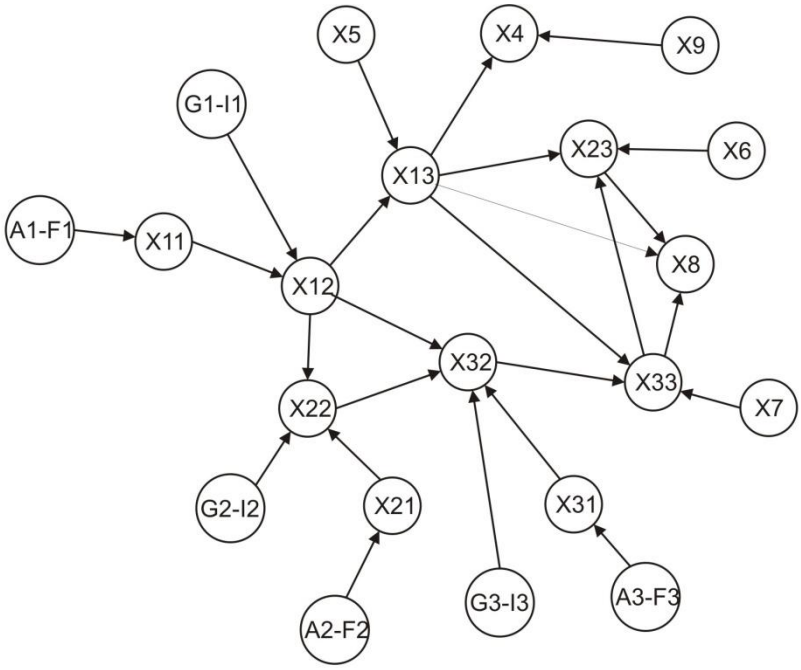


Рис. 3.4. Граф-схема інформаційних потоків

Для упорядкування інформаційних потоків виконується їхня мінімізація за допомогою алгоритму [103] (рис. 3.5), на базі якого розроблено програмне забезпечення для визначення оптимальних інформаційних зв'язків у проектованій системі.

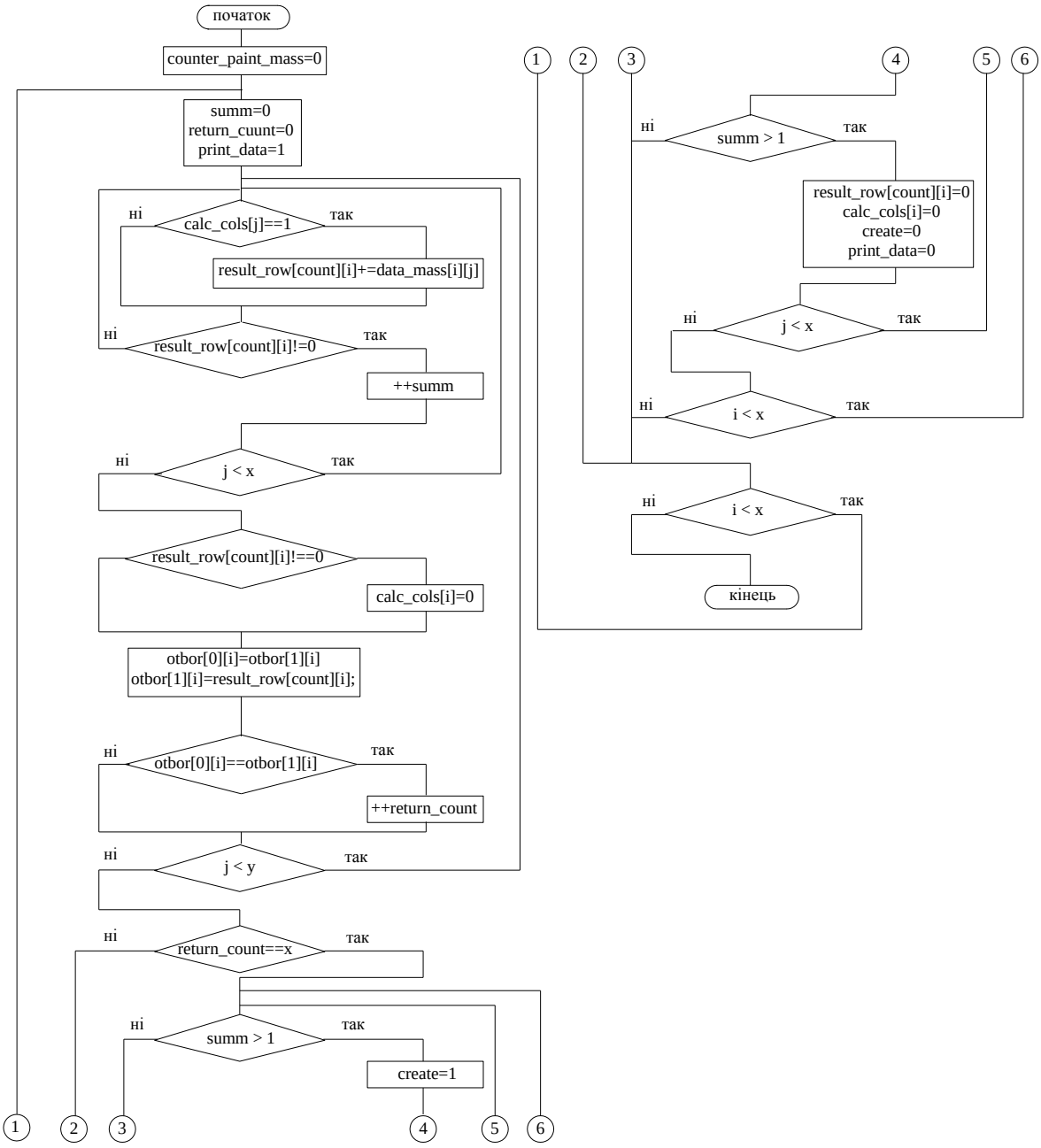


Рис. 3.5. Алгоритм мінімізації

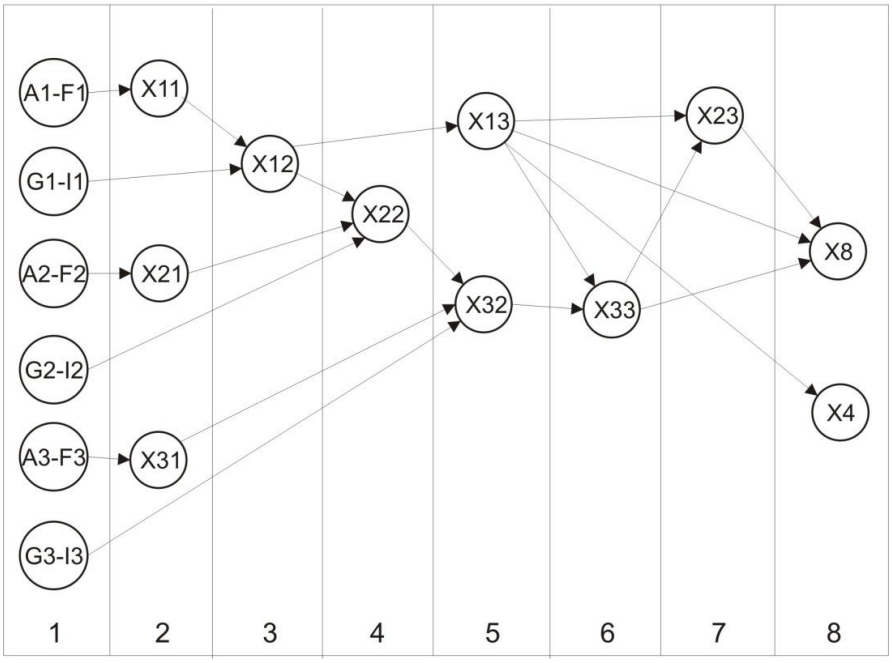


Рис. 3.6. Упорядкована граф-схема

Після цього етапу інформаційні потоки систематизуються по групам інформації. Після одержання певних груп інформації, з'являється потрібність у визначенні ступеня значимості їх для користувачів різних рівнів. Для цього доцільно застосувати теорію нечітких множин, де ступінь належності елемента множині може бути будь-яким числом одиничного інтервалу [0, 1], що забезпечує можливість формалізації нечітких, "розмитих" понять, у яких перехід від неналежності до належності відбувається поступово. Формалізація нечітких понять за допомогою теорії нечітких множин виявилася винятково плідною й при розробці принципів штучного інтелекту, що моделюють процеси мислення, визначення оцінок і прийняття рішень [46].

3.2. Оптимальна стратегія доступу до каналу зв’язку

Для забезпечення високої надійності системи керування мережа повинна мати високу зв’язність. При відмові окремих компонентів загальна працездатність системи керування повинна зберігатися. Вибір топології мережі багато в чому визначається наявністю додаткових зв'язків між

вузлами. Так, лінійна топологія, що являє собою одномірний масив, об'єднаний у ланцюжок, володіє рядом недоліків: час пересилання між двома вузлами залежить від відстані між ними, а відмова одного з вузлів приводить до неможливості посилки повідомлення. Використання кільцевої топології також недоцільне, оскільки вона не має можливість нарощування (додавання нового вузла вимагає демонтажу мережі), навантаження на канал передачі між різними вузлами буде різним, відсутні резервні зв'язки [100, 101]. Найбільш універсальною топологією є повнозв’язна топологія, оскільки має високу надійність, можливість передачі пакетів по різних каналах, можливість побудови різних топологій на логічному рівні [29]. Разом з тим, використання повнозв’язної топології ускладнено через необхідність використання великої кількості апаратних засобів для організації зв'язку між пристроями (комутатори, розширювачі портів, хаби й т.п.). Тому зручно використати радіоканал для організації зв'язку між вузлами мережі. Однак у такому випадку постає задача організації доступу до радіоканалу. При одночасній передачі даних від двох і більше вузлів виникає колізія й робота мережі стане неможливою. В якості пристроїв, що виконують передачу й прийом даних, використовуються пакетні радіомодеми. Особливість таких модемів полягає в тому, що можлива передача на різних частотах, які встановлюються програмно, тобто мережа володіє декількома каналами.

По наявним з доступних літературних джерел даним на даний час відсутня єдина методика проектування таких мереж. У роботах [86, 184] відзначається те, що мало досліджені питання оптимізації функціонування мереж інтегрального обслуговування, а також розробки алгоритмів керування процесами обміну інформацією в мережі. Складність рішення цих завдань в основному пов'язана з тим, що запити відрізняються один від одного не тільки інтенсивністю надходження й вимогами до якості обслуговування, але й числом каналів, викликаних у маршруті [29]. У роботі [115] розглядаються питання оптимальної маршрутизації інформаційних потоків, однак не враховується, що між вузлами є тільки один канал передачі.

Короткий огляд питань, присвячених дослідженню моделей мультиресурсних систем можна знайти в [86]. У роботі [139] відзначено, що в якості критерія оптимізації функціонування мережі може виступати максимізація завантаження каналів мережі. Однак у такому випадку постає задача організації доступу до радіоканалу. При одночасній передачі даних від двох і більше вузлів виникає колізія й робота мережі стане неможливою. В якості пристроїв, що виконують передачу й прийом даних, використовуються пакетні радіомодеми. Особливість таких модемів полягає в тому, що можлива передача на різних частотах, які встановлюються програмно, тобто мережа володіє декількома каналами.

Далі досліджується модель мережі з режимом комутації каналів і фіксованою маршрутизацією. У ній запит, що надійшов, вимагає для обслуговування одночасно декілька каналів у кожній лінії, що входить у його маршрут. При цьому, якщо хоча б в одній лінії маршруту відсутня необхідна кількість вільних каналів, то запит, що надійшов, одержує відмову. У більшості відомих робіт передбачається використати просту стратегію доступу, відповідно до якої запит будь-якого типу приймається в мережу, якщо в момент його надходження є достатнє число каналів у кожній лінії, що входить у його маршрут. Тип запиту визначається числом каналів, викликаних їм для обслуговування. Однак очевидно, що проста стратегія доступу не дозволяє оптимізувати функціонування мережі щодо обраного критерію якості, при цьому найчастіше критерієм є максимізація завантаження каналів мережі [184]. У такому випадку інформаційні потоки можуть бути розподілені як по різним маршрутам, так і по різним каналам.

Нехай на вхід N-канальної системи, $N \geq 2$, надходять різнотипні заявки K типів, $K \geq 2$. Заявки k-го типу (k-заявки) надходять відповідно до пуассонівського закону з інтенсивністю λ_k, $k = \overline{1, K}$. Для обслуговування k-заявки буде потрібно одночасно b_k каналів, $b_k \geq 1$. Припустимо, що $1 \leq b_1 \leq b_2 \leq ... \leq b_k \leq N...$ Канали повністю комутуються, тобто для обслуговування k-заявки можуть бути виділені будь-які b_k вільні канали. Заявки k-го типу

займають всі b_k каналів на весь період обслуговування, іншими словами, у момент закінчення обслуговування k-заявки звільняються всі b_k каналів одночасно. Тривалість займання каналів k-заявкою є випадковою величиною, підлеглою експоненціальній функції розподілу з параметром μ_k, $k = \overline{1, K}$; при цьому $\mu_k \neq \mu_k$, при $k \neq k'$. Множина можливих станів системи, тобто фазовий простір станів (ФПС), задається наступним виразом:

$$E_{\text{cs}} := \{n : (n,b) \leq N\}, \tag{3.1}$$

де $b = (b_1, \ldots, b_k)$ і (n,b) означає скалярний добуток векторів n і b, тобто

$$(n, b): \sum_{k=1}^{K} n \cdot p_k \ . \tag{3.2}$$

Запис $a := b$ означає, що α визначається виразом b. Максимальне число заявок у системі при простій стратегії доступу дорівнює $[N/b_1]$, ($[x]$ означає цілу частину x).

Стаціонарну ймовірність стану $n \in E_{CS}$ позначимо через $p(n)$. Відомо, що при простій стратегії доступу [4] стаціонарний розподіл $\{p(n): n \in E_{CS}\}$ системи має мультиплікативний вигляд:

$$p(n) = G^{-1}(E_{CS}) \prod_{k=1}^{K} \beta_k(n_k), \tag{3.3}$$

де

$$\beta_k(n_k) := \frac{\prod_{i=1}^{n_k - 1} \lambda_i}{n_k! \, \mu_k^{nk}}, \tag{3.4}$$

$$G^{-1}(E_{CS}) := \sum_{n \in E_{CS}} \prod_{k=1}^{K} \beta_k(n_k). \tag{3.5}$$

Необхідно знайти таку стратегію доступу, при якій було б можливим максимізувати завантаження каналів системи, тобто метою дослідження є рішення завдання досягнення

$$\max H(E_{cs}), \tag{3.6}$$

де

$$H(E_{CS}) := \sum_{n \in E_{CS}} (n, b)\, p(n). \tag{3.7}$$

Стратегія доступу, що є рішенням рівняння (3.6), буде оптимальною. З метою рішення рівняння (3.6) необхідно ввести керуючі ситуаційні параметри, що оцінюють імовірності прийняття різних рішень у моменти надходження заявок у систему. Заявка будь-якого типу, що надійшла в момент, коли в системі відсутня необхідна кількість каналів, з імовірністю 1 одержує відмову.

Для побудови моделей продуктивності обчислювальних систем широко використовується апарат теорії масового обслуговування. Пакети проходять через комутуючі пристрої. У загальному випадку, потік заявок приходить у випадкові моменти часу. Найбільш характерний момент функціонування систем масового обслуговування – наявність черг, у яких заявки очікують звільнення ресурсу (каналу зв'язку). Система масового обслуговування складається з вхідного потоку заявок *a*, черги *Q*, дисципліни обслуговування *D,* що визначає порядок вибору заявок із черги, і обслуговуючого приладу ПК, або однаковим обслуговуючим приладам (каналів) $П_1 ... П_k$. Функціонування системи полягає в постановці заявок, що надходять у чергу на обслуговування, виборі із черги заявки, що підлягає першочерговому обслуговуванню, і наданні канала заявці на певний час.

На рис. 3.3, б, представлена структура багатоканальної системи масового обслуговування, за допомогою якої виконується обробка пакетів у вузлах мережі.

Розглянемо моменти надходження *k*-заявки. Нехай у цей момент система перебуває в стані *n*, у якому $q(n) \ge b_k$ (у протилежному випадку, як було зазначено вище, *k*-заявка одержує відмову). Тоді можливе прийняття одного із двох рішень: або *k*-заявка приймається на обслуговування, або вона одержує відмову. Імовірності прийняття цих рішень позначені відповідно через $a_k^+(n)$ і $a_k^-(n)$. Визначені таким чином ситуаційні параметри задовольняють наступним умовам:

$$1 \ge a_k{}^{\pm}(n) \ge 0,\ q(n) \ge b_k,\ k = \overline{1, K},\ a_k^+(n) + a_k^-(n) = 1. \qquad (3.8)$$

Позначимо $\eta(n, n')$ інтенсивність потоку зі стану *n* у стан *n′*. Ця величина визначається як добуток інтенсивності переходу *n*→*n′* на стаціонарну ймовірність стану *n*. Тоді в прийнятих позначеннях $\eta(n, n')$, n, $n' \in E_{CS}$ визначаються в такий спосіб:

$$\eta(n, n') = \begin{cases} \lambda_k p(n) a_k^+(n), & \text{якщо } q(n) \ge b_k, n' = n + e \\ n_k \mu_k p(n), & \text{якщо } n' = n - e_k \\ 0, & \text{в інших випадках} \end{cases}. \qquad (3.9)$$

На підставі (3.9) складається система рівнянь рівноваги даної моделі:

$$\sum_{n' \in E_{CS}{}^-(n)} \eta(n, n') = \sum_{n' \in E_{CS}{}^+(n)} \eta(n, n'),\ n \in E_{CS}$$
$$\sum_{n \in E_{CS}} p(n) = 1, \qquad (3.10)$$

де $E_{CS}{}^-(n)$ – підмножина тих станів $n' \in E_{CS}$, у які можна потрапити зі стану n за один крок;

$E_{CS}{}^+(n)$ – підмножина тих станів $n' \in E_{CS}$ з яких можна потрапити в стан *n* за один крок.

Отже, у явному вигляді задача (3.6) записується в такий спосіб: потрібно знайти $\max(E_{CS})$ при обмеженнях:

$$\sum_{k=1}^{K} \lambda_k a_k + (n) I(b_k \le q(n)) + \sum_{k=1}^{K} [n_k \mu_k] p(n) = \sum_{k=1}^{K} \lambda_k p(n - e_k) a_k{}^+(n - e_k) u(n_k) +$$
$$+ \sum_{k=1}^{K} (n_k + 1) \mu_k p(n + e_k) I(b_k \le q(n)), \quad n \in E_{CS}; \qquad (3.11)$$

$$\sum_{m\in E_{CS}} p(n)=1;\ 0\le a_k{}^{\pm}(n)\le 1; \qquad a_k{}^{+}(n)+a_k{}^{-}(n)=1, \qquad \forall k\in F_{CS}(n), \tag{3.12}$$

де

$$u(x)=\begin{cases}1, & x\ge 0\\ 0, & x\le 0\end{cases}, \tag{3.13}$$

I(A) – індикатор події A; $F_{CS}(n):=\left\{k\in R_k^+ : b_k\le q(n)\right\}$, $R_k^+:=\{1,\ldots,K\}$.

Дана задача може бути вирішена з використанням методів лінійного програмування; вона має оптимальне рішення, відповідно до якого значення $a_k\pm(n)$ рівні або 0, або 1 для кожного $k\in F_{CS}(n)$. Остання обставина дозволяє побудувати алгоритм, що реалізує знайдену оптимальну, невипадкову стратегію доступу (рис. 3.7).

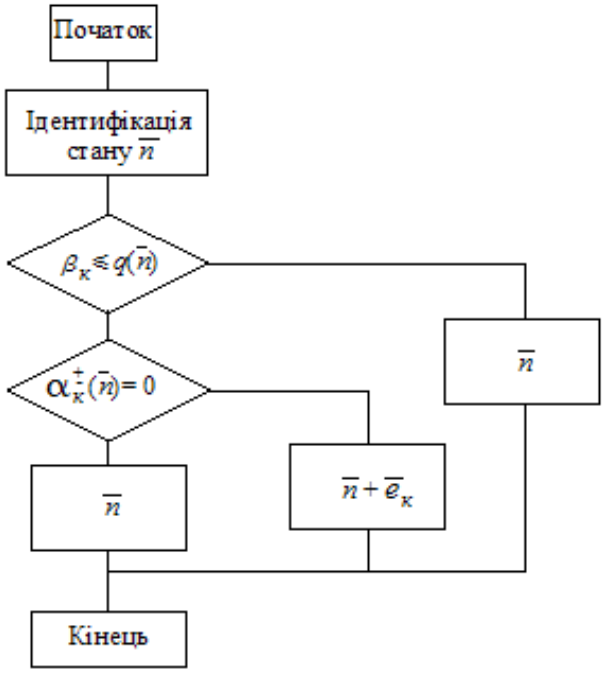


Рис. 3.7. Алгоритм оптимальної стратегії доступу

Використання лінійного програмування дозволяє враховувати й деякі додаткові лінійні обмеження, наприклад, обмеження на ймовірності втрати певних типів заявок.

3.3. Мінімізація колізій інформаційних потоків в мережах розподілених систем керування

Найбільш відомим механізмом керування локальної обчислювальної мережі є метод множинного доступу з контролем несучої й виявленням колізій (CSMA/CD), що використовується в мережах Ethernet.

Мережі з контролем несучої (з колізіями) є прикладом рівнорангових безприорітетних систем [132]. Цей підхід широко використовується в локальних обчислювальних мережах і, зокрема, знаходить застосування в рамках специфікації Ethernet і стандарту IEEE 802.3. У мережі з контролем несучої всі вузли вважаються рівноправними. Однак є можливість введення системи пріоритетів на основі різних підходів. Перш ніж почати передачу потрібно, щоб станції ”прослухали” канал і визначили, чи є канал активним. Якщо канал перебуває в стані спокою, будь-яка станція, що має дані для передачі, може послати свої кадри в канал. Аналогічна ситуація відбувається, коли замість каналу є інший загальний ресурс.

У мережах з контролем несучої практикується використання декількох методів захоплення. Одним їхнім методом є метод ”ненаполегливого” контролю несучої, що забезпечує всім вузлам можливість почати передачу негайно після того, як виявиться, що канал вільний. У випадку, якщо канал зайнятий, вузли вичікують випадковий період часу перед тим, як знову перевірити стан каналу. Іншим методом, що використовується в системах із квантуванням часу, є метод ”*p*–наполегливого” контролю несучої. Він передбачає для кожного вузла деякий алгоритм очікування (”*p*” означає ймовірність). Наприклад, вузли A и B не починають негайну передачу після того, як вони виявили, що канал вільний. У цьому випадку кожна станція викликає програму генерації випадкового числа – часу очікування (кілька мікросекунд). Якщо вузол виявляє, що канал зайнятий, вона вичікує деякий період часу й робить нову спробу. Він зробить передачу в канал, що звільнився, з імовірністю *p* і з імовірністю 1-*p* відкладе передачу.

Розглянемо випадок одного неподільного ресурсу – каналу зв'язку (рис. 3.8). Стан *i*-го процесу, у якому *i*-й процес очікує надання ресурсу, який вимагається декількома процесами, називається активним станом і позначається A_i^i. Стан, у якому *i*-й процес не вимагає надання ресурсу, називається пасивним станом і позначається A_0^i. Простір станів *i*-го процесу позначається $L_i = \{A_0^i, A_i^i\}$. Конфлікт виникає, якщо кілька процесів перебувають в активному стані.

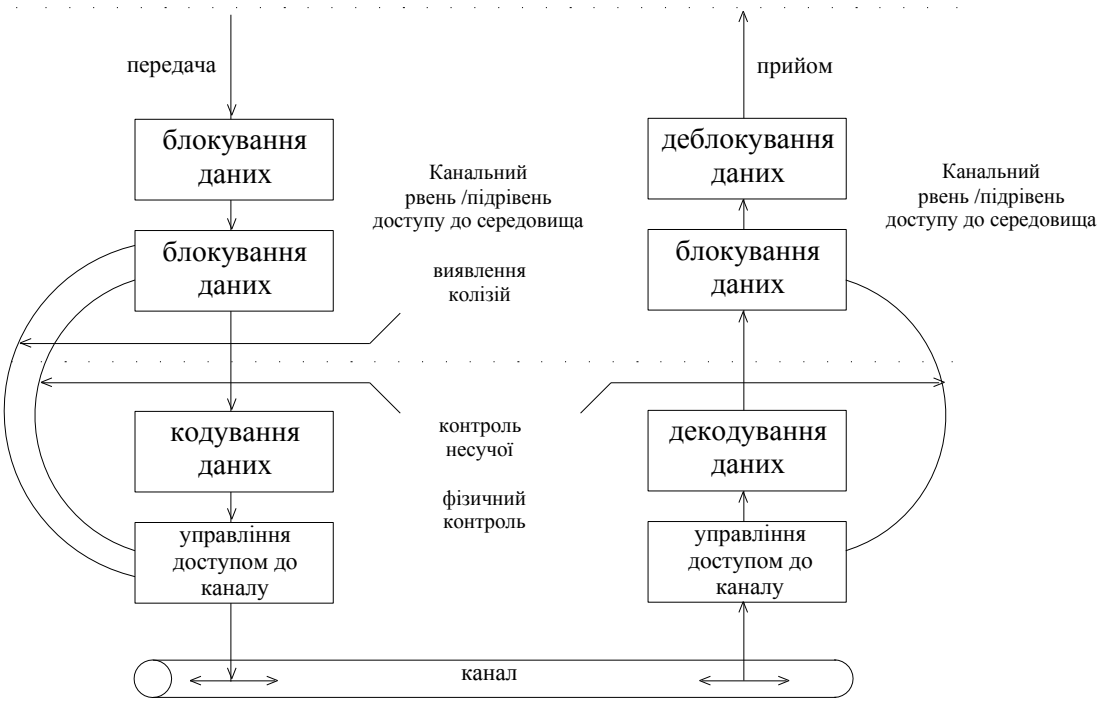


Рис. 3.8. Модель OSI для мереж CSMA/CD

Розв’язанням конфлікту для одного процесу вважається надання загального ресурсу одному з конфліктуючих процесів і затримку інших конфліктуючих процесів, що перебувають в активному стані. Вибір такого процесу визначається правилом *R* дозволу конфлікту.

Імовірність виходу *i*-го процесу з активного стану позначено:

$$P_i = P_{00}^i = P_{10}^i,$$

і в активний стан:

$$1 - P_i = P_{01}^i = P_{11}^i,$$

де i – номер процесу, індекс 0 позначає пасивний стан, індекс 1 – активний стан.

Оптимальним називається правило R_0, що мінімізує середню частоту конфліктів. Знайдемо оптимальне правило розв'язання конфліктів для критерію мінімізації середньої частоти конфліктів при нескінченному часі функціонування системи. У роботі [14] наведений критерій оптимізації, що має вигляд:

$$\psi^R(S,K) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} M_S^R \chi(S^t, K) \to \min_{\{R\}},$$

де M_S^R – математичне очікування при початковому стані $S \in L$;

R – правило дозволу конфліктів;

S^t – поточний стан сукупності процесів;

$\chi(S,K) = \begin{cases} 1, S \in K \\ 0, S \notin K \end{cases}$ – характеристична функція; K – множина станів у просторі L.

Оптимальним правилом дозволу конфліктів R_0, коли на один ресурс претендує кілька процесів, є правило, відповідно до якого при виникненні конфлікту ресурс надається конфлікту процесу, що має мінімальну ймовірність влучення в активний стан [131]. Даний результат має наступний фізичний зміст: при виникненні конфлікту варто віддавати перевагу тому з конфліктуючих процесів, що рідше вступає в конфлікт із іншими процесами.

Розглянемо випадок декількох неподільних ресурсів для двох паралельних процесів. Нехай два процеси виконуються паралельно, але існує кілька ресурсів, через які вони конфліктують. Така ситуація виникає в широко розповсюджених двопроцесорних обчислювальних системах із загальною пам'яттю.

Таким чином, у зазначеному випадку n=2, r=1÷ m (кількість ресурсів), i-й процес попадає в стан A_r^i при відсутності іншого процесу з імовірністю $P_r^i, \sum_{r=0}^{m} P_r^i = 1$, i=1,2. Конфліктна множина має вигляд:

$$K = \{S \in L,\ S_1 = A_r^1 \ \wedge\ S_2 = A_r^2,\ 1 \le r \le m\}.$$

Позначимо $\alpha_i^n(A_r^1, A_r^2)$ – середнє число конфліктів за *n* кроків виходячи зі станів (A_r^1, A_r^2), якщо на першому кроці перевага віддається *i*-му процесу, а потім використається оптимальне правило розв'язання конфліктів; $\alpha_0^n(A_r^1, A_r^2)$ – середнє число конфліктів за *n* кроків, виходячи зі стану (A_r^1, A_r^2), при оптимальному правилі розв'язання конфліктів. Відповідно до правила оптимальності Беллмана [14]:

$$\alpha_0^n(A_r^1, A_r^2) = \min(\alpha_1^n(A_r^1, A_r^2), \alpha_2^n(A_r^1, A_r^2)), \quad \textit{де}$$
$$\alpha_1^n(A_r^1, A_r^2) = 1 + P_r^1 \alpha_0^{n-1}(A_r^1, A_r^2) + (1 - P_r^1)\alpha_0^{n-1}(N);$$
$$\alpha_2^n(A_r^1, A_r^2) = 1 + P_r^2 \alpha_0^{n-1}(A_r^1, A_r^2) + (1 - P_r^2)\alpha_0^{n-1}(N);$$
$$\alpha_0^{n-1}(N) = \sum_{j=1}^{m} P_j^1 P_j^2 \alpha_0^{n-2}(A_j^1, A_j^2) + (1 - \sum_{j=1}^{m} P_j^1 P_j^2)\alpha_0^{n-2}(N).$$

У роботі [14] показано, що

$$n\alpha_0^n(A_r^1, A_r^2) \ge \alpha_0^n(N),$$

звідки при $P_r^2 \ge P_r^1$ одержимо:

$$\alpha_0^n(A_r^1, A_r^2) = \alpha_1^n(A_r^1, A_r^2),$$

Оптимальним правилом розв'язання конфліктів *R* для випадку, коли два процеси претендують на кілька ресурсів, є вищенаведене правило.

Ефективність оптимального правила можна оцінити аналітично. Для цього необхідно знайти середню частоту конфліктів при оптимальному правилі дозволу конфліктів. У роботі [84] показано, що

$$\psi^{R_0}(S, K) = \sum_{i=1}^{m} \pi_{R_0}(A_r^1, A_r^2),$$

де $\pi_{R_0}(A_r^1, A_r^2)$ – стаціонарна ймовірність стану при фіксованому правилі R_0.

Для ймовірностей $\pi_{R_0}(A_r^1, A_r^2)$, $i=1 \div m$ можна виписати наступну систему рівнянь з врахуванням нормування:

$$\sum_{i=1}^{m} \pi_{R_0}(A_r^1, A_r^2) + \pi_{R_0}(N) = 1,$$

де $N=L/K$, тобто неконфліктний стан.

$$\pi_{R_0}(A_r^1, A_r^2) = \min(P_r^1, P_r^2)\pi_{R_0}(A_r^1, A_r^2) + P_r^1, P_r^2 \pi_{R_0}(N),$$

звідки

$$\pi_{R_0}(A_r^1, A_r^2) = \frac{P_i^1 P_i^2}{1 - \min(P_i^1 P_i^2)} \pi_{R_0}(N),$$

і отже

$$\sum_{i=1}^{m} \pi_{R_0}(A_r^1, A_r^2) = 1 - \pi_{R_0}(N) = \left[\sum_{i=1}^{m} \frac{P_i^1 P_i^2}{1 - \min(P_i^1 P_i^2)} \right] / \left[1 + \sum_{i=1}^{m} \frac{P_i^1 P_i^2}{1 - \min(P_i^1 P_i^2)} \right].$$

У цьому випадку:

$$\psi^{R_0}(S,K) = \frac{\sum_{i=1}^{m} P_i^1 P_i^2 / \left[1 - \min(P_i^1 P_i^2)\right]}{1 + \sum_{i=1}^{m} P_i^1 P_i^2 / \left[1 - \min(P_i^1 P_i^2)\right]}.$$

Аналогічно можна одержати вираз для середньої частоти конфліктів S_c коли з рівною ймовірністю віддається перевага тому або іншому процесу. У цьому випадку

$$\psi^{S_c}(S,K) = \left[\sum_{i=1}^{m} \frac{2P_i^1 P_i^2}{2 - P_i^1 - P_i^2} \right] / \left[1 + \sum_{i=1}^{m} \frac{2P_i^1 P_i^2}{2 - P_i^1 - P_i^2} \right].$$

На рис. 3.9 представлені співвідношення середньої частоти конфліктів залежно від кількості конфліктуючих процесів.

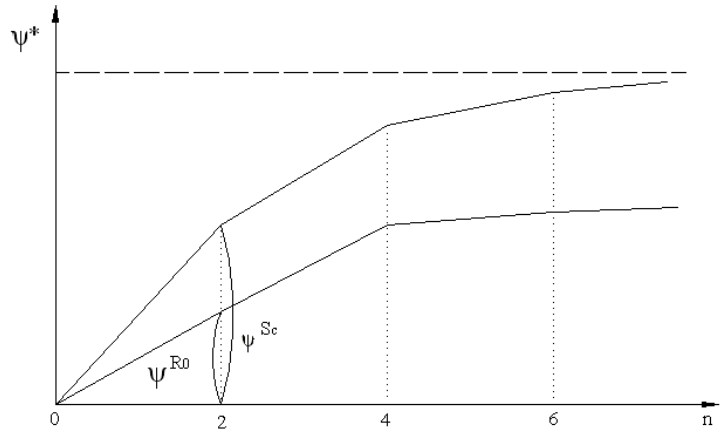


Рис. 3.9. Співвідношення середньої частоти конфліктів

Застосування оптимального правила для розв’язання конфліктів майже вдвічі знижує частоту конфліктів у порівнянні із правилом, випадково

(рівновірогідно) обираючим одні з конфліктуючих процесів. Включення описаного алгоритму диспетчеризації у функціональні стандарти прикладного й канального рівнів і побудова закінченого машинного алгоритму практично у два рази зменшить частоту конфліктів за ресурси інформаційно-обчислювальної системи й, у такий же спосіб, поліпшить її якість.

3.4. Алгоритм обробки інформаційних пакетів в маршрутизаторі

У мережі будь-який кінцевий вузол може обмінюватися даними з будь-яким іншим кінцевим вузлом. Фізичний шлях, через який здійснюється такий обмін, може проходити через багато проміжних вузлів. Для реалізації процесу маршрутизації необхідні:

1. Інформація для кінцевих вузлів.

2. Методи, що використовуються кінцевими вузлами й системами для збирання цієї інформації.

3. Методи, що використовуються кінцевими вузлами для розподілу цієї інформації.

4. Функції, що виконуються кінцевими вузлами над цією інформацією для визначення шляхів, через які піде потік даних.

Для роботи алгоритму маршрутизації використовується метрична інформація, що являє собою набір чисел, що характеризують якість шляху [100, 106]. Це дозволяє маршрутизатору порівнювати шляхи, дані про які він одержує від інших маршрутизаторів, і вирішувати, який з них використати. База маршрутної інформації повинна містити:

– таблиці маршрутної інформації, які можуть використовуватися для просування блоків даних ближче до одержувача;

– списки сусідніх кінцевих вузлів, які дозволяють оцінити локальну топологію;

– виміряні характеристики вимог до передачі, які дозволяють маршрутизатору адаптуватися до зміни цих вимог;

– карти мережі, які можуть використовуватися для обчислення найкоротших шляхів до адресованих пунктів доступу шляхом використання одного з багатьох показників маршрутизації.

Маршрутизатори створюють свої бази маршрутної інформації, збираючи інформацію зі свого локального оточення й від інших систем. Системи можуть повідомляти іншим системам дані, необхідні для їхніх баз маршрутної інформації, шляхом розподілу цих даних.

У запропонованому для реалізації алгоритмі маршрутизації виконується структурування метрики для роздільного опису часу затримки, пропускної здатності каналу, його надійності й завантаження. Основними вимогами до алгоритму є:

– стійка маршрутизація в складних мережах без утворення маршрутних петель;

– швидка реакція на зміни в топології мережі;

– малий власний трафік;

– можливість розбивки трафіка по декількох маршрутах, коли вони мають подібні характеристики;

– здатність обробляти різні типи сервісу;

– підтримка маршрутизації в мережах, побудованих на базі різних стандартів.

Метрична інформація містить у собі такі параметри:

– час затримки, що представляє собою інтервал часу, який потрібно для досягнення пакетом місця призначення при русі по ненавантаженому шляху;

– пропускну здатність самого вузькосмугового каналу;

– завантаження каналу, що являє собою частину пропускної здатності, яка використовується на даний момент;

– надійність, що являє собою долю пакетів, які досягають місця призначення неушкодженими.

В одній підмережі бажано спростити роботу кінцевих систем, поклавши функції маршрутизації на проміжні системи. Це робиться, оскільки

кінцевих систем буде набагато більше, ніж проміжних. Крім того, проміжні системи значною мірою спеціалізовані на функціях маршрутизації й ретрансляції. Для кінцевих систем імовірність бути підключеними до декількох підмереж менше, ніж для проміжних і, отже, у них менший вибір при маршрутизації.

На підставі метричної інформації для шляху повинна розраховуватися одна складна метрика, що поєднує різні компоненти в одне число, що представляє характеристику якості шляху. Пропонується вибирати шлях на підставі наступної метрики:

$$M = (K1/C + K2 \cdot t) \cdot R, \qquad (3.14)$$

де $K1$, $K2$ – константи для визначення вимог до обслуговування пакетів; $C3$ – поточна пропускна здатність каналу, тобто різниця між пропускною здатністю розвантаженого каналу й завантаженого каналу; t – затримка доставки пакета по незавантаженому шляху, що складається із затримки комутації й затримки передачі пакета; R – надійність каналу. Шлях, що має мінімальну метрику, буде оптимальним.

В якості критерія оптимізації обрано мінімальний час і вартість проходження інформаційних потоків по каналах зв'язку. Позначимо через $C_{ij} = C\{x_i, x_j\}$ пропускну здатність каналу зв'язку мережі, $t_{ij} = t(x_i, x_j)$ – час проходження одиниці по каналу зв'язку (x_i, x_j).

Нехай дана мережа з найбільшим потоком Ψ_Z^0 і потік $\Psi_Z < \Psi_Z^0$, що повинен бути переданий по мережі. Потрібно знайти такий розподіл потоку Ψ_Z по каналах мережі (дугам графа), що забезпечує мінімальний час проходження потоку. При цьому, для кожної дуги повинне виконуватися співвідношення $\Psi(x_i, x_j) \le C_{ij}$, а час проходження потоку $\Psi(x_i, x_j)$ по дузі (x_i, x_j) дорівнює $t_{ij} \times \Psi(x_i, x_j)$. Час доставки пакетів залежить від числа пакетів, що перебувають у вузлі мережі в режимі очікування передачі. Залежність часу від довжини дуги для мережі є досить складною, тому зроблено

припущення про детерміновану прив'язку часу передачі до довжини дуги, по якій передається даний пакет. Таким чином, для рішення завдання величини t_{ij} будуть розглядатися як довжини відповідних дуг. У цьому випадку час проходження потоку Ψ будь-яким шляхом λ від входу мережі до виходу з мережі Z буде дорівнювати добутку довжини цього шляху на значення цього потоку ψ, і задача мінімізації часу проходження потоку зведеться до пошуку найкоротшого шляху в графі від x_0 до Z. Якщо відсутні обмеження на пропускну здатність дуг, то найкоротшим шляхом буде той, котрий забезпечує мінімальний час проходження потоку.

При наявності обмеження на пропускну здатність дуг задача вирішується в кілька етапів шляхом знаходження часткових потоків на кожному етапі.

Для цього в графі $G_1 = (X, \Gamma)$, що зображує мережу з довжинами дуг $t_{ij} = l(x_i, x_j)$, потрібно знайти найкоротший шлях λ_1 від x_0 до Z. Нехай C_1 – пропускна здатність шляху λ_1. Цим шляхом пропускається інформаційний потік

$$\Psi_1 = \begin{cases} \Psi_Z, якщо \ \ \Psi_Z \le C_1 \\ C_1, якщо \ \ \Psi_Z > C_1 \end{cases}. \qquad (3.15)$$

Якщо $\Psi_Z \le C_1$, то задача вирішена й шлях є оптимальним для потоку Ψ_Z.

Якщо $\Psi_Z > C_1$, то потік Ψ_1 розглядається як частковий потік, і переходимо до графа G_2, що виходить із графа G_1 шляхом заміни пропускних здатностей дуг C_{ij} на $C_{ij}^{'}$ зі співвідношення:

$$C_{ij}^{'} = \begin{cases} C_{ij} - C_1, для \ \ U \in \lambda_1 \\ C_{ij}, для \ \ U \notin \lambda_1 \end{cases}, \qquad (3.16)$$

де U – дуга графа (дуга U є насиченою, коли $\Psi(u) = C(u)$, а потік є повним, якщо кожний шлях з x_0 в Z містить, принаймні, одну насичену дугу). Дуги, у

яких $C_{ij}^{'} = 0$, виключаються з розгляду. Потік, розподіл якого шукається в графі G_2, приймається рівним

$$\Psi_Z^{'} = \Psi_Z - \Psi_1, \tag{3.17}$$

Тепер виникає задача пошуку найбільш оптимального розподілу потоку, але вже стосовно графа G_2. Її рішення дає шлях λ_2 із пропускною здатністю C_2, через який пропускається частковий потік

$$\Psi_2 = \begin{cases} \Psi_Z^{'}, если \quad \Psi_Z^{'} \le C_2 \\ C_2, если \quad \Psi_Z^{'} > C_2 \end{cases}, \tag{3.18}$$

Аналогічно, якщо $\Psi_Z^{'} \le C_2$, те задача вирішена й найбільш оптимальним розподілом потоків у графі G_1 буде проходження потоку Ψ_1 по шляху λ_1 і потоку Ψ_2 по шляху λ_2.

Якщо $\Psi_Z^{'} > C_2$, то слід перейти до нового графа G_3 й знайти новий частковий потік Ψ_3. Цей процес повторюється доти, поки сума часткових потоків не досягне значення Ψ_Z. Ці потоки, пропущені по графу G_1, і дадуть найбільш оптимальний розподіл потоку.

Таким чином, є можливість побудувати алгоритм з метою оптимального розподілу потоку повідомлень.

Проблема оптимізації маршрутів потоків даних представляє собою знаходження рішення системи лінійних рівнянь у невід'ємних числах, які мінімізують лінійну форму. Систему лінійних рівнянь можна записати в такий спосіб:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_m x_n = b_m \\ x_j \ge 0, (j = 0,1,2,..,n). \end{gathered} \tag{3.19}$$

При цих умовах

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n = Z \to \min, \tag{3.20}$$

де Z – змінна, що мінімізується.

У матричній формі система рівнянь має вигляд:

$$AX = B, (B \geq 0, X \geq 0), \tag{3.21}$$

$$Z(x) = Z_x \rightarrow \min . \tag{3.22}$$

При пуассонівських вхідних потоках задача знаходження оптимального шляху (з мінімальним часом доставки) статичної процедури маршрутизації зводиться до задачі мінімізації функції T:

$$T = \frac{1}{A}\sum_{(i,j)} \frac{x_{ij}}{c_{ij} - x_{ij}}, \text{ де } x_{ij} = \sum_{k=1}^{n} X_{ij}^{(k)} ; \; A = \sum_{(i,j)} \lambda_{ij} , \tag{3.23}$$

де x_{ij} – інтенсивність потоку, що протікає по каналу (ij);

λ – сумарний вхідний потік;

c_{ij} – пропускна здатність каналу.

На рис. 3.10 представлена блок-схема алгоритму обробки маршрутизатором вхідних пакетів (D – місце призначення, зазначене в пакеті).

