Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «Дослідження ефективності використання формальних граматик для формування мережевого трафіку»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ2222: / Андрій СЕРБЕНЮК /

Керівник: / Олена КУРОП'ЯТНИК /

Нормоконтролер: / Світлана ВОЛКОВА /

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент Сербенюк А.В.

Дніпро – 2024 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis

on the topic: «Research of effectiveness of using formal grammar to form network traffic»
according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ2222: / Andrii SERBENIUK /
Scientific Supervisor: / Olena KUROPIATNYK /
Normative controller: / Svitlana VOLKOVA /

Dnipro – 2024

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем

Кафедра: Комп’ютерні інформаційні технології

Рівень вищої освіти: магістр

Освітня програма: Інженерія програмного забезпечення

Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ 20__ р.

ЗАВДАННЯ

На кваліфікаційну роботу магістра
студенту Сербенюку Андрію Васильовичу

1 Тема дипломної роботи: «Дослідження ефективності використання формальних граматик для формування мережевого трафіку»
Керівник роботи: Куроп’ятник Олена Сергіївна
затверджені наказом 1196 ст від 05.12. 2022 року

2 Строк подання студентом роботи 08.01.2024-14.01.2024.

3 Вихідні дані до дипломної роботи:
_____.

4 Зміст пояснювальної записки (перелік питань до розробки): огляд аналогів, розробка граматики, розробка ПЗ, проведення експериментів
5 Перелік демонстраційного матеріалу: презентація; демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Дослідження предметної області	04.09.2023-12.11.2023	
2	Постановка задачі	13.11.2023-26.11.2023	
3	Визначення критеріїв ефективності використання формальних граматик для формування мережевого трафіку	04.12.2023-10.12.2023	Виконання 30% роботи
5	Розробка програми	11.12.2023-17.12.2023	Виконання 60% роботи
7	Проведення експериментів	25.12.2023-31.12.2023	
8	Підготовка звітної документації	01.01.2024-07.01.2024	Виконання 100% роботи

Студент _____ / Андрій СЕРБЕНЮК /

Керівник роботи _____/ Олена КУРОП’ЯТНИК /

РЕФЕРАТ

Об’єктом цього дослідження є використання формальних граматик для задання інтерфейсу представлення мережевого трафіку.

Метою дослідження є визначення ефективності використання формальних граматик для представлення мережевого трафіку.

Методи дослідження: порівняння ергономічних характеристик використання інтерфейсів програм-аналогів, та розробленого інтерфейсу, що використовує розроблену формальну граматику.

Результати та їх новизна: дослідження робить внесок у визначення доцільності використання формальних граматик для представлення мережевого трафіку.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, бібліографічного списку, та двох додатків:

- у вступі описується сутність розробки, її актуальність. Складається із 3 сторінок;
- в першому розділі висвітлено аналіз предметної області представлення довільного мережевого трафіку, проаналізовано програмні аналоги, їх можливості та види інтерфейсів. Складається із 7 сторінок;
- в другому розділі надано обґрунтування експериментального методу дослідження. Складається із 8 сторінок;
- в третьому розділі представлене проєктування та розробка інструментального програмного забезпечення для дослідження. Складається з сторінок;
- в четвертому розділі описано виконані дослідження. Складається з сторінок;
- додатки містять технічне завдання та тези із наукової конференції.

Таблиць — 7, рисунків — 15, бібліографія — 7 джерел.

Ключові слова: мережевий трафік, мережеві протоколи, формальна граматика.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ

TCP — Transmission Control Protocol

HTTP — Hypertext Transfer Protocol

ВСТУП

Актуальність роботи. Мережевий трафік є однією із основ функціонування сучасного світу. Для забезпечення працездатності мережевих систем існує безліч практик, починаючи із розробки та використання мережевих протоколів, закінчуючи тестуванням прохідності трафіку через мережу.

Для тестування працездатності мережевих протоколів необхідний деякий трафік, для якого перевірятиметься відповідність протоколу. Існує безліч протоколів та безліч способів генерації трафіку для нього, однак існуючі способи генерації все ще не завжди здатні відповідати всім вимогам тестування. Здебільшого, подібні інструменти мають деякий сталий набір функціоналу, який важко розширити. Також, в більшості випадків вони використовують графічний інтерфейс користувача, що також значно ускладнює здатність автоматизації генерації трафіку подібним інструментом.

В даній роботі розглядається варіант зняття даних обмежень за допомогою використання мовного інтерфейсу для представлення мережевого трафіку. Такий підхід повинен мати ряд переваг, пов’язаних із легкістю додавання нового функціоналу, підтримки нових видів протоколів, простоти автоматизації генерації трафіку.

Тема роботи: «Дослідження ефективності використання формальних граматик для формування мережевого трафіку».

Об’єктом дослідження є процеси формування мережевого трафіку.

Предметом дослідження є оцінка ефективності використання формальних граматик для формування мережевогот трафіку.

Метою дослідження є визначення доцільності використання формальних граматик для формування мережевого трафіку. Завданнями дослідження є:

- аналіз існуючих аналогів програм формування мережевого трафіку;
- розробка інструменту, який реалізує функціонал формування мережевого трафіку із використанням формальної граматики;

- визначення показників ефективності використання граматик для формування трафіку.

Методи дослідження. Для оцінки показників ефективності представлення трафіку використовується емпіричний метод збору числових значень ергономічних лічильників, що підраховують кількість взаємодій користувача із певними елементами інтерфейсу.

Наукова новизна. Набуло подальшого розвитку використання формальних граматик як способу формування мережевого трафіку, методи формалізації мережевого трафіку засобами формальних граматик. Сформовано критерії оцінки ефективності формування трафіку та отримано їх значення для трафіку виду деякої кількості одноманітних пакетів, та сесії протоколу TCP.

Практична значущість. Використання результатів цього дослідження на практиці може значно полегшити роботу інженерам, які мають справу із ручним формуванням мережевого трафіку. З допомогою подібного підходу можна зробити можливість задання параметрів навіть низькорівневих протоколів. В теорії, мовний інтерфейс здатен бути більш ефективним при дрібних модифікаціях трафіку, повторному використанні розроблених представлень, швидкій заміні представлення на інше. Також, він дозволятиме автоматизовувати виконання програми за рахунок зовнішніх інструментів, що для графічних інтерфейсів робити недоцільно.

Подібний спосіб формування трафіку є неординарним, адже в більшості випадків інструменти штучного формування трафіку мають графічний інтерфейс. Також існує можливість формування трафіку за допомогою бібліотек вже існуючих мов програмування, але це не завжди доцільне рішення з точки доступності використання таких інструментів, або часу розробки.

Оскільки вхідні дані програми представлені у вигляді тексту, їх дуже легко зберігати та модифікувати. Також, при додаванні нового функціоналу, розробка інтерфейсу для нього зводиться до додавання нових типів даних в мову. При цьому, користувач не навантажений ознайомленням із новими елементами

інтерфейсу, натомість, йому просто стають доступними нові значення у вже існуючому інтерфейсі, представленому текстом.

Оскільки додавання нового функціоналу в існуючий програмний каркас настільки спрощене для розробників, є сенс також реалізувати підтримку динамічного розширення функціоналу програми. Це можна зробити, наприклад, системою плагінів, які відвантажуються в програму на ходу та без рекомпіляції.

Апробація. Основні положення магістерської роботи доповідалися на наукових семінарах кафедри комп’ютерних інформаційних технологій УДУНТу 06.10.23, 10.11.23 та XII міжнародній науково-технічній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» 6-7 грудня 2023.

1 АНАЛІЗ ПРОБЛЕМ ФОРМУВАННЯ ТРАФІКУ

Залежно від вимог, формування довільного трафіку може бути складним завданням. В основному це пов’язано із складністю структури трафіку, адже кінцевий сформований пакет представляє собою бінарні дані, складені за декількома протоколами, які накладаються один поверх іншого. В кінцевому пакеті присутні багато полів та частинок інформації від різних протоколів, і забезпечити контроль над формуванням кожного із них може бути проблематично навіть через саму їх кількість.

Звісно існує безліч інструментів які можуть формувати трафік. Втім, в більшості випадків вони спеціалізуються на покритті тільки декількох протоколів та не надають доступ до редагування довільних частин трафіку. Наприклад, якщо інстумент дозволяє формувати HTTP трафік, він може не надавати доступу до частин протоколів нижчих рівнів, таких як TCP. Це може не бути недоліком таких інструментів, адже вони і не створювались для таких цілей, і це ж означає що вони для такого не можуть використовуватись. Тобто, часто інструменти фокусуються на високорівневих протоколах і дозволяють зручно формувати за ними трафік, але не надають доступу до всіх аспектів формування трафіку. Прикладом таких інструментів може слугувати, наприклад, Postman [1].

Часто найдоступнішим способом формування трафіку є використання інструментів, які використовують його для роботи. Наприклад, поштові клієнти користуються поштовими протоколами SMTP, POP3, IMAP, і формують трафік за ними. Або, браузери користуються різними версіями протоколів HTTP та SSL для комунікації із рештою інтернету. Як результат, формується мережевий трафік. Однак, таким чином отримується досить низька ступінь контролю над його формуванням, адже в цьому випадку формування довільного трафіку і не є ціллю інструментів. Але, залежно від вимог, їх може бути достатньо.

Найпростішим прикладом тут може слугувати інструмент ping [2], який формує ICMP трафік.

Звісно існують інструменти, завданням яких є саме формування довільного трафіку. Це досягається за рахунок доступу до формування низькорівневих протоколів рівнів 1-4 моделі OSI. Подібний підхід також дозволяє доступ до формування протоколів вищих рівнів, адже на нижчих рівнях ці протоколи просто є сирими даними. Хоча, протоколи вищих рівнів у такому випадку може бути досить незручно формувати, адже необхідно вручну забезпечувати правильність їх формування. За приклад подібного інструменту візьмемо Ostinato[3].

1.1 Огляд програмних аналогів

Як аналог для порівняння, розглянемо програму Ostinato. Вона дозволяє генерувати трафік за вручну створеними пакетами.

Для задання трафіку, в програмі використовуються «потоки» («stream»). Один потік дозволяє сформувати один пакет. Для деяких полів пакету можна налаштувати змінні значення, щоб при генерації наступного пакету вони змінились. Таким чином, з використанням одного потоку можна генерувати довільну кількість одноманітних пакетів. Потоків може бути безліч, і є можливість налаштувати кількість пакетів, які вони генерують, та переходи між потоками.

При створенні потоку, Ostinato надає можливість модифікації практично всіх полів протоколів рівнів 1-4 моделі OSI, та дещо обмежено для рівня 5, для якого, залежно від протоколу, може автоматично виставлятись порт за замовчуванням.

Генерація трафіку протоколів вищих рівнів є дещо складнішим завданням із Ostinato. Оскільки потік Ostinato по суті являє собою одноманітний пакет, то для представлення сесії протоколів вищих рівнів потрібно вручну створити по одному потоку Ostinato на кожен пакет в сесії.

Ostinato використовує графічний інтерфейс користувача, але також існує API для мови Python. В інтерфейсі користувача кожне окреме поле даних протоколів представлене деяким графічним елементом інтерфейсу, переважно текстовими полями. Інтерфейс створення потоку викладений досить зрозуміло, але, через велику кількість параметрів для конфігурації протоколів, він все одно досить сильно забитий (рис.1.1-1.4).

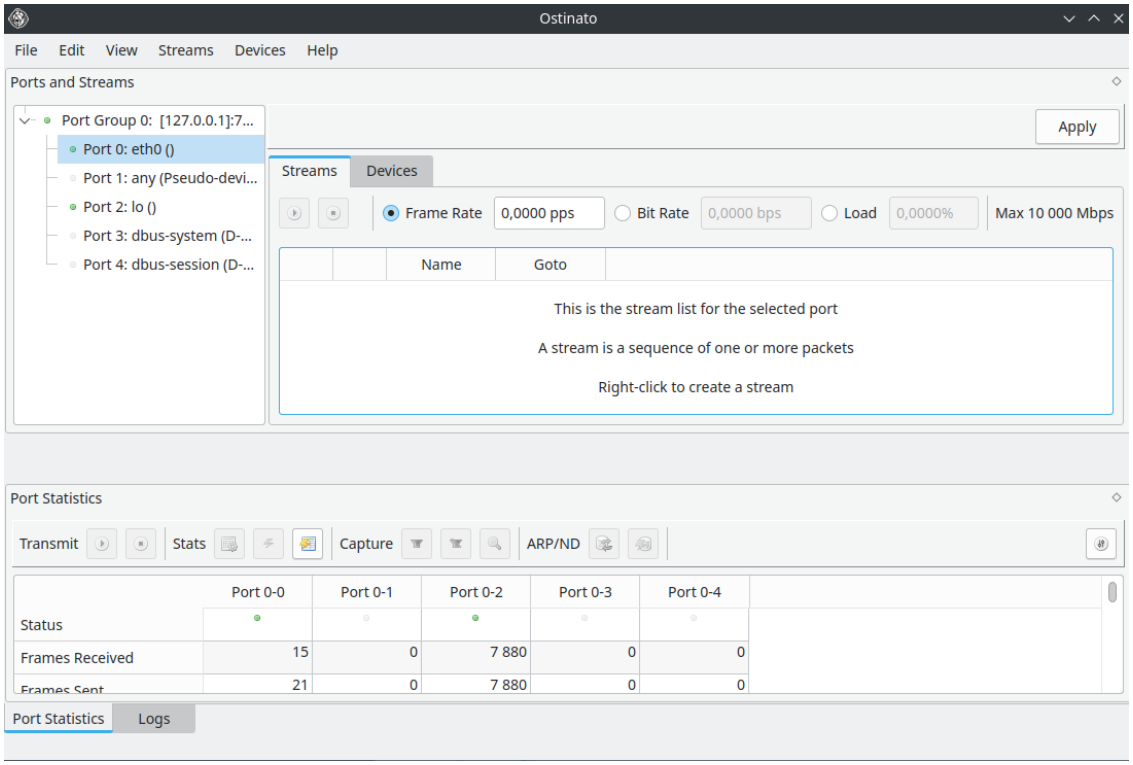

Рисунок 1.1 — Вибір інтерфейсу для генерації трафіку в Ostinato

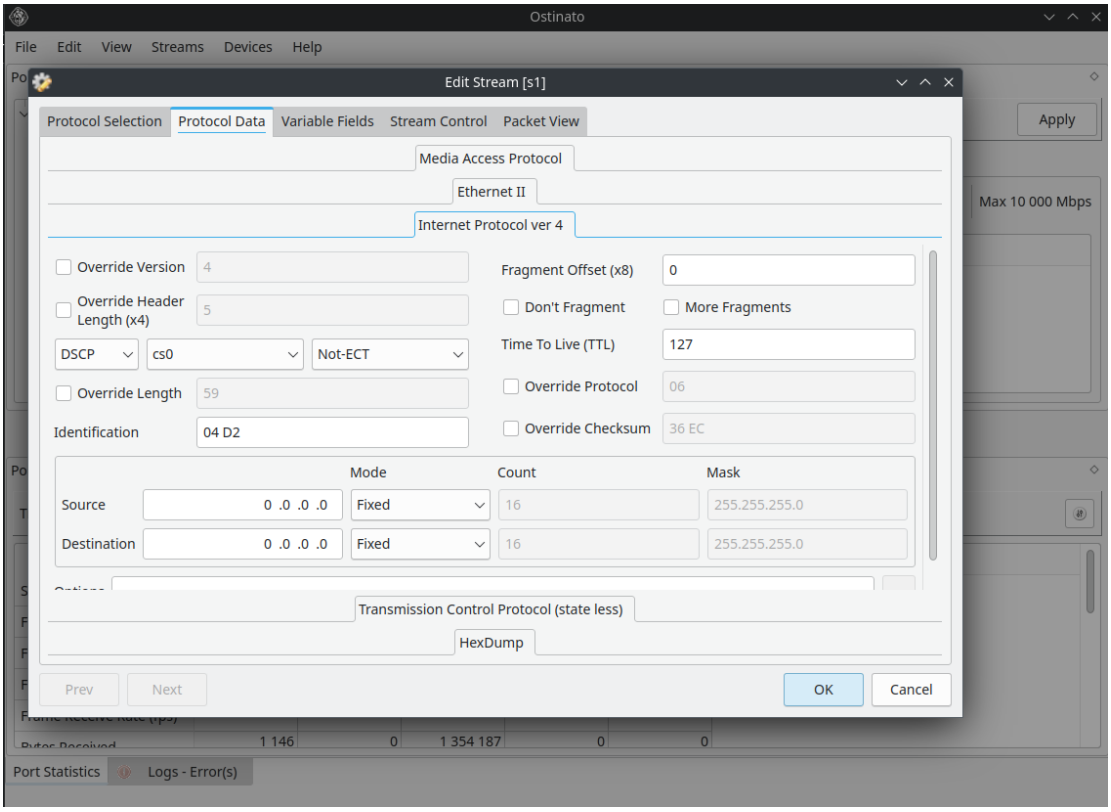

Рисунок 1.2 — Введення даних протоколів потоку в Ostinato

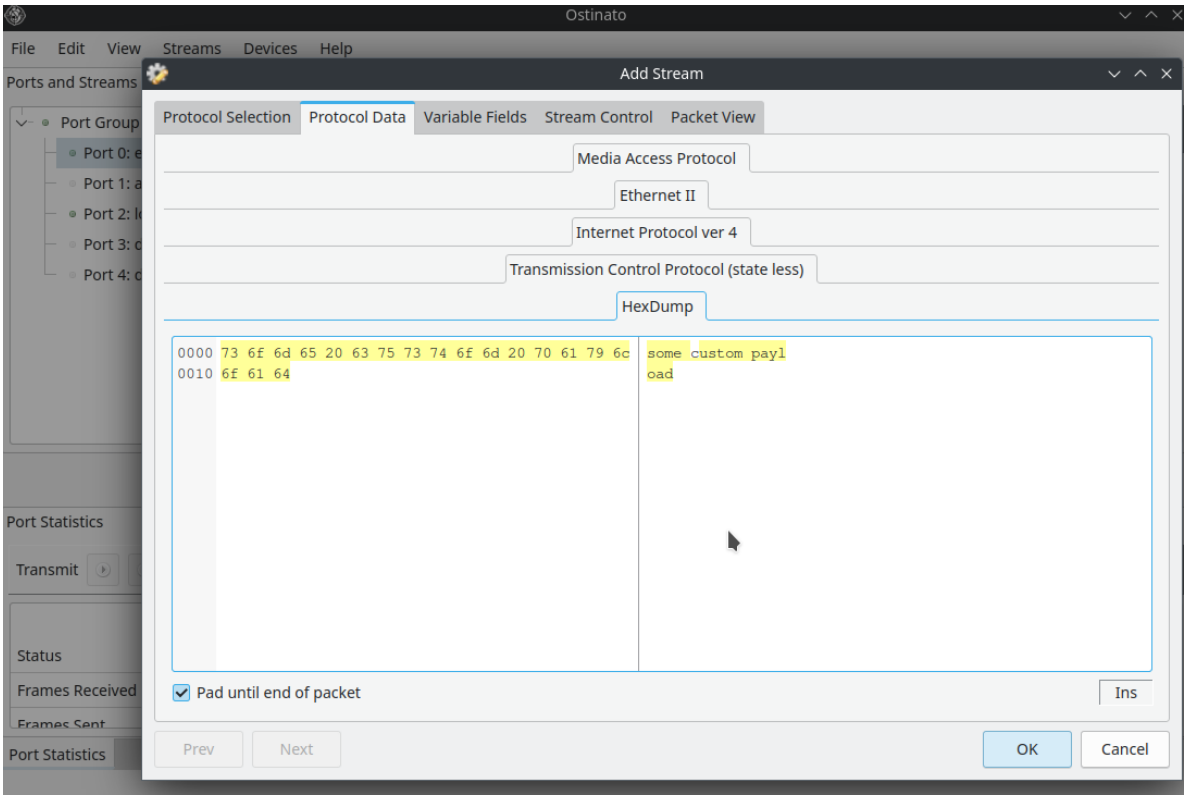

Рисунок 1.3 — Введення довільних даних пакету потоку в Ostinato

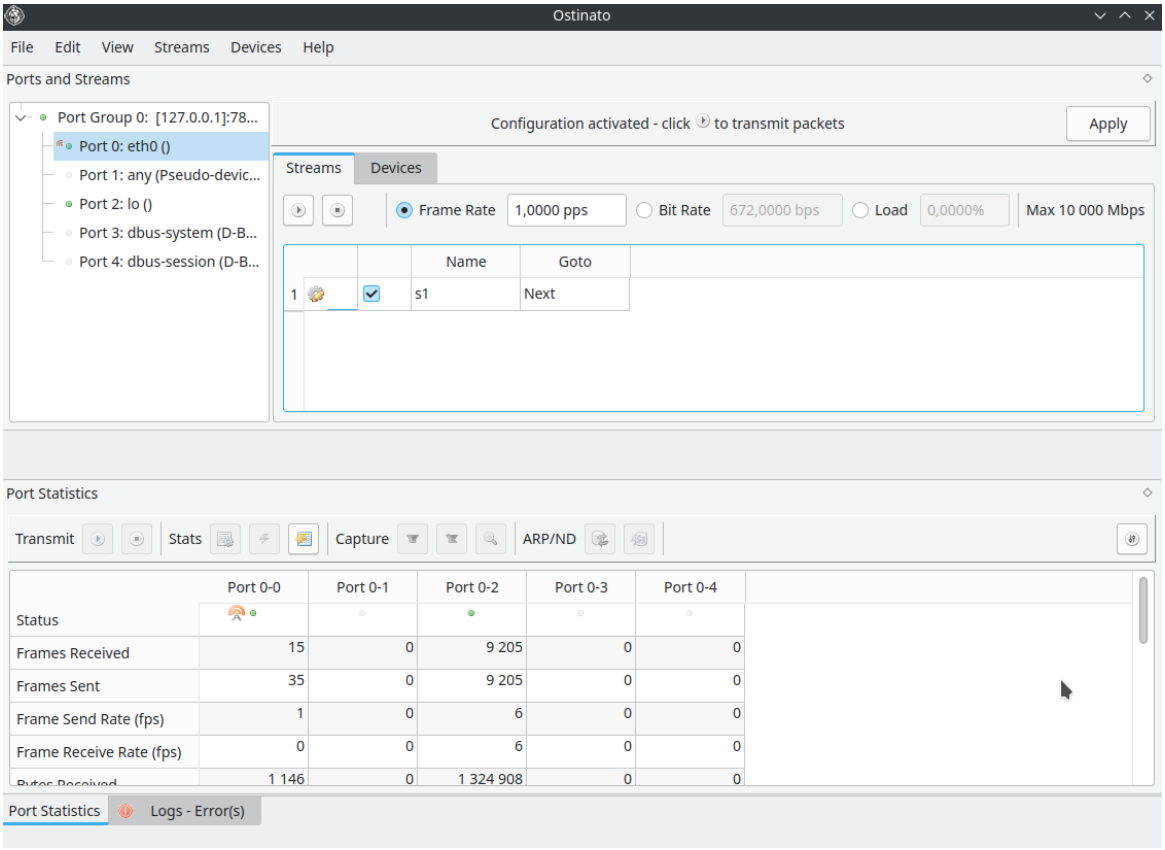

Рисунок 1.4 — Приклад генерації трафіку в Ostinato

Зважаючи на згадане вище, можна виділити наступні недоліки даного аналогу:

- обмеженість графічного інтерфейсу користувача, використання різних графічних елементів інтерфейсу для задання значень для великої кількості полів даних, що заставляє часто між ним переключатись;
- відсутність прямої підтримки протоколів вищих рівнів моделі OSI, хоча це і можливо, якщо вручну формувати трафік вищих протоколів у вигляді даних нижчих.

1.2 Постановка задачі

Із згаданого вище, можна виділити завдання роботи:

- розробити програму, що використовує формальні граматики для опису мережевого трафіку:
  - розробити граматику, синтаксис якої дозволить описувати мережевий трафік;

- розробити програму, що обробляє текст, написаний за розробленою граматикою, зчитуючи та обробляючи лексеми із нього;
- провести дослідження ефективності використання розробленої формальної граматики для опису мережевого трафіку.

Вхідні дані програми:

- текст із командами, написаний за розробленою граматикою;
- шлях із розташуванням плагінів та самі плагіни у вигляді файлів динамічних бібліотек.

Вихідні дані програми:

- код виконання програми: 0 — програма завершилась успішно;
- вихідні дані окремих команд із вхідного тексту: повідомлення користувачу, файли, опис сформованого трафіку.

Програма повинна підтримувати формування довільного трафіку. Аналоги досягають цього за рахунок підтримки низькорівневих протоколів. Це є хорошою ідеєю та буде використано в даній програмі. За рахунок цього, буде доступна і підтримка високорівневих протоколів.

На етапі розробки, достатньому для проведення дослідження, буде реалізована підтримка протоколу TCP.

Також, розробленій програмі варто надати рішення для недоліків аналогу, про обмеженість графічного інтерфейсу, та відсутність прямої підтримки високорівневих протоколів. Оскільки програма використовуватиме граматику для представлення трафіку, у неї немає потреби для графічного інтерфейсу. Щодо формування трафіку високорівневих протоколів, в першу чергу граматика повинна мати здатність їх підтримувати на тому ж рівні що і низькорівневі. Однак, для даної розробки реалізується тільки функціонал, що підтримується аналогом, тому на даному етапі розробки не обов’язково додавати для них пряму підтримку, буде достатньо навести аргументи що це буде можливо в майбутньому. Одним із питань складності підтримки високорівневих протоколів

у аналогу є обмеженість інтерфейсу, тому, за її вирішення, також частково вирішиться питання підтримки високорівневих протоколів.

Для протоколу TCP, аналог не має прямої підтримки формування трафіку для сесій, що складаються із декількох пакетів. Це означає, що в аналогу в таких випадках потрібно проставляти вручну всі поля, що відповідають за призначення пакету до сесії, такі як адреси одержувача та відправника, номер в послідовності, тощо. В розробленій програмі це можна вирішити за рахунок реалізації підтримки сесій протоколу TCP.

З цього можна зробити висновок, що вирішення недоліків аналогу більшою мірою зводиться до вдалої розробки граматики.

Висновки до розділу 1

Для формування мережевого трафіку вже існує безліч інструментів. Однак, вони зазвичай фокусуються на підтримці деякого сталого набору протоколів, і вибудовують свій функціонал навколо цієї вимоги. Найближчим рішенням до вільного формування довільного трафіку є аналог Ostinato, адже він дозволяє формувати дані протоколів низьких рівнів моделі OSI. Однак, по цій же причині, він не дуже зручний для формування трафіку сесій протоколів вищих рівнів. Якщо сформувати пакети для сесії такого протоколу не є великою проблемою концептуально, але необхідність створення великої кількості окремих пакетів у графічному інтерфейсі дещо ускладнює та подовжує таку роботу. В цьому випадку, для кожного пакету потрібно заповнити багато полів із даними, які розміщені на різних панелях інтерфейсу. Хоч це може компенсуватись досвідом користування інструментом, знаходженням дещо альтернативних способів задання пакету, таких як копіювання вже існуючих пакетів та заміна в копії меншої кількості необхідної інформації, все одно існує необхідність частого переміщення між елементами інтерфейсу, які можуть не знаходитись близько одне від одного.

Для надання великої свободи задання трафіку, необхідна підтримка протоколів нижчих рівнів моделі OSI, адже поверх них можна і вручну формувати сесії протоколів вищих рівнів. Однак, для цих сесій не завжди потрібно вказувати всі дані які тільки можна, тобто більшість із них будуть ті ж самі між пакетами, або передбачувано змінюватись. Наприклад, коли в сесії протоколу TCP дві машини обмінюються пакетами, їх IP, MAC адреси, та порти не будуть змінюватись, але будуть по різному сприйматись як адреси призначення чи відправлення. Тобто, для цих даних достатньо вказати їх тільки один раз для сесії, а потім при створенні пакетів достатньо тільки вказувати його напрямок. Зважаючи на це, в даному розділі було сформовано мету та завдання дослідження.

2 РОЗРОБКА ГРАМАТИКИ ДЛЯ ОПИСУ ФОРМУВАННЯ ТРАФІКУ

Питання дослідження способів представлення мережевого трафіку є актуальним, незважаючи на те що вже існує безліч способів генерації трафіку. Свобода представлення трафіку завжди обмежена можливостями програми, що його генерує, в тому числі, її інтерфейсом. Більшість програм концентруються на деякому конкретному виді трафіку, тому підлаштовують свій функціонал та інтерфейс під нього. Натомість, наявність виду інтерфейсу користувача, який міг би бути доцільним для представлення будь якого виду трафіку, міг би надати користувачу більше свободи в цьому завданні. Використання формальної граматики може добре підійти для цього, адже воно дозволить визначити деякий синтаксис що підходитиме для цього.

Ідея опису трафіку за допомогою граматики зводиться до того, що синтаксис деякої граматики дозволятиме представляти мережевий трафік. Цей синтаксис повинен підходити для будь якого виду трафіку. За умови наявності таких якостей синтаксису, він дозволятиме легко розширювати функціонал мови для підтримки нових протоколів та функціоналу.

2.1 Показники ефективності граматики

Для проведення експерименту із визначення ефективності використання подібного мовного інтерфейсу, його можна порівняти із інтерфейсами аналогів, при формулюванні одного виду трафіку в обох програмах. Для визначення результатів експерименту, можна вести ергономічний лічильник. Ефективність можна визначити за значеннями лічильника: якщо значення для розробленої програми будуть меншими — її можна вважати ефективнішою, та навпаки.

Діями, що підраховуватимуться при проведенні експерименту, будуть:

- введення символу із клавіатури:
    - для символів, введення яких потребує натиску клавіш-модифікаторів, натиск цієї клавіші буде рахуватись один раз на групу із послідовних символів, для яких вона використовується. Це пов’язано із тим, що в

таких випадках ця клавіша залишається натиснутою для багатьох послідовних символів;

- натиск на кнопку миші — окрім самого натиску, вважається що в цьому випадку потрібно також навести мишу на правильний елемент інтерфейсу;
- зміна способу введення між мишею та клавіатурою — зміна фокусу потребує деякого часу, щоб перемістити руку із миші на клавіатуру (нехай в експерименті вважається що користувач використовує обидві руки для друку).

2.2 Проєктування граматики

Граматика повинна використовуватись для представлення мережевого трафіку в текстовому вигляді. Оскільки вона повинна мати здатність використовуватись із будь якими видами трафіку, її синтаксис повинен бути достатньо гнучким для цього.

В моделі OSI мережеві протоколи поділяються на рівні, які спираються один на інший, та у кожного протоколу є певна визначена структура в якій він зберігає дані. Виходячи із цього можна сказати, що дані протоколу можна представити деяким об’єктом із атрибутами. При цьому, дані одного протоколу здатні бути атрибутом іншого. Тобто, трафік можна представити в об’єктно-орієнтованому вигляді. На основі цього можна розробити синтаксис, що буде для цього використовуватись.

Для об’єктно-орієнтованого синтаксису достатньо реалізувати хоча б такі операції:

- оголошення змінних — необхідне для створення об’єктів, можливості доступу до них за іменем;
- присвоєння значень змінним — необхідне для задання значень об’єктам;

- доступ до атрибутів змінних — атрибути об’єктів теж є об’єктами, тому маючи можливість доступу до них, до них також можна застосувати операцію присвоєння.

Для деяких типів може виникнути необхідність виконання деяких дій. Наприклад, це можуть бути операції отримання значень об’єкту, із деякою попередньою обробкою, представлення об’єкту в текстовому вигляді, запис у файл, тощо. Тому, є сенс також реалізувати операцію виклику функцій.

