POLITECHNIKA ŚLĄSKA W GLIWICACH

WYDZIAŁ ELEKTRYCZNY

Katedra Energoelektroniki, Napędu Elektrycznego i Robotyki

Kierunek Elektrotechnika

PRACA DYPLOMOWA

MAGISTERSKA

Rejestracja i dekodowanie danych produkcyjnych, przesyłanych w przemysłowej sieci Ethernet

Promotor:

Dr inż. Zbigniew Mantorski

Wykonał:

Marcin Szendzielorz

Opieka techniczna w firmie INAT GmbH:

Dipl. Ing. Werner Krings

Spis treści

1. Wprowadzenie

Obecny, szybki rozwój techniki oraz badania i prace prowadzone przy wdrażaniu nowych technologii we wszystkich gałęziach przemysłu prowadzą do zwiększenia wydajności produkcji i poprawy jakości produktów. Taki postęp technologiczny i innowacyjna myśl techniczna pozwalają na budowanie urządzeń tańszych, mniejszych, niezawodnych a przy tym bardzo dokładnych i często bezobsługowych.

Konstruktorzy, projektujący fabryki, linie produkcyjne oraz zautomatyzowane maszyny, oprócz dobrej jakości mają za zadanie zapewnić jak największą sprawność produkcyjną przy jak najniższych kosztach. Taka koncepcja prowadzi do możliwie maksymalnego zautomatyzowania zarówno procesów produkcyjnych, jak i zastąpienia ludzi w obiektach, których praca może być monitorowana i wykonywana zdalnie. Wszędzie tam umieszcza się różnego rodzaju sterowniki kierujące i kontrolujące procesy oraz przesyłające dane do innych sterowników i do stacji nadzorczych, odpowiedzialnych za zapisywanie danych i monitoring.

Na równi z tak dynamicznym rozwojem techniki muszą „iść" ludzie, specjaliści, którzy będą w stanie odpowiednio dane urządzenia zaprogramowywać i łączyć ze sobą w mniejsze lub większe grupy. Konkurencja wśród producentów różnego rodzaju urządzeń prowadzi do obniżenia ich cen i polepszenia jakości. Niestety towarzyszy temu zjawisku, mimo przyjętych europejskich i światowych standardów, według których producenci powinni tworzyć swoje urządzenia, niekompatybilność, która czasami prowadzi do kosztownych awarii lub długotrwałych przestojów linii produkcyjnych. Zarówno duże, jak i te mniejsze zakłady przemysłowe muszą dbać o poprawną wymianę sygnałów sterujących procesami i eliminować ewentualne przyczyny zakłóceń.