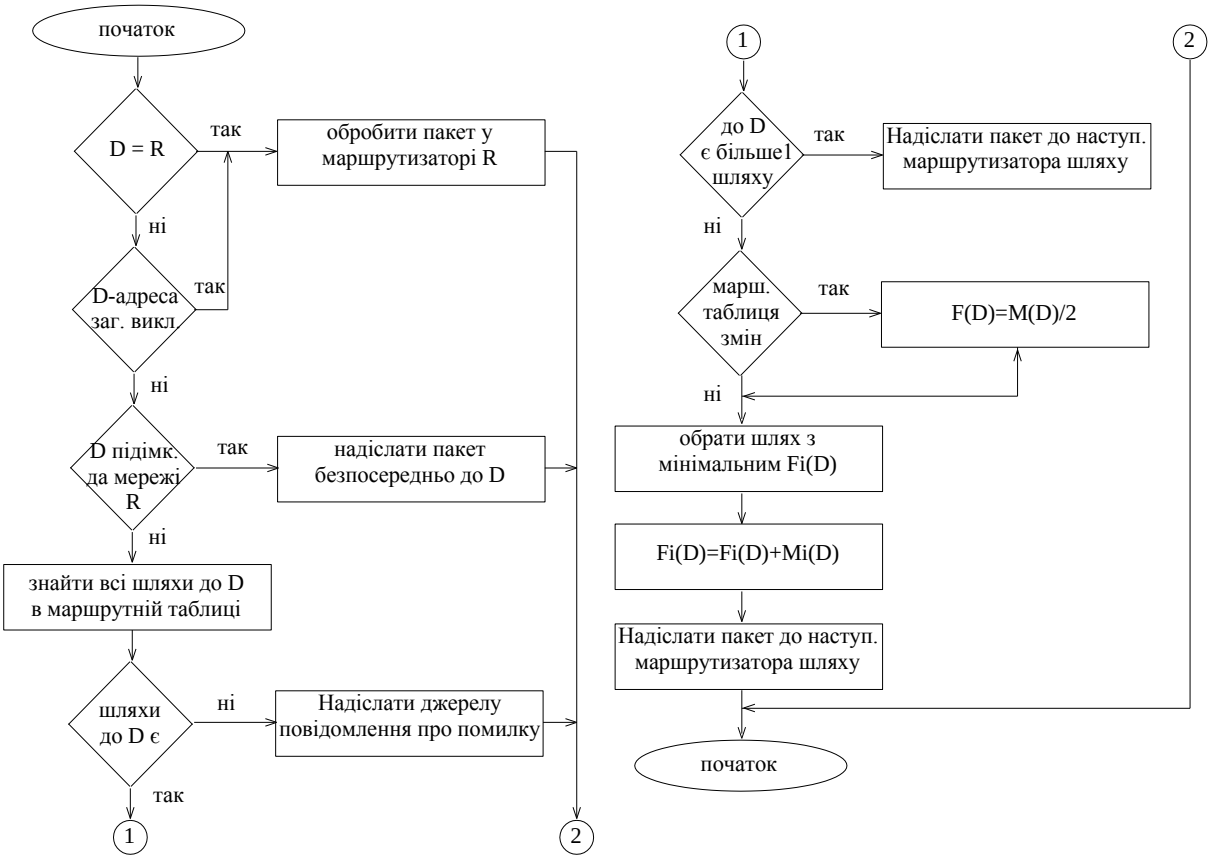


Рис. 3.10. Алгоритм обробки пакета в маршрутизаторі R

3.5. Рішення задачі оптимальної маршрутизації інформаційних потоків

Розподіл навантаження по декільком паралельним каналам передачі даних між двома вузлами є вкрай важливим. У багатьох практичних контекстах, сумарний потік між вузлами може бути таким, що жоден канал окремо (отже, і жоден маршрут) не зможе його пропустити. Однак сумарний потік може бути менше, ніж максимально припустимий потік через поперечний переріз, що розділяє два вузли. У цьому випадку, єдино можливим рішенням може бути розподіл основного потоку на декілька потоків.

Складний потік можна представити як суму простих потоків [67]. Така декомпозиція є зручним засобом для потокових алгоритмів. Для двох заданих потоків ξ і ψ через $\xi + \psi$ позначимо потік, значення якого на дузі (x_i, x_j) дорівнює $\xi_{ij}+\psi_{ij}$. Якщо ξ – довільний потік від s до t у графі G, то ξ можна представити у вигляді:

$$\xi = h(P_1) + h(P_2) + \ldots + h(P_v) + h(\Phi_1) + h(\Phi_2) + \ldots + h(\Phi_k), \qquad (1)$$

де $P_1,\ldots, P_v$ – прості ланцюги від s до t у графі G,

$\Phi_1,\ldots, \Phi_k$ – прості цикли в G.

На рис. 3.11 представлений граф потоку.

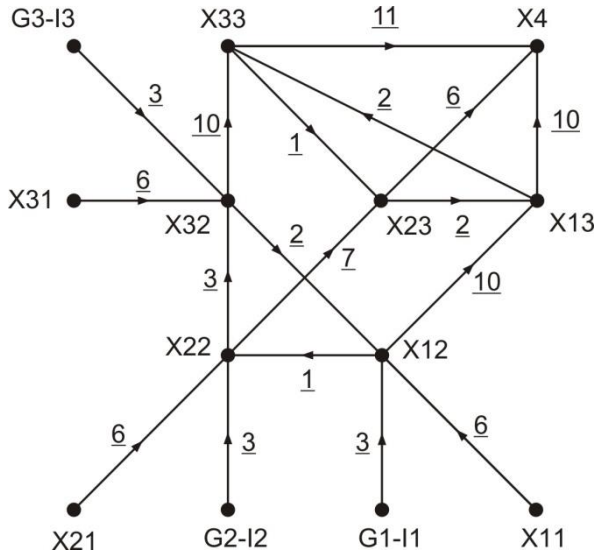


Рис. 3.11. Граф потоку ξ

3.5.1. Алгоритм обробки зміни маршрутизації

1. Визначити з пакета його джерело *S*.
2. Вибрати з пакета дані про чергове місце призначення *D*.
3. Якщо черга оброблена повністю, то перейти до п. 16.
4. Якщо маршрутні дані по *D* не можуть бути прийняті для розрахунку метрики шляху й *D* немає в маршрутній таблиці, то перейти до п. 2.
5. Виконати процедуру обчислення метрики для шляху *P* и перейти до місця призначення *D* через маршрутизатор *S*.
6. Якщо місце призначення є в маршрутній таблиці *D*, то перейти до п. 7.
7. Додати шлях *P* до маршрутної таблиці. Перейти до п. 14.
8. Порівняти нову складну метрику шляху *P* з найкращою метрикою в маршрутній таблиці для місця призначення *D*. Якщо нова метрика не перевищує цю найкращу метрику, то перейти до п. 11.
9. Якщо місце призначення *D* показане як недосяжне, або нова складна метрика для *P* не входить в інтервал метрик припустимих шляхів, то перейти до п. 10.
10. Якщо шляху *P* немає в маршрутній таблиці, то додати його, інакше помістити в маршрутну таблицю для *P* нову метричну інформацію. Перейти до п. 14.
11. Якщо шлях *P* є в маршрутній таблиці, то удалити його. Перейти до п. 14.
12. Визначити новий інтервал метрик припустимих шляхів до *D* відповідно до коефіцієнта перерахунку відносно нового мінімального значення метрики.
13. Переглянути шляхи до *D* у маршрутній таблиці. Видалити ті шляхи, метрики яких не входять в інтервал метрик припустимих шляхів.
14. Помістити нову метричну інформацію для шляху *P* у маршрутну таблицю.
15. Установити ознаку генерації змін.
16. Перейти до п. 2.

17.Кінець.

На рис. 3.12 представлена блок-схема алгоритму.

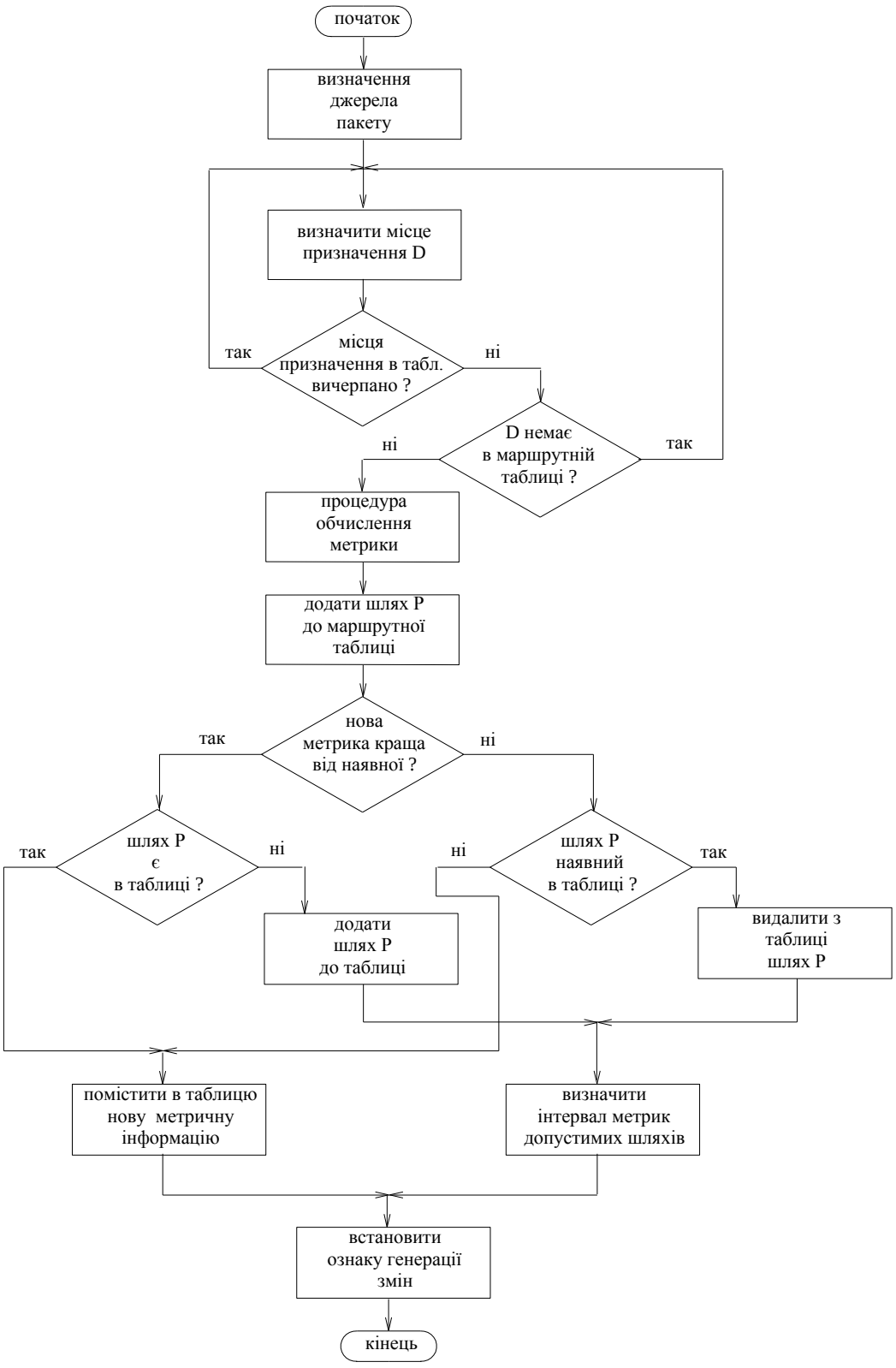


Рис. 3.12. Блок-схема алгоритму обробки зміни маршрутизації

3.5.2. Алгоритм генерації змін маршрутизації

Виконується після виконання алгоритму обробки змін маршрутизації, якщо в ньому була встановлена ознака генерації зміни, а також через певні інтервали часу загального виклику.

1. Вибрати черговий мережевий інтерфейс *I*, підключений до шлюзу. Якщо список інтерфейсів вичерпаний, то перейти до п. 8.
2. Створити порожнє повідомлення про зміну.
3. Вибрати з маршрутної таблиці чергове місце призначення *D*. Якщо місця призначення в маршрутній таблиці вичерпані, то перейти до п.6.
4. Переглянути в маршрутній таблиці всі шляхи до місця призначення *D*, за винятком тих, що мають наступний крок по інтерфейсу *I*. Визначити серед них шлях з мінімальним значенням складної метрики. Якщо таких шляхів до *D* немає, то перейти до п. 3.
5. Записати характеристики вибранного шляху в повідомленні про зміну. Перейти до п. 3.
6. Якщо повідомлення про зміну не пусте, то відправити його по интерфейсу *I*.
7. Перейти до п. 1.
8. Кінец.

На рис. 3.13 представлена блок-схема алгоритму.

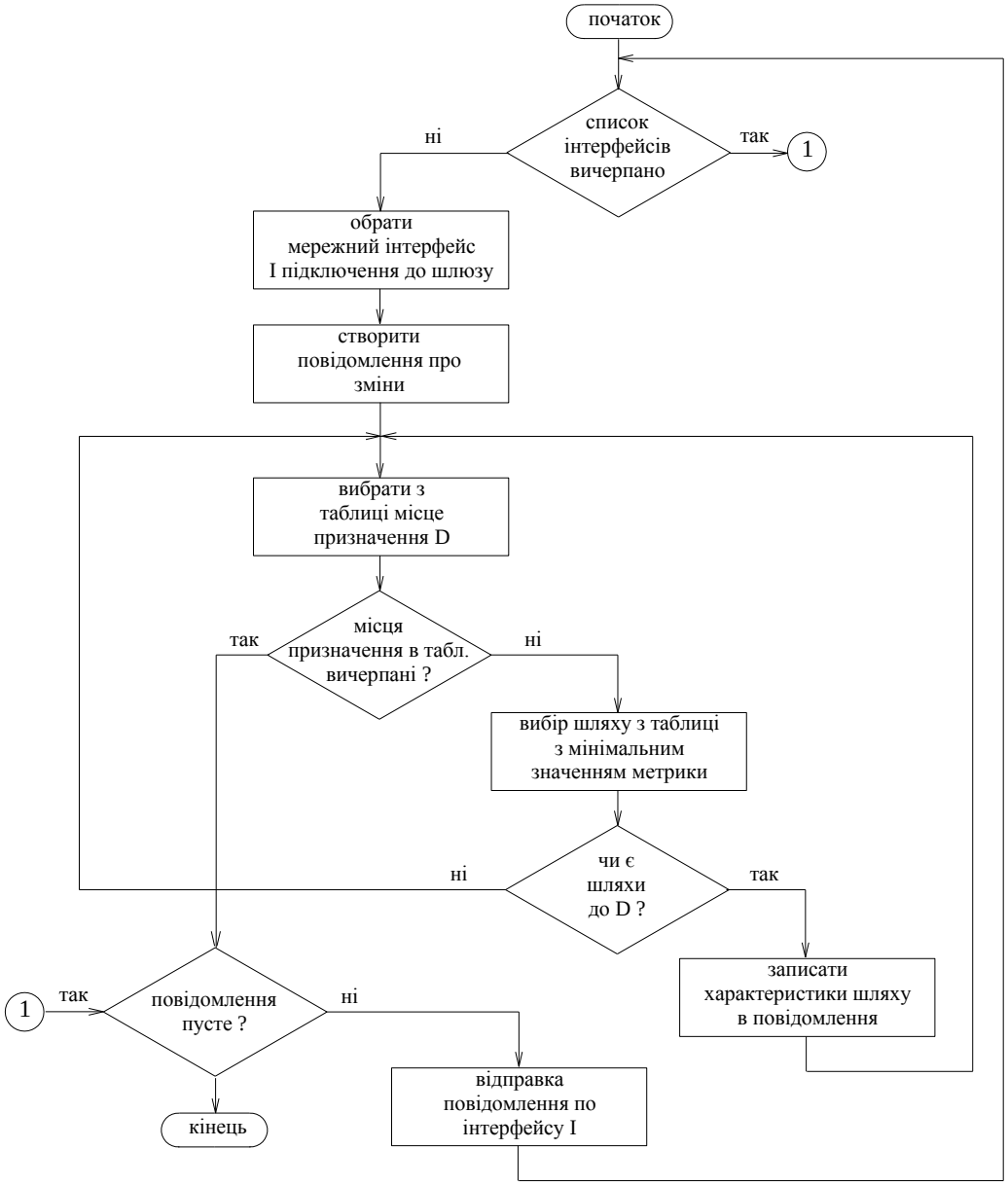


Рис. 3.13. Блок-схема алгоритму генерації змін маршрутизації

3.5.3. Процедура обчислення метрики шляху

Складна метрика шляху P від поточного шлюзу через інтерфейс I до місця призначення D розраховується по формулах:

$$T = T_n + T_I\,, \tag{3.24}$$

де T – затримка,

T_n – затримка з пакета,

T_I – топологічна затримка інтерфейсу I.

$$\lambda = \min(\lambda_n, \lambda_I)\,, \tag{3.25}$$

де λ – пропускна здатність,

λ_n – пропускна здатність із пакета,

λ_I – пропускна здатність інтерфейсу *I*.

$$R = \min(R_n, R_I), \tag{3.26}$$

де R – надійність каналу,

R_n – надійність із пакета,

R_I – надійність інтерфейсу *I*.

$$\mu = \max(\mu_n, \mu_I), \tag{3.27}$$

де μ – завантаження каналу,

μ_n – завантаження каналу з пакета,

μ_I – завантаження каналу інтерфейсу *I*.

Під пакетом мається на увазі пакет зміни маршрутизації, що прийшов від маршрутизатора *S* по інтерфейсу *I*. Розраховані значення параметрів використовуються в пакетах зміни маршрутизації, що розповсюджуються поточним маршрутизатором.

3.5.4. Алгоритм розподілу пакетів по різних шляхах до вузла призначення

Нехай до місця призначення *D* є *k* шляхів P_1, P_2, …, P_k, значення їхніх складних метрик, що дорівнюють M_1, M_2, …, M_k, а кількість відправлених по цих шляхах пакетів N_1, N_2, …, N_k... Тоді для забезпечення циклічного чергування шляхів із частотою, обернено пропорційній їхній складній метриці, повинне виконуватись співвідношення:

$$N_1 \cdot M_1 = N_2 \cdot M_2 = \ldots = N_k \cdot M_k . \tag{3.28}$$

Метою пропонованого алгоритму є вибір такого порядку нарощування N_1, N_2, …, N_k, який забезпечував би максимально можливе наближення до наведеного співвідношення в кожен поточний момент часу.

Алгоритм повинен виконуватися в умовах сталості значень M_1, M_2, …, M_k. Це означає, що він повинен почати виконуватися після зміни маршрутної таблиці й продовжувати виконуватися до наступної її зміни.

Передбачається, що для кожного шляху P_i визначається й зберігається параметр розподілу F_i. Ненульові початкові значення F_i забезпечують прийнятний розподіл на початковому етапі виконання.

1. Початок. Маршрутна таблиця зафіксована після зміни.
2. $F_i = M_i / 2$, i=1, 2, …k ...
3. Якщо маршрутна таблиця змінюється, то перейти до п. 7.
4. Вибрати шлях P_j для відправлення чергового пакета такий, що $F_j = \min(F_i)$, i=1, 2, …k ...
5. Відправити пакет по шляху P_j.
6. $F_j = F_j + M_j$. Перейти до п. 3.
7. Кінець.

3.6. Мінімізація затримки передачі даних по мережі

При побудові мереж велике практичне значення має рішення задачі оптимізації функціонування мереж з погляду найбільш економічного розподілу потоку даних по каналах зв'язку. Є можливість знайти оптимальні шляхи проходження потоку інформації з різних критеріїв – мінімізація часу доставки пакетів, вартості проходження пакета, середньої затримки, або мінімізація пропускної здатності. У багатьох практичних контекстах, сумарний потік між вузлами може бути таким, що жоден канал окремо (отже, і жоден маршрут) не зможе його пропустити. Однак сумарний потік може бути більше, ніж максимально припустимий потік через поперечний переріз, що розділяє два вузли. У цьому випадку, єдино можливим рішенням може бути розподіл основного потоку на кілька потоків з метою мінімізації часу передачі даних по мережі системи керування.

Припустимо, що відома топологія мережі й пропускна здатність каналів. Потрібно мінімізувати час затримки T потоку повідомлень λ_{ij} при заданих обмеженнях на потоки. Сума інтенсивностей в i-тому вузлі дорівнює:

$$\sum_j \lambda_{ij} = \lambda^i$$

де $i, j = \overline{1,n}$, $i \neq j$, n – число вузлів.

Позначимо

$$\lambda_k = \sum_{\substack{i=1 \\ j=1}}^{a} \lambda_{ij} k$$

як інтенсивність потоку в *k*-тому каналі.

Час затримки виражається формулою [1]:

$$T = \sum_{k=1}^{m} \frac{\lambda_k}{U} \cdot \left(\frac{1}{\mu C_k - \lambda_k} \right), \qquad (3.29)$$

де m – число каналів;

U – сумарна швидкість надходження повідомлень;

$C_к$ – пропускна здатність *k*-того каналу;

$1/\mu$ – середня довжина повідомлень (у бітах).

Підставимо в (3.29) значення λ_k:

$$T = \sum_{k=1}^{m} \frac{\sum_{i,j=1}^{n} \lambda_{ijk}}{U} \cdot \left(\frac{1}{\mu C_k - \sum_{i,j=1}^{n} \lambda_{ijk}} \right).$$

Введемо обмеження:

$$\sum_{k \in S_{ij}} \lambda_{ijk} - \lambda_{ij} = 0,$$

де S_{ij} – будь-який перетин графа, що розділяє *i*-у й *j*-у вершини. Причому $\lambda_{ijk} > 0$. Використовуючи відомий метод невизначених множників Лангранжа, маємо функцію Лангранжа у вигляді

$$T' = \sum_{k=1}^{m} \frac{\sum_{i,j=1}^{n} \lambda_{ijk}}{U} \cdot \left(\frac{1}{\mu C_k - \sum_{i,j=1}^{n} \lambda_{ijk}} \right) + \sum_{i,j=1, i \neq j}^{n} \sum_{S_{ij}} \gamma_{Sij} \sum_{K \in S_{ij}} \lambda_{ijk} - \lambda_{ij}, \qquad (3.30)$$

де γ_{sij} – множник Лагранжа.

Для пошуку оптимуму диференціюємо вираз (3.30) і прирівнюємо до нуля.

$$\frac{dT'}{d\lambda_{\mu\nu r}} = \frac{1}{U}\cdot\left(\frac{1}{\mu C_r - \sum\limits_{i,j=1,i\neq j}^{n}\lambda_{ijr}}\right) + \frac{\sum\limits_{i,j=1}^{n}\lambda_{ijr}}{U}\cdot\frac{1}{(\mu C_r - \sum\limits_{i,j=1,i\neq j}^{n}\lambda_{ijr})^2} - \sum\gamma_{s\overline{\mu r}}\cdot\Re(r\in S_{\overline{\mu r}}) = 0$$

де

$$\Re(r\in S_{\overline{\mu\nu}}) = \begin{cases} 0, \textit{якщо } r\notin S_{\overline{\mu\nu}} \\ 1, \textit{якщо } r\in S_{\overline{\mu\nu}} \end{cases}$$

Привівши до загального знаменника, одержуємо систему рівнянь:

$$(\mu C_\kappa - \sum_{i,j=1}^{n}\lambda_{yr}) + \sum_{i,j=1}^{n}\lambda_{yr} + U\cdot(\mu C_\kappa - \sum_{i,j=1}^{n}\lambda_{yr})^2\cdot\sum_{S_{\tilde{i}\tilde{\nu}}}r_{Sij}\cdot\Re\cdot(r\in S_{\overline{\mu\nu}}) = 0$$

де

$$\tilde{\mu},\tilde{\nu} = \overline{1,n};\ \ r - \overline{1,m};\ \ \sum_{k=S_{ij}}\lambda_{ijk} = \lambda_{ij};\ \ i,j = \overline{1,n}$$

Дана система кратних рівнянь вирішується відносно λ_{ijk} при застосуванні чисельного методу (метод Ньютона), для чого є відповідні пакети прикладних програм.

Розглянемо інший підхід у рішенні поставленого завдання. Розглядається мережа комутації пакетів, що складає з вузлів. Топологія мережі описується матрицею *C* розміру $n\times n$, елементи якої мають значення пропускних здатностей каналів зв'язку між *i*-м і *j*-м вузлами. Матриця є симетричною з нульовими вузлами й ненегативними недіагональними елементами.

Якщо $C_{ij} - C_{ji} > 0$, то існує канал зв'язку із пропускною здатністю C_{ij} біт/с, що зв'язує *i*-й і *j*-й вузли мережі.

З боку абонентів мережі в її вузли надходять потоки повідомлень для передачі іншим абонентам. Потоки повідомлень передбачаються пуассоновськими з інтенсивностями r_i повідомлень у секунду. Середня довжина повідомлень (пакетів) у мережі передбачається рівною $1/\mu$. Для кожного вузла передбачається відомою середня інтенсивність потоку повідомлень Y_1 для його абонента.

При передачі пакетів по каналу зв'язку між двома вузлами середній час очікування в черзі на обробку визначається вираженням

$$\bar{t}_{ожс} = \frac{\mu}{\mu C - u} \cdot \frac{1}{\mu C}$$

де *u*-інтенсивність потоку повідомлень [1/с];

C – ефективна пропускна здатність каналу зв'язку [біт/сек];

$1/\mu$ – середня довжина повідомлення [біт];

Якщо *i*-й вузол з'єднується каналами зв'язку з іншими вузлами, то задача оптимального розподілу трафіка формулюється в такий спосіб. Позначимо через u_{ij} інтенсивності потоку повідомлень із *i*-го вузла в *j*-й в [1/с]. При цьому $u_{ij} = 0$, якщо C_{ij} (немає зв'язку). Умовою існування стаціонарного режиму в мережі є виконання рівності

$$\left(r_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} (u_{ji} - u_{ij}) \right) = y_i, \qquad (3.31)$$

і нерівності

$$\mu C_{ij} > u_{ij} \geq 0 \qquad (3.32)$$

де r_i – інтенсивність повідомлень від абонента *i*-го вузла;

y_i – інтенсивність повідомлень від вузла до абонента, зміст якого полягає в тому, що всі повідомлення повинні знаходити своїх адресатів.

Сумарний час очікування в чергах на передачу в стаціонарному режимі мережі визначається виразом:

$$T = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{u_{ij}}{\mu C_{ij} - u_{ij}} \cdot \frac{1}{\mu C_{ij}} \right) \to \min \qquad (3.33)$$

Задача оптимізації полягає у визначенні таких $u_{ij}\ i, j = \overline{1,n}$, які задовольняють обмеженням (3.31) і (3.32) і мінімізують функціонал (3.33). Для аналітичного рішення задачі скористаємося методом множників Лагранжа й необхідною умовою мінімуму функції декількох змінних. Нехай γ_i – множник Лагранжа, тоді:

$$T^1 = T + \sum_{i=1}^{n} \gamma_i \left(r_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(u_{ji} - u_{ij} \right) \right). \qquad (3.34)$$

Диференціюючи по тим u_{ij}, для яких $C_{ij} > 0$, одержуємо в якості необхідної умови екстремума систему алгебраїчних рівнянь:

$$\begin{cases} \dfrac{1}{(\mu C_{ij} - u_{ij})} + \lambda_i = 0 & i = \overline{1,n} \\ & j = \overline{1,n} \\ \displaystyle\sum_{i=1}^{n} \left(r_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(u_{ji} - u_{ij} \right) \right) = 0 & j \neq i \\ & C_{ij} > 0 \end{cases} \qquad (3.35)$$

Система (3.35) вирішується чисельним методом. Атрибути, пов'язані з каналами передачі даних і ресурсами, а також параметри, асоційовані з маршрутизацією, у сукупності являють собою набір керуючих змінних, які можуть бути модифіковані для того щоб привести мережу в бажаний стан.

3.6. Розробка протоколу мережевого рівня

Для мережних комунікацій і розробки мережних протоколів використовується еталонна модель взаємодії відкритих систем (ВВС) OSI. Кожен рівень обслуговує свою частину процесу взаємодії. Завдяки такій структурі спільна робота мережного устаткування й програмного забезпечення стає набагато простіша й зрозуміліша. Семирівнева модель ВВС є теоретичною, і містить ряд недоробок. Реальні мережні протоколи змушені відхилятися від неї, забезпечуючи непередбачувані можливості, тому прив'язка деяких з них до рівнів ВВС є трохи умовною. У роботі [45] сформульовані вимоги до апаратно-програмних засобів керування технічними об'єктами в режимі реального часу, однак розроблена структура системи не враховує особливостей об'єкта керування й топології мережі. Тому актуальною є задача розробки спеціалізованого протоколу, у якому була б врахована можливість збору даних про стан мережі й на основі цих даних виконувати оптимальну маршрутизацію. У роботі [64] розглядається метод керування якістю передачі пакетів на основі оцінки прогнозованих

значень запізнювання пакетів. Однак використовуваний протокол не містить у собі засобів для виміру зазначеного параметра, і оцінка часу запізнювання виконується на основі прогнозу й відрізняється невисокою точністю. У зв'язку із цим необхідно оптимізувати протокол, включивши в нього засоби контролю параметрів мережі.

Для підвищення продуктивності, надійності мікропроцесорної мережі керування автономної електроенергетичної системи, до яких відноситься й суднова, необхідна розробка спеціалізованого протоколу мережного рівня, який би задовольняв наступним вимогам:

1. Надійність передачі даних на своєму рівні. Бажано, щоб кодування даних у пакеті мало достатню надмірність, і при втраті одного пакета на більш низькому рівні (одного байта в протоколі RS-232 повністю) дані можна було відновити.

2. Контроль цілісності даних – надійно і просто реалізується за допомогою алгоритму підрахунку CRC. У цьому випадку вибрана CRC8, оскільки в такому випадку сума має 8 розрядів і вміщується в 1 пакет протоколу більш низького рівня, що підвищує надійність передачі.

3. Контроль передачі пакета. У назві кожного пакета міститься його номер і загальна кількість пакетів, для того щоб можна було контролювати повні втрати пакетів і давати запит на повторну передачу даних.

4. Адресація пакета. Якщо необхідно адресувати дані одному пристрою, розподіл приймача/передавача на порти недоцільно, оскільки у якості приймача може виступати електронний пристрій на однозадачному мікроконтролері, а отже немає необхідності після прийому розподіляти пакети між різними додатками. При наявності декількох приймачів необхідно вводити поле адреси.

5. Додаткові дані: поле для службових даних, наприклад, зміст пакета – дані або команди; пакет для пінга, вимагає повернення для визначення наявності вузла в мережі з метою побудови таблиць маршрутизації.

Специфіка проблеми полягає в тому, що протокол мережного рівня фактично буде працювати на різних канальних і фізичних протоколах, так як при передачі даних від одного пристрою до іншого на цих рівнях інформація перетворюється, які представлені на рис. 3.14.

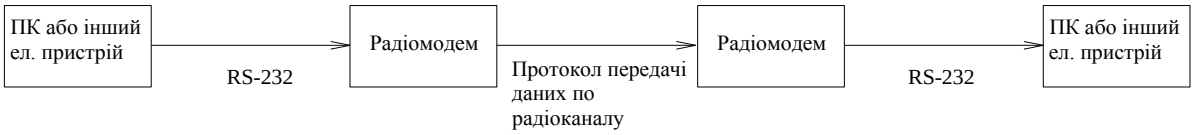


Рис. 3.14. Структура каналу зв'язку

За основу для розробки спеціалізованого протоколу мережного рівня взятий протокол Internet, який був створений для використання в об'єднаних системах комп'ютерних комунікаційних мереж з комутацією пакетів. Структура пакета протоколу IP представлена на рис. 3.15 [101, 130].

Версія	IHL	Тип сервіса	Загальний розмір	
Ідентифікатор			Прапори	Зсув
Час життя TTL		Протокол	CRC заголовка	
Адреса джерела				
Адреса отримувача				
Додаткові опції			Вирівнювання	

Рис. 3.15. Приклад заголовка Internet датаграми

На основі будови пакета протоколу IP можна сформулювати вимоги до заголовка власного протоколу:

1. Оскільки протокол розробляється для вузького використання, його модифікації будуть не систематизовані й передавати версію протоколу недоцільно.

2. Оскільки топологія мережі – товсте дерево, тобто всі пристрої зв'язані один з одним на одному ієрархічному рівні по радіоканалу, першим потрібно посилати адресу одержувача (переадресація пакетів, якщо вона необхідна, може відбуватися усередині обробного коду приймача, самим

протоколом вона не передбачена), а потім – адреса відправника пакета, для ідентифікації.

3. Після адрес передається розмір пакета. У даному протоколі розмір пакета динамічний, обмежений 16 байтами, вирівнювання додатковими нулями не робиться, тому при передачі масиву даних останній пакет не буде відповідати фіксованому розміру.

4. Контрольна сума даних переданих у пакеті. CRC голови пакета не передається.

5. Службова інформація (визначено тільки 2 прапори: 0b00000000 – звичайний пакет з даними й 0b00000001 – пакет для пінга, вимагає повернення себе для контролю зв'язку в мережі).

6. Блок даних, довжина до 16 байт (визначена досвідченим шляхом при роботі протоколу на радіомодемах, при інших фізичних протоколах або способах передачі даних може бути перевизначена).

Графічне зображення пакета представлене на рис. 3.16.

0	1	2	3	4	5	6	7
Адреса отримувача							
Адреса відправника							
Розмір пакета							
Контрольна сума даних							
Службові прапори							
Дані							

Кінець даних							

Рис. 3.16. Структура пакета мікропроцесорної мережі

Протокол використовує поля в заголовкі для фрагментації й відновлення пакетів, коли це необхідно для їхньої передачі через мережі з малим розміром пакетів. Сценарій дії полягає в тому, що змінюється розмір

пакета на кожному з вузлів, задіяних у комунікації й на кожному зі шлюзів, що забезпечують взаємодію між мережами різних ієрархічних рівнів мережі [100].

У службових даних розташовується поле для визначення типу обслуговування для позначення необхідної послуги. Ці параметри повинні використатися для керування вибором реальних робочих характеристик при передачі пакета через мережу. Обслуговування із пріоритетом дає перевагу для просування даного пакета в порівнянні з усіма іншими [115]. Вибір здійснюється між трьома альтернативами: малою затримкою, високою вірогідністю й високою пропускною здатністю.

Механізм часу життя служить для вказівки верхньої межі часу життя інформаційного пакета. Цей параметр установлюється відправником і зменшується в кожній точці на прохідному маршруті. Якщо параметр часу життя стане нульовим до того, як пакет досягне одержувача, цей пакет буде знищений.

Механізм опцій надає функції керування, які є необхідними або просто корисними при певних ситуаціях, однак він не потрібний при звичайних комунікаціях. Механізм опцій надає такі можливості, як тимчасові штампи, безпека, спеціальна маршрутизація.

Контрольна сума заголовка забезпечує перевірку того, що інформація, яка використовується для обробки пакетів, передана правильно. Дані можуть містити помилки. Якщо контрольна сума невірна, то пакет буде зруйнований, як тільки помилка буде виявлена, а відправникові буде передано повідомлення про втрату пакета.

На даному етапі проектування можна оптимізувати пакет, як показано на рис. 3.17.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="3">Адреса відправника</td><td colspan="3">Адреса отримувача</td><td colspan="2">Службові прапори</td></tr>
<tr><td colspan="4">Розмір пакета</td><td colspan="4">Вирівнюючі нулі</td></tr>
<tr><td colspan="8">Контрольна сума даних</td></tr>
<tr><td colspan="8">Дані</td></tr>
<tr><td colspan="8">-------</td></tr>
<tr><td colspan="8">Кінець даних</td></tr>
</table>

Рис. 3.17. Оптимізована структура пакета

Як видно з рисунка, якщо врахувати обмеження конкретного випадку й виключити надмірність таких полів як адреси й прапори, то можна скоротити розмір заголовка пакета з 5 до 3 байт, що дозволить включити в пакет додаткові дані – номер пакета і їх загальна кількість для контролю передачі. У розглянутому випадку розрядність адрес зменшена з 8 до 3, розрядність поля прапорів – з 8 до 2, розмір пакета з 8 до 4 біт.

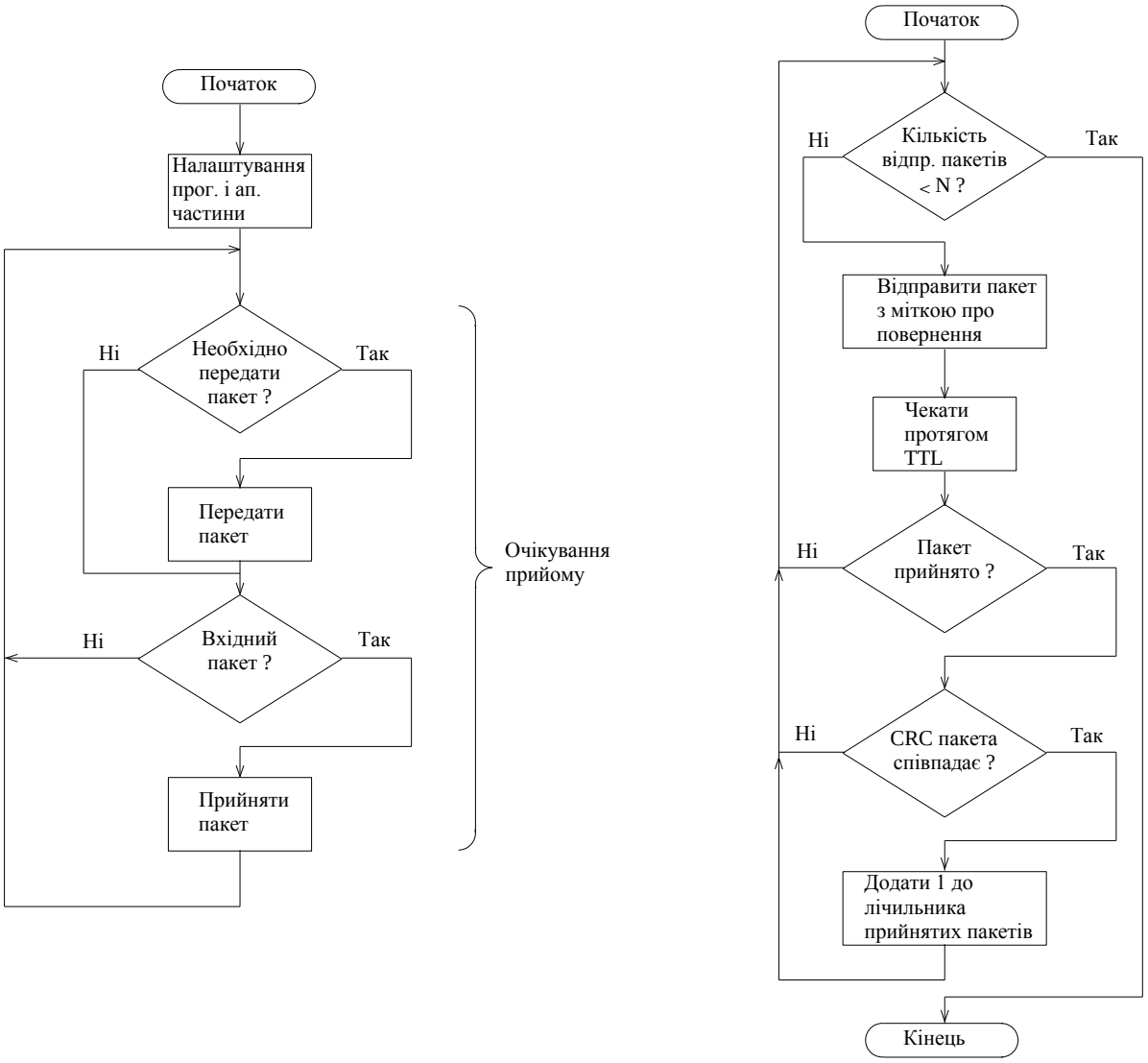


Рис. 3.18. Блок-схеми алгоритмів:
а) – алгоритму обміну даними; б) – алгоритму перевірки надійності каналу зв'язку

Для перевірки надійності передачі даних по каналі використовується наступний алгоритм: N кількість разів відсилається пакет з позначкою, що його блок даних необхідно повернути відправникові. Потім запускається таймер, який відраховує час життя пакета TTL (установлюється перед перевіркою, залежно від того яка швидкодія каналу вважається недостатньою у конкретному випадку). Якщо пакет не повернувся назад до закінчення часу життя TTL або повернувся, але CRC його даних не збігається з розрахованим

другий раз, то пакет вважається загубленим. Результат роботи функції обчислюється по наступній формулі:

$$s = \frac{1}{N}\sum_{i=1}^{N} P(ttl, crc) \cdot \qquad (3.36)$$

де *P(ttl,crc)* – функція, що приймає залежно від аргументів наступні значення:

P(ttl,crc) = 1 якщо *ttl* менше заданого перед перевіркою й *crc* збігається з розрахованим відправником;

P(ttl,crc) = 0 якщо *ttl* більше заданого перед перевіркою й *crc* не збігається з розрахованим відправником.

На рис. 3.18, а, представлена блок-схема алгоритму обміну даними. На рис. 3.18, б, представлена блок-схема алгоритму перевірки надійності каналу зв'язку.

Розроблений протокол мережного рівня мікропроцесорної мережі керування дозволяє містити необхідні засоби для вирішення задач маршрутизації потоків, приорітетній обробці, контролю цілісності й дослідженню статистичних характеристик потоків. Зниження пропускної здатності каналу зв'язку приведе до створення черг у вузлах мережі, і в найгіршому разі втрати пакетів при переповненні черг. Вибір оптимального параметра оптимізації визначається задачами, які вирішуються мережею й у загальному випадку може змінюватися в процесі функціонування мережі залежно від режиму роботи автономної електростанції.

3.7. Завадостійке кодування інформації з використанням SWITCH –технології

Високі вимоги пред'являються до вірогідності повідомлень, які передаються по мережі. Це приводить до необхідності використовувати спеціальні засоби заходу щодо зниження кількості помилок. Одним з найбільш ефективних мір є завадостійке кодування інформації. Вид

завадостійкого коду залежить від структури системи зв'язку, узагальнена схема якої наведена на рис. 3.19.

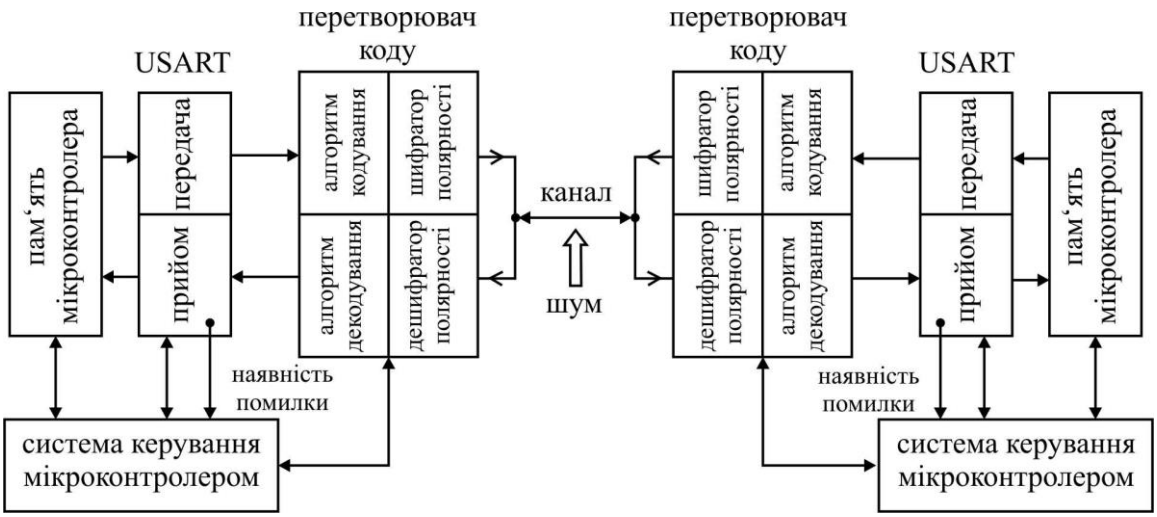


Рис. 3.19. Структура системи зв'язку

Пристрої кодування й декодування інформації можна розглядати як скінченні автомати, зміни станів яких відбуваються залежно від значень біт у вхідному слові.

