Dydaktyczny symulator wybranych rozwiązań warstwy fizycznej sieci Ethernet

Imię i nazwisko studenta: **Michał Iwanicki**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Algorytmy i modelowanie systemów**

Imię i nazwisko studenta: **Mateusz Bauer**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Teleinformatyka**

Imię i nazwisko studenta: **Marcin Garnowski**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Inteligentne Systemy Interaktywne**

Promotor pracy: **dr. inż. Krzysztof Nowicki**

Spis treści

1 Środowisko programistyczne

1.1 Język programowania

Do stworzenia symulatora wybrano język programowania Python z uwagi na kilka istotnych powodów. Przede wszystkim, czytelność składni stanowi ogromne ułatwienie podczas wspólnego tworzenia oprogramowania, a prostota pozwala skupić się na istocie problemu, nie tracąc czasu na pokonywanie trudności języka.

Dodatkowo, wybór Pythona jest motywowany chęcią rozwijania naszych umiejętności w tym środowisku, zarówno na poziomie indywidualnym, jak i zawodowym. Python cieszy się dużą popularnością jako uniwersalny język programowania, używany w różnych dziedzinach, takich jak analiza danych, sztuczna inteligencja czy aplikacje webowe. Posiadanie umiejętności programowania w Pythonie otwiera drzwi do szerszych możliwości zawodowych i dostępu do różnorodnych ciekawych projektów.

Jednym z najważniejszych argumentów przemawiających za wyborem Pythona jest jego ogromna popularność. Związana z tym społeczność programistyczna tworzy rozbudowany ekosystem, oferujący dostęp do wielu gotowych rozwiązań, bibliotek i frameworków. W kontekście tworzenia symulatora, istnieje wiele bibliotek w Pythonie, które mogą okazać się niezwykle przydatne. Na przykład, biblioteki umożliwiające tworzenie interfejsów graficznych ułatwią korzystanie z symulatora, biblioteki do analizy i przetwarzania sygnałów pomogą modelować różne aspekty transmisji, a biblioteki do wizualizacji pozwolą na przedstawienie wyników w przystępny sposób.

Inną cechą, która wyróżnia ten język programowania, jest jego przenośność. To ma dla nas duże znaczenie przy tworzeniu symulatora, który musi działać w warunkach laboratoryjnych, a więc na dowolnym popularniejszym systemie operacyjnym oraz charakteryzować się łatwością instalacji. Te wymagania Python w naszej ocenie spełnia.

1.2 Narzędzia, biblioteki i moduły

Jednym z celów postawionych przez promotora jest wykorzystanie gotowych rozwiązań podczas pracy nad symulatorem. W tym rozdziale zostaną przedstawione biblioteki i moduły języka Python oraz inne narzędzia, które mogą zostać wykorzystane w programie.

Python oferuje wiele bibliotek, które mogą okazać się kluczowe: od interfejsu graficznego po gotowe narzędzia do symulacji. Oto przegląd kilku z nich, na które się zdecydowano:

1. PyQt będzie biblioteką wykorzystywaną do stworzenia interfejsu graficznego użytkownika (GUI) dla symulatora. PyQt zapewnia szeroki zakres narzędzi do tworzenia rozbudowanych i przyjaznych użytkownikowi interfejsów, co jest szczególnie ważne w symulatorze dydaktycznym, gdzie interfejs musi być intuicyjny i nie stanowić niepotrzebnego wyzwania lub problemu dla biorących udział studentów

2. NumPy jest najpopularniejszą biblioteką Python implementującą algorytmy matematyczne. Między innymi oferuje generatory liczb pseudolosowych o różnych rozkładach, co jest wymagane do prawidłowego generowania ramek ethernetowych i błędów

3. Scapy umożliwia tworzenie i przetwarzanie różnego rodzaju pakietów sieciowych, w tym ramek Ethernet, co jest kluczową funkcjonalnością symulatora