1.1. Cel projektu

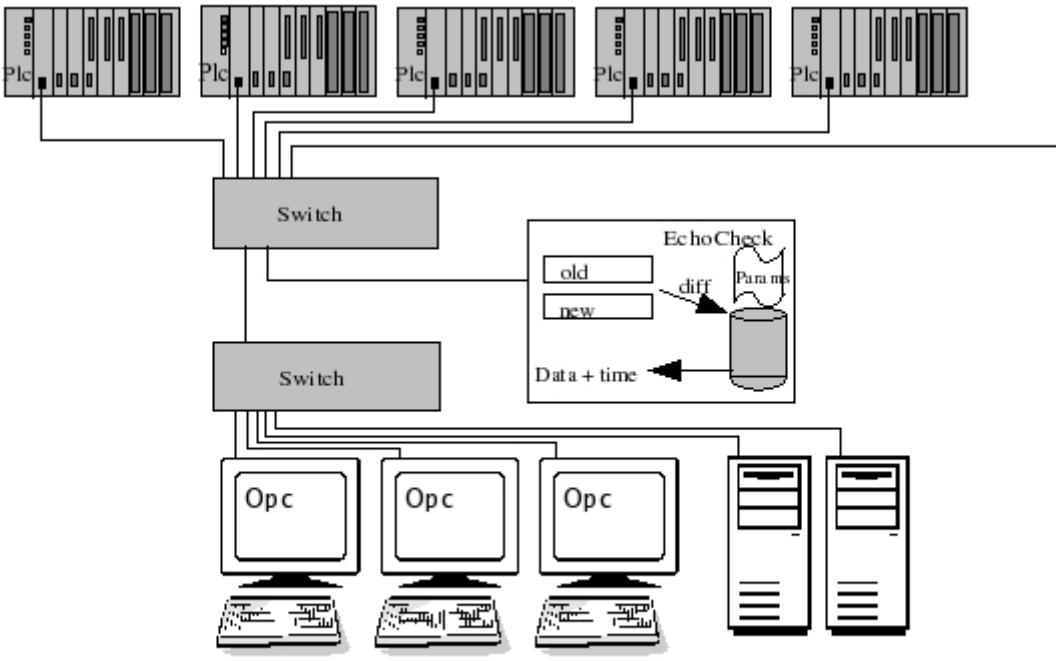
W sieciach przemysłowych od kilku lat coraz częściej i chętniej stosuje się komunikację sieciową opartą o standard, wykorzystywany w budowie lokalnych sieci komputerowych – Ethernet. Duża popularność i wytestowana przez miliony ludzi niezawodność otwarła temu systemowi komunikacji „wrota" do przemysłu. Jednakże bazujące na tym standardzie inne warstwy stosowane również w aplikacjach przemysłowych zawierają czasami błędy. Wadliwa praca danego urządzenia lub całej linii produkcyjnej, podłączonej do sieci bazującej na standardzie Ethernet, często nie jest spowodowana błędami samej sieci lecz nieodpowiednimi telegramami, wysyłanymi przez inne stacje w jej obrębie. W przemyśle bardzo ważna jest niezawodność urządzeń i ciągłość produkcji, toteż operatorzy tak odpowiedzialnych urządzeń muszą posiadać stosowną wiedzę, znać dobrze zadania, jakie ma dane urządzenie wykonywać. Niestety nierzadko zdarza się, że programiści nieumyślnie błędnie przeprogramują dane urządzenie lub co gorsza wyniki tego błędu ujawniają się nie od razu lecz po czasie, co również prowadzi do przestoju linii produkcyjnej lub awarii.

Opisane powyżej przykłady nieprawidłowości zachodzących w przemyśle oraz częste zapytania ze strony osób nadzorujących produkcję odnośnie przeciwdziałania i diagnozie takich zdarzeń, jak również brak na rynku analizatorów sieciowych, pracujących jako niewielkie, kompaktowe urządzenie, analizujące sieć w różnych częściach linii produkcyjnej lub różnych częściach fabryki jednocześnie, przyczyniły się do rozpoczęcia projektu „EchoCheck", realizowanego w firmie INAT GmbH w Norymbergii. Stworzenie bazowej części tego projektu, obejmującej tematykę sieci przemysłowych zbudowanych na standardzie Ethernet, wykorzystującej protokoły TCP/IP, jak również specjalne protokoły sterowników programowalnych stanowi **cel** mojej **pracy dyplomowej**.

W ramach mojej **pracy dyplomowej** wykonany został program, potrafiący analizować telegramy przesyłane poprzez sieć Ethernet, tym samym filtrować i rozkodowywać tylko te telegramy, które trafiają do danego sterownika i zmieniają jego ustawienia lub uniemożliwiają jego poprawne funkcjonowanie. Te bardzo ważne informacje zostają zapisane ze stemplem czasowym.

