ДОДАТОК А

Програмні коди

```
#pragma pack(1)
/ Заголовок мережного пакета
struct TNetPacket
{
TNetPacket() {
memset(this, 0x0, sizeof(TNetPacket));
}
BYTE ndir; // номер директиви
BYTE abtk; // ознаки
BYTE nkip; // номер НИП
BYTE nktc; // номер КТС
WORD nka; // номер КА
BYTE nvtk; // номер витка
BYTE ncc; // номер сеансу
WORD nform; // вид інформації
WORD datn; // дата початку
DWORD vrn; // час початку
WORD datk; // дата кінця
DWORD vrk; // час кінця
WORD ds; // довжина повідомлення
BYTE kvit; // код квитанції
/ Повертає значення номера КА в десятковому виді
WORD GetKa() const
{ return (nka & 0x) * 1
+ ((nka & 0x0) >> 4) * 10
+ ((nka & 0x00) >> 8) * 100
+ ((nka & 0x000) >> 12) * 1000;
}/
Установлює значення
номера КА void SetKa(WORD ka) {
this->nka = ((ka / 1000) << 12)
+ (((ka % 1000)/100) << 8)
+ (((ka % 100)/10) << 4)
+ (ka % 10);
}
};
#pragma pack()
```

При описі структури використаються наступні типи даних:

94

BYTE = unsigned char (1 byte) ,
WORD = unsigned short (2 bytes)
DWORD = unsigned int (4 bytes)

Для читання/встановлення значення поля nka використовуються методи GetKa / SetKa

Для запиту параметрів ТМІ в СОТМ у транспортному заголовку повинні бути заповнені наступні поля:

НомерДирективи = 158

ВиглядІнформації = 1888
Номер
ДовжинаІнформації.
1. НомерДирективи = 158
2. ВиглядІнформації = 1888
3. Номер
4. КодКвитанції (0x02 – у випадку коректності транспортного заголовку)
dd.MM.yyyy hh:mm:ss, де
dd - день місяця з лідируючим нулем (01 до 31)
M - номер місяця з лідируючим нулем (01 до 12)
yyyy - рік у вигляді чотиризначного числа
hh - годинники в 24 форматі з лідируючим нулем (00 до 23)
mm - хвилини з лідируючим нулем (00 до 59)
ss - секунди з лідируючим нулем (00 до 59)
Загальний вид XML повідомлення
95

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="...">
<Ka>...</Ka>
<Nip>...</Nip>
<Kts>...</Kts>
<Params ValueType="..." StartTime="..." EndTime="..."
Interval="...">
<Item Index="...">
<Value State="..." Time="...">...</Value>
...
</Item>
...
</Params>
<ViItems ValueType="..." ViCount="..." ViBadCount="...">
<ViStatus DateTime="..." Name="..." Length="..." Validity="..." />
...
</ViItems>
<Error Code="...">...</Error>
</SotmDialog>
```

Докладний опис елементів XML повідомлення наведено в розділі 4.
1.Опис елементів повідомлення
<?xml version="1.0" encoding="utf-8"?>
Стандартний заголовок повідомлення. Його формат повинен зберігатися без змін.
4.1 SotmDialog [Елемент]
96

```
#include <iostream>
#include <cstring>
#include <cmath>
#include <cstdio>
```

```
#include "audio.h"
#include "compress.h"
#include "mpeg.h"

size_t DecompressMpeg::GetSamples(AudioSample *outBuff, size_t wanted) {
   long remaining = wanted;
   while(remaining > 0) {
      if(_samplesRemaining == 0) { // buffer empty?
         NextFrame();  // Get more data
         if(_samplesRemaining == 0) // End of data?
            return wanted-remaining;
      }
      switch(_channels) {
      case 1: //
         while((_samplesRemaining > 0) && (remaining > 0)) {
            *outBuff++ = *_pcmSamples[0]++;
            _samplesRemaining--;
            remaining--;
         }
         break;
      case 2: // Stereo, copy both channels
         while((_samplesRemaining > 0) && (remaining > 0)) {
            *outBuff++ = *_pcmSamples[0]++; // L
            *outBuff++ = *_pcmSamples[1]++; // R
            _samplesRemaining--;
            remaining -= 2;
         }
         break;
      }
   }
   return wanted-remaining;
}
void DecompressMpeg::NextFrame() {
   if (ParseHeader()) { // If no more headers
      _samplesRemaining = 0;
      return;
   }

   // Initialize for decoding
   _pcmSamples[0] = _sampleStorage[0];
   _pcmSamples[1] = _sampleStorage[1];

   switch(_layer) {
   case 1:
        Layer1Decode();
        MainWnd->fileInfo.mpegLayer = "Layer 1";
   break;
   case 2:
        Layer2Decode();
        MainWnd->fileInfo.mpegLayer = "Layer 2";
   break;
   case 3:
        Layer3Decode();
        MainWnd->fileInfo.mpegLayer = "Layer 3";
```