4. Matplotlib to popularna biblioteka do tworzenia wykresów. Może okazać się przydatna przy tworzeniu wykresów sygnałów

Istnieją również inne popularne narzędzia, które mogą być użyteczne do symulacji rozwiązań warstwy fizycznej sieci Ethernet:

1. SPICE (Simulation Program with Integrated Circuit Emphasis) jest powszechnie stosowanym narzędziem do symulacji obwodów elektronicznych. Jest to rozbudowany program, który umożliwia modelowanie i analizę zachowania obwodów złożonych, takich jak układy analogowe, cyfrowe czy mikroelektroniczne

2. MATLAB to znane i powszechnie używane narzędzie do obliczeń numerycznych, analizy danych i modelowania systemów. Posiada szeroki zakres narzędzi i funkcji przeznaczonych do tworzenia modeli matematycznych, symulacji dynamicznych itp

Jednak zdecydowano się na stworzenie własnego modułu do symulacji warstwy fizycznej sieci Ethernet z wykorzystaniem wyżej wymienionych bibliotek.

2 Kodowanie korekcyjne Reeda-Solomona

2.1 Wstęp

Kodowanie korekcyjne Reeda-Solomona zostało stworzone przez Irvina S. Reeda oraz Gustava Solomona w 1960 roku [1,]

Kody Reeda-Solomona charakteryzują się 3 parametrami, rozmiarem alfabetu q interpretowanym w ciele skończonym $\mathbb{F}_q$, długością wiadomości do zakodowania k oraz długością słowa kodowego n gdzie $k < n \leq q$ oraz $q = p^n$ gdzie p to liczba pierwsza a $n \in \mathbb{N}^+$

2.2 Wykorzystanie w standardach Ethernetowych

Różne kody Reeda-Solomona są wykorzystywane w wielu standardach Ethernet, wyróżnione w tablicy 1

Tablica 1: Kodowania RS w różnych standardach [2]

Kodowanie RS	Standardy
RS(528,514)	10GBASE-R, 25GBASE-R, 100GBASE-CR4, 100GBASE-KR4, 100GBASE-SR4
RS(544,514)	50GBASE-R, 100GBASE-KP4, 100GBASE-CR2, 100GBASE-SR2, 100GBASE-DR, 100GBASE-FR1, 100GBASE-LR1, 200GBASE-R, 400GBASE-R
RS(450,406)	1000BASE-T1
RS(192,186)	25GBASE-T, 40GBASE-T
RS(360,326)	2.5GBASE-T1, 5GBASE-T1, 10GBASE-T1

2.3 Tworzenie kodu

Istnieje wiele różnych sposobów tworzenia kodu które tworzą kod o innych właściwościach.

2.3.1 Oryginalny sposób

Sposób kodowania przedstawiony w pracy Reeda i Solomona polega na stworzeniu wielomianu $p_m(x) = \sum_{i=0}^{k-1} m_i x^i$, gdzie $m_i \in \mathbb{F}_q$ to i-ty element wiadomości, po czym za pomocą tego wielomianu obliczane jest słowo kodowe $C(m) = (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1}))$ gdzie a_i to różne elementy ciała $\mathbb{F}_q$.

2.3.2 Kod systematyczny

Za pomocą niewielkiej modyfikacji można stworzyć kod systematyczny czyli taki w którym słowo kodowe zawiera w sobie kodowaną wiadomość. Żeby stworzyć kod systematyczny musimy zmodyfikować sposób tworzenia wielomianu w taki sposób by $p_m(x_i) = m_i$ dla $i \in \{0, 1, \ldots, k-1\}$.