1.2. Sposób rozwiązania

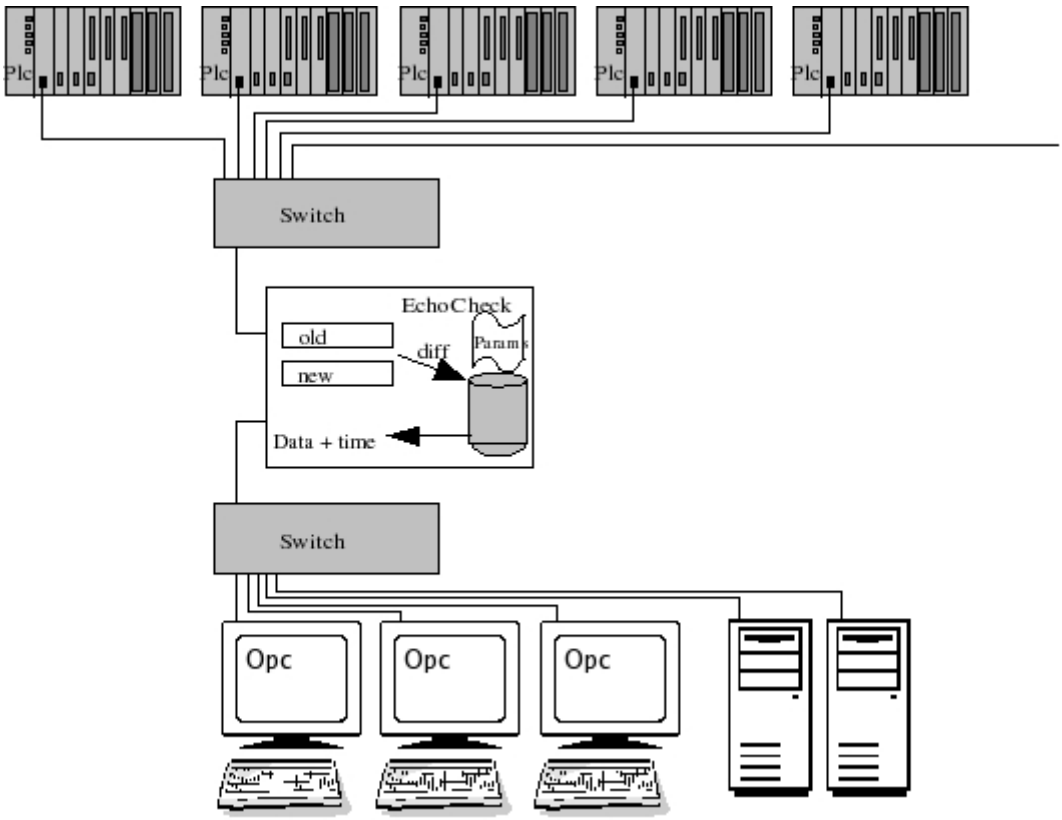
Na etapie planowania i projektowania urządzenia, powstały dwa modele przedstawiające sposób wpięcia urządzenia monitorującego do sieci Ethernet.



Rysunek 1. „EchoCheck" podpięty do portu monitorującego switch-a.

Powyższe rozwiązanie (Rysunek 1) przedstawia metodę monitorowania komunikacji pomiędzy dwoma switch-ami, możliwą tylko w przypadku zastosowania switch-a wyposażonego w tzw. port lustrzany (mirror port). Jest to przykład, gdzie sterowniki programowalne (PLC) są połączone ze stacjami nadzorującymi, serwerami lub panelami operatorskimi. Aby monitorowanie danych było sensowne i w pełni zrealizowane należy urządzenie monitorujące podłączyć do jedynej drogi prowadzącej od sterowników do reszty urządzeń. W przeciwnym razie część interesujących nas danych może zostać przesłana inną drogą do sterownika. Port lustrzany umożliwia „podglądanie " danych a następnie ich przetwarzanie w „EchoCheck-u". W zależności od ustawienia parametrów następuje porównanie przesyłanych danych z wcześniej otrzymanymi, co w razie różnicy – pojawienia się odpowiednich, nieoczekiwanych zmian – powoduje automatyczny zapis wybranych danych ze stemplem czasowym.

Innym rozwiązaniem, spełniającym nasze założenia jest konfiguracja urządzenia „EchoCheck” jako switch-a (Rysunek 2).



Rysunek 2. „EchoCheck” skonfigurowany i podpięty do sieci jako switch.

Switch, jak nazwa wskazuje przełącza w systemie 1:1, co oznacza, że wszystkie informacje dostarczone do portu źródłowego są chwilowo zapamiętywane i przesyłane do odpowiedniego portu docelowego. Wykorzystując takie funkcjonowanie switch-y jesteśmy w stanie również przejąć te chwilowo zapamiętane dane i je przetworzyć. Takie rozwiązanie pozwala na zmniejszenie kosztów związanych z podłączeniem analizatora do sieci w miejscach nie wyposażonych w specjalne switch-e (posiadające „mirror port”).