```
   break;
   }

   // Initialize for reading
   _pcmSamples[0] = _sampleStorage[0];
   _pcmSamples[1] = _sampleStorage[1];
}
//
void DecompressMpeg::FillBuffer() {
   // We ran off the end of the buffer?!
   if (_header > _bufferStorage+sizeof(_bufferStorage)) {
      cerr << "Internal error; buffer exhausted!\n";
      _buffer = _header;
   }
   if (_buffer > _header) { //
      cerr << "Synchronization error;\n";
      _buffer = _header;
   }
   // Avoid frequent small calls to ReadBytes()
   if (_buffer < (_bufferStorage+512)) return;

   int totalBufferSize = sizeof(_bufferStorage);
   int bufferSize = _bufferEnd - _buffer;
   memmove(_bufferStorage,_buffer, bufferSize);
   _header -= _buffer - _bufferStorage;
   _bufferEnd -= _buffer - _bufferStorage;
   _buffer = _bufferStorage;
   _bufferEnd += ReadBytes(_bufferEnd, totalBufferSize - bufferSize);
}
void DecompressMpeg::ResetBits() {
   _bitsRemaining = 8;
}

int masks[] = {0,1,3,7,0xF,0x1F,0x3F,0x7F,0xFF};

long DecompressMpeg::GetBits(int numBits) {
   if(_bitsRemaining == 0) { // If no bits in this byte ...
      _buffer++;           // ... move to the next
      _bitsRemaining = 8;
   }
   if(_bitsRemaining >= numBits) { // Can I fill it from this byte?
      _bitsRemaining -= numBits;
      return (*_buffer>>_bitsRemaining)&masks[numBits];
   }
   // Use up rest of this byte, then recurse to get more bits
   long result = (*_buffer & masks[_bitsRemaining])
               << (numBits - _bitsRemaining);
   numBits -= _bitsRemaining; // I don't need as many bits now
   _bitsRemaining = 8;  // Move to next bit
   _buffer++;
   return result | GetBits(numBits);
}
static short bitRateTable[2][4][16] = { // ID, Layer, Code
{  //
```

```
   {0 }, // Reserved
   // Layer 1
   {0,32,48,56,64,80,96,112,128,144,160,176,192,224,256,0 },
   // Layer 2 and 3 are the same
   {0,8,16,24,32,40,48,56,64,80,96,112,128,144,160,0 },
   {0,8,16,24,32,40,48,56,64,80,96,112,128,144,160,0 },
}, {  // ID bit == 1 for MPEG 1 bitrates
   {0 }, // Reserved
   // Layer 1
   {0,32,64,96,128,160,192,224,256,288,320,352,384,416,448,0 },
   // Layer 2
   {0,32,48,56,64,80,96,112,128,160,192,224,256,320,384,0 },
   // Layer 3
   {0,32,40,48,56,64,80,96,112,128,160,192,224,256,320,0 },
}};

static long samplingRateTable[2][4] = { // ID, Code
   {22050,24000,16000,0},  // rates
   {44100,48000,32000,0}
};

bool DecompressMpeg::ParseHeader() {
   FillBuffer(); // Advance buffer
   if (_bufferEnd - _header < 4)
      return true; // No more frames

   if((_header[0] != 0xFF)||((_header[1] & 0xF0) != 0xF0)) {
      cerr << ".";
      return true; // no more frames
   }
   _id = (_header[1] & 8)>>3; // 2
   _layer = -(_header[1] >> 1) & 3; //
   _protection = (~_header[1] & 1); // 0->CRC, 1->no CRC

   int bitrateIndex = (_header[2] & 0xF0)>>4;
   _bitRate = bitRateTable[_id][_layer][bitrateIndex] * 1000;
   MainWnd->fileInfo.bitRate = _bitRate;
   int samplingRateIndex = (_header[2] & 0x0C)>>2;
   _samplingRate = samplingRateTable[_id][samplingRateIndex];
   MainWnd->fileInfo.samplingRate=_samplingRate;
   _padding = (_header[2] & 0x02)>>1;
   _private = (_header[2] & 0x01);
   _mode = (_header[3] & 0xC0) >> 6;
   _modeExtension = (_header[3] & 0x30)>>4;
   switch(_mode) {
   case 0: // Stereo: all subbands
            _channels = 2;
            _bound = 32;
            MainWnd->fileInfo.mode = "";
            break;
   case 1:
            _channels = 2;
            _bound = (_modeExtension + 1)<<2;
            MainWnd->fileInfo.mode = "
```

```
            break;
   case 2:
            _channels = 2;
            _bound = 32;
            MainWnd->fileInfo.mode = "";
            break;
   case 3:
            _channels = 1;
            _bound = 0;
            MainWnd->fileInfo.mode = "";
            break;
   }
   MainWnd->fileInfo.channels =  _channels;
   _copyright = (_header[3] & 0x08);
   _original = (_header[3] & 0x04);
   _emphasis = (_header[3] & 0x03);