Зазвичай шум припускають нормально розподіленим (білим гауссовським шумом). Як правило, рівні й тривалості імпульсних перешкод приймають розподіленими за експонентним законом, а ймовірність появи m імпульсів на інтервалі часу $T_{\text{і.з.}}$ (відповідно до розподілу Пуассона):

$$P_{i.з.}(m) = (LT_{i.з.})m \cdot \exp(\frac{-LT_{i.з.}}{m!})$$

де L – середнє число імпульсів на інтервалі $T_{\text{і.з.}}$

Коди AMI (Alternate Mark Inversion) і HDB3 (High-Density Bipolar 3 Zeros) є різновидами псевдотроїчного кодування, тобто сигнал передачі в лінії має три рівні: позитивний, негативний і нульовий. У цих кодах двійкові одиниці представляються позмінно позитивними й негативними імпульсами, за рахунок чого досягається відсутність постійної складової сигналу в лінії передачі даних [100].

У коді AMI одиниці кодуються поперемінно позитивними й негативними імпульсами, а двійковий нуль представляється нульовим рівнем. Тому при довгій послідовності нулів може бути втрата синхронізації.

Код HDB3 виправляє цю ситуацію, не допускаючи відсутності імпульсів більше 3-х тактів. Чотири нулі, що йдуть підряд, кодуються в такий спосіб:

Таблиця 3.1

Кодування довгої послідовності нулів

	Кількість імпульсів після останньої заміни	
Останній імпульс	непарна	парна
Негативний	000–	+00+
Позитивний	000+	–00–

У таблиці: "0" – нульовий рівень, "–" – негативний імпульс, "+" – позитивний імпульс.

На прикладі коду HDB3 розглядаються принципи побудови графа переходів, що описує послідовність кодування інформації з метою виконання програмної реалізації кодера й декодера із застосуванням SWITCH-технології. Алгоритмізація й формалізація завдань, пов'язаних з побудовою графа, докладно описується в [55, 67].

Код HDB3 виправляє будь-які чотири нулі, що йдуть один за одним у вихідній послідовності. Кожні чотири нулі заміняються чотирма сигналами, у яких є один сигнал V. Для придушення постійної складової полярність сигналу V чергується при послідовних замінах. Крім того, для заміни використовуються два зразки чотиритактових кодів. Якщо перед заміною вихідний код містив непарне число одиниць, то використовується послідовність 000V, а якщо число одиниць було парним – послідовність 1*00V.

Щоб зрозуміти принцип утворення коду зобразимо всі можливі комбінації шифратора у вигляді графи-схеми:

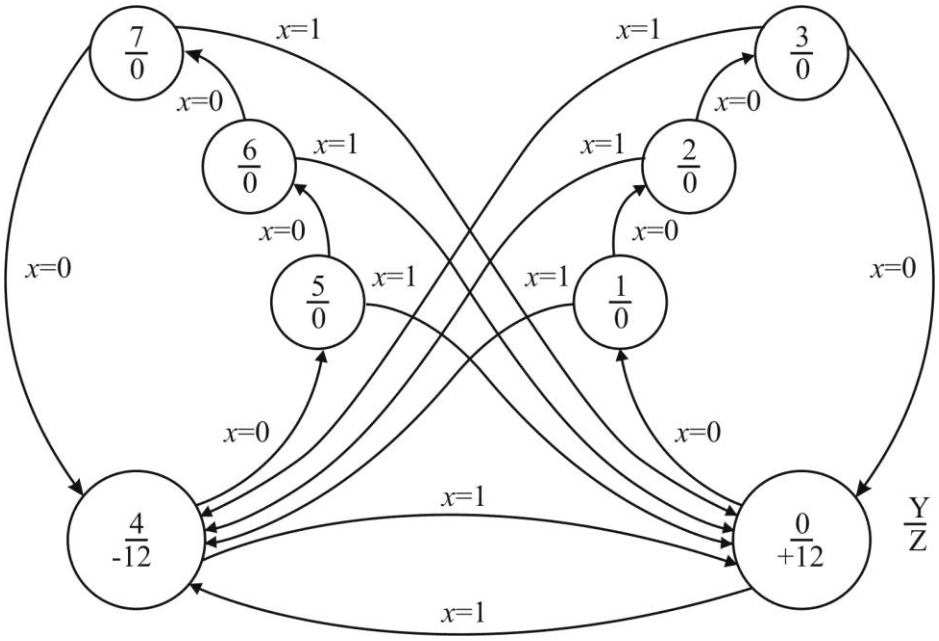


Рис. 3.20. Граф станів – автомата перетворювача коду

Переходи передавача між трьома можливими станами (+12В, 0, –12В) можна простежити по діаграмі, наведеній на рис. 3.20.

Фізично сигнал нуля у двополярній системі має один логічний стан. Оскільки через кожні три послідовності нулів відбувається зміна двополярного сигналу, то дану послідовність нулів можна представити окремими логічними станами. Це спрощує розробку алгоритму й робить зрозумілим принцип роботи передавача HDB 3-коду.

Декодування сигналів у приймачі пояснюється тією ж діаграмою, що й у передавачі (рис. 3.20), але з трохи іншою інтерпретацією подій.

Стан, з якого відбувається перехід у новий стан, відповідає раніше прийнятому з лінії стану трьохрівневого сигналу. Цифра 0 або 1 біля стрілки відповідає значенню чергового прийнятого з лінії біта DATA.

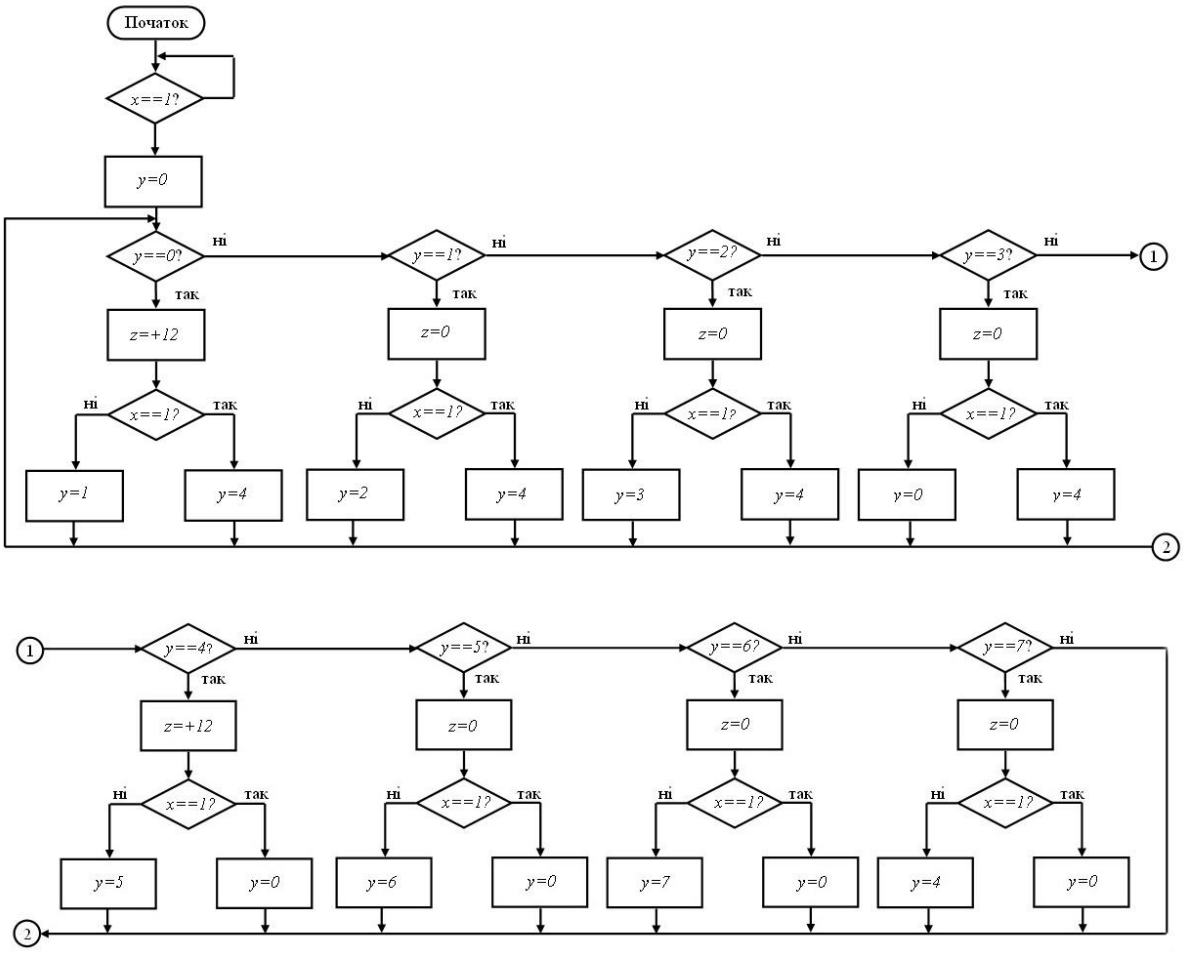


Рис. 3.21. Блок-схема алгоритму кодування

Найбільш просто вирішити проблему перетворення алгоритму в текст програми досягається при використанні такої керуючої конструкції, як оператор switch мови Сі (SWITCH-технологія, автоматне програмування). Приведемо програму, що використовує цей оператор для реалізації графа переходів передавача (рис. 3.20) і граф-схеми алгоритму (рис. 3.21):

```
switch (Y)
{
case 0 :
z=h;         // H - змінна, котра відповідає рівню «+12В»
if(x) Y=4 else Y=1;
break;
case 1 :
z=0;
if(x) Y=4 else Y=2;
break;
case 2 :
z=0;
```

```
if(x) Y=4 else Y=3;
break;
case 3 :
z=0;
if(x) Y=4 else Y=0;
break;
case 4 :
z=L;        // L - змінна, котра відповідає рівню «-12В»
if(x) Y=0 else Y=5;
break;
case 5 :
z=0;
if(x) Y=0 else Y=6;
break;
case 6 :
z=0;
if(x) Y=0 else Y=7;
break;
case 7 :
z=0;
if(x) Y=0 else Y=4;
break;
}
```

Для системи взаємозалежних графів переходів за один програмний цикл у кожному графі реалізується не більше одного переходу. Це робить доступним номер будь-якого стану.

Будь-який канал зв'язку з обмеженими смугою частот, часом передачі й динамічним діапазоном (значень амплітуд) має кінцеву пропускну здатність. При необмежено малій імовірності помилок швидкість передачі завжди менше пропускної здатності. У каналі з помилками максимальне значення швидкості одержують шляхом використання завадостійкого кодування. Останнє вимагає введення надмірності в переданий сигнал: за часом, частоті або амплітуді. Якщо код погоджений з каналом, тобто код дозволяє виправляти найбільш імовірні помилки, введена надмірність стає виправданою. Якщо код не погоджений з каналом, помилки можуть бути не тільки не виправлені, але й розмножені кодом. Для узгодження коду з каналом зв'язку необхідно мати максимальний обсяг відомостей про можливі впливи, що заважають, у каналах.

3.8. Модельні дослідження роботи систем керування автономними електроенергетичними установками

Моделювання інформаційних систем та питання діагностування мереж обробки інформації розглядається в роботах [65, 68, 69, 75, 82, 93, 94 122, 129]. Використанню паттернових мереж в задачах аналізу та моделювання обчислювальних систем присвячені роботи [62, 63].

Запропонована методика проведення дослідження передбачає моделювання автономної енергосистеми з паралельним включенням генераторів, керуванням споживачами, розрахунком параметрів і їхньою індикацією, а також обмін даними по послідовному протоколі RS-232.

Апаратно реалізована система керування приймає й обробляє дані енергосистеми, яка моделюється в реальному часі, управляючи генераторами й споживачами. Система керування складається з декількох взаємозалежних між собою модулів, забезпечуючи розподіленість як на рівні обчислень, так і в апаратному рішенні. Для кожного з генераторів передбачений окремий обчислювальний модуль на основі 8-бітного мікроконтролера, що приймає й обробляє дані й головний обчислювальний модуль на основі 32-бітного мікроконтролера, що приймає розрахунки інших модулів, що формує керуючі пакети й передачу даних для моніторингу. На рис. 3.22 представлений вигляд лабораторного стенду, на основі якого реалізовано систему керування.

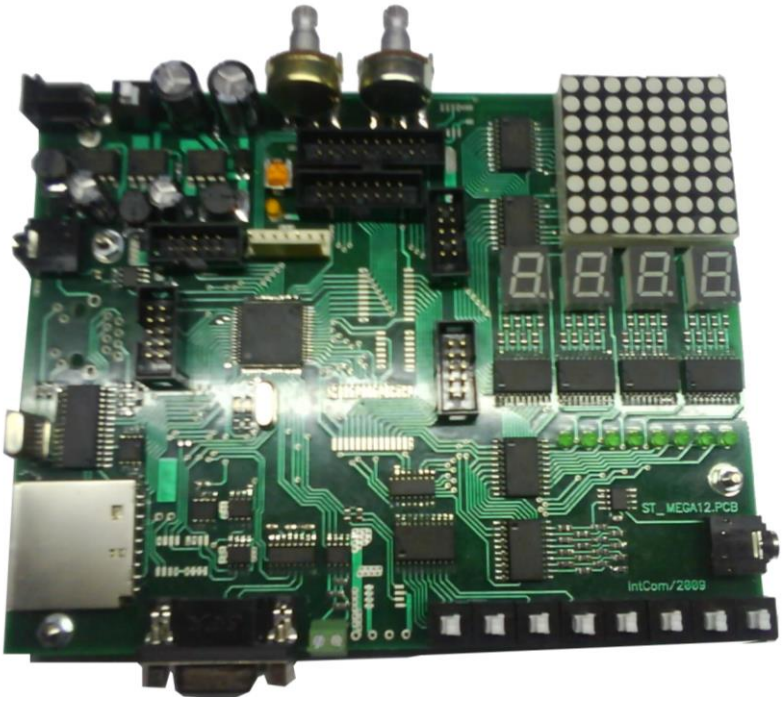


Рис. 3.22. Стенд для моделювання системи керування

Наявність інтерфейсів RS485, Ethernet дозволяє використовувати його при проектуванні та моделюванні систем дистанційного керування обладнанням.

На рис. 3.23 представлена структура системи керування моделю AEEC.

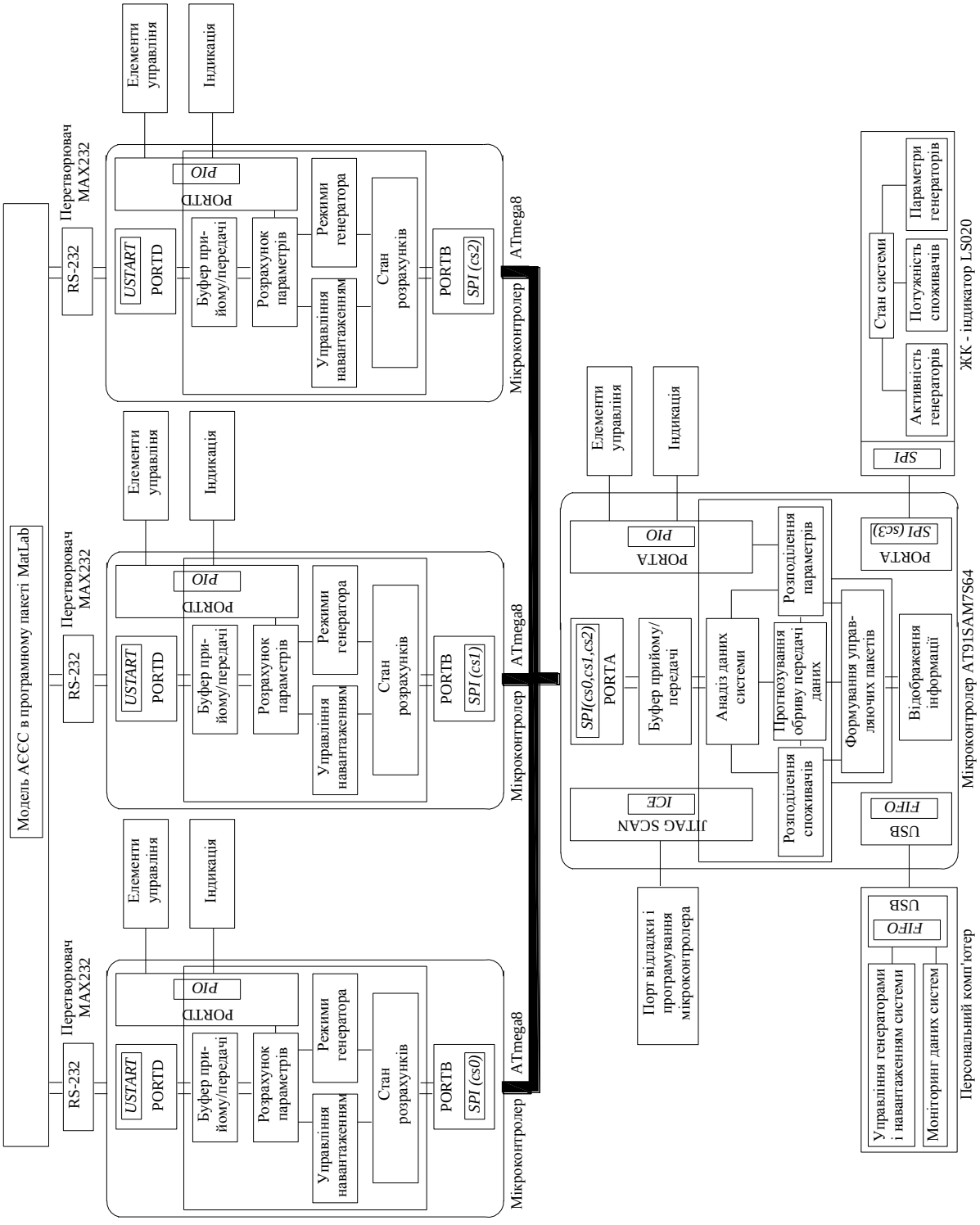


Рис. 3.23. Блок-схема системи керування

Модель автономної енергосистеми представлена у вигляді 3-х дизель-генераторів, кожен з яких має свою систему керування зі зворотним зв'язком, керований комутатор навантаження й блок споживачів.

Модель дозволяє аналізувати стан АЕЕС в момент запуску генераторів, при перевищенні номінальної потужності генераторів, характер струмів та напруг системи, а також передавати дані для подальших розрахунків в систему управління.

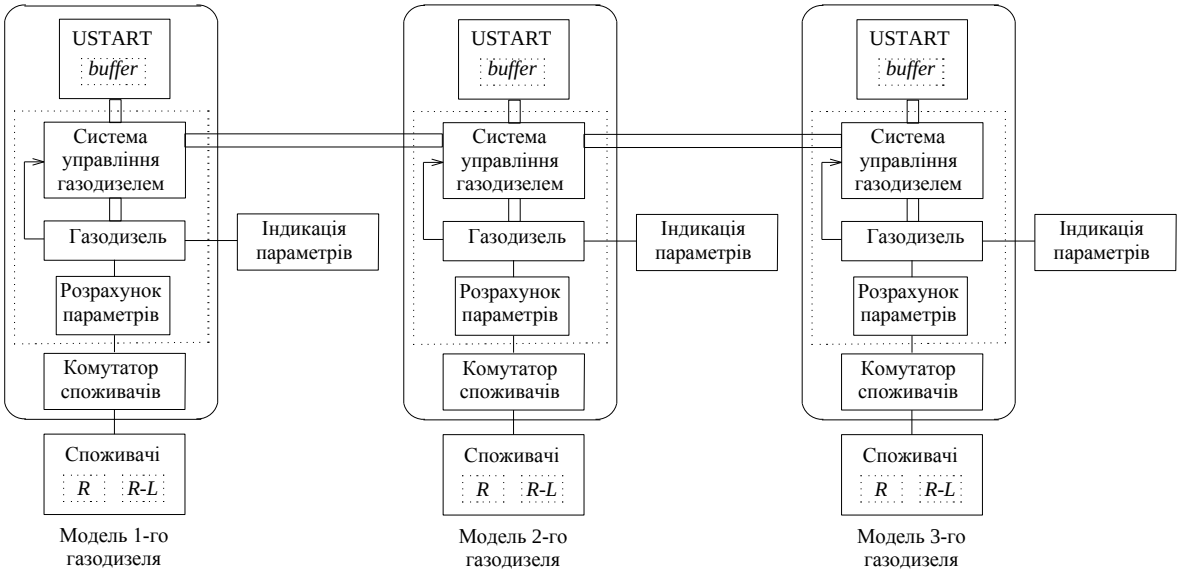


Рис. 3.24. Блок-схема моделі АЕЕС

Система керування обробляє дані й передає на USB-порт персонального комп'ютера для їхнього подальшого моніторингу. Система моніторингу даних являє собою ряд індикаторів (шкальний, графічний і чисельний), які дозволяють відображати осцилограми струмів і напруг генераторів, поточне значення частоти й споживаної потужності. Так само передбачені управляючі сигнали для генераторів і споживачів. Система моніторингу дозволить створювати базу даних системи з урахуванням установленого інтервалу часу. Розподіл виконуваних завдань на паралельні потоки й запис проміжних даних у стеки дозволить підвищити швидкодію й надійність програмного продукту.

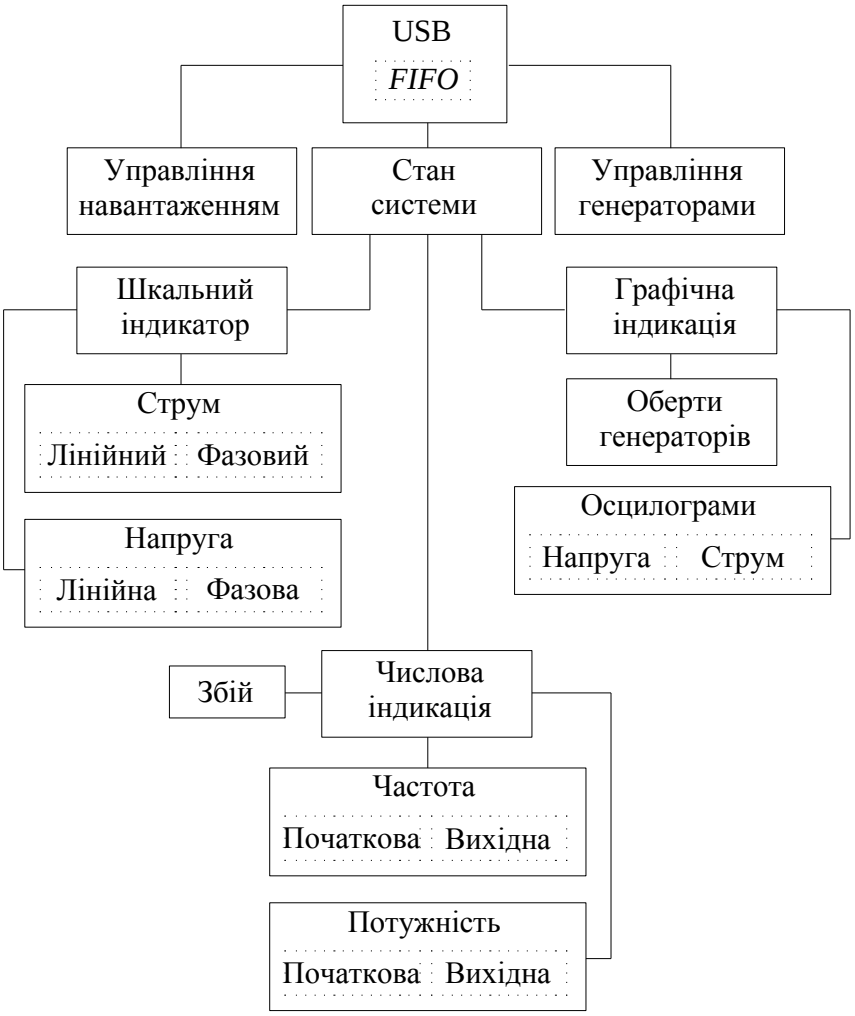


Рис. 3.25. Блок-схема системи моніторингу даних

3.8.1. Моделювання мікропроцесорної системи керування АЕЕС

Модель дизель-генераторного агрегату, що розроблена у пакеті Matlab Simulink [41, 140], представлена на рис. 3.26 і має у своєму складі наступні блоки: модель дизеля, модель генератора, система збудження генератора, комутатори, навантаження різної потужності (100 кВт, 140 кВА, 210 кВА), блоки виміру сигналів, блоки для роботи із COM-портом.

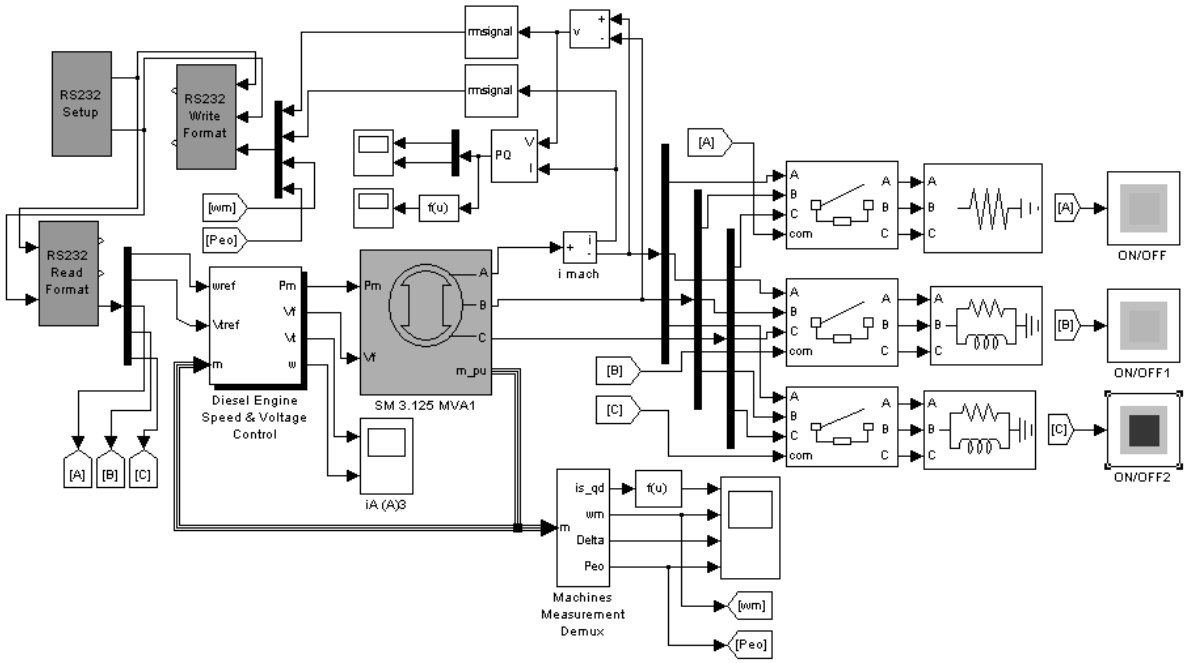


Рис. 3.26. Модель контрольованого устаткування

Модель дизеля описується наступними передатними функціями:

$$H_c = \frac{k \cdot (1 + T_3 p)}{1 + T_1 p + T_1 T_2 p^2}, \tag{3.37}$$

$$H_a = \frac{(1 + T_4 p)}{p(1 + T_5 p)(1 + T_6 p)}, \tag{3.38}$$

де H_c – передатна функція регулятора,

$T_1 – T_3$ – постійні часу регулятора,

H_a – передатна функція актуатора,

$T_4 – T_6$ – постійні часу актуатора.

Блок «Diesel Engine Speed & Voltage Control» показаний на рис. 3.27.

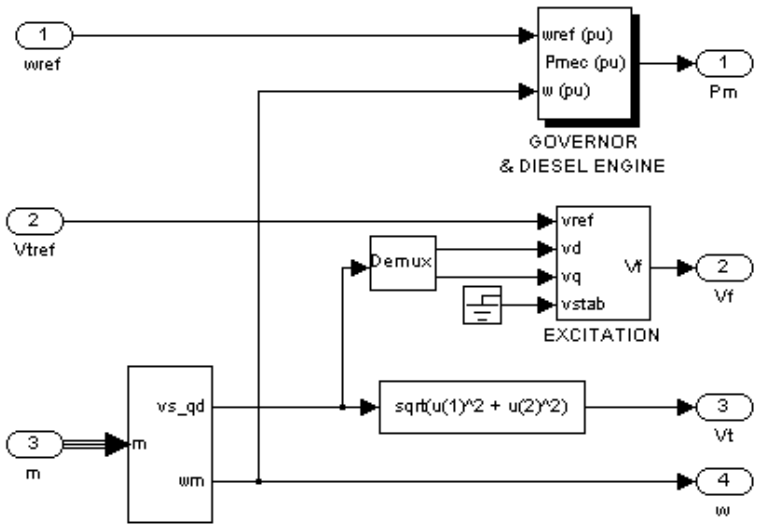


Рис. 3.27. Моделі дизеля й системи збудження генератора

Вхідними сигналами в блок дизеля є необхідна частота оборотів (wref) (у відносних одиницях) і фактичне значення частоти оборотів (wm). На виході формується сигнал, який відповідає потужності, що віддається дизелем, (Pm). На систему збудження подаэться опорне значення напруги (Vtref) і фактичне значення напруги на статорі (vs_dq). На виході формується сигнал, що управляє збудженням синхронного генератора (Vf).

Для моделювання процесу керування й моніторингу використовувалися два комп'ютери з'єднані через послідовний порт за допомогою нуль-модемного кабелю. При створенні моделі пульта керування (рис. 3.28) використовувались бібліотеки «Dials and Gauges Blockset» і «RS232 Blockset». У даному прикладі бібліотека для роботи із COM-портом представлена блоками: «RS232 Setup», «RS232 Write Format» і «RS232 Read Format».

Модуль «RS232 Setup» виконує підключення до COM-порту. У вікні настроювань установлюються номер порту, швидкість передачі, кількість біт переданих даних, кількість стопових біт і наявність перевірки парності. Вихідним сигналом цього модуля є дескриптор відкритого порту, що використовується іншими блоками. Для передачі даних використовується блок «RS232 Write Format». На вхід цього блоку надходять дані, які будуть передаватися. Якщо необхідно передати масив, то для формування вектора

використовується блок мультиплексора (Mux). У діалоговому вікні настроювання параметрів блоку вказується маска, відповідно до якої буде сформований пакет для передачі. Наприклад, якщо на вхід блоку надходить вектор з 4-х елементів, то для передачі цих даних маска повинна мати вигляд: msg1;%f,%f,%f,%f;end. В одній моделі може використовуватися кілька блоків передачі даних, причому працювати вони можуть із різними портами. Другий параметр, який необхідно вказати – це інтервал передачі (Sample time). Пакети даних будуть формуватися й передаватися в порт через проміжок часу, що відповідає цьому значенню. У наведеній моделі період передачі пакетів даних становить 0,1 с і при необхідності може бути зменшений. Однак необхідно стежити за тим, щоб час передачі чергового пакета даних був менше інтервалу часу між пакетами. У противному випадку можлива втрата даних. Також при малих значеннях інтервалу передачі можливо істотне вповільнення процесу моделювання.

Для зчитування даних з порту використовується модуль «RS232 Read Format». У вікні настроювань параметрів модуля вказується маска, відповідно до якої формується вихідний вектор, що розділяється на скалярні величини за допомогою блоку демультиплексора (Demux). Ця маска точно відповідає зазначеній в блоці передачі даних. Також налаштовується час вибірки – період опитування порту на наявність прийнятих даних. Якщо необхідно обробляти дані в міру надходження, то це значення встановлюється рівним –1.

Принцип роботи моделі полягає в наступному: на одному з комп'ютерів запускається модель пульта керування (рис. 3.28), а на іншому – модель дизель-генератора, що працює на певне навантаження (рис. 3.26).

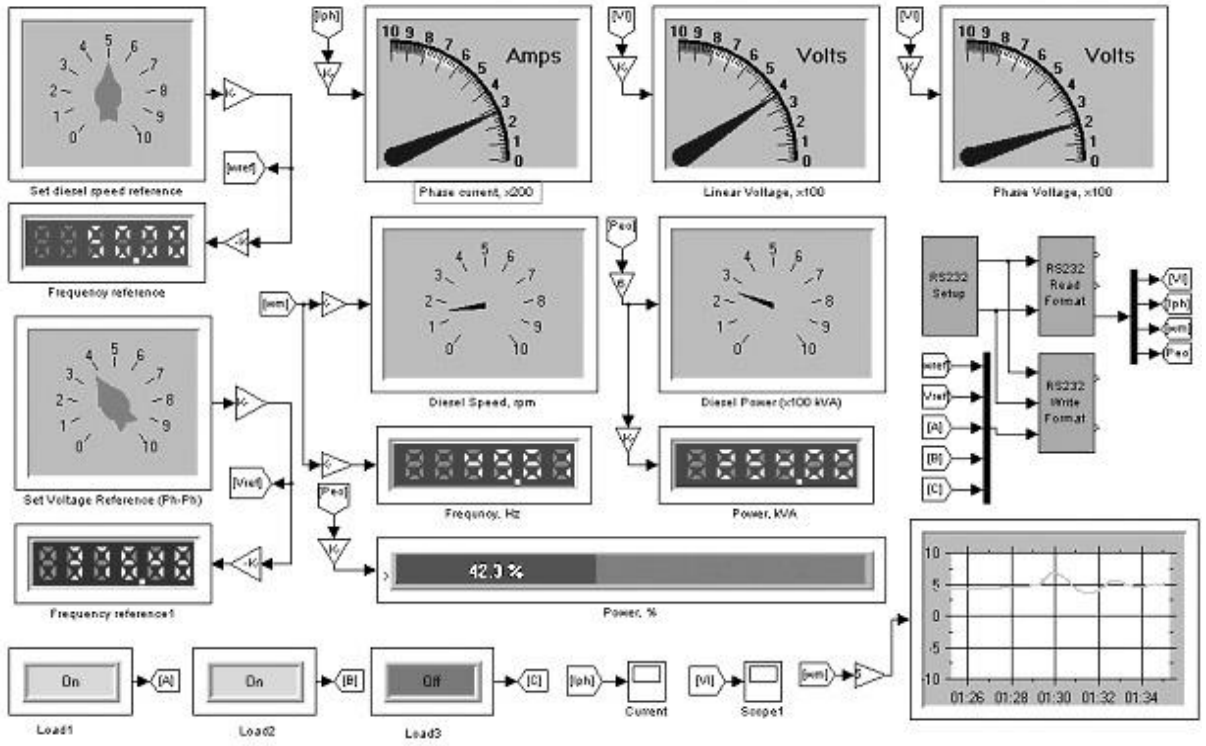


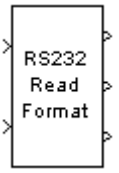
Рис. 3.28. Модель контролюючого устаткування

За допомогою елементів керування на моделі контролюючого устаткування (у цьому випадку це два блоки «Generic Knob» і три кнопки - «Square Red») задаються необхідні значення частоти напруги (оборотів дизеля) і величина лінійної напруги на обмотках статора генератора. Кнопки використовуються для підключення до генератора різного навантаження. У свою чергу виконується моніторинг таких параметрів, як струм і напруга статора генератора, обороти дизеля, частота напруги мережі й потужність, споживана навантаженням (числове значення й у відсотках від номінальної потужності генератора). Також використаний віртуальний осцилограф для того, щоб відслідковувати характер зміни контрольованих параметрів об'єкта.

У табл. 3.2 приводиться докладний опис модулів бібліотеки для роботи із COM-портом, які були використані в моделі.

Таблиця 3.2

Опис блоків бібліотеки «RS232 Blockset»

	Цей блок ініціалізує послідовний порт із заданими параметрами, а по завершенні роботи закриває його. Блок має два вихідні порти: **– Output1**: порт передає керуючий сигнал для відкриття зв'язку, до нього повинні бути підключені всі блоки, що працюють через цей порт. **– Output2**: порт використовується, щоб повідомити стан порту (0: закритий, 1: відкритий) в даний момент.
	Цей блок відправляє на послідовний порт форматований рядок даних (аналог функції *sprintf* у мові С). Блок має три вхідні порти: **– Input1:** з'єднується з портом керуючого сигналу блоку «RS232 Setup». **– Input2:** на цей порт подається сигнал, що дозволяє виконання запису (0: не дозволена, 1: дозволена). **– Input 3:** цей порт містить дані, передані через COM-порт Блок має два вихідні порти: **– Output1**: повторює керуючий сигнал, що приймається через порт Input1. **– Output2**: передає вихідний сигнал, що вказує на результат процедури запису (запис пройшов чи успішно ні).
	Цей блок читає байти на послідовному порту, додає їх у вхідний буфер для форматування їх у рядок (аналог функції s*scanf* у мові С), видаляє файли з буфера, які не відповідають запиту. Блок має три вхідні порти: **– Input1**: з'єднується з портом керуючого сигналу блоку «RS232 Setup». **– Input2:** на цей порт надходить сигнал, що дозволяє початок операції читання (0: дозволено, 1: заборонено). Блок має три вихідні порти: **– Output1**: повторює керуючий сигнал, що приймається через порт Input1. **– Output2:** передає вихідний сигнал, що вказує на результат процедури читання (читання пройшло чи успішно ні). **– Output3:** порт, що містить форматований рядок даних прочитаних з послідовного порту.

Використання описаних модулів дозволяє передавати й приймати дані з Matlab в інші додатки, наприклад Proteus, у якому створена мікропроцесорна система керування. Також з використанням модулів для роботи із COM-портом можливе керування моделлю електростанції, створеної в Matlab.

Постійно зростаюча складність керованих людиною машин і механізмів, об'єктів і процесів вимагають нових методів і засобів для навчання персоналу. Matlab дозволяє створювати не тільки системи

моніторингу, але й вилученого керування устаткуванням, наприклад електростанцією. При цьому дані можуть передаватися як по послідовному інтерфейсі RS232, так і по локальній мережі Ethernet. Таке керування дозволяє вивести персонал з області роботи устаткування, тим самим поліпшити умови праці на виробництві. При запусканні процесу моделювання можлива ситуація, коли масштаб часу моделювання буде різним для моделі електростанції й контролюючого устаткування. Якщо ж з використанням Matlab виконується керування деяким фізичним об'єктом, то необхідно виконувати її в режимі реального часу. У такому випадку необхідно використати спеціальний блок – Real Time Blockset, у якому настроюється єдиний параметр – інтервал часу між виконаннями розрахунків чергової ітерації моделювання. Однак робота в реальному масштабі часу можлива на досить потужних комп'ютерах.

На рис. 3.29 представлена схема мікропроцесорної системи керування, що виконує керування моделлю електростанції. Схема містить рідкокристалічний алфавітно-цифровий індикатор, за допомогою якого відображається значення струму статора, оборотів дизеля й потужності навантаження.

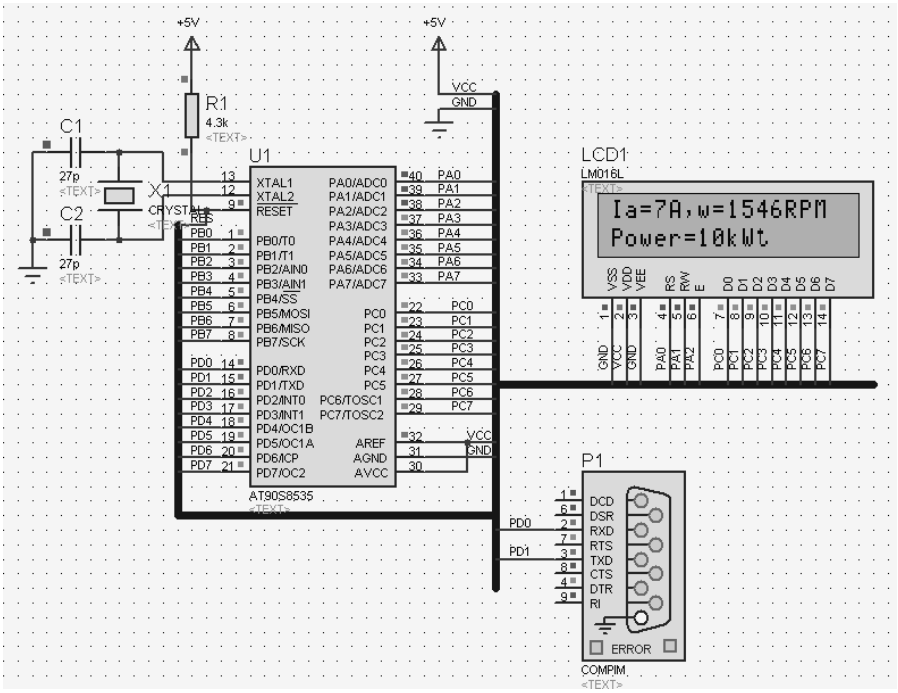


Рис. 3.29. Модель мікропроцесорної системи керування в Proteus

Структура пакетів даних, прийнятих з Matlab, значно полегшує процедуру «розпізнавання», оскільки для цих цілей може бути використана бібліотечна функція scanf. Для формування пакетів, які будуть передаватися в Matlab, використовується функція sprintf. Фрагмент програми для мікроконтролера, що виконує прийом даних і висновок прийнятих значень на рідкокристалічний індикатор, представлений нижче.

```
void main(void)                // головна функція
{
int f1=0, f2=0, f3=0, f4=0;   // оголошення змінних
int r1=5000, r2=10;
char text1[16];     // оголошення масивів для формування рядків
char text2[16];
init_cpu();    // виклик функції ініціалізації контролера
init_lcd();    // виклик функції ініціалізації ЖКІ
DDLCD=0xFF;
sprintf(text1,"Ia=%d,w=%dRPM",f1,f2);   // формування рядків
sprintf(text2,"Power=%d kWt",f3);
clear_lcd();         // очищення екрана ЖКІ
out_text(text1,0);  // висновок текстових рядків на ЖКІ
out_text(text2,64);
while (1)            // вічний цикл
{
   scanf("msg1;%d,%d,%d,%d;end",&f1,&f2,&f3,f4);
   sprintf(text1,"Ia=%d,w=%dRPM",f2,f3); // формування рядків
   sprintf(text2,"Power=%dkWt",f4);
   clear_lcd();                    // очищення екрана ЖКІ
   out_text(text1,0);              // виведення тексту на екран
   out_text(text2,64);
   printf("msg2;%d,%d;end",r1, r2);     // передача пакета даних
   delay_ms(100);                       // затримка 100 мс
};
}
```

3.8.2. Аналіз впливу динаміки Ethernet-мережі на якість керування

Для рішення поставленої задачі необхідно виконати побудову моделі електростанції й регулятора, між якими ввімкнена мережа як елемент передачі даних. Керування збудженням синхронного генератора виконується з використанням дистанційного ПІД-регулятора.

Структурна схема розподіленої системи керування з ПІД-регулятором представлена на рис. 3.30.

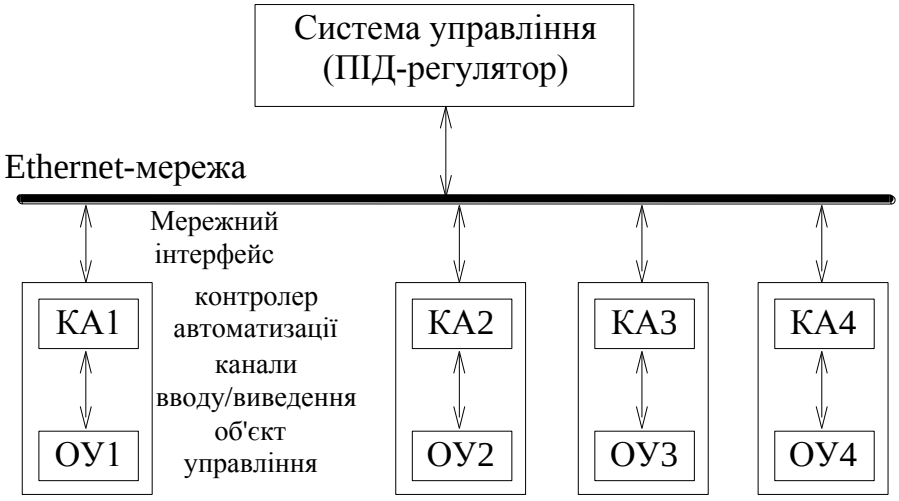


Рис. 3.30 Структурна схема розподіленої системи

Об'єктом керування (ОК) є обмотка збудження синхронного генератора. Контролер автоматизації (система збудження) через мережний інтерфейс зв'язана з хост-контролером, на якому знаходиться ПІД-регулятор. Збір даних і керування здійснюється через канали введення/виводу.

