ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

КАФЕДРА ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН

КВАЛІФІКАЦІЙНА РОБОТА НА ТЕМУ:
«МЕТОД ВИБІРКОВОЇ ОБРОБКИ ДАНИХ В ДЕГРАДУЮЧИХ ПОЛІНГОВИХ МЕРЕЖАХ»

Виконав:
ст. гр. СПм-21-1
Головін В.Д.

Керівник:
доц. Ткачов В.М.

МЕТА ТА ЗАДАЧІ РОБОТИ

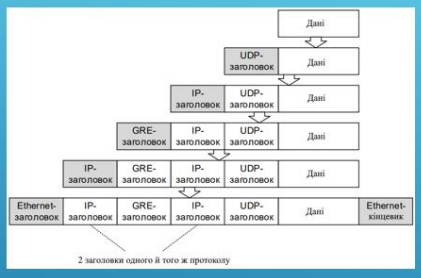
Метою кваліфікаційної роботи є розробка методу гарантованої доставки даних в деградуючих полінгових мережах шляхом часткової її обробки на вузлах такої мережі для зменшення надмірності інтенсивності потоків. Досягнення мети можливе шляхом розробки нового методу вибіркової обробки даних в зазначених мережах.

Основні задачі роботи:

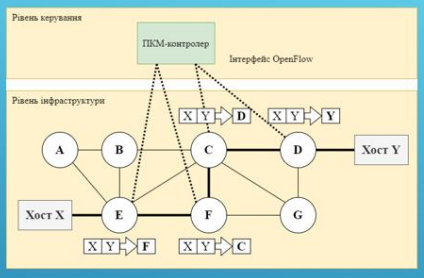
- аналіз принципів побудови та функціонування деградуючих полінгових мереж;
- аналіз основних методів аналізу деградуючих полінгових мереж;
- розробка та дослідження методу вибіркової обробки даних в деградуючих полінгових мережах.

2

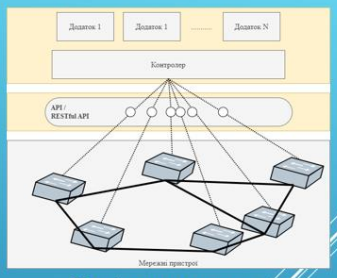
Основні принципи побудови сучасних полінгових мереж