   _buffer = _header+4;
   if (_protection) _buffer += 2; //

if (_bitRate == 0) { //

if (_headerSpacing) { //
   if (_layer == 1) { //
      _header += _headerSpacing * 4;
      if(_padding) _header += 4;
   } else {  // Layers 2 and 3 have 1-byte slots
      _header += _headerSpacing;
      if(_padding) _header += 1;
   }
```

ДОДАТОК Б
Слайди презентації

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження методів ідентифікації та класифікації трафіку для безпеки мереж

Науковий керівник:
проф. Шостак І.В.

Виконав:
ст. гр. ІПЗмзд-18-1 Воропаєв А.В.

1

Мета роботи

- є підвищення рівня захисту й надійності СПД за рахунок поліпшення ідентифікації трафіку в реальному часі на рівні додатків, та в тунелях, з використанням схованої марківської моделі

2

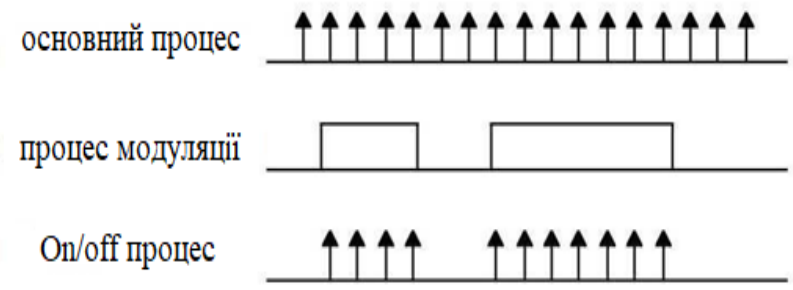
Завдання роботи

- аналіз існуючих моделей трафіку СПД, виявлення їх переваг і недоліків і особливостей організації трафіку стосовно до завдання ідентифікації;
- порівняльний аналіз існуючих методів ідентифікації трафіку СПД і розробка моделі, методу й алгоритмів ідентифікації мережного трафіку в реальному часі з використанням його статистичних характеристик і схованої марківської моделі (СММ);
- розробка алгоритму обчислення значень параметрів пропонованої моделі з використанням ітераційної процедури Баума-Велша;
- розробка алгоритму ініціалізації СММ на основі моделі гаусової суміші, що забезпечує оптимальну збіжність процедури Баума-Велша в рамках необхідного якості ідентифікації;
- розробка методики підготовки наборів даних на основі реального та модельного мережного трафіку на етапах навчання й тестування запропонованої моделі;
- модифікація запропонованих алгоритмів ідентифікації для вирішення завдання ідентифікації тунельних додатків;
- розробка програмних засобів ідентифікації на основі запропонованих алгоритмів і їх експериментальне дослідження

3

Модель “On/Off”

- як модульований випадковий процес



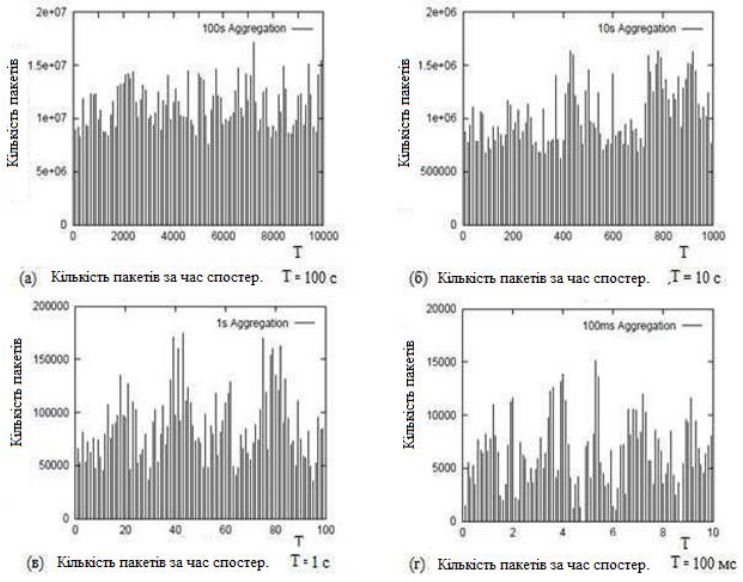
4

Моделі часових рядів

- з авторегресійним ковзним середнім ARMA (Autoregressive moving average)