1.3. Zakres pracy dyplomowej

Realizacja pracy dyplomowej składała się z trzech etapów. Pierwszy etap obejmował zapoznanie się z podstawami teoretycznymi oraz opracowanie sposobów i metod realizacji poszczególnych kroków przy analizie komunikacji w Ethernet-owych sieciach przemysłowych. Druga część pracy to odpowiednie zaimplementowanie wcześniej przygotowanych podstaw teoretycznych dotyczących analizy i dekodowania telegramów Ethernet-owych w programie „Microsoft Visual C++ 6.0", w trybie konsolowym. W trzeciej, ostatniej fazie pracy dyplomowej, przeprowadzone zostały wnikliwe testy komunikacji na poziomie protokołów (PLC), również z uwzględnieniem skrajnych przypadków. Komunikacja testowa przeprowadzona została przy pomocy programu „NetTstnt" (program stworzony przez firmę INAT GmbH, do testowania komunikacji. Jest to darmowy, dostępny na stronie producenta program, do którego jako stacja odbiorcza została w czasie testów wykorzystana karta S5-TCP/IP - CP (Communication Processor). Karta ta, będąca również produktem firmy INAT GmbH, przewidziana jest do zastosowania jej w sterownikach z rodziny S5, jednak wyposażona została ona również możliwość komunikacji w sieci Ethernet-owej poprzez protokoły RFC1006, INAT PLC Header, S7 oraz ModbusTCP. Poniżej przedstawione są poszczególne kroki realizacji wszystkich etapów pracy dyplomowej.

Część teoretyczna:

- ✓ Ogólne zaznajomienie się z sieciami przemysłowymi.
- ✓ Zapoznanie się z sieciami Ethernet-owymi.
- ✓ Zaznajomienie się z protokołami TCP/IP i OSI.
- ✓ Rozpoznawanie granic sieciowych, w szczególności przy analizie komunikacji Ethernet-owej w sieciach przemysłowych.
- ✓ Zapoznanie się z protokołami komunikacyjnymi sterowników programowalnych, ze względu na wymagania dotyczące komunikacji w sieciach ze sterownikami PLC.
- ✓ Ustalenie logicznego przebiegu komunikacji pomiędzy stacjami Ethernet-owym.
- ✓ Zdefiniowanie błędów komunikacyjnych i ich korekcja.
- ✓ Zaznajomienie się z protokołami sterowników programowalnych - PLC.

Kroki przy tworzeniu programu w „C”:

- ✓ Zapis telegramów pochodzących z karty sieciowej.
- ✓ Rozpoznanie połączeń.
- ✓ Analiza strumienia telegramów połączona z obróbką błędów.
- ✓ Gromadzenie danych przeznaczonych do dekodowania.
- ✓ Dekodowanie protokołów związanych ze sterownikami programowalnymi.
- ✓ Rozpoznanie telegramów „o charakterze zapisu” – Write.
- ✓ Zapisanie wyników w postaci listy.

Testowanie programu:

- ✓ Komunikacja „Fetch oraz Write” poprzez protokół S7 osadzony na RFC 1006.
- ✓ Komunikacja „Fetch oraz Write” poprzez protokół S5 AP osadzony na RFC 1006.
- ✓ Komunikacja „Fetch oraz Write” poprzez protokół S7 osadzony na RFC 1006.
- ✓ Komunikacja "Fetch oraz Write" poprzez protokół ModbusTCP.
- ✓ Komunikacja „Fetch oraz Write” poprzez protokół S7 osadzony bezpośrednio na TCP.
- ✓ Komunikacja „Fetch oraz Write” poprzez protokół S5 osadzony na PLC Header.
- ✓ Komunikacja „Write” poprzez protokół S7 z niepełnym nagłówkiem – „zbieranie”. kolejnych telegramów i ich połączenie w celu możliwości dokonania dekodowania osadzony