Інкапсуляція пакетів при тунелюванні



Переміщення даних в полінгових мережах

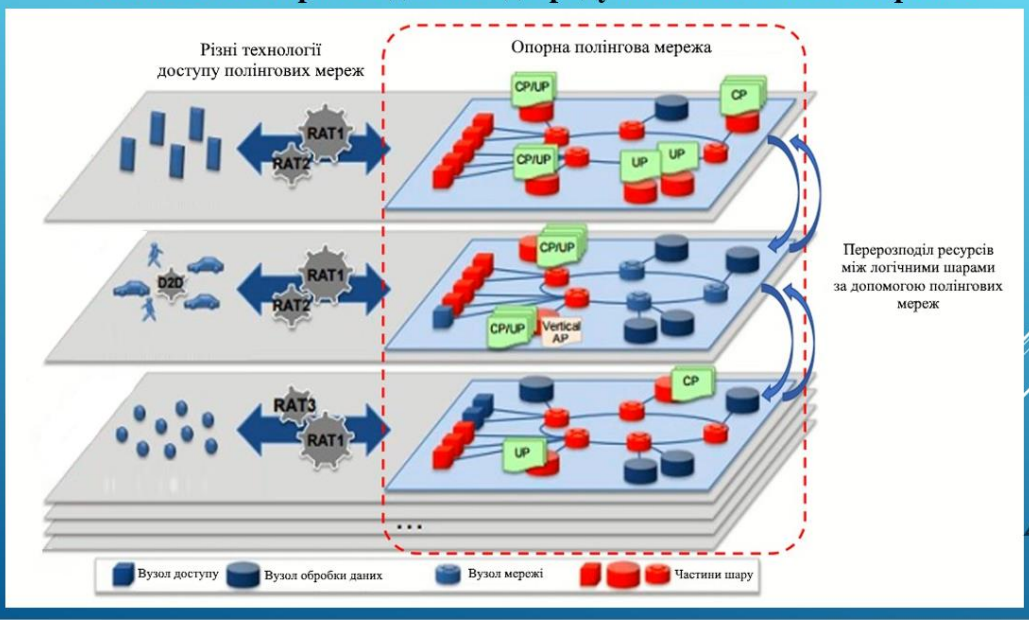


Полінгова мережа, створена на основі існуючих API

Полінгова мережа - розподілена мережа, що самоорганізується та складається із безлічі датчиків (вузлів) і виконуючих пристроїв, об'єднаних між собою за допомогою різних каналів зв'язку. Область покриття подібної мережі може становити від декількох метрів до декількох кілометрів за рахунок здатності ретрансляції повідомлень від одного елемента до іншого.

3

Особливості обробки даних в деградуючих полінгових мережах



4

Використання контурного методу при побудові деградуючих полінгових мереж

$\lambda^{\bar{i}} = C_i^{\bar{i}} \lambda^i$ - зв'язок між контурними інтенсивностями примітивної і досліджуваної деградуюих полінгових мереж

$t_{ji} = t_{\bar{j}\bar{i}} C_j^{\bar{j}} C_i^{\bar{i}'}$ - матриця тривалостей обслуговування деградуючої полінгової мережі

$\lambda^i = (t_{ji})^{-1} \rho_j$ - значення контурних інтенсивностей

$\lambda^{\bar{i}}_{\text{гілок}} = C_i^{\bar{i}} \lambda^i$ - значення потоків у кожній гілці деградуючої полігової мережі

$\lambda^{\bar{i}}_{\text{гілок–загальне}} = C_i^{\bar{i}} \sum_{q=1}^{N} \sum_{p=1}^{K} \lambda^i_{qp}$ - результуюча система рівнянь для визначення потоків мереж у кожній гілці

5

Використання вузлового методу при побудові деградуючих полінгових мереж

$\lambda^{\bar{i}} = A_i^{\bar{i}} \lambda^i$ - матриця вузлових інтенсивностей надходження потоків даних

$\rho_{\bar{j}} = A_j^{\bar{j}} \rho_j$ - зв'язок вузлових завантажень примітивної мережі з досліджуваною деградуючою полінговою мережею

$\mu_{ij} = \mu_{\bar{i}\bar{j}} A_j^{\bar{j}} A_i^{\bar{i}'}$ - результат простих матричних перетворень

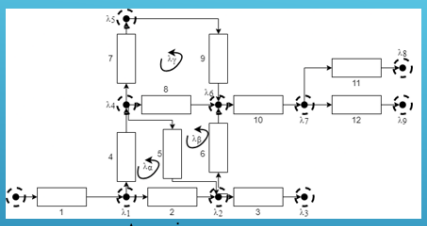
$\rho_j = (\mu_{ij})^{-1} \lambda^i$ - значення вузлових навантажень досліджуваної деградуючої полінгової мережі

$\rho_{\text{гілок}} = A_i^{\bar{i}} \rho_i$ - значення гілкових навантажень досліджуваної деградуючої полінгової мережі

$\rho_{\text{гілок–загальне}} = A_i^{\bar{i}} \sum_{q=1}^{N} \sum_{p=1}^{K} \rho_{qp}$ - результуюча система рівнянь для визначення потоків мереж у кожній гілці

6

Використання ортогонального методу при побудові деградуючих полінгових мереж



Аналізована мережа

$\rho_j = (\mu^{ji})^{-1}\lambda^i$ - значення вузлових навантажень досліджуваної деградуючої полінгової мережі