- з авторегресійним інтегрованим ковзним середнім ARIMA (Autoregressive integrated moving average)

5

Фрактальні моделі



(а) Кількість пакетів за час спостер. T = 100 с
(б) Кількість пакетів за час спостер. T = 10 с
(в) Кількість пакетів за час спостер. T = 1 с
(г) Кількість пакетів за час спостер. T = 100 мс

6

Ланцюг Маркова

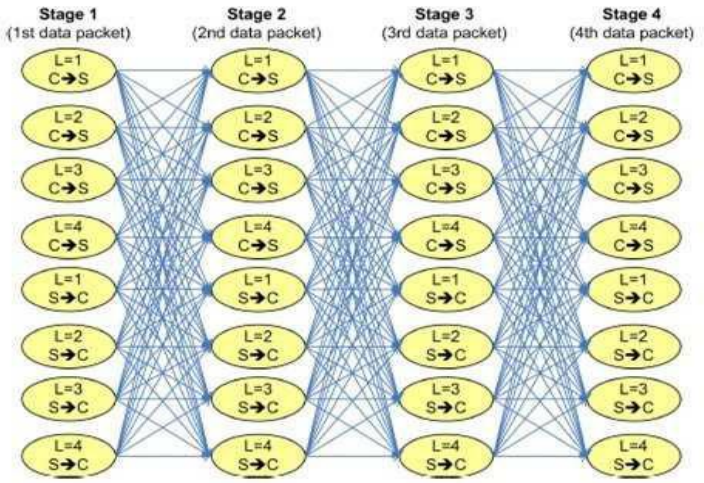
$$a_{ij}(n) = a_{ij}, \quad \forall n \in \mathbb{N}$$

- Марківська модель використовується для моделювання мережного трафіку в тих випадках, коли спостережувана величина (стан) залежить тільки від
- попереднього стану, наприклад, від стану системи (успіх/невдача),
- від стану користувача (активний/неактивний) або
- при моделюванні одержання пакетів у плині деякого інтервалу часу

7

Модель Мунза

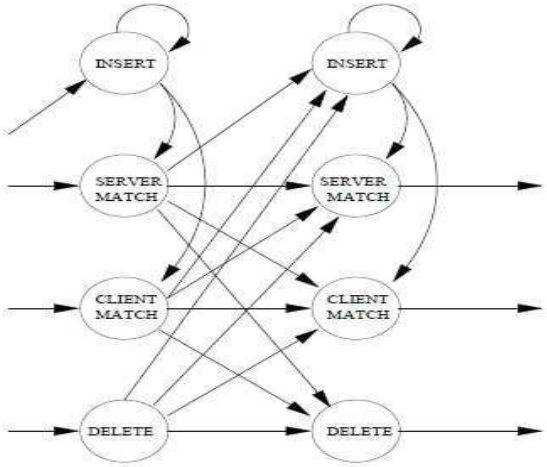
- для ідентифікації мережних додатків на основі марківської моделі



8

Модель Райта

- для ідентифікації мережних додатків на основі CMM



9

Класифікація мережного трафіку

- Класифікація на основі аналізу статистичних властивостей пакетів є діючим методом, тому що вирішує завдання ідентифікації трафіку декількох типів додатків з високою точністю.

- Обмеженням може служити висока обчислювальна складність, однак спільне його використання зі CMM дозволить одержати ефективний інструмент ідентифікації,

  що має високу точність та продуктивність

10

Постановка завдання

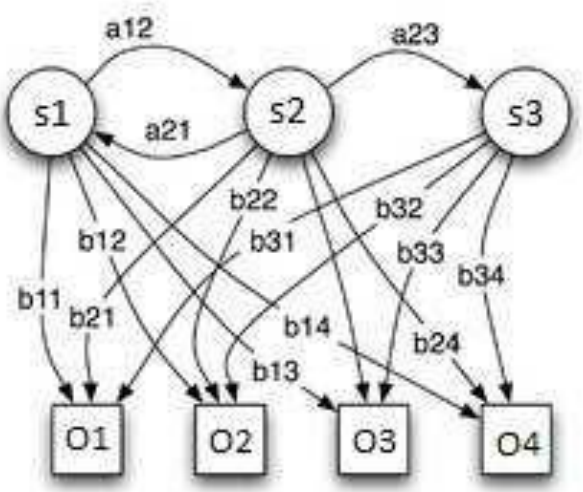
Під класифікацією мережного трафіку розуміється процес поділу згенерованих додатками мережних потоків на групи (класи)

Класифікація на основі типу додатка потрібна, щоб забезпечити виконання політик безпеки використання ресурсів:

- рівень додатка: метою даного виду класифікації є ідентифікація певних додатків, наприклад, Skype, WhatsApp, Google Maps та ін.;
- рівень протоколу: даний вид класифікації необхідний для віднесення трафіку до відомого протоколу, наприклад, HTTP, SMTP, FTP та ін.;
- рівень вмісту (контенту) прикладного рівня: класифікація проводиться на основі типу даних, які передаються по мережі, наприклад, текст, зображення, бінарні дані, відео контент, зашифровані дані та ін.;