Jednym ze sposobów stworzenia takiego wielomianu jest użycie metody interpolacji wielomianów. Słowo kodowe wygenerowane z tego wielomianu będzie zawierało wiadomość w pierwszych k elementach.

$$C(m) = (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1})) = (m_0, m_1, \ldots, m_{k-1}, p_m(a_k), p_m(a_{k+1}), \ldots, p_m(a_{n-1}))$$

2.4 Dekodowanie

2.4.1 Algorytm Berlekampa-Welcha

W roku 1986 Lloyd R. Welch oraz Elwyn R. Berlekamp uzyskali patent na dekoder umożliwiający uzyskanie oryginalnego wielomianu $p_m(x)$ oraz wielomianu $E(x)$ który zwraca 0 dla punktów x w których nastąpiło przekłamanie [3]

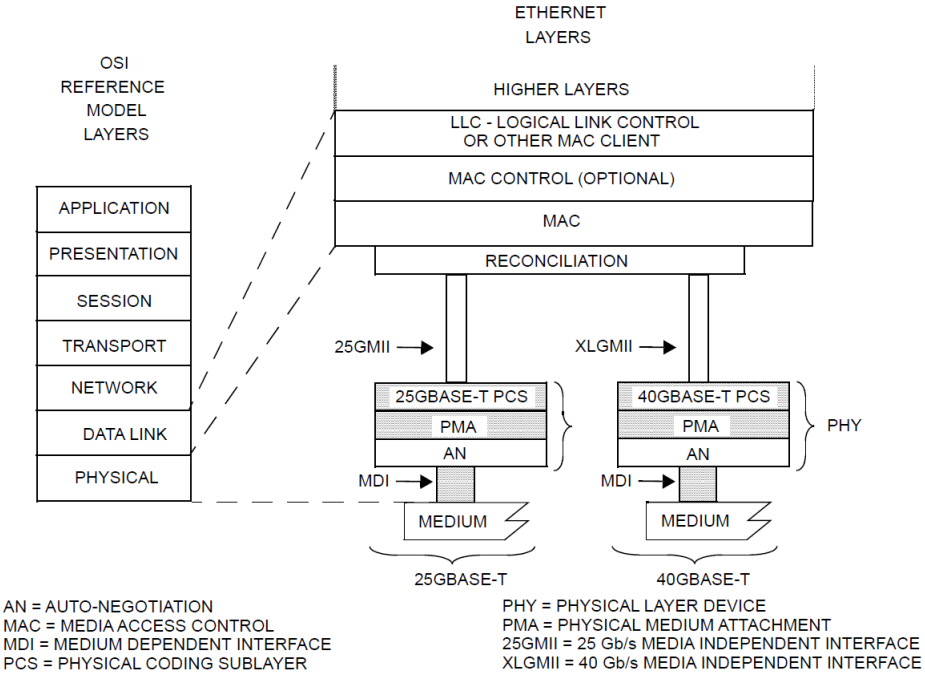
3 40GBASE-T

3.1 Wprowadzenie

40GBASE-T jest technologią transmitowania ramek Ethernetowych z prędkością 40 gigabitów na sekundę, wykorzystując skrętkę jako medium. Technologia została zdefiniowana po raz pierwszy jako część standardu IEEE 802.3ba w 2010 roku.

40GBASE-T zapewnia dwukierunkową komunikację przez cztery przewody skrętki. Każdy przewód transmituje $\frac{1}{4}$ zgromadzonych danych.

3.2 Położenie 40GBASE-T w modelu OSI



3.3 Modulacja w 40GBASE-T

3.3.1 Wprowadzenie

Technologia 40GBASE-T wykorzystuje 16-poziomową modulację PAM - mając 16 różnych poziomów amplitudy możemy zakodować 16 różnych wartości co daje nam 4 bity danych. Oprócz bitów niosących dane, przesyła się również bity pomocnicze (auxiliary bits), przez co w rzeczywistości jeden symbol PAM16 koduje 3,125 bitów informacji. W ciągu każdej sekundy przesyłanych jest 3200 milionów symboli co przekłada się na prędkość transmisji równą 10 Gb/s (3,125 bit/symbol * 3200 MBd) na każdej z czterech par skrętki, co sumarycznie daje transmisję 40 Gb/s.