$\rho_{\text{гілок}} = X_j^j \rho_j$ - значення гілкових навантажень досліджуваної деградуючої полінгової мережі

$\rho_{\text{гілок}-\text{загальне}} = X_j^j \sum_{q=1}^{N}\sum_{p=1}^{K}\rho_{qp}$ - результуюча система рівнянь для визначення потоків мереж у кожній гілці

Вузлові інтенсивності

$$\begin{cases} \lambda_1 = \bar{\lambda}_1 - \bar{\lambda}_2 - \bar{\lambda}_4 \\ \lambda_2 = \bar{\lambda}_2 - \bar{\lambda}_3 + \bar{\lambda}_5 - \bar{\lambda}_6 \\ \lambda_3 = \bar{\lambda}_3 \\ \lambda_4 = \bar{\lambda}_4 - \bar{\lambda}_5 - \bar{\lambda}_7 - \bar{\lambda}_8 \\ \lambda_5 = \bar{\lambda}_7 - \bar{\lambda}_9 \\ \lambda_6 = \bar{\lambda}_6 + \bar{\lambda}_8 + \bar{\lambda}_9 - \bar{\lambda}_{10} \\ \lambda_7 = \bar{\lambda}_{10} - \bar{\lambda}_{11} - \bar{\lambda}_{12} \\ \lambda_8 = \bar{\lambda}_{11} \\ \lambda_9 = \bar{\lambda}_{12} \end{cases}$$

Контурні інтенсивності

$$\begin{cases} \lambda_\alpha = \bar{\lambda}_5 \\ \lambda_\beta = \bar{\lambda}_6 \\ \lambda_\gamma = \bar{\lambda}_7 \end{cases}$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \\ \lambda_9 \\ \lambda_\alpha \\ \lambda_\beta \\ \lambda_\gamma \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \bar{\lambda}_1 \\ \bar{\lambda}_2 \\ \bar{\lambda}_3 \\ \bar{\lambda}_4 \\ \bar{\lambda}_5 \\ \bar{\lambda}_6 \\ \bar{\lambda}_7 \\ \bar{\lambda}_8 \\ \bar{\lambda}_9 \\ \bar{\lambda}_{10} \\ \bar{\lambda}_{11} \\ \bar{\lambda}_{12} \end{bmatrix}$$

7

Особливості деградуючих полінгових мереж при введенні обмеження на якість обслуговування

Представлені вище результуючі системи рівнянь можуть мати нескінченну кількість рішень. Це викликано тим фактом, що маршрутів пересування інформації між потоками може бути безліч, а рішенням систем рівнянь є визначення одного з таких маршрутів.

Основні способи використання цих систем рівнянь:

1. Завдяки тому, що потоки в гілках систем виражені у вигляді лінійно незалежних завантажень, після визначення завантаження/потоків у лінійно незалежних гілках, також автоматично будуть визначені завантаження/потоки в усіх інших.
2. Замість підбора значень лінійно незалежних компонент, можна знайти будь-яке рішення, що буде описувати можливі маршрути між потоками. Але в такому варіанті необхідним для вирішення системи рівнянь являється використання обмежень, які представляють собою системи нерівностей.

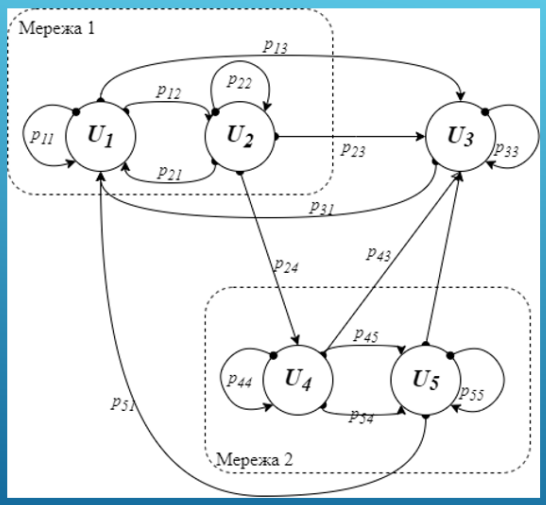
Обов'язкові обмеження для вирішення систем рівнянь другим способом

1. Величина сумарного потоку в каналі зв'язку завжди повинна бути менша ніж величина пропускної здатності каналу.
2. Необхідно забезпечити невід'ємність потоків, які створені каналом q-ї мережі.
3. Необхідно запобігти появу петльових маршрутів.