- рівень характеру мережного трафіку: класифікація ґрунтується на певному характері трафіку, який визначається методом передачі даних між мережними обладнаннями, наприклад, інтерактивний трафік, що пульсує, потоковий і т.ін.

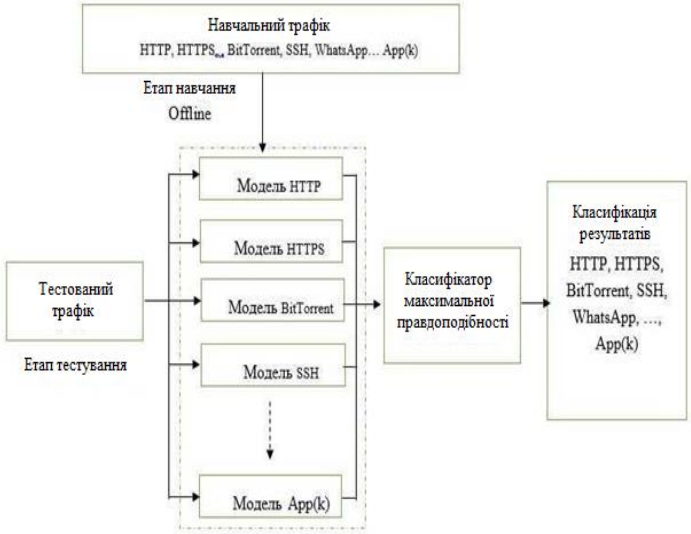
11

Приклад схованої марківської моделі



12

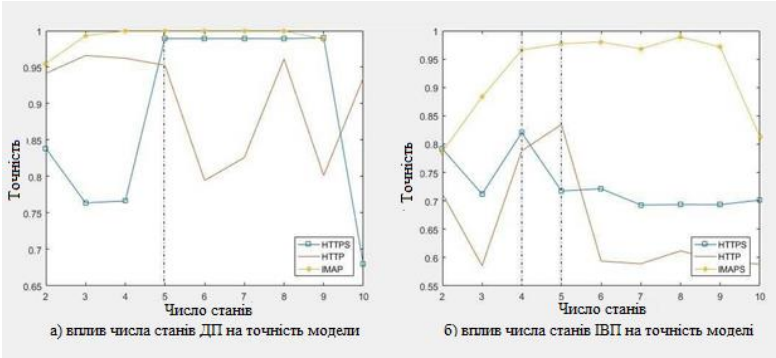
Визначення параметрів спостереження



Модель ідентифікації мережних додатків

13

Вплив числа станів моделі на точність ідентифікації додатків



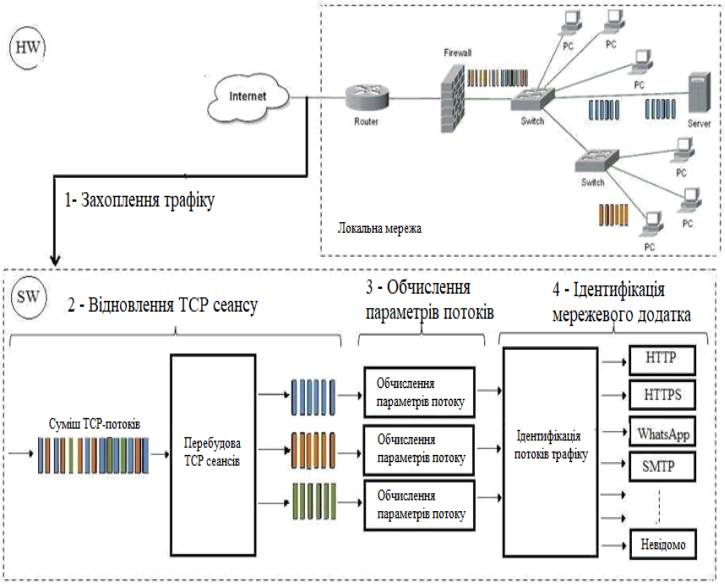
а) вплив числа станів ДП на точність моделі

б) вплив числа станів ІВП на точність моделі

14

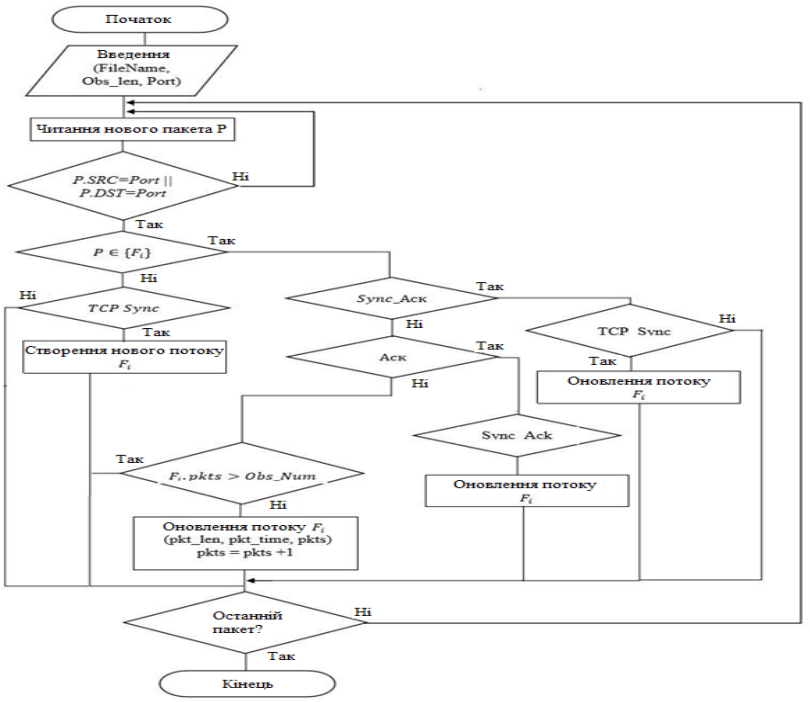
Алгоритм підготовки трафіку для ідентифікації

– захоплення трафіку;
– відновлення потоків;
– обчислення параметрів потоків.



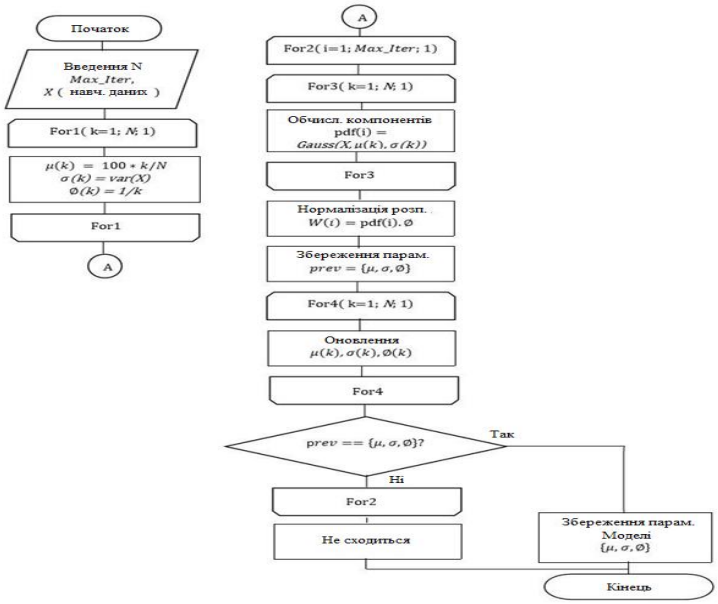
15

Схема алгоритму поділу потоків мережного трафіку



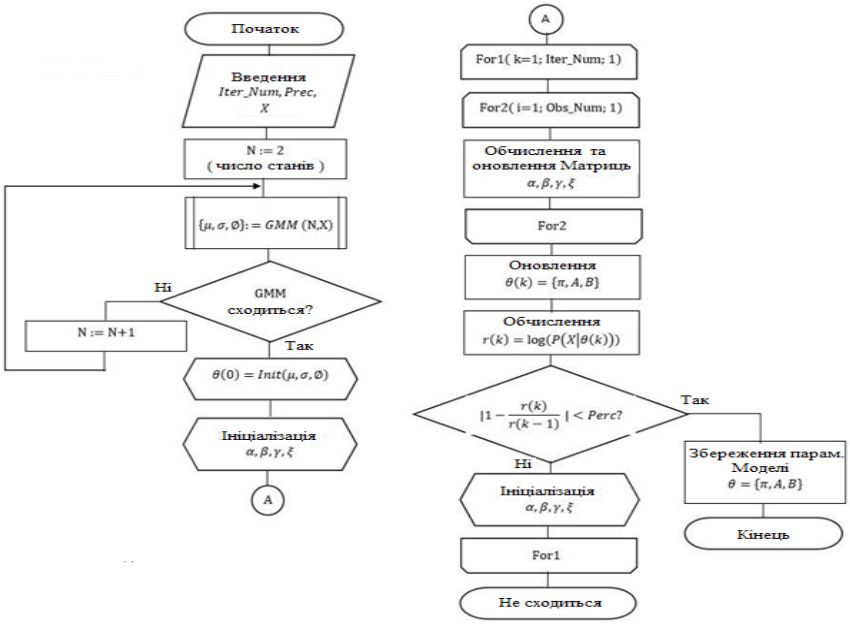
16

Схема алгоритму ініціалізації значень параметрів моделі на основі моделі гауссової суміші



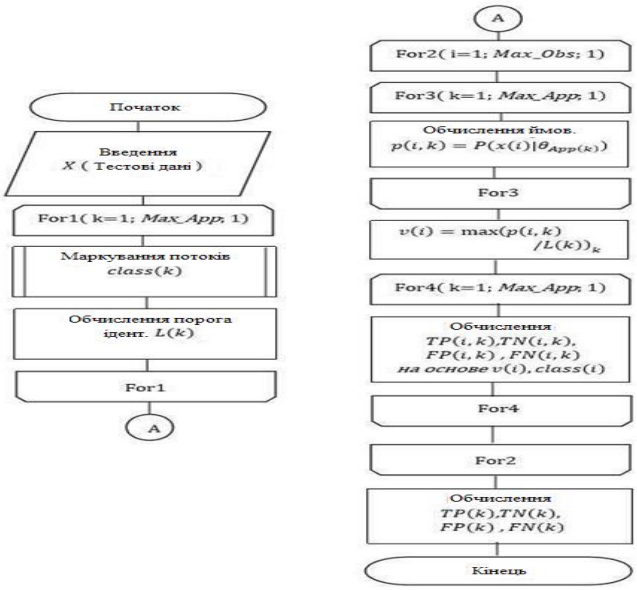
17

Схема алгоритму обчислення параметрів моделі ідентифікації на основі процедури Баума-Велша



18

Схема алгоритму оцінки якості моделі ідентифікації мережного трафіку



19

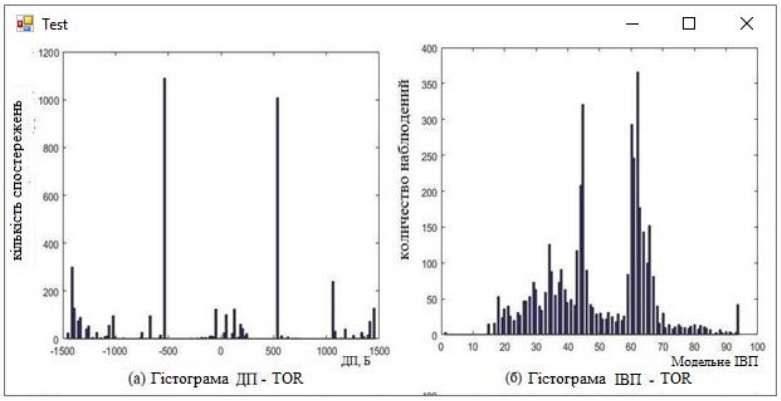
Модель ідентифікації

обрано шість додатків для тестування моделі ідентифікації мережного трафіку:

- веб-додатки HTTP
- безпечні веб-додатки HTTPS.
- безпечна електронна пошта IMAPS.
- однорангові додатки P2P.
- чат-додаток WhatsApp.
- тунельний додаток TOR

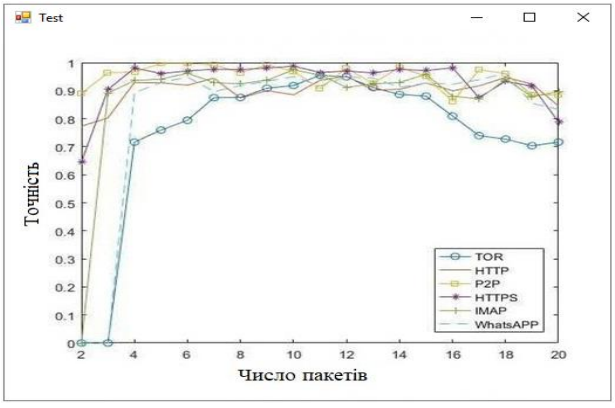
20

Гістограми спостережень однорангового додатка TOR



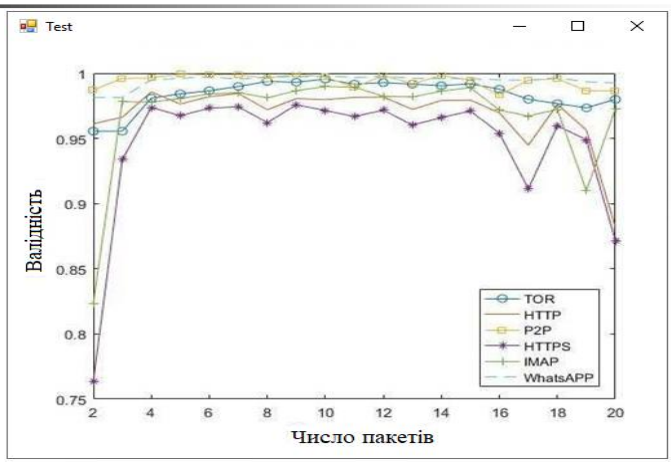
21

Вплив числа пакетів на точність ідентифікації додатків у моделі



22

Екранна форма «Вплив числа пакетів на валідність ідентифікації додатків у моделі»



23

Екранна форма «Точність ідентифікації досліджуваних додатків»

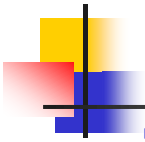
Test

	Вихідний класифікатор (відсотки)						
Протокол	**TOR**	**HTTP**	**P2P**	**HTTPS**	**IMAPS**	**WhatsApp**	**None**
TOR	99.33	0.67	0.00	0.00	0.00	0.00	0.00
HTTP	**1.60**	**91.53**	**2.29**	**5.49**	**0.00**	**0.92**	**0.00**