Враховуючи вказані вимоги, можна спроектувати синтаксис, який дозволить це реалізувати. Представимо його з використанням нотації Бекуса-Наура:

```
<Application> ::= <Line><Application>?
<Line> ::= <Command><Comment>?<Newline>
<Newline> ::= "\r\n|\n"
<Whitespace> ::= [" ""\t"]
<PrintableChar> ::= [^<Newline>]
<Comment> ::= "#"<PrintableChar>*
<Command> ::= <Declaration>|<Assignment>|<MembAccess>
<VarName> ::= [a-zA-Z_\-][a-zA-Z0-9_\-]*
<Declaration> ::= <Type><Whitespace><VarName><AssignmentOp>?
<Assignment> ::= <Var><AssignmentOp>
<AssignmentOp> ::= <Whitespace>*"="<Whitespace>*<Value>
<MembAccess> ::= <VarName><MembAccessOp>
<MembAccessOp> ::= "."<Var>
<Var> ::= <VarName>|<MembAccess>|<FuncCall>
<Value> ::= <Int>|<Ipv4Addr>|<String>|<Bool>|<Var>
<Number> ::= [0-9]+
<Int> ::= -?<Number>
<Ipv4Addr> ::= <Number>"."<Number>"."<Number>"."<Number>
<String> ::= "\""<StringContents>"\""
<StringContents> ::= [<PrintableChar>|<EscapedChar>|<Newline>]*
<EscapedChar> ::= "\\"[nr\\"]
<FuncCall> ::= <Var>"("<ArgList>?")"
<ArgList> ::= <Value><Whitespace>*(","<Whitespace>*<Value>)*
```

Виходячи із цього, можна описати граматику:

G = { N, T, P, S }, де

множина терміналів T = { a-z, A-Z, 0-9, \r, \n, " ", \t, !, @, #, $, %, ^, &, *, (,), _, +, -, =, /, ,, ., <, >, ?, ', ", ;, :, \, `, ~ },

множина нетерміналів N = { Application, ArgList, ArgListCont, Assignment, AssignmentOp, Comma, Command, Comment, CommentBody, CommentChar, CommentStart, Declaration, Dot, DoubleQuote, EOF, Equal, EscapedChar, FuncCall, Int, Ipv4Addr, Letters, Line, MembAccess, MembAccessOp, Minus, Newline, Numbers, NumberValue, ParenthesesClose, ParenthesesOpen, SpecialChars, String, StringChar, StringContents, Value, Var, VarName, VarNameBody, VarNameCont, VarNameStart, Whitespace },

множина правил P = {

1. Newline -> \r\n | \n ;

2. Whitespace -> " " | \t ;

3. Numbers -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |9 ;

4. Letters -> a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z ;

5. SpecialChars -> ! | @ | # | $ | % | ^ | & | * | [|] | { | } | / | + | ` | ~ | (|) | , | . | < | > | ? | ' | ; | : | | ;

6. EscapedChar -> \\n | \\r | \\\ | \\" ;

7. Minus -> - ;

8. Dot -> . ;

9. Comma -> , ;

10. DoubleQuote -> " ;

11. Equal -> = ;

12. ParenthesesOpen -> (;

13. ParenthesesClose ->) ;

14. VarNameStart -> Letters | _ | - ;

15. VarNameBody -> Letters | Numbers | _ | - ;

16. CommentStart -> # ;

17. EOF -> ε ;

18. Application -> Line Application | EOF ;

19. Line -> (Command | Command Comment) Newline ;

20. Command -> Declaration | Assignment | MembAccess ;

21. Declaration -> Type Whitespace VarName | Type Whitespace Varname AssignmentOp ;

22. Assignment -> Var AssignmentOp ;

23. MembAccess -> VarName MembAccessOp ;

24. CommentChar -> Letters | Numbers | Whitespace | SpecialChars ;

25. Comment -> CommentStart CommentChar ;

26. CommentBody -> CommentChar | CommentChar CommentBody ;

27. Value -> Int | Ipv4Addr | String | Var ;

28. Int -> NumberValue | Minus NumberValue ;

29. Ipv4Addr -> NumberValue Dot NumberValue Dot NumberValue Dot NumberValue ;

30. String -> DoubleQuote StringContents DoubleQuote | DoubleQuote DoubleQuote ;

31. Var -> Varname | MembAccess | FuncCall ;

32. Type -> Varname

33. VarName -> VarNameStart VarNameCont | VarNameStart;

34. VarNameCont -> VarNameBody | VarNameBody VarNameCont ;

35. NumberValue -> Numbers | Numbers NumberValue ;

36. StringChar -> Letters | Numbers | Whitespace | SpecialChars | Newline | EscapedChars ;

37. StringContents -> StringChar | StringChar StringContents ;

38. AssignmentOp -> Equal Value ;

39. MembAccessOp -> Dot Var ;

40. FuncCall -> Var ParenthesesOpen ArgList ParenthesesClose | Var ParenthesesOpen ParenthesesClose ;

41. ArgList -> Value | Value ArgListCont ;

42. ArgListCont -> Comma ArgList ;

},

аксіома S = Application

Наведемо приклад виведення ланцюжка «N num=5». Для цього, виконаємо ряд операцій підстановки, наведений нижче:

Application

->(18) Line EOF

->(19) Command EOF

->(20) Declaration EOF

->(21) Type Whitespace VarName AssignmentOp EOF

->(32, 38) VarName Whitespace VarName Equal Value EOF

->(2, 11, 27) VarName Whitespace VarName Equal Int EOF

->(28) VarName Whitespace VarName Equal NumberValue EOF

->(33, 35) VarNameStart Whitespace VarNameStart VarNameCont Equal Numbers EOF

->(34) VarNameStart Whitespace VarNameStart VarNameBody VarNameBody Equal Numbers EOF

->(14, 15) Letters Whitespace Letters Letters Letters Equal Numbers EOF

->(1, 2, 3, 4, 11, 17) N num=5

2.3 Приклади опису трафіку

За розробленою граматикою можна навести можливі приклади опису трафіку.

Наприклад, трафік із простим HTTP запитом і відповіддю міг би мати такий вигляд:

```
Stream my_stream
my_stream.transport = proto_tcp
my_stream.src_addr = 192.168.0.20
my_stream.dst_addr = 192.168.1.30
my_stream.dst_port = 80
my_stream.src_port = any
```

```
HttpSession http_ssn
HttpRequest req1
req1.url = "index.html"
req1.headers = "Content-Type: text/html; charset=UTF-8\\nConnection: Keep-Alive"
HttpResponse resp1
resp1.code = 404
resp1.body = "custom response string"

TcpPacket raw_resp1 = resp1.raw
raw_resp1.ack = false

http_ssn.insert(req1)
http_ssn.insert(resp1)
my_stream.insert(http_ssn)
my_stream.write_pcap("file_name.pcap")
```

Або, схожий трафік, описаний за допомогою окремих TCP пакетів:

```
TcpStream s
s.src_port=1337
s.dst_port=80
s.mac.src="01:23:45:67:89:ab"
s.mac.dst="cd:ef:1f:ff:ff:ff"
s.ipv4.src=192.168.1.1
s.ipv4.dst=192.168.2.1

s.add(C2S,SYN)
s.add(S2C,SYN,ACK)
s.add(C2S,ACK)

s.add(C2S,"GET / HTTP/1.1\nHost: some.domain.net")
s.add(S2C,ACK,"HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
```

```
Content-Length: 12
Content-Type: text/html

Hello world!")
s.add(C2S,ACK)

s.add(C2S,FIN)
s.add(S2C,FIN,ACK)
s.add(S2C,ACK)
```

Або ж, трафік із повторенням одного TCP пакету декілька разів:

```
TcpPacket p

p.mac.src = "aa:bb:cc:11:22:33"
p.mac.dst = "dd:ee:ff:44:55:66"
p.ip.src = 192.168.1.1
p.ip.dst = 192.168.2.1
p.flags.set(ACK, SYN)
p.seq_num = 42
p.payload="Some custom payload"

TcpStream s
s.add(p, 11)
```

Висновки до розділу 2

У розділі визначено показники для визначення ефективності використання формальної граматики для формування мережевого трафіку. Ці показники є ергономічними, підраховують кількість дій, які необхідно виконати користувачу.

Також було визначено підхід для розробки граматики — їй варто описувати об’єктно-орієнтований синтаксис із підтримкою декількох базових операцій. Наведено приклади синтаксису, що відповідає вказаній граматиці. Приклади описують використання граматики для представлення трафіку протоколів різних рівнів, і всі вони досить природньо описуються нею.

Синтаксис граматики не виділяє зарезервованих слів, а правила для назв типів та змінних однакові. Тому, перевірка правильності назв, відповідності типів повинна відбуватись вже після виділення лексем.

Також, по цій же причині, програмі буде потрібно мати програмне представлення типів представлення трафіку. Як було визначено, цей синтаксис підходить для багатьох протоколів. Але одразу підтримка для них всіх не буде додана. Більше того, з часом з’являються нові протоколи, нові версії існуючих протоколів. Тому, є сенс на етапі проєктування передбачити механізм розширення списку протоколів, що підтримуються.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Проєктування структури програми

3.1.1Базові відомості про ОС, мову, бібліотеки

Програма розроблятиметься з використанням мови C++ для UNIX-подібних операційних систем.

Вибір мови програмування пов’язаний з відносною легкістю портування її коду на інші операційні системи, хороша сумісність із цільовою операційною системою, наявністю бібліотек для низькорівневого доступу до роботи із мережевим трафіком, його формування. Вибір UNIX-подібних операційних систем як цільових пов’язаний із їх поширеністю в галузі мережевих систем, підтримкою стандарту POSIX.

Оскільки вхідними даними програми є текст, необхідності в графічному інтерфейсі немає, тому, програма використовуватиме консольний інтерфейс. Текст для обробки надаватиметься програмі в її стандартний вхідний потік. Подібний спосіб подання вхідних даних робить програму сумісною із механізмом конвеєрів UNIX-подібних операційних систем, що дозволить отримувати вхідні дані як напряму із інших програм, так і при ручному введенні від користувача.

Аргументи запуску для програми на даному етапі не передбачені, однак, за необхідності, їх використання дозволене та не повинно впливати на працездатність програми.

3.1.2Визначення засобів розбору тексту за граматикою

Для парсингу вхідного тексту за розробленою граматикою буде використовуватись бібліотека Flex[4]. З її допомогою, розроблену граматику можна представити в програмному коді у вигляді, близькому до нотації Бекуса-Наура. На цьому кроці відловлюються синтаксичні помилки у вхідному тексті.

Таким чином, вхідний текст можна розбити на лексеми, та виконувати їх подальшу обробку. Подальша обробка включає в себе семантичні перевірки та обробку операцій. Семантичними помилками можуть бути:

- оголошення неіснуючого типу;
- використання неоголошеної змінної;
- невідповідність типів при присвоєнні (тут потрібно мати на увазі, що кожен тип сам визначає з якими типами сумісний, тому присвоєння різних типів не обов’язково є помилкою).

Виконання операцій виконується одразу після їх обробки, результати їх виконання зберігаються у внутрішню пам’ять програми. Взаємодія із зовнішнім середовищем програми, така як збереження результатів у файл, тощо, виконується всередині функцій, визначених для типів представлення трафіку.

3.1.3Визначення внутрішнього представлення типів

Оскільки синтаксис, описаний граматикою, є об’єктно-орієнтованим, у цих об’єктів повинні бути деякі типи.

В подальшому, в даному розділі словом «тип» в розумінні типу даних позначатиметься тип представлення трафіку, доступний в розробленій програмі, а не тип мови C++. В разі, коли матиметься на увазі тип мови C++, його буде явно названо «типом мови C++».

Оскільки програма націлена на майбутнє розширення функціоналу та підтримку більшої кількості типів, необхідно, щоб їх додавання та ідентифікація були легкими для масштабування. Для цього, в програмі повинен бути тип мови C++, роботою якого буде контроль над типами, присвоєння їм унікальних значень. Оскільки у типів повинні бути унікальні ідентифікатори, їх присвоєння повинно відбуватись після запуску програми.

У кожного типу повинна бути можливість збереження значення. Тому, для кожного значення деякого типу повинен бути присутній C++ тип, який визначатиме внутрішні дані типу. C++ клас представлення значення типу може

зберігати значення в C++ типах. Також, тип може мати атрибути та методи, які в свою чергу є змінними деяких типів. Користувач має можливість доступу до них з допомогою операції доступу до атрибуту.

Функції теж будуть представлені окремим типом зі спеціальним значенням. Змінні цього типу можна використовувати для виклику функцій при виконанні користувачем операції виклику функції.

Програма надаватиме декілька вбудованих типів:

- Int — ціле число в діапазоні [-9,223,372,036,854,775,808; 9,223,372,036,854,775,807];
- String — текстовий рядок;
- IPv4Value — адреса протоколу Ipv4:
- Function — функція, яку можна викликати. Для цього типу не передбачене присвоєння значень.

3.1.4Визначення внутрішнього представлення змінних

Змінні зберігатимуться в пам’ять програми при їх оголошенні. Даними, що міститимуть змінні, є:

- ім’я змінної — використовується для її ідентифікації;
- тип змінної;
- значення типу.

C++ тип для представлення змінної надає доступ до її значення, дозволяє виконання операцій присвоєння та виклику функції.

Для контролю над оголошенням та пошуку змінних за іменем використовуватиметься ще один C++ клас, що відповідатиме за це. Окрім контролю над змінними, оголошеними користувачем, цей клас може використовуватись у значенням типів для роботи із їх атрибутами.

3.1.5Представлення плагінів

Програма націлена на використання плагінів для розширення свого функціоналу.

Інтерфейс плагіну являє собою структуру, яка містить інформацію про типи та змінні оголошені в ньому. При завантаженні плагіну, в програму додаються вказані в ньому типи, та оголошуються вказані в ньому змінні. Після цього, вони стають доступними користувачу.

Фізично, плагін являє собою динамічну бібліотеку. Вона відкривається та завантажується в програму після її запуску. Програма шукатиме плагіни за шляхом, вказаним у змінній середовища «TRAFFIC_PLUGINS_PATH». Якщо в ній немає значення — програма не намагатиметься завантажувати плагіни.

3.2 Діаграма класів

На основі описаних рішень, можна спроєктувати діаграму класів програми (рис. 3.1):

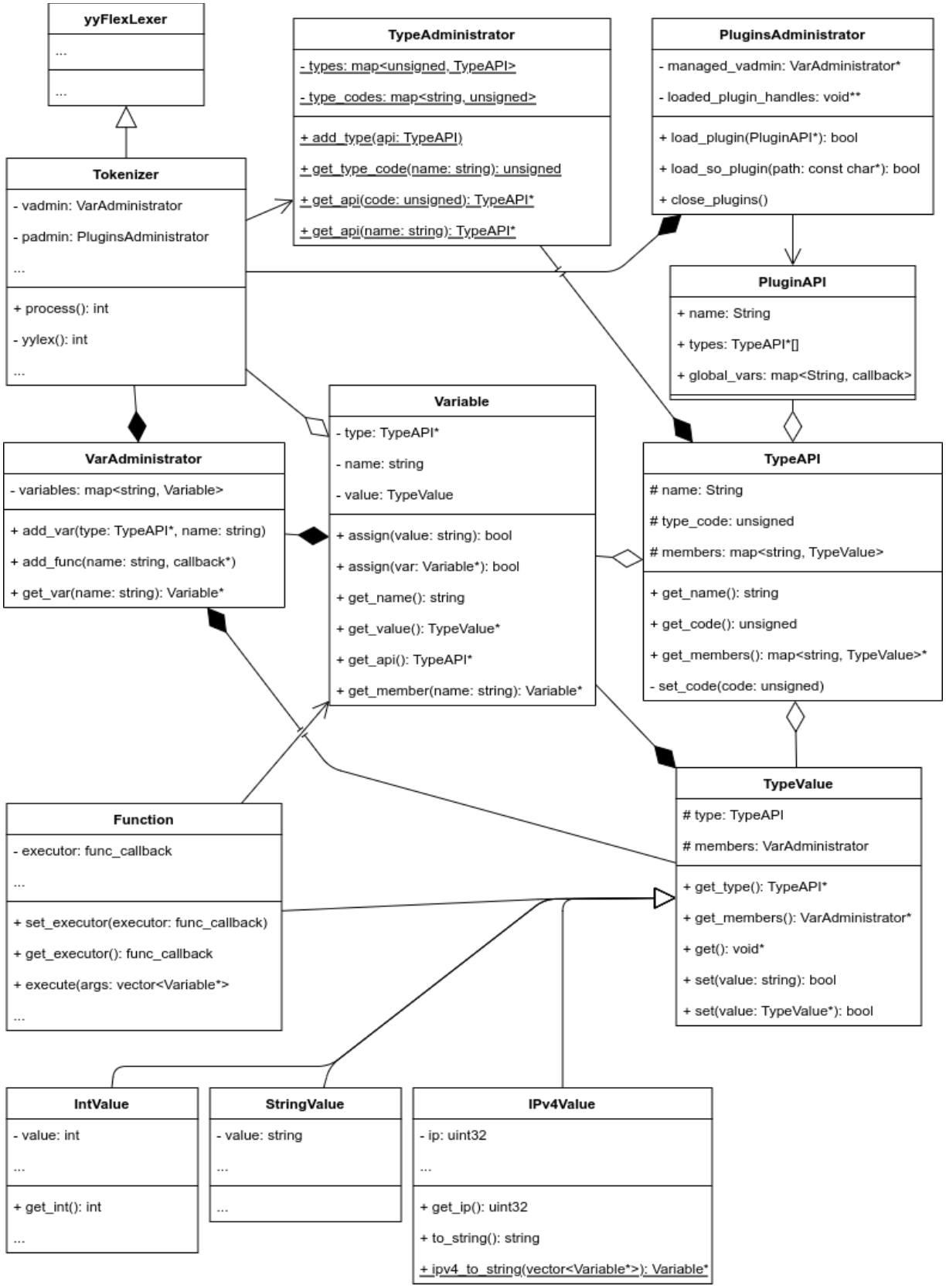

Рисунок 3.1 — Діаграма класів програми

3.3 Проєктування плагіну для протоколу TCP

В даній роботі, розробка підтримки формування мережевого трафіку націлена на протокол TCP. Для цього потрібно розробити типи, що представлятимуть цей вид трафіку.

3.3.1Типи для представлення TCP трафіку

Протокол TCP підтримує сесію між двома пристроями. Кожен TCP пакет в цій сесії має декілька головних речей, які відповідають за асоціацію його із нею:

- адреси відправника та одержувача:
  - MAC адреси;
  - IP адреси;
  - номери портів;
- номер в послідовності;
- номер підтвердження.

Для цих значень, є сенс контролювати їх автоматично, щоб вони виставлялись при оголошенні пакету. Тобто, при додаванні пакету можна вказати його напрямок, що автоматично встановить адреси одержувача та відправника, встановити номери послідовності та підтвердження.

Сесія складається із набору пакетів. Разом із значеннями, згаданими вище, TCP пакет також містить такі значення, які можуть бути цікавими користувачу:

- прапорці пакету;
- дані пакету.

Ці дані задаються користувачем тільки вручну.

Виходячи із цього можна описати потрібні для плагіну типи:

TcpStream — зберігає пакети сесії, відповідає за присвоєння пакетам значень, що ідентифікують сесію.

Атрибути:

- mac — інформація про MAC адреси користувачів сесії;
- ip — інформація про IP адреси користувачів сесії;

- src_port — номер порту відправника;
- dst_port — номер порту одержувача;

Методи:

- add(packet, amount) — додати в сесію попередньо визначений пакет. В цьому випадку, значення пакету не змінюватимуться, і його можна вставити довільну кількість разів;
- add(direction, flags, payload) — додати в сесію пакет із заданими параметрами. Пакет створюється всередині методу, йому автоматично присвоюються значення деяких полів;
- to_pcap(path) — записати сформований трафік в файл формату захоплення пакетів pcap. На даному етапі розробки, перетворення внутрішнього представлення пакету на дані справжнього мережевого пакету не реалізоване. Натомість, зараз ця функція виводить повідомлення про те, що її потрібно реалізувати;
- to_string() — формує текстовий рядок, що містить інформацію про всі пакети в сесії.

TcpPacket — представляє об’єкт TCP пакету.

Атрибути:

- mac — інформація про MAC адреси відправника та одержувача;
- ip — інформація про IP адреси відправника та одержувача;
- dir — напрямок руху пакету, від ініціатора сесії, чи до нього;
- src_port — порт відправника пакету;
- dst_port — порт отимувача пакету;
- seq_num — номер послідовності пакету;
- ack_num — номер підтвердження пакету;
- flags — прапорці пакету;
- payload — дані пакету.

Методи:

- to_string() — формує текстовий рядок із інформацією про пакет.

MacValue — представляє MAC адресу

Методи:

- to_string() — формує текстовий рядок із MAC адресою.

MacInfo — зберігає інформацію про MAC адреси сесії.

Атрибути:

- src — MAC адреса відправника пакету;
- dst — MAC адреса одержувача пакету.

IPv4Info - зберігає інформацію про IPv4 адреси сесії.

Атрибути:

- src — IPv4 адреса відправника пакету;
- dst — IPv4 адреса одержувача пакету.

TcpFlags — зберігає інформацію про встановлені прапорці пакету. Всього є шість можливих прапорців: FIN, SYN, RST, PSH, ACK, URG. Ці прапорці повинні бути доступними у вигляді вбудованих змінних із відповідними значеннями. Один об’єкт TcpFlags може встановлювати довільну кількість прапорців.

Методи:

- set(flags) — встановити значення прапорців об’єкту. Нові прапорці додаються до вже існуючих;
- unset(flags) — вилучити у об’єкту вказані прапорці.
- to_string() — сформувати текстовий рядок із інформацією про встановлені прапорці.

Direction — зберігає інформацію про напрямок руху пакету в сесії. Може набувати двох можливих значень: C2S (Client-To-Server), S2C (Server-To-Client). Ці значення повинні бути доступними у вигляді вбудованих змінних із відповідними значеннями.

Методи:

- set() — встановити значення напрямку;
- to_string() — сформувати текстовий рядок із напрямком руху.

Тобто, плагін для формування трафіку за протоколом TCP надаватиме програмі наступне:

- типи даних:
  - TcpStream;
  - TcpPacket;
  - MacValue;
  - MacInfo;
  - IPv4Info;
  - TcpFlags;
  - Direction;
- вбудовані змінні:
  - FIN: TcpFlags;
  - SYN: TcpFlags;
  - RST: TcpFlags;
  - PSH: TcpFlags;
  - ACK: TcpFlags;
  - URG: TcpFlags;
  - C2S: Direction;
  - S2C: Direction.

3.4 Діаграма класів плагіну

Враховуючи описані рішення, можна спроєктувати діаграму класів для плагіну підтримки протоколу TCP (рис. 3.2)

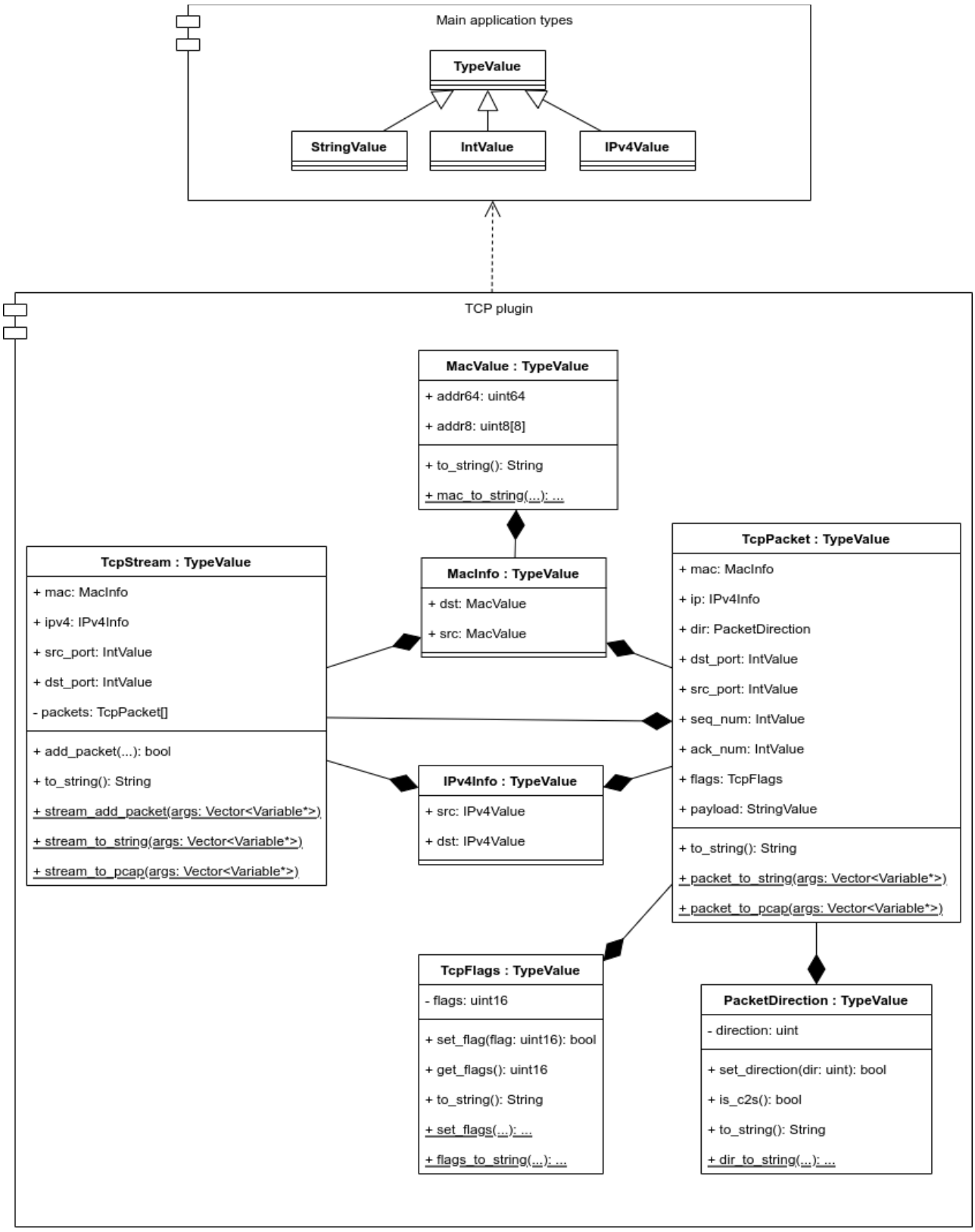

Рисунок 3.2 — Діаграма класів плагіну протоколу TCP

3.5 Тестування

Основною задачею програми є обробка лексем із вхідного тексту, подальше виконання зчитаних команд, та надання каркасу для використання плагінів. Тому, для програми варто протестувати ці частини.

Плагін для протоколу TCP додає нові типи та функціонал до них, тому для плагіну є сенс протестувати наявність цих типів, та правильність роботи функціоналу, що вони надають.

Вихідний код написаних тестів можна побачити в Додатку *вставити номер додатку з програмним кодом*.

3.5.1Модульне тестування

Для перевірки працездатності програми було проведено модульне тестування. Для цього була використана бібліотека Catch2[5]. Модульне тестування плагіну протоколу TCP менш доцільне, адже в плагінах широко застосовуються символи із головної програми, тому для плагіну модульне тестування не проводилось.

Модульні тести для програми націлені на покриття виведених за граматикою ланцюжків, працездатність класів каркасу програми. Тести можна розділити на дві групи:

- тести граматики — перевіряють правильність виявлення лексем;
- тести обробки лексем — перевіряють правильність обробки лексем, правильність роботи класів програмного каркасу.

Вхідними даними тестів є текст, написаний за розробленою граматикою. Очікуваним результатом тестів є виведення деякого тексту у вихідний потік. Для модульних тестів було увімкнено виведення даних для відлагодження.

Для тестів граматики не виконується подальша програмна обробка лексем, а тільки їх правильне розпізнавання, тому в них можна очікувати побачити невідповідності значень і типів, тощо.

Для тестів обробки лексем також використовується тестовий плагін, що додає тестовий тип «Test» із деякими атрибутами і методами. Таким чином також тестується функціонал використання плагінів.

Розроблені модульні тести можна побачити в таблицях 3.1 та 3.2.