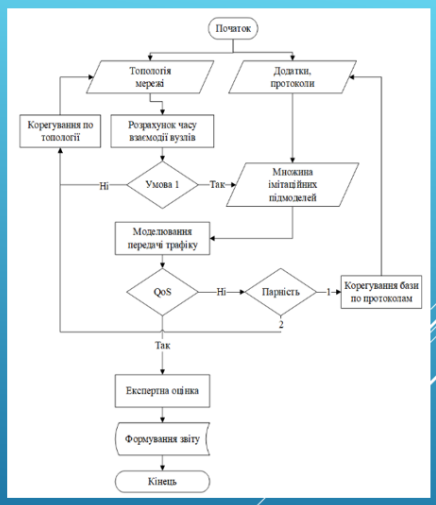
8

Як додаткові обмеження для системи рівнянь можна виділити обмеження наскрізної затримки та ймовірності втрат даних у віртуальній мережі.

Розробка методу вибіркової обробки даних в деградуючих полінгових мережах



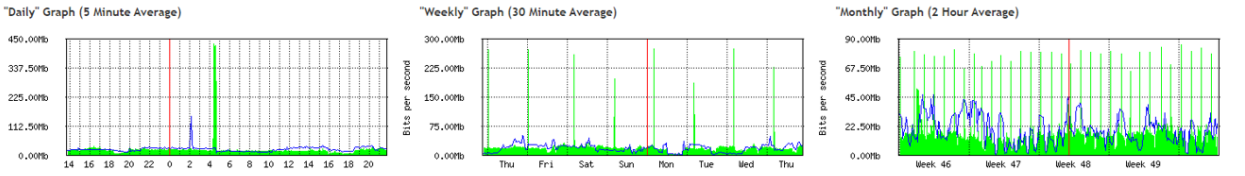
Алгоритм подій зміни каналу між мережами при деградації вузлів



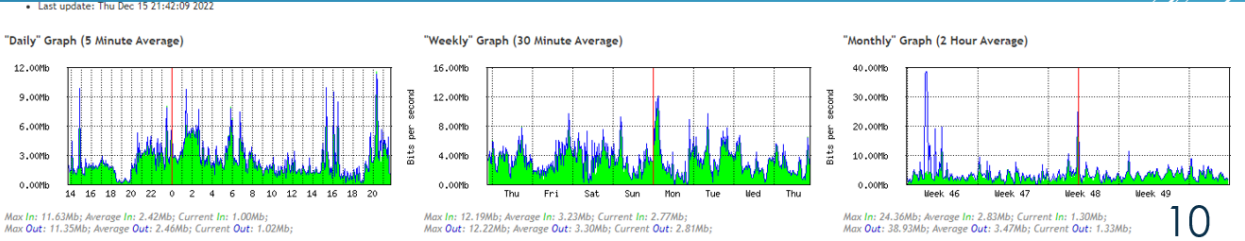
Методу вибіркової обробки даних в деградуючих полінгових мережах