P2P	0.00	0.00	99.71	0.29	0.00	0.00	0.00
HTTPS	**0.00**	**3.38**	**0.00**	**93.38**	**3.24**	**0.00**	**0.00**
IMAPS	1.86	0.34	0.34	0.68	97.13	0.00	0.00
WhatsApp	3.23	2.68	0.00	4.45	0.00	90.32	0.00

Проведений порівняльний аналіз із існуючими розробками, показав перевагу розроблених алгоритмів і можливість їх ефективного використання при ідентифікації трафіку СПД на рівні додатків в умовах реального часу

24

Висновки



- Проаналізовано метод ідентифікації трафіку, що задовольняє сформульованим вимогам, що підтвердив можливість і доцільність створення нового інструмента рішення завдання ідентифікації на основі класифікації трафіку з використанням статистичного аналізу, СММ і ітераційної процедури Баума-Велша для обчислення її параметрів.
- Розроблений алгоритм ініціалізації СММ із використанням моделі гауссової суміші, що забезпечує оптимальну збіжність процедури Баума-Велша в рамках необхідного якості ідентифікації.
- Застосування цього алгоритму в загальному алгоритмі ідентифікації дозволило виконати її в реальному часі з точністю більш 90%, повнотою більш 80%, загальної валідності більш 95% і часток помилок менш 5% при обмеженим числі пакетів (10 пакетів).
- Розроблено тестове ПЗ підготовки наборів даних на основі реального й модельного мережного трафіку на етапах навчання й тестування запропонованої моделі, що складає основу її функціонування й експериментального дослідження.

ДОДАТОК В

Апробація результатів роботи

Подано тези на Міжнародний молодіжний форум

ДОСЛІДЖЕННЯ МЕТОДІВ ІДЕНТИФІКАЦІЇ ТА КЛАСИФІКАЦІЇ ТРАФІКУ ДЛЯ БЕЗПЕКИ МЕРЕЖ

Воропаєв А.В.
Науковий керівник – проф. Шостак І.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії,
тел.: (057) 702-14-46)
E-mail: i.v.shostak@gmail.com

In the work the method of estimating the maturity (perfection) of the projects performed in the testing of software systems is developed. This method is based on TMM's five-level maturity model. Also, the use of mathematical model, the basic process of testing software data processing systems under the conditions of limited resources, as well as several methods aimed at improving the quality of software products, have been developed and substantiated.