Структурна схема контуру дистанційного ПІД-регулятора представлена на рис. 3.31.

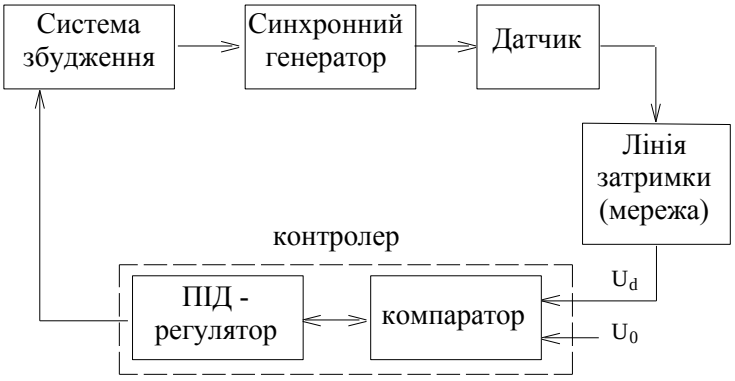


Рис. 3.31. Структурна схема контуру

Лінія затримки вносить запізнювання сигналу на величину, яка залежить від інтервалу дискретизації й інтенсивності запитів:

$$T = f(\nu, \lambda). \qquad (3.34)$$

Розроблена модель включає мінімальний набір об'єктів, зв'язаних між собою, і які допускають зовнішнє керування. Генератор представлений моделлю синхронної машини з демпферною обмоткою. Параметри машини задаються в системі абсолютних одиниць. Потужність генератора 400 кВт, лінійна напруга на статорі 380 В. Генератор має 2 пари полюсів (номінальна частота обертання 1500 об/хв) (рис. 3.32.).

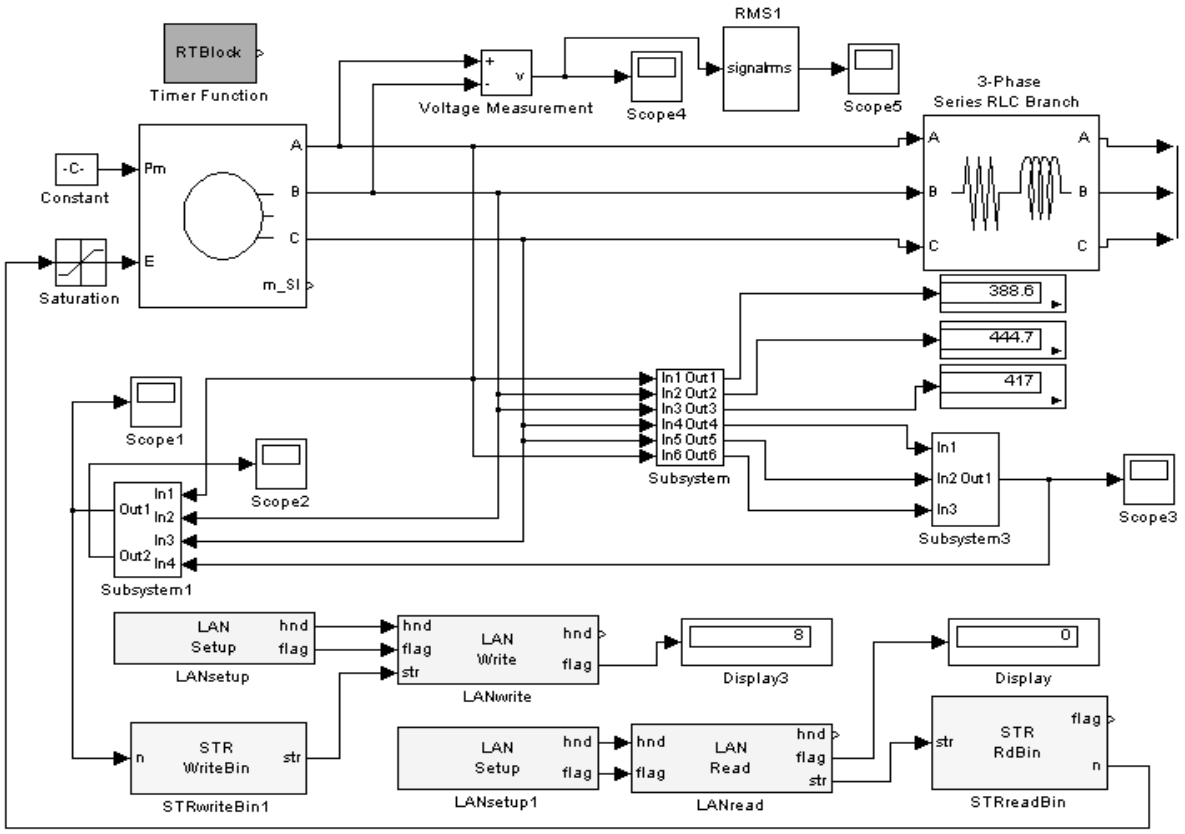


Рис. 3.32. MatLab-модель досліджуваної системи

Для моделювання процесу використовуються два комп'ютери з'єднані через локальну мережу Ethernet. При створенні моделі використовувались бібліотеки «CommStr6.5 Blockset». У даноій моделі використовуються блоки: «LAN Setup», «LAN Write» і «LAN Read». Для форматування даних, які передаються використані блоки «STR WriteBin» і «STR RdBin».

Модуль «LAN Setup» виконує налаштування з'єднання по локальній мережі. У вікні налаштувань встановлюється номер порту та IP-адреса

комп'ютера. Вихідним сигналом цього модуля є дескриптор відкритого порту, який використовується іншими блоками. Для передачі даних використовується блок «LAN Write». На вхід цього блоку надходять дані, які будуть передаватися. У діалоговому вікні налаштування параметрів блоку «STR WriteBin» вказується маска, відповідно до якої буде сформовано пакет для передачі.

Моделювання об'єкта керування припускає використання асинхронного обміну інформацією між додатками, які працюють на різних комп'ютерах. У розглянутому випадку використовується клієнт-серверна архітектура. Модель об'єкта керування (електростанції) повинна відповідати деяким вимогам. Модель не повинна обмежуватися моделюванням тільки одного об'єкта, наприклад генератора, без урахування впливу на нього електроенергетичної системи, в якій він працює. Крім того, модель повинна забезпечувати можливість одночасного моделювання декількох зв'язаних між собою об'єктів, а також виконувати моделювання в режимі реального часу..

Для забезпечення виконання моделювання об'єкта в режимі реального часу використано блок «RTBlock».

Основними показниками якості роботи регулятора, які обчислюються по перехідній характеристиці, є наступні [21]:

– перерегулювання σ:

$$\sigma = \frac{x_{m1} - x}{x}, \tag{3.35}$$

де x_{m1} – амплітуда першої хвилі перехідної характеристики,

x – стале значення параметра;

– коливальність δ:

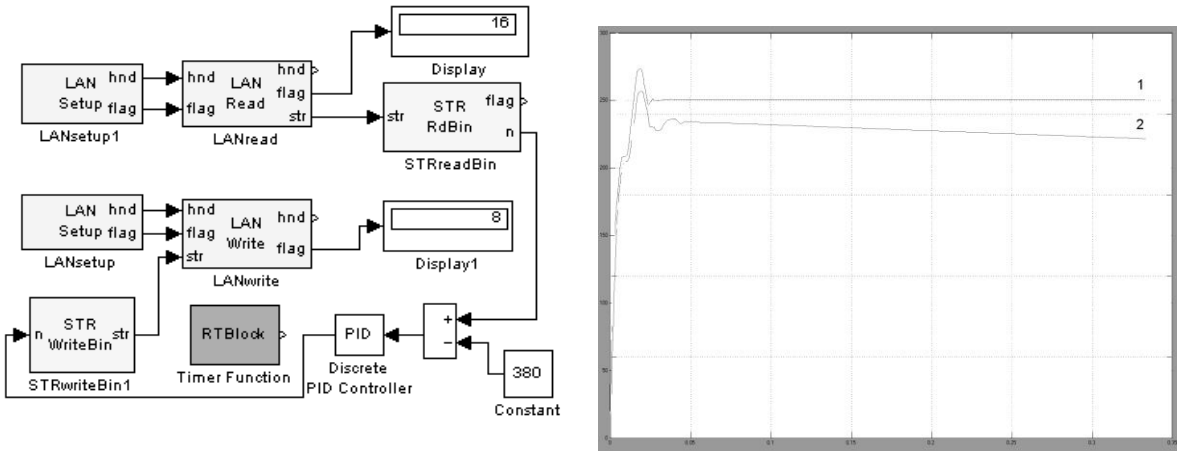
$$\delta = \frac{x_{m2}}{x_{m1}}, \tag{3.36}$$

де x_{m2} – амплітуда другої хвилі перехідної характеристики;

– інтегральна оцінка A:

$$A = \int_0^\infty (h(t) - h_{cm}(t))^2 dt. \tag{3.37}$$

На рис. 3.33 представлена модель дистанційного регулятора, яка містить безпосередньо сам регулятор, а також блоки для організації зв'язку по локальній мережі.



а) б)

Рис. 3.33. Модель дистанційного ПІД-регулятора та результати моделювання

В результаті моделювання були отримані дві криві перехідного процесу. Крива (2) є графіком перехідного процесу при моделюванні дистанційного ПІД-регулятора.

Таким чином, для керування об'єктами автоматизації в режимі реального часу необхідно мінімізувати затримки передачі пакетів по мережі. Цього можна досягти пошуком оптимального маршруту передачі повідомлень, оптимізації алгоритмів обробки потоків у вузлах мережі, розподіл потоків на класи обслуговування з метою виділення найбільш пріоритетних.

РОЗДІЛ 4
АВТОМАТИЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ РОЗПОДІЛЕНИХ СИСТЕМ КЕРУВАННЯ ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ UML

4.1. Статична UML – модель системи керування

В існуючий час росте інтерес до нових технологій проектування та до архітектур розподілених систем керування (СК) для автоматизації виробництва. Важливим моментом у процесі проектування СК є реалізація гнучкості й можливості швидкої реконфігурації. При цьому засоби проектування й моделювання систем керування повинні мати можливість швидкої модифікації проекту для адаптації до конкретного об'єкта керування [30].

Для правильного рішення питання керування технічним об'єктом важливе знання процесів, які відбуваються в об'єкті, і методології вивчення процесу керування, основою якого є системний підхід [38]. Для кожного елемента вказується безліч можливих режимів роботи. Сукупність режимів всіх елементів визначає режими роботи об'єкта [125].

Уніфікація й стандартизація інтерфейсів дозволили перейти до побудови систем, заснованих на просторовому й функціональному розподілі складових частин, а також розподілі їх по керуванню. Заміна одного мікропроцесора (МП) декількома дає можливість розосередити їх по всьому судну. На кожний із МП можна покласти певні функції, що дозволить програмісту модифікувати робочу програму, що обслуговує окрему підсистему, без будь-якого втручання в робочі програми інших підсистем.

Використання уніфікованої мови моделювання UML при розробці розподілених систем керування, якою і є система керування електроенергетичною установкою, дозволяє значно полегшити й прискорити процес проектування шляхом реалізації загальних функціональних блоків з їх подальшим повторним використанням і можливістю вдосконалення.

При розробці розподілених систем керування повинні братися до уваги такі положення як створення програмного коду керуючого додатка, розбивка коду на функціональні модулі з подальшим розташуванням у керуючих пристроях і вибір топології системи керування й реалізація зв'язку між керуючими блоками [33, 152, 158, 159].

Для кожної генераторної секції розробляються схеми керування з урахуванням прийнятого обсягу автоматизації кожного генераторного агрегату. На рис. 4.1 представлена UML – діаграма однієї генераторної секції й зв'язку між окремими функціональними пристроями.

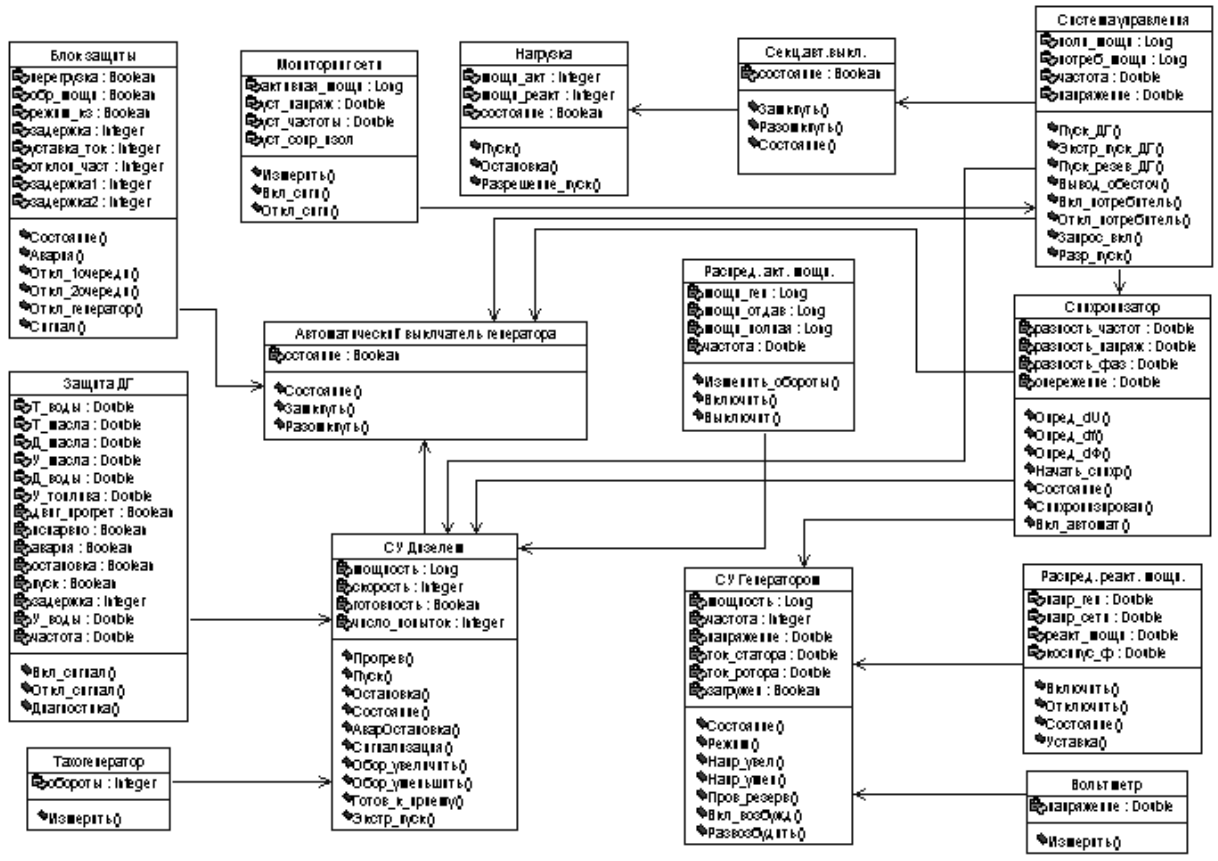


Рис. 4.1. Структурна схема генераторної секції

Модель на рис. 4.1 містить систему керування дизель-генератором (ДГ), блок захисту ДГ, пристрою розподілу активної/реактивної потужності, синхронізатор, блоки захисту й моніторингу мережі, а також деякі вимірювальні прилади. Кожен із пристроїв представлений у вигляді класу, що має набір функцій, за допомогою яких описуються алгоритми роботи

пристроїв і набір атрибутів, що визначають поточний стан кожного із пристроїв. Залежно від розв'язуваних конкретною системою керування задач, набір функцій кожного із пристроїв може мінятися.

Система керування дизельним агрегатом виконує наступні функції: підтримка ДГ у прогрітому стані (Прогрів()); пуск ДГ із холодного або прогрітого стану й вивід його на номінальну частоту обертання з видачею сигналу в систему керування про готовність до прийому навантаження (Екстр_пуск(), Пуск(), Готовий_до_прийому); захист ДГ при відхиленні параметрів (тиску масла й води, температури масла й води, частоти обертання) з видачею сигналу розшифровки причини аварії (Діагностика(), Вкл_сигнал(), Сигналізація()); видачу попереджувальних сигналів про стан параметрів ДГ (Стан()); зупинку ДГ по спеціальній програмі або екстрену (аварійну) зупинку (Зупинка(), Авар зупинка()). Кожний клас на діаграмі містить атрибути, з якими оперує система керування. Для дизеля це такі параметри як потужність, номінальна частота, стан (готовність до підключення навантаження), число спроб запуску. При роботі системи керування значення цих параметрів постійно обновляються, і залежно від їхніх значень відбувається зміна режимів роботи ДГ (стабілізація частоти обертання, перевірка наявності резерву потужності, діагностика).

Після того, як розроблена статична модель системи керування, виконується опис алгоритмів роботи системи керування АЕЕС і окремих пристроїв автоматизації при різних режимах роботи. Повнота й точність опису алгоритмів дозволить виконати функціональну розбивку завдань по керуванню електроенергетичною системою й визначитися з кількістю мікропроцесорів, необхідних для рішення завдання керування із заданою якістю й надійністю. У свою чергу це дозволить описати типи зв'язків між окремими керуючими модулями, визначити обсяг переданої інформації.

До складу програмного забезпечення повинні входити засоби для автоматизації розробки програм і мікропрограм, і їхню перевірку. Цим вимогам задовольняє пакет Rational Rose [142, 143].

Систему алгоритмів керування АЕЕС будують за ієрархічним принципом, що забезпечує зручність розробки, рішення завдань програмування й налагодження її елементів. У ній можуть бути виділені структурна, функціональна й організаційна ієрархії [10].

На рис. 4.2 представлена діаграма послідовності, що описує послідовність включення потужного споживача електроенергії. З появою заявки на включення якого-небудь потужного споживача формується сигнал, що імітує потужність, що потрібна і системою керування визначається достатність наявного резерву потужності, що генерується. Це завдання система керування виконує шляхом звертання до системи керування дизель-генератором (команда Пров_резерв()). У систему керування повертається значення доступної потужності (атрибут «завантаження»). Якщо це значення перевищує потужність споживача, що підключається, то видається сигнал на включення автоматичного вимикача. Якщо ж резерв потужності відсутній, система керування передає команду пуску резервного дизель-генератора. Коли оберти дизеля досягнуть номінального значення, подається сигнал на включення збудження генератора. Далі відбувається опитування системи керування дизелем на предмет готовності до прийняття навантаження. Виконується синхронізація генератора з мережею й після того, як синхронізація виконана, подається команда на дозвіл включення споживача. Діаграма послідовності показує взаємодію між окремими блоками системи керування [30].

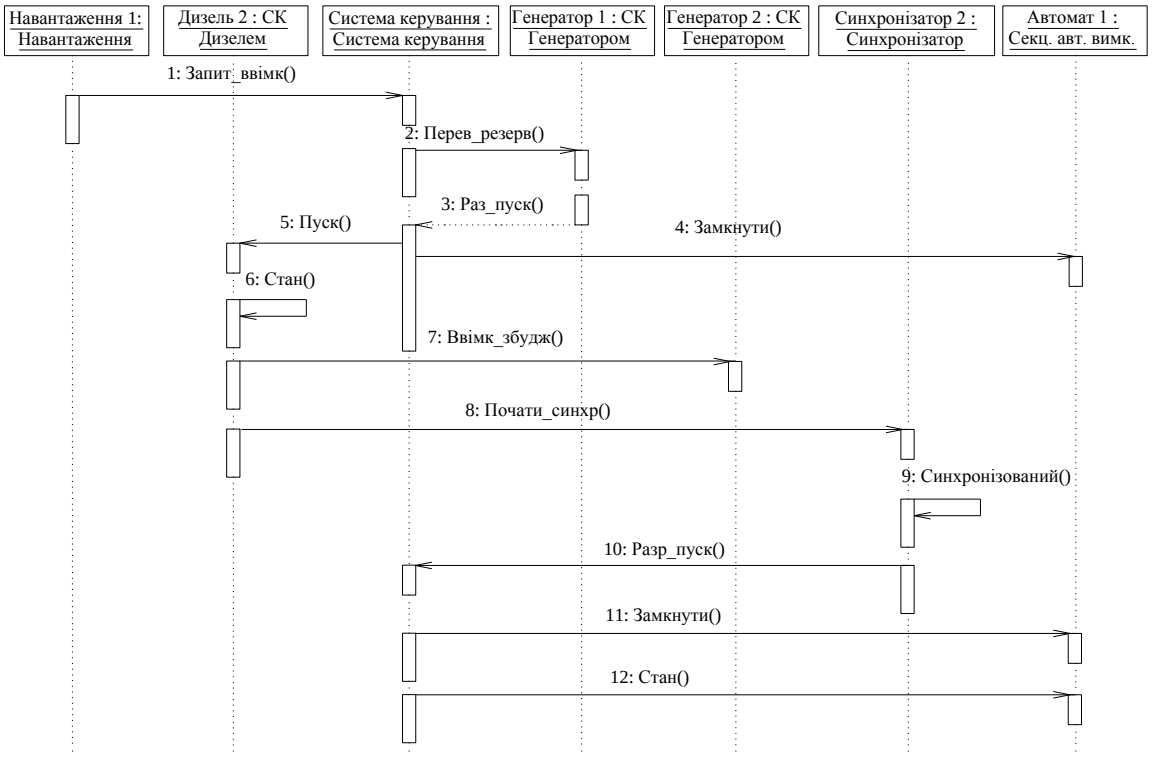


Рис. 4.2. Послідовність включення потужних споживачів

Організаційна ієрархія алгоритмів керування АЕЕС припускає наявність у системі елементів з різними пріоритетами дії. Причому є елементи рівних пріоритетів і елементи, що приймають рішення, які розташовуються ієрархічно. Дана ієрархія дозволяє здійснити класифікацію елементів системи алгоритмів по пріоритетах, забезпечити їхню взаємодію й реалізувати режим мультипрограмування. Реалізація режиму поділу часу й виконання програм у реальному масштабі часу забезпечуються відповідним вибором тимчасових характеристик функціональних елементів системи алгоритмів на підставі аналізу функціональних особливостей алгоритмів і програм, вивчення перехідних процесів у АЕЕС.

4.2. Розробка UML – моделі інформаційних потоків

В основі побудови динамічної моделі лежать раніше розроблені прецеденти (події). У АЕЕС до таких прецедентів відносяться:

– точна автоматична синхронізація генераторів;

– автоматичний розподіл активного навантаження між паралельно працюючими генераторами;

– автоматичне розвантаження генераторів, включення резервного генератора;

– включення потужних споживачів електроенергії;

– екстрений пуск дизель-генераторів і виведення судна зі знеструмленого стану;

– контроль параметрів працюючого дизель-генератора, захисту по частоті обертання;

– контроль і діагностування електричної мережі;

– звукова й світлова сигналізація й ін.

Для кожного прецеденту треба визначити об'єкти, що беруть участь у ньому, і те, як вони взаємодіють у цьому прецеденті. Ці взаємодії відображаються на діаграмі кооперації.

Розглянемо діаграму взаємодії при екстреному пуску резервних ДГ і виведенні судна зі знеструмленого стану. Екстрений пуск резервних ДГ необхідний для забезпечення електропостачання судна в аварійних ситуаціях. Команда на екстрений пуск формується при наявності однієї з наступних умов: напруга на шинах ГРЩ дорівнює нулю або є заявка на включення потужного споживача електроенергії, що викликає дефіцит генеруємої потужності. Сигнал екстреного пуску блокує затримку його пуску й датчик попереднього прокачування масла (цим скорочується час пуску ДГ).

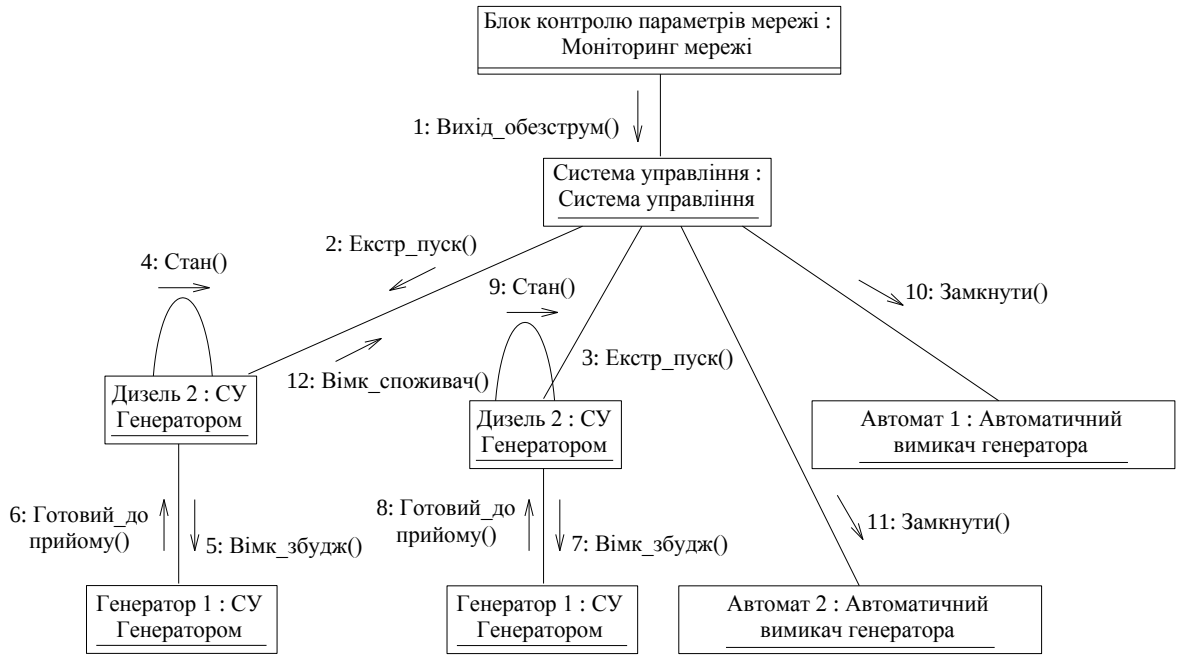


Рис. 4.3. Діаграма взаємодії при екстреному пуску ДГ

Знеструмлення АЕЕС оцінюється її системою керування по зникненню напруги на шинах ГРЩ або відключенню всіх автоматичних вимикачів генераторів. Ці функції виконує блок контролю параметрів мережі. При знеструмленні система керування електроенергетичною установкою забезпечує екстрений пуск мінімум двох резервних ДГ. Перший ДГ, що досяг номінальної частоти обертання, після подачі порушення на генератор, одержує команду включення на шини ГРЩ. Якщо він за якимись причинами не включиться, він буде блокований і наступний ДГ одержить аналогічну команду. Пуск декількох ДГ передбачається для запобігання невдалого пуску одного з них в умовах знеструмлення судна й надійного забезпечення швидко зростаючого навантаження після знеструмлення.

Сигнал на включення автоматичного вимикача резервного ДГ формується при виконанні наступних умов: відсутнє блокування ДГ (немає аварії, частота перевищує мінімально припустиму межу, немає команди на включення автоматичних вимикачів інших генераторів). При цьому можливо підключення тільки один резервного ДГ, що першим вийшов на задану частоту обертання. Сигнал на включення автоматичного вимикача подається

протягом певного часу, після закінчення якого його включення блокується. Якщо напруги на шинах ГРЩ немає, то надається можливість для підключення наступного із тих, що запустилися ДГ.

При навантаженні працюючого ГА близького до 80% його номінальній потужності, можливі включення споживачів електроенергії, потужність яких перевищує 20% резерву потужності на шинах ГРЩ. Підключення таких споживачів може викликати перевантаження працюючого ГА, його відключення й знеструмлення судна. Надійне включення в мережу потужних споживачів електроенергії забезпечується системою керування СЕЕУ. Оцінюється потужність навантаження, яке вмикається, і якщо потужність електростанції недостатня, пуск цього навантаження блокується доти, поки резервний ДГ не буде пущений і синхронізований із шинами ГРЩ. Попередня оцінка резерву генеруючої потужності дозволить виконати запуск дизель-генератора й зробити його синхронізацію з мережею для виключення можливості знеструмлення судна. На рис. 4.4 представлена діаграма, що описує взаємодію об'єктів при відсутності резерву потужності.

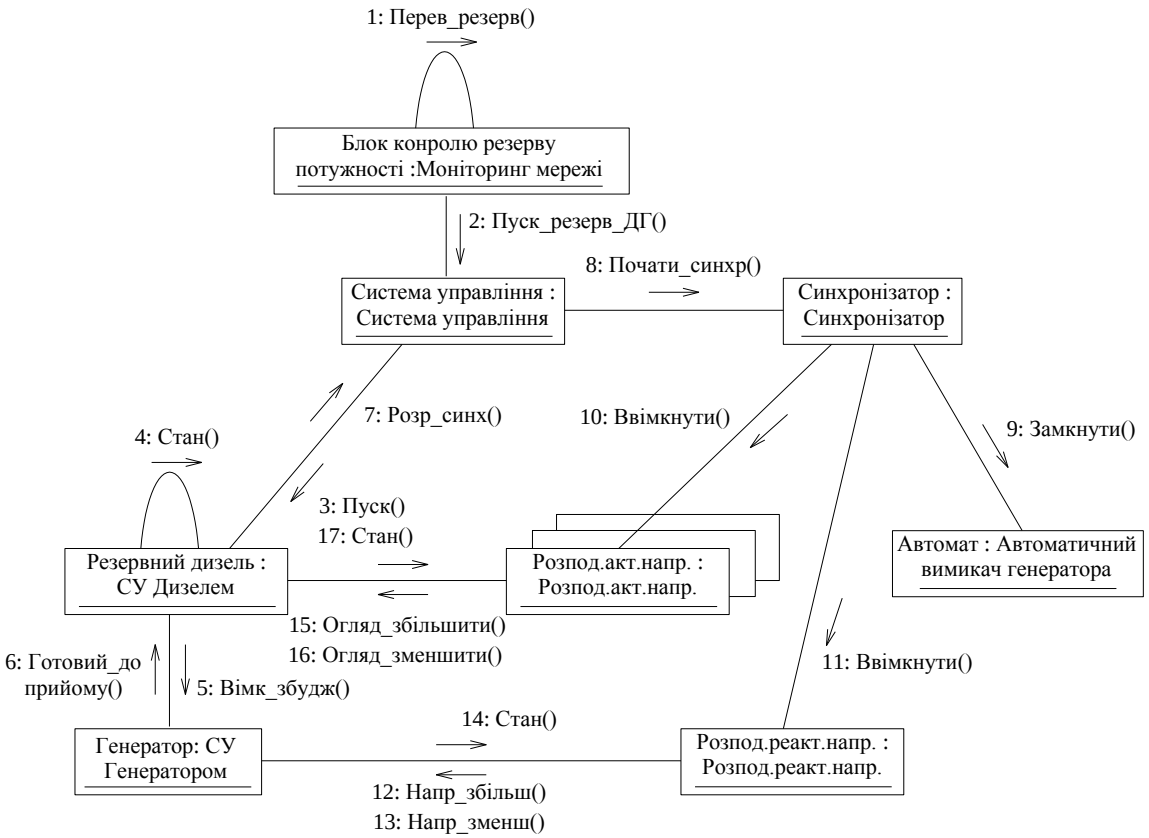


Рис. 4.4. Взаємодія об'єктів при введенні в роботу резервного ДГ

Систему керування можна розділити на кілька підсистем. Динамічні взаємодії між підсистемами можна зображати на діаграмі кооперації підсистем, що являє собою високорівневу консолідовану діаграму кооперації.

Консолідована діаграма взаємодії, яка показана на рис.3, синтезується шляхом накладення одна на одну діаграм кооперацій для окремих прецедентів. У кожному випадку в консолідовану діаграму включаються нові об'єкти й повідомлення, тому вона постійно ускладнюється. Деякі об'єкти або повідомлення можуть бути присутніми декілька разів, але зображуються на консолідованій діаграмі тільки один раз. Розробка консолідованої діаграми кооперацій для кожної підсистеми й високорівневої діаграми кооперації підсистем дозволяє оцінити ступінь зв'язку між об'єктами у всій ієрархії системи керування. На консолідованій діаграмі взаємодії, що покликана дати повну картину всіх комунікацій, повинні бути представлені всі повідомлення. У такому випадку відображуваної інформації на діаграмі виявляється більше. Якщо один об'єкт посилає іншому кілька окремих повідомлень, то вони показуються не окремо, а в складі агрегованого повідомлення.

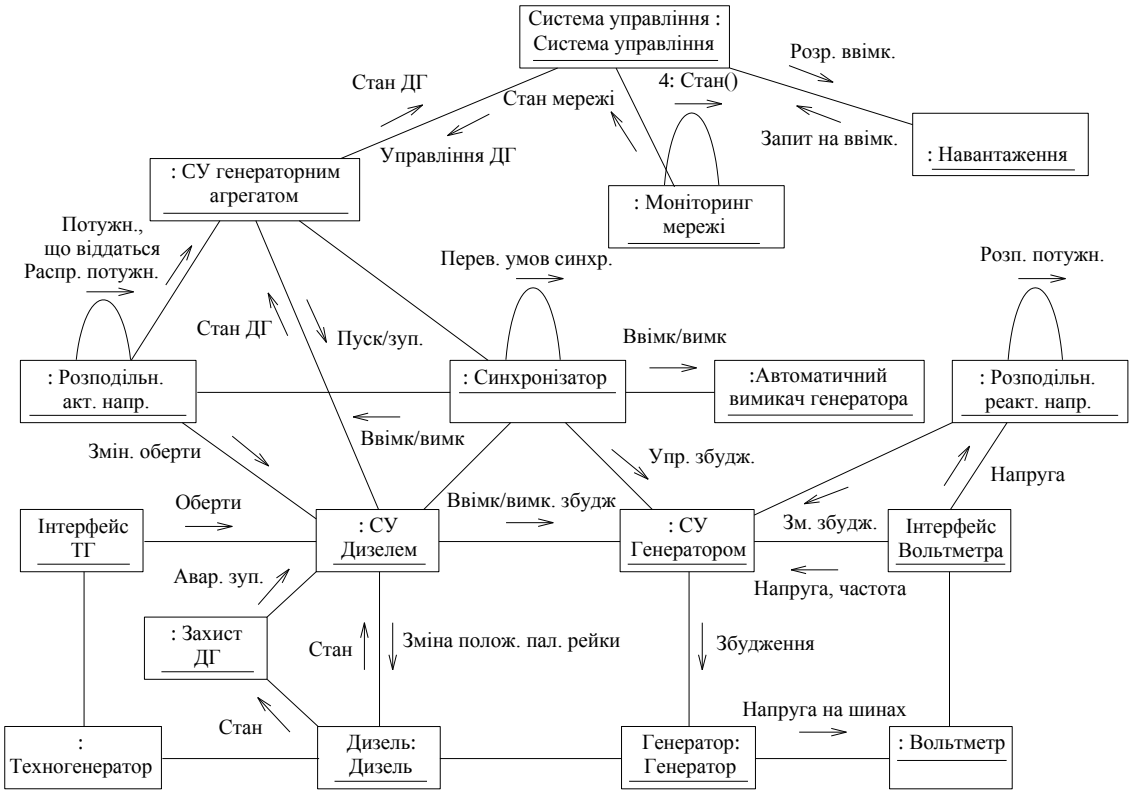


Рис. 4.5. Консолідована діаграма взаємодії

Мета динамічного аналізу полягає у визначенні взаємодії між керуючим об'єктом, об'єктами, які посилають повідомлення керуючому об'єкту, а також іншими об'єктами, які також беруть участь у прецеденті. Взаємодії можуть відбуватися між об'єктами як одного рівня ієрархії, так і різних рівнів.

Динамічний аналіз дозволяє розібратися в тому, як об'єкти взаємодіють між собою в різних прецедентах. Динамічний аналіз виконується для кожного прецеденту. Він починається з розгляду зовнішньої події, що надходить в об'єкт, після чого простежується весь ланцюжок внутрішніх подій. При виникненні внутрішніх подій відбувається обмін повідомленнями між об'єктами-учасниками прецеденту. Використання уніфікованої мови моделювання дозволяє виконати аналіз всіх повідомлень, якими обмінюються об'єкти в процесі взаємодії.

4.3. Використання UML-FB в проектуванні та моделюванні систем керування автономними електростанціями

Існуючі технології проектування й побудови складних інформаційних систем технічного напрямку орієнтовано на відомі технології, які характеризуються значною надмірністю, високою вартістю й складністю при рішенні комплексу задач. Фактично це пояснюється складністю в узагальненому описі таких систем.

До останнього часу для рішення таких задач використовувалися автоматні методи [11, 144–147], а також методи системного аналізу й синтезу. В останні роки для проектування, специфікації, візуалізації, конструювання й документування різних систем використовується мова UML, що представляє собою об'єднання інженерних прийомів, які раніше успішно використовувалися при моделюванні великих і складних систем [142]. Але, як показала практика, використання інструментів UML при розробці систем реального часу, зокрема систем керування, досить складне [39].

З метою рішення цієї проблеми було розроблено стандарт UML-FB (IEC 61499), у якому об'єкти представлені не у вигляді класів, а у вигляді функціональних блоків (ФБ), ідеалізованих "пристроїв", що виконують ту або іншу функцію, відповідно до внутрішнього алгоритму, і, що мають вхідні й вихідні події й змінні. На основі таких ФБ можлива побудова систем реального часу будь-якої складності з контролем чисельних параметрів, а, отже, – проведення аналізу ідеалізованої системи. Однак розвиток цього стандарту для аналізу й синтезу технічних систем поки мало досліджено.

Розглянемо як приклад побудову моделі розподіленої системи керування електроенергетичної системи, що складається з 3-х дизель-генераторів з використанням програмного пакета FBDK. В якості середовища для розробки був обраний програмний продукт FBDK (Function Block Development Kit, Holobloc Inc; Version 2006-10-17) [161].

Розглянемо побудову моделі розподіленої системи керування електроенергетичної системи, що складається з 3-х дизель-генераторів з використанням програмного пакета FBDK. В якості середовища для розробки був обраний програмний продукт FBDK (Function Block Development Kit, Holobloc Inc; Version 2006-10-17).

Першим кроком у побудові моделі є загальний аналіз системи [111–113]. Необхідно визначити всі об'єкти, що входять у модель, а також їх інтерфейси. Для рішення цього завдання зручно використовувати діаграму класів, на якій кожен об'єкт представлений у вигляді окремого класу зі своїми аргументами (змінними, що характеризують стан або режим роботи об'єкта керування) і методами (фактично, функціями об'єкта). Для електроенергетичної системи це такі об'єкти, як дизелі, генератори, засоби індикації стану, навантаження, блок керування. Завдання керування структурою енергетичної системи (підключення/відключення дизель-генераторів, навантаження, завдання розподілу потужності, функції захисту) виконуються блоком керування й при моделюванні реалізуються програмно.

Дизель представляє собою двигун із системою стабілізації обертів, що управляє рівнем подачі палива (залежно від навантаження на валу). Основні стани дизеля наступні:

– не запущений;

– запущений;

– навантаження на валу не перевищують максимальну потужність двигуна – номінальна частота обертання вала підтримується за рахунок збільшення подачі палива. Приводиться у відсотках від максимального значення

$$Acceleration = \frac{P_{load}}{P_{max}} \cdot 100\%$$

– навантаження на валу більше максимальної потужності двигуна. У цьому випадку подача палива максимальна (і не зростає), частота обертання вала починає падати й визначається формулою:

$$\Omega = 0.05 \cdot (P_{max} - P_{load}) - 157$$

де 0.05 – ваговий коефіцієнт впливу навантаження на валу на оберти двигуна, 157 – номінальна частота обертання (рад/с);

– навантаження на валу досягають критичного значення (на малих потужностях $P_{load_ÊÐÈÒ} = 2P_{max}$, або частота обертання ротора падає до 100 рад/сек) – аварійна зупинка двигуна.

Таким чином, ФБ "DIESEL" має три вхідних змінних (рис. 4.6) – максимальна потужність двигуна (VAR POWER_MAX тип REAL, задається користувачем), потужність навантаження на валу (VAR POWER_LOAD тип REAL, задається валом генератора), дозвіл на запуск (VAR ENABLE_START тип BOOL, установлюється системою); і одна вхідна подія – старт (EVENT "START"). Вихідними даними будуть – поточний рівень подачі палива у відсотках від максимального значення (VAR ACCELERATION тип REAL), частота обертання валу двигуна (VAR ROT_FREQ тип REAL), сигнал про перевантаження (VAR DROPOUT тип BOOL), сигнал про аварійну зупинку

(VAR EMERGENCY_STOP тип BOOL), інформація про виконання функції (EVENT "ON").

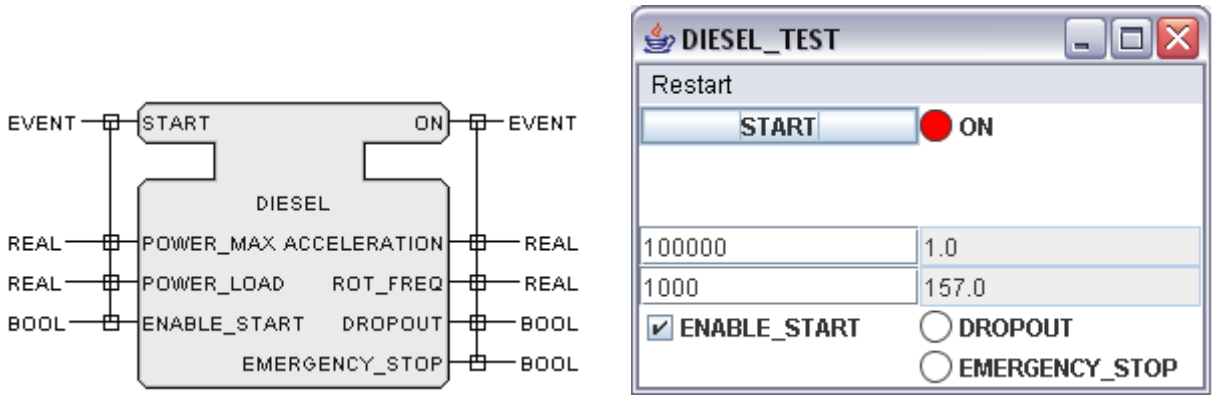


Рис. 4.6. Зовнішній вигляд ФБ "Дизель" і тестове вікно

Генератор представляє собою сам генератор і систему стабілізації рівня вихідної напруги, за допомогою зміни струму збудження. Основні стани генератора:

– відключений (якщо вал дизеля не обертається);

– включений;

– навантаження в мережі не перевищують максимальної потужності генератора (номінальне значення рівня напруги на виході підтримується за рахунок зміни струму порушення – значення якого приводиться у відсотках від максимального значення:

$$Stimulation = \frac{P_{load} \cdot 0.05 + 1 \cdot (157 - \Omega)}{P_{max} \cdot 0.05} \cdot 100\%$$

де 0,05 і 1 – вагові коефіцієнти впливу зміни параметрів на рівень вихідної напруги);

– навантаження в мережі перевищують максимальну потужність генератора або/і частота обертання ротора падає до критичного значення – порушення максимальне й не зростає, рівень вихідної напруги визначається формулою:

$$U = P_{max} \cdot 0.05 - P_{load_electric} \cdot 0.05 - (157 - \Omega) \cdot 1 + 380,$$

де 157 – номінальна частота обертання (рад/с),

380 – номінальна діюча напруга на виході;

– навантаження в мережі або/і частота обертання ротора досягають критичного значення, при яких

$$\frac{P_{load} \cdot 0.05 + 1 \cdot (157 - \Omega)}{P_{max} \cdot 0.05} \cdot 100\% > 200\%,$$

або напруга на виході менше 300В – аварійне відключення від мережі.

Частота вихідної напруги буде визначатися, виходячи з кількості пар полюсів генератора, і дорівнює:

$$f = \frac{\Omega}{\pi},$$

а потужність на валу:

$$P_{load_mechanic} = \frac{P_{load_electric}}{85\%},$$

з урахуванням ККД=85%.

Таким чином, ФБ "GENERATOR" (рис. 4.7) має три вхідних змінних: максимальні потужності генератора (VAR POWER_MAX тип REAL, задається користувачем), потужність навантаження в мережі (VAR POWER_LOAD_EL тип REAL), частота обертання валу дизеля (VAR ROT_FREQ тип REAL); і одна вхідна подія – старт (EVENT "ROTATE"). Вихідними даними будуть – поточний рівень струму порушення, у відсотках від максимального значення (VAR STIMULATION тип REAL), механічне навантаження на валу генератора (VAR POWER_LOAD_MC тип REAL), інформація про вихідний EPC: діюче значення (VAR VOLTAGE тип REAL), частота (VAR FREQ тип REAL), сигнал про перевантаження (VAR DROPOUT тип BOOL), сигнал про аварійне відключення (VAR EMERGENCY_SHUTDWN тип BOOL), інформація про виконання функції (EVENT "ON").