Таблиця 3.1 — Модульні тести граматики

№	Назва тесту	Вхідні дані	Очікуваний результат
1	Коротке оголошення	Int variable	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: EOF
2	Повне оголошення без пробілів	Int variable=21	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF
3	Повне оголошення із пробілами та значенням типу Int	Int variable \t\t = \t 21	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF
4	Повне оголошення зі значенням типу IPv4	Int variable = 192.168.0.1	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN IP: 192.168.0.1 [DEBUG]: EOF
5	Повне оголошення зі значенням типу String	Int variable = "text \n multiline"	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: \"text \\n multiline\" [DEBUG]: EOF
6	Повне оголошення із присвоєнням іншої змінної	Int variable = another_var	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: another_var [DEBUG]: EOF

Продовження таблиці 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат
7	Повне оголошення із доступом до атрибуту	Int variable = another_var.member.another_member	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: another_var [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: another_member [DEBUG]: EOF
8	Повне оголошення із викликом методу	Int variable = another_var.method()	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: variable [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: another_var [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: method [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: EOF
9	Присвоєння значення типу Int без пробілів	int=21	[DEBUG]: INIT VAR_NAME: int [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF
10	Присвоєння значення типу Int із пробілами	int \t =\t21	[DEBUG]: INIT VAR_NAME: int [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF
11	Присвоєння значення типу IPv4	addr = 192.168.1.1	[DEBUG]: INIT VAR_NAME: addr [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN IP: 192.168.1.1 [DEBUG]: EOF
12	Присвоєння значення типу String	str = "string"	[DEBUG]: INIT VAR_NAME: str [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: \"string\" [DEBUG]: EOF
13	Присвоєння іншої змінної	var = other_var	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other_var [DEBUG]: EOF

Продовження таблиці 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат
14	Присвоєння значення із атрибуту	var = other.member	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: EOF
15	Присвоєння значення атрибуту	var.member = other	[DEBUG]: INIT VAR_NAME: var [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: EOF
16	Присвоєння значення поверненого функцією	var = other.getter()	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: getter [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: EOF
17	Присвоєння значенню, поверненому функцією	var.getter() = other	[DEBUG]: INIT VAR_NAME: var [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: getter [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: EOF
18	Присвоєння значення літералу	216.58.215.110 = "google.com'	[DEBUG]: Unknown token: 2
19	Доступ до атрибуту	var = other.member	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: EOF
20	Операція доступу до атрибуту без вказання імені	var = other.	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: Unexpected EOF

Продовження таблиці 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат
21	Операція доступу до атрибуту із пробілами з двох сторін крапки	var = other . member	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: Unknown token:
22	Операція доступу до атрибуту із пробілом після крапки	var = other. member	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: Unknown token:
23	Рекурсивний доступ до атрибутів	var = other.member.other_member	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: other_member [DEBUG]: EOF
24	Виклик методу	other.method()	[DEBUG]: INIT VAR_NAME: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: method [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: EOF
25	Виклик методу із пробілом перед дужками	var.method ()	[DEBUG]: INIT VAR_NAME: var [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: method [DEBUG]: Unknown token:
26	Виклик методу атрибуту змінної	var = other.member.method()	[DEBUG]: INIT VAR_NAME: var [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: other [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: member [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: method [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: EOF

Продовження таблиці 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат
27	Аргумент функції типу Int	call(21)	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG INT: 21 [DEBUG]: METHOD CALL END [DEBUG]: EOF
28	Аргумент функції типу IPv4	ping(8.8.8.8)	[DEBUG]: INIT VAR_NAME: ping [DEBUG]: METHOD CALL START [DEBUG]: ARG IP: 8.8.8.8 [DEBUG]: METHOD CALL END [DEBUG]: EOF
29	Аргумент функції типу String	ping("google.com")	[DEBUG]: INIT VAR_NAME: ping [DEBUG]: METHOD CALL START [DEBUG]: ARG STRING: "google.com" [DEBUG]: METHOD CALL END [DEBUG]: EOF
30	Змінна як аргумент функції	call(mom)	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: METHOD CALL END [DEBUG]: EOF
31	Атрибут змінної як аргумент функції	call(mom.phone)	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: phone [DEBUG]: METHOD CALL END [DEBUG]: EOF
32	Результат методу як аргумент функції	call(mom.phone())	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: phone [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: METHOD CALL END [DEBUG]: EOF
33	Декілька аргументів із пробілами	call(mom, 5, "hi there")	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: ARG INT: 5 [DEBUG]: ARG STRING: "hi there" [DEBUG]: METHOD CALL END [DEBUG]: EOF
34	Декілька аргументів без пробілів	call(mom,5,"hi there")	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: ARG INT: 5 [DEBUG]: ARG STRING: "hi there" [DEBUG]: METHOD CALL END [DEBUG]: EOF

Продовження таблиці 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат
35	Декілька аргументів із пробілами між усіма лексемами в списку аргументів	call(mom , 5 , \"hi there\")	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: ARG INT: 5 [DEBUG]: ARG STRING: "hi there" [DEBUG]: METHOD CALL END [DEBUG]: EOF
36	Відсутній аргумент після коми в списку аргументів	call(mom,)	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG VARIABLE: mom [DEBUG]: Unknown token:)
37	Кінець введення в списку аргументів	call(-1	[DEBUG]: INIT VAR_NAME: call [DEBUG]: METHOD CALL START [DEBUG]: ARG INT: -1 [DEBUG]: Unexpected EOF
38	Коментар на початку рядку закінчується кінцем введення	# comment	[DEBUG]: COMMENT [DEBUG]: EOF
39	Коментар після команди закінчується кінцем введення	Int i # comment	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: COMMENT [DEBUG]: EOF
40	Коментар на початку рядку закінчується новим рядком	# comment\n	[DEBUG]: COMMENT [DEBUG]: EOF
41	Коментар після команди закінчується новим рядком	Int i # comment\n	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: COMMENT [DEBUG]: EOF

Таблиця 3.2 — Модульні тести обробки лексем

№	Назва тесту	Вхідні дані	Очікуваний результат
1	Виявлення типу Int	Int int	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: int [DEBUG]: EOF Змінна «int» має тип «Int»
2	Виявлення типу String	String str	[DEBUG]: INIT VAR_NAME: String [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: str [DEBUG]: EOF Змінна «str» має тип «String»
3	Виявлення типу Test	Test test	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: test [DEBUG]: EOF Змінна «test» має тип «Test»
4	Виявлення типу що не підтримується	Arbitrary var	[DEBUG]: INIT VAR_NAME: Arbitrary [DEBUG]: DECLARATION START [DEBUG]: Unsupported type: Arbitrary
5	Присвоєння літералу типу Int	Int i i = 21	DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: i [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF Змінна «i» має значення 21

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
6	Присвоєння літералу типу IPv4	IPv4 ip ip = 192.168.0.255	[DEBUG]: INIT VAR_NAME: IPv4 [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: ip [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: ip [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN IP: 192.168.0.255 [DEBUG]: EOF Змінна «ip» має значення192.168.0.255
7	Присвоєння літералу типу String	String s s = "text"	[DEBUG]: INIT VAR_NAME: String [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: s [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: s [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: "text" [DEBUG]: EOF Змінна «s» має значення «text»
8	Присвоєння літералу типу String із екранованими символами	String s = "\\escaped\\ \n \"quoted\r\""	[DEBUG]: INIT VAR_NAME: String [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: s [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: "\\escaped\\ \ n \"quoted\r\"" [DEBUG]: EOF Змінна «s» має значення «\\ escaped\\ \n \"quoted\r\"»

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
9	Присвоєння змінних одного типу	Int i1 = 21 Int i2 i2 = i1	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i1 [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i2 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: i2 [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: i1 [DEBUG]: EOF Змінні «i1» та «i2» мають значення 21
10	Присвоєння літералу типу Int при оголошенні змінної	Int i1 = 21	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i1 [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF Змінна «» має значення 21
11	Присвоєння літералу типу String при оголошенні змінної	String s = "text"	[DEBUG]: INIT VAR_NAME: String [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: s [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: "text" [DEBUG]: EOF Змінна «s» має значення «text»

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
12	Присвоєння літералу типу IPv4 при оголошенні змінної	IPv4 ip = 192.168.0.255	[DEBUG]: INIT VAR_NAME: IPv4 [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: ip [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN IP: 192.168.0.255 [DEBUG]: EOF Змінна «ip» має значення 192.168.0.255
13	Присвоєння змінної того ж типу	Int i1 = 21 Int i2 = i1	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i1 [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i2 [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: i1 [DEBUG]: EOF Змінні «i1» та «i2» мають значення 21
14	Присвоєння літералу іншого типу	Int i = "string"	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN STRING: \"string\" [DEBUG]: Invalid value

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
15	Присвоєння змінної іншого типу	String s Int i = s	[DEBUG]: INIT VAR_NAME: String [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: s [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: s [DEBUG]: Assignment error
16	Доступ до існуючого атрибуту змінної	Test t t.num	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: EOF
17	Доступ до неіснуючого атрибуту змінної	Test t t.no	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: no [DEBUG]: There is no such member on this variable

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
18	Спроба доступу до атрибутів для типу, у якого їх немає	Int i i.no	[DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: i [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: no [DEBUG]: There is no such member on this variable
19	Присвоєння значення атрибуту	Test t t.num = 21	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: EOF Змінна «t.num» має значення 21

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
20	Присвоєння атрибуту значення іншого атрибуту	Test t1 Test t2 t1.num=21 t2.num=t1.num	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t1 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t2 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t2 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: EOF Змінна «t2.num» має значення 2

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
21	Присвоєння змінній значення атрибуту	Test t t.num=21 Int i=t.num	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: Int [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: i [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN VARIABLE: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: EOF Змінна «i» має значення 21
22	Виклик функції	print("hello world!")	[DEBUG]: INIT VAR_NAME: print [DEBUG]: METHOD CALL START [DEBUG]: ARG STRING: \"hello world!\" [DEBUG]: METHOD CALL END hello world! [DEBUG]: EOF

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
24	Виклик методів у аргументах виклику методу	Test t1 t1.num=21 t1.print(21, t1.get(), t1.num)	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t1 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: ASSIGNMENT START [DEBUG]: ASSIGN INT: 21 [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: print [DEBUG]: METHOD CALL START [DEBUG]: ARG INT: 21 [DEBUG]: ARG VARIABLE: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: get [DEBUG]: METHOD CALL START [DEBUG]: METHOD CALL END [DEBUG]: ARG VARIABLE: t1 [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: num [DEBUG]: METHOD CALL END printing Ints with t1.print: 21 21 21 [DEBUG]: EOF

Продовження таблиці 3.2

№	Назва тесту	Вхідні дані	Очікуваний результат
23	Виклик методу	Test t t.print(21)	[DEBUG]: INIT VAR_NAME: Test [DEBUG]: DECLARATION START [DEBUG]: DECLARATION VAR_NAME: t [DEBUG]: Command line ended [DEBUG]: INIT VAR_NAME: t [DEBUG]: MEMBER ACCESS START [DEBUG]: ACCESS TO SUB-MEMBER: print [DEBUG]: METHOD CALL START [DEBUG]: ARG INT: 21 [DEBUG]: METHOD CALL END printing Ints with t.print: 21 [DEBUG]: EOF

Результати проведення модульних тестів граматики:

==========

All tests passed (82 assertions in 1 test case)

Результати проведення модульних тестів обробки лексем:

==========

All tests passed (99 assertions in 1 test case)

3.5.2Системне тестування

Системне тестування виконувалось із використанням плагіну для протоколу TCP, перевіряло можливість програми працювати із плагінами, та працездатність функціоналу, що надається плагіном. Для написання тестів використовувався каркас bats[6]. Розроблені тести можна побачити в таблиці 3.3.

Таблиця 3.3 — Системні тести із плагіном протоколу TCP

№	Назва тесту	Вхідні дані	Очікуваний результат

1	Тестування типів для роботи із MAC адресами	MacValue mac_src = "01:23:45:67:89:ab"
		MacInfo mac_info
		mac_info.src = mac_src
		mac_info.dst = "cd:ef:1f:ff:ff:ff"
		MacInfo other = mac_info
		# check that values were copied correctly
		print(mac_info.src.to_string())
		print(mac_info.dst.to_string())
		print(other.src.to_string())
		print(other.dst.to_string())
			1:23:45:67:89:ab:0:0
			cd:ef:1f:ff:ff:ff:0:0
			1:23:45:67:89:ab:0:0
			cd:ef:1f:ff:ff:ff:0:0

2	Тестування типів для роботи із IPv4 адресами	IPv4 src_ip = 192.168.0.1
		IPv4Info ip_info
		ip_info.src = src_ip
		ip_info.dst = 192.168.1.1
		IPv4Info other = ip_info
		# check that values were copied correctly
		print(ip_info.src.to_string())
		print(ip_info.dst.to_string())
		print(other.src.to_string())
		print(other.dst.to_string())
			192.168.0.1
			192.168.1.1
			192.168.0.1
			192.168.1.1

3	Тестування типу Direction	Direction dir = S2C Direction other = dir print(dir.to_string()) print(other.to_string()) print(C2S.to_string()) print(S2C.to_string())	S2C S2C C2S S2C

Продовження таблиці 3.3

№	Назва тесту	Вхідні дані	Очікуваний результат
4	Тестування типу TcpFlags	TcpFlags flags flags.set(ACK, SYN) print(flags.to_string()) flags.unset(SYN) print(flags.to_string()) print(FIN.to_string()) print(SYN.to_string()) print(RST.to_string()) print(PSH.to_string()) print(ACK.to_string()) print(URG.to_string())	0000000000010010 SYN ACK 0000000000010000 ACK 0000000000000001 FIN 0000000000000010 SYN 0000000000000100 RST 0000000000001000 PSH 0000000000010000 ACK 0000000000100000 URG
5	Тестування типу TcpPacket	# packet here isn't supposed to be valid protocol-wise TcpPacket p p.dir = C2S Int src_port = 1337 p.ip.src = 192.168.2.2 p.src_port = src_port p.dst_port = 80 p.seq_num = 20 p.ack_num = 40 p.flags.set(ACK, SYN) p.payload = "GET / HTTP/1.1\nHost: some.domain.net" ... print(p.payload) print(p.to_string())	.... GET / HTTP/1.1 Host: some.domain.net 192.168.2.2:1337 -> 0.0.0.0:80 Flags: 0000000000010010 SYN ACK Seq: 20, Ack: 40

Продовження таблиці 3.3

№	Назва тесту	Вхідні дані	Очікуваний результат
6	Тестування типу TcpStream	TcpStream stream stream.src_port = 1337 stream.dst_port = 80 MacValue mac_src = "01:23:45:67:89:ab" MacInfo mac_info mac_info.src = mac_src mac_info.dst = "cd:ef:1f:ff:ff:ff" stream.mac = mac_info IPv4 src_ip = 192.168.0.1 IPv4Info ip_info ip_info.src = src_ip ip_info.dst = 192.168.1.1 stream.ip = ip_info print(stream.src_port) ... print(stream.ip.dst.to_string()) # traffic here isn't supposed to be valid protocol-wise TcpPacket p p.dir = C2S p.src_port = 1337 p.dst_port = 80 p.seq_num = 20 p.ack_num = 40 p.flags.set(ACK, SYN) p.payload = "GET / HTTP/1.1\\nHost: some.domain.net" stream.add(p) stream.add(p, 3) stream.add(S2C, RST) print(stream.to_string())	1337 ... 192.168.1.1 0.0.0.0:1337 -> 0.0.0.0:80 Flags: 0000000000010010 SYN ACK Seq: 20, Ack: 40 0.0.0.0:1337 -> 0.0.0.0:80 Flags: 0000000000010010 SYN ACK Seq: 20, Ack: 40 0.0.0.0:1337 -> 0.0.0.0:80 Flags: 0000000000010010 SYN ACK Seq: 20, Ack: 40 0.0.0.0:1337 -> 0.0.0.0:80 Flags: 0000000000010010 SYN ACK Seq: 20, Ack: 40 192.168.1.1:80 -> 192.168.0.1:1337 Flags: 0000000000000100 RST Seq: 21, Ack: 0

Продовження таблиці 3.3

№	Назва тесту	Вхідні дані	Очікуваний результат
7	Опис сесії протоколу TCP	TcpStream s s.src_port = 1337 s.dst_port = 80 s.mac.src = "01:23:45:67:89:ab" s.mac.dst = "cd:ef:1f:ff:ff:ff" s.ip.src = 192.168.1.1 s.ip.dst = 192.168.2.1 # A simple HTTP1 session over TCP ... s.add(C2S, "GET / HTTP/1.1\nHost: some.domain.net") s.add(S2C, ACK, "HTTP/1.1 200 OK Date: Sat, 09 Oct 2010 14:28:02 GMT Server: Apache Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT Content-Length: 12 Content-Type: text/html Hello world!") s.add(C2S, ACK) ... print(s.to_string())	... 192.168.1.1:1337 -> 192.168.2.1:80 Flags: 0000000000000000 Seq: 4, Ack: 0 192.168.2.1:80 -> 192.168.1.1:1337 Flags: 0000000000010000 ACK Seq: 5, Ack: 4 192.168.1.1:1337 -> 192.168.2.1:80 Flags: 0000000000010000 ACK Seq: 6, Ack: 5 ...

Результати проведення системного тестування:

```
# tests/plugins/tcp/stateful_stream/test.bats:
1..1
ok 1 plugins/tcp/stateful_stream

# tests/plugins/tcp/types/ipv4/test.bats:
1..1
ok 1 plugins/tcp/types/ipv4

# tests/plugins/tcp/types/pkt_direction/test.bats:
1..1
ok 1 plugins/tcp/types/pkt_direction

# tests/plugins/tcp/types/tcp_stream/test.bats:
1..1
ok 1 plugins/tcp/types/tcp_stream
```

```
# tests/plugins/tcp/types/tcp_flags/test.bats:
1..1
ok 1 plugins/tcp/types/tcp_flags

# tests/plugins/tcp/types/mac/test.bats:
1..1
ok 1 plugins/tcp/types/mac

# tests/plugins/tcp/types/tcp_packet/test.bats:
1..1
ok 1 plugins/tcp/types/tcp_packet
```

Висновки до розділу 3

Виконано проєктування архітектури інструментального забезпечення для дослідження ефективності використання формальної граматики для формування мережевого трафіку.

Розроблена програма здатна отримувати вхідні дані у вигляді тексту, написаного відповідно до синтаксису розробленої формальної граматики, та обробляти записані в ньому команди. Програма надає функціонал для розробки довільних плагінів, що розширюють її можливості.

Було розроблено плагін для підтримки протоколу TCP. Це дозволить використовувати програму не тільки для представлення трафіку протоколу TCP, але й інших протоколів вищих рівнів, що базуються на ньому.

На даному етапі можна виділити, що розроблений плагін надає деяку підтримку сесій цього протоколу. Це досягається за рахунок автоматизованого виставлення значень асоціації пакету із сесією, таких як адреси одержувача та відправника, тощо.

Модульне тестування підтвердило успішну роботу програми. Модульне тестування було націлене на перевірку синтаксису, обробки лексем, внутрішні типи представлення лексем. Також, деякі тести використовують тестовий плагін, що неявно тестує функціонал із використання плагінів.

Системне тестування підтвердило успішну роботу плагіну із підтримкою протоколу TCP, типів та значень що він додає. Тести були націлені на оголошення змінних нових типів, їх подальше використання, доступ до їх

атрибутів. Деякі тести намагаються імітувати види трафіку, складені у відповідності до мережевих протоколів.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ФОРМУВАННЯ ТРАФІКУ ЗА ДОПОМОГОЮ ГРАМАТИК

Проведемо експерименти із представлення деякого виду трафіку з використанням обох програм. При цьому, будемо вимірювати метрики, вказані в постановці задачі.

Експериментальна база буде змінюватись залежно від виду трафіку, що представляється. Інтерфейси різних програм можуть бути ефективними для різних видів трафіку, тому варто розглянути різні випадки ефективності представлення трафіку аналогом, та порівняти представлення того ж трафіку розробленою програмою.

4.1 Експеримент 1. Представлення трафіку із одноманітними пакетами

Мета: порівняння мовного та графічного інтерфейсів для представлення виду трафіку, для якого програма-аналог має ефективний інтерфейс.

Експериментальна база: для цього експерименту, спробуємо представити потік одинадцяти однакових пакетів протоколу TCP, із використанням протоколів MAC, Ethernet2 та IPv4. Для них, виставимо значення:

- MAC адреси:
  - відправник: aa:bb:cc:11:22:33;
  - одержувач: dd:ee:ff:44:55:66;
- Ipv4 адреси:
  - відправник: 192.168.1.1;
  - одержувач: 192.168.2.1;
- значення протоколу TCP:
  - прапорці: ACK, SYN;
  - номер в послідовності: 42;
- дані пакету: «Some custom payload»

Хід експерименту:

Кроки виконання експерименту із використанням програми-аналогу Ostinato можна побачити в таблиці 4.1.

Таблиця 4.1 — Хід експерименту 1 із використанням аналогу

№	Опис дії	Приклад виконання	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
1	Вибір мережевого інтерфейсу для передачі трафіку	Рис.4.1	2	0	0
2	Створення нового потоку	Рис.4.2	2	0	0
3	Вибір протоколів. L1 — MAC, L2 — Ethernet 2, L3 — Ipv4, L4 — TCP, Payload — Hex Dump, Frame Length - 77	Рис.4.3	5	3	2
4	Введення значень протоколу MAC. Окрім введення самих адрес, необхітно перейти на сторінку з полями для введення, та змінити тип значень адрес на «Fixed»	Рис.4.4	7	46(10)	4
5	Введення значень протоколу IPv4	Рис.4.5	3	18	4
6	Введення значень протоколу TCP	Рис.4.6	4	3	2
7	Введення даних пакету	Рис.4.7	2	20(1)	2

Продовження таблиці 4.1

№	Опис дії	Приклад виконання	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
8	Введення кількості пакетів у потоці	Рис.4.8	2	2	2
9	Завершення створення потоку	Рис.4.8 (кнопка «OK» в низу вікна), рис.4.9	2	0	0
	Підсумок		29	92(11)	16