До головних напрямків програмної інженерії відносяться задачі вдосконалення процесів життєвого циклу ПС, зокрема процесу тестування. Неухильне підвищення якості програмних продуктів – основна мета програмної інженерії та предмет турботи розробників ПС.

Відповідь на питання, як підвищити конкурентоспроможність українських програмних продуктів, як знизити ризик проектів, як досягти балансу сторін трикутника «тривалість – вартість – цілі проекту», протягом останніх років намагаються знайти відповідь не лише керівники організацій-розробників ПЗ і менеджери проектів, але і замовники, і споживачі, що використовують програмні продукти низької якості. Наслідки постачання програмних продуктів низької якості – це завжди збитки користувачів: не лише матеріальні і фінансові втрати, але і падіння престижу. Ці проблеми, в свою чергу, негативно відображаються на конкурентоспроможності організації-розробника програмних продуктів. Для забезпечення необхідного рівня якості ПС в міжнародній практиці знаходять застосування два підходи: продукто-орієнтований і процесо-орієнтований. В першому акцент робиться на оцінку якості шляхом тестування готового програмного продукту. Цей підхід базується на припущенні, що чим більше знайдено і усунено дефектів в ПС при тестуванні, тим вище його якість.

Тестування – важливий етап розроблення ПС, оскільки, з одного боку, вимагає значних витрат на проведення, а з іншого – робить великий внесок у їхню якість. Через теоретично доведену неможливість вичерпного тестування та велику потенційну вартість втрат через відмови у ПС, потрібен чітко визначений та ефективний *процес тестування*, базований на ухваленні збалансованих рішень щодо тривалості та вартості тестування для досягнення необхідного рівня довіри до якості ПС.

Методи визначення кількісних критеріїв завершення тестування та керування процесом тестування з використанням кількісних вимірів ще мало використовуються в проектах створення ПС, що призводить до того, що якість та надійність залишаються не передбачуваними. Лише за наявності достовірної та своєчасної інформації щодо стану ПС, видів ризиків і можливих втрат через відмови може бути забезпечене ефективне виконання процесу тестування.