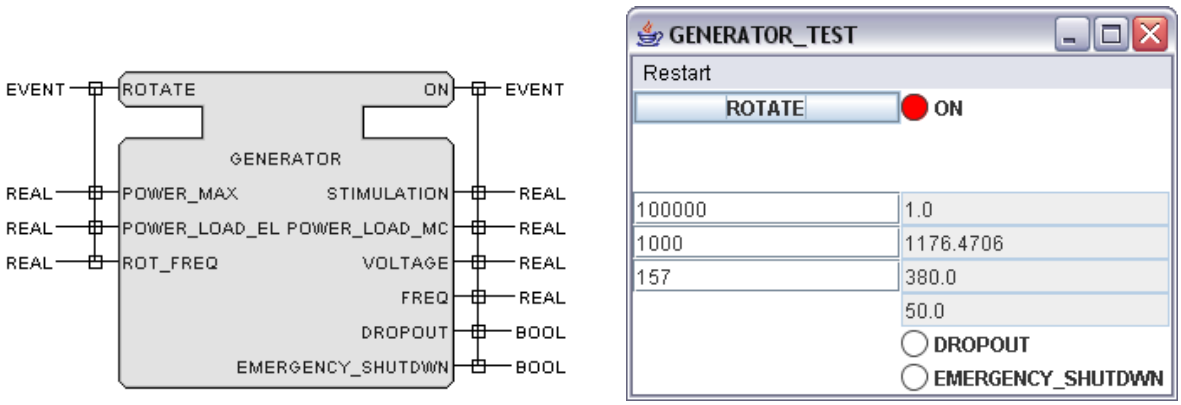


Рис. 4.7. Зовнішній вигляд ФБ "Генератор" і тестове вікно

На рис. 4.8 представлені алгоритми роботи ФБ дизеля й генератора.

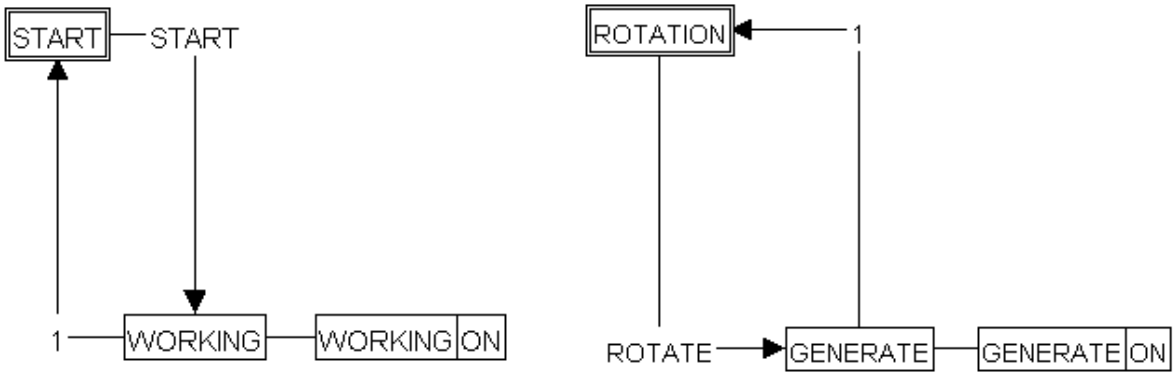


Рис. 4.8. Алгоритми роботи ФБ дизеля й генератора

Після того, як створені моделі дизеля й генератора, можна створити складовий ФБ дизель-генератора. Визначивши тип блоку, і вказавши напрямку потоків даних, вийде об'єднаний ФБ, що представляє собою модель дизель-генератора. У складеному блоці відсутні алгоритми, або програми – він усього лише визначає зв'язки і їх напрямки між базовими ФБ.

Вхідними параметрами блоку є:

– максимальна потужність генератора;

– максимальна потужність дизеля;

– потужність навантаження мережі;

– дозвіл на запуск.

Вихідними параметрами блоку:

– рівень вихідної напруги;

– частота вихідної напруги;

– рівень подачі палива в дизелі;

– рівень струму збудження в генераторі;

– сигнал про аварійне відключення генератора;

– сигнал про аварійну зупинку дизеля;

– сигнал про перевантаження дизеля;

– сигнал про перевантаження генератора.

На рис. 4.9 представлено вид складеного ФБ "DIESEL_GENERATOR" і вікна моделювання.

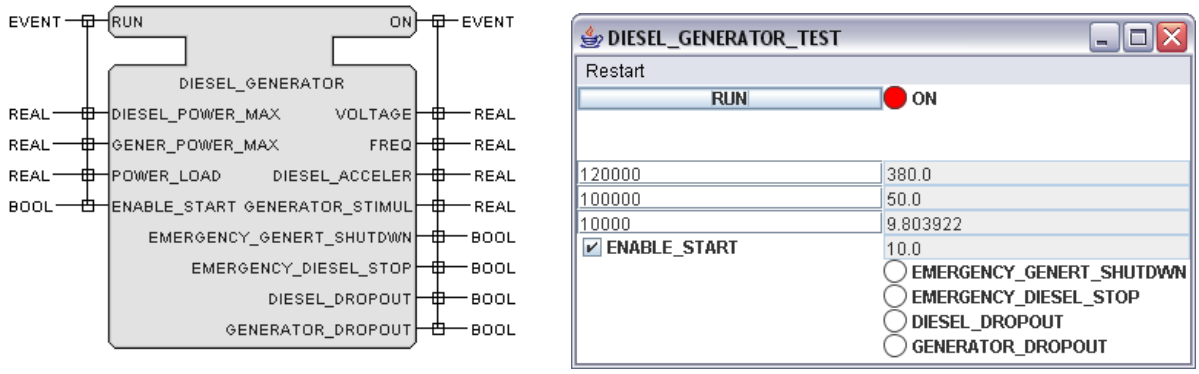


Рис. 4.9. Зовнішній вигляд і тестове вікно ФБ

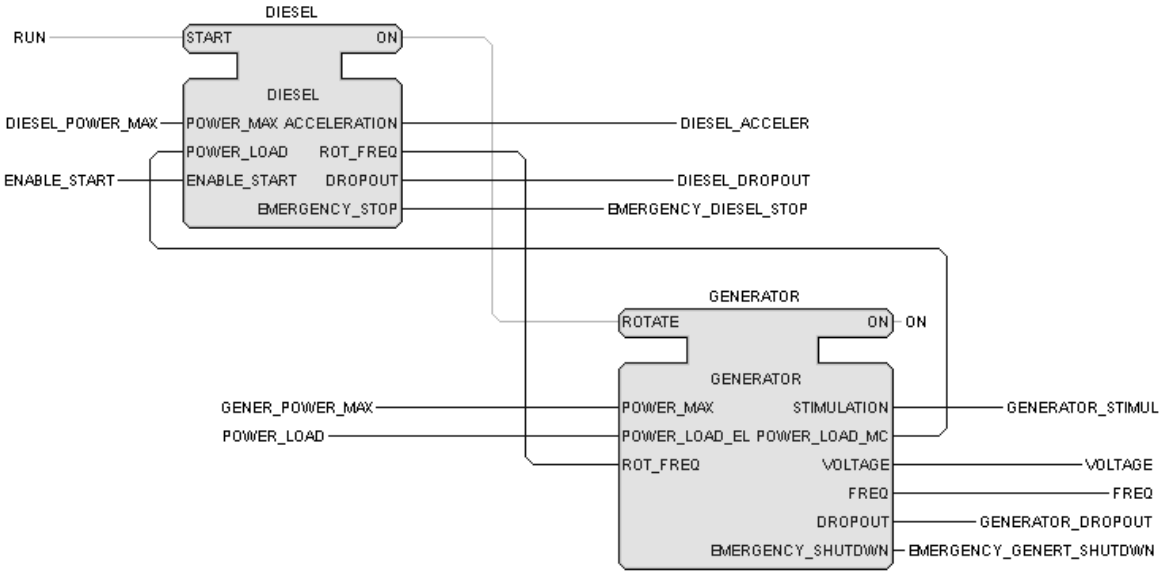


Рис. 4.10. Внутрішня структура блоку DIESEL_GENERATOR

На рис. 4.11 представлено тестове вікно для функціонального блоку дизель-генератора.

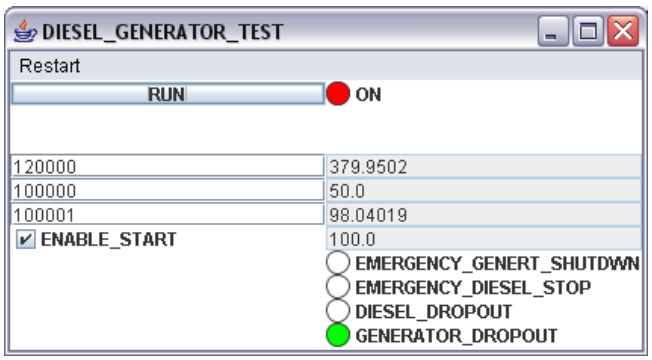


Рис. 4.11. Моделювання роботи дизель-генератора

Вхідними параметрами моделі є потужність дизеля, потужність генератора й потужність навантаження. У наведеному прикладі потужність навантаження більше потужності генератора, тому видається сигнал про перевантаження генератора. Також відображаються значення напруги мережі, частоти, рівень подачі палива й струм збудження. Ці параметри визначаються статичними характеристиками дизеля й генератора, які описуються на етапі проектування блоку. В окремих випадках більш важливим є перевірка логіки роботи системи, чим одержання точних числових значень. Тому можна використати спрощену модель дизель-генератора, у якій частота завжди підтримується рівної 50 Гц, а напруга на шинах 380 В.

На рис. 4.12 представлений зовнішній вигляд програми FBDK і функціонального блоку дизель-генератора (спрощений).

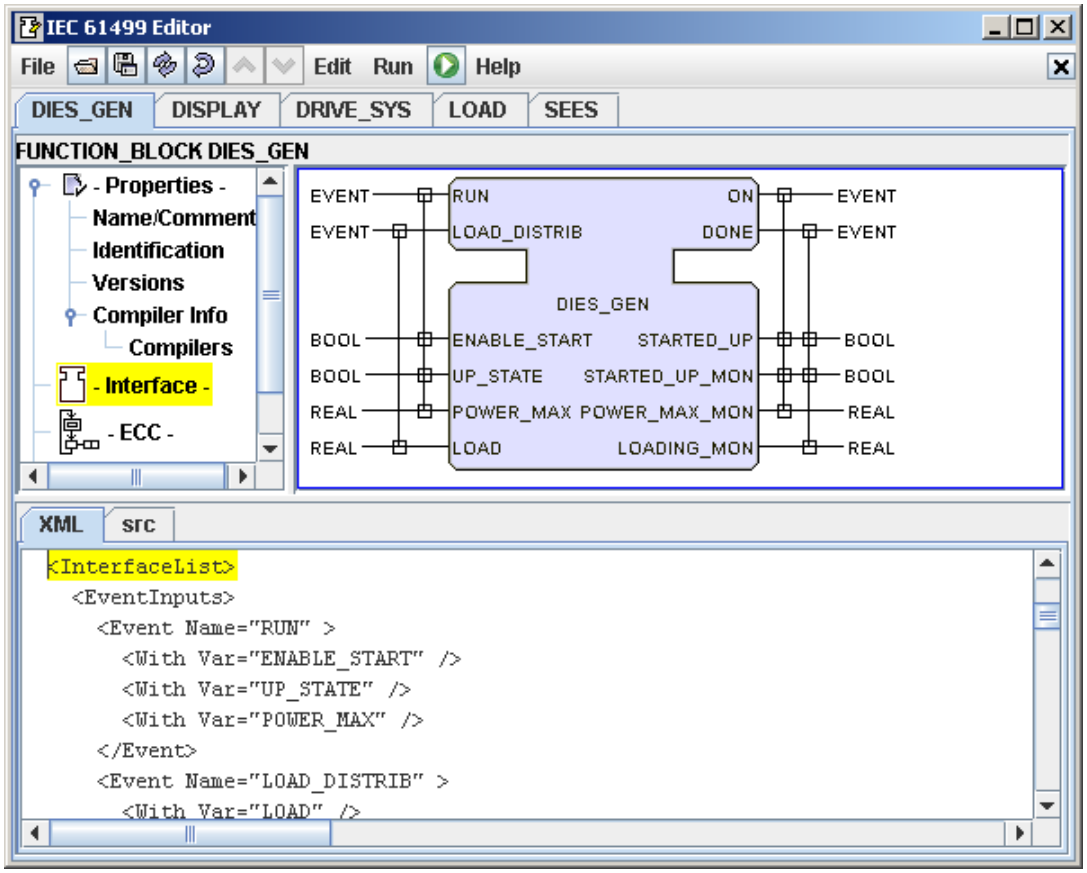


Рис. 4.12. Інтерфейс програми FBDK

Вхідними сигналами блоку є сигнал запуску (RUN), розподілу потужності (LOAD_DISTRUB), вихідними – сигнал готовності (DONE). Також для блоку дизель-генератора задаються такі параметри, як потужність (POWER_MAX), навантаження (LOAD), сигнал дозволу запуску й сигнал (ENABLE_START) можливості запуску (UP_STATE). Останній використовується для моделювання відмов пуску ДГ внаслідок, наприклад, поломки. Вихідні параметри показують, запущений генератор чи ні (STARTED_UP), а також його завантаження. Ці сигнали надходять у блок індикації.

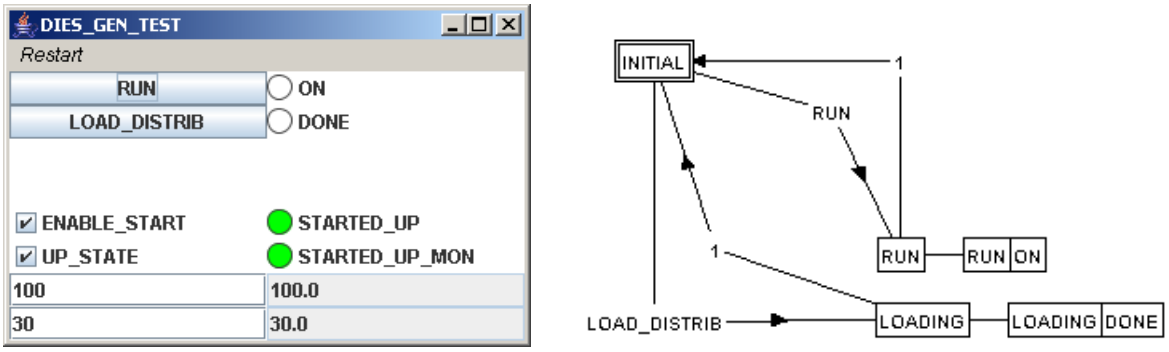


Рис. 4.13. Тестове вікно й алгоритм роботи ФБ DIES_GEN

Опис поводження блоку в програмі FBDK виконується мовою JAVA. На рис. 4.14 представлений вигляд вікна із програмою.

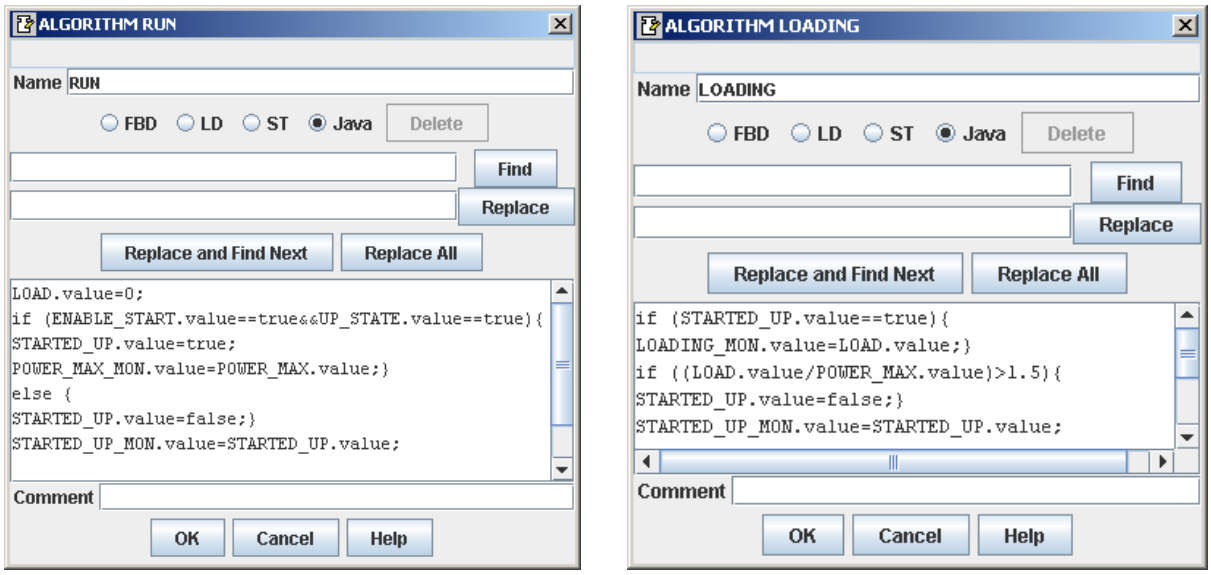


Рис. 4.14. Опис алгоритму пуску й розподілу потужності

На рис. 4.15 представлено блок індикації стану електроенергетичної системи й алгоритм його роботи. У цей блок надходять сигнали від кожного дизель-генератора й дозволяють визначити стан агрегату, його відсоток завантаження та інші параметри.

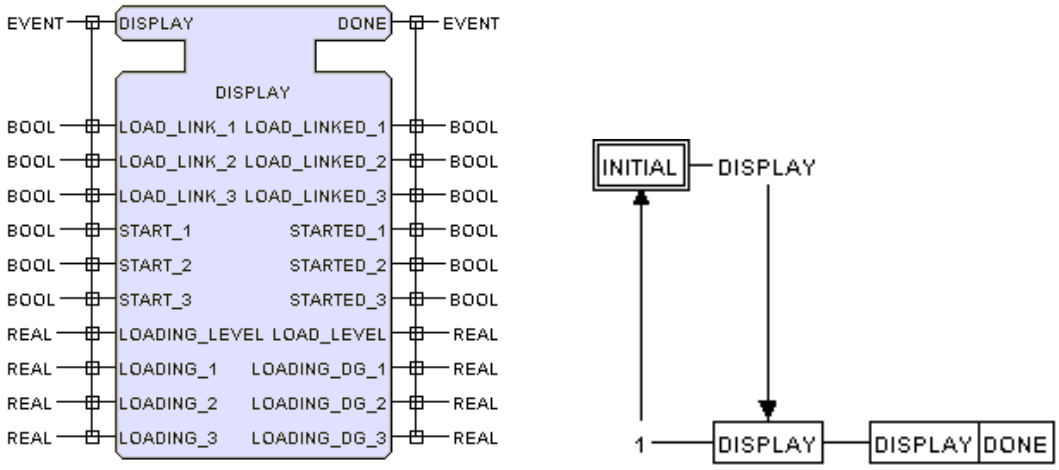


Рис. 4.15. Блок відображення даних про стан системи

На рис. 4.16 представлений вид функціонально блоку системи керування. Всі завдання по керуванню структурою енергетичної системи виконує система керування.

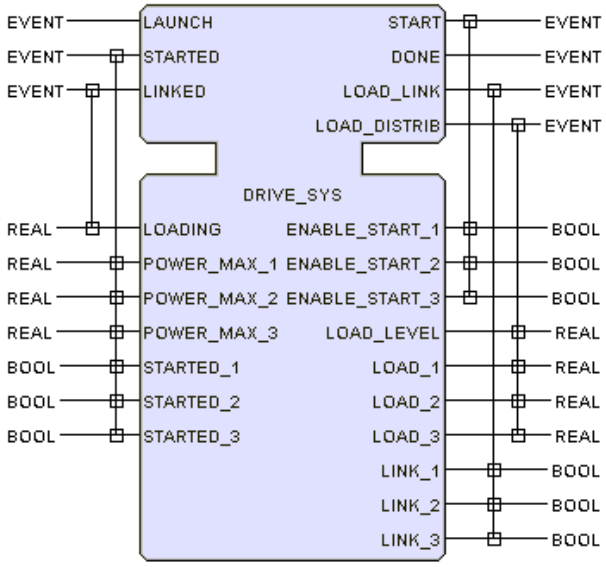


Рис. 4.16. Вигляд ФБ системи керування

На рис. 4.17 представлено алгоритм роботи функціонального блоку DRIVE_SYS.

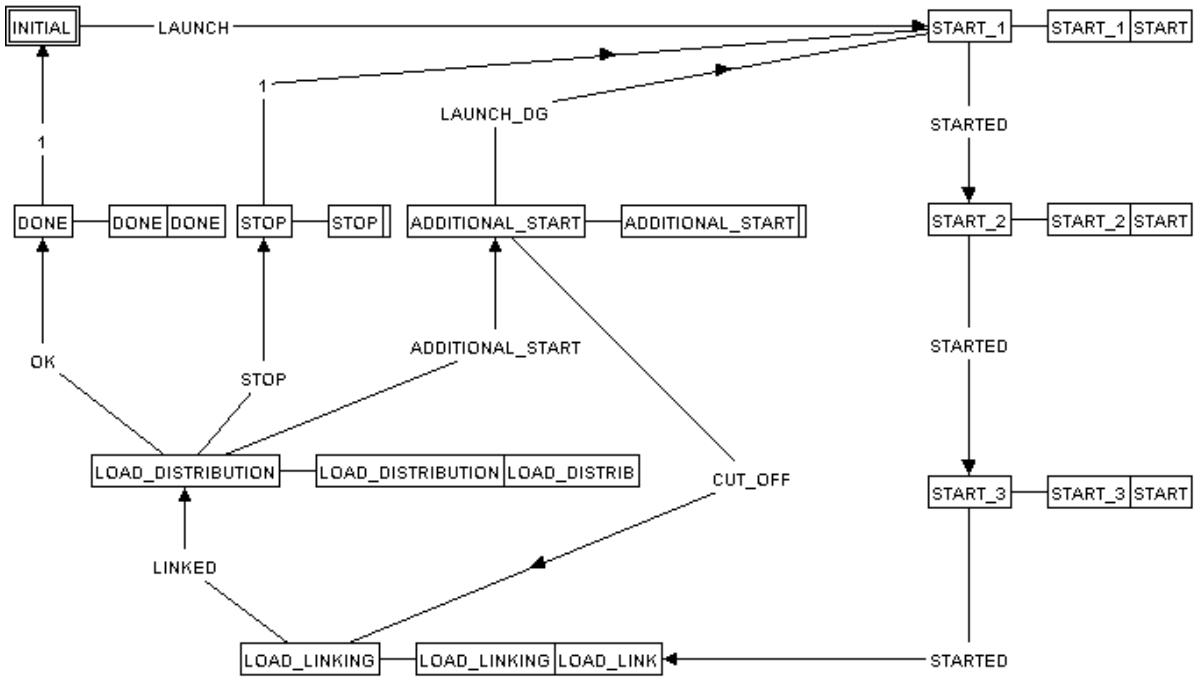


Рис. 4.18. Алгоритм роботи блоку системи керування

Тестове вікно представлено на рис. 4.19. У цьому вікні задаються сумарна потужність навантаження (у кВт), потужності кожного з дизель-генераторних агрегатів і сигнали дозволу запуску. Відображається величина завантаження кожного з дизель-генераторів. Моделювання виконується шляхом натискання на кнопки LAUNCH (пуск), STARTED (пуск ДГ), LINKED (підключення до шин). При цьому відбувається перехід з одного стану в інший відповідно до діаграми, наведеної на рис. 4.19.

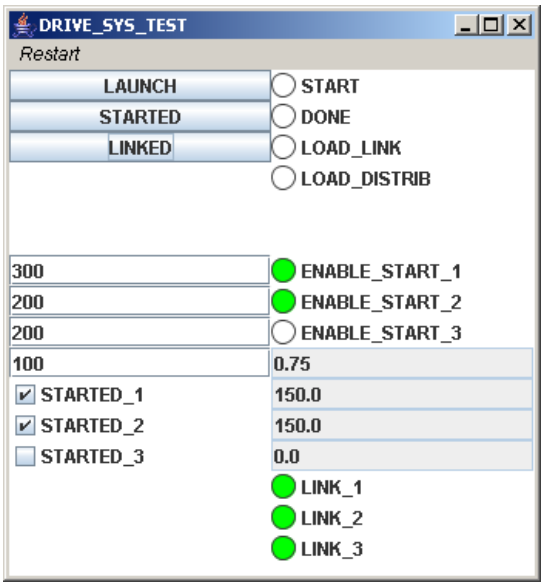


Рис. 4.19. Тестове вікно ФБ системи керування

На рис. 4.20 представлено функціональний блок навантаження й тестове вікно.

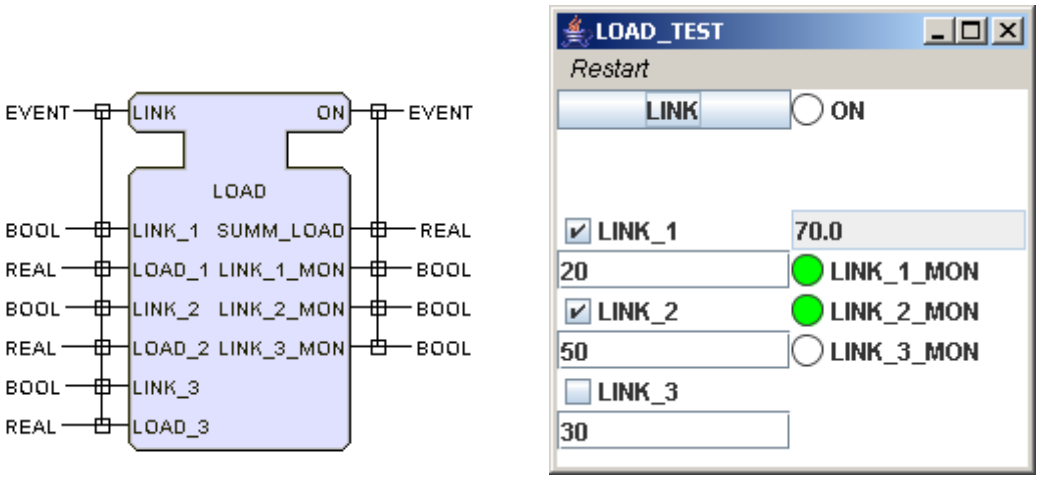


Рис. 4.20. ФБ навантаження й тестове вікно

За допомогою цього блоку задається навантаження на дизель-генератори. У блоці можна задавати сумарну потужність 3-х груп споживачів і управляти підключенням кожної із груп.

Після того, як всі блоки описані, можна створити модель електроенергетичної системи (рис. 4.21).

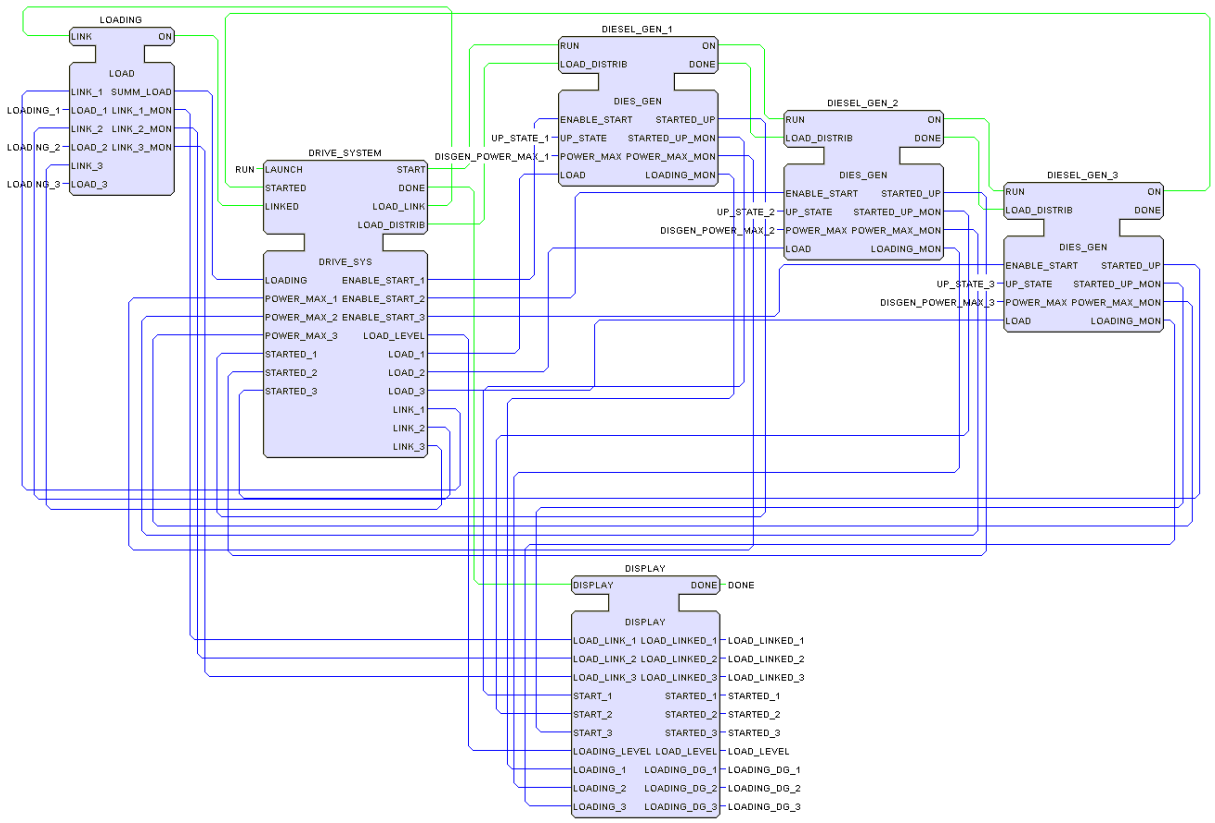


Рис. 4.21. Модель системи керування електроенергетичною системою

Існують програми, що дозволяють перетворювати створені моделі на UML-FB у моделі “класичного” UML (і навпаки). Для цього необхідно створити діаграму класів і діаграми станів для кожного класу, які описують алгоритм його роботи.

Основні переваги нового стандарту стосовно “класичного” – можливість одержання чисельних характеристик системи й наближеність ідеології до особливостей систем реального часу.

4.4. Вимоги до автоматизованих робочих місць оператора та програмного забезпечення

Система моніторингу представлена у вигляді програмного комплексу, який вирішує завдання збору й обробки даних, прийняття рішень і організації баз даних.

При розробці системи моніторингу, її графічного інтерфейсу використовувалися методи концептуального моделювання інформаційних процесів і інтелектуальних систем [16].

Концептуальне моделювання інформаційного забезпечення – це модель діяльності процесів, необхідна для надання знань про структуру досліджуваної області. Дана модель враховує процеси предметної області, вхідні й вихідні інформаційні об'єкти й сукупність керуючих елементів. Подібна організація інформаційних зв'язків сприяє більше наочному переходу від неформальних знань предметної області до їхнього формального подання.

Загальною концептуальною моделлю графічного інтерфейсу можна назвати наступний ряд процесів і об'єктів:

$$S_{ГИ} = \langle P, D, G, H_p, H_d, H_g, S_p, S_d, S_g, InD, InG, OutD, OutG \rangle,$$

де $P = \{p_i\}$ – множина процесів побудови зображень і обробки інформації;

$D = \{d_i\}$ – множина інформаційних об'єктів;

$G = \{g_i\}$ – безліч графічних об'єктів;

H_p, H_d, H_g – установлюють відповідність між окремими об'єктами моделі й безліччю підлеглих їм підоб’єктів;

S_p, S_d, S_g – проходження процесів, які задають частковий порядок виконання процесів;

InD – вхідні інформаційні об'єкти процесу;

InG – вхідні графічні об'єкти процесу;

$OutD$ – вихідні інформаційні об'єкти процесу;

$OutG$ – вихідні графічні об'єкти процесу.

До безлічі процесів побудови й обробки інформації $P = \{p_i\}$ можна віднести функції високого рівня, що формують крапкові зображення (растрові подання об'єктів, видимі контури об'єктів і ін.); апроксимуючі до

кусково-лінійних функцій (графічні подання об'єктів, індикатори та ін.); функції обробки вхідних даних; функції формування керуючих пакетів; функції розподілу процесів на потоки, що виконуються та інші.

До множини інформаційних об'єктів $D=\{d_i\}$ можна віднести пакети даних, що надходять; змінні і їх структури, асоційовані з елементами керування; проміжні стеки даних, формовані репорти передачі даних та інші.

Множина графічних об'єктів $G=\{g_i\}$ представлено у вигляді ряду індикаторів (графічні, стрілочні, шкальні, чисельні й ін.), керуючих об'єктів (елементи введення/виводу, індикації станів, відображення строкових змінних і ін.), видимої області системи моніторингу.

Однією з головних складових графічної моделі є концепція графічних атрибутів. Набори атрибутів описують основні властивості графічних об'єктів, такі як стиль лінії, її колір, розміри контурів і колір фону. Модель графічних атрибутів може бути описана наступним рядом:

$$A_g=\langle C, F_c, N_c, T_c, n_c, t_c, InC, OutC\rangle,$$

де $C=\{c_i\}$ – множина графічних атрибутів;

F_c – графічне зображення;

N_c, n_c – множина імен графічних атрибутів;

T_c, t_c – множина імен типів графічних атрибутів;

InC – процес, що запитує значення атрибута;

$OutC$ – процес ініціалізації сукупності запитаних атрибутів графічного об'єкта. Подібним чином складається модель графічних ресурсів:

$$A_r=\langle R, F_r, N_r, T_r, n_r, t_r, InR, OutR, DelR\rangle,$$

де $R=\{r_i\}$ – множина графічних ресурсів;

$DelR$ – процедура відображення графічного ресурсу.

При проектуванні концептуальної моделі використовувались визначені типи об'єктів, до яких відносяться графічні примітиви, ресурси й атрибути. У

графічних бібліотеках кожному із примітивів зіставлена певна кількість атрибутів.

Для спрощення аналізу концептуальної моделі використовують шаблони предметної області. Вони представлені окремими фрагментами моделі і побудовані щодо обраних процесів, з огляду на взаємодії з іншими об'єктами.

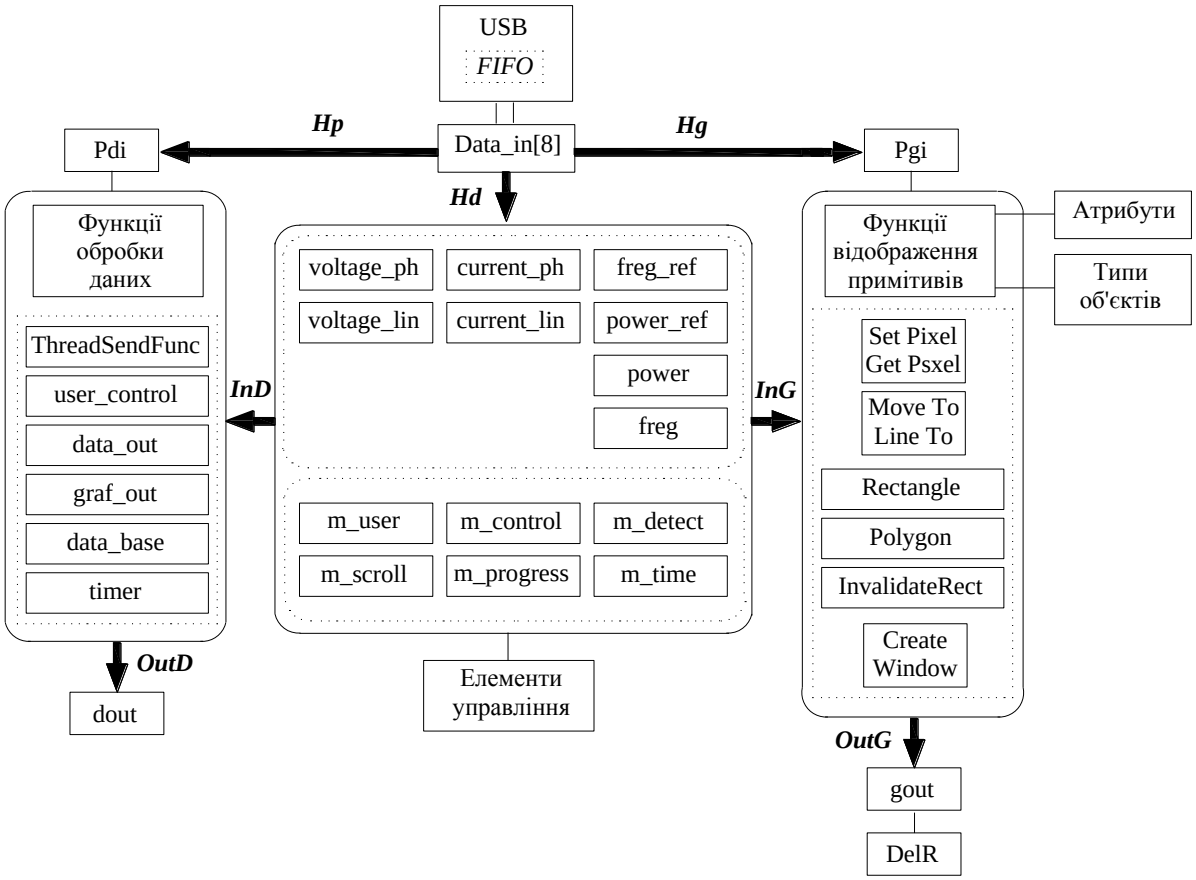


Рис. 4.22. Концептуальна модель системи моніторингу

Аналізуючи структуру зображень щодо кожного окремо взятого процесу можна сформулювати наступні залежності вихідних графічних об'єктів:

– Вихідне зображення для даного процесу є єдиним, типи ієрархій процесу й зображення збігаються;
– Вихідному зображенню відповідає хоча б один процес, для якого хоча б один компонент зображення є вхідним;

– Компоненти зображення, що є строго вхідним стосовно шаблону не можуть бути вихідними зображеннями підпроцесів;
– Будь-який підпроцес може мати не більше одного вихідного зображення;
– Вихідному даному процесу шаблона відповідає хоча б один компонент, що є вхідним для підпроцесу;
– Будь-які дані, взаємодіючі з підпроцесами, є або компонентами вхідних або вихідних даних процесу, або локальними даними, використовуваними для передачі інформації між підпроцесами;
– Компоненти даного, що є строго вхідним стосовно процесу не можуть виступати в якості вихідних даних підпроцесів.

Виходячи з вищевказаних залежностей концептуальна модель системи моніторингу має регулярну структуру зображень і регулярну структуру даних, тобто дана модель є регулярною.

Проходження в регулярній концептуальній моделі пов'язане з надмірністю моделі, що вносить одночасне використання відносин проходження процесів S_p і зображень S_g. Вивід зображень залежить від пріоритету проходжень, тому що ними побічно визначається порядок виконання процесів. При декомпозиції зображень відносини проходження успадковуються всіма компонентами зображень.

При виборі імен екземплярів і типів об'єктів у моделі повинні враховуватися наступні обмеження:

– Два підоб’єкта повинні мати різні імена, що не збігаються з іменем головного об'єкта;
– Об'єкти, що мають тотожну структуру підоб’єктів повинні мати схожі імена, що вказують їхню приналежність до загального процесу;
– Щоб запобігти однойменних і однотипних екземплярів об'єктів шляхом попарного перебору роблять синтаксичний контроль моделі.

Використання концептуальної моделі дозволяє аналізувати процеси обробки й виводу зображень, зменшити кількість операцій, що виконуються шляхом виключення тотожних функцій, від початку зіставити проходження процесів і організацію неструктурованих даних, розподілити ресурси й операції на окремі потоки, що виконуються знижуючи навантаження на обчислювальні засоби. Подібна організація програмного комплексу в значній мірі збільшує швидкодію й надійність системи моніторингу.

4.5. Методика створення програмного забезпечення для систем автоматизації електроенергетичних установок

Автоматизована система керування електроенергетичною установкою має багаторівневу структуру [98]. Нижній рівень – збір інформації з датчиків і видача керуючих впливів на виконавчі механізми. Середній – рівень первинної обробки й маршрутизації пакетів. Верхній – рівень прийому, обробки й відображення даних, включає сервер збору даних й автоматизоване робоче місце оператора.

До основних функцій системи відносяться:

1. Інформаційні функції:

– вимір і контроль напруги на шинах головного розподільного щита;

– контроль гранично припустимих значень частоти напруги;

– сигналізація роботи устаткування;

– виявлення, сигналізація та реєстрація відхилень параметрів від установлених границь;

– ручне введення даних;

– формування й видача оперативних даних;

– архівування параметрів технологічного процесу вироблення електроенергії.

2. Керуючі функції:

– пряме цифрове керування контурами (параметрами) технологічного процесу;

– реалізація протиаварійних захистів;

– видача керуючих впливів (здійснюється із мнемосхем на дисплеї).

3. Допоміжні функції:

– тестування комплексу технічних засобів;

– переналаштування системи (реконфигурація програмного забезпечення).

4. Організаційно-технологічні функції:

– розрахунок споживання й вироблення електроенергії за годину, добу, з метою реалізації можливості прогнозування завантаження електростанції й рішення завдань упереджуючого керування;

– облік часу пробігу (простою) устаткування;

– оцінка діяльності технологічного персоналу;

– розрахунок техніко-економічних показників.

На етапі створення програмного забезпечення (ПЗ) необхідно визначиться з набором типових блоків, з яких буде складатися електростанція. До таких блоків відносяться різні автомати керування режимами роботи електростанції та комутуються оператором або автоматично, залежно від режиму роботи АЕЕС, генератори, дизелі, конектори, блоки відображення стану автоматів з метою діагностики, блоки контролю параметрів елементів електростанції: частоти, напруги, параметрів потужності, cosφ.

Для кожного із блоків призначаються параметри, що характеризують його властивості:

– для генератора – потужність, напруга, оберти та ін.;

– для автомата – струм, час спрацьовування та ін.;

– для навантаження – потужність, тип навантаження та ін.

Із блоків створюється бібліотека. Оскільки програмне забезпечення виконує керування зовнішнім пристроєм, то кількість блоків повинна відповідати кількості керованих об'єктів. Зокрема, кількість автоматичних

вимикачів в автономній електростанції повинне відповідати кількості блоків у бібліотеці.

Основна ідея організації програмного забезпечення вводу-виводу складається в розбивці його на кілька рівнів, причому нижні рівні забезпечують екранування особливостей апаратури від верхніх, а ті, у свою чергу, забезпечують зручний інтерфейс для користувачів.

Кожний із блоків автоматики програмно зв'язується з певним реле, що управляється автоматично. Приклад структури даних, що посилають, показаний на рис. 4.23.

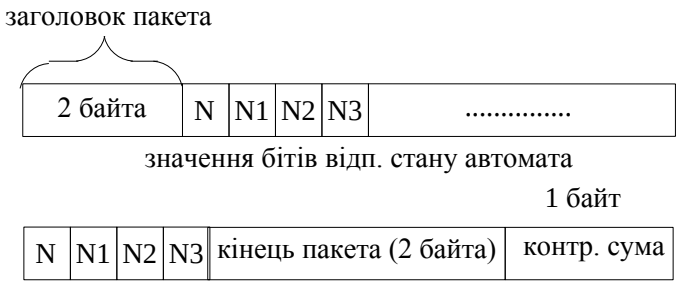


Рис. 4.23. Структура інформаційного пакета

Оскільки не виключені одиничні помилки для підвищення надійності поле зі станом автоматів дублюється. Нехай біти позначені b_{N-1}, b_{N-2}, ……..,b_3 , b_2, b_1, b_0, а в дублі C_{N-1}, C_{N-2},……., C_3, C_2, C_1, C_0. Тоді перевіряється умова:

$$e = \overline{b_n \oplus C_n}\ .$$

Якщо $e = 1$, це означає, що при передачі даних відбулася помилка й керуючий пристрій ігнорує отриманий пакет. При цьому відправляється зворотний пакет з кодом помилки й запитом повторної передачі.