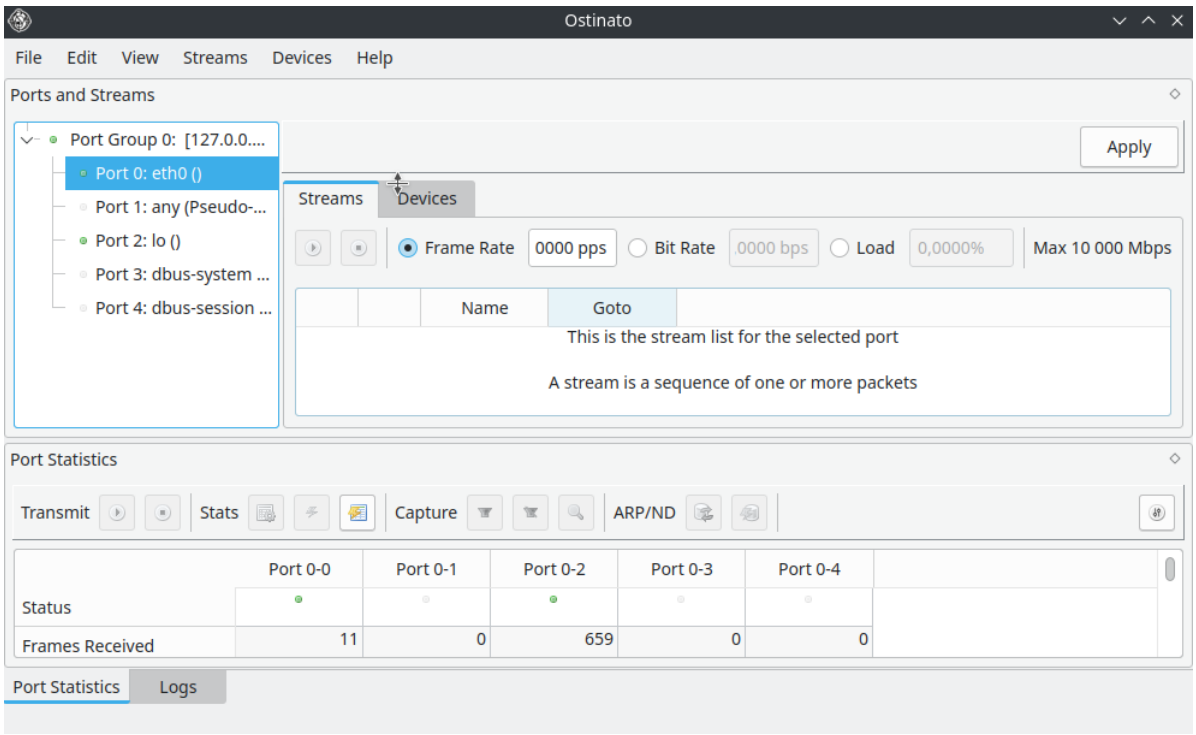

Рисунок 4.1 — Вибір інтерфейсу для передачі трафіку

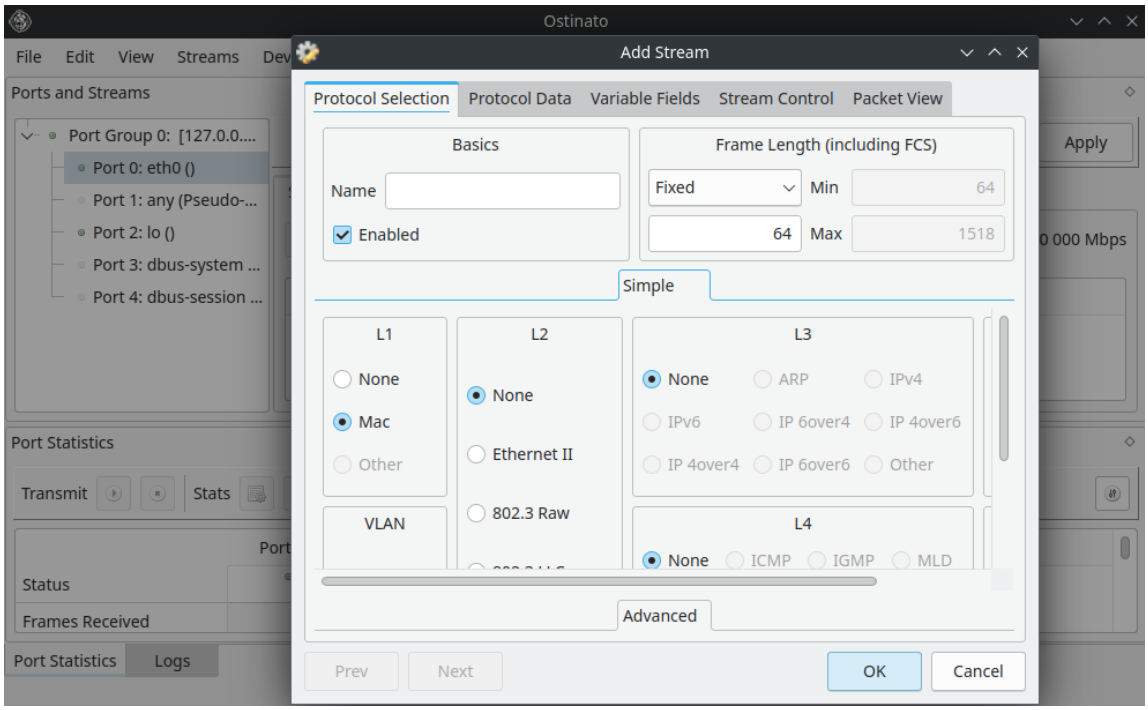

Рисунок 4.2 — Створення нового потоку

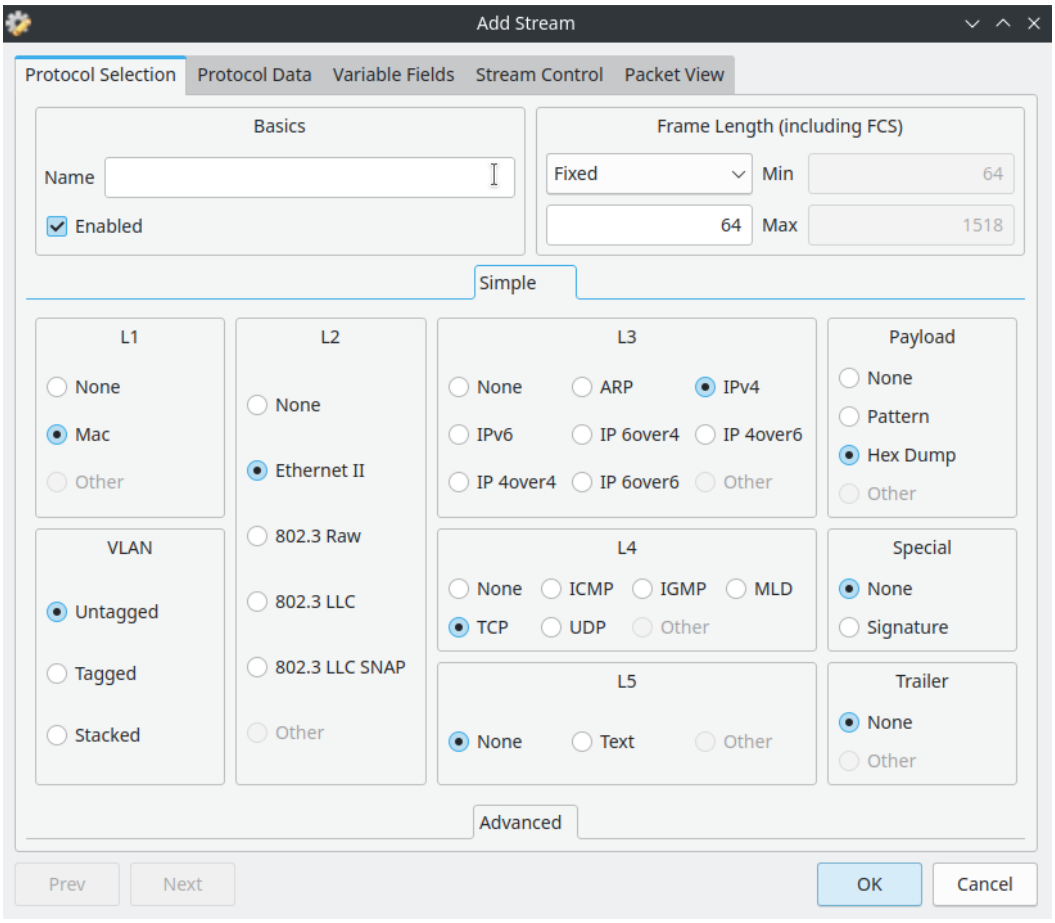

Рисунок 4.3 — Вибір протоколів потоку

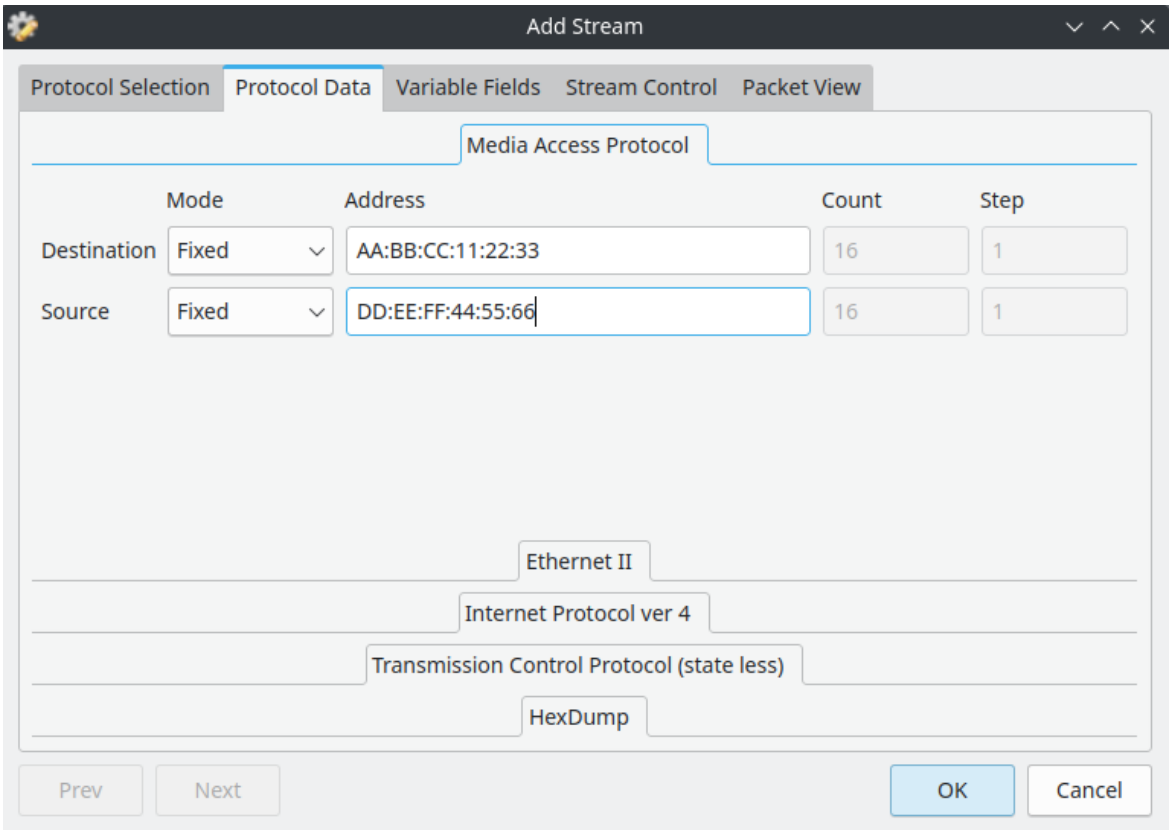

Рисунок 4.4 — Введення значень протоколу MAC

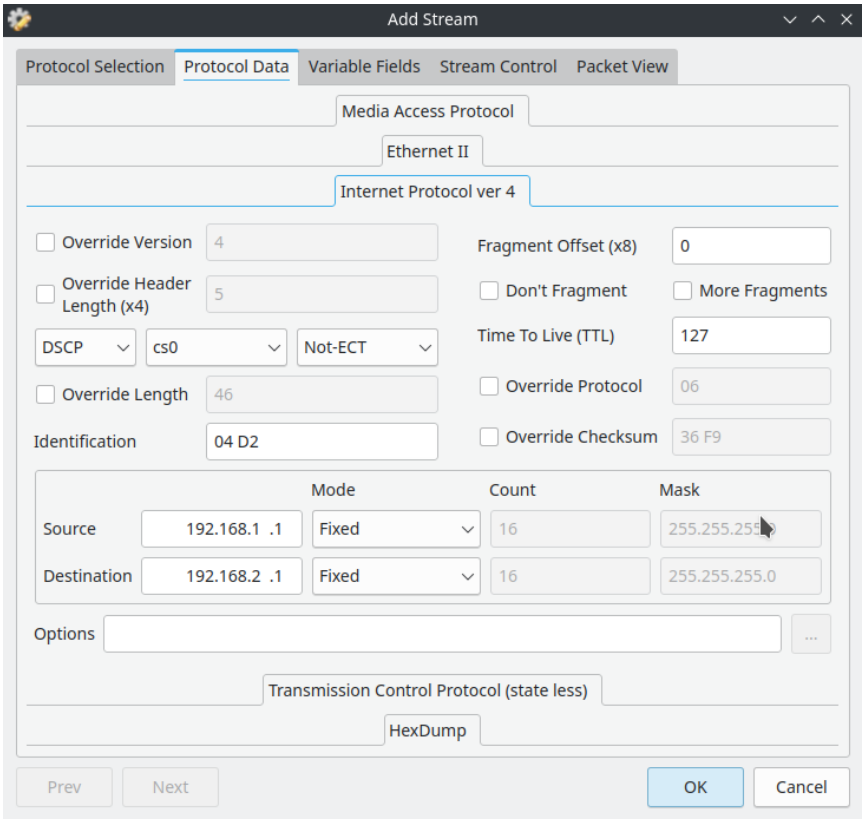

Рисунок 4.5 — Введення значень протоколу IPv4

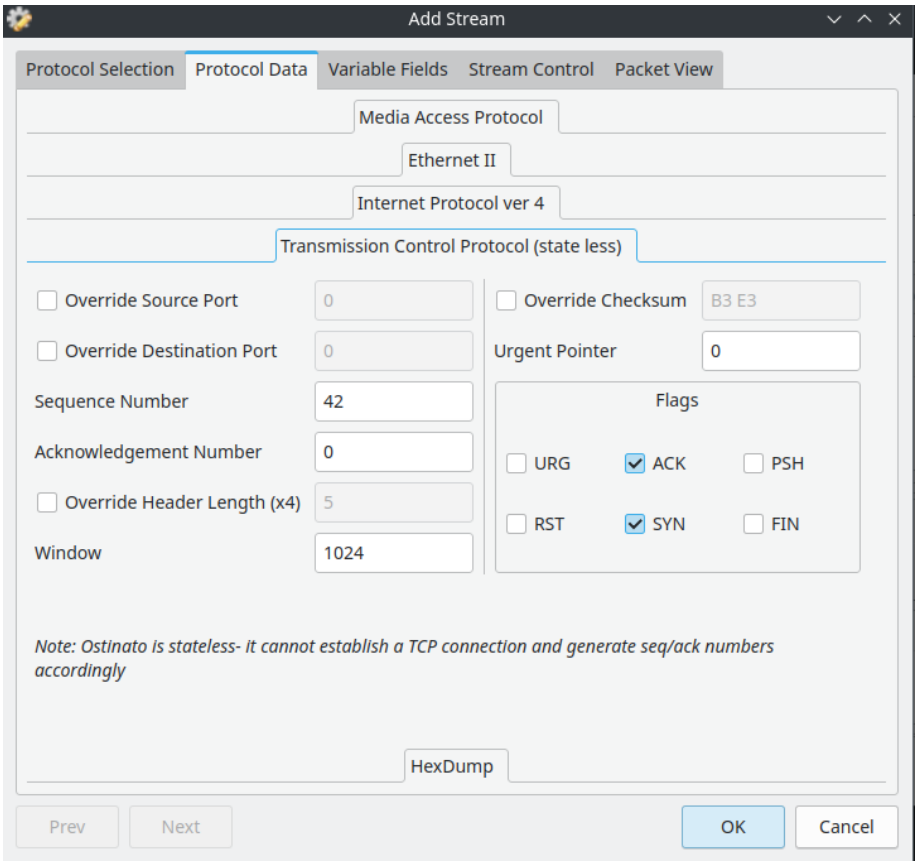

Рисунок 4.6 — Введення значень протоколу TCP

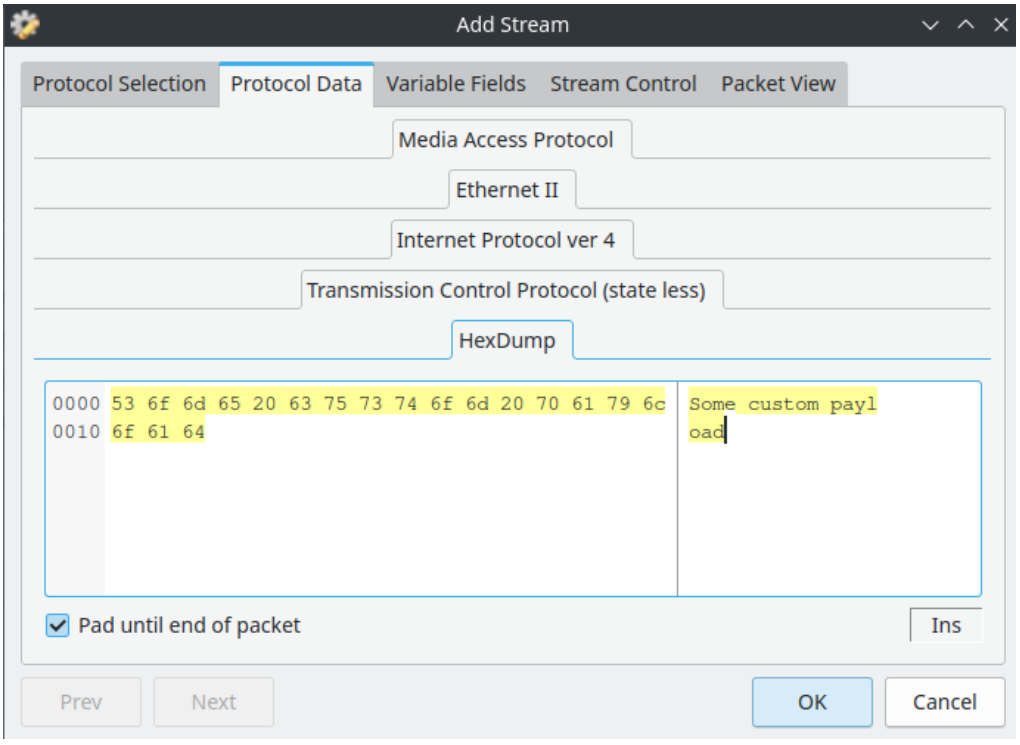

Рисунок 4.7 — Введення даних пакету

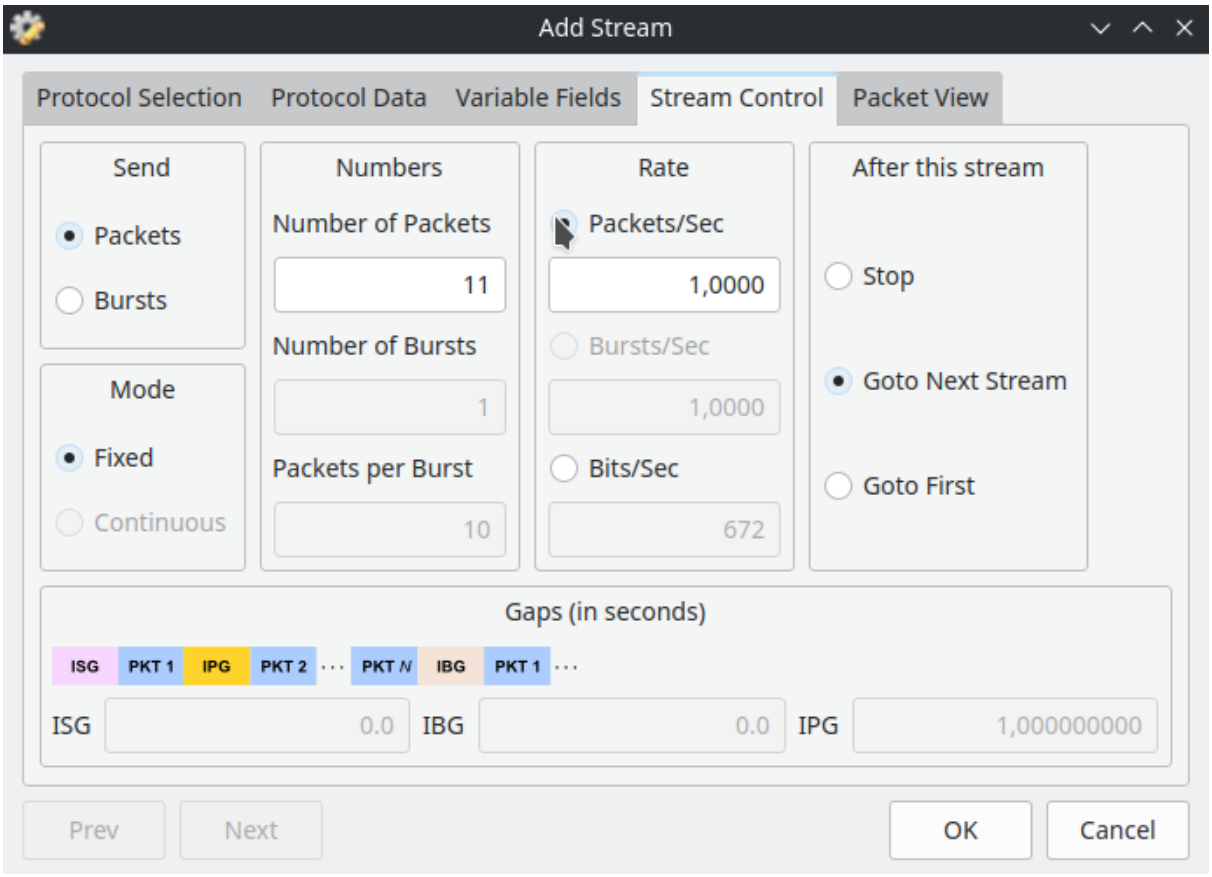

Рисунок 4.8 — Введення кількості пакетів

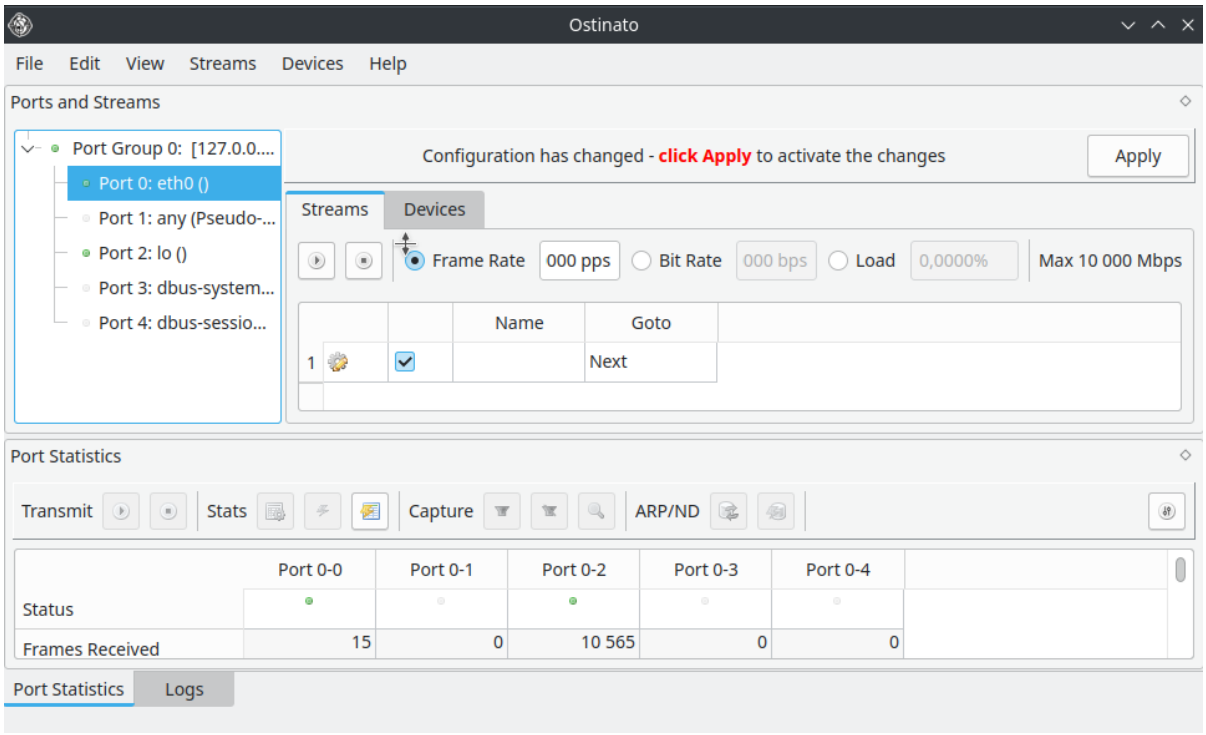

Рисунок 4.9 —Активація створеного потоку

Із використанням розробленої програми:

Текст, що представляє вказаний вид трафіку, виглядає наступний чином:

```
TcpPacket p
p.mac.src="aa:bb:cc:11:22:33"
p.mac.dst="dd:ee:ff:44:55:66"
p.ip.src=192.168.1.1
p.ip.dst=192.168.2.1
p.flags.set(ACK,SYN)
p.seq_num=42
p.payload="Some custom payload"
TcpStream s
s.add(p,11)
```

Кроки виконання експерименту із використанням розробленої програми можна побачити в таблиці 4.2.

Таблиця 4.2 — Хід експерименту 1 із використанням розробленої програми

№	Опис дії	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
1	Введення тексту представлення трафіку	0	225(22)	0

Результати:

- при використанні графічного аналогу:
  - 29 натискань кнопок миші;
  - 92 натиски клавіш клавіатури;
  - 16 змін фокусу між мишею та клавіатурою;
- при використанні розробленої граматики:

- 225 натискань клавіш клавіатури.

4.2 Експеримент 2. Представлення трафіку із різноманітними пакетами

Мета: порівняння мовного та графічного інтерфейсів для представлення виду трафіку, для якого програма-аналог не має ефективного інтерфейсу.

Експериментальна база: для цього експерименту, спробуємо представити деяку сесію, що триває протягом декількох пакетів. Нехай, це буде сесія протоколу HTTP1, клієнт завантажує сторінку за посиланням «some.domain.net/», та отримує у відповідь HTML файл із вмістом «Hello world!».

Цю сесію необхідно сформувати вручну із TCP пакетів. Тобто, пакети в цій сесії виглядатимуть так:

- триходове рукостискання початку з'єднання:
  - пакет від клієнта до сервера, прапорці: SYN;
  - пакет від сервера до клієнта, прапорці: SYN, ACK;
  - пакет від клієнта до сервера, прапорці: ACK;
- HTTP сесія:
  - пакет від клієнта до сервера із GET запитом:
    - дані: «GET / HTTP/1.1\nHost: some.domain.net»
  - пакет від сервера до клієнта із відповіддю:
    - прапорці: ACK;
    - дані: «HTTP/1.1 200 OK\nDate: Sat, 09 Oct 2010 14:28:02 GMT\ nServer: Apache\nLast-Modified: Tue, 01 Dec 2009 20:18:22 GMT\ nContent-Length: 12\nContent-Type: text/html\n\nHello world!»;
  - пакет від клієнта до сервера із підтвердженням отримання відповіді, прапорці: ACK;
- чотириходове рукостискання закінчення з'єднання:
  - пакет із прапорцями: FIN;
  - пакет із прапорцями: FIN, ACK;

- пакет із прапорцями: ACK.

Дані про клієнта та сервер сесії:

- клієнт:
  - MAC: cd:ef:1f:ff:ff:ff;
  - Ipv4: 192.168.1.1;
  - порт: 1337;
- сервер:
  - MAC: 01:23:45:67:89:ab;
  - Ipv4: 192.168.2.1;
  - порт: 80.

Хід експерименту:

Оскільки Ostinato не має підтримки трафіку із станами, їх потрібно контролювати вручну. Це означає, що окрім даних сесії, вказаних вище, також необхідно для кожного пакету вручну вказувати його номер в послідовності.

Також, оскільки «потоки» Ostinato являють собою інформацію про один пакет, то для представлення кожного пакету із нашої сесії потрібно створити по «потоку».

Кроки виконання експерименту із використанням розробленої програми можна побачити в таблиці 4.3.

Таблиця 4.3 — Хід експерименту 2 із використанням аналогу

№	Опис дії	Приклад виконання	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
1	Вибір мережевого інтерфейсу для передачі трафіку	Рис.4.1	2	0	0

Продовження таблиці 4.3

№	Опис дії	Приклад виконання	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
2	Створення нового потоку для пакету 1.1	Рис.4.2	2	0	0
3	Вибір протоколів. L1 — MAC, L2 — Ethernet 2, L3 — Ipv4, L4 — TCP	Рис.4.3	4	0	0
4	Введення значень протоколу MAC	Рис.4.4	7	46(10)	4
5	Введення значень протоколу IPv4	Рис.4.5	3	18	4
6	Введення значень протоколу TCP: прапорець ACK	Рис.4.6	7	10	8
7	Завершення створення потоку	Рис.4.6 (кнопка «OK» в низу вікна), рис.4.9	2	0	0
8	Повторити кроки 2-7 для пакету 1.2		26	74(10)	16
9	Повторити кроки 2-7 для пакету 1.3		25	74(10)	16

Продовження таблиці 4.3

№	Опис дії	Приклад виконання	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
10	Повторити кроки 2-6 для пакету 2.1		24	77(10)	18
11	Ввести дані пакету 2.1		2	42(3)	2
12	Повторити крок 7 для пакету 2.1		2	0	0
13	Повторити кроки 10-12 для пакету 2.2		29	273(38)	20
14	Повторити кроки 2-7 для пакету 2.3		25	74(10)	16
15	Повторити кроки 2-7 для пакету 3.1		25	74(10)	16
16	Повторити кроки 2-7 для пакету 3.2		26	74(10)	16
17	Повторити кроки 2-7 для пакету 3.3		25	74(10)	16
	Підсумок		236	910(121)	152

Із використанням розробленої програми:

В даному випадку, у розробленої програми є перевага, оскільки в ній вже реалізована підтримка потоків зі станами. Тому, немає необхідності ручного контролю над номером пакету в послідовності, та необхідності введення повних адрес відправника та одержувача для кожного пакету. Достатньо вказати цю

інформацію один раз на увесь потік, після чого тільки вказувати напрямок пакету.

Текст, що представляє вказаний вид трафіку, виглядає наступний чином:

```
TcpStream s
s.src_port=1337
s.dst_port=80
s.mac.src="01:23:45:67:89:ab"
s.mac.dst="cd:ef:1f:ff:ff:ff"
s.ipv4.src=192.168.1.1
s.ipv4.dst=192.168.2.1
s.add(C2S,SYN)
s.add(S2C,SYN,ACK)
s.add(C2S,ACK)
s.add(C2S,"GET / HTTP/1.1\nHost: some.domain.net")
s.add(S2C,ACK,"HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
Content-Length: 12
Content-Type: text/html

Hello world!")
s.add(C2S,ACK)
s.add(C2S,FIN)
s.add(S2C,FIN,ACK)
s.add(S2C,ACK)
```

Кроки виконання експерименту із використанням розробленої програми можна побачити в таблиці 4.4.

Таблиця 4.4 — Хід експерименту 2 із використанням розробленої програми

№	Опис дії	Натиски кнопок миші	Натиски клавіш клавіатури	Кількість змін фокусу між мишею та клавіатурою
1	Введення тексту представлення трафіку	0	575(78)	0

Результати:

- при використанні графічного аналогу:
  - 236 натискань кнопок миші;
  - 910 натискань клавіш клавіатури;
  - 152 зміни фокусу між мишею та клавіатурою;
- при використанні розробленої граматики:
  - 575 натискань клавіш клавіатури.

Висновки до розділу 4

Проведені експерименти показали суттєву відмінність у кількості дій між двома програмами.

В першому експерименті, аналогу сумарно знадобилось 137 дій, розробленій програмі — 225. Бачимо, що аналогу знадобилось менше дій для представлення одноманітного трафіку. Як згадувалось у першому розділі, Ostinato представляє трафік «потоками», які містять інформацію про один мережевий пакет, тому представлення одного пакету відбувається досить легко.

З використанням формальної граматики потрібно було виконувати 225 окремих дій, всі з яких — введення з клавіатури. Число стало більшим за рахунок того, що вибір полів для введення інформації про пакет також відбувається в текстовому вигляді з необхідністю вводити слова в декілька символів, і, відповідно, декілька дій, в той час як в графічному аналогу це

відбувається одним кліком миші. Бачимо, що при необхідності опису одного пакету, аналог із графічним інтерфейсом має перевагу.

В другому експерименті, аналогу сумарно знадобилось 1298 дій, розробленій програмі — 575. Бачимо, що розробленій програмі знадобилось менше дій для опису більшої кількості різноманітних пакетів.

Досить незвично, бачимо, що аналогу із графічним інтерфейсом знадобилось 910 натискань клавіатури, проти 575 натискань у розробленій програмі із використанням граматики. Це пояснюється тим, що в цьому випадку у розробленої програми є перевага завдяки тому, що вона вміє не тільки просто описувати окремі пакети, але й об’єднувати їх в сесію. Це дозволяє уникнути ручного введення одноманітної інформації для кожного пакету в ній. В тексті опису трафіку для цього експерименту бачимо, що адреси клієнта і сервера були вказані тільки один раз, після чого вказувався тільки напрямок пакету, і ці адреси правильно інтерпретувались. В графічному аналогу подібного функціоналу немає, тому в ньому для кожного пакету необхідно було окремо вводити дані сесії.

Бачимо, що для опису більшої кількості різноманітних пакетів розроблена програма має перевагу за рахунок підтримки функціоналу, якого немає в програмі-аналогу.

Судячи із результатів експериментів можна дійти висновку, що представлення трафіку із використанням формальних граматик здатне мати переваги над аналогами, залежно від виду трафіку, що представляється, і дій, які для цього необхідно виконати.

В експерименті 1 обидві програми виконували однакові загальні дії для представлення однакового трафіку, і в цьому випадку графічний інтерфейс аналогу мав перевагу над розробленою програмою. В експерименті 2 обидві програми описували однаковий трафік, але різними діями опису, що дозволило розробленій програмі взяти першість за рахунок покращеного функціоналу для конкретно того виду трафіку. При цьому, новий функціонал був доданий без

необхідності зміни розробленої граматики в будь якому вигляді, що вказує на його гнучкість.

Недоліком проведених експериментів є те, що вони вважають що у всіх підрахованих дій однакова вага, але в реальних умовах це не так. В майбутньому, це варто взяти до уваги.

ВИСНОВКИ

Формування мережевого трафіку з допомогою формальних граматик не є загальноприйнятим способом опису трафіку. Як правило, існуючі програми, що виконують формування трафіку, використовую або графічний, або консольний інтерфейс із опціями запуску програми. Крім того, вони, як правило, спеціалізуються тільки на описі деякого одного виду трафіку. Цього можна очікувати і в абсолютній більшості випадків користувачам цього достатньо. Однак, для користувачів, які часто мають справу із різноманітними видами трафіку це може доставляти незручності, адже їм тоді потрібно використовувати декілька різних програм, швидше за все із різними інтерфейсами.

Для таких випадків, зручно мати уніфікований інтерфейс для будь яких видів трафіку. Подібного результату можна досягти із використанням формальної граматики, синтаксис якої міг би дозволяти описувати будь який трафік.

Під час розробки було визначено, що цього можна досягти із використанням об’єктно-орієнтованого синтаксису, адже якщо трафік можна поділити на смислові частини, їх можна представити у вигляді об’єкту.

Оскільки програма що використовує формальну граматику для опису трафіку націлена на можливість представлення будь якого виду трафіку, є сенс додати їй можливість динамічного розширення функціоналу підтримки видів трафіку, без рекомпіляції. Це надає такі дві переваги:

1. відсутність необхідності рекомпіляції дозволить розробнику не поширювати нові версії цілого додатку при додаванні нового функціоналу, та, відповідно, користувачам не потрібно при цьому його обновляти;
2. користувач має свободу не використовувати непотрібний функціонал, звільнивши цей обчислювальний ресурс під щось інше.

Даних умов було досягнуто із використанням системи плагінів, які представлені у вигляді динамічних бібліотек та відвантажуються в програму після її запуску.

Із використанням подібного мовного інтерфейсу вхідні дані являють собою просто текст. Це дозволяє легко змінювати вхідні дані навіть із використанням програмних засобів, що відкриває двері до досить легкої автоматизованого формування процедурно генерованого трафіку, за потреби. Цього в певній мірі можна досягти і з консольним інтерфейсом що приймає вхідні дані у вигляді опцій та змінних середовища. Однак, із графічним інтерфейсом досягти подібного дуже проблематично, якщо звичайно сам інтерфейс не має прямої підтримки чогось подібного.

Для підтримки широкого спектру протоколів для опису, в даній розробці була додана підтримка протоколу TCP. Це зумовлено тим, що із використанням низькорівневого протоколу можна описувати високорівневі протоколи, що базуються на ньому. Підтримка цього протоколу виконана у вигляді окремого плагіну.

Експерименти із збору та порівняння кількості дій для опису трафіку в розробленій програмі та програмі-аналогу показали, що при виконанні ідентичних дій з опису трафіку в графічному та мовному інтерфейсах, графічний інтерфейс вимагає менше дій із введення даних від користувача. Цього можна очікувати, адже коли в графічному інтерфейсі вибір потрібного елементу відбувається одним кліком миші, в текстовому інтерфейсі ці дані потрібно обирати шляхом введення тексту.

Однак, в даних експериментах дії кліку миші та натиску клавіші клавіатури вважаються рівноцінними, хоча в реальності це не так. Перед натиском на кнопку миші, потрібно попередньо навести курсор на потрібний елемент, перед чим знайти його на екрані, що займає час. Для більшої точності інтерпретації результатів експериментів можна в майбутньому використати деякі ваги для кожної дії введення від користувача.

Хоча, в будь якому випадку, дані експерименти показують що текстовий інтерфейс вимагає більше дій від користувача ніж графічний. Поглянувши на вхідний текст опису трафіку, що використовувався в експериментах, можна помітити напрямки покращення граматики для зменшення кількості дій, що вимагаються від користувача.

А саме, в обох експериментах помітно, що часто відбувається доступ до атрибутів однієї і тієї ж змінної. Наприклад:

```
TcpStream s
s.src_port=1337
s.dst_port=80
s.mac.src="01:23:45:67:89:ab"
s.mac.dst="cd:ef:1f:ff:ff:ff"
s.ipv4.src=192.168.1.1
s.ipv4.dst=192.168.2.1
s.add(C2S,SYN)
s.add(S2C,SYN,ACK)
s.add(C2S,ACK)
s.add(C2S,"GET / HTTP/1.1\nHost: some.domain.net")
s.add(...
```

Як спосіб покращення граматики, можна додати можливість опускати назву змінної, до якої послідовно виконується доступ. Наприклад, зробити так щоб коли рядок починається із символу крапки, це вважалось доступом до атрибуту першої змінної, до якої відбувався доступ в попередньому рядку. Синтаксис із подібним функціоналом міг би виглядати ось так:

```
TcpStream s
s.src_port=1337
.dst_port=80
.mac.src="01:23:45:67:89:ab"
.mac.dst="cd:ef:1f:ff:ff:ff"
.ipv4.src=192.168.1.1
.ipv4.dst=192.168.2.1
.add(C2S,SYN)
.add(S2C,SYN,ACK)
.add(C2S,ACK)
.add(C2S,"GET / HTTP/1.1\nHost: some.domain.net")
.add(...
```

Із подібними покращеннями до синтаксису, в даному випадку необхідність введення змінної «s» зменшилась на десять разів. Якби ім’я цієї змінної було довшим, це покращення було б досить відчутним.

Також, як варіант, можна додати особливі види змінних, які представляли б «значення за замовчуванням» або «випадкові» значення, адже користувачу може не обов’язково завжди потрібно використовувати конкретні значення.

В цілому, бачимо що граматика для опису трафіку має простір для покращення.

Тобто, бачимо що за рахунок своєї гнучкості при додавання нового функціоналу, розроблений текстовий інтерфейс опису трафіку здатен мати перевагу над існуючими аналогами. А можливі покращення граматики опису трафіку здатні покращити її ефективність порівняно із аналогами навіть без додавання нових функцій опису, хоча ефективності одного кліку миші їй і не досягти.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Postman API platform [Електронний ресурс]. Режим доступу: https://www.postman.com (Дата звернення: 07.01.2024);

2. ping(8) — Linux manual page [Електронний ресурс]. Режим доступу: https://www.man7.org/linux/man-pages/man8/ping.8.html (Дата звернення: 07.01.2024);

3. Ostinato traffic generator [Електронний ресурс]. Режим доступу: https://ostinato.org (Дата звернення: 07.01.2024);

4. Flex — a scanner generator [Електронний ресурс]. Режим доступу: https://ftp.gnu.org/old-gnu/Manuals/flex-2.5.4 (Дата звернення: 07.01.2024);

5. Catch2 testing framework [Електронний ресурс]. Режим доступу: https://catch2-temp.readthedocs.io/en/latest (Дата звернення: 07.01.2024);

6. Bash Automated Testing System [Електронний ресурс]. Режим доступу: https://github.com/bats-core/bats-core (Дата звернення: 07.01.2024).

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ

Проректор
Українського державного університету науки і технології
Анатолій РАДКЕВИЧ

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01354-01-ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Олена КУРОП'ЯТНИК

Виконавець

_____Андрій СЕРБЕНЮК

Нормоконтролер

_____Світлана ВОЛКОВА

2023

ЗАТВЕРДЖЕНО

44165850.01354-01

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Технічне завдання

Листів 13

2023

ЗМІСТ

ВСТУП

Програмний продукт «Дослідження ефективності використання формальних граматик для формування мережевого трафіку» призначений для формування довільного мережевого трафіку.

Причина виникнення необхідності розробки програмного забезпечення полягає у відсутності програмних аналогів, які формували б трафік із використанням власних розроблених формальних граматик.

Області, в який передбачається використання програмного продукту — проведення досліджень із формування мережевого трафіку із використанням формальних граматик, тестування програм або програмних систем, які потребують сформованого вручну трафіку.

А.1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Пшінько О.М. «Про затвердження тем та призначення керівників дипломних проектів» №1196 ст від 05.12.2022 року.

Тема проекту: «Дослідження ефективності використання формальних граматик для формування мережевого трафіку».

Керівник дипломного проекту: Куроп’ятник О.С.

А.2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення — формування довільного мережевого трафіку, заданого в текстовому вигляді за синтаксисом розробленої формальної граматики.

Експлуатаційне призначення — можливість уніфікації опису трафіку різних протоколів різних рівнів.

А.3 ВИМОГИ ДО ПРОГРАМИ

А.3.1 Вимоги до функціональних характеристик

Програма матиме консольний інтерфейс.

Програма повинна формувати мережевий трафік, описаний текстом за синтаксисом, що відповідає розробленій граматиці.

Програма повинна виконувати наступні функції:

- отримання вхідних даних від користувача;
- обробка команд в поданому користувачем тексті, зчитування та обробка лексем в ньому;
- підвантаження плагінів та їх подальше використання при роботі.

Для цього потрібно розробити граматику, синтаксис якої дозволить описувати мережевий трафік.

Плагіни представлятимуть собою динамічні бібліотеки, які підвантажуватимуться в програму під час її роботи, перед початком обробки вхідного тексту. Плагіни міститимуть функціонал, що здатен розширювати можливості програми із опису трафіку, розуміння команд у вхідному тексті.

Вхідні дані:

- текст із командами, написаний за розробленою граматикою;
- шлях із розташуванням плагінів та самі плагіни у вигляді файлів динамічних бібліотек.