3.3.2 Warstwa PCS

40GBASE-T PCS (Physical Coding Sublayer) jest warstwą odpowiedzialną m.in. za kodowanie i dekodowanie oraz skramblowanie i deskramblowanie.

PCS pobiera dane od warstwy MAC przez interfejs XLGMII. Jest to tzw. Media-Independent Interface, który powstał po to aby warstwy MAC oraz PHY były niezależne od siebie - pozwala pracę kontrolera MAC z warstwą PHY, bez względu na to jakie medium jest w użyciu.

Dane trafiają z XLGMII do PCS przez wektor TXD<63:0> i gromadzone są w 64-bitowe bloki. Po zebraniu 50 takich bloków, pierwsze 48 z nich transkodowane są w 512-bitowe bloki, a pozostałe są do nich dołączane. Dane są następnie skramblowane i dołączany jest do nich bit pomocniczy (auxiliary bit), po czym dzielone są na dwa zbiory — pierwszy z nich trafia do kodera Reed-Solomona, a drugi jest przetwarzany przez koder LDPC (Low density parity check). Otrzymuje się w ten sposób $512 * 3$ bitów zakodowanych przez RS-FEC oraz $512 * 4$ bitów — LDPC, które łączone są w 7-bitowe grupy u_0, u_1, u_2, c_0, c_1, c_2, c_3.

3.3.3 Mapowanie ramki LDPC na DSQ128

Zamianę 7-bitów u_0, u_1, u_2, c_0, c_1, c_2, c_3 pokazuje poniższy algorytm:

Krok 1:

$$x_{13} = \neg u_0 * u_2$$
$$x_{12} = u_0 \oplus u_2$$
$$x_{11} = c_0$$
$$x_{10} = c_0 \oplus c_1$$
$$x_{23} = (u_1 * u_2) + (u_0 * \neg u_1)$$
$$x_{22} = u_1 \oplus u_2$$
$$x_{21} = c_2$$
$$x_{20} = c_2 \oplus c_3$$

Krok 2:

$$x_1 = 8x_{13} + 4x_{12} + 2x_{11} + x_{10}$$
$$x_2 = 8x_{23} + 4x_{22} + 2x_{21} + x_{20}$$

Krok 3:

$$y_1 = (x_1 + x_2) \mod 16$$
$$y_2 = (-x_1 + x_2) \mod 16$$

Krok 4:

$$\mathrm{PAM16}_1 = 2y_1 - 15$$

$$\mathrm{PAM16}_2 = 2y_2 - 15$$

Otrzymane w ten sposób symbole są transmitowane na odpowiednich parach skrętki:

Pair A	$\mathrm{PAM16}_1$<0>	$\mathrm{PAM16}_2$<0>	$\mathrm{PAM16}_1$<4>	$\mathrm{PAM16}_2$<4>	...	$\mathrm{PAM16}_1$<508>	$\mathrm{PAM16}_2$<508>
Pair B	$\mathrm{PAM16}_1$<1>	$\mathrm{PAM16}_2$<1>	$\mathrm{PAM16}_1$<5>	$\mathrm{PAM16}_2$<5>	...	$\mathrm{PAM16}_1$<509>	$\mathrm{PAM16}_2$<509>
Pair C	$\mathrm{PAM16}_1$<2>	$\mathrm{PAM16}_2$<2>	$\mathrm{PAM16}_1$<6>	$\mathrm{PAM16}_2$<6>	...	$\mathrm{PAM16}_1$<510>	$\mathrm{PAM16}_2$<510>
Pair D	$\mathrm{PAM16}_1$<3>	$\mathrm{PAM16}_2$<3>	$\mathrm{PAM16}_1$<7>	$\mathrm{PAM16}_2$<7>	...	$\mathrm{PAM16}_1$<511>	$\mathrm{PAM16}_2$<511>

Literatura

[1] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[2] Ieee standard for ethernet. *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pages 1–7025, 2022.

[3] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, U.S. Patent 4 633 470, Dec. 30, 1986.