Контрольна сума LRC обчислюється шляхом додавання всіх байт у пакеті без обліку переповнення:

$$LRC = \left[\sum_{i=0}^{N-1} B_i \right].$$

Необхідно вирішити проблему створення топології електростанції, що у програмі створює користувач, при цьому спосіб опису топології повинен

задовольняти вимогам зручності формалізації, збережена і зчитування в/із файлу, перевірки на наявність топологічних помилок (несумісні блоки не повинні бути з'єднані). Для рішення цієї проблеми кожному блоку призначається унікальний номер *Id*. Тоді вся бібліотека може бути описана набором

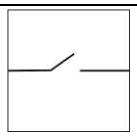
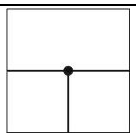
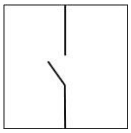
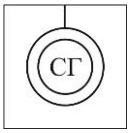
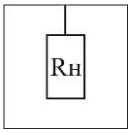
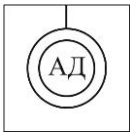
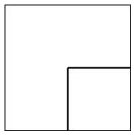
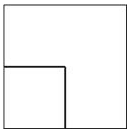
$$Id = \{Id_0, Id_1 \ldots Id_M\},$$

де M – кількість типів блоків, що використовуються в електростанції.

У табл. 4.1 представлений приклад бібліотеки блоків, що містить у собі автомати, *L*- і *T*-конектори, синхронний генератор, асинхронний двигун, активне навантаження.

Таблиця 4.1

Бібліотека типових блоків

Блок	Id	Блок	Id
	Id_0		Id_1
	Id_2		Id_3
	Id_4		Id_5
	Id_6		Id_7

З типових блоків на робочому полі програми розроблювачем складається схема електростанції. Оскільки кожен блок має свій унікальний ідентифікатор, те вся топологія може бути описана за допомогою матриці:

$$T = \begin{bmatrix} Id_5 & Id_3 & Id_2 & \dots & \dots & Id_9 \\ Id_7 & Id_8 & \dots & \dots & \dots & \dots \\ : & \dots & \dots & \dots & \dots & \dots \\ : & \dots & \dots & \dots & \dots & \dots \\ Id_2 & \dots & \dots & \dots & \dots & Id_4 \end{bmatrix}.$$

Розмірність матриці може бути фіксованою, а може бути динамічною. На початковому етапі розробки пропонується робити матрицю фіксованої розміром 10×10.

Знаючи значення елементів матриці, на діалоговому вікні програми в заданому порядку розташовуються блоки з бібліотеки, які мають ідентифікатори такі ж, як елементи матриці. Таким чином, матриця однозначно відповідає схемі автономної електростанції. Використання матричного подання доцільно, оскільки дозволяє використовувати масиви при реалізації програмного коду. Крім того, математичний апарат добре відпрацьований і з використанням матриць зручно формалізувати топології, вводити додаткові перевірки на наявність помилок.

Друга проблема складається в створенні зв'язків між блоками. Рішення цвєї задачі дозволить виконати перевірку на коректність схеми електростанції та попередити виникнення помилок. У цілому, можна обійтися і без цієї матриці, і виконати перевірку безпосередньо по топологічній матриці. З табл. 4.1 видно, що блоки Id_0 й Id_2 не можуть бути розташовані по вертикалі поруч, а блоки Id_1 й Id_3 не можуть бути розташовані по горизонталі поруч. На рис. 4.24 показаний приклад схеми, складеної з типових блоків.

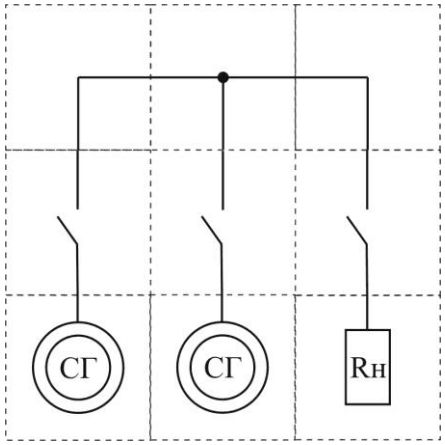


Рис. 4.24. Схема, побудована із блоків

Схема може бути описана матрицею 3×3:

$$S_1 = \begin{bmatrix} 6 & 1 & 7 \\ 2 & 2 & 2 \\ 3 & 3 & 4 \end{bmatrix}.$$

Оскільки кожен блок має свій унікальний номер, елементи матриці представлять собою ці ідентифікатори.

Інший приклад схеми представлений на рис. 4.25. У цій схемі допущені помилки при розташуванні блоків на робочому полі.

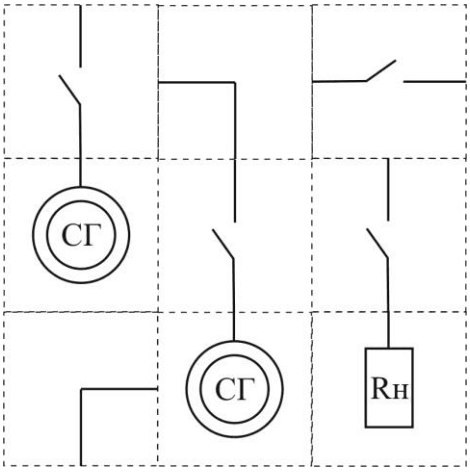


Рис. 4.25. У схемі розташування блоків некоректне

Матриця, що описує топологію наведеної схеми, має вигляд:

$$S_2 = \begin{bmatrix} 2 & 7 & 0 \\ 3 & 2 & 2 \\ 6 & 3 & 4 \end{bmatrix}.$$

Для першої матриці всі елементи розташовані коректно для другої матриці видно, що по вертикалі елементи 6 й 2 у першому стовпці розташовані неправильно. Можна ввести для цих блоків Id_{6nk} та Id_{2ij} умову:

$$k = j \ , \ n = i - 1.$$

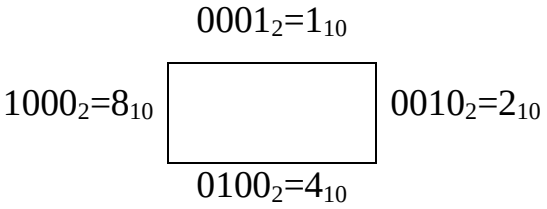
Тоді якщо $i=2$, і блок автомат розташований в комірці, то блок L-конектор відносно цього блоку може бути розташований у цій же колонці, але на попередньому рядку.

Аналогічно, для елементів Id_2 та Id_4. Умова коректного розташування на робочому полі блоків Id_{2mn} і Id_{4ij} має вигляд:

$$n=j,\ i=m+1.$$

На етапі створення схеми перевіряється умова для кожного блоку й забезпечується коректність схеми.

Оскільки кожен блок може мати вхідний і вихідний зв'язок з кожної зі сторін, для створення матриці зв'язків використовується наступний принцип: наявність або відсутність зв'язку позначається “1” або “0” відповідно. Тоді кожної зі сторін блоку призначається число, наприклад:



Для схеми на рис. 4.24 матриця зв'язків має вигляд:

$$C=\begin{bmatrix} 0110_2 & 1110_2 & 1100_2 \\ 0101_2 & 0101_2 & 0101_2 \\ 0001_2 & 0001_2 & 0001_2 \end{bmatrix}_2 = \begin{bmatrix} 7 & 14 & 12 \\ 5 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}_{10}$$

Для збереження у файл досить записати елементи обох матриць (топологічної матриці й матриці зв'язків), а потім у тій же послідовності зчитати. Знаючи значення елементів цих матриць, можна розташувати блоки на робочому полі та виконати їхнє з'єднання. У такий спосіб реалізується можливість збереження розробленої схеми.

Для зберігання координат блоків зручно використовувати матрицю координат. Якщо розміри блоків фіксовані, то досить зберігати тільки координати верхнього лівого кута для кожного блоку відносно робочого поля. Якщо ж розміри блоків різні, тоді додатково вказується ширина та висота блоку. Приклад матриці координат:

$$P_{xy} = \begin{bmatrix} P_{00} & P_{01} & P_{02} \\ P_{10} & P_{11} & P_{12} \\ P_{20} & P_{21} & P_{22} \end{bmatrix},$$

де

$$P_{ij} = [x, y, Id],$$

x – координата блоку по горизонталі;

y – координата блоку по вертикалі.

Поле Id введене для додаткового контролю й перевірки від помилок, у загальному випадку може бути відсутнє, оскільки розміри топологічної матриці й матриці координат будуть однаковими. Якщо розміри блоків різні то елемент матриці координат представляє собою набір значень:

$$P_{ij} = [x, y, W, H, Id],$$

де W – ширина блоку,

H – висота блоку.

Використання такої структури матриці дозволяє контролювати розташування блоків на полі так, щоб вони не перетиналися. Для цього при з'єднанні елементів повинна перевірятися й виконуватися умова:

$$x_{i+1,j+1} > x_{i,j} + W,$$

$$Y_{i+1,j+1} > Y_{i,j} + H.$$

Таким чином, використання матриць дозволяє вирішувати завдання формалізації схем і виявляти помилки на етапі проектування.

Для реалізації основних вимог при побудові програмного забезпечення автоматизованих систем повинні бути дотримані принципи, які можна розділити на три групи: проектування, системотехнічні й організаційні. До принципів проектування ставляться принципи системності, комплексності,

ефективності, принципи нових завдань. Системотехнічними є принципи комплексного використання обчислювальної техніки й програмних засобів, типізації рішень, єдності інформаційної бази, організації безпосереднього спілкування користувача із системою, розвитку системи. Більшість завдань варто вирішувати в діалоговому режимі.

При розробці програмного забезпечення першим етапом є збір даних про об'єкт автоматизації й здійснюваних видів діяльності, оцінка якості функціонування об'єкта, виявлення проблем, рішення яких можливо засобами автоматизації.

Завершальним етапом при розробці програмного забезпечення є аналіз функціонування системи, виявлення відхилень фактичних експлуатаційних характеристик ПЗ від проектних значень, установлення причин цих відхилень, усунення виявлених недоліків і забезпечення стабільності експлуатаційних характеристик ПЗ, внесення необхідних змін у документацію на ПЗ.

Правильний вибір методу, засобів і технології проектування створює умови для зниження витрат і скорочення строків проектування. До числа основних методів, які доцільно використати при проектуванні організаційних завдань, варто віднести метод пошуку аналога, метод синтезу на базі модулів і типових організаційних рішень, метод моделювання.

4.6. Статична модель інтерфейсу користувача

На першому етапі створення програмного забезпечення необхідно визначиться з набором типових блоків, з яких буде складатися електростанція. До таких блоків належать різні автомати керування режимами роботи електростанції, що комутуються оператором або автоматично, залежно від режиму роботи автономної електроенергетичної системи, генератори, дизелі, коннектори, блоки відображення стану автоматів з метою діагностики, блоки контролю параметрів елементів електростанції: частоти, напруги, параметрів потужності, cosφ.

Для кожного із блоків призначаються параметри, що характеризують його властивості:

– для генератора – потужність, напруга, оберти й ін.;

– для автомата – струм, час спрацьовування й ін.;

– для навантаження – потужність, тип навантаження та ін.

Із блоків створюється бібліотека. Оскільки програмне забезпечення виконує керування зовнішнім пристроєм, то кількість блоків повинне відповідати кількості керованих об'єктів. Зокрема, кількість автоматичних вимикачів в автономній електростанції повинно відповідати кількості блоків у бібліотеці. Процес створення зв'язку показаний на рис. 4.26.

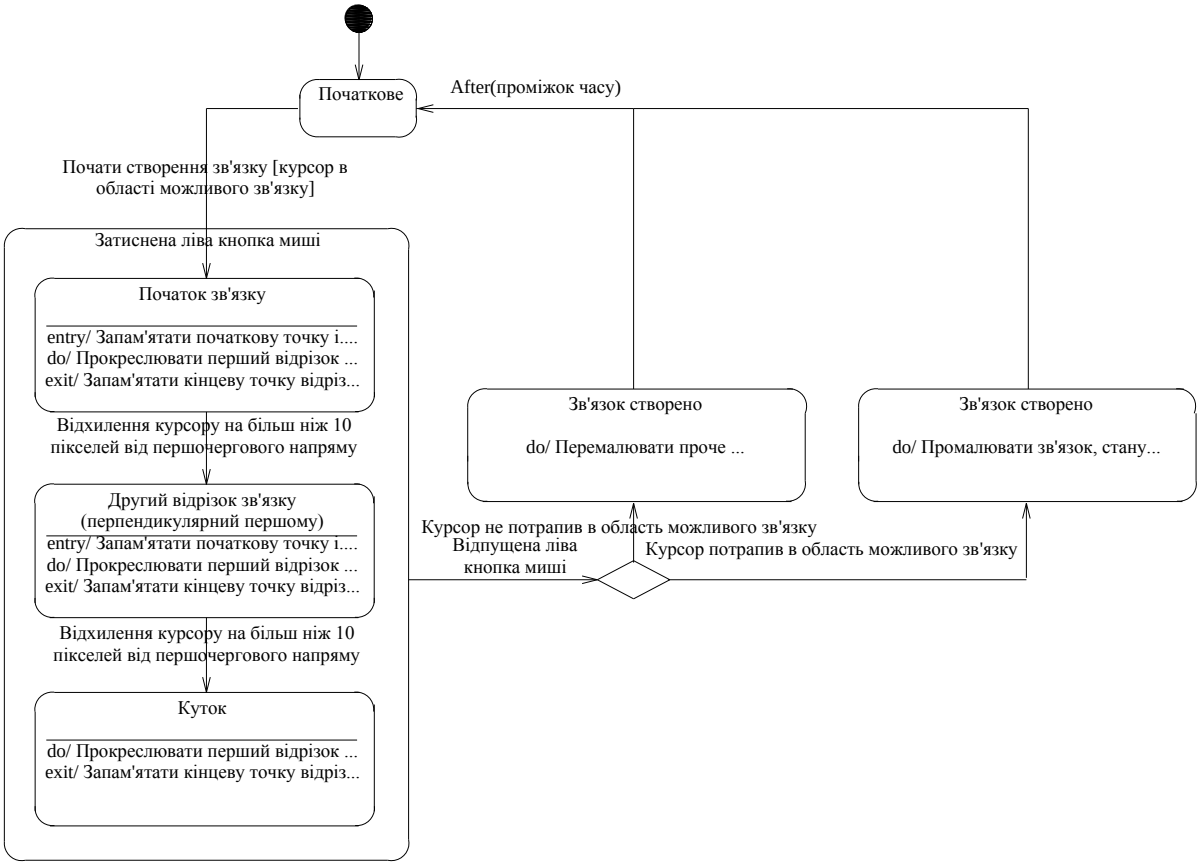


Рис. 4.26. Процес створення зв'язку між об'єктами

Користувач взаємодіє із системою за допомогою клавіатури та миші. Користувач ініціює наступні прецеденти (рис. 4.26):

1. *Створити Блок.*

Передумова. Даний блок є в бібліотеці.

Опис. Під створенням блоку розуміється перетаскування на робоче поле кожного з доступних у бібліотеці блоків (технологія Drag and Drop).

Альтернатива. При відсутності блоку в бібліотеці вивести повідомлення.

Постумова. Новий блок створений і виділений.

2. *Видалити Блок.*

Передумова. Блок виділений (натисканням лівої кнопки миші).

Опис. Видалити блок можна натисканням кнопки Delete на клавіатурі або викликом відповідного пункту контекстного меню блоку. Якщо блок був пов'язаний з іншими елементами, зв'язки також знищуються.

Постумова. Об'єкт зник з робочого поля, а кількість екземплярів даного елемента в бібліотеці збільшилося на 1.

3. *Перемістити Блок.*

Передумова. Блок виділений.

Опис. Переміщення блоку здійснюється слідом за курсором при затиснутій лівій кнопці миші або при натисканні кнопок “↑”, “↓”, “→”, “←” на клавіатурі. Якщо блок має зв'язок, він повинен переміщатися разом з ним.

Альтернатива. При досягненні блоком границь робочого поля він повинен зупинятися.

Постумова. Блок змінив положення й виділений.

4. *Створити Зв'язок між Блоками.*

5. *Видалити Зв'язок між Блоками.*

Передумова. Зв'язок виділений нитисканням на одному з її відрізків.

Опис. Видалення зв'язку відбувається при натисканні кнопки Delete на клавіатурі або виборі відповідного пункту контекстного меню.

Постумова. Зв'язок зник з робочого поля.

6. *Перемістити Лінію Зв'язку.*

Передумова. Лінія зв'язку виділена натисканням на одному з її відрізків.

Опис. При переміщенні курсору при затиснутій лівій кнопці миші лінія зв'язку пересувається слідом за курсором.

Альтернатива. При вилученні курсору за межі робочого поля зв'язок переміщається до краю робочого поля.

Постумова. Лінія зв'язку змінила положення й виділена.

7. *Копіювати Об'єкт або Лінію Зв'язку.*

Передумова. Блок (лінія зв'язку) виділений.

Опис.

1. Користувач натискає комбінацію “Ctrl+C” або вибирає пункт контекстного меню “Копіювати”.

2. Користувач натискає комбінацію “Ctrl+V” або вибирає пункт контекстного меню “Вставити”.

Альтернатива. Якщо екземпляри блоку закінчилися в бібліотеці, операція не виконується й виводиться повідомлення, що повідомляє.

Постумова. Створено й виділений блок (лінія зв'язку); лінія зв'язку промальована червоними кольорами (якщо копіювалася лінія зв'язку).

8. *Створити Вузол.*

Передумова. На робочому полі є хоча б одне перетинання ліній зв'язку.

Опис. Вузол можна створити при щічці правою кнопкою миші на перетинанні ліній зв'язку й виборі відповідного пункту контекстного меню.

Альтернатива. Якщо в цьому місці вузол уже створений, операція не виконується.

Постумова. Створено вузол на перетинанні ліній зв'язку.

9. *Повернути блок.*

Передумова. Блок виділений.

Опис. Повернути блок на 90^0 можна при натисканні “Ctrl+R” або виборі відповідного пункту контекстного меню блоку. Якщо блок має лінії зв'язку, вони повертаються слідом за блоком.

Постумова. Блок повернутий на 90^0 і виділений.

Сутність структурного підходу до розробки ПС полягає в декомпозиції (розбивці) системи на функції, що автоматизуються, які у свою чергу діляться на підфункції, на завдання й так далі. Процес декомпозиції триває

аж до визначення конкретних процедур. При цьому система, що автоматизується зберігає цілісне уявлення, у якому всі складові компоненти взаємозалежні.

В основі структурного методу лежать такі загальні принципи:

– розбивка системи на безліч незалежних задач, доступних для розуміння й рішення;

– ієрархічне упорядкування, тобто організація складових частин проблеми в деревоподібні структури з додаванням нових деталей на кожному рівні.

Кожен додаток, написаний для Windows, має одне головне вікно, що складається із двох компонентів – фрейму та клієнтської (робочої) області. Воно може бути як стандартним вікном Windows (наприклад, при побудові SDI- і MDI-додатків), так і звичайним блоком діалогу. У кожному разі всередині головного вікна додаток надає користувачеві всю необхідну для роботи інформацію, що представляє собою не тільки дані (блоки й зв'язки між ними), але й елементи інтерфейсу – меню, смуги прокручування, панелі інструментів, рядок стану і т.д.

Додатки, що базуються на блоках діалогу, не мають ніякого меню, крім системного, не можуть відкривати/зберігати документи. SDI- і MDI-додатки дозволяють працювати з файлами, причому в SDI-додатку можна в кожен момент часу мати відкритим тільки один документ, а в MDI – декілька. Оскільки розроблювальне ПЗ повинне мати можливість зберігати і відновлювати топологію схеми, але при цьому в кожен момент часу може бути зібрана тільки одна схема (тому що вона повинна відповідати реальним об'єктам), то при виборі типу додатка рекомендується зупинитися на SDI-додатку.

В основі SDI-додатка лежить архітектура “документ/подання”. Вона ґрунтується на трьох глобальних поняттях – фрейм, документ і подання. У самому загальному випадку, під документом розуміють ті дані, з якими працює додаток. Це може бути текст, картинка і т.п. Відображення цих даних

на екрані здійснюється у фреймі документа. Розроблювачі бібліотеки MFC створили для фреймів спеціальні класи вікон – подання, які відображають дані документа і управляють взаємодією користувача з ними.

Розглянута архітектура ґрунтується на створенні відповідних фреймів документів, що містять спеціальні вікна – подання. Крім того, фрейм здійснює координацію взаємодії між поданнями різних типів, направляючи їм команди й одержуючи від них повідомлення.

Архітектура “документ/подання” розділяє між собою документи і їхні зображення. Документ представлений у вигляді об'єкта, що забезпечує простір для зберігання даних у пам'яті й відповідає за такі операції, як запис і читання документа з диска. Очевидно, що більша частина таких операцій виконується самим додатком, а не класами бібліотеки MFC. Крім того, для кожного використовуваного подання також створюється окремий об'єкт, що являє собою екранне вікно, відповідальне за взаємодію з користувачем й об'єктом документа, печатка й т.д. Спільну роботу перерахованих об'єктів координує фрейм, що у більшості випадків є головним вікном додатка.

Коли запускається додаток під Windows, користувач взаємодіє з документами за допомогою їхніх зображень у фреймах. Фрейми документів мають два основних компоненти: властиво фрейм і його вміст. Для них бібліотека MFC використає два різних класи. Клас фрейму управляє безпосередньо фреймом, а клас подання – його вмістом. При цьому вікно подання є дочірнім стосовно фрейму, тобто розміщається в його робочій області. Сам фрейм документа є вікном SDI-додатка.

Таким чином, для каркаса розроблювального додатка знадобляться чотири класи: клас фрейму (для зв'язку документа з його поданнями), клас документа (для зберігання даних, що стосуються топології схеми) і два класи подання (один – для відображення бібліотеки блоків, другий – для відображення топології схеми). Кожен блок також можна представити у вигляді окремого класу. Проаналізувавши діаграму варіантів використання можна помітити, що існує певна сукупність методів (а виходить, і атрибутів),

однакових для всіх блоків. Тому можна створити абстрактний базовий клас, спадкоємцями якого й будуть класи блоків. Тоді на початковому етапі розробки ПЗ можна створити кілька класів блоків, відробити на них основні методи, а потім уже розширювати бібліотеку.

Використовуючи поняття мови UML можна визначити, що класи документа (у розроблювальному додатку це клас CDockWndDoc), базовий клас (CBase) і його спадкоємці (CGenerator – генератор CSwitch – ключ; їх можна об'єднати в пакет Бібліотека) є сутностями; клас фрейму (CFrameWnd) – керуючим класом і класи представлень (CTreeElements й CDockWndView; їх можна об'єднати в пакет Представлення) – класами представлення. Діаграми класів, у яких показані відносини між перерахованими вище класами, представлені на рис. 4.27 й рис. 4.28.

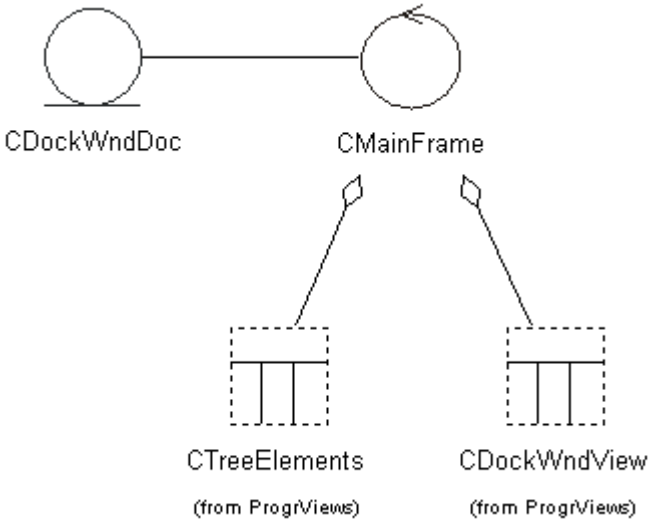


Рис. 4.27. Діаграма класів каркаса додатка

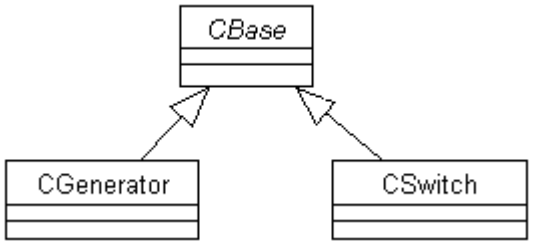


Рис. 4.28. Діаграма класів бібліотеки

Оскільки кількість блоків кожного типу повинне відповідати кількості керованих об'єктів, то програма повинна давати можливість створювати

строго фіксовану кількість екземплярів кожного класу бібліотеки. Для зручності роботи зі створеними блоками (а саме зчитування/запису їх із/в документ) бажано щоб екземпляри, що відносяться до одного класу, були якось зв'язані між собою. Тому пропонується для кожного типу блоків створити масив екземплярів відповідного класу, причому його розмірність повинна відповідати кількості фізичних об'єктів, що представляються цими блоками. Ці масиви (їх буде стільки, скільки типів блоків) можна зберігати в основному об'єкті представлення (CDockView). Тоді при створенні блоку буде створюватися не екземпляр класу, а тільки вікно, що відповідає йому (самі екземпляри класу будуть існувати протягом роботи додатка). (Можна, звичайно, використати динамічні масиви, і при створенні блоку просто додавати в кінець відповідного масиву новий екземпляр класу. Але тоді створення (а також видалення) блоку може зайняти багато часу, оскільки процес додавання нового елемента в масив складається з декількох етапів: резервування пам'яті під новий масив, копіювання елементів поточного масиву в новий, додавання в кінець нового масиву елемента, звільнення пам'яті.).

Для зберігання кількості блоків кожного типу будемо використати іменовані константи (наприклад, CountGen – для числа блоків “Генератор”, CountSwtchs – “Ключ” і т.д.), описані в окремому файлі (total.h). Також у цьому файлі можна зберігати загальне число типів блоків (TotalCount, також іменована константа) і визначення типу (використовується для того, щоб кожному блоку поставити у відповідність не індекс, а ім'я; наприклад, ChsdElmnt {Generator = 0, Switch, Nothing} – змінні типу ChsdElmnt).

Далі варто вирішити питання створення зв'язків. У бібліотеці MFC немає класу, що інкапсулює роботу з лініями, тому створимо власний клас CLine. Він буде зберігати координати початкової й кінцевої крапок відрізка; покажчики на з'єднані з початковою й кінцевою крапками відрізку; покажчики на блоки, які з'єднує відрізок (найчастіше один з них приймає нульове значення, тому що найчастіше лінія зв'язку складається з декількох

відрізків). Знаючи початкові й кінцеві крапки відрізків лінії зв'язку, можна за допомогою контексту пристрою, пов'язаного з вікном робочого поля, накреслити лінію зв'язку. Однак додаток повинен мати можливість не тільки створювати лінію зв'язку, а й працювати з нею – переміщати й видаляти. При переміщенні одного з відрізків лінії зв'язку повинні змінюватися й координати кінців пов'язаних з ним відрізків, тому виникає необхідність зберігання в об'єкті CLine покажчиків на з'єднані з його початковою і кінцевою крапками відрізки. Видалення зв'язку відбувається при виділенні одного з відрізків і наступному натисканні кнопки Delete, тому тут також потрібні покажчики на з'єднані з даним відрізком відрізки. Також при видаленні лінії зв'язку блоки, які вона з'єднувала, повинні сповіщатися про те, що вони вже не зв'язані. Тому клас CLine повинен зберігати покажчики на блоки, які даний відрізок з'єднує.

Оскільки натискання на лінії зв'язку класифікується як просто натискання на робочому полі, для розпізнання цієї події треба використати спеціальний алгоритм. Наприклад, знаючи, що кольори робочого поля й кольори лінії зв'язку не збігаються, можна одержати кольори пікселя (або сукупності пікселів, що перебувають поблизу положення курсору), на якому натиснув користувач, зрівняти його з кольорами робочого поля, і, якщо вони не збігаються, виходить, обрана лінія зв'язку. Однак при цьому неможливо ідентифікувати обрану лінію зв'язку, оскільки відомі тільки координати крапки, на якій клацнув користувач. Для рішення цієї проблеми пропонується наступний підхід. Можна створити клас бінарного дерева CTree й, оскільки всі відрізки лини зв'язку ортогональні, два його екземпляри – один для горизонтальних відрізків, другий – для вертикальних. При створенні відрізка лінії зв'язку його напрямок відомо, тому цей відрізок (точніше покажчик на нього) можна додати до відповідного бінарного дерева. Критерієм для знаходження положення при додаванні відрізка до дерева може служити його координата по осі *Oy* для горизонтальних відрізків (оскільки значення *y*-координат початкової й для кінцевої крапок відрізка

збігаються) і по осі *Ox* – для вертикальних. Якщо елемент із таким значенням постійної координати в дереві вже існує, пошук місця для його додавання ведеться по другій координаті початкової крапки лінії. При цьому пріоритет віддається лініям лежачої вище (для горизонтальних) або лівіше (для вертикальних). Далі при розпізнанні події “Користувач клацнув на відрізку” можна опитуванням кольори пікселів зверху, знизу, праворуч і ліворуч з'ясувати напрямок обраного відрізка. Потім можна звернутися до відповідного цього напрямку бінарному дереву, задавши для пошуку координату крапки, що залишається постійної для даного напрямку відрізка (для горизонтального – *y*-координату, вертикального – *x*). Далі, оскільки можуть існувати кілька відрізків, що належать різним лініям зв'язку й лежать на одній прямій, необхідно провести пошук в обраній галузі дерева по другій координаті крапки, на якій клацнув користувач. І тільки тепер можна точно ідентифікувати лінію зв'язку, що виділив користувач.

Альтернативою створенню бінарних дерев є використання динамічних масивів, однак, додавання, а самий головне – пошук елементів у бінарному дереві здійснюються набагато швидше. Тому в додатку будемо використати клас CTree.

Таким чином, у програму додалися 2 класи – CLine й CTree. Відношення між ними й вищерозглянутими класами показано на рис. 4.29.

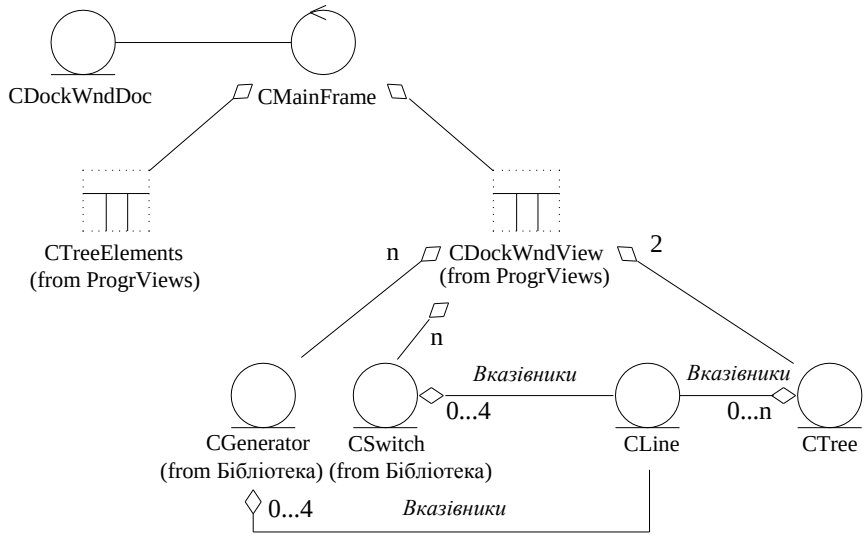


Рис. 4.29. Статична модель додатка

Клас робочого поля (CDockWnd) містить 2 дерева, елементами яких є покажчики на всі створені в ході роботи з додатком відрізки ліній зв'язку. Клас кожного із блоків містить покажчики на всі пов'язані із блоком відрізки (їх може бути від 0 до 4, оскільки кожен блок має всього 4 сторони). Крім дерев, до складу класу CDockWnd входять n_1 об'єктів класу CSwitch й n_2 – класи CGenerator.

Для класів, що представляють різні типи блоків, існує сукупність методів і атрибутів, виконуваних і використовуваних аналогічно, то був створений абстрактний клас CBase, спадкоємцями якого є класи CGenerator і CSwitch (це показано в статичній моделі додатка). Сам клас CBase є спадкоємцем класу CStatic бібліотеки MFC і містить атрибути й методи, показані на рис.3. Клас CGenerator є стандартним нащадком класу CBase і містить тільки його атрибути й методи. Об'єкти класу CSwitch (який також є нащадком класу CBase) має ще 2 додаткові стани: “ключ замкнутий” і “ключ розімкнений”.

На рисунку атрибути мають наступне призначення:

m_EnConnect – число, що дозволяє зв'язки (використовується для визначення чи має можливість кожна зі сторін блоку мати вхідний або вхідний зв'язок);

m_CanConnect – змінна, котра показує чи потрапив курсор в область можливого зв'язку;

m_Angle[2] – змінні, що кодують положення блоку;

m_PrevPoint – змінна для зберігання попередніх координат лівого верхнього кута блоку; використовується при обчисленні відстані, на яке перемістився блок;

m_CurPoint – змінна для зберігання поточних координат лівого верхнього кута блоку відносно робочого поля;

m_IsMoving – змінна для зберігання відомостей про переміщення об'єкта (0 – блок коштує на місці, 1 – переміщається);

m_pLines[4] – масив покажчиків на крайні відрізки зв'язку, з'єднані зі сторонами блоку.

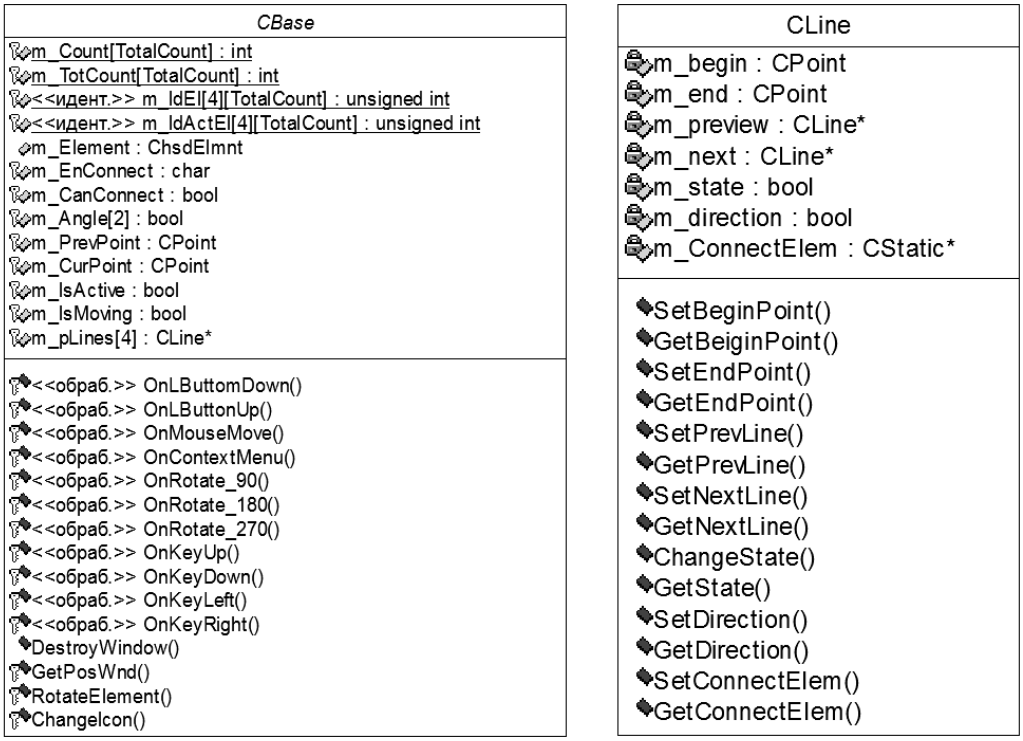


Рис. 4.30. Методи й атрибути класів

Інші атрибути пов'язані з тим, що для класів, що представляють різні типи блоків, немає необхідності в кожному екземплярі зберігати дані, що стосуються зображення блоків, а також кількості вже створених вікон. Ці дані повинні бути зв'язані безпосередньо із класом. Для цього можна використати змінні із кваліфікатором static. Однак такі змінні створюються тільки один раз при визначенні класу, і тому при спадкуванні класів вони не успадковуються, а є загальними для всіх класів-нащадків. Тому що з кожним класом-нащадком класу CBase повинні бути зв'язані свої іконки й своє число створених вікон, то пропонується в базовому класі створити масиви відповідних змінних (розмірність масиву буде дорівнює числу блоків у бібліотеці (іменована константа TotalCount); елементам масивів з однаковими індексами будуть відповідати дані щодо одного класу-нащадка). Всі методи, що мають стереотип <<оброб.>> є функціями-оброблювачами однойменних повідомлень, що посилають вікну блоку. Метод ChangeIcon відповідає за

зміну зображення блоку, RotateElement – за зміну змінних, що кодують положення блоку, і числа, яке дозволяє зв'язки.

Оскільки одним з базових принципів об’єктно-орієнтованого програмування є інкапсуляція, то всі атрибути даного класу є приватними змінними, а методи орієнтовані на роботу з ними об'єктами інших класів. Призначення більшості атрибутів докладно описано при описі класу CLine у статичній моделі.

4.7. Динамічна модель та аналіз повідомлень

На етапі динамічного моделювання розглядаються динамічні, або поведінкові аспекти системи. Динамічна модель є одночасно міжоб’єктною (що описує взаємодію об'єктів) і внутрішньооб'єктною (що характеризує, як залежний від стану об'єкт визначається кінцевим автоматом і зображується на діаграмі станів). Внутрішньооб’єктна сторона динамічного моделювання має відношення до кінцевих автоматів та діаграм станів.

Взаємодія між об'єктами описується за допомогою UML-діаграм кооперацій. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але й всі структурні відносини між об'єктами, що беруть участь у цій взаємодії. Повідомлення на діаграмі кооперації специфікує комунікацію між двома об'єктами, один із яких передає іншому деяку інформацію. При цьому перший об'єкт очікує, що після одержання повідомлення другим об'єктом піде виконання деякої дії. Таким чином, повідомлення є причиною або стимулом для початку виконання операцій, відправлення сигналів, створення й знищення окремих об'єктів. Зв'язок забезпечує канал для спрямованої передачі повідомлень між об'єктами від об'єкта-джерела до об'єкта-одержувача.

У динамічній моделі виконується аналіз повідомлень, якими обмінюються об'єкти при виконанні кожного з варіантів використання. Для цього використовуються діаграми кооперації для кожного із прецедентів:

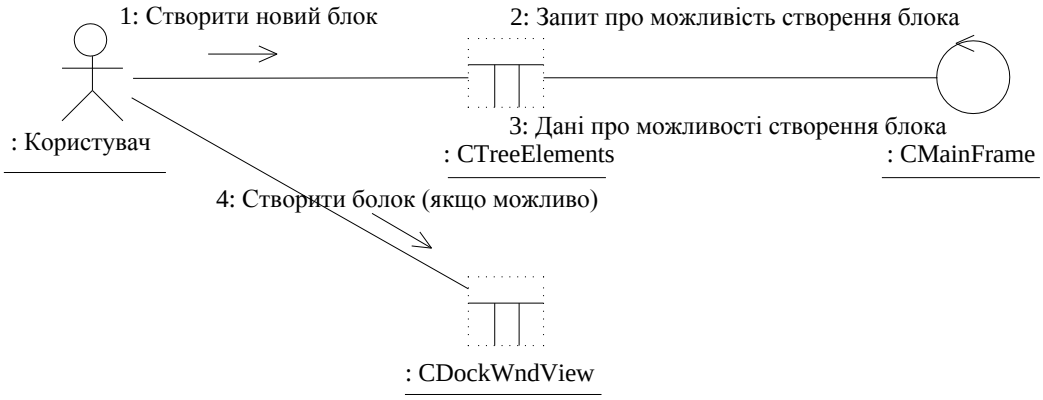


Рис. 4.31. Діаграма кооперації «Створити блок»

У процесі створення нового блоку беруть участь 3 об'єкти: при спробі створення користувач перетаскує іконку блоку з бібліотеки (об'єкт подання CTreeElements), далі об'єкт подання посилає повідомлення головному фрейму з питанням, чи є ще дані блоки в бібліотеці, і якщо головний фрейм “дозволив” створити блок при перетаскуванні іконки на робоче поле поданню CDockView посилає повідомлення “Створити блок”. У цьому повідомленні повинна втримуватися інформація щодо того, який саме блок треба створити. Зберігання цієї інформації покладає на об'єкт CMainFrame. Таким чином, одним з атрибутів цього класу буде індекс (або ідентифікатор) обраного для створення користувачем блоку.

В описі варіанта використання “Створити Блок” з'ясовано, що створення блоку відбувається в момент відпускання лівої кнопки миші на робочому полі, якщо до цього елемент був обраний у поданні бібліотеки. Тому функцію створення блоку (у програмі вона названа CreateNewElement) можна викликати у функції-оброблювачі повідомлення, що генерується при відпусканні лівої кнопки миші в робочому полі (OnLButtonUp). Аргументом функції CreateNewElement буде константа (типу ChsdElmnt), що ідентифікує обраний блок і зберігається в об'єкті CMainFrame.

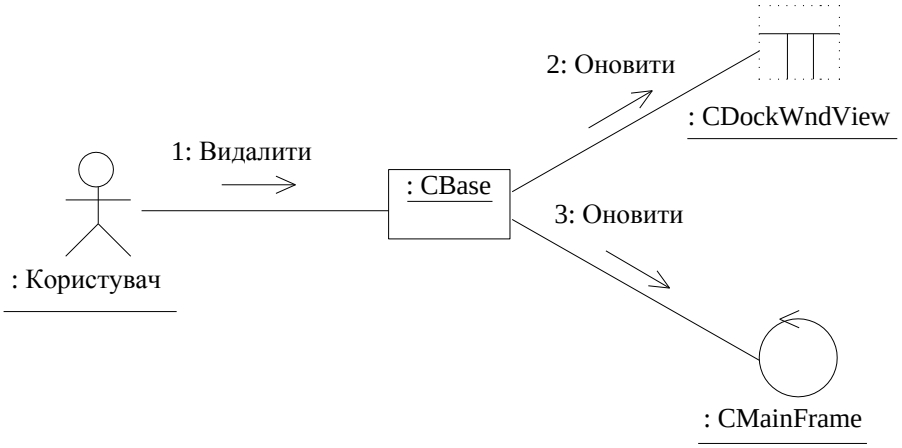


Рис. 4.32. Діаграма кооперації «Видалити блок»

При видаленні блоку відправляється відповідне повідомлення блоку, після чого блок посилає повідомлення поданню й головному фрейму із проханням їхнього відновлення (для об'єкта CDockView воно полягає у видаленні ліній зв'язку, якими був зв'язаний даний блок з іншими; для CMainFrame – у відновленні інформації про кількість створених на робочому полі на даний момент блоків).

Для видалення об'єкта можна перевизначити функцію DestroyWindow для блоку CBase таким чином, щоб при його виклику послідовно здійснювалося відновлення подання, потім даних щодо кількості блоків і безпосередньо сам виклик функції видалення вікна об'єкта (при цьому сам об'єкт не зникає).

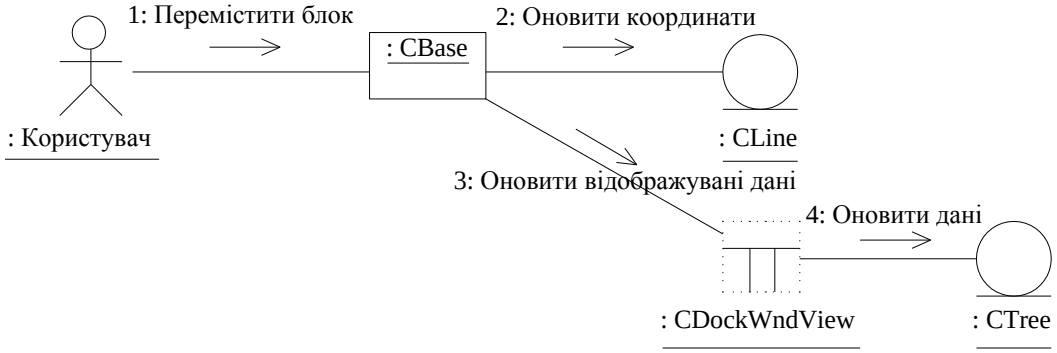


Рис. 4.33. Діаграма кооперації «Перемістити блок»

При переміщенні блоку посилає відповідне повідомлення екземпляру блоку. Коли переміщення закінчене, об'єкт блоку посилає головному поданню повідомлення “Обновити” (перемалювати блок і вихідні (вхідні) з (в) його лінії зв'язку).