9

Дослідження методу вибіркової обробки даних в деградуючих полінгових мережах



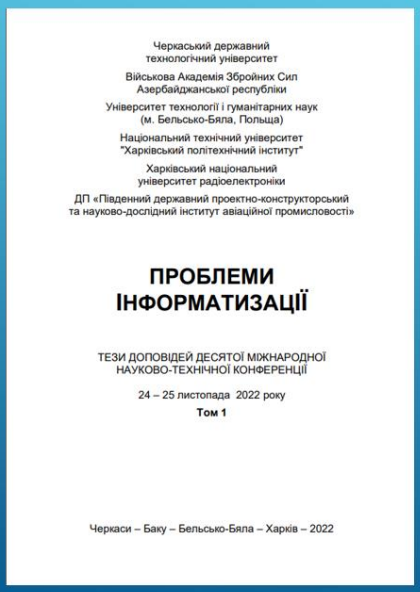
Інтенсивність використання резервного маршруту при передачі даних без використанням методу вибіркової обробки даних в деградуючих полінгових мережах



10

Інтенсивність використання резервного маршруту при передачі даних з використанням методу вибіркової обробки даних в деградуючих полінгових мережах

Апробація результатів роботи



Черкаський державний
технологічний університет
Військова Академія Збройних Сил
Азербайджанської республіки
Університет технології і гуманітарних наук
(м. Бельсько-Бяла, Польща)
Національний технічний університет
"Харківський політехнічний інститут"
Харківський національний
університет радіоелектроніки
ДП «Південний державний проектно-конструкторський
та науково-дослідний інститут авіаційної промисловості»

ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ

ТЕЗИ ДОПОВІДЕЙ ДЕСЯТОЇ МІЖНАРОДНОЇ
НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

24 – 25 листопада 2022 року
Том 1

Черкаси – Баку – Бельсько-Бяла – Харків – 2022

Ткачов В.М. Метод вибіркової обробки даних в деградуючих полінгових мережах / В.М. Ткачов, М.А. Гунько, В.Д. Головін // Зб. тез. доповідей X Міжнародної науково-технічної конференції "Проблеми інформатизації". Т. 1. - Черкаси - Баку - Бельсько-Бяла - Харків, 24-25.11.2022. - С. 33.

11

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розібрано принципи побудови та функціонування деградуючих полінгових мереж, досліджені основні методи аналізу мереж, а саме: контурний, вузловий та ортогональний методи, розроблено та досліджено методу вибіркової обробки даних в деградуючих полінгових мережах.

Експериментальні дослідження проведено на базі датацентру Харківського національного університету радіоелектроніки.