Текст із командами подається у стандартний потік введення програми, що дозволить користувачу вводити його вручну, та перенаправляти його в програму із вихідних потоків інших програм, або файлів.

Шлях із розташуванням плагінів потрібно занести в змінну середовища із назвою «TRAFFIC_PLUGINS_PATH». Якщо шлях до плагінів не вказаний, програма не намагатиметься їх завантажувати.

Вихідні дані:

- код виходу програми: 0 — програма завершилась успішно;
- вихідні дані окремих команд із вхідного тексту: повідомлення користувачу, файли з описом сформованого трафіку.

Обробка команд із вхідного тексту передбачає виконання деяких дій, серед яких можуть бути виведення повідомлень користувачу у вихідний текстовий потік, створення файлів, тощо.

А.3.2 Вимоги до надійності

Для забезпечення надійного функціонування необхідно забезпечити наявність архівного коду тексту програми на зовнішньому носії.

А.3.3 Умови експлуатації

Для забезпечення надійної роботи програмного продукту користувачу необхідно дотримуватись таких умов:

- програмний засіб повинен використовуватись у приміщеннях, призначених для роботи з ЕОМ з відповідними кліматичними умовами;
- працювати з програмним засобом може людина, що має навички роботи з ПК та ознайомлена з керівництвом користувача.

А.3.4 Вимоги до складу і параметрів технічних засобів

Програмний продукт розрахований для функціонування на пристроях, що мають такі мінімальні характеристики:

- процесор: Intel Core 2 Duo;
- місце на диску: 256 Mb;
- об’єм оперативної пам’яті: 2 Gb;
- маніпулятори: клавіатура.

А.3.5 Вимоги до інформаційної і програмної сумісності

Для роботи програмного продукту на ЕОМ має бути встановлене наступне програмне забезпечення:

- операційна система: UNIX-подібна операційна система, основана на ядрі Linux, FreeBSD, macOS.
- бібліотека Flex[1];

Необов’язкові бібліотеки та програми, що використовуються для тестування програми:

- Catch2[2];
- bats[3].

А.4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- керівництво користувача;
- текст програми.

Вся документація до програмного забезпечення повинна задовольняти вимоги державного стандарту до оформлення програмної документації ГОСТ 19.101-77.

А.5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки наведені в табл. 5.1.

Таблиця А.5.1 — Стадії та етапи розробки

Стадія	Зміст робіт	Термін виконання
Технічне завдання	Постановка завдання, збір інформації, викладення та обґрунтування критеріїв розробки; попередній вибір методів вирішення задач; визначення вимог до технічних засобів; узгодження та затвердження технічного завдання.	04.09.2023-03.12.2023
Робочий проєкт	Проєктування програми, написання програмного коду, тестування	04.12.2023-31.12.2023
	Розробка, узгодження та затвердження програмної документації	01.01.2024-14.01.2024

А.6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль за виконанням роботи здійснює керівник розробки Куроп’ятник О. С.

Прийом здійснюється комісією у складі, визначеному університетом.

БІБЛОГРАФІЧНИЙ СПИСОК

1. Flex — a scanner generator [Електронний ресурс]. Режим доступу: https://ftp.gnu.org/old-gnu/Manuals/flex-2.5.4 (Дата звернення: 07.01.2024);

2. Catch2 testing framework [Електронний ресурс]. Режим доступу: https://catch2-temp.readthedocs.io/en/latest (Дата звернення: 07.01.2024);

3. Bash Automated Testing System [Електронний ресурс]. Режим доступу: https://github.com/bats-core/bats-core (Дата звернення: 07.01.2024);

4. Mastering Cmake [Електронний ресурс]. Режим доступу: https://cmake.org/cmake/help/book/mastering-cmake (Дата звернення: 07.01.2024).

ДОДАТОК Б

Керівництво користувача

ЗАТВЕРДЖУЮ

Проректор
Українського державного університету науки і технології
Анатолій РАДКЕВИЧ

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Керівництво користувача
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01354-01 І3 01 ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Олена КУРОП’ЯТНИК

Виконавець

_____Андрій СЕРБЕНЮК

Нормоконтролер

_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.01354-01 13 01

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Керівництво користувача

Листів 17

2024

ЗМІСТ

ВСТУП

Сфера застосування. Області, в який передбачається використання програмного продукту — проведення досліджень із формування мережевого трафіку із використанням формальних граматик, тестування програм або програмних систем, які потребують сформованого вручну трафіку.

Короткий опис можливостей. Програма приймає на вхід текст, написаний відповідно до розробленої граматики опису мережевого трафіку, та виконує команди, написані в ньому. Доступні в мові команди націлені на опис мережевого трафіку — представлення частин трафіку, операцій його формування та запису.

Програма реалізує можливість використання плагінів, що додають новий функціонал із опису трафіку. Наприклад, плагіни для опису деяких мережевих протоколів.

Рівень підготовки користувача. Для роботи із програмою користувач повинен мати змогу вільно користуватись клавіатурою, бути ознайомленим із оперуванням UNIX-подібних операційних систем, бути ознайомленим із мережевим протоколом, трафік якого описуватиметься.

Для встановлення програми користувач повинен бути ознайомленим із механізмом компіляції та інсталяції програм із використанням інстументів CMake[1].

Б.1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Б.1.1 Види діяльності, для яких призначено цей засіб

Програмний продукт «Використання формальних граматик для формування мережевого трафіку» призначений для формування довільного мережевого трафіку. Це відбувається шляхом написання тексту опису трафіку.

Б.1.2 Умови використання

Програмний засіб розроблений для використання на сімействі UNIX-подібних операційних систем. Програма має консольний інтерфейс.

Програма отримує вхідні дані із стандартного вхідного потоку. Механізми керування стандартними вхідними потоками в UNIX-подібних операційних системах дозволяють вводити вхідні дані вручну, перенапрявляти із файлу, або із виходу іншої програми.

Б.2 ПІДГОТОВКА ДО РОБОТИ

Б.2.1 Дистрибуція програми

Програма поширюється у вигляді програмного її програмного коду, який необхідно зкомпілювати, та встановити результуючі файли.

Б.2.2 Порядок завантаження даних і програм

Для встановлення програми її спершу потрібно зкомпілювати. Для цього на ЕОМ повинні бути присутні інструменти компіляції CMake, компілятор, бібліотеки flex[2], catch2[3].

Після отримання програмного коду, необхідно перейти в кореневу папку із кореневим CMakeLists.txt файлом. В цьому місці достатньо ввести в консолі наступні команди:

```
cmake -S . -B build
cd build
make install
```

На даному етапі також компілюється і встановлюється плагін із підтримкою опису трафіку протоколу TCP. Щоб програма могла його використовувати, необхідно вказати шлях до нього в змінній середовища «TRAFFIC_PLUGINS_PATH». Наприклад:

```
TRAFFIC_PLUGINS_PATH=/home/user/traffic_grammar/build/install/lib
```

Б.2.3 Порядок перевірки працездатності

Після встановлення програми, для перевірки її працездантості можна ввести в тій же папці build команду ctest.

Приклад успішного виклику та завершення команди ctest:

```
[user@machine build]$ ctest
Test project /home/user/traffic_grammar/build
    Start 1: Tokenizer tests
1/2 Test #1: Tokenizer tests .....   Passed    0.01 sec
```

```
    Start 2: Parser framework
2/2 Test #2: Parser framework .....   Passed    0.02 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   0.03 sec
```

Б.3 ОПИС ОПЕРАЦІЙ

Програма отримує на вхід в стандартний вхідний потік текст із командами опису трафіку. Синтаксис, якому повинен відповідати вхідний текст, описаний наступною формою Бекуса-Наура:

<Application> ::= <Line><Application>?

<Line> ::= <Command><Comment>?<Newline>

<Newline> ::= "\r\n|\n"

<Whitespace> ::= [" ""\t"]

<PrintableChar> ::= [^<Newline>]

<Comment> ::= "#"<PrintableChar>*

<Command> ::= <Declaration>|<Assignment>|<MembAccess>

<VarName> ::= [a-zA-Z_\-][a-zA-Z0-9_\-]*

<Declaration> ::= <Type><Whitespace><VarName><AssignmentOp>?

<Assignment> ::= <Var><AssignmentOp>

<AssignmentOp> ::= <Whitespace>*"="<Whitespace>*<Value>

<MembAccess> ::= <VarName><MembAccessOp>

<MembAccessOp> ::= "."<Var>

<Var> ::= <VarName>|<MembAccess>|<FuncCall>

<Value> ::= <Int>|<Ipv4Addr>|<String>|<Bool>|<Var>

<Number> ::= [0-9]+

<Int> ::= -?<Number>

<Ipv4Addr> ::= <Number>"."<Number>"."<Number>"."<Number>

<String> ::= "\""<StringContents>"\""

<StringContents> ::= [<PrintableChar>|<EscapedChar>|<Newline>]*

<EscapedChar> ::= "\\"[nr\\"]

<FuncCall> ::= <Var>"("<ArgList>?")"

<ArgList> ::= <Value><Whitespace>*(","<Whitespace>*<Value>)*

Синтаксис підтримує операції оголошення змінної, присвоєння, доступ до атрибутів, вивклик функції. Доступ до функціоналу опису мережевого трафіку відбувається із використанням типів, доступних в даній мові.

За замовчуванням, в мові доступні наступні стандартні типи:

- Int — ціле число в діапазоні [-9,223,372,036,854,775,808; 9,223,372,036,854,775,807];
- String — текстовий рядок;
- IPv4Value — адреса протоколу Ipv4:
- Function — функція, яку можна викликати. Для цього типу не передбачене присвоєння значень.

Плагін із підтримкою протоколу TCP, що постачається разом із основною програмною, описує наступні типи:

TcpStream — зберігає пакети сесії, відповідає за присвоєння пакетам значень, що ідентифікують сесію.

Атрибути:

- mac — інформація про MAC адреси користувачів сесії;
- ip — інформація про IP адреси користувачів сесії;
- src_port — номер порту відправника;
- dst_port — номер порту одержувача;

Методи:

- add(packet, amount) — додати в сесію попередньо визначений пакет. В цьому випадку, значення пакету не змінюватимуться, і його можна вставити довільну кількість разів;
- add(direction, flags, payload) — додати в сесію пакет із заданими параметрами. Пакет створюється всередині методу, йому автоматично присвоюються значення деяких полів;
- to_pcap(path) — записати сформований трафік в файл формату захоплення пакетів pcap. На даному етапі розробки, перетворення внутрішнього

представлення пакету на дані справжнього мережевого пакету не реалізоване. Натомість, зараз ця функція виводить повідомлення про те, що її потрібно реалізувати;

- to_string() — формує текстовий рядок, що містить інформацію про всі пакети в сесії.

TcpPacket — представляє об’єкт TCP пакету.

Атрибути:

- mac — інформація про MAC адреси відправника та одержувача;
- ip — інформація про IP адреси відправника та одержувача;
- dir — напрямок руху пакету, від ініціатора сесії, чи до нього;
- src_port — порт відправника пакету;
- dst_port — порт отимувача пакету;
- seq_num — номер послідовності пакету;
- ack_num — номер підтвердження пакету;
- flags — прапорці пакету;
- payload — дані пакету.

Методи:

- to_string() — формує текстовий рядок із інформацією про пакет.

MacValue — представляє MAC адресу

Методи:

- to_string() — формує текстовий рядок із MAC адресою.

MacInfo — зберігає інформацію про MAC адреси сесії.

Атрибути:

- src — MAC адреса відправника пакету;
- dst — MAC адреса одержувача пакету.

IPv4Info - зберігає інформацію про IPv4 адреси сесії.

Атрибути:

- src — IPv4 адреса відправника пакету;

- dst — IPv4 адреса одержувача пакету.

TcpFlags — зберігає інформацію про встановлені прапорці пакету. Всього є шість можливих прапорців: FIN, SYN, RST, PSH, ACK, URG. Ці прапорці повинні бути доступними у вигляді вбудованих змінних із відповідними значеннями. Один об’єкт TcpFlags може встановлювати довільну кількість прапорців.

Методи:

- set(flags) — встановити значення прапорців об’єкту. Нові прапорці додаються до вже існуючих;
- unset(flags) — вилучити у об’єкту вказані прапорці.
- to_string() — сформувати текстовий рядок із інформацією про встановлені прапорці.

Direction — зберігає інформацію про напрямок руху пакету в сесії. Може набувати двох можливих значень: C2S (Client-To-Server), S2C (Server-To-Client). Ці значення повинні бути доступними у вигляді вбудованих змінних із відповідними значеннями.

Методи:

- set() — встановити значення напрямку;
- to_string() — сформувати текстовий рядок із напрямком руху.

Тобто, плагін для формування трафіку за протоколом TCP надаватиме програмі наступне:

- типи даних:
  - TcpStream;
  - TcpPacket;
  - MacValue;
  - MacInfo;
  - IPv4Info;
  - TcpFlags;

    - Direction;
- вбудовані змінні:
    - FIN: TcpFlags;
    - SYN: TcpFlags;
    - RST: TcpFlags;
    - PSH: TcpFlags;
    - ACK: TcpFlags;
    - URG: TcpFlags;
    - C2S: Direction;
    - S2C: Direction.

Б.4 АВАРІЙНІ СИТУАЦІЇ

Програма не зберігає стану виконання, тому їй не загрожують ситуації втрати даних.

У випадку, коли вхідний текст містить помилки, програма виводитиме помилки тільки у випадку, коли вона зкомпільована із ввімкненими повідомленнями відлагодження. Це недолік, який повинен бути виправленим в майбутньому. Для ввімкнення повідомлень відлагодження, перед компіляцією необхідно змінити програмний код, а саме — значення змінної «debug_print» в файлі tokenizer.h на рядку 21 із значення false на true.

У випадку, коли виклик функції завершився невдало — відповідальність за сповіщення користувача лежить на функції.

Б.5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Для конфігурації середовища виконання програми, можна задавати значення потрібних змін середовища в системних скриптах що запускаються при запуску терміналу (наприклад, «.bashrc» у випадку коли використовуєтсья bash[4]). Таким чином, можна автоматично присвоювати значення із шляхом до плагінів. Наприклад:

```
export TRAFFIC_PLUGINS_PATH=/home/user/traffic_grammar/build/install/lib
```

Також, можна додати шлях до виконуваного файлу програми в змінну PATH. Наприклад:

```
export PATH=$PATH:/home/user/traffic_grammar/build/install/bin
```

Б.5.1 Контрольний приклад

Нехай, маємо файл із найзвою traffic.txt, що містить текст із описом трафіку. Текст опису трафіку наступний:

```
TcpStream s

s.src_port = 1337
s.dst_port = 80

s.mac.src = "01:23:45:67:89:ab"
s.mac.dst = "cd:ef:1f:ff:ff:ff"

s.ip.src = 192.168.1.1
s.ip.dst = 192.168.2.1

# A simple HTTP1 session over TCP

s.add(C2S, SYN)
s.add(S2C, SYN, ACK)
s.add(C2S, ACK)

s.add(C2S, "GET / HTTP/1.1\nHost: some.domain.net")
s.add(S2C, ACK, "HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
Content-Length: 12
Content-Type: text/html
```

```
Hello world!")
s.add(C2S, ACK)

s.add(C2S, FIN)
s.add(S2C, FIN, ACK)
s.add(C2S, ACK)

print(s.to_string())
```

В цьому випадку, запуск програми може виглядати наступним чином:

```
$ traffic_grammar < traffic.txt
```

В результаті виконання програми, в стандатний вихідний потік буде виведено наступний текст:

```
192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000000010
 SYN
Seq: 1, Ack: 0

192.168.2.1:80 -> 192.168.1.1:1337
Flags: 0000000000010010
 SYN ACK
Seq: 2, Ack: 1

192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000010000
 ACK
Seq: 3, Ack: 2

192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000000000
Seq: 4, Ack: 0

192.168.2.1:80 -> 192.168.1.1:1337
Flags: 0000000000010000
 ACK
Seq: 5, Ack: 4

192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000010000
 ACK
Seq: 6, Ack: 5

192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000000001
 FIN
Seq: 7, Ack: 0
```

192.168.2.1:80 -> 192.168.1.1:1337
Flags: 0000000000010001
FIN ACK
Seq: 8, Ack: 7

192.168.1.1:1337 -> 192.168.2.1:80
Flags: 0000000000010000
ACK
Seq: 9, Ack: 8

БІБЛОГРАФІЧНИЙ СПИСОК

1. Flex — a scanner generator [Електронний ресурс]. Режим доступу: https://ftp.gnu.org/old-gnu/Manuals/flex-2.5.4 (Дата звернення: 07.01.2024);

2. Catch2 testing framework [Електронний ресурс]. Режим доступу: https://catch2-temp.readthedocs.io/en/latest (Дата звернення: 07.01.2024);

3. Bash Automated Testing System [Електронний ресурс]. Режим доступу: https://github.com/bats-core/bats-core (Дата звернення: 07.01.2024);

4. Mastering Cmake [Електронний ресурс]. Режим доступу: https://cmake.org/cmake/help/book/mastering-cmake (Дата звернення: 07.01.2024).

ДОДАТОК В

Текст програми

ЗАТВЕРДЖУЮ

Проректор
Українського державного
університету науки і
технології
Анатолій РАДКЕВИЧ

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01354-01 12 01 ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Олена КУРОП’ЯТНИК

Виконавець

_____Андрій СЕРБЕНЮК

Нормоконтролер

_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО
44165850.01354-01 12 01

ВИКОРИСТАННЯ ФОРМАЛЬНИХ ГРАМАТИК ДЛЯ ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

Текст програми

44165850.01354-01 12 01

Листів 53

2024

АНОТАЦІЯ

Документ 44165850.01354-01 12 01 «Використання формальних граматик для формування мережевого трафіку. Текст програми».

Об’єкт дослідження – використання формальних граматик для формування мережевого трафіку.

Мета роботи – Розробка додатку для формування мережевого трафіку із використанням формальних граматик.

ЗМІСТ

В.1 ТЕКСТ ПРОГРАМИ

В.1.1 Головна програма

В.1.1.1 main.cc

```
#include <iostream>

#include "parser/tokenizer.h"

int main(int, char**)
{
   std::ostream out(nullptr);

   Tokenizer t(std::cin, out);
   t.process();

   return 0;
}
```

В.1.1.2 tokenizer.h

```
#ifndef TOKENIZER_H
#define TOKENIZER_H

#if ! defined(yyFlexLexerOnce)
#include <FlexLexer.h>
#endif
#include <stack>

#include "standard_types.h"
#include "plugins_management.h"

class Tokenizer : public yyFlexLexer
{
public:
   enum Result
   {
      VALID = 0,
      INVALID
   };

   bool debug_print = false;

   Tokenizer(std::istream& in, std::ostream& out);
   ~Tokenizer();

   int process();
   const TypeAPI* string_api = nullptr;
      bool load_custom_plugin(const char* lib_path);

   private:
      enum AssignmentStatus : unsigned short
      {
         ASSIGN_OK = 0,
         ASSIGN_WRONG_TYPE,
         ASSIGN_INVALID_VALUE
      };

      VarAdministrator vadmin;
      PluginsAdministrator padmin;
      std::string last_match;
      std::stack<Variable*> var_stack;
      std::stack<std::unique_ptr<Variable>> func_ret_vals;
      std::stack<std::vector<Variable*>> func_args;
      // saving literals temporarily probably isn't the best idea
performance-wise,
      // but for now it saves effort for actually coming up with a
way to store them.
      std::vector<std::unique_ptr<Variable>> tmp_literals;
      Variable* last_accessed = nullptr;
      size_t func_call_level = 0;

      // built-in types
      const TypeAPI* int_api = nullptr;
      const TypeAPI* ipv4_api = nullptr;
```

```
    const TypeAPI* func_api = nullptr;

    unsigned last_decl_type = 0;

    int yylex() override;

    bool init_var_proc();
    bool decl_check_type();
    unsigned short declare_var();
    bool assign_variable();
    unsigned short assign_literal(const TypeAPI& type);
    unsigned short assign_int();
    unsigned short assign_string();
    unsigned short assign_ipv4();
    bool get_variable();
    bool get_member(Variable*& out);
    bool proc_arg_variable(const char* name);
    unsigned short proc_literal_arg(const TypeAPI& type);
    unsigned short proc_arg_int();
    unsigned short proc_arg_string();
    unsigned short proc_arg_ipv4();
    bool init_func_call();
    bool call_func();

    friend class ParserTestDriver;
};

#endif
```

B.1.1.3 tokenizer.l

```
%option c++
%option yyclass="Tokenizer"
%option align full 8bit batch never-interactive
%option noinput nounput noyywrap

%{
#include <iostream>
#include "tokenizer.h"

#define DBG_PRINT(text)                         \
    if (debug_print)                            \
        std::cout << "[DEBUG]: " << text << std::endl;

#define DBG_PRINT_VALUE(label, value)           \
    if (debug_print)                            \
        std::cout << "[DEBUG]: " << label << ": "       \
            << value << std::endl;

%}

/* Allowed characters */
NEWLINE         \r\n|\n
PRINTABLE_CHAR  .
WHITESPACE      [ \t]

COMMENT         #{PRINTABLE_CHAR}*

/* Value types */
NUMBER          [[:digit:]]+
INT             -?{NUMBER}
IPV4ADDR        {NUMBER}.{NUMBER}.{NUMBER}.{NUMBER}
ESCAPED_CHAR    \\[nr\\\"]
STRING_CONTENTS ([^\"]|{ESCAPED_CHAR})*
STRING          \"{STRING_CONTENTS}\"
VAR_NAME        [a-zA-Z_][[:alnum:]_\-]*

/* Command processing states */
%x var_start
%x var_generic
%x assignment
%x assign_var
%x declaration
%x memb_access
%x arg_list_start
%x arg_list
%x arg_var
%x arg_more
%x decl_done
%x command_end

%%

<INITIAL>{WHITESPACE}|{NEWLINE} {/*skip*/}

<INITIAL>{VAR_NAME} {
    DBG_PRINT_VALUE("INIT VAR_NAME", yytext);
    last_match = yytext;
    init_var_proc();
    BEGIN(var_start);
}

<var_start,var_generic,decl_done>{WHITESPACE}*={WHITESPACE}* {
    // assignment detected
```

```
    DBG_PRINT("ASSIGNMENT START");
    if (var_stack.empty() && !get_variable())
    {
        DBG_PRINT("Variable wasn't declared");
        return INVALID;
    }
    BEGIN(assignment);
}

<var_start>{WHITESPACE}+ {
    // declaration detected
    DBG_PRINT("DECLARATION START");

    if (!decl_check_type())
    {
        DBG_PRINT_VALUE("Unsupported type", last_match);
        return INVALID;
    }

    BEGIN(declaration);
}

<var_start>\. {
    // Variable member access
    BEGIN(var_generic);
    DBG_PRINT("MEMBER ACCESS START");
    if (var_stack.empty() && !get_variable())
    {
        DBG_PRINT("Variable doesn't list this member");
        return INVALID;
    }
    yy_push_state(memb_access);
}

<var_start>\( {
    BEGIN(var_generic);
    DBG_PRINT("METHOD CALL START");
    if (var_stack.empty() && !get_variable())
    {
        DBG_PRINT("This function wasn't declared");
        return INVALID;
    }
    init_func_call();
    yy_push_state(arg_list_start);
}

<var_generic,assign_var,arg_var>\. {
    // Variable member access
    DBG_PRINT("MEMBER ACCESS START");
    if (var_stack.empty() && !get_variable())
    {
        DBG_PRINT("Variable wasn't declared");
        return INVALID;
    }
    yy_push_state(memb_access);
}

<var_generic,assign_var,arg_var>\( {
    DBG_PRINT("METHOD CALL START");
    init_func_call();
    yy_push_state(arg_list_start);
}

<arg_list_start,arg_list,arg_more,arg_var>{WHITESPACE}
{/*skip*/}

<arg_list_start,arg_more,arg_var>\) {
    DBG_PRINT("METHOD CALL END");
    if (!call_func())
    {
        DBG_PRINT("Error while calling function");
        return INVALID;
    }
    yy_pop_state();
}

<arg_list_start,arg_list>{VAR_NAME} {
    DBG_PRINT_VALUE("ARG VARIABLE", yytext);

    if (!proc_arg_variable(yytext))
    {
        DBG_PRINT("Variable wasn't declared");
        return INVALID;
    }

    BEGIN(arg_var);
}

<arg_list_start,arg_list>{INT} {
    DBG_PRINT_VALUE("ARG INT", yytext);

    switch (proc_arg_int())
    {
        case VarAdministrator::VarRet::VAR_OK:
            break;
        case VarAdministrator::VarRet::VAR_EXISTS:
            // a valid case
            break;
```

```
      case VarAdministrator::VarRet::VAR_ERR_GENERIC:
      default:
        DBG_PRINT("Variable declaration error");
        return INVALID;
        break;
    }

    BEGIN(arg_more);
}

<arg_list_start,arg_list>{IPV4ADDR} {
    DBG_PRINT_VALUE("ARG IP", yytext);

    switch (proc_arg_ipv4())
    {
      case VarAdministrator::VarRet::VAR_OK:
        break;
      case VarAdministrator::VarRet::VAR_EXISTS:
        // a valid case
        break;
      case VarAdministrator::VarRet::VAR_ERR_GENERIC:
      default:
        DBG_PRINT("Variable declaration error");
        return INVALID;
        break;
    }

    BEGIN(arg_more);
}

<arg_list_start,arg_list>{STRING} {
    DBG_PRINT_VALUE("ARG STRING", yytext);

    switch (proc_arg_string())
    {
      case VarAdministrator::VarRet::VAR_OK:
        break;
      case VarAdministrator::VarRet::VAR_EXISTS:
        // a valid case
        break;
      case VarAdministrator::VarRet::VAR_ERR_GENERIC:
      default:
        DBG_PRINT("Variable declaration error");
        return INVALID;
        break;
    }

    BEGIN(arg_more);
}

    <arg_var>{WHITESPACE}+ {
      BEGIN(arg_more);
    }

    <arg_more,arg_var>\, {
      BEGIN(arg_list);
    }

    <memb_access>{VAR_NAME} {
      DBG_PRINT_VALUE("ACCESS TO SUB-MEMBER",
yytext);
      Variable* member = nullptr;
      if (!get_member(member))
      {
        DBG_PRINT("There is no such member on this
variable");
        return INVALID;
      }
      yy_pop_state();
    }

    <assignment>{VAR_NAME} {
      // assigned another variable
      DBG_PRINT_VALUE("ASSIGN VARIABLE", yytext);
      last_match = yytext;
      if (!get_variable())
      {
        DBG_PRINT("Variable wasn't declared");
        return INVALID;
      }
      BEGIN(assign_var);
    }

    <assign_var>{WHITESPACE}*{COMMENT} {
      DBG_PRINT("COMMENT");
      if (!assign_variable())
      {
        DBG_PRINT("Assignment error");
        return INVALID;
      }
      BEGIN(INITIAL);
    }

    <assign_var>{WHITESPACE}*$ {
      DBG_PRINT("Command line ended");
      if (!assign_variable())
      {
        DBG_PRINT("Assignment error");
```

```
            return INVALID;
        }
        BEGIN(INITIAL);
    }

    <assign_var><<EOF>> {
        if (!assign_variable())
        {
            DBG_PRINT("Assignment error");
            return INVALID;
        }
        DBG_PRINT("EOF");
        return VALID;
    }

    <assignment>{INT} {
        DBG_PRINT_VALUE("ASSIGN INT", yytext);

        switch (assign_int())
        {
        case ASSIGN_OK:
            break;
        case ASSIGN_WRONG_TYPE:
            DBG_PRINT("Can't assign int to this var");
            return INVALID;
            break;
        case ASSIGN_INVALID_VALUE:
            DBG_PRINT("Invalid value");
            return INVALID;
            break;
        default:
            DBG_PRINT("Something went wrong when trying to
assign int");
            return INVALID;
        };

        BEGIN(command_end);
    }

    <assignment>{IPV4ADDR} {
        DBG_PRINT_VALUE("ASSIGN IP", yytext);

        switch (assign_ipv4())
        {
        case ASSIGN_OK:
            break;
        case ASSIGN_WRONG_TYPE:
            DBG_PRINT("Can't assign IPv4 to this var");
            return INVALID;
            break;
        case ASSIGN_INVALID_VALUE:
            DBG_PRINT("Invalid value");
            return INVALID;
            break;
        default:
            DBG_PRINT("Something went wrong when trying to
assign IPv4");
            return INVALID;
        };

        BEGIN(command_end);
    }

    <assignment>{STRING} {
        DBG_PRINT_VALUE("ASSIGN STRING", yytext);

        switch (assign_string())
        {
        case ASSIGN_OK:
            break;
        case ASSIGN_WRONG_TYPE:
            DBG_PRINT("Can't assign string to this var");
            return INVALID;
            break;
        case ASSIGN_INVALID_VALUE:
            DBG_PRINT("Invalid value");
            return INVALID;
            break;
        default:
            DBG_PRINT("Something went wrong when trying to
assign string");
            return INVALID;
        };

        BEGIN(command_end);
    }

    <declaration>{VAR_NAME} {
        // new variable name
        DBG_PRINT_VALUE("DECLARATION VAR_NAME",
yytext);

        switch (declare_var())
        {
            case VarAdministrator::VarRet::VAR_OK:
                break;
            case VarAdministrator::VarRet::VAR_EXISTS:
                DBG_PRINT("Variable already exists");
```

```
            return INVALID;
            break;
        case VarAdministrator::VarRet::VAR_ERR_GENERIC:
        default:
            DBG_PRINT("Variable declaration error");
            return INVALID;
            break;
      }
      BEGIN(decl_done);
    }

    <INITIAL,command_end,decl_done>{WHITESPACE}*{CO
MMENT} {
      DBG_PRINT("COMMENT");
      BEGIN(INITIAL);
    }

    <command_end,decl_done,var_generic>{WHITESPACE}*$ {
      DBG_PRINT("Command line ended");
      BEGIN(INITIAL);
    }

    <INITIAL,command_end,decl_done,var_generic><<EOF>> {
      DBG_PRINT("EOF");
      return VALID;
    }

    <INITIAL,var_start,var_generic,assignment,declaration,memb
_access,arg_list,arg_list_start,arg_var,arg_more,command_end,decl_
done,assign_var>. {
      DBG_PRINT_VALUE("Unknown token", yytext);
      return INVALID;
    }

    <var_start,assignment,declaration,memb_access,arg_list,arg_li
st_start,arg_var,arg_more><<EOF>> {
      DBG_PRINT("Unexpected EOF");
      return INVALID;
    }

    %%
```