Метою роботи є проведення комплексу досліджень з інженерії тестування ПС оброблення даних, формування ефективної стратегії тестування програмних систем, спрямованої на зниження ризику відмов під час експлуатації. Для цього в роботі розв'язуються наступні задачі:

- дослідження сучасних вимог до процесу тестування ПС;
- аналіз існуючих моделей надійності ПС, розроблення алгоритмів та програм їх реалізації;
- побудова моделі визначення оптимального часу тестування модулів ПС з урахуванням ризиків відмов;
- визначення структури базового процесу, що регламентує всі дії з підготовки, проведення та оцінювання результатів тестування, та розроблення методики виконання процесу тестування ПС оброблення даних;
- проектування та реалізація програмного комплексу підтримки інженерії тестування;
- аналіз сучасних підходів до визначення рівня зрілості процесу тестування ПС та розроблення методики оцінювання процесу тестування;
- впровадження запропонованих підходів, моделей та методик інженерії тестування в проекти з розроблення ПС оброблення даних та опис результатів випробування запропонованих моделей оцінювання оптимального часу тестування та методу оцінювання ризиків відмов програмних модулів.

З метою визначення ефективності впровадження базового процесу тестування був розроблений метод оцінювання зрілості (досконалості) виконуваних у проектах дій з тестування ПС. Цей метод ґрунтується на п'ятирівневій моделі зрілості TMM.

Випробування моделей мають виконуватися в конкретній інформаційно-аналітичної системі підтримки прийняття управлінських рішень, яка має складатися з програмних комплексів, об'єднаних діловим процесом оброблення даних. В ході практичної реалізації аналіз загроз відмов системи показав, що з позицій цілісності інформації найбільший внесок у ризик її відмов робить ПК контролю і введення даних до БД ORACLE, який встановлюється та одночасно функціонує на 10 робочих місцях. Для п'яти модулів цього ПК були виконані оцінки ризику відмов та часу тестування.