12

ДОДАТОК Б

Фрагмент коду програмного забезпечення для аналізу трафіку мережі

```
# This part of program implements parser
# that intends to gather the traffic from
# network and prepare it for the future
# leverage in order to analyze network.
import socket
import struct
import textwrap

TAB_1 = "\t - "
TAB_2 = "\t\t - "
TAB_3 = "\t\t\t - "
TAB_4 = "\t\t\t\t - "

DATA_TAB_1 = "\t "
DATA_TAB_2 = "\t\t "
DATA_TAB_3 = "\t\t\t "
DATA_TAB_4 = "\t\t\t\t "

# Utils

def get_mac_addr(bytes_addr):
    bytes_addr = map("{:02x}".format, bytes_addr)
    return ":".join(bytes_addr).upper()

def ipv4(addr):
    return ".".join(map(str, addr))

def format_multi_line(prefix, string, size=80):
    size -= len(prefix)
    if isinstance(string, bytes):
        string = "".join(r"\x{:02x}".format(byte) for byte in string)
        if size % 2:
            size -= 1
    return "\n".join([prefix + line for line in textwrap.wrap(string, size)])

# Protocol data parsers
```

```
def ethernet_frame(data):
    dest_mac, src_mac, protocol = struct.unpack("! 6s 6s H", data[:14])
    return (
        get_mac_addr(dest_mac),
        get_mac_addr(src_mac),
        socket.htons(protocol),
        data[14:],
    )

def ipv4_segment(raw_data):
    version_header_length = raw_data[0]
    version = version_header_length >> 4
    header_length = (version_header_length & 15) * 4
    ttl, protocol, src, target = struct.unpack("! 8x B B 2x 4s 4s",
raw_data[:20])
    data = raw_data[header_length:]
    return version, header_length, ttl, protocol, src, ipv4(target), data

def icmp_segment(data):
    icmp_type, code, checksum = struct.unpack("! B B H", data[:4])
    return icmp_type, code, checksum, data[4:]

def tcp_segment(raw_data):
    (src_port, dest_port, sequence, acknowledgment, offset_reserved_flags) =
\
        struct.unpack("! H H L L H", raw_data[:14])

    offset = (offset_reserved_flags >> 12) * 4
    flag_urg = (offset_reserved_flags & 32) >> 5
    flag_ack = (offset_reserved_flags & 16) >> 4
    flag_psh = (offset_reserved_flags & 8) >> 3
    flag_rst = (offset_reserved_flags & 4) >> 2
    flag_syn = (offset_reserved_flags & 2) >> 1
    flag_fin = offset_reserved_flags & 1
    data = raw_data[offset:]
    return (
        src_port, dest_port, sequence, acknowledgment, flag_urg,
        flag_ack, flag_psh, flag_rst, flag_syn, flag_fin, data,
    )

def udp_segment(data):
    src_port, dest_port, size = struct.unpack("! H H 2x H", data[:8])
    return src_port, dest_port, size, data[8:]
```

```
def main():
    connection = socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
socket.ntohs(3))

    while True:
        raw_data, _ = connection.recvfrom(65535)
        dest_mac, src_mac, eth_protocol, eth_data = ethernet_frame(raw_data)

        print("\nEthernet Frame:")
        print(f"Destination: {dest_mac}, Source: {src_mac}, Protocol:
{eth_protocol}")

        # Protocol == IPv4
        if eth_protocol == 8:
            version, header_length, ttl, protocol, src, target, data =
ipv4_segment(eth_data)
            print(f"{TAB_1}IPv4 Packet:")
            print(f"{TAB_2}Version: {version}, Header Length:
{header_length}, TTL:{ttl},")
            print(f"{TAB_2}Protocol: {protocol}, Source: {src},
Target:{target}")

            # Protocol == ICMP
            if protocol == 1:
                icmp_type, code, checksum, icmp_data = icmp_segment(eth_data)
                print(f"{TAB_1}ICMP Packet:")
                print(f"{TAB_2}Type: {icmp_type}, Code: {code},
Checksum:{checksum}")
                print(f"{TAB_2}ICMP Data:")
                print(format_multi_line(DATA_TAB_3, icmp_data))

            # Protocol == TCP
            elif protocol == 6:
                (
                    src_port, dest_port, sequence, acknowledgment, flag_urg,
                    flag_ack, flag_psh, flag_rst, flag_syn, flag_fin,
tcp_data,
                ) = tcp_segment(eth_data)
                print(f"{TAB_1}TCP Segment:")
                print(f"{TAB_2}Source Port: {src_port}, Destination Port:
{dest_port}")
```

```
                print(f"{TAB_2}Sequence: {sequence}, Acknowledgment:
{acknowledgment}")
                print(f"{TAB_2}Flags:")
                print(f"{TAB_3}URG: {flag_urg}, ACK: {flag_ack},
PSH:{flag_psh}")
                print(f"{TAB_3}RST: {flag_rst}, SYN: {flag_syn},
FIN:{flag_fin}")
                print(f"{TAB_2}Data:")
                print(format_multi_line(DATA_TAB_3, tcp_data))

            elif protocol == 17:
                src_port, dest_port, length, udp_data = udp_segment(eth_data)
                print(f"{TAB_1}UDP Segment:")
                print(f"{TAB_2}Source Port: {src_port}, Destination Port:
{dest_port}, Length:{length}")
                print(f"{TAB_2}Data:")
                print(format_multi_line(DATA_TAB_3, udp_data))

main()
```