Розглянемо докладніше процес переміщення. Він складається з декількох етапів: спочатку користувач натискає ліву кнопку миші при розміщенні курсору над блоком, потім при затиснутій лівій кнопці миші переміщає курсор й, нарешті, відпускає ліву кнопку миші. Таким чином, для реалізації даного варіанта використання треба розглядати функції-оброблювачі трьох подій, що посилають вікну блоку: натискання лівої кнопки миші (OnLButtonDown), переміщення курсору (OnMouseMove) і відпускання лівої кнопки миші (OnLButtonUp). Процес обчислення відносного переміщення блоку має свої особливості: оскільки повідомлення, генерируемые при зміні положення курсору, посилають екземпляру блоку, координати поточного положення курсору обчислюються щодо лівого верхнього кута блоку. Тому для того щоб визначити, куди переміщається об'єкт (вправо, уліво, нагору або вниз), необхідно щораз порівнювати координати поточного положення курсору з попереднім. Наприклад, якщо в один момент координати курсору були (16, 16), а в наступний – (19, 17), те блок треба перемістити на 3 пикселя вправо та на 1 ліворуч (вісь *Ox* спрямована ліворуч праворуч, *Oy* – зверху вниз).

При перемальовуванні блоку на головному поданні треба знати координати його вершин щодо вікна подання. Оскільки розміри всіх блоків фіксовані (32×32), те досить знати координати однієї з його вершин, наприклад, лівої верхньої. Для цього можна один раз при натисканні лівої кнопки миші обчислити ці координати (використовуючи метод класу CDockView ScreenToClient; аргументом цього методу є адреса на прямокутник, що містить координати дочірнього вікна, а результатом виклику – зміна цих координат щодо зухвалого вікна), а далі при переміщенні курсору змінювати їх відповідно до вищеописаного алгоритму.

При цьому поки затиснута ліва кнопка миші, курсор повинен бути захопленим вікном блоку (тобто приймати всі повідомлення від миші повинне тільки вікно блоку). Для візуального контролю переміщення можна використати нестандартний курсор, що з'являється при початку переміщення й зникає в момент відпускання лівої кнопки миші.

Таким чином, для виконання переміщення блоку клас CBase повинен мати 3 додаткових атрибути: координати попереднього положення лівої вершини (m_PrevPoint типу CPoint), що тече положення (m_CurPoint типу CPoint) і прапор, що показує переміщається чи об'єкт ні (m_IsMoving типу bool). Сам процес переміщення можна описати наступним псевдокодом:

```
   при натисканні лівої кнопки миші
   виділити об'єкт
   зберегти координати лівої верхньої вершини об'єкта в змінної m_PrevPoint
   дозволити переміщення (установити значення m_IsMoving в “істина”)
   при переміщенні миші
   захопити курсор і змінити його вид
   обчислити різниці між однойменними координатами крапки поточного положення курсору й крапки, що зберігає попередні координати (m_PrevPoint)
   додати отримані значення до відповідних координат крапки, що зберігає координати поточного положення блоку (m_CurPoint)
   зберегти поточні координати курсору в змінної m_PrevPoint
   при відпусканні лівої кнопки миші
   відпустити курсор і змінити його вид на стандартний
   якщо в даного блоку є лінії зв'язку
     змінити їхньої координати
     перемалювати лінії зв'язку
   перемалювати вікно відповідно до нових координат (значення координат лівого верхнього кута зберігається в змінної m_CurPoint, правого нижнього - кожна на 32 більше)
```

заборонити переміщення (установити значення m_IsMoving в “неправду”)

При аналізі цього псевдокоду може здатися, що змінна m_IsMoving не потрібна. Насправді вона використовується для відділення процесу переміщення від інших процесів (виділення об'єкта, створення зв'язку).

4.8. Розробка програмних засобів керування електростанцією

На рис. 4.34 представлено інтерфейс розробленої програми керування автономною електростанцією. Програмне забезпечення системи керування автономною електростанцією дозволяє автоматизувати процес генерування електроенергії, контролювати режим роботи електростанції, попереджати виникнення аварійних ситуацій, управляти напругою генераторів і частотою обертів дизелів, виконувати синхронізацію генераторів.

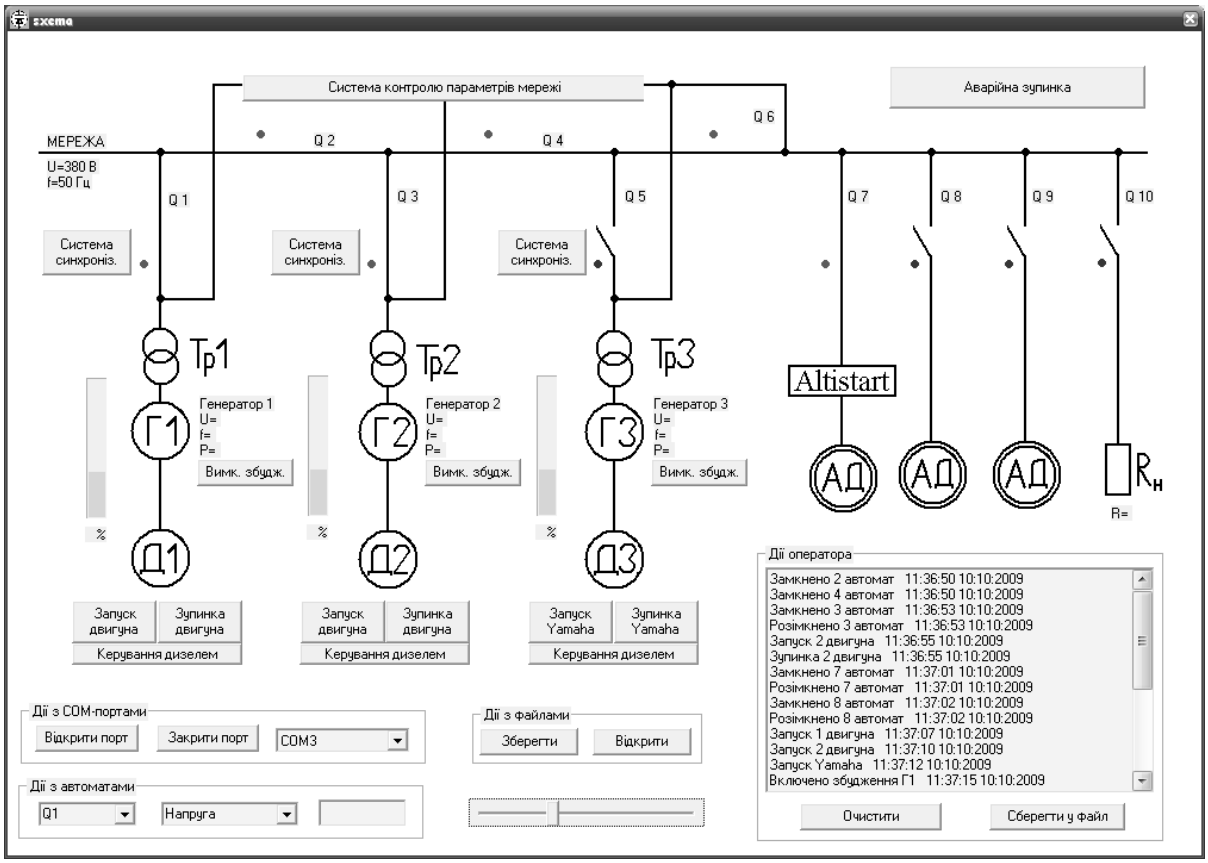


Рис. 4.34. Інтерфейс програми керування електростанцією

Програмне забезпечення мікропроцесорної системи контролю параметрів генератора дозволяє спостерігати за осцилограмами напруги та струму генератора, контролювати активну, реактивну та повну потужність, cosφ. Отримані дані можливо зберегти у файл для подальшого аналізу, а також на основі отриманих даних вирішувати наукові задачі ідентифікації параметрів генераторного агрегату, оцінювати якість роботи систем керування, виконувати спектральний аналіз напруги та струму. На рис. 4.35 представлено головне вікно програми.

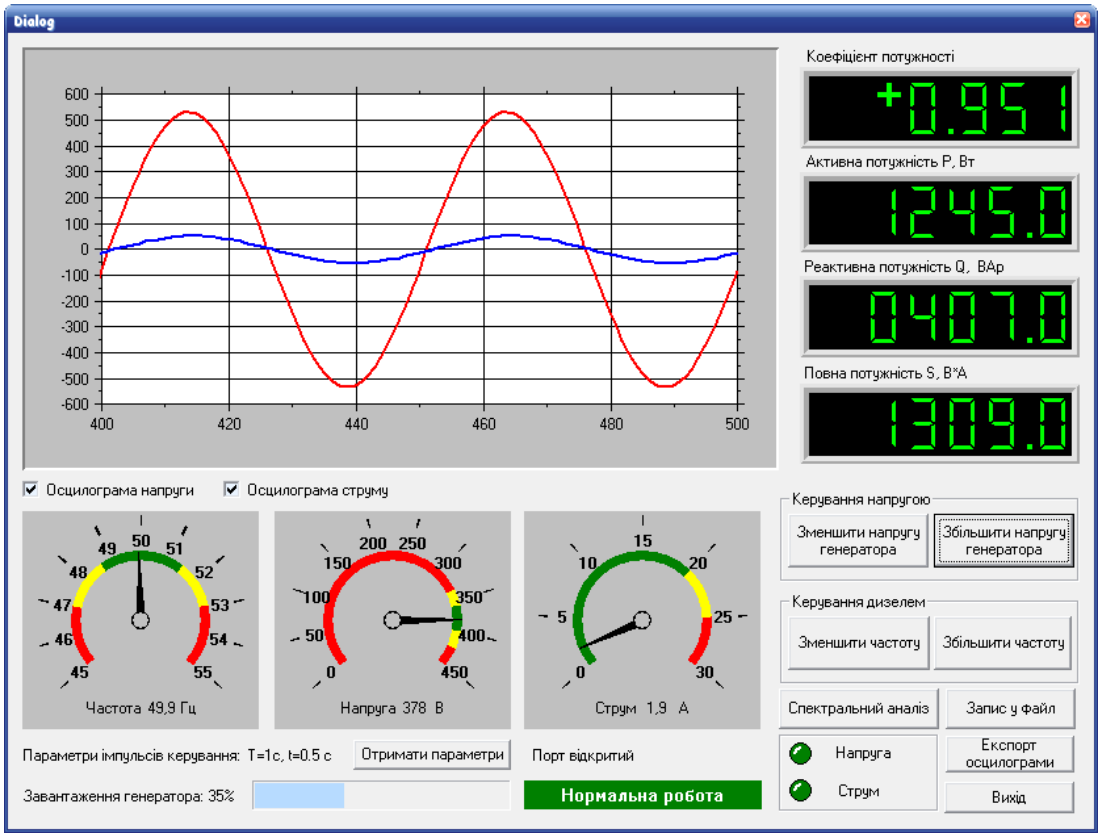


Рис. 4.35. Інтерфейс програми

Програмно-апаратна система поєднує в собі всі необхідні функціональні вузли і програмні компоненти для ефективної роботи оператора. Програми з розвиненими засобами візуального інтерфейсу не потребують від користувача тривалого навчання. А працюючи в комплексі з іншими додатками здатні вирішувати найширше коло науково-технічних і прикладних завдань. Програмне забезпечення керування автономною електростанцією дозволяє автоматизувати процес генерування

електроенергії, контролювати режим роботи електростанції, попереджати виникнення аварійних ситуацій, керувати напругою генераторів та частоту обертів приводних двигунів, виконувати синхронізацію генераторів.

РОЗДІЛ 5
МЕТОД СИНТЕЗУ МАТЕМАТИЧНИХ МОДЕЛЕЙ ЛОГІКО–ДИНАМІЧНИХ ПРОЦЕСІВ КОНТРОЛЮ Й КЕРУВАННЯ

5.1. Аналіз стану інформаційних технологій при синтезі математичних моделей логіко–динамічних процесів контролю й керування

Оскільки «інформаційні технології» є матеріалізованими носіями різної інформації, а сам носій, власне кажучи, являє собою формалізований запис цієї інформації. Тому одним з основних завдань «інформаційних технологій», у тому числі й в електроенергетиці, є розробка нових методів формалізованого запису різних логіко–динамічних процесів перетворення аналого–цифрових і аналогових сигналів у системах керування й контролю. І цей формалізований запис будь–яких процесів перетворення сигналів повинен бути у вигляді аналітичних символів, які у своїй послідовності повинні формувати функціонально закінчену математичну модель, а вона у своєму записі дозволяла б зробити доступним свій інформаційний зміст. При цьому основною якістю функціонально закінченої математичної моделі логіко–динамічного процесу перетворення аналогових сигналів не тільки в електроенергетиці, але й в обчислювальній техніці повинна бути мінімізація словесного опису її смислового змісту. І таке завдання може бути вирішено, оскільки в науці є такі математичні моделі, які або мають мінімізований коментар, як в органічній хімії, або виключають всі словесні описи, як у музичних нотах.

У цей час у якості узагальненої функціональної структури для математичної моделі будь–якого логіко–динамічного процесу перетворення аналого–цифрових і аналогових сигналів також повинен бути обраний символ. Тому і в якості такого символу для формування конкретних функціональних структур, необхідно взяти з математичного аналізу її фундаментальний символ (1),

$$y = f(x) \qquad (5.1)$$

який не знайшов свого розвитку в математичному аналізі. І це пов'язано, у першу чергу з тим, що основним завданням математичного аналізу було формування розрахункових формул для знаходження конкретних значень в аналізованих процесах, і не ставилося завдання формування логіко–динамічного процесу такого розрахунку. При цьому слід зазначити, що сам логіко–динамічний процес розрахунку в математичному аналізі представлений у вигляді різних правил, за якими потрібно виконувати процес обчислення. Але якщо поставити завдання створення математичних моделей, як обчислювального процесу, так і будь–якого іншого процесу, то символ (}) математичного аналізу є «ідеальним» символом у цій ситуації, оскільки він загальновідомий і містить у собі смисловий зміст функціональної структури й може бути уточнений із застосуванням різних індексів і символів, які широко використовуються в конкретних наукових областях. При цьому слід зазначити, що такий перехід від загальноприйнятої форми запису до математичної моделі істотно спрощується тим, що відомі методи запису процесів, власне кажучи, включають тимчасову послідовність процесів перетворення сигналів. Оскільки будь–який формалізований процес у своєму записі включає з однієї сторони різнорідні процеси перетворення сигналів, наприклад, електрохімічні, електротехнічні, а також електронні у вигляді функціональних блоків системи керування й контролю. При цьому для кожного процесу перетворення розроблено свій метод формального відображення. Наприклад, у хімії – у вигляді хімічних формул, в електротехніці – у вигляді аналітичних формул, а в електроніці – у вигляді структурних схем, які настільки не сумісні, що говорити про загальну математичну модель різнорідного процесу перетворення не можна. А саме завдання формування математичної моделі різнорідних процесів перетворення виникає при комплексному дослідженні й оптимізації розглянутого процесу. Варто відмітити, що кожний з методів формального відображення має свої переваги, а точніше свої структурні якості, які природно необхідно зберегти й підсилити структурними якостями інших

методів формального відображення. У зв'язку із чим розглянемо метод формування математичної моделі структурних схем.

Відомі формалізовані методи запису структурних схем систем керування (рис. 5.1), де, власне кажучи, відсутня аналітична форма запису.

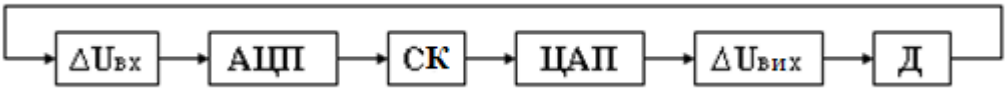


Рис. 5.1. Структурна схема типової системи керування

СК – система керування; ΔUвх – блок формування керуючого сигналу; АЦП – аналого–цифровий перетворювач; ЦАП – цифро–аналоговий перетворювач; ΔUвих – блок формування коригувального сигналу; Д – датчик системи керування.

З цього виходить, що для розуміння інформаційного змісту формалізованих виразів і необхідності формування різних способів мислення, а при перемиканні з одного способу мислення на інший обов'язково губиться логічний зв'язок, який необхідний при формуванні єдиного процесу. При цьому слід зазначити, що коли виникає необхідність «стикування» аналітично несумісних двох логіко–динамічних процесів у вищій математиці ця проблема називається крайовою задачею. У цьому випадку крайова задача вирішується простим методом, що, з однієї сторони містить у собі ***розробку загальної структурно–функціональної мови опису процесів*** перетворення, з іншої сторони запис процесів перетворення повинен виконуватися на аналітичному рівні у вигляді ***єдиної математичної моделі***. Тому зміна форми запису процесів перетворення в системах керування й контролю технологічним процесом, не тільки в енергетиці, очевидна. Оскільки аналітична форма запису необхідна для наступної формалізованої оптимізації конкретного технологічного процесу, тому для цього необхідно щоб математична модель, як одна з інформаційних технологій була проста в записі й мала максимальний інформаційний зміст.

5.2. Формування аналітичної форми запису функціональних структур з елементами інформаційних технологій

Аналіз логіки обчислень у розрахункових формулах математичного апарату й формування математичних моделей обчислювальних процесів. Логіка формування аналітичного запису функціональних аналогових і функціональних цифрових структур для практичного застосування при синтезі математичних моделей конкретних логіко–динамічних процесів перетворення сигналів.

5.2.1. Особливість структури розрахункової формули в основах математичного апарату

Якщо аналізувати проблему аналізу й синтезу елементів і пристроїв обчислювальної техніки й систем керування в цілому, то характерною основною рисою є структурний фактор, тобто структурна форма їхнього запису. При цьому в кожній структурі, незалежно від її складності, присутні вхідні аргументи й дії над ними, які прийнято позначати не в аналітичному виді, а у вигляді геометричних фігур зі спрямованими функціональними зв'язками, що не зовсім коректно. Всі аргументи у вигляді аналогових сигналів мають прийняту аналітичну форму запису за винятком дій або функцій, за допомогою яких здійснюється функціональне перетворення вхідних аргументів. Інакше кажучи, переходу від геометричних функціонально зв'язаних фігур до аналітичної форми запису структури функцій перешкоджає тільки відсутність методики формування аналітичних функцій. А якщо врахувати, що математичний апарат є аналітичним інструментом аналізу й синтезу логіко–динамічних процесів перетворення аргументів, то є сенс проаналізувати коректність його фундаментальних понять, визначень і правил у площині іншого фундаментального поняття, такого як структура функцій і методи можливого його аналітичного запису.

В основі аналітичної математики лежить поняття «функціональна залежність». Під цим поняттям мається на увазі деякий зв'язок змінної

величини, що називають *незалежною* або *аргументом* із *залежною змінною,* або *функцією*, й записують у вигляді:

$$y = f(x), \qquad (5.2)$$

при цьому залежно від конкретного напрямку в математиці виразу (5.2) дають різні визначення, при цьому в тих напрямках математики, які ближче до прикладних наук, вираз (3.1) визначають наступним чином: «*змінна величина* «y» *називається функцією змінної* «x»*, якщо кожному припустимому значенню* «x» *відповідає єдине значення* «y».

«При цьому існує стійка думка, що змінні величини в математиці відіграють важливу роль, оскільки вони служать засобом вивчення явищ природи й технологічних процесів перетворення аргументів. При цьому змінні величини використовуються не у відриві один від одного, а в їхньому взаємному зв'язку, у певній залежності між ними».

Наведений коментар щодо змінних величин досить сумнівний і не однозначний. По–перше, ніколи змінна величина або їхня сукупність у практичному змісті не була засобом вивчення технологічних процесів, а є предметом дослідження. По–друге, змінні величини ніколи не використовуються у взаємному зв'язку, оскільки вони самі по собі перебувають у взаємному зв'язку. По–третє, наявність асортиментів понять змінних величин приводить до невизначеності. По–четверте, досить сумнівний зв'язок між аргументом і функцією. Оскільки практичний досвід показує, що в природі існують вихідні аргументи, наприклад, хімічні речовини, електричні сигнали й т.д., які піддаються перетворенню, що приводить до кінцевого результату у вигляді перетвореного аргументу або їхньої сукупності. А, враховуючи, що під «функцією» розуміється дія або послідовність якихось дій, то в цьому зв'язку аргументами можуть бути тільки вихідні, проміжні й результуючі аналогові сигнали. У цьому зв'язку вираз (5.2) не коректний. Якщо розглянути вираз (5.2) як процес перетворення аргументів, то в цьому випадку «y» є продуктом якихось дій f(→) над вхідним аргументом «x», тому він є кінцевим результатом цих дій,

тому з урахуванням елементів функціональної структури, якій властивий часовий параметр вихідний вираз (5.3) є сенс записати або у вигляді

$$f(x) = y, \text{ або } f(x) \rightarrow y, \qquad (5.3)$$

якщо аналізувати скоректований узагальнений процес перетворення аргументів (3.2) з урахуванням послідовності в записі структурних схем обчислювальної техніки й систем керування, то цей вираз може бути представлений в узагальненому вигляді, як процес перетворення аргументу «**x**» в аргумент «**y**» у вигляді

$$x \rightarrow f(x) \rightarrow y \qquad (5.4)$$

$$x \rightarrow x,y \rightarrow y \qquad (5.5)$$

де f(x) і «x,y» – узагальнений запис функціональної структури, у якій вхідний аргумент x перетворюється у вихідний аргумент «y».

З аналізу аналітичних виразів (5.4) і (5.5) випливає, що з однієї сторони запропонований метод аналітичного запису процесу перетворення аргументів (5.4) більшою мірою відповідає схемним структурам, у яких функції зображуються у вигляді логічних елементів, які є електронними системами. Тому, якщо замість геометричної фігури логічного елемента використати математичний знак системи (**}**), то, наприклад, для елемента «І» можемо сформувати аналітичний запис його у вигляді

де x_1 і x_2 – вхідні аргументи;

f(&) – логічна функція «І».

З іншого боку метод аналітичного запису (5.5) процесу перетворення аргументів може бути використаний при аналітичному записі функціональних структур, у яких функціонально закінчені структури функцій представлені у вигляді функціональних блоків. Наприклад, аналітичний запис функціональних блоків аналого–цифрового й цифро–аналогового перетворення:

$$U_{вх} \rightarrow U_{вх},[m_i] \rightarrow [m_i]$$

$$[m_i] \rightarrow [m_i], U_j \rightarrow U_{вих}$$

Оскільки практично будь–який процес у пристроях обчислювальної техніки й систем керування, як правило, являє собою сукупність функціонально закінчених структур, то є сенс проаналізувати таке поняття, як «складна функція» у математичному апараті. З аналізу літературних джерел [36, 128] випливає , що, якщо дано, наприклад, дві тригонометричні функції

$$y = \sin(x) \tag{5.6}$$

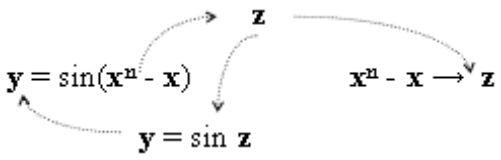
і

$$y = \sin(x^n - x), \tag{5.7}$$

то у функції (5.6) аргументом служить «x», а у функції (5.7) аргументом є вираз ($x^n - x$), що залежить від «x». І формулюється визначення «*функція, аргументом якої служить функція, називається складною функцією*» [10].

Якщо аналізувати смисловий зміст визначення складної функції, з урахуванням структурно–функціонального підходу в запису математичних моделей процесів перетворення аргументів в обчислювальній техніці й системах керування, то він представлений не коректно. Оскільки в математичному апараті відсутня чітка грань між функцією (дією або діями) і аргументом (об'єктом, над яким виконується ця дія). При цьому відсутнє таке поняття, як функціональна структура (структура функцій із вхідними аргументами), що власне кажучи й мається на увазі під тим аргументом у визначенні «складної функції».

Якщо повернутися до виразів (3.6) і виконати заміну функціональної структури ($x^n - x$) на аргумент «z», числове значення якого визначається в першу чергу, як:



незалежної функціональної структури, що має свою структуру функцій і свої вхідні аргументи, а результатом перетворення, якого є аргумент «z», то для того, щоб перевести формалізований процес перетворення аргументів (5.7) з категорії математичної формули в категорію математичної моделі, є сенс застосувати принципи запису аналітичних структур (5.8) і (5.9). У результаті формуємо математичну модель процесу формування перетвореного аргументу у вигляді виразу (5.8).

$$\left.\begin{array}{l} \mathbf{x} \rightarrow \mathbf{x}, \mathbf{x}^{\mathbf{n}} \rightarrow (\mathbf{x}^{\mathbf{n}}) \rightarrow \\ \qquad\qquad\qquad\qquad \mathbf{x} \rightarrow \end{array}\right\} (\mathbf{x}^{\mathbf{n}} - \mathbf{x}) \longrightarrow \sin(\mathbf{x}^{\mathbf{n}} - \mathbf{x}), \mathbf{y} \rightarrow \mathbf{y} \qquad (5.8)$$

З аналізу математичної моделі (5.8) процесу перетворення аргументу «x» в «y» витікає, що вона повністю адекватна процесу обчислення аргументу «y», оскільки перш ніж визначитися з перетвореним аргументом дії $\sin(x^n - x)$ необхідно в першу чергу виконати дію $x \rightarrow x$, $x^n \rightarrow (x^n)$, а потім чергову дія $(x^n - x)$ і тільки після виконання зазначеної послідовності дій може бути сформований перетворений аргумент «y». Інакше кажучи, функціональна послідовність дій, що представлена математичною формулою (5.7) позбавлена тимчасового параметра, який характерний структурі математичної моделі процесу перетворення аргументів (5.8), що й привело до формування неаналітичних методів формалізованого запису процесів перетворення аргументів, як у пристроях обчислювальної техніки, так і систем керування.

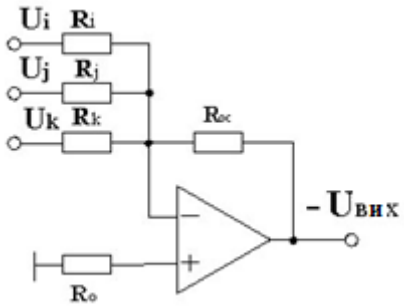
Слід особливо зазначити, що пропонований структурно–функціональний запис математичних моделей у вигляді структури функцій має практичне застосування при структурно–функціональному запису процесів перетворення аргументів в обчислювальній техніці й системах керування.

5.2.2. Логіка формування аналітичного запису функціональних аналогових і функціональних цифрових структур

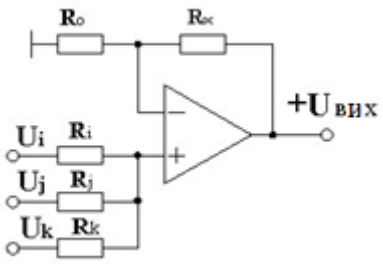
Після синтезу різноманіття схемних рішень функціонально закінчених вузлів і виготовлення їх у вигляді інтегральних мікросхем з'явилася необхідність у розробці методів синтезу математичних моделей процесів перетворення на базі функціональних вузлів, в основі яких лежить операційний підсилювач. Слід зазначити, що розроблені схемні рішення функціональних вузлів, які реалізують конкретні логіко–динамічні процеси перетворення сигналів, наприклад, додавання, множення в схемній реалізації й диференціювання, інтегрування й т.д. у формульній реалізації не сумісні що, в кінцевому разі, не дозволяє синтезувати функціонально закінчену математичну модель технологічного процесу перетворення сигналів. Слід також зазначити, що така несумісність формалізованого запису взаємозалежних технологічних процесів безпосередньо пов'язана з відсутністю загальної символіки, а існуючі символи в математичному апараті дозволяють вирішити цю проблему, оскільки є єдиним інструментом формалізованого запису будь–яких процесів перетворення сигналів.

Розглянемо можливість аналітичного запису різних схемних технічних рішень, при цьому слід зазначити, що аналітична форма запису структурно складається із вхідних сигналів, у системі конкретної функціональної структури й перетвореного сигналу. А всі схемні технічні рішення ставляться до категорії функціональних структур, які по своїй суті дискретні й тільки тому вони можуть бути записані, наприклад, у вигляді узагальненого аналітичного виразу «x = f(x) = y» для цифрових логічних структур або у вигляді узагальненого аналітичного виразу «x → f(x) → y» для функціональних аналогових структур. При цьому слід зазначити, що особливістю узагальнених аналітичних виразів «x = f(x) = y» і «x → f(x) → y» є символи (=) і (→) функціональних зв'язків вхідних сигналів «x» і перетворених сигналів «y», під якими будемо розуміти функціональний цифровий зв'язок у структурі функціональних логічних структур та

функціональний аналоговий зв’язок у структурі функціональних аналогових структур. Оскільки логіка формування аналітичної форми запису функціональних аналогових структур досить проста, тому приведемо два приклади такого запису, наприклад, функціональної структури з операційним підсилювачем f(ОП), які запишемо у вигляді аналітичних виразів (5.9) і (5.10).



$$\begin{array}{l} U_i \rightarrow f(R_i) \rightarrow \\ U_j \rightarrow f(R_j) \rightarrow \\ U_k \rightarrow f(R_i)_k \rightarrow \\ \downarrow f(R_{oc}) \rightarrow \\ U_o \rightarrow f(R_0) \rightarrow \end{array} \begin{array}{l} (-) \\ \\ \\ (+) \end{array} \Big\} \begin{array}{l} \rightarrow -U_{вих} \\ (f(ОП) \\ \rightarrow f(R_{oc})\uparrow \end{array} \qquad (5.9)$$



$$\begin{array}{l} U_o \rightarrow f(R_0) \rightarrow \\ \\ \downarrow f(R_{oc}) \rightarrow \\ \\ U_i \rightarrow f(R_i) \rightarrow \\ U_j \rightarrow f(R_j) \rightarrow \\ U_k \rightarrow f(R_i)_k \rightarrow \end{array} \begin{array}{l} (-) \\ \\ \\ \\ (+) \end{array} \Big\} \begin{array}{l} (f(ОП) \\ \\ \end{array} \begin{array}{l} \rightarrow f(R_{oc})\uparrow \\ \\ \rightarrow -U_{вих} \end{array} \qquad (5.10)$$

де U_i – U_k – вхідні аналогові сигнали;

U_0 – нульові аналогові сигнали;

– $U_{вих}$ –перетворений аналоговий сигнал;

$\downarrow f(R_{зз})$ і $f(R_{зз})\uparrow$ – резисторна функція зворотнього зв'язку;

$f(R_i)$ – $f(R_i)_k$ і $f(R_0)$ – вхідні резисторні функції входу, що інвертується (–) і входу, що не інвертується (+) функціональної структури операційного підсилювача f(ОП).

З аналізу аналітичних виразів (5.9) і (5.10) витікає, що їхній інформаційний зміст повністю еквівалентний інформаційному змісту їх схемних реалізацій.

5.3. Синтез математичної моделі логіко–динамічного процесу перетворення сигналів в електроенергетиці

Оскільки логіка формування функціональних аналітичних структур істотно не залежить від конкретного технологічного процесу і в її основі лежить узагальнений аналітичний вираз «x → f(x) → y», то приведемо приклад переходу структурної схеми процесу контролю й керування збудженням синхронного генератора [10, 81, 114, 138] при зміні частоти вихідної напруги до аналітичного її запису, а для цього запишемо графоаналітичний вираз (рис. 5.2).

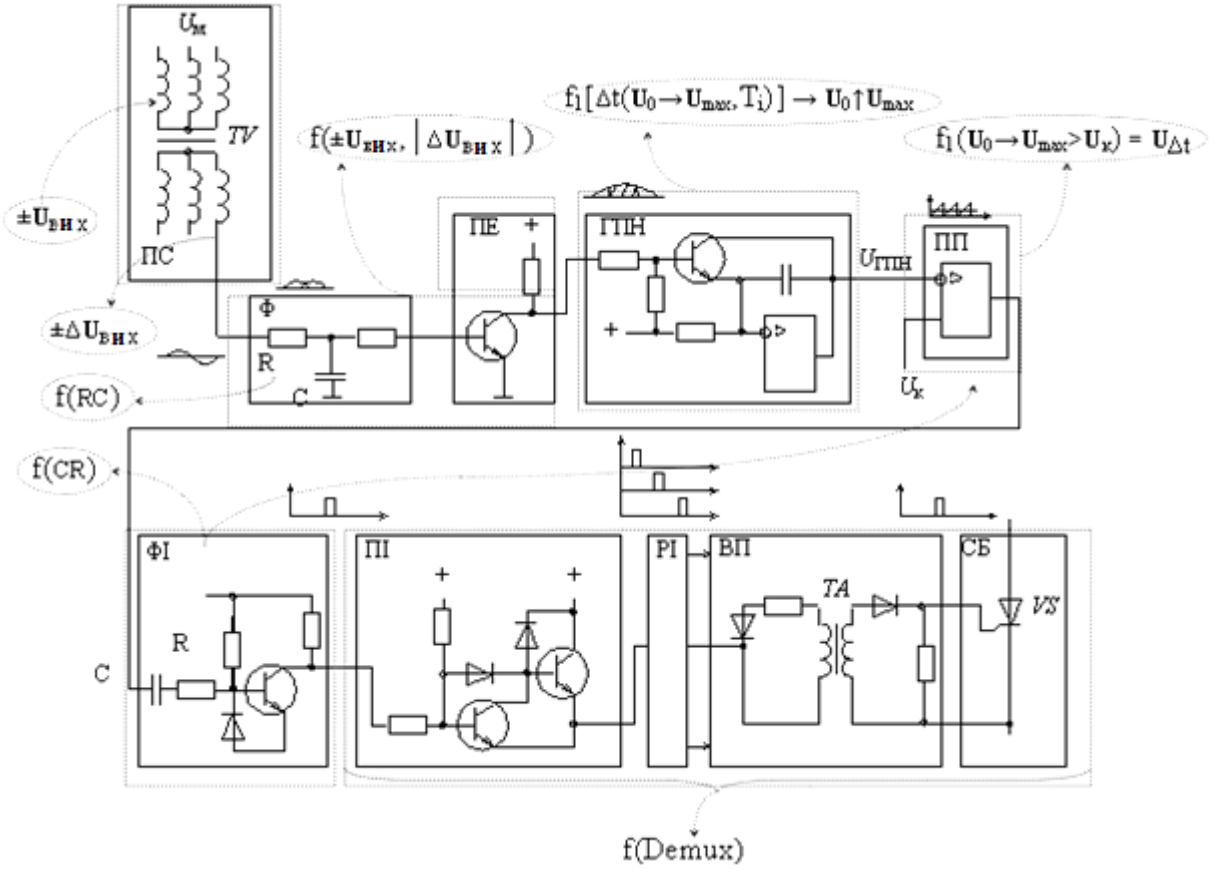


Рис. 5.2. Система керування збудженням синхронного генратора

Схему контролю й керування збудженням синхронного генератора можна розділити на сім функціональних структур, першою з якої є функціональна структура трансформатора, що, у даній ситуації, доцільно записати у вигляді результуючого змінного сигналу $\pm U_{вих}$. Другою й третьою функціональною структурою є структура f(RC) з інтегруючими

властивостями, оскільки включає послідовні резестивну f(R) і f(RC) ємнісну функціональні структури, і структуру із граничними властивостями, що виконує функцію мінімізації вихідної напруги генератора $\pm U_{вих}$ до величини $\pm\Delta U_{вих} \rightarrow |\Delta U_{вих}|$. В узагальненому вигляді це можна записати за допомогою аналітичного виразу $f(\pm U_{вих}, |\Delta U_{вих}|$. Четвертою функціональною структурою $f_1[\Delta t(U_0 \rightarrow U_{max}, T_i)] \rightarrow U_0 \uparrow U_{max}$ є функціональна структура, що формує пилоподібну напругу $U_0 \rightarrow U_{max}$ тривалістю Δt і періодом проходження T_i. Якщо об'єднати п'яту й шосту функціональні структури, які виконують функцію граничного елемента, і диференціатор f(CR), то їхній смисловий зміст можна записати у вигляді аналітичного виразу $f_1(U_0 \rightarrow U_{max} > U_к) = U_{\Delta t}$, що виконує функцію порівняння пилоподібної напруги $U_0 \rightarrow U_{max}$ з опорною $U_к$. Сьомою функціональною структурою є функціональна структура демультиплексора f(Demux). У результаті, відповідно до графоаналітичного виразу (3.4.1), процес стабілізації напруги синхронного генератора виконано відповідно до математичної моделі вигляду

$$\left.\begin{array}{l} U_{вих} \rightarrow f(\pm U_{кнх}, |\Delta U_{вих}|) \rightarrow f_1[\Delta t(U_0 \rightarrow U_{max}, T_i)] \rightarrow \\ U_j \rightarrow \end{array}\right\} f_1(U_0 \uparrow U_{max} > U_j) = (U_{\Delta t}) = f(\text{Demux}) \left\{\begin{array}{l} = U_{\varphi 1.n}(\pm\Delta t_{n+1}) \\ = U_{\varphi 2.n}(\pm\Delta t_{n+1}) \\ = U_{\varphi 3.n}(\pm\Delta t_{n+1}) \end{array}\right.$$

Для стабілізації роботи синхронного генератора фіксується момент зміни вихідної напруги генератора (момент перетинання напруги 0), і перетворюється вихідна напруга генератора $U_{вих}$ в аналоговий сигнал $\Delta U_{вих}$ за допомогою перетворюючої функціональної структури $f(\pm U_{вих}, |\Delta U_{вих}|)$ для подальшого порівняння його з опорною напругою U_j. Після цього формується пилоподібна напруга $U_0 \rightarrow U_{max}$ тривалістю Δt і періодом проходження T_i, де *i* – число періодів, за допомогою функціональної структури $f_1[\Delta t(U_0 \rightarrow U_{max}, T_i)]$. Далі відбувається порівняння пилоподібної напруги $U_0 \rightarrow U_{max}$ тривалістю Δt з опорною напругою U_j за допомогою функціональної структури $f_1(U_0 \rightarrow U_{max} > U_j)$, і формуються імпульсні керуючі сигнали, які після посилення подаються на розподільник імпульсів – демультиплексор f(Demux), за допомогою якого формуються послідовні керуючі імпульси $U_{\varphi 1.}$

$_{n}$, $U_{\varphi 2.\,n}$ і $U_{\varphi 3.\,n}$, які подаються на силові тиристори для формування струму збудження синхронного генератора:

$$U_{вих} \rightarrow f(\pm U_{вих}, |\Delta U_{вих}|) \rightarrow f_1[\Delta t(U_0 \rightarrow U_{max}, T_i)] \rightarrow \left.\begin{array}{l} \\ U_j \rightarrow \end{array}\right\} f_1(U_0 \uparrow U_{max} > U_j) = (U_{\Delta t}) = f(\text{Demux}) \left\{\begin{array}{l} = U_{\varphi 1.n}(\pm \Delta t_{n+1}) \\ = U_{\varphi 2.n}(\pm \Delta t_{n+1}) \\ = U_{\varphi 3.n}(\pm \Delta t_{n+1}) \end{array}\right.$$

де $U_{вих}$, $\Delta U_{вих}$– перетворююча функціональна структура;

$f_1[\Delta t(U_0 \rightarrow U_{max}, T_i)]$ – функціональна структура, яка формує пилоподібну напругу $U_0 \rightarrow U_{max}$ тривалістю Δt і періодом проходження T_i;

$f_1(U_0 \rightarrow U_{max} > U_j)$ – функціональна структура порівняння пилоподібної напруги $U_0 \rightarrow U_{max}$ з опорним U_j;

f(Demux) – функціональна структура демультиплексора.

Недоліком відомого способу є нестабільність роботи генератора через зміну його частоти вихідної напруги при зміні навантаження.

Реалізується запропонований спосіб керування збудженням синхронного генератора при зміні частоти вихідної напруги в такий спосіб. Попередньо, відповідно до аналітичного виразу (5.11):

$$\pm U_{вих} \dashrightarrow f(\pm U_{вих}, |\Delta U_{вих}|) \left\{\begin{array}{l} U_k \rightarrow \\ \rightarrow \end{array}\right\} f_1(|\Delta U_{вих}| < U_k) = U_{\Delta t} \uparrow \quad \left\{\begin{array}{l} \rightarrow \\ [U_j] \rightarrow \\ \downarrow U_{\Delta t} = \end{array}\right\} f_1(\text{АЦП}) = [U_j]_{вих} \uparrow \qquad (5.11)$$

вихідна напруга $U_{вих}$ синхронного генератора перетворюється функціональною структурою $f(\pm U_{вих}, |\Delta U_{вих}|)$ в аналоговий сигнал одного знаку $|\Delta U_{вих}|$. Потім перетворений аналоговий сигнал одночасно подається на функціональну структуру f_1(АЦП) з опорними аналоговими сигналами $[U_j]$, де «*j*» – число інформаційних логічних аналогових сигналів, і компаратор $f_1(|\Delta U_{вих}| < U_k)$, з опорним аналоговим сигналом U_k. При цьому, величина опорного аналогового сигналу U_k вибирається не вище лінійної частини перетвореного аналогового сигналу $|\Delta U_{вих}|$ синхронного генератора, і вихідний аналоговий сигнал $U_{\Delta t}\uparrow$ тривалістю Δt у компараторі $f_1(|\Delta U_{вих}| < U_k)$

формується тільки в лінійній частині перетвореного аналогового сигналу $|\Delta U_{вих}|$, яким активується функціональна структура f_1(АЦП) для формування структури аналогових сигналів $[U_j]_{вих.}$ Після цього, відповідно до функціональної структури (5.12)

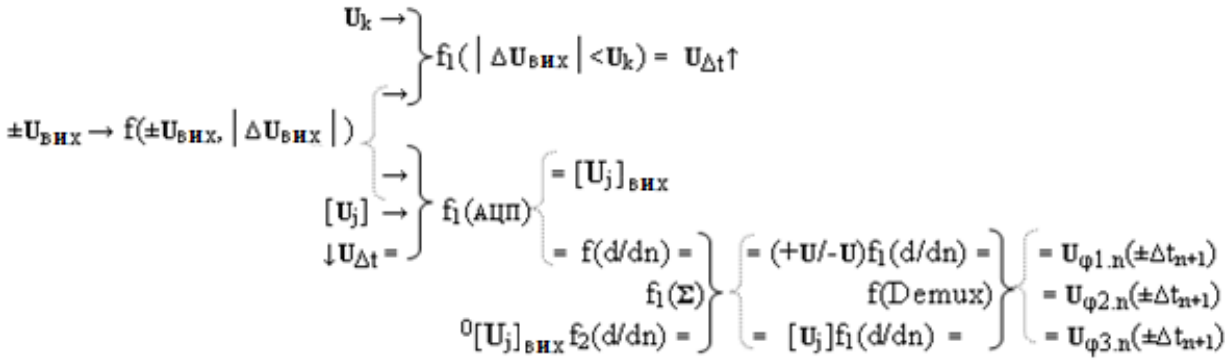
$$\left.\begin{array}{r} \downarrow[U_j]_{вих} = f(d/dn) = \\ f_1(\Sigma) \\ {}^0[U_j]_{вих}\, f_2(d/dn) = \end{array}\right\} \left\{\begin{array}{l} = (+U/-U) f_1(d/dn)\uparrow \\ \\ = [U_j] f_1(d/dn)\uparrow \end{array}\right. \qquad (5.12)$$

формується структура аналогових сигналів швидкості зміни аналогових сигналів $\downarrow[U_j]_{вих}$ за допомогою логічного диференціювання f(d/dn), і виконується логічне вирахування із заданої еталонної структури аналогових сигналів ${}^0[U_j]_{вих}$ $f_2(d/dn)$, з використанням суматора $f_1(\Sigma)$ для формування результуючого знака $(+U/–U)f_1(d/dn)$, і величини $[U_j]f_1(d/dn)$ логічної похідної. Потім, відповідно до функціональної структури (3)

$$\left.\begin{array}{r} \downarrow(+U/-U) f_1(d/dn) = \\ f(\text{Demux}) \\ \downarrow[U_j] f_1(d/dn) = \end{array}\right\} \left\{\begin{array}{l} = U_{\varphi 1.n}(\pm\Delta t_{n+1}) \\ = U_{\varphi 2.n}(\pm\Delta t_{n+1}) \\ = U_{\varphi 3.n}(\pm\Delta t_{n+1}) \end{array}\right. \qquad (5.13)$$

аналізується сформований у виразі (5.13) логічний сигнал знака $\downarrow(+U/–U)f_1(d/dn)$, і аналогові сигнали $\downarrow[U_j]f_1(d/dn)$, а також коректується момент початку формування керуючих імпульсів $U_{\varphi 1.n+1}$, $U_{\varphi 2.n+1}$ і $U_{\varphi 3.n+1}$ у демультиплексорі f(Demux) з випередженням або відставанням відносно початку формування попередньої послідовності керуючих імпульсів $U_{\varphi 1.n}(\pm\Delta t_{n+1})$, $U_{\varphi 2.n}(\pm\Delta t_{n+1})$ і $U_{\varphi 3.n}(\pm\Delta t_{n+1})$, які потім подаються на силові тиристори для формування струму збудження синхронного генератора.