B.1.1.4 tokenizer.cpp

```
#include "tokenizer.h"
#include "type_management.h"

Tokenizer::Tokenizer(std::istream& in, std::ostream& out)
  : yyFlexLexer(in, out), padmin(vadmin)
{
  padmin.load_plugin(&built_ins);
  int_api = TAdmin::get_api(INT_NAME);
  string_api = TAdmin::get_api(STRING_NAME);
  ipv4_api = TAdmin::get_api(IPv4_NAME);
  func_api = TAdmin::get_api(FUNC_TYPE_NAME);

  // load plugins installed on the system
  padmin.load_so_plugins();
}

int Tokenizer::process()
{
  return yylex();
}

bool Tokenizer::init_var_proc()
{
  // clear out all processing buffers for new line
  var_stack = std::stack<Variable*>();
  func_args = std::stack<std::vector<Variable*>>();
  func_ret_vals = std::stack<std::unique_ptr<Variable>>();
  tmp_literals.clear();

  return true;
}

bool Tokenizer::decl_check_type()
{
  unsigned type =
TypeAdministrator::get_type_code(last_match.c_str());

  if (!type)
    return false;

  last_decl_type = type;
  return true;
}

unsigned short Tokenizer::declare_var()
{
  TypeAPI* tapi =
TypeAdministrator::get_api(last_decl_type);
  if (!tapi)
    return VarAdministrator::VarRet::VAR_WRONG_TYPE;
```

```
        std::pair<unsigned short, Variable*> added =
vadmin.add_var(*tapi, yytext);
        if (added.first == VarAdministrator::VarRet::VAR_OK)
            var_stack.push(added.second);

        return added.first;
    }

    bool Tokenizer::assign_variable()
    {
        Variable* r = var_stack.top();
        var_stack.pop();
        Variable* l = var_stack.top();
        var_stack.pop();
        return l->assign(*r);
    }

    unsigned short Tokenizer::assign_literal(const TypeAPI &type)
    {
        if (!var_stack.top()->assign(yytext))
            return ASSIGN_INVALID_VALUE;
        return ASSIGN_OK;
    }

    unsigned short Tokenizer::assign_int()
    {
        return assign_literal(*int_api);
    }

    unsigned short Tokenizer::assign_string()
    {
        return assign_literal(*string_api);
    }

    unsigned short Tokenizer::assign_ipv4()
    {
        return assign_literal(*ipv4_api);
    }

    bool Tokenizer::get_variable()
    {
        Variable* var = vadmin.get_var(last_match.c_str());

        if (!var)
            return false;

        var_stack.push(var);
        last_accessed = nullptr;
        return true;
    }

    bool Tokenizer::get_member(Variable *&out)
    {
        Variable* member = func_call_level > 0 ?
            func_args.top().back()->get_member(yytext) :
            var_stack.top()->get_member(yytext);

        if (!member)
            return false;

        out = member;
        if (func_call_level > 0)
        {
            last_accessed = *func_args.top().rbegin();
            *func_args.top().rbegin() = member;
        }
        else
        {
            last_accessed = var_stack.top();
            var_stack.pop();
            var_stack.push(member);
        }
        return true;
    }

    bool Tokenizer::proc_arg_variable(const char* name)
    {
        Variable* var = vadmin.get_var(yytext);
        last_accessed = nullptr;

        if (!var)
            return false;

        func_args.top().push_back(var);

        return true;
    }

    unsigned short Tokenizer::proc_literal_arg(const TypeAPI&
type)
    {
        tmp_literals.emplace_back(std::make_unique<Variable>("",
type));
        tmp_literals.back()->assign(yytext);
        func_args.top().push_back(tmp_literals.back().get());

        return VarAdministrator::VAR_OK;
    }
```

```
unsigned short Tokenizer::proc_arg_int()
{
   return proc_literal_arg(*int_api);
}

unsigned short Tokenizer::proc_arg_string()
{
   return proc_literal_arg(*string_api);
}

unsigned short Tokenizer::proc_arg_ipv4()
{
   return proc_literal_arg(*ipv4_api);
}

bool Tokenizer::init_func_call()
{
   ++func_call_level;

   Variable* self_func;

   // insert func
   if (func_call_level > 1)
      self_func = func_args.top().back();
   else
      self_func = var_stack.top();
   func_args.push(std::vector<Variable*>());

   // insert self caller var
   func_args.top().push_back(self_func);
   if (last_accessed)
      func_args.top().push_back(last_accessed);

   return true;
}

bool Tokenizer::call_func()
{
   Variable* func;

   if (func_call_level > 1)
      func = func_args.top().front();
   else
      func = var_stack.top();

   if (func->get_api()->get_code() != func_api->get_code())
      return false;

   auto ret = static_cast<Function*>(func->get_value())-
>execute(func_args.top());
   func_ret_vals.push(std::move(ret));
   func_args.pop();

   if (func_call_level > 1)
   {
      *func_args.top().rbegin() = func_ret_vals.top().get();
   }

   --func_call_level;

   return true;
}

Tokenizer::~Tokenizer()
{
   vadmin.clear();
   padmin.close_plugins();
}

bool Tokenizer::load_custom_plugin(const char *lib_path)
{
   return padmin.load_so_plugin(lib_path);
}
```

B.1.1.5 type_api.h

```
#ifndef TYPE_API_H
#define TYPE_API_H

#include <string>
#include <map>
#include <memory>

class TypeValue;
class Variable;

class TypeAPI
{
public:
   typedef std::unique_ptr<TypeValue> member_type;
   typedef member_type (*member_init)();
   typedef const std::map<std::string, member_init>
members_map; // TODO: change to set
   typedef const members_map* const members_map_p;
```

```
        typedef const std::pair<std::string, member_init>
member_entry;

        std::unique_ptr<TypeValue> (*construct)();

        TypeAPI() = delete;
        TypeAPI(const char* name,
          std::unique_ptr<TypeValue> (*constructor)(),
          members_map_p members = nullptr
        );
        virtual ~TypeAPI() = default;

        const char* get_name() const;
        unsigned get_code() const;
        members_map_p get_members() const;

    protected:
        std::string name;
        unsigned type_code;
        members_map_p members;

        friend class TypeAdministrator;

    private:
        void set_code(unsigned);
    };

    #endif
```

B.1.1.6 type_management.h

```
    #ifndef TYPE_MANAGEMENT_H
    #define TYPE_MANAGEMENT_H

    #include <unordered_map>

    #include "type_api.h"

    class TypeAdministrator
    {
    public:
        static unsigned add_type(const TypeAPI& api);
        static unsigned get_type_code(const char* name);
        static TypeAPI* get_api(unsigned code);
        static TypeAPI* get_api(const char* name);

    private:
        static std::unordered_map<unsigned, TypeAPI> types; //
TODO: change to set
        static std::map<std::string, unsigned> type_codes;

        static unsigned max_id;
    };

    typedef TypeAdministrator TAdmin;

    #endif
```

B.1.1.7 type_management.cpp

```
    #include "type_management.h"
    #include "standard_types.h"

    std::unordered_map<unsigned, TypeAPI>
TypeAdministrator::types;
    std::map<std::string, unsigned>
TypeAdministrator::type_codes;
    unsigned TypeAdministrator::max_id = 1;

    TypeAPI::TypeAPI(const char* name,
        std::unique_ptr<TypeValue> (*constructor)(),
        members_map_p members
    )
        : name(name), type_code(0), construct(constructor),
members(members)
    {}

    const char *TypeAPI::get_name() const
    {
        return name.c_str();
    }

    unsigned TypeAPI::get_code() const
    {
        return type_code;
    }

    TypeAPI::members_map_p TypeAPI::get_members() const
    {
        return members;
    }
```

```
void TypeAPI::set_code(unsigned value)
{
   if (!type_code)
      type_code = value;
}

unsigned TypeAdministrator::add_type(const TypeAPI& api)
{
   auto ret = types.emplace(max_id, api);
   if (ret.second)
   {
      TypeAPI& inserted = ret.first->second;
      inserted.set_code(max_id);
      ++max_id;
      type_codes.emplace(inserted.get_name(),
inserted.get_code());

      return max_id - 1;
   }

   return 0;
}
```

```
unsigned TypeAdministrator::get_type_code(const char
*name)
{
   // TODO: optimise to only look up once
   if (type_codes.count(name))
      return type_codes.at(name);
   return 0;
}

TypeAPI *TypeAdministrator::get_api(unsigned code)
{
   // TODO: optimise to only look up once
   if (types.count(code))
      return &types.at(code);
   return nullptr;
}

TypeAPI *TypeAdministrator::get_api(const char *name)
{
   unsigned code = get_type_code(name);
   return get_api(code);
}
```

B.1.1.8 variable.h

```
#ifndef VARIABLE_H
#define VARIABLE_H

#include <vector>

#include "type_api.h"

class Variable
{
public:
   Variable(const char* name, const TypeAPI& type);
   Variable(const char *name, TypeAPI::member_type value);
   Variable(const char* name,
      std::unique_ptr<Variable> (*func_callback)
(std::vector<Variable*>&));
   Variable(const Variable& other);
   TypeValue* get_value();
   const TypeValue* get_const_value() const;
   const TypeAPI* get_api() const;
   Variable* get_member(const char* name);

   const char* get_name() const;
   bool assign(const char* value);
   bool assign(const Variable& value);

private:
   const TypeAPI& type;
   std::string name;
   std::unique_ptr<TypeValue> value;
};

#endif
```

B.1.1.9 variables_management.h

```
#ifndef VARIABLEs_MANAGEMENT_H
#define VARIABLEs_MANAGEMENT_H

#include <vector>

#include "type_api.h"
#include "variable.h"

class VarAdministrator
{
```

```
    public:
        enum VarRet : unsigned short
        {
            VAR_OK,
            VAR_EXISTS,
            VAR_WRONG_TYPE,
            VAR_ERR_GENERIC
        };

        VarAdministrator() = default;
        virtual ~VarAdministrator() = default;

        void clear();
        std::pair<unsigned short, Variable*> add_var(const
TypeAPI& type, const char* name);
        std::pair<unsigned short, Variable*> add_func(const char*
name,
            std::unique_ptr<Variable> (*executor)
(std::vector<Variable*>& args));
        std::pair<unsigned short, Variable*> add_var(const
Variable* var);
        std::pair<unsigned short, Variable *>
add_var(TypeAPI::member_entry var);
        Variable* get_var(const char* name);
        const Variable* get_const_var(const char* name) const;

    private:
        std::map<std::string, Variable> variables; // TODO: change
to set
    };

    class TypeValue
    {
    public:
        TypeValue() = delete;
        TypeValue(const TypeAPI& type);
        virtual ~TypeValue() = default;

        virtual bool set(const char* value) = 0;
        virtual bool set(const TypeValue* value) = 0;
        virtual void* get() const = 0;

        const TypeAPI& get_type() const;
        VarAdministrator& get_members();
        TypeValue* get_member(const char* name);
        const TypeValue* get_const_member(const char* name)
const;

    protected:
        const TypeAPI& type;
        VarAdministrator members;

    private:
        void init_members();
    };

    #endif
```

B.1.1.10 variables_management.cpp

```
    #include "variables_management.h"
    #include "standard_types.h"
    #include "variable.h"

    Variable::Variable(const char *name, const TypeAPI &type)
        : name(name), type(type), value(type.construct())
    {}

    Variable::Variable(const char *name, TypeAPI::member_type
value)
        : name(name), type(value->get_type()),
value(std::move(value))
    {}

    Variable::Variable(const Variable &other)
        : name(other.name), type(other.type),
value(type.construct())//
    {
        value->set(other.value.get());
    }

    TypeValue *Variable::get_value()
    {
        return value.get();
    }

    const TypeValue *Variable::get_const_value() const
    {
        return value.get();
    }

    const TypeAPI *Variable::get_api() const
    {
        return &type;
    }
```

```
Variable *Variable::get_member(const char *name)
{
   return value->get_members().get_var(name);
}

const char *Variable::get_name() const
{
   return name.c_str();
}

bool Variable::assign(const char *value)
{
   if (!value)
      return false;

   if (!this->value.get())
      this->value = type.construct();

   return this->value->set(value);
}

bool Variable::assign(const Variable &value)
{
   if (!this->value.get())
      this->value = type.construct();

   return this->value->set(value.value.get());
}

void VarAdministrator::clear()
{
   variables.clear();
}

std::pair<unsigned short, Variable *>
VarAdministrator::add_var(
      const TypeAPI &type, const char *name)
{
   if (variables.count(name))
      return { VAR_EXISTS, nullptr }; // TODO: return the
existing variable

   auto ret = variables.emplace(name, Variable(name, type));
   if (!ret.second)
      return { VAR_ERR_GENERIC, nullptr };

   return { VAR_OK, &ret.first->second };
}

std::pair<unsigned short, Variable*>
VarAdministrator::add_func(const char *name,
      std::unique_ptr<Variable> (*executor)
(std::vector<Variable*>& args))
{
   if (variables.count(name))
      return { VAR_EXISTS, nullptr }; // TODO: return the
existing variable

   auto ret = variables.emplace(name, Variable(name,
executor));
   if (!ret.second)
      return { VAR_ERR_GENERIC, nullptr };

   return { VAR_OK, &ret.first->second };
}

std::pair<unsigned short, Variable *>
VarAdministrator::add_var(const Variable *var)
{
   if (!var)
      return { VAR_ERR_GENERIC, nullptr };

   if (variables.count(var->get_name()))
      return { VAR_EXISTS, nullptr }; // TODO: return the
existing variable

   auto ret = variables.emplace(var->get_name(),
Variable(*var));
   if (!ret.second)
      return { VAR_ERR_GENERIC, nullptr };

   return std::pair<unsigned short, Variable *>();
}

std::pair<unsigned short, Variable *>
VarAdministrator::add_var(TypeAPI::member_entry var)
{
   if (!var.first.c_str() || !var.second)
      return { VAR_ERR_GENERIC, nullptr };

   if (variables.count(var.first.c_str()))
      return { VAR_EXISTS, nullptr }; // TODO: return the
existing variable

   auto declared = var.second();

   auto ret = variables.emplace(std::piecewise_construct,
      std::forward_as_tuple(var.first.c_str()),
```

```
            std::forward_as_tuple(var.first.c_str(),
std::move(declared)));

        if (!ret.second)
            return { VAR_ERR_GENERIC, nullptr };

        return std::pair<unsigned short, Variable *>(VAR_OK,
nullptr);
    }

    Variable *VarAdministrator::get_var(const char *name)
    {
        // TODO: optimise to only look up once
        if (variables.count(name))
            return &variables.at(name);
        return nullptr;
    }

    const Variable *VarAdministrator::get_const_var(const char
*name) const
    {
        // TODO: optimise to only look up once
        if (variables.count(name))
            return &variables.at(name);
        return nullptr;
    }

    TypeValue::TypeValue(const TypeAPI& type)
        : type(type), members()
    {
        init_members();
    }

    const TypeAPI &TypeValue::get_type() const
    {
        return type;
    }

    VarAdministrator &TypeValue::get_members()
    {
        return members;
    }

    TypeValue *TypeValue::get_member(const char *name)
    {
        return members.get_var(name)->get_value();
    }

    const TypeValue *TypeValue::get_const_member(const char
*name) const
    {
        return members.get_const_var(name)->get_const_value();
    }

    void TypeValue::init_members()
    {
        auto members_to_add = type.get_members();

        if (!members_to_add)
            return;

        for (auto member : *members_to_add)
            members.add_var(member);
    }
```

B.1.1.11 plugin.h

```
    #ifndef PLUGIN_H
    #define PLUGIN_H

    #include "variable.h"

    struct PluginAPI
    {
        const char* name = nullptr;
        // should terminate with nullptr
        const TypeAPI** types = nullptr;
        const TypeAPI::members_map_p global_vars = nullptr;
    };

    #endif
```

B.1.1.12 plugins_management.h

```
    #ifndef PLUGINS_MANAGEMENT_H
    #define PLUGINS_MANAGEMENT_H

    #include "variables_management.h"
    #include "plugin.h"

    #define PLUGINS_PATH_VAR
"TRAFFIC_PLUGINS_PATH"
```

```
class PluginsAdministrator
{
public:
   PluginsAdministrator(VarAdministrator& vadmin);
   ~PluginsAdministrator();

   bool load_plugin(const PluginAPI* plugin);
   bool load_so_plugin(const char* lib_path);
   bool load_so_plugins();
   void close_plugins();

private:
   VarAdministrator& managed_vadmin;
   std::vector<void*> loaded_plugins_handles;
};

#endif
```

B.1.1.13 plugins_management.cpp

```
#include <dlfcn.h>
#include <filesystem>

#include "plugins_management.h"
#include "type_management.h"

PluginsAdministrator::PluginsAdministrator(VarAdministrator
&vadmin)
   : managed_vadmin(vadmin)
{}

PluginsAdministrator::~PluginsAdministrator()
{
   close_plugins();
}

bool PluginsAdministrator::load_plugin(const PluginAPI
*plugin)
{
   if (!plugin)
      return false;

   size_t idx = 0;
   const TypeAPI* type = plugin->types ?
      plugin->types[idx] :
      nullptr;

   while (type)
   {
      TypeAdministrator::add_type(*type);
      type = plugin->types[++idx];
   }

   if (!plugin->global_vars)
      return true;

   for (const auto& var : *plugin->global_vars)
      managed_vadmin.add_var(var);

   return true;
}

bool PluginsAdministrator::load_so_plugin(const char
*lib_path)
{
   void* handle;

   handle = dlopen(lib_path, RTLD_LAZY);
   if(!handle)
      return false;

   const PluginAPI*(*get_plugin)() = (const PluginAPI*(*)
())dlsym(handle, "get_plugin");
   const PluginAPI* loaded_plugin = get_plugin();

   load_plugin(loaded_plugin);

   loaded_plugins_handles.push_back(handle);

   return true;
}

bool PluginsAdministrator::load_so_plugins()
{
   const char* plugins_path =
std::getenv(PLUGINS_PATH_VAR);

   if (!plugins_path || !std::filesystem::exists(plugins_path))
      return false;

   for (auto& file :
std::filesystem::directory_iterator{ plugins_path })
   {
      if (file.is_regular_file())
```

```
            load_so_plugin(file.path().c_str());
        }

        return true;
    }
```

```
    void PluginsAdministrator::close_plugins()
    {
        for (auto handle : loaded_plugins_handles)
            dlclose(handle);
    }
```

B.1.1.14 standard_types.h

```
#ifndef STANDARD_TYPES_H
#define STANDARD_TYPES_H

#include <vector>

#include "plugin.h"
#include "variables_management.h"

// === Int ===

#define INT_NAME "Int"

class IntValue : public TypeValue
{
public:
    IntValue(const TypeAPI& self_type);

    virtual bool set(const char* value) override;
    virtual bool set(const TypeValue* value) override;
    bool set(int value);
    virtual void* get() const override;
    int get_int() const;

private:
    int value;
};

// === String ===

#define STRING_NAME "String"

class StringValue : public TypeValue
{
public:
    StringValue(const TypeAPI& self_type);

    virtual bool set(const char* value) override;
    virtual bool set(const TypeValue* value) override;
    virtual void* get() const override;
    const char* get_string() const;

    private:
        std::string value;
    };

    // === IPv4 ===

    #define IPv4_NAME "IPv4"

    class IPv4Value : public TypeValue
    {
    public:
        IPv4Value(const TypeAPI& self_type);

        virtual bool set(const char* value) override;
        virtual bool set(const TypeValue* value) override;
        virtual void* get() const override;
        const uint32_t get_ip() const;
        std::string to_string() const;

        static std::unique_ptr<Variable>
ipv4_to_string(std::vector<Variable*>& args);

    private:
        union
        {
            uint32_t ip;
            uint8_t ip_byte[4];
        };
    };

    // === Function ===

    #define FUNC_TYPE_NAME "Func"

    class Function : public TypeValue
    {
    public:
        typedef std::unique_ptr<Variable> (*func_callback)
(std::vector<Variable*>& args);
```

```
        Function(const TypeAPI& self_type, func_callback executor
= nullptr);

        virtual bool set(const char* value) override;
        virtual bool set(const TypeValue* value) override;
        virtual void* get() const override;
        void set_executor(func_callback executor);
        func_callback get_executor() const;
        std::unique_ptr<Variable> execute(std::vector<Variable*>&
args);

    private:
        func_callback executor;
    };

    // ===== Built-ins plugin =====

    extern PluginAPI built_ins;

    #endif
```

B.1.1.15 standard_types.cpp

```
    #include "standard_types.h"

    #include <cstring>
    #include <iostream>
    #include <regex>

    #include "type_management.h"

    // === Int ===

    IntValue::IntValue(const TypeAPI &self_type)
        : TypeValue(self_type), value(0)
    {}

    bool IntValue::set(const char *value)
    {
        int parsed;

        try
        {
            parsed = std::stoi(value);
        }
        catch(const std::exception& e)
        {
            return false;
        }

        this->value = parsed;
        return true;
    }

    bool IntValue::set(const TypeValue* value)
    {
        if (!value)
            return false;
        if (type.get_code() != value->get_type().get_code())
            return false;

        this->value = ((IntValue*)value)->value;

        return true;
    }

    bool IntValue::set(int value)
    {
        this->value = value;

        return true;
    }

    void *IntValue::get() const
    {
        return (void*)&value;
    }

    int IntValue::get_int() const
    {
        return value;
    }

    static std::unique_ptr<TypeValue> construct_int()
    {
        return std::unique_ptr<TypeValue>(
            new IntValue(*TAdmin::get_api(INT_NAME)));
    }

    TypeAPI int_api = TypeAPI(
        INT_NAME,
        construct_int
    );
```

```
// === String ===

static const std::vector<std::pair<std::regex, std::string>>
unescapes = {
   std::make_pair(std::regex(R"(\\\\)"), "\\"),
   std::make_pair(std::regex(R"(\\n)"), "\n"),
   std::make_pair(std::regex(R"(\\r)"), "\r"),
   std::make_pair(std::regex(R"(\\")"), "\"")
};

StringValue::StringValue(const TypeAPI &self_type)
   : TypeValue(self_type)
{}

bool StringValue::set(const char *value)
{
   if (*value == '"')
      value++;
   this->value = value;

   if (this->value.back() == '"')
      this->value.pop_back();

   for (const auto& unesc : unescapes)
   {
      this->value = std::regex_replace(this->value, unesc.first,
unesc.second);
   }

   return true;
}

bool StringValue::set(const TypeValue* value)
{
   if (!value)
      return false;

   if (type.get_code() != value->get_type().get_code())
      return false;

   this->value = ((StringValue*)value)->value;

   return true;
}

void *StringValue::get() const
{
   return (void*)&value;
}

const char *StringValue::get_string() const
{
   return value.c_str();
}

static std::unique_ptr<TypeValue> construct_string()
{
   return std::unique_ptr<TypeValue>(
      new StringValue(*TAdmin::get_api(STRING_NAME)));
}

TypeAPI string_api = TypeAPI(
   STRING_NAME,
   construct_string
);

// === IPv4 ===

#define IP_DELIM "."

IPv4Value::IPv4Value(const TypeAPI &self_type)
   : TypeValue(self_type)
{}

bool IPv4Value::set(const char *value)
{
   char* proc_buf = new char[strlen(value)];
   strcpy(proc_buf, value);

   char* byte = std::strtok(proc_buf, IP_DELIM);
   unsigned short byte_idx = 0;
   while (byte && byte_idx < 4)
   {
      try
      {
         int parsed = atoi(byte);
         if (parsed < 0 || parsed >= 256)
         {
            delete[] proc_buf;
            return false;
         }
         ip_byte[byte_idx++] = parsed;
      }
      catch(const std::exception& e)
      {
         delete[] proc_buf;
         return false;
      }
```

```
            byte = std::strtok(nullptr, IP_DELIM);
        }

        delete[] proc_buf;

        return true;
    }

    bool IPv4Value::set(const TypeValue* value)
    {
        if (!value)
            return false;

        if (type.get_code() != value->get_type().get_code())
            return false;

        this->ip = ((IPv4Value*)value)->ip;

        return true;
    }

    void *IPv4Value::get() const
    {
        return (void*)&ip;
    }

    const uint32_t IPv4Value::get_ip() const
    {
        return ip;
    }

    std::string IPv4Value::to_string() const
    {
        return std::to_string(ip_byte[0]) + IP_DELIM +
            std::to_string(ip_byte[1]) + IP_DELIM +
            std::to_string(ip_byte[2]) + IP_DELIM +
            std::to_string(ip_byte[3]);
    }

    std::unique_ptr<Variable>
IPv4Value::ipv4_to_string(std::vector<Variable*>& args)
    {
        if (args.size() != 2)
            return nullptr;

        std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));

        StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

        IPv4Value* ipv4 = (IPv4Value*)args[1]->get_value();

        ret_string->set(((IPv4Value*)args[1]->get_value())-
>to_string().c_str());

        return ret_val;
    }

    static TypeAPI::members_map ipv4_members = {
        { "to_string", []() -> TypeAPI::member_type {
            return
std::make_unique<Function>(*TAdmin::get_api(FUNC_TYPE_NA
ME),
                IPv4Value::ipv4_to_string);
        } }
    };

    static std::unique_ptr<TypeValue> construct_ipv4()
    {
        return std::unique_ptr<TypeValue>(
            new IPv4Value(*TAdmin::get_api(IPv4_NAME)));
    }

    TypeAPI ipv4_api = TypeAPI(
        IPv4_NAME,
        construct_ipv4,
        &ipv4_members
    );

    // === Function ===

    Variable::Variable(const char *name, Function::func_callback
executor)
        : name(name),
type(*TAdmin::get_api(FUNC_TYPE_NAME)),
        value(new Function(type, executor))
    {}

    Function::Function(const TypeAPI &self_type, func_callback
executor)
        : TypeValue(self_type), executor(executor)
    {}

    bool Function::set(const char *value)
    {
        return false;
```

```
}

bool Function::set(const TypeValue *value)
{
  if (type.get_code() != value->get_type().get_code())
    return false;

  this->executor = ((Function*)value)->executor;

  return true;
}

void *Function::get() const
{
  return nullptr;
}

void Function::set_executor(func_callback executor)
{
  this->executor = executor;
}

Function::func_callback Function::get_executor() const
{
  return executor;
}

std::unique_ptr<Variable>
Function::execute(std::vector<Variable *> &args)
{
  return executor(args);
}

static std::unique_ptr<TypeValue> construct_func()
{
  return std::unique_ptr<TypeValue>(
    new Function(*TAdmin::get_api(FUNC_TYPE_NAME),
nullptr));
}

TypeAPI func_api = TypeAPI(
  FUNC_TYPE_NAME,
  construct_func
);

// ==== Built-in objects ====
static std::unique_ptr<Variable>
print_str(std::vector<Variable*>& args)
{
  if (args.size() != 2 || !args[1])
    return nullptr;

  unsigned arg_type = args[1]->get_api()->get_code();

  if (arg_type == TAdmin::get_type_code(STRING_NAME))
    std::cout << ((StringValue*)args[1]->get_value())-
>get_string() << std::endl;
  else if (arg_type == TAdmin::get_type_code(INT_NAME))
    std::cout << ((IntValue*)args[1]->get_value())->get_int()
<< std::endl;

  return nullptr;
}

TypeAPI::member_init print_func = []() ->
TypeAPI::member_type {
  return
std::make_unique<Function>(*TAdmin::get_api(FUNC_TYPE_NA
ME),
    print_str);
};

// ===== Built-ins plugin =====

static const TypeAPI* built_in_types[] =
{
  &func_api,
  &int_api,
  &string_api,
  &ipv4_api,
  nullptr
};

static const TypeAPI::members_map built_in_vars =
{
  { "print", print_func }
};

PluginAPI built_ins = {
  "built_in",
  built_in_types,
  &built_in_vars
};
```

В.1.2 Модульні тести головної програми

В.1.2.1 tokenizer_stubs.cpp

```
#include "tokenizer.h"

Tokenizer::Tokenizer(std::istream& in, std::ostream& out)
   : yyFlexLexer(in, out), padmin(vadmin)
{}

Tokenizer::~Tokenizer()
{}

int Tokenizer::process()
{
   return yylex();
}

bool Tokenizer::init_var_proc()
{
   return true;
}

bool Tokenizer::decl_check_type()
{
   return true;
}

unsigned short Tokenizer::declare_var()
{
   return VarAdministrator::VarRet::VAR_OK;
}

bool Tokenizer::assign_variable()
{
   return true;
}

unsigned short Tokenizer::assign_literal(const TypeAPI &type)
{
   return ASSIGN_OK;
}

unsigned short Tokenizer::assign_int()
{
   return ASSIGN_OK;
}

unsigned short Tokenizer::assign_string()
{
   return ASSIGN_OK;
}

unsigned short Tokenizer::assign_ipv4()
{
   return ASSIGN_OK;
}

bool Tokenizer::get_variable()
{
   return true;
}

bool Tokenizer::get_member(Variable *&out)
{
   return true;
}

bool Tokenizer::proc_arg_variable(const char* name)
{
   return true;
}

unsigned short Tokenizer::proc_literal_arg(const TypeAPI&
type)
{
   return VarAdministrator::VarRet::VAR_OK;
}

unsigned short Tokenizer::proc_arg_int()
{
   return VarAdministrator::VarRet::VAR_OK;
}

unsigned short Tokenizer::proc_arg_string()
{
   return VarAdministrator::VarRet::VAR_OK;
}

unsigned short Tokenizer::proc_arg_ipv4()
{
   return VarAdministrator::VarRet::VAR_OK;
}
```

```
bool Tokenizer::init_func_call()
{
   return true;
}

bool Tokenizer::call_func()
{
   return true;
}
```

B.1.2.2 tokenizer_tests.cpp

```
#include <catch2/catch.hpp>

#include <iostream>
#include <sstream>

#include "tokenizer.h"

#define TOKENIZER_TEST(in_str, expected_str,
expected_code) \
{                                          \
   std::stringstream str(in_str);                \
   std::string expected(expected);                \
                                          \
   Tokenizer t(str, out);                        \
   t.debug_print = true;                         \
   CHECK(t.process() == expected_code);          \
   REQUIRE(dbg.str() == expected_str);           \
}

#define TOKENIZER_VALID(in_str, expected_str)        \
   TOKENIZER_TEST(in_str, expected_str,
Tokenizer::VALID)

#define TOKENIZER_INVALID(in_str, expected_str)      \
   TOKENIZER_TEST(in_str, expected_str,
Tokenizer::INVALID)

TEST_CASE("Tokenizer tests", "[parser]")
{
   std::ostream out(nullptr);

   std::streambuf *sbuf = std::cout.rdbuf();
   std::ostringstream dbg;
   std::cout.rdbuf(dbg.rdbuf());

   SECTION("Declaration")
   {
      SECTION("Short")
      {
         TOKENIZER_VALID(
            "Int variable",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: variable\n\
[DEBUG]: EOF\n\
"
         );
      }

      SECTION("Full no spaces")
      {
         TOKENIZER_VALID(
            "Int variable=21",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: variable\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: EOF\n\
"
         );
      }

      SECTION("Full with spaces")
      {
         TOKENIZER_VALID(
            "Int variable \t\t = \t 21",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: variable\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: EOF\n\
"
         );
      }

      SECTION("Full with IpvAddr")
      {
         TOKENIZER_VALID(
            "Int variable = 192.168.0.1",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: variable\n\
```

```
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN IP: 192.168.0.1\n\
        [DEBUG]: EOF\n\
        "
                );
            }

            SECTION("Full with String")
            {
                TOKENIZER_VALID(
                    "Int variable = \"text \\n multiline\"",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: variable\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN STRING: \"text \\n multiline\"\n\
        [DEBUG]: EOF\n\
        "
                );
            }

            SECTION("Full with variable name")
            {
                TOKENIZER_VALID(
                    "Int variable = another_var",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: variable\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN VARIABLE: another_var\n\
        [DEBUG]: EOF\n\
        "
                );
            }

            SECTION("Full with member access")
            {
                TOKENIZER_VALID(
                    "Int variable =
another_var.member.another_member",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: variable\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN VARIABLE: another_var\n\
        [DEBUG]: MEMBER ACCESS START\n\
        [DEBUG]: ACCESS TO SUB-MEMBER: member\n\
        [DEBUG]: MEMBER ACCESS START\n\
        [DEBUG]: ACCESS TO SUB-MEMBER: another_member\n\
        [DEBUG]: EOF\n\
        "
                );
            }

            SECTION("Full with method_call")
            {
                TOKENIZER_VALID(
                    "Int variable = another_var.method()",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: variable\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN VARIABLE: another_var\n\
        [DEBUG]: MEMBER ACCESS START\n\
        [DEBUG]: ACCESS TO SUB-MEMBER: method\n\
        [DEBUG]: METHOD CALL START\n\
        [DEBUG]: METHOD CALL END\n\
        [DEBUG]: EOF\n\
        "
                );
            }
        }

        SECTION("Assignment")
        {
            SECTION("basic int no spaces")
            {
                TOKENIZER_VALID(
                    "int=21",
                    "[DEBUG]: INIT VAR_NAME: int\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN INT: 21\n\
        [DEBUG]: EOF\n\
        "
                );
            }

            SECTION("basic int with spaces")
            {
                TOKENIZER_VALID(
                    "int \t                =\t21",
                    "[DEBUG]: INIT VAR_NAME: int\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN INT: 21\n\
        [DEBUG]: EOF\n\
        "
                );
            }
```

```
    SECTION("Ipv4Addr")
    {
      TOKENIZER_VALID(
        "addr = 192.168.1.1",
        "[DEBUG]: INIT VAR_NAME: addr\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN IP: 192.168.1.1\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("String")
    {
      TOKENIZER_VALID(
        "str = \"string\"",
        "[DEBUG]: INIT VAR_NAME: str\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN STRING: \"string\"\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Variable")
    {
      TOKENIZER_VALID(
        "var = other_var",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other_var\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Member of a variable as value")
    {
      TOKENIZER_VALID(
        "var = other.member",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: member\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Member of a variable as target")
    {
      TOKENIZER_VALID(
        "var.member = other",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: member\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Method call as value")
    {
      TOKENIZER_VALID(
        "var = other.getter()",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: getter\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Method call as target")
    {
      TOKENIZER_VALID(
        "var.getter() = other",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: getter\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("Literal as target")
```

```
    {
      TOKENIZER_INVALID(
        "216.58.215.110 = \"google.com\"",
        "[DEBUG]: Unknown token: 2\n"
      );
    }
  }

  SECTION("Member access")
  {
    SECTION("basic")
    {
      TOKENIZER_VALID(
        "var = other.member",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: member\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("no sub-member")
    {
      TOKENIZER_INVALID(
        "var = other.",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: Unexpected EOF\n"
      );
    }

    SECTION("with spaces between members")
    {
      SECTION("before dot")
      {
        TOKENIZER_INVALID(
          "var = other . member",
          "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: Unknown token:  \n\
"
        );
      }

      SECTION("after dot")
      {
        TOKENIZER_INVALID(
          "var = other. member",
          "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: Unknown token:  \n\
"
        );
      }
    }

    SECTION("sub-sub-member")
    {
      TOKENIZER_VALID(
        "var = other.member.other_member",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: member\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: other_member\n\
[DEBUG]: EOF\n\
"
      );
    }
  }

  SECTION("Function calls")
  {
    SECTION("basic")
    {
      TOKENIZER_VALID(
        "other.method()",
        "[DEBUG]: INIT VAR_NAME: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: method\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
      );
    }

    SECTION("space after name")
```

```
    {
      TOKENIZER_INVALID(
        "var.method ()",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: method\n\
[DEBUG]: Unknown token:  \n\
"
      );
    }

    SECTION("sub-member method")
    {
      TOKENIZER_VALID(
        "var = other.member.method()",
        "[DEBUG]: INIT VAR_NAME: var\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: other\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: member\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: method\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
      );
    }
  }

  SECTION("FUNCTION ARGS")
  {
    SECTION("Arg types")
    {
      SECTION("Int")
      {
        TOKENIZER_VALID(
          "call(21)",
          "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG INT: 21\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
        );
      }

      SECTION("Ipv4Addr")
      {
        TOKENIZER_VALID(
          "ping(8.8.8.8)",
          "[DEBUG]: INIT VAR_NAME: ping\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG IP: 8.8.8.8\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
        );
      }

      SECTION("String")
      {
        TOKENIZER_VALID(
          "ping(\"google.com\")",
          "[DEBUG]: INIT VAR_NAME: ping\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG STRING: \"google.com\"\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
        );
      }

      SECTION("Variable")
      {
        TOKENIZER_VALID(
          "call(mom)",
          "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
        );
      }

      SECTION("Sub-member")
      {
        TOKENIZER_VALID(
          "call(mom.phone)",
          "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: phone\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
```

```
            );
        }

        SECTION("Call value")
        {
            TOKENIZER_VALID(
                "call(mom.phone())",
                "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: phone\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
            );
        }
    }

    SECTION("Multiple args")
    {
        SECTION("Basic")
        {
            TOKENIZER_VALID(
                "call(mom, 5, \"hi there\")",
                "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: ARG INT: 5\n\
[DEBUG]: ARG STRING: \"hi there\"\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
            );
        }

        SECTION("No spaces")
        {
            TOKENIZER_VALID(
                "call(mom,5,\"hi there\")",
                "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: ARG INT: 5\n\
[DEBUG]: ARG STRING: \"hi there\"\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
            );
        }

        SECTION("all spaces")
        {
            TOKENIZER_VALID(
                "call(  mom  , 5 , \"hi there\"  )",
                "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: ARG INT: 5\n\
[DEBUG]: ARG STRING: \"hi there\"\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: EOF\n\
"
            );
        }

        SECTION("No arg after comma")
        {
            TOKENIZER_INVALID(
                "call(mom, )",
                "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG VARIABLE: mom\n\
[DEBUG]: Unknown token: )\n\
"
            );
        }
    }

    SECTION("EOF")
    {
        TOKENIZER_INVALID(
            "call(-1",
            "[DEBUG]: INIT VAR_NAME: call\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG INT: -1\n\
[DEBUG]: Unexpected EOF\n\
"
        );
    }
}

SECTION("Comments")
{
    SECTION("Ends with EOF")
    {
```

```
        SECTION("At line start")
        {
          TOKENIZER_VALID(
            "# comment",
            "[DEBUG]: COMMENT\n\
[DEBUG]: EOF\n\
"
          );
        }

        SECTION("After a command")
        {
          TOKENIZER_VALID(
            "Int i # comment",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i\n\
[DEBUG]: COMMENT\n\
[DEBUG]: EOF\n\
"
          );
        }
      }

      SECTION("Ends with newline")
      {
```

```
        SECTION("At line start")
        {
          TOKENIZER_VALID(
            "# comment\n",
            "[DEBUG]: COMMENT\n[DEBUG]: EOF\n"
          );
        }

        SECTION("After a command")
        {
          TOKENIZER_VALID(
            "Int i # comment\n",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i\n\
[DEBUG]: COMMENT\n\
[DEBUG]: EOF\n\
"
          );
        }
      }
    }

    std::cout.rdbuf(sbuf);
}
```

B.1.2.3 test_type.h

```
#ifndef TEST_TYPE_H
#define TEST_TYPE_H

#include "plugin.h"

#define TEST_TYPE_NAME "Test"

extern "C" const PluginAPI* const get_plugin();

#endif
```

B.1.2.4 test_type.cpp

```
#include <cassert>
#include <iostream>

#include "standard_types.h"
#include "plugin.h"
#include "type_management.h"
#include "test_type.h"

// === Test type ===

std::unique_ptr<Variable> print_func(std::vector<Variable*>&
args)
{
    assert(args.size() >= 2);

    std::cout << "printing Ints with " << args[1]->get_name() <<
"."
      << args[0]->get_name() << ": ";

    auto arg = std::next(std::next(args.begin()));
    while (arg != args.end())
    {
      IntValue* int_val = (IntValue*)(*arg)->get_value();
      std::cout << int_val->get_int() << " ";
      arg++;
    }
```

```
        std::cout << std::endl;

        return nullptr;
    }

    std::unique_ptr<Variable> get_int(std::vector<Variable*>&
args)
    {
        assert(args.size() >= 2);

        Variable* ret_val = args[1]->get_member("num");

        if (!ret_val)
            return nullptr;
        return std::unique_ptr<Variable>(new Variable(*ret_val));
    }

    static TypeAPI::members_map test_members = {
        { "num", []() -> TypeAPI::member_type {
            return
std::make_unique<IntValue>(*TAdmin::get_api(INT_NAME));
        } },
        { "print", []() -> TypeAPI::member_type {
            return
std::make_unique<Function>(*TAdmin::get_api(FUNC_TYPE_NA
ME),
                print_func);
        } },
        { "get", []() -> TypeAPI::member_type {
            return
std::make_unique<Function>(*TAdmin::get_api(FUNC_TYPE_NA
ME),
                get_int);
        } }
    };

    class TestType : public TypeValue
    {
    public:
        TestType(const TypeAPI& self_type)
            : TypeValue(self_type)
        {}

        virtual bool set(const char* value) override
        {
            // can't be assigned
            return false;
        }
        virtual bool set(const TypeValue* value) override
        {
            return false;
        }
        virtual void* get() const override
        {
            return nullptr;
        }
    };

    static std::unique_ptr<TypeValue> construct_test_type()
    {
        return std::unique_ptr<TypeValue>(
            new
TestType(*TAdmin::get_api(TEST_TYPE_NAME)));
    }

    static TypeAPI test_type_api = TypeAPI(
        TEST_TYPE_NAME,
        construct_test_type,
        &test_members
    );

    // ===== Test plugin =====

    static const TypeAPI* test_types[] =
    {
        &test_type_api,
        nullptr
    };

    static PluginAPI test_plugin =
    {
        "Test plugin",
        test_types
    };

    const PluginAPI *const get_plugin()
    {
        return &test_plugin;
    }
```

B.1.2.5 parser_test_driver.h

```
    #ifndef PARSER_TEST_DRIVER_H
    #define PARSER_TEST_DRIVER_H

    #include <cstring>
```

```
#include "../tokenizer.h"
#include "../type_management.h"

class ParserTestDriver
{
public:
    static Variable* get_var(Tokenizer& t, const char* name)
    {
        return t.vadmin.get_var(name);
    }

    static bool verify_var(const Variable* var, unsigned
type_code)
    {
        return var && var->get_api()->get_code() == type_code;
    }

    static bool verify_int_value(Variable* var, int value)
    {
        if (!verify_var(var,
TAdmin::get_type_code(INT_NAME)))
            return false;

        IntValue* int_var = (IntValue*)var->get_value();

        return int_var->get_int() == value;
    }

    static bool verify_string_value(Variable* var, const char*
value)
    {
        if (!verify_var(var,
TAdmin::get_type_code(STRING_NAME)))
            return false;

        StringValue* string_var = (StringValue*)var-
>get_value();

        return !strcmp(string_var->get_string(), value);
    }

    static bool verify_ipv4_value(Variable* var, const char*
value)
    {
        if (!verify_var(var,
TAdmin::get_type_code(IPv4_NAME)))
            return false;

        IPv4Value* ip_var = (IPv4Value*)var->get_value();

        return !strcmp(ip_var->to_string().c_str(), value);
    }
};

#endif
```

B.1.2.6 parser_tests.cpp

```
/*
    Start this test from the folder its executable is located in.
    This test loads in a custom plugin with a test type, which is
built in the
    same directory, and its path is specified as relative.
    It's relative because i didn't find it worth it to try to find
absolute path
    in runtime, or install the test plugin on the system.
    This is done automatically when run with ctest.
*/
#include <catch2/catch.hpp>

#include <iostream>
#include <sstream>

#include "tokenizer.h"
#include "type_management.h"
#include "test_type.h"

#include "parser_test_driver.h"

#define PARSER_TEST(in_str, expected_str, expected_code)
\
    std::stringstream str(in_str);                      \
    std::string expected(expected);                     \
                                                        \
    Tokenizer t(str, out);                              \
    t.debug_print = true;                               \
    REQUIRE(t.load_custom_plugin("./libtest_type.so")); \
    CHECK(t.process() == expected_code);                \
    REQUIRE(dbg.str() == expected_str);                 \
    dbg.flush();

#define PARSER_VALID(in_str, expected_str)              \
    PARSER_TEST(in_str, expected_str, Tokenizer::VALID)

#define PARSER_INVALID(in_str, expected_str)            \
```

```
    PARSER_TEST(in_str, expected_str, Tokenizer::INVALID)

typedef ParserTestDriver Driver;

static std::ostream out(nullptr);
static std::streambuf *sbuf = std::cout.rdbuf();
static std::ostringstream dbg;

TEST_CASE("Parser framework", "[parser]")
{
  std::cout.rdbuf(dbg.rdbuf());
  std::ostream out(nullptr);

  std::streambuf *sbuf = std::cout.rdbuf();
  std::ostringstream dbg;
  std::cout.rdbuf(dbg.rdbuf());

  SECTION("Type detection")
  {
    SECTION("Built-in types")
    {
      SECTION("Int")
      {
        PARSER_VALID(
          "Int int",
          "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: int\n\
[DEBUG]: EOF\n\
"
        )
        Variable* declared = Driver::get_var(t, "int");
        REQUIRE(Driver::verify_var(declared,
TAdmin::get_type_code(INT_NAME)));
      }

      SECTION("String")
      {
        PARSER_VALID(
          "String str",
          "[DEBUG]: INIT VAR_NAME: String\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: str\n\
[DEBUG]: EOF\n\
"
        )
        Variable* declared = Driver::get_var(t, "str");
        REQUIRE(Driver::verify_var(declared,
TAdmin::get_type_code(STRING_NAME)));
      }

      // A testing type used only in this executable
      SECTION("Test")
      {
        PARSER_VALID(
          "Test test",
          "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: test\n\
[DEBUG]: EOF\n\
"
        )

        Variable* declared = Driver::get_var(t, "test");
        REQUIRE(Driver::verify_var(declared,
TAdmin::get_type_code(TEST_TYPE_NAME)));
      }
    }

    SECTION("Basic invalid")
    {
      PARSER_INVALID(
        "Arbitrary var",
        "[DEBUG]: INIT VAR_NAME: Arbitrary\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: Unsupported type: Arbitrary\n\
"
      )
    }
  }

  SECTION("Assignment")
  {

    SECTION("Literal Int")
    {
      PARSER_VALID(
        "Int i\ni = 21",
        "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: i\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: EOF\n\
"
      )
```

```
            Variable* declared = Driver::get_var(t, "i");
            REQUIRE(Driver::verify_int_value(declared, 21));
        }

        SECTION("Literal String")
        {
            PARSER_VALID(
                "String s\ns = \"text\"",
                "[DEBUG]: INIT VAR_NAME: String\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: s\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: s\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN STRING: \"text\"\n\
[DEBUG]: EOF\n\
"
            )

            Variable* declared = Driver::get_var(t, "s");
            REQUIRE(Driver::verify_string_value(declared,
"text"));
        }

        SECTION("Literal String with excaped chars")
        {
            PARSER_VALID(
                R"(String s = "\\escaped\\ \n \"quoted\r\"")",
                R"([DEBUG]: INIT VAR_NAME: String
[DEBUG]: DECLARATION START
[DEBUG]: DECLARATION VAR_NAME: s
[DEBUG]: ASSIGNMENT START
[DEBUG]: ASSIGN STRING: "\\escaped\\ \n \"quoted\r\""
[DEBUG]: EOF
)"
            )

            Variable* declared = Driver::get_var(t, "s");
            REQUIRE(Driver::verify_string_value(declared,
                "\\escaped\\ \n \"quoted\r\""));
        }

        SECTION("Literal IPv4")
        {
            PARSER_VALID(
                "IPv4 ip\nip = 192.168.0.255",
                "[DEBUG]: INIT VAR_NAME: IPv4\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: ip\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: ip\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN IP: 192.168.0.255\n\
[DEBUG]: EOF\n\
"
            )

            Variable* declared = Driver::get_var(t, "ip");
            REQUIRE(Driver::verify_ipv4_value(declared,
"192.168.0.255"));
        }

        SECTION("Variable")
        {
            PARSER_VALID(
                "Int i1 = 21\nInt i2\n i2 = i1",
                "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i1\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i2\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: i2\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: i1\n\
[DEBUG]: EOF\n\
"
            )
            Variable* i1 = Driver::get_var(t, "i1");
            REQUIRE(Driver::verify_int_value(i1, 21));
            Variable* i2 = Driver::get_var(t, "i2");
            REQUIRE(Driver::verify_int_value(i2, 21));
        }

        SECTION("On declaration")
        {
            SECTION("Literal Int")
            {
                PARSER_VALID(
                    "Int i1 = 21",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i1\n\
```

```
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN INT: 21\n\
        [DEBUG]: EOF\n\
        "
                )
                Variable* i1 = Driver::get_var(t, "i1");
                REQUIRE(Driver::verify_int_value(i1, 21));
            }

            SECTION("Literal String")
            {
                PARSER_VALID(
                    "String s = \"text\"",
                    "[DEBUG]: INIT VAR_NAME: String\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: s\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN STRING: \"text\"\n\
        [DEBUG]: EOF\n\
        "
                )
                Variable* s = Driver::get_var(t, "s");
                REQUIRE(Driver::verify_string_value(s, "text"));
            }

            SECTION("Literal IPv4")
            {
                PARSER_VALID(
                    "IPv4 ip = 192.168.0.255",
                    "[DEBUG]: INIT VAR_NAME: IPv4\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: ip\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN IP: 192.168.0.255\n\
        [DEBUG]: EOF\n\
        "
                )
                Variable* ip = Driver::get_var(t, "ip");
                REQUIRE(Driver::verify_ipv4_value(ip,
"192.168.0.255"));
            }

            SECTION("Variable")
            {
                PARSER_VALID(
                    "Int i1 = 21\nInt i2 = i1",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: i1\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN INT: 21\n\
        [DEBUG]: Command line ended\n\
        [DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: i2\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN VARIABLE: i1\n\
        [DEBUG]: EOF\n\
        "
                )
                Variable* i1 = Driver::get_var(t, "i1");
                REQUIRE(Driver::verify_int_value(i1, 21));
                Variable* i2 = Driver::get_var(t, "i2");
                REQUIRE(Driver::verify_int_value(i2, 21));
            }
        }

        SECTION("Wrong type")
        {
            SECTION("Literal")
            {
                PARSER_INVALID(
                    "Int i = \"string\"",
                    "[DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: i\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN STRING: \"string\"\n\
        [DEBUG]: Invalid value\n\
        "
                )
            }

            SECTION("Variable")
            {
                PARSER_INVALID(
                    "String s\nInt i = s",
                    "[DEBUG]: INIT VAR_NAME: String\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: s\n\
        [DEBUG]: Command line ended\n\
        [DEBUG]: INIT VAR_NAME: Int\n\
        [DEBUG]: DECLARATION START\n\
        [DEBUG]: DECLARATION VAR_NAME: i\n\
        [DEBUG]: ASSIGNMENT START\n\
        [DEBUG]: ASSIGN VARIABLE: s\n\
        [DEBUG]: Assignment error\n\
        "
```

```
            )
          }
        }

        SECTION("Chained")
        {
          // isn't supported
          PARSER_INVALID(
            "Int i1\nInt i2\ni1 = i2 = 20",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i1\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i2\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: i1\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: i2\n\
[DEBUG]: Unknown token:  \n\
"
          )
        }
      }

      SECTION("Member access")
      {
        SECTION("Basic")
        {
          PARSER_VALID(
            "Test t\nt.num",
            "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: EOF\n\
"
          )
        }

        SECTION("Member doesn't exist")
        {
          PARSER_INVALID(
            "Test t\nt.no",
            "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: no\n\
[DEBUG]: There is no such member on this variable\n\
"
          )
        }

        SECTION("Type doesn't have members")
        {
          PARSER_INVALID(
            "Int i\n i.no",
            "[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: i\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: no\n\
[DEBUG]: There is no such member on this variable\n\
"
          )
        }

        SECTION("Assignment")
        {
          SECTION("r value")
          {
            PARSER_VALID(
              "Test t\nt.num = 21",
              "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: EOF\n\
"
            )
            Variable* var = Driver::get_var(t, "t");
            REQUIRE(Driver::verify_var(var,
TAdmin::get_type_code(TEST_TYPE_NAME)));
            var = var->get_member("num");
```

```
            REQUIRE(Driver::verify_int_value(var, 21));
        }

        SECTION("l value")
        {
            PARSER_VALID(
                "Test t\nt.num=21\nInt i=t.num",
                "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: Int\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: i\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: EOF\n\
"
            )
            Variable* var = Driver::get_var(t, "i");
            REQUIRE(Driver::verify_int_value(var, 21));
        }

        SECTION("r and l values with different variables")
        {
            PARSER_VALID(
            "Test t1\n Test t2\nt1.num=21\nt2.num=t1.num",
            "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t1\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t2\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t2\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: EOF\n\
"
            )
            Variable* var = Driver::get_var(t, "t2");
            REQUIRE(Driver::verify_var(var,
TAdmin::get_type_code(TEST_TYPE_NAME)));
            var = var->get_member("num");
            REQUIRE(Driver::verify_int_value(var, 21));
        }

        SECTION("r and l values with same variable")
        {
            PARSER_VALID(
                "Test t1\n\nt1.num=21\nt1.num=t1.num",
                "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t1\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN VARIABLE: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: EOF\n\
"
            )
            Variable* var = Driver::get_var(t, "t1");
            REQUIRE(Driver::verify_var(var,
TAdmin::get_type_code(TEST_TYPE_NAME)));
            var = var->get_member("num");
            REQUIRE(Driver::verify_int_value(var, 21));
        }
    }
}
```

```
  SECTION("Function calls")
  {
    SECTION("Global function call")
    {
      PARSER_VALID(
        "print(\"hello world!\")",
        "[DEBUG]: INIT VAR_NAME: print\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG STRING: \"hello world!\"\n\
[DEBUG]: METHOD CALL END\n\
hello world!\n\
[DEBUG]: EOF\n\
"
      )
    }

    SECTION("Methods")
    {
      PARSER_VALID(
        "Test t\n t.print(21)",
        "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: print\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG INT: 21\n\
[DEBUG]: METHOD CALL END\n\
printing Ints with t.print: 21 \n\
[DEBUG]: EOF\n\
"
      )
    }

    SECTION("Nested calls")
    {
      PARSER_VALID(
        "Test t1\nt1.num=21\n t1.print(21, t1.get(), t1.num)",
        "[DEBUG]: INIT VAR_NAME: Test\n\
[DEBUG]: DECLARATION START\n\
[DEBUG]: DECLARATION VAR_NAME: t1\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: ASSIGNMENT START\n\
[DEBUG]: ASSIGN INT: 21\n\
[DEBUG]: Command line ended\n\
[DEBUG]: INIT VAR_NAME: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: print\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: ARG INT: 21\n\
[DEBUG]: ARG VARIABLE: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: get\n\
[DEBUG]: METHOD CALL START\n\
[DEBUG]: METHOD CALL END\n\
[DEBUG]: ARG VARIABLE: t1\n\
[DEBUG]: MEMBER ACCESS START\n\
[DEBUG]: ACCESS TO SUB-MEMBER: num\n\
[DEBUG]: METHOD CALL END\n\
printing Ints with t1.print: 21 21 21 \n\
[DEBUG]: EOF\n\
"
      )
    }
  }

  std::cout.rdbuf(sbuf);
}
```

B.1.3 Плагін підтримки протоколу TCP

B.1.3.1 tcp_plugin.h

```
#ifndef TCP_PLUGIN_H
#define TCP_PLUGIN_H

#include "parser/plugin.h"

#define TCP_PLUGIN_NAME "TCP_Grammar"

extern "C" const PluginAPI* const get_plugin();

#endif
```

B.1.3.2 tcp_plugin.cpp

```
#include "tcp_plugin.h"

#include "parser/variables_management.h"
#include "parser/type_management.h"
#include "parser/standard_types.h"

#include "protocol_types.h"
#include "utils.h"

// --- TcpStream ---

#define TCP_STREAM_NAME "TcpStream"

#define TCPSTREAM_MEMB_MAC "mac"
#define TCPSTREAM_MEMB_IPV4 "ip"
#define TCPSTREAM_MEMB_SRC_PORT "src_port"
#define TCPSTREAM_MEMB_DST_PORT "dst_port"

#define TCPSTREAM_METH_ADD "add"
#define TCPSTREAM_METH_TOSTRING "to_string"

class TcpStream : public TypeValue
{
public:
   TcpStream(const TypeAPI& self_type)
      : TypeValue(self_type)
   {}

   virtual bool set(const char* value) override
   {
      return false;
   }

   virtual bool set(const TypeValue* value) override
   {
      return false;
   }

   virtual void* get() const override
   {
      return nullptr;
   }

   /*
      TcpPacket packet - packet to be inserted into the stream,
will be copied,
      IntValue count - repeat the packet "count" times
      OR

      PacketDirection dir,
      TcpFlags flags (optional, many consecutive values
allowed),
      StringValue payload (optional)
   */

   bool add_packet(const TcpPacket& p, unsigned count = 1)
   {
      while (count--)
         packets.emplace_back(p);

      return true;
   }

   bool add_packet(const PacketDirection& dir, const
StringValue* payload,
      const TcpFlags* flags = nullptr)
   {
      TcpPacket& last = packets.emplace_back();

      if (packets.size() == 1)
         last.get_member(TCP_MEMB_SEQ)->set("1");
      else
      {
         int prev_packet_seq =
((IntValue*)std::next(packets.rbegin())
            ->get_const_member(TCP_MEMB_SEQ))-
>get_int();
         ((IntValue*)last.get_member(TCP_MEMB_SEQ))-
>set(prev_packet_seq + 1);
      }

      last.get_member(TCP_MEMB_MAC)-
>set(get_const_member(TCPSTREAM_MEMB_MAC));
      last.get_member(TCP_MEMB_IP)-
>set(get_const_member(TCPSTREAM_MEMB_IPV4));
      last.get_member(TCP_MEMB_SRC)-
>set(get_const_member(TCPSTREAM_MEMB_SRC_PORT));
      last.get_member(TCP_MEMB_DST)-
>set(get_const_member(TCPSTREAM_MEMB_DST_PORT));

      last.get_member(TCP_MEMB_DIR)->set(&dir);
      if (payload)
```

```
            last.get_member(TCP_MEMB_PAYLOAD)-
>set(payload);
         if (flags)
         {
            last.get_member(TCP_MEMB_FLAGS)->set(flags);

            if (flags->is_flag_set(TCP_FLAG_ACK))
            {
               auto prev_packet = std::next(packets.rbegin());
               if (prev_packet != packets.rend())
               {
                  last.get_member(TCP_MEMB_ACK)-
>set(prev_packet->get_member(
                     TCP_MEMB_SEQ));
               }
            }
         }

         return true;
      }

      std::string to_string() const
      {
         std::string res;
         for (const auto& p : packets)
         {
            res += p.to_string() + "\n";
         }
         return res;
      }

      static std::unique_ptr<Variable>
stream_add_packet(std::vector<Variable*>& args)
      {
         if (args.size() < 3)
            return nullptr;

         size_t arg_idx = 1;

         TcpStream* caller = (TcpStream*)args[arg_idx++]-
>get_value();

         if (args[arg_idx]->get_api()->get_code() ==
TAdmin::get_type_code(TCP_PACKET_NAME))
         {
            const TcpPacket* p = (const TcpPacket*)args[arg_idx+
+]->get_value();
            if (p)
            {
               unsigned p_count = 1;

               if (arg_idx < args.size() && args[arg_idx]-
>get_api()->get_code() ==
                  TAdmin::get_type_code(INT_NAME))
                  p_count = ((const IntValue*)args[arg_idx++]-
>get_value())->get_int();

               caller->add_packet(*p, p_count);
            }
            return nullptr;
         }

         if (args.size() < 4)
            return nullptr;

         if (args[arg_idx]->get_api()->get_code() !=
TAdmin::get_type_code(PKT_DIRECTION_NAME))
            return nullptr;

         const PacketDirection* dir = (const
PacketDirection*)args[arg_idx++]->get_value();
         if (!dir)
            return nullptr;

         unsigned flag_code =
TAdmin::get_type_code(TCP_FLAGS_NAME);
         std::unique_ptr<TcpFlags> fl(nullptr);

         while (arg_idx < args.size() &&
            args[arg_idx]->get_api()->get_code() == flag_code)
         {
            if (!fl)
               fl.reset(new TcpFlags());
            const TcpFlags* flag = (const TcpFlags*)args[arg_idx+
+]->get_value();
            fl->set_flag(flag->get_flags());
         }

         const StringValue* payload = arg_idx < args.size() &&
            args[arg_idx]->get_api()->get_code() ==
TAdmin::get_type_code(STRING_NAME) ?
            (const StringValue*)args[arg_idx++]->get_value() :
            nullptr;

         caller->add_packet(*dir, payload, fl.get());

         return nullptr;
      }
```

```
        static std::unique_ptr<Variable>
stream_to_string(std::vector<Variable *> &args)
        {
          if (args.size() != 2)
            return nullptr;

          std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));
          StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

          ret_string->set(((TcpStream*)args[1]->get_value())-
>to_string().c_str());

          return ret_val;
        }

      private:
        std::vector<TcpPacket> packets;
      };

      static std::unique_ptr<TypeValue>
construct_tcp_stream_type()
      {
        return std::unique_ptr<TypeValue>(
          new
TcpStream(*TAdmin::get_api(TCP_STREAM_NAME)));
      }

      static TypeAPI::members_map tcp_stream_members = {
        MEMBER_ENTRY(MacInfo,
TCPSTREAM_MEMB_MAC, MAC_INFO_NAME),
        MEMBER_ENTRY(IPv4Info,
TCPSTREAM_MEMB_IPV4, IPV4_INFO_NAME),
        MEMBER_ENTRY(IntValue,
TCPSTREAM_MEMB_SRC_PORT, INT_NAME),
        MEMBER_ENTRY(IntValue,
TCPSTREAM_MEMB_DST_PORT, INT_NAME),
        METHOD_ENTRY(TCPSTREAM_METH_ADD,
TcpStream::stream_add_packet),
        METHOD_ENTRY(TCPSTREAM_METH_TOSTRING,
TcpStream::stream_to_string)
      };

      static const TypeAPI tcp_stream_api = TypeAPI(
        TCP_STREAM_NAME,
        construct_tcp_stream_type,
        &tcp_stream_members
      );

      // ===== Plugin API =====

      static const TypeAPI* tcp_types[] =
      {
        // info storage types
        &mac_value_api,
        &mac_info_api,
        &ipv4_info_api,
        &tcp_flags_api,
        &packet_dir_api,
        // packet type
        &tcp_packet_api,
        // stream type
        &tcp_stream_api,
        nullptr
      };

      static const TypeAPI::members_map tcp_vars =
      {
        // tcp flags
        { TCP_FLAG_ACK_NAME, flag_ack},
        { TCP_FLAG_SYN_NAME, flag_syn},
        { TCP_FLAG_FIN_NAME, flag_fin},
        { TCP_FLAG_RST_NAME, flag_rst},
        { TCP_FLAG_PSH_NAME, flag_psh},
        { TCP_FLAG_URG_NAME, flag_urg},
        // packet directions
        { DIR_C2S_NAME, dir_c2s },
        { DIR_S2C_NAME, dir_s2c }
      };

      static const PluginAPI tcp_plugin =
      {
        TCP_PLUGIN_NAME,
        tcp_types,
        &tcp_vars
      };

      const PluginAPI* const get_plugin()
      {
        return &tcp_plugin;
      }
```

B.1.3.3 protocol_types.h

```
#ifndef PROTOCOL_TYPES_H
#define PROTOCOL_TYPES_H

#include "parser/variables_management.h"

#define MAC_VALUE_NAME "MacValue"

// ---

class MacValue : public TypeValue
{
public:
   MacValue(const TypeAPI& self_type);

   virtual bool set(const char* value) override;
   virtual bool set(const TypeValue* value) override;
   virtual void* get() const override;

   static std::unique_ptr<Variable>
mac_to_string(std::vector<Variable*>& args);

private:
   union
   {
      uint64_t addr64;
      uint8_t addr8[8];
   };

   std::string to_string() const;
};

extern const TypeAPI mac_value_api;

// ---

#define MAC_INFO_NAME "MacInfo"

class MacInfo : public TypeValue
{
public:
   MacInfo(const TypeAPI& self_type);

   virtual bool set(const char* value) override;
   virtual bool set(const TypeValue* value) override;
   virtual void* get() const override;
};

extern const TypeAPI mac_info_api;

// ---

#define IPV4_INFO_NAME "IPv4Info"

#define IPV4INFO_MEMB_SRC "src"
#define IPV4INFO_MEMB_DST "dst"

class IPv4Info : public TypeValue
{
public:
   IPv4Info(const TypeAPI& self_type);

   virtual bool set(const char* value) override;
   virtual bool set(const TypeValue* value) override;
   virtual void* get() const override;
};

extern const TypeAPI ipv4_info_api;

// ---
#define TCP_PACKET_NAME "TcpPacket"

#define TCP_MEMB_MAC "mac"
#define TCP_MEMB_IP "ip"
#define TCP_MEMB_DIR "dir"
#define TCP_MEMB_SRC "src_port"
#define TCP_MEMB_DST "dst_port"
#define TCP_MEMB_SEQ "seq_num"
#define TCP_MEMB_ACK "ack_num"
#define TCP_MEMB_FLAGS "flags"
#define TCP_MEMB_PAYLOAD "payload"
#define TCP_METH_TOSTRING "to_string"

class TcpPacket : public TypeValue
{
public:
   TcpPacket();
   TcpPacket(const TypeAPI& self_type);

   virtual bool set(const char* value) override;
   virtual bool set(const TypeValue* value) override;
   virtual void* get() const override;

   std::string to_string() const;
```

```
        static std::unique_ptr<Variable>
packet_to_string(std::vector<Variable*>& args);
    };

    extern const TypeAPI tcp_packet_api;

    // ---

    #define TCP_FLAGS_NAME "TcpFlags"

    #define TCP_FLAG_FIN_NAME "FIN"
    #define TCP_FLAG_SYN_NAME "SYN"
    #define TCP_FLAG_RST_NAME "RST"
    #define TCP_FLAG_PSH_NAME "PSH"
    #define TCP_FLAG_ACK_NAME "ACK"
    #define TCP_FLAG_URG_NAME "URG"

    #define TCP_FLAG_FIN 0b0000000000000001
    #define TCP_FLAG_SYN 0b0000000000000010
    #define TCP_FLAG_RST 0b0000000000000100
    #define TCP_FLAG_PSH 0b0000000000001000
    #define TCP_FLAG_ACK 0b0000000000010000
    #define TCP_FLAG_URG 0b0000000000100000

    class TcpFlags : public TypeValue
    {
    public:
      TcpFlags();
      TcpFlags(const TypeAPI& self_type);
      TcpFlags(const TypeAPI& self_type, uint16_t value);

      virtual bool set(const char* value) override;
      virtual bool set(const TypeValue* value) override;
      virtual void* get() const override;

      bool set_flag(uint16_t flag);
      bool unset_flag(uint16_t flag);
      bool is_flag_set(uint16_t flag) const;
      uint16_t get_flags() const;
      std::string to_string() const;

      static std::unique_ptr<Variable>
set_flags(std::vector<Variable*>& args);
      static std::unique_ptr<Variable>
unset_flags(std::vector<Variable*>& args);
      static std::unique_ptr<Variable>
flags_to_string(std::vector<Variable*>& args);

    private:
      uint16_t flags;

    };

    extern TypeAPI::member_init flag_ack;
    extern TypeAPI::member_init flag_syn;
    extern TypeAPI::member_init flag_fin;
    extern TypeAPI::member_init flag_rst;
    extern TypeAPI::member_init flag_psh;
    extern TypeAPI::member_init flag_urg;

    extern const TypeAPI tcp_flags_api;

    // ---

    #define PKT_DIRECTION_NAME "Direction"

    #define DIR_C2S_NAME "C2S"
    #define DIR_S2C_NAME "S2C"

    class PacketDirection : public TypeValue
    {
    public:
      enum Direction
      {
        C2S,
        S2C
      };

      PacketDirection(const TypeAPI& self_type);
      PacketDirection(const TypeAPI& self_type, Direction
value);

      virtual bool set(const char* value) override;
      virtual bool set(const TypeValue* value) override;
      virtual void* get() const override;

      bool set_direction(Direction dir);
      bool is_c2s() const;
      std::string to_string() const;

      static std::unique_ptr<Variable>
dir_to_string(std::vector<Variable*>& args);

    private:
      Direction direction;
    };

    extern TypeAPI::member_init dir_c2s;
```

```
extern TypeAPI::member_init dir_s2c;

extern const TypeAPI packet_dir_api;
```

B.1.3.4 protocol_types.cpp

```
#include "protocol_types.h"

#include <cstring>
#include <bitset>

#include "parser/type_management.h"
#include "parser/standard_types.h"

#include "utils.h"

// ---

#define MAC_DELIM ".-:"

#define MACVAL_METH_TOSTRING "to_string"

MacValue::MacValue(const TypeAPI& self_type)
   : TypeValue(self_type)
{}

bool MacValue::set(const char *value)
{
   // can be assigned with a string containing the value

   // TODO: make MAC address a built-in type, the same as for
IPv4 addresses
   // trim string double quotes
   if (*value == '"')
      value++;

   size_t val_len = strlen(value);

   if (value[val_len - 1] == '\"')
      val_len--;

   char* proc_buf = new char[val_len];
   strncpy(proc_buf, value, val_len);

   char* byte = std::strtok(proc_buf, MAC_DELIM);
   unsigned short byte_idx = 0;
   while (byte && byte_idx < 8)
   {
      try
      {
         unsigned long parsed = strtoul(byte, nullptr, 16);
         if (parsed < 0 || parsed >= 256)
         {
            delete[] proc_buf;
            return false;
         }
         addr8[byte_idx++] = parsed;
      }
      catch(const std::exception& e)
      {
         delete[] proc_buf;
         return false;
      }

      byte = std::strtok(nullptr, MAC_DELIM);
   }

   delete[] proc_buf;

   return true;
}

bool MacValue::set(const TypeValue *value)
{
   // can be assigned from another MacValue
   if (value->get_type().get_code() != type.get_code())
      return false;

   this->addr64 = ((MacValue*)value)->addr64;

   return true;
}

void *MacValue::get() const
{
   return (void*)&addr64;
}

std::string MacValue::to_string() const
{
   char buffer[8 * 2 + 7 + 1]; // 8 bytes 2 chars each + 7
dividers + \0

   sprintf(buffer, "%x:%x:%x:%x:%x:%x:%x:%x",
#endif
```

```
            addr8[0], addr8[1], addr8[2], addr8[3],
            addr8[4], addr8[5], addr8[6], addr8[7]);

        return buffer;
    }

    std::unique_ptr<Variable>
MacValue::mac_to_string(std::vector<Variable *> &args)
    {
        if (args.size() != 2)
            return nullptr;

        std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));
        StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

        ret_string->set(((MacValue*)args[1]->get_value())-
>to_string().c_str());

        return ret_val;
    }

    static std::unique_ptr<TypeValue> construct_mac_value()
    {
        return std::unique_ptr<TypeValue>(
            new
MacValue(*TAdmin::get_api(MAC_VALUE_NAME)));
    }

    static TypeAPI::members_map macvalue_members = {
        METHOD_ENTRY(MACVAL_METH_TOSTRING,
MacValue::mac_to_string)
    };

    const TypeAPI mac_value_api = TypeAPI(
        MAC_VALUE_NAME,
        construct_mac_value,
        &macvalue_members
    );

    // ---

    #define MAC_MEMB_SRC "src"
    #define MAC_MEMB_DST "dst"

    MacInfo::MacInfo(const TypeAPI& self_type)
        : TypeValue(self_type)
    {}

    bool MacInfo::set(const char *value)
    {
        return false;
    }

    bool MacInfo::set(const TypeValue *value)
    {
        // can be assigned from another MacInfo
        if (value->get_type().get_code() != type.get_code())
            return false;

        MacInfo* other = (MacInfo*)value;

        members.get_var(MAC_MEMB_SRC)->get_value()->set(
            other->get_members().get_var(MAC_MEMB_SRC)-
>get_value());
        members.get_var(MAC_MEMB_DST)->get_value()->set(
            other->get_members().get_var(MAC_MEMB_DST)-
>get_value());

        return true;
    }

    void *MacInfo::get() const
    {
        return nullptr;
    }

    static TypeAPI::members_map macinfo_members = {
        MEMBER_ENTRY(MacValue, MAC_MEMB_SRC,
MAC_VALUE_NAME),
        MEMBER_ENTRY(MacValue, MAC_MEMB_DST,
MAC_VALUE_NAME)
    };

    static std::unique_ptr<TypeValue> construct_mac_info()
    {
        return std::unique_ptr<TypeValue>(
            new MacInfo(*TAdmin::get_api(MAC_INFO_NAME)));
    }

    const TypeAPI mac_info_api = TypeAPI(
        MAC_INFO_NAME,
        construct_mac_info,
        &macinfo_members
    );

    // ---
```

```
IPv4Info::IPv4Info(const TypeAPI& self_type)
    : TypeValue(self_type)
{}

bool IPv4Info::set(const char *value)
{
    return false;
}

bool IPv4Info::set(const TypeValue *value)
{
    // can be assigned from another IPv4Info
    if (value->get_type().get_code() != type.get_code())
        return false;

    IPv4Info* other = (IPv4Info*)value;

    members.get_var(IPV4INFO_MEMB_SRC)->get_value()->set(
        other->get_members().get_var(IPV4INFO_MEMB_SRC)->get_value());
    members.get_var(IPV4INFO_MEMB_DST)->get_value()->set(
        other->get_members().get_var(IPV4INFO_MEMB_DST)->get_value());

    return true;
}

void *IPv4Info::get() const
{
    return nullptr;
}

static TypeAPI::members_map ipv4info_members = {
    MEMBER_ENTRY(IPv4Value, IPV4INFO_MEMB_SRC, IPv4_NAME),
    MEMBER_ENTRY(IPv4Value, IPV4INFO_MEMB_DST, IPv4_NAME)
};

static std::unique_ptr<TypeValue> construct_ipv4_info()
{
    return std::unique_ptr<TypeValue>(
        new IPv4Info(*TAdmin::get_api(IPV4_INFO_NAME)));
}

const TypeAPI ipv4_info_api = TypeAPI(
    IPV4_INFO_NAME,
    construct_ipv4_info,
    &ipv4info_members
);

// ---

TcpPacket::TcpPacket()
    : TypeValue(*TAdmin::get_api(TCP_PACKET_NAME))
{}

TcpPacket::TcpPacket(const TypeAPI &self_type)
    : TypeValue(self_type)
{}

bool TcpPacket::set(const char *value)
{
    return false;
}

bool TcpPacket::set(const TypeValue *value)
{
    // can be assigned from another TcpPacket
    if (value->get_type().get_code() != type.get_code())
        return false;

    return true;
}

void *TcpPacket::get() const
{
    return nullptr;
}

std::string TcpPacket::to_string() const
{
    const IPv4Info* ip_info = (const IPv4Info*)get_const_member(TCP_MEMB_IP);
    const IPv4Value* ip_src = (const IPv4Value*)ip_info->get_const_member(IPV4INFO_MEMB_SRC);
    const IPv4Value* ip_dst = (const IPv4Value*)ip_info->get_const_member(IPV4INFO_MEMB_DST);

    const IntValue* port_src = (const IntValue*)get_const_member(TCP_MEMB_SRC);
    const IntValue* port_dst = (const IntValue*)get_const_member(TCP_MEMB_DST);
    const IntValue* seq = (const IntValue*)get_const_member(TCP_MEMB_SEQ);
```

```
        const IntValue* ack = (const
IntValue*)get_const_member(TCP_MEMB_ACK);



        const PacketDirection* dir =
((PacketDirection*)get_const_member(TCP_MEMB_DIR));

        std::string src = ip_src->to_string() + ':' +
std::to_string(port_src->get_int());
        std::string dst = ip_dst->to_string() + ':' +
std::to_string(port_dst->get_int());

        const TcpFlags* flags =
((TcpFlags*)get_const_member(TCP_MEMB_FLAGS));

        return (dir->is_c2s() ?
           src + " -> " + dst :
           dst + " -> " + src ) +
           "\nFlags: " + flags->to_string() +
           "\nSeq: " + std::to_string(seq->get_int()) +
           ", Ack: " + std::to_string(ack->get_int()) + '\n';
    }

    std::unique_ptr<Variable>
TcpPacket::packet_to_string(std::vector<Variable *> &args)
    {
       if (args.size() != 2)
          return nullptr;

       std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));
       StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

       ret_string->set(((TcpPacket*)args[1]->get_value())-
>to_string().c_str());

       return ret_val;
    }

    static TypeAPI::members_map tcppacket_members = {
       MEMBER_ENTRY(MacInfo, TCP_MEMB_MAC,
MAC_INFO_NAME),
       MEMBER_ENTRY(IPv4Info, TCP_MEMB_IP,
IPV4_INFO_NAME),
       MEMBER_ENTRY(PacketDirection, TCP_MEMB_DIR,
PKT_DIRECTION_NAME),
       MEMBER_ENTRY(IntValue, TCP_MEMB_SRC,
INT_NAME),
       MEMBER_ENTRY(IntValue, TCP_MEMB_DST,
INT_NAME),
       MEMBER_ENTRY(IntValue, TCP_MEMB_SEQ,
INT_NAME),
       MEMBER_ENTRY(IntValue, TCP_MEMB_ACK,
INT_NAME),
       MEMBER_ENTRY(TcpFlags, TCP_MEMB_FLAGS,
TCP_FLAGS_NAME),
       MEMBER_ENTRY(StringValue, TCP_MEMB_PAYLOAD,
STRING_NAME),
       METHOD_ENTRY(TCP_METH_TOSTRING,
TcpPacket::packet_to_string)
    };

    static std::unique_ptr<TypeValue> construct_tcp_info()
    {
       return std::unique_ptr<TypeValue>(
          new
TcpPacket(*TAdmin::get_api(TCP_PACKET_NAME)));
    }

    const TypeAPI tcp_packet_api = TypeAPI(
       TCP_PACKET_NAME,
       construct_tcp_info,
       &tcppacket_members
    );

    // ---

    #define TCPFLAGS_METH_SET "set"
    #define TCPFLAGS_METH_UNSET "unset"
    #define TCPFLAGS_METH_TOSTRING "to_string"

    TcpFlags::TcpFlags()
       : TypeValue(*TAdmin::get_api(TCP_FLAGS_NAME)),
flags(0)
    {}

    TcpFlags::TcpFlags(const TypeAPI &self_type)
       : TypeValue(self_type), flags(0)
    {}

    TcpFlags::TcpFlags(const TypeAPI &self_type, uint16_t value)
       : TypeValue(self_type), flags(value)
    {}

    bool TcpFlags::set(const char *value)
    {
       return false;
```

```
}

bool TcpFlags::set(const TypeValue *value)
{
   // can be assigned from another TcpFlags
   if (value->get_type().get_code() != type.get_code())
      return false;

   flags = ((TcpFlags*)value)->flags;

   return true;
}

void *TcpFlags::get() const
{
   return nullptr;
}

bool TcpFlags::set_flag(uint16_t flag)
{
   flags |= flag;

   return true;
}

bool TcpFlags::unset_flag(uint16_t flag)
{
   flags &= ~flag;

   return true;
}

bool TcpFlags::is_flag_set(uint16_t flag) const
{
   return flags & flag;
}

uint16_t TcpFlags::get_flags() const
{
   return flags;
}

std::unique_ptr<Variable>
TcpFlags::set_flags(std::vector<Variable *> &args)
{
   if (args.size() < 3 || args[1]->get_api()->get_code() !=
      TAdmin::get_api(TCP_FLAGS_NAME)->get_code())
      return nullptr;

   TcpFlags* caller = (TcpFlags*)args[1]->get_value();
   size_t flag_idx = 2;

   while(flag_idx < args.size())
   {
      if (args[flag_idx]->get_api()->get_code() !=
         TAdmin::get_api(TCP_FLAGS_NAME)->get_code())
         return nullptr;

      TcpFlags* flag = (TcpFlags*)args[flag_idx++]->get_value();
      caller->set_flag(flag->flags);
   }

   return nullptr;
}

std::unique_ptr<Variable>
TcpFlags::unset_flags(std::vector<Variable *> &args)
{
   if (args.size() < 3 || args[1]->get_api()->get_code() !=
      TAdmin::get_api(TCP_FLAGS_NAME)->get_code())
      return nullptr;

   TcpFlags* caller = (TcpFlags*)args[1]->get_value();
   size_t flag_idx = 2;

   while(flag_idx < args.size())
   {
      if (args[flag_idx]->get_api()->get_code() !=
         TAdmin::get_api(TCP_FLAGS_NAME)->get_code())
         return nullptr;

      TcpFlags* flag = (TcpFlags*)args[flag_idx++]->get_value();
      caller->unset_flag(flag->flags);
   }

   return nullptr;
}

std::unique_ptr<Variable>
TcpFlags::flags_to_string(std::vector<Variable *> &args)
{
   if (args.size() != 2)
      return nullptr;

   std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));
```

```
        StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

        ret_string->set(((TcpFlags*)args[1]->get_value())-
>to_string().c_str());

        return ret_val;
    }

    std::string TcpFlags::to_string() const
    { // TODO: should bitset be used as the main storage for these
values?
        std::string ret = std::bitset<16>(flags).to_string();

        if (flags)
            ret += "\n";
        if (flags & TCP_FLAG_FIN)
            ret += " " TCP_FLAG_FIN_NAME;
        if (flags & TCP_FLAG_SYN)
            ret += " " TCP_FLAG_SYN_NAME;
        if (flags & TCP_FLAG_RST)
            ret += " " TCP_FLAG_RST_NAME;
        if (flags & TCP_FLAG_PSH)
            ret += " " TCP_FLAG_PSH_NAME;
        if (flags & TCP_FLAG_ACK)
            ret += " " TCP_FLAG_ACK_NAME;
            if (flags & TCP_FLAG_URG)
            ret += " " TCP_FLAG_URG_NAME;

        return ret;
    }

    static std::unique_ptr<TypeValue> construct_tcp_flags()
    {
        return std::unique_ptr<TypeValue>(
            new
TcpFlags(*TAdmin::get_api(TCP_FLAGS_NAME)));
    }

    static TypeAPI::members_map tcpflags_members = {
        METHOD_ENTRY(TCPFLAGS_METH_SET,
TcpFlags::set_flags),
        METHOD_ENTRY(TCPFLAGS_METH_UNSET,
TcpFlags::unset_flags),
        METHOD_ENTRY(TCPFLAGS_METH_TOSTRING,
TcpFlags::flags_to_string)
    };

    const TypeAPI tcp_flags_api = TypeAPI(
        TCP_FLAGS_NAME,
        construct_tcp_flags,
        &tcpflags_members
    );

    TypeAPI::member_init flag_ack =
TCP_FLAG_DECL(TCP_FLAG_ACK);
    TypeAPI::member_init flag_syn =
TCP_FLAG_DECL(TCP_FLAG_SYN);
    TypeAPI::member_init flag_fin =
TCP_FLAG_DECL(TCP_FLAG_FIN);
    TypeAPI::member_init flag_rst =
TCP_FLAG_DECL(TCP_FLAG_RST);
    TypeAPI::member_init flag_psh =
TCP_FLAG_DECL(TCP_FLAG_PSH);
    TypeAPI::member_init flag_urg =
TCP_FLAG_DECL(TCP_FLAG_URG);

    // ---

    #define DIR_METH_SET "set"
    #define DIR_METH_TOSTRING "to_string"

    PacketDirection::PacketDirection(const TypeAPI& self_type)
        : TypeValue(self_type), direction(S2C)
    {}

    PacketDirection::PacketDirection(const TypeAPI &self_type,
Direction value)
        : TypeValue(self_type), direction(value)
    {}

    bool PacketDirection::set(const char *value)
    {
        return false;
    }

    bool PacketDirection::set(const TypeValue *value)
    {
        // can be assigned from another PacketDirection
        if (value->get_type().get_code() != type.get_code())
            return false;

        direction = ((PacketDirection*)value)->direction;

        return true;
    }

    void *PacketDirection::get() const
```

```
    {
        return nullptr;
    }

    bool PacketDirection::set_direction(Direction dir)
    {
        direction = dir;

        return true;
    }

    bool PacketDirection::is_c2s() const
    {
        return direction == C2S;
    }

    std::unique_ptr<Variable>
PacketDirection::dir_to_string(std::vector<Variable*>& args)
    {
        if (args.size() != 2)
            return nullptr;

        std::unique_ptr<Variable> ret_val(new Variable("",
*TAdmin::get_api(STRING_NAME)));
        StringValue* ret_string = (StringValue*)ret_val.get()-
>get_value();

        ret_string->set(((PacketDirection*)args[1]->get_value())-
>to_string().c_str());

        return ret_val;
    }

    std::string PacketDirection::to_string() const
    {
        return direction == C2S ? DIR_C2S_NAME :
DIR_S2C_NAME;
    }

    static std::unique_ptr<TypeValue>
construct_packet_direction()
    {
        return std::unique_ptr<TypeValue>(
            new
PacketDirection(*TAdmin::get_api(PKT_DIRECTION_NAME)));
    }

    static TypeAPI::members_map direction_members = {
        METHOD_ENTRY(DIR_METH_TOSTRING,
PacketDirection::dir_to_string)
    };

    const TypeAPI packet_dir_api = TypeAPI(
        PKT_DIRECTION_NAME,
        construct_packet_direction,
        &direction_members
    );

    TypeAPI::member_init dir_c2s =
GLOBAL_VAR_DECL(PacketDirection,
        PKT_DIRECTION_NAME, PacketDirection::C2S);
    TypeAPI::member_init dir_s2c =
GLOBAL_VAR_DECL(PacketDirection,
        PKT_DIRECTION_NAME, PacketDirection::S2C);
```

B.1.3.5 utils.h

```
#ifndef UTILS_H
#define UTILS_H

#define MEMBER_ENTRY(type, var_name, type_name) \
    { var_name, []() -> TypeAPI::member_type {  \
        return std::make_unique<type>(          \
            *TAdmin::get_api(type_name));       \
    } }

#define METHOD_ENTRY(method_name, callback)                    \
    { method_name, []() -> TypeAPI::member_type {              \
        return std::make_unique<Function>(                     \
            *TAdmin::get_api(FUNC_TYPE_NAME), callback);       \
    } }
```

```
#define GLOBAL_VAR_DECL(type, type_name, var_value)     \
   []() -> TypeAPI::member_type {                       \
      return std::make_unique<type>(                    \
         *TAdmin::get_api(type_name), var_value);    \
}

#define TCP_FLAG_DECL(var_value)\
   GLOBAL_VAR_DECL(TcpFlags, TCP_FLAGS_NAME, var_value)

#endif
```

ДОДАТОК Г

Тези доповіді

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XII Міжнародної науково-практичної конференції молодих учених та студентів
6-7 грудня 2023 року

УКРАЇНА
ТЕРНОПІЛЬ – 2023

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University (Ukraine)
Pierre and Marie Curie University (The French Republic)
University of Maribor (The Republic of Slovenia)
Technical University of Kosice (The Slovak Republic)
Vilnius Gediminas Technical University (The Republic of Lithuania)
International Academy Mohammed VI of Civil Aviation (Morocco)
T. Shevchenko Scientific Society

CURRENT ISSUES IN MODERN TECHNOLOGIES

Book

of abstracts

of the XII International scientific and practical conference of young researchers and students

December, 6th-7th, 2023

UKRAINE
TERNOPIL – 2023

A43
Actual problems of modern technologies: book of abstracts of the XII International scientific and practical conference of young researchers and students, (Ternopil, December, 6th-7th, 2023) / Ministry of Education and Science of Ukraine, Ternopil Ivan Puluj National Technical University [and other.]. – Ternopil: PE Palianytsia V.A., 2023. – 500.

ISBN 978-617-7875-71-9

PROGRAM COMMITTEE

The address of the organization committee:
TNTU, Ruska str. 56, Ternopil, 46001,
tel. (0352) 519724, fax (0352) 254983
E-mail: **: tarasdowbush@gmail.com**
Editing, design, layout: Dovbush T.A.

TOPICS OF THE CONFERENCE

– Physical and Technical Fundamentals of New Technologies Development;
– New Materials, Strength and Durability of the Constructions Elements;
– Modern Technologies in Construction, Machine- and Instrument-Building;
– Modern Technologies in Transport Area;
– Electrical Engineering and Energy Efficiency;
– Fundamental Issues of Food, Bio and Nanotechnologies;
– Economic and Social Aspects of New Technologies;
– Computer and Information Technologies and Communication Systems.

ISBN 978-617-7875-71-9

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XII Міжнародної науково-практичної конференції молодих учених та студентів
6-7 грудня 2023 року

УКРАЇНА
ТЕРНОПІЛЬ – 2023

УДК 004.45+004.94
А. В. Сербенюк, О. С. Куроп'ятник, к. т. н., доц.
(Український державний університет науки і технологій, Україна)

ФОРМУВАННЯ МЕРЕЖЕВОГО ТРАФІКУ З ВИКОРИСТАННЯМ ФОРМАЛЬНИХ ГРАМАТИК

A. V. Serbeniuk, O. S. Kuropiatnyk, Ph. D., Assoc. Prof.

TRAFFIC SHAPING BY USING FORMAL GRAMMAR

В сучасному світі важко уявити функціонування суспільства без мережі Інтернет. Її використання дозволяє вирішувати безліч проблем. Тому, забезпечення працездатності мережі Інтернет та її складових є важливим завданням. Це включає в себе безліч дій: від перевірки фізичної справності компонентів та тестування систем на вразливість до атак та нестандартного Інтеренет-трафіку, до простої перевірки наявності з'єднання з мережею.

Для тестування трафіку існує чимало інструментів, які можуть його формувати. Здебільшого для їх використання користувачам надається графічний або консольний інтерфейс із фіксованим набором функцій, та способи використання програми обмежені тим, що в ньому надається. У випадку, коли наданого функціоналу недостатньо, виникає питання розширення функціоналу або пошуку альтернативних засобів.

Великим обмеженням в цьому питанні є те, що людина не завжди може точно сформулювати та передати вимоги до генерованого трафіку для програми, яка це робить. Проблема опису і передачі вимог між людьми вирішується засобами природної мови, тому пропонується рішення цієї проблеми з використанням деякої формальної мови і для представлення трафіку програмі-генератору. Такий спосіб повинен дозолити користувачу описувати вимоги до трафіку точніше та зручніше. Для опису такої мови та подальшої її автоматизованої обробки доцільним є використання формальних граматик автоматного типу.

Щоб надати таку ступінь гнучкості опису трафіку, повинна бути можливість доступу до багатьох рівнів моделі OSI. Механізми опису для будь якого рівня або протоколу може бути той же — у вигляді деякого об'єкту зі значеннями. Наприклад, може бути як об'єкт для опису базового мережевого пакету протоколу TCP, так і об'єкт для опису всього потоку пакетів у транзакціях протоколу HTTP.

Судячи із цього, синтаксис граматики, що описуватиме цю мову, повинен надавати можливість оголошення цих об'єктів, присвоєння значення їм та їхнім складовим частинам, доступ до їхніх складових частин. Крім того, ці об'єкти можуть мати власний функціонал, тому потрібно передбачити можливість доступу до нього. Тобто, синтаксис повинен надавати можливість виклику функцій, передачі їм аргументів. Прикладом використання подібних функцій може бути доступ до елементів об'єктів, які були створені або підраховані безпосередньо в процесі обробки вхідного тексту, як розмір деяких об'єктів, кількість пакетів у них, тощо.

Граматика повинна описувати базові синтаксичні елементи мови, а безпосередньо функціонал буде вибудовуватись на її основі. Функціонал представлення трафіку буде надаватись у вигляді типів даних, які описують його або його складові. Наприклад, тип для представлення TCP пакету. Така граматика описується четвіркою $G = \langle T, N, A, P \rangle$, де T – множина терміналів, що описує компоненти пакетів та команди, N – множина не терміналів, які використовуються як допоміжні елементи при формуванні ланцюжку-опису пакету, $A \in N$ – початковий символ (аксіома), P – множина правил продукцій. Наведемо приклади елементів множини правил, де всі лексеми, що пишуться з великої букви, позначають нетермінали, символ «|» – варіативність підстановки:

$$A \rightarrow Line|LineA,$$
$$Line \rightarrow Command|CommandComment$$
$$Command \rightarrow Declaration|Assignment|Access$$
$$Declaration \rightarrow Type\ Var|Type\ Var\ Assignment$$

Оскільки мережевих протоколів та способів їх використання багато, то варто врахувати можливість розширення функціоналу програми за рахунок додаткових модулів та бібліотек.

В якості програмної реалізації формування трафіку на основі формальних граматик пропонується консольний додаток мовою C++, розробка якого триває. Розробка орієнтована на UNIX-подібні операційні системи.

Додаток отримуватиме вхідний текст для обробки із стандартного вхідного потоку, що дозволяє інтегрувати його в механізм конвеєрів UNIX-подібних операційних систем та надає свободу в способі безпосередньо передачі тексту в програму, від ручного введення, до зчитування із файлу, або обробки виведення іншого додатку.

Очікуваним результатом виконання програми є обробка вхідного тексту та команд із нього, тому вихідними даними додатку є індикація успішності обробки тексту. Формування описаного трафіку ініціюватиметься командою у вхідному тексті.

Додаток працює зі спроектованою граматикою. Синтаксис цієї граматики підтримує операції оголошення змінних деякого статичного типу, присвоєння значень, доступ до елементів змінних, виклику функцій. Вбудовані типи складаються із цілого числа, рядка тексту, та адреси протоколу IPv4. Типи для представлення трафіку оформлюватимуться у вигляді окремих модулів, які завантажуватимуться додатково. В межах даної розробки, буде розроблено типи для представлення протоколу TCP.

Для забезпечення гнучкості пропонується система плагінів. Плагін – це динамічна бібліотека із розширенням «.so», що містить інформацію про типи даних, які можуть використовуватись основним додатком. Це дозволяє розширювати функціонал додатку без внесення змін в нього самого, що також дозволяє уникати його рекомпіляції при цьому. Для підключення плагіну достатньо помістити його в директорію, де програма зможе його знайти. Шлях до директорії буде задаватись за допомогою змінної оточення, яку, за бажання, користувач також може змінити.

Результатом підвантаження плагіну є доступність нових типів представлення трафіку в мові, якими користувач може негайно ж користуватись. Відсутність сталого програмного інтерфейсу для представлення трафіку дозволяє уникнути необхідності його розширення,

Таким чином, перевагами додатку, що розробляється, є:

- більш природний спосіб опису складових трафіку за рахунок використання для цього граматики;
- простота автоматизації, що досягається завдяки сумісності і відповідності стандартам консольних додатків UNIX-подібних операційних систем;
- можливість розширення існуючого функціоналу за рахунок системи плагінів.

Основними напрямами для подальшої роботи є додавання підтримки різноманітних мережевих протоколів. Наявність функціоналу, відповідного аналогам, дозволить провести їх пряме порівняння.

У підсумку можна дійти висновку, що подібна розробка може бути корисною для представлення широкого спектру видів мережевого трафіку. Навіть якщо деякий вид трафіку не має готової імплементації, щоб його представити, її можна розробити та інтегрувати в існуючий програмний каркас, а текстове представлення трафіку значно полегшує доступність нового функціоналу користувачам, адже не потрібно додавати нові елементи інтерфейсу.