Для формування математичної моделі функціонально закінченого логіко–динамічного процесу послідовного коректування струму збудження синхронного генератора для стабілізації його реактивної потужності виконаємо об'єднання функціональних структур (5.11) – (5.13) і запишемо результуючий аналітичний вираз логіко–динамічного процесу перетворення аналогових сигналів:



5.4. Приклад застосування математичних моделей

Ще один спосіб корегування струму збудження синхронного генератора при зміні частоти вихідної напруги відрізняється тим, що перетворений аналоговий сигнал вихідної напруги $\Delta U_{вих}$ перетворюється шляхом його порівняння (після аналого–цифрового перетворювача f_1(АЦП)) з позитивною й негативною опорними напругами $\pm U_0$, які розташовані поблизу нульового рівня аналогового сигналу $\pm\Delta U_{вих}$, в аналогові логічні сигнали знаку $\pm U_{вих}$ вхідної напруги $\pm U_{вих}$,, якими фіксують початок формування поточної структури аналогових сигналів $[U_j]f_1(T_t)$ за допомогою лічильника f(СТ), де «*j*» – число інформаційних логічних аналогових сигналів і аналогового сигналу $\pm U_{\Delta t}f(U_{вих}\downarrow\uparrow 0)$ знаку переходу через нульове значення. Після цього виконується логічне віднімання поточної структури аналогових сигналів $[U_j]f_1(T_t)$ і еталонної структури аналогових сигналів $[U_j]^0_{вих}$ $f_2(T_{ет})$, що відповідає контрольованому періоду Т вихідної напруги генератора $\pm U_{вих}$, з використанням суматора $f_1(\Sigma)$, і формується структура аналогових сигналів $[U_j]f_1(\Delta T)$ величини відхилення поточного періоду напруги генератора, а також аналоговий сигнал знаку $\pm Uf_1(\Delta T)\uparrow$ відхилення, що корегується з урахуванням аналогового сигналу $\downarrow\pm Uf_1(\Delta T)$ вихідного знака за допомогою логічної функції f_1(&)–I. Формується аналоговий сигнал +U/–U скорегованого знака й, спільно зі структурою аналогових сигналів $[U_j]f_1(\Delta T)$ величини відхилення поточного періоду напруги генератора, корегується початок формування послідовних керуючих імпульсів $U_{\varphi 1.\ n+1}$, $U_{\varphi 2.\ n+1}$ і $U_{\varphi 3.\ n+1}$ у демультиплексорі f(Demux) з випередженням або відставанням відносно

початку формування попередньої послідовності керуючих імпульсів $U_{\varphi 1.\,n}$, $U_{\varphi 2.\,n}$ і $U_{\varphi 3.\,n}$. Процес корегування струму збудження синхронного генератора виконаний відповідно до математичної моделі вигляду:

$$\pm U_{вих} \rightarrow f(\pm U_{вих}, \pm \Delta U_{вих}) \rightarrow \left.\begin{matrix} f_1(АЦП) \\ \pm U_0 \rightarrow \end{matrix}\right\} = \pm U_{вих} = R_0 \left\{\begin{matrix} [U_j]_{\Delta t} = C \\ f(CT) \\ = [U_j] f_1(T_t) = \end{matrix}\right. \left\{\begin{matrix} = \pm U_{\Delta t} f(U_{вих}\downarrow\uparrow 0) = \\ \\ = \left\{\begin{matrix} \downarrow \pm U f_1(\Delta T) = \\ \&_1 \end{matrix}\right\} = (+U/-U) = \\ f(Demux) \\ = [U_j] f_1(\Delta T) = \end{matrix}\right. \left\{\begin{matrix} = U_{\varphi 1.n}(\pm \Delta t_{n+1}) \\ = U_{\varphi 2.n}(\pm \Delta t_{n+1}) \\ = U_{\varphi 3.n}(\pm \Delta t_{n+1}) \end{matrix}\right.$$

$$f_1(\Sigma) \left\{\begin{matrix} [U_j]^0_{вих} \\ f_2(T_{e\tau}) = \end{matrix}\right. = \pm U f_1(\Delta T)\uparrow$$

де $\left.\begin{matrix} \rightarrow \\ \rightarrow \end{matrix}\right\} =$ і $\left.\begin{matrix} = \\ = \end{matrix}\right\} \left\{\begin{matrix} = \\ = \end{matrix}\right.$ – система аналогових зв'язків (→) і аналого–логічних

функціональних зв'язків (=) у функціональних структурах;

$\left.\begin{matrix} = \\ \&_1 \\ = \end{matrix}\right\} =$ – логічний елемент $f_1(\&)$–I;

$\left.\begin{matrix} = C \\ = R_0 \end{matrix}\right\} f(CT) =$ – функціональна структура логічного лічильника, у якому

(= R_0) – вхід скидання й (= C) – вхід прийому рахункових імпульсів;

$f_{1–3}$(АЦП) – функціональні структури аналого–цифрового перетворювача;

$f_1(\mathbf{\Sigma})$ – функціональна структура суматора;

f(d/dn) – функціональна структура логічного диференціювання;

f(Demux) – функціональна структура демультиплексора.

Розроблений метод синтезу логіко–динамічних процесів контролю й керування в електроенергетиці для формування математичних моделей дозволяє реалізувати їх з підвищеними технологічними якостями. Запропонований метод дозволяє перейти від різних структурних схем до їхньої математичної моделі, для подальшого аналізу та оптимізації.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Авен О.И. Оценка качества и оптимизация вычислительных систем / Авен О.И., Горин Н.Н., Коган Я.А. – М. : Наука, Главная редакция физ. мат. литературы, 1982.
2. Аветов Ю. В. Автоматизация процесса верификации протоколов информационно–вычислительных сетей / Ю. В. Аветов, А. А. Суконщиков, В. Пилецка // Вторая научно–техническая конференция советских и польских молодых ученых. – Киев, 1986. – С. 284 – 287.
3. Алексеев А. В. Применение нечеткой математики в задачах принятия решений // Методы и системы принятия решений : сборник. – Рига : РПИ, 1983. – С. 38 – 42.
4. Алексеев А. В. Проблемы разработки математического обеспечения выполнения нечетких алгоритмов / Алексеев А. В. // Модели выбора альтернатив в нечеткой среде : сборник. – Рига, 1984. – С. 79 – 82.
5. Алексеев Н. А. Микропроцессорная система управления судовой электроэнергетической системой “ASA–S” / Н. А. Алексеев, А. П. Баранов. – СПб. : ГМА им. адм. С. О. Макарова, 2000. – 132 с.
6. Алиев Р. А. Методы и алгоритмы координации в промышленных системах управления / Р. А. Алиев, М. И. Либерзон. – М : Радио и связь, 1987. – 208 с.
7. Алтунин А. Е. Модели и алгоритмы принятия решений в нечетких условиях: монография / А. Е. Алтунин, М. В. Семухин. – Тюмень: Издательство Тюменского гос. университета, 2000. – 352 с.
8. Аоки М. Введение в методы оптимизации / М. Аоки. – М : Наука, 1977. – 344 с.
9. Апанасович В.В. Matlab и Simulink для Windows. Руководство пользователя ПЭВМ / Апанасович В.В., Скакун В.В. – Мн. : Белгосуниверситет, 1997. –72с.

10. Баранов А. П. Автоматическое управление судовыми электроэнергетическими установками / А. П. Баранов. – М. : Транспорт, 1981.
11. Баранов С. И. Синтез микропрограммных автоматов (граф – схемы и автоматы) / С. И. Баранов. – Л. : Энергия, 1979.
12. Баркалов О. О. Синтез пристроїв керування на програмованих логічних пристроях / О. О. Баркалов. – Донецьк : РВА ДонНТУ, 2002. – 262 с.
13. Блэк Ю. Сети ЭВМ: протоколы, стандарты, интерфейсы / Блэк Ю. Пер. с англ. – М. : Мир, 1990.
14. Беллман Р. Динамическое программирование / Беллман Р. – М. : Изд–во иностр. лит., 1960.
15. Бельков Д.В. ИССЛЕДОВАНИЕ СЕТЕВОГО ТРАФИКА. / Бельков Д.В. // Наукові праці ДонНТУ, серія "Інформатика, кібернетика та обчислювальна техніка", № 10. – С. 212 – 215.
16. Богданов Д.В. Модели и алгоритмы концептуального проектирования автоматизированных систем управления / Д.В. Богданов, Е.Б. Мазаков, О.Б. Неилко, С.Г. Чекинов. Под. ред. С.Г. Чекинова. – М. : Компания Спутник+, 2004. – 324 с. ISBN 5-93406-793-1.
17. Богуславский Л. В. Оценка производительности распределенных информационно–вычислительных систем архитектуры «КЛИЕНТ–СЕРВЕР» / Л. В. Богуславский, А. И. Ляхов // Автоматика и телемеханика. – 1995. – № 9. – С. 160 – 175.
18. Бойченко Е. В. Методы схематического проектирования распределенных информационно–вычислительных микропроцессорных систем / Е. В. Бойченко. – М. : Энергоатомиздат, 1988.
19. Бокша В. В. Нечеткое целевое управление системами с заданным конечным состоянием / В. В. Бокша, В. Б. Силов // Автоматика. – № 3. – 1985. – С. 3 – 8.

20. Болховитинов В. К. Структурная и аппаратная реализация системы управления корабельной электроэнергетической системой / В. К. Болховитинов, В. Л. Вайнер. – Автоматизированные системы управления НК и ПЛ.
21. Брайтон А. Прикладная теория оптимального управления / А. Брайтон, Ю–ши Хо. – М. : Мир, 1972. – 544 с.
22. Бронштейн О. И. Модели приоритетного обслуживания в информационно–вычислительных сетях / О. И. Бронштейн, И. М. Духовный. – М. : Наука. – 1976.
23. Брудник С.С. Определение экономической эффективности программных средств в АСУ / Брудник С.С., Кочегарова И.А., Степин Ю.П., Чикиров А.Б. – М. : ГАНГ, 1995г.
24. Бурлаков М. В. Ситуационное управление в системах массового обслуживания / М. В. Бурлаков. – Киев : Наукова думка, 1991. – 160 с.
25. Буч Г. UML. Руководство пользователя / Г. Буч, Г. Рамбо, И. Якобсон. – М. : ДМК, 2000. – 358 с.
26. Вишневский В. М. Теоретические основы проектирования компьютерных сетей / В. М. Вишневский. – Москва : Техносфера, 2003. – 512 с.
27. Воробьев В. И. Теория и практика вей влет–преобразования / В. И. Воробьев, В. Г. Грибунин. – СПб. : Изд–во ВУС, 1999.
28. Герхардт Х.–Д. Комби–сети – обобщение сетей Петри для развития и представления комбинированных сетевых моделей / Х.–Д. Герхардт // Теория сложных систем и методы их моделирования. ВНИИ системных исследований. – 1984, № 4. – С. 49 – 57.
29. Головин Ю. А. Информационные сети и телекоммуникации. Ч. 1 : уч. пособие / Ю. А. Головин, А. А. Суконщиков. – Вологда : ВоГТУ, 2001. – 144 с.

30. Гома Хассан. UML. Проектирование систем реального времени, распределенных и параллельных приложений / Хассан Гома. – М. : ДМК, 2002.
31. Горбатов В. А. Логическое управление распределенными системами / В. А. Горбатов, М. И. Смирнов, И. С. Хлытчев. – М. : Энергоиздат, 1991.
32. Горцев А. М. Управление и адаптация в системах массового обслуживания / А. М. Горцев, А. А. Назаров, А. Ф. Терпугов. – Томск : Изд–во томского университета, 1978. – 208 с.
33. Грэхем И. Объектно–ориентированные методы. Принципы и практика / И. Грэхем. – М. : Вильямс, 2004. – 768 с.
34. Давыдов Д. В. Нечёткие методы анализа надежности локальных вычислительных сетей / Д. В. Давыдов // Информация – Коммуникация – Общество (ИКО–2002): тезисы докладов и выступлений Международной научной конференции. – Санкт–Петербург : СПб., 2002. – С. 71 – 73.
35. Давыдов Д. В. Нечеткая логика при анализе локальных вычислительных сетей / Д. В. Давыдов // Управление и информационные технологии. Всероссийская научная конференция 3 – 4 апреля 2003 г. Санкт–Петербург : сборник докладов в 2 т. Том 1. – 2003. – С. 305 – 310.
36. Демидович Б. П. Основы вычислительной математики / Б. П. Демидович, И. А. Марон. – М. : Наука, 1970 . – 664 с.
37. Денисов А. А. Теория больших систем управления / А. А. Денисов, Д. Н. Колесников. – Ленинград: Энергоиздат. Ленингр. отделение, 1982 .
38. Директор С. Введение в теорию систем / С. Директор, Р. Рокер. – М. : Мир, 1974. – 464 с.
39. Дубинин В. Н. Проектирование распределенных систем управления промышленными процессами с использованием UML–FB / В. Н. Дубинин, В. В. Вяткин // Известия высших учебных заведений.

Поволжский регион. Серия : технические науки. –2004. – № 2. – С. 136 – 146. – ISSN 1728–628X.

40. Дудников Е. Е. Типовые задачи оперативного управления непрерывным производством / Е. Е. Дудников, Ю. М. Цодиков. – М. : Энергия, 1979. – 279 с.
41. Дьяконов В. Математические пакеты расширения Matlab : специальный справочник / В. Дьяконов, В. Круглов. – СПб. : Питер, 2001. – 480 с.
42. Евдокимов А. Г. Потокораспределение в инженерных сетях / А. Г. Евдокимов, В. В. Дубровский, А. Д. Тевяшев. – М. : Стройиздат, 1979. – 199 с.
43. Евреинов Э. В. Цифровые автоматы с настраиваемой структурой / Э. В. Евреинов, И. В. Прангишвили. – М. : Энергия, 1984.
44. Жожикашвили В. А. Сети массового обслуживания. Теория и применение к сетям ЭВМ / В. А. Жожикашвили, В. М. Вишневский. – М. : Радио и связь, 1988.
45. Загребной Ю.В. Исследование проблемы управления комплексом устройств в режиме реального времени / Загребной Ю.В. // Матеріали IX Науково–технічної конференції ЗДІА. – Запоріжжя : ЗДІА, 2004, №2. – с. 42 – 43.
46. Заде Л. А. Размытые множества и их применение в распознавании образов и кластер–анализе / Л. А. Заде // Классификация и кластер : сборник. – М. : Мир, 1980. – С. 208 – 247.
47. Зайцев Д.А. Оценка характеристик сетей Ethernet с помощью параметрических моделей Петри / Зайцев Д.А., Шмелёва Т.Р. // Зв'язок, № 4, 2007. – С. 62 – 67.
48. Зайцев Д.А., Литвин Д.А. Оценка производительности сетей с коммутацией меток в моделирующей системе NS / Зайцев Д.А., Литвин Д.А. // Збірник Наукових праць ОНАЗ ім. О.С. Попова, № 1, 2007. – С. 25 – 31.
49. Зайцев Д.А. Моделирование телекоммуникационных сетей в системе

NS / Зайцев Д.А., Шинкарчук Т.Н. // Збірник Наукових праць ОНАЗ ім. О.С. Попова, № 2, 2006. – С. 35 – 43.

50. Зайцев Д.А. Исследование эффективности технологии MPLS с помощью раскрашенных сетей Петри / Зайцев Д.А., Сакун А.Л. // Зв'язок. – 2006, №5. – С. 49–55.
51. Зайцев Д.А. Решение линейных систем с помощью декомпозиции / Зайцев Д.А. // Системні дослідження та інформаційні технології, 2005, №2, – С. 131 – 143.
52. Зоркальцев А. В. Анализ локального управления потоками в узле коммутации пакетов / А. В. Зоркальцев // АВТ. – 1992. – № 4. – С. 20 – 27.
53. Иванов А.В. Разработка и исследование алгоритмов прогнозирования и управления очередями в компьютерных сетях / Іванов А.В. // Автореферат диссертации. Санкт-Петербург. – 2001. – 18 с.
54. Иыуду К. А. Надежность, контроль и диагностика вычислительных машин и систем: учеб. пособие для вузов по спец. “Вычислительные машины, комплексы, системы и сети” / К. А. Иыуду. – М. : Высш.шк., 1989. – 216 с. : ил.
55. Карпов Ю. Г. Теория алгоритмов и автоматов / Ю. Г. Каропов. – СПб.: Геликон Плюс, 2000.
56. Карпухин А.В. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ХАОТИЧЕСКИХ ЯВЛЕНИЙ В ВЫСОКОСКОРОСТНЫХ СЕТЕВЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ С ПРОТОКОЛОМ TCP / Карпухин А.В. // Системи обробки інформації, 2009, випуск 4 (78). – С. 64 – 69. ISSN 1681-7710.
57. Карпухин А.В. Анализ поведения информационных систем с большой нелинейностью / А.В. Карпухин, Ю.П. Мачехин // Материалы междунар. науч. конф. ISDMCI’2000, 18-23 мая 2008 г. Евпатория. – Херсон, 2008. – Т. 3. – 192 с.

58. Кейн В. М. Оптимизация систем управления по минимаксному критерию / В. М. Кейн. – М. : Наука, 1985. – 248 с.
59. Клейнрок Л. Вычислительные системы с очередями / Л. Клейнрок : пер. с англ. – М. : Мир, 1979. – 600 с.
60. Клейнрок Л. Теория массового обслуживания / Клейнрок Л. – М. : Машиностроение, 1979.
61. Коваленко А.А. Проблемы производительности протокола TCP в гетерогенных сетях и методы ее улучшения / Коваленко А.А., Завизиступ Ю.Ю., Партыка С.А. // Вестник ХНТУ. – 2005, №1(21). – С. 305 – 311.
62. Коваленко И.И. Паттерновые сети и некоторые примеры их применения / И.И. Коваленко, В.М. Рябенький, О.А. Кудин // Проблемы информационных технологий,№2 (4). Херсон : Изд-во ХНТУ, – 2008. – С. 170 – 175.
63. Коваленко И.И. Представление структур вычислительных систем паттерновыми сетями / И.И. Коваленко, О.А. Кудин // Збірник наукових праць НУК. – Миколаїв : НУК, 2009. – №6 (429). – С. 164 – 172.
64. Ковальчук М.В. Мониторинг качества передачи случайного трафика в компьютерной TCP/IP сети / Ковальчук М.В. // Матеріали конференції «Інформаційно–керуючі системи і комплекси». – Миколаїв : НУК, 2004. – С. 142 – 143.
65. Колесник А.Б. Моделирование системы инициализации маршрутов для оптимальной маршрутизации и загрузки безпроводных сетей / А.Б. Колесник, Э.Г. Петров, Л.В. Колесник // Вестник ХНТУ. – 2007, №4(27). – С. 147 – 151.
66. Краснов В. В. Основы теории и расчета судовых электроэнергетических систем / В. В. Краснов, П. А. Мещанинов.– Л. : Судостроение, 1989. – 325 с.
67. Кристофидес Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – М. : Мир, 1978. – 432 с.

68. Кривуля Г.Ф. Моделирование функциональных состояний компьютерной системы / Г.Ф. Кривуля, М.А. Лаптев // Вестник ХНТУ. – 2007, №4(27). – С. 258 – 262.
69. Кривуля Г.Ф. Диагностирование отказов компьютерных систем с использованием интеллектуальных средств / Г.Ф. Кривуля, Д.Е. Кучеренко, А.А. Давыдов // Вестник ХНТУ. – 2010, №2(38). – С. 266 – 271.
70. Крюков В.А. Иерархическое конфигурационное управление / Крюков В.А., Максимов А.В., Петренко А.К., Полилова Т.А. – Программирование, №2, 1994.
71. Кудзиновская И.П. МЕТОД ОБНАРУЖЕНИЯ ПЕРЕГРУЗКИ СЕТЕВЫХ УЗЛОВ ПО СКОРОСТИ РОСТА ОЧЕРЕДЕЙ / Кудзиновская И.П. // Наукові записки УНДІЗ, №2(10), 2009. – С. 74 – 78.
72. Кудзиновская И. П. Анализ методов обеспечения качества обслуживания в высокоскоростных компьютерных сетях // Проблеми інформатизації та управління. – 2008. – №1(23). – С. 182 – 187.
73. Кузнецов О. П. Дискретная математика для инженера / О. П. Кузнецов, Г. М. Адельсон–Вельский. – Изд. 2–е, перераб. и доп. – М. : Энергоатомиздат, 1988. – 480 с. : ил.
74. Кузнецов Б. П. Стандартная реализация управляющих программ / Б. П. Кузнецов // Судостроительная промышленность. Серия: системы автоматизированного проектирования. – 1986. – С. 51 – 55.
75. Кучук Г.А. Аналіз та моделі самоподібного трафіка / Кучук Г.А., Можаєв О.О., Воробйов О.В // Авиационно-космическаятехника и технология. – 2006. – № 9(35). – С. 173–180.
76. Лазарев В. Г. Динамическое управление потоками в сетях связи / В. Г. Лазарев, Ю. В. Лазарев. – М. : Радио и связь, 1983. – 216 с.
77. Лазарев В. Г. Проектирование дискретных устройств автоматики / В. Г. Лазарев, Н. П. Маркин. – М. : Радио и связь, 1985.

78. Лазарев В. Г. Исследование вероятностного метода адаптивной маршрутизации для сети коммутации пакетов / В. Г. Лазарев, Н. Я. Паршенков, Л. Фратта // Сетевые протоколы и управление в распределенных вычислительных системах. – М. : Наука, 1986. – С. 3 – 14.
79. Леоненков А. В. Нечеткое моделирование в среде Matlab и fuzzyTECH / А. В. Леоненков. – СПб. : БХВ – Петербург, 2003. – 736 с.
80. Лескин А. А. Сети Петри в моделировании и управлении / А. А. Лескин, П. А. Мальцев, А. М. Спиридонов. – Л.: Наука, 1989. – 133 с.
81. Лясковский Г. Д. Справочник по наладке судовой полупроводниковой автоматики / Г. Д. Лясковский, В. М. Рябенький, В. Л. Скороходов. – Л. : Судостроение, 1991. – С. 34 – 45.
82. Математическое моделирование сильно нелинейных информационных систем / А.В. Карпухин, Л.О. Кириченко, Ю.П. Мачехин, Т.А. Радивилова // Материалы 1-й междунар. научн. конф. «Электронная компонентная база. Состояние и перспективы развития, 30 сентября – 3 октября 2008 г. Судак. – Х. : ХНУРЭ, 2008. – 346 с.
83. Марк А. Маршрутизаторы / Марк А. Миллер // Computerworld, №11, 1994.
84. Майн Х. Марковские процессы принятия решений / Майн Х. Осаки С. – М. : Наука, 1977.
85. Медведев Г.А. Характеристика случайных процессов в ЛВС со случайным доступом и несимметричной нагрузкой / Медведев Г.А. // АВТ. – 1994, №3. – С.40–48.
86. Меликов А.З. Математические модели мультиресурсных систем обслуживания и их оптимизация / Меликов А.З., Пономаренко Л.А. – Киев : Ин–т кибернетики. – 1999. – 19 с.
87. Мишель Ж. Программируемые контроллеры. Архитектура и применение / Ж. Мишель. – М. : Машиностроение, 1992.
88. Можаев А.А. УСОВЕРШЕНСТВОВАНИЕ ТРАНСПОРТНЫХ

ПРОТОКОЛОВ В МУЛЬТИСЕРВИСНЫХ СЕТЯХ / А.А. Можаев, А.А. Подорожняк, И.В. Ильина, А.А. Коваленко // Системи обробки інформації, 2007, випуск 5 (63). – С. 86 – 92.

89. Можаєв О.О. Властивість довгочасової залежності ієрархічної моделі / Можаєв О.О., Ільїна І.В., Пантелей Ю.В. // Системи озброєння і військова техніка. – 2007, №1(9). – С. 98 – 103.
90. Можаев А.А. Нейродинамическое прогнозирование телекоммуникационного трафика средств космической связи / Можаев А.А., Подорожняк А.А., Воробьев О.В. // Авиационно-космическая техника и технология. – 2006, № 6(32). – С. 67 – 70.
91. Моисеев Н. Н. Математические задачи системного анализа / Н. Н. Моисеев. – М. : Наука, 1981. – 488 с.
92. Нагорный Ю.И. Некоторые подходы к комплексной автоматизации промышленных производств / Ю.И. Нагорный, А.А. Балтовский // Вестник ХНТУ. – 2007, №4(27). – С. 308 – 311.
93. Надежность автоматизированных ИС : уч. пособие / Фатуев В. А. и др. – Тула, 1998. – 104 с.
94. Нессер Д. Д. Оптимизация и поиск неисправностей в сетях / Нессер Д. Д. – Киев : Изд. Диалектика, 1996 г.
95. Немченко В.П. Автоматный подход к тестированию сетевых протоколов / Немченко В.П., Чепелев Ю.А., Изотов А.С. // Вестник ХНТУ. – 2010. – №2(38). – С. 272 – 275.
96. Нечёткие множества в моделях управления и искусственного интеллекта / Аверкин А. Н. и др. – М. : Наука, 1986. – 380 с.
97. Обработка нечёткой информации в системах принятия решений / Борисов А. Н. и др. – М. : Радио и связь, 1989. – 234 с.
98. Овчаренко Н. И. Автоматика электрических станций и электроэнергетических систем. – М. : Издательство НЦ ЭНАС, 2001. – 504 с.

99. Одинаев В.А. Математическая модель пространства состояний корабельной электроэнергетической системы. Принятие оперативных решений / Одинаев В.А. // Судостроение, – 2002. № 5. – С. 42 – 44.
100. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В. Г. Олифер, Н. А. Олифер. – 2–изд. – СПб. : Питер–пресс, 2002. – 672 с.
101. Олифер В. Искусство оптимизации трафика. LAN / В. Олифер, Н. Олифер // Журнал сетевых решений. – 2001. – № 12.
102. Олифер В. Г. Новые технологии и оборудование IP–сетей BHV / В. Г. Олифер, Н. А. Олифер. – СПб., 2000. – 512 с.
103. Передерій В. І. Визначення та оцінка значимості параметрів комп'ютерних систем та мереж за допомогою теорії нечітких множин / В. І. Передерій
104. Передерий В. И. Информационное обследование предприятий / В. И. Передерий, В. Э. Ходаков // Механизация и автоматизация управления. – 1990. – № 1.
105. Петров В.В. Структура телетрафика и алгоритм обеспечения качества обслуживания при влиянии эффекта самоподобия / Петров В.В. // Автореферат диссертации. Москва. – 2005. – 20 с.
106. Полетаева А.Н. Механизмы сбора и использования маршрутной информации / А.Н. Полетаева, В.В. Карманова // Вестник ХНТУ. – 2009. – №1(34). – С. 362 – 368.
107. Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.x. В 2 т. Т. 2. – М. : ДИАЛОГ–МИФИ, 1999. – 304 с.
108. Рябенький В.М. Реализация системы синхронизации генераторов с использованием SWITCH–технологи / Рябенький В.М., Ушкаренко А.О. // Технічна електродинаміка, частина 4. Київ, 2006. – С. 108 – 112.
109. Рябенький В.М. Використання програмованих логічних матриць для реалізації блоку синхронізації генераторів / Рябенький В.М.,

Ушкаренко А.О. // Технічна електродинаміка, частина 7. Київ, 2006. – С. 89–92.

110. Рябенький В.М. Иерархическая модель системы управления судовой электроэнергетикой / В.М.Рябенький, А.О.Ушкаренко, В.С.Буряк // Вестник ХНТУ. – Херсон : ХНТУ, 2006. №1 (24) – С. 333 – 337.
111. Рябенький В.М. Проектирование распределенных систем управления с использованием UML / Рябенький В.М., Ушкаренко А.О. // Проблемы информационных технологий. – Херсон : ХНТУ, 2007, №1. – С. 132 – 139.
112. Рябенький В.М. Использование UML–FB в проектировании и моделировании систем управления электроэнергетическими системами / Рябенький В.М., Ушкаренко А.О., Спекторенко Р.А. // Вестник ХНТУ. – Херсон : ХНТУ, 2007, №4. – С. 185 – 194.
113. Рябенький В.М. Описание информационных потоков в распределенных системах управления с использованием UML / Рябенький В.М., Ушкаренко А.О. // Проблемы информационных технологий. – Херсон : ХНТУ, 2007, №1. – С. 127–131.
114. Рябенький В.М. Моделирование микропроцессорных систем управления газодизель–генераторными установками / В.М. Рябенький, А.О. Ушкаренко, В.И. Воскобоенко // Вісник Національного університету «Львівська політехніка». – Львів : НУ «Львівська політехніка», 2009, №637. – С. 78 – 82.
115. Рябенький В.М. Управление информационными потоками в сетях распределенных систем управления / В.М. Рябенький, А.О. Ушкаренко, А. Сулейман, А.Л. Белоконь. // Вестник ХНТУ. – Херсон : ХНТУ, 2009, №1(34). – С. 265 – 270.
116. Рябенький В.М. Минимизация коллизий информационных потоков в сетях распределенных систем управления / В.М. Рябенький, А.О. Ушкаренко, А. Сулейман // Вестник ХНТУ. – Херсон : ХНТУ, 2009, №1(34). – С. 271 – 274.

117. Рябенький В.М. Определение параметров сетей обработки информации аналитическими средствами / В.М. Рябенький, А.О. Ушкаренко, А. Сулейман, А.Л. Белоконь // Проблеми інформаційних технологій. Херсон : ХНТУ, 2009. №5. С. 171–174.
118. Рябенький В.М. Методика создания программного обеспечения для систем автоматизации электроэнергетических установок / В.М. Рябенький, А.О. Ушкаренко // Технічна електродинаміка. Тем. випуск. Київ, 2009. №3 – С. 72–75.
119. Рябенький В.М. Разработка программного обеспечения для автоматизированных электроэнергетических систем с использованием UML / Рябенький В.М., Ушкаренко А.О. // Вестник ХНТУ. – Херсон : ХНТУ, 2010, №2(38). – С. 244 – 249.
120. Рябенький В.М. Исследование аналитической модели сети при изменении параметров информационных потоков / Рябенький В.М., Ушкаренко А.О., Сулейман А. // Вестник ХНТУ. – Херсон : ХНТУ, 2010, №2(38). – С. 250 – 253.
121. Рябенький В.М. Помехоустойчивое кодирование информации с использованием SWITCH–технологи / Рябенький В.М., Ушкаренко А.О., Сулейман А. // Вестник ХНТУ. – Херсон : ХНТУ, 2010, №2 (38). – С. 254 – 257.
122. Саламех Осама Иззат. Методы и модели расчета характеристик производительности ЛВС / Саламех Осама Иззат // Автореферат дис. канд. техн. наук. — Одесса : ОПУ, 1995.
123. Советов Б.Я. Применение микропроцессорных средств в системах передачи информации / Б. Я. Советов, О. И. Кутузов, Ю. А. Головин, Ю. В. Аветов. Учеб. пособие для вузов по спец. “АСУ” – М. : Высш. шк., 1987. – 256 с.
124. Советов Б. Я. Моделирование систем / Б. Я. Советов, С. А. Яковлев. – М. : Высшая школа, 1995. – 320 с.

125. Сосокин В. Л. Программное управление технологическим оборудованием / В. Л. Сосокин. – М. : Машиностроение, 1991. – 532 с.
126. Стасєв Ю.В. Вплив помилки прогнозування характеристик трафіка на ефективність транспортного протоколу мережі / Стасєв Ю.В., Кучук Г.А., Можаєв О.О. // Системи озброєння і військова техніка. – 2007. – №2(10). – С. 69–74.
127. Суконщиков А. А. Применение аппарата нечёткой логики в сетевых технологиях / А. А. Суконщиков, Д. В. Давыдов // Перспективные технологии автоматизации: тезисы докладов международной электронной научно–технической конференции. – Вологда : ВоГТУ, 1999. – С. 33 – 34.
128. Схрейвер А. Теория линейного и целочисленного программирования. / А.Схрейвер. В 2-х т. – М. : Мир, 1991. – 726 с.
129. Танянский С.С. Модель процессов выполнения множества запросов на основе сетей Петри / С.С. Танянский // Вестник ХНТУ. – 2009. – №1 (34). – С. 298 – 302.
130. Тарасюк О.М. Анализ избыточности протокола стека TCP/IP / О.М. Тарасюк, А.В. Горбенко, Мохамед Саид Газал // Системи обробки інформації, 2009, №1 (75). – С. 130 – 133.
131. Трахтенгерц Э.А. Программное обеспечение параллельных процессов / Трахтенгерц Э.А. – М. : Наука, 1987.
132. Тютюнник М. Н. Web–технологии в промышленной автоматизации / М. Н. Тютюнник, А. В. Юрчак // Корпоративные системы. – 1999. – № 4.
133. Усов А.В. Оптимизация прогнозных характеристик информационных потоков для повышения эффективности функционирования корпоративных компьютерных сетей / А.В. Усов, О.А. Усова, Ю.С. Нестеренко // Вестник ХНТУ. – 2010. – №2 (38). – С. 217 – 220.
134. Уткин Л. В. и др. Ускоренная оценка качества функционирования вычислительных сетей при нечётких исходных данных // Труды

Международной конференции по проблемам функционирования вычислительных сетей ПСИФ – 91. – Новосибирск. Россия, 1991. – Вып. 3, – С. 232 – 239.

135. Ушкаренко О.О. Реалізація системи керування трифазним тиристорним випрямлячем з використанням автоматного програмування / Ушкаренко О.О., Спекторенко Р.А., Світлинський К.В. // Електротехніка і електромеханіка : Матеріали Всеукраїнської науково–технічної конференції студентів, аспірантів, молодих вчених з міжнародною участю. – Миколаїв : НУК, 2007. – С. 87–91.
136. Ушкаренко А.О. Мониторинг и управление электроэнергетической системой с использованием MATLAB / Ушкаренко А.О., До Ань Туан, Спекторенко Р.А., Светлинский К.В. // Проблемы информационных технологий. – Херсон : ХНТУ, 2007, №2.
137. Фісун М.Т. Впровадження GRID-технологій для проведення лабораторних занять з розподілених обчислень / М.Т. Фісун, Л.О. Макаров // Вестник ХНТУ. – 2007. – №4 (27). – С. 328 – 335.
138. Фрейдзон И. Р. Микропроцессорные системы управления техническими средствами судов / И. Р. Фрейдзон, Л. Г. Филиппов, Р. И. Фрейдзон. – Л. : Судостроение, 1985. – 248 с.
139. Ховард Р.А. Динамическое программирование и марковские процессы / Ховард Р.А. – М. : Радио, 2004. – 191 с.
140. Черных И. В. Simulink: Среда создания инженерных приложений / И. В. Черных. – М. : Изд–во : Диалог–МИФИ, 2004. – 496 с.
141. Чуи К. Введение в вэйвлеты / К. Чуи. – М. : Мир, 2001.
142. Шаповалова Е. А. Моделирование с использованием языка UML. Пакет Rational Rose. (http://www.uran.donetsk.ua/).
143. Шаповалова Е. А. Язык UML, http://www.uran.donetsk.ua/~masters/2005/kita/shapovalova/ind/index.htm
144. Шалыто А. А. От тьюрингова программирования к автоматному / А. А. Шалыто, Н. И. Туккель // Мир ПК. – 2002. – № 2. – С. 144 – 149.

145. Шалыто А. А. SWITCH–технология. Алгоритмизация и программирование задач логического управления / А. А. Шалыто. – СПб. : Наука, 1998. – 628 с.
146. Шалыто А.А. Использование граф–схем алгоритмов и графов переходов при программной реализации алгоритмов логического управления / Шалыто А.А. // Автоматика и телемеханика. 1996. № 6, 7.
147. Шалыто А.А. Программная реализация управляющих автоматов / Шалыто А.А. // Судостроит. пром–ть. Сер. Автоматика и телемеханика. – 1991. Вып.13.
148. Ярошенко А.В. Вопросы использования алгоритмов математического описания технологической взаимозависимости всех систем корабля / А.В. Ярошенко // Судостроение, – 2000. № 2. – С. 26 – 28.
149. Ярошенко А.В. Единое управление разнородными взаимосвязанными процессами в судостроении / А.В. Ярошенко // Судостроение, – 2001. № 5. – С. 30 – 31.
150. Ярошенко А.В. Математическое описание технологической взаимозависимости всех систем и механизмов корабля и алгоритм его практического применения / А.В. Ярошенко // Судостроение, – 2000. № 1. – С. 40 – 42.
151. Ahuja R. K., Magnati T. L., Orlin J. B. Network flows : Theory, Algorithms, and Applications. Prentice–Hall, 1993.
152. Arpnikanondt C. A Platform–Centric UML–/XML–Enhanced HW/SW Codesign Method for the Development of SoC Systems – Georgia Institute of Technology, 2004 – 317 p.
153. Bellman R., Gierts M. On the analytical formalism of theory of fuzzy sets. “Inform.Sci.”, 1973, v.5, N2, p.149 – 156.
154. Bunday B.D. Basic optimization methods. – School of Mathematical Sciences, Univercity of Bradford. – Edward Arnold – London. –1984 – P.126.
155. Chandy K.M., Sauer C.H. Approximate methods for analyzing queueing

network models of computing systems. Compute. Surv., Vol. 10 No 3. (1988), p. 281–317.

156. Chandy and J Misra, "Distributed computation on graphs: Shortest path algorithms," Commun. ACM, vol. 25, no. 11, pp. 833–837, Nov. 1982. ("Распределенные вычисления на графах: алгоритмы кратчайшего пути").
157. Coves C., Crestani D., prunet F., Design and analysis of workflow processes with Petri nets, in Proceedings of IEEE International Conference on Systems, Man an Cybernetics, vol.1, 1998, pp.101–106.
158. Coyle P.F. Thornton A.M. From UML to HDL: a Model Driven Architectural Approach to Hardware–Software Co–Design – Dallas, Southern Methodist University, 2005 – 6 p.
159. Coyle P.F. Thornton A.M. From UML to HDL: a Model Driven Architectural Approach to Hardware–Software Co–Design – Dallas, Southern Methodist University, 2005 – 6 p.
160. Crawley E., Nair R., Rajagopalen B., Sandick H. A Framework for QoS based Routing in the Internet. //JETF RFC 2386. – August 1998.
161. CORFU Engineering Support System http://seg.ee.upatras.gr/corfu/
162. Diagle J.N. Queueing theory for telecommunications. – Addison–Wesley Publishing Company, Inc., 1992.
163. Dahbura A.T., Sabnany K.K., Vyar M.U. Algoritmic generation of protocol conformance tests // AT&T technical J. –1990. – Jan./Feb. – P. 101 – 118.
164. Ford L.R., Jr. and D.R.Fulkerson. Flows in Networks Princeton Univ.Press, Princeton, NJ, 1962.
165. Ford L.R., Jr. and D.R.Fulkerson. Maximal Flow Through a Network. Canadian Journal of Math., 8:399–404, 1056.
166. Fratta L., Gerla M., Kleinrock L., The flow deviation method: an approach to store–and–forward communication network design // Networks. – 1973. – V.3, N 2. – P. 97–133.

167. Gafni M., Bertsekas D. Asymptotic Optimality of Shortest Path Routing Algorithms // IEEE Trans, of Information Theory. – 1987. – N1.
168. Genrich H.I., Lautenbach K., Thiagarajan P.S. Elements of general net theory // Lecture Notes in / Computer Science – 1980. – Vol.84; – p. 21 – 163.
169. Goldberg A.V. Efficient graph algorithms for sequential and parallel computers. PhD thesis, M.I.T., January 1987.
170. Goldberg V.A. Recent Developments in Maximum Flow Algorithms. NEC Research Institute, Inc., 4 Independence Way, Princeton, NJ 08540, Technical Report # 98–045, NEC Research Institute, Inc., April 1997.
171. Ignizio James P. Expert Systems. The Development and Implementation of Rule–Based Expert Systems. The University of Huston McGraw–Hill, Inc. 1991.
172. Jeong–Ky Min. Perfomance of end–to–end flow control in LAN/WAN interconnections / Jeong–Ky Min // Computer communications, vol. 17 . #5, 1994. – pp. 348–362.
173. Krukov V.A. GRAPHIT–graphic integrated environment for real–time system development / Krukov V.A., Petrenko A.K., Trunov Yu.V., Fedorov K.B. // Тезисы межд. конференции "Real–Time Data – RTD– 94", Дубна, июнь 1994.
174. Mehdi R.Zargham. Computer Architecture: single and parallel systems. : Prentice–Hall International, Inc. 1996. – 472 p.
175. Mellor S. et al. Executable UML: A Foundation for Model Driven Architecture. MA : Addison–Wesley, 2002. — P. 258.
176. Min, J. Flow and congestion control of interconnected networks PhD dissertation, Polytechnic University, Brooklyn, NY. 1991.
177. Panjaitan Seno, Frey Georg. Functional design for Distributed Control Systems using UML Activity Diagrams. International onference on Instrumentations, Communications and Information Technology ICICI 2005, Bandung, Indonesia.

178. "Analysis of routing table update activity after resource recovery in a distributed computer network," in Proc. Seventeenth Hawaii Int. Conf. Syst. Sci., Honolulu, HI, 1984, pp. 96–102.
179. Lougheed and Y. Rekhter, "Border gateway protocol 3 (BGP–3)," RFC 1267, SRI Int., Menlo Park, CA, Oct. 1991. 88. Merlin and A. Segall, "A failsafe distributed routing protocol," IEEE Trans. Commun., vol. COM–27, no. 9, pp. 1280–1288, Sept. 1979.
180. J.McQuillan, "Adaptive routing algorithms for distributed computer networks," BBN Rep. 2831, Bolt Beranek and Newman Inc., Cambridge, MA, May 1974.
181. J.McQuillan et al., "The new routing algorithm for the ARRANET," IEEE Trans. Commun., vol. COM–28, May 1980.
182. Murata T. Petri Nets and their applications. // Soc. Instrum. And Contr. – Vol. 22 – №3; – pp. 265–273.
183. Murata T. Petri Nets: Properties, analysis and applications, in: Proceedings of the IEEE. –Vol. 77 1989. – pp. 541–580.
184. Ross K.W. Optimal circuit access policies in an ISDN environment. A Markov decision approach / Ross K.W., Tsang D.H.K. //IEEE Trans. on commun. – 1999. – Vol.37 №9. –P. 934–939.
185. Schwartz, Telecommunication Networks: Protocols, Modeling and Analysis. Menlo Patk, C A, Addison–Wesley, 1986, ch. 6.
186. Segall, "Distributed network protocols," IEEE Trans. Inform. Theory, vol. IT–29, no. 1, pp. 23–35, Jan. 1983.
187. Seeger and A. Khanna, "Reducing routing overhead in a growing DDN," in Proc. MILCOM'86, Monterey, CA, Oct. 1986, pp. 15.3.1–15.3.13.
188. G. Shin and M. Chen, "Performance analysis of distributed routing strategies free of ping–pong–type looping," IEEE Trans. Comput., vol. COMP–36, no. 2, pp. 129–137, Feb. 1987.

189. Zaumen and J. J. Garcia–Luna–Aceves, "Dynamics of distributed shortest–path routing algorithms," ACM. Comput. Commmun. Rev., vol.21, no. 4, pp. 31–42, Sept. 1991.
190. http://www.citforum.ru/nets/optimize/locnop_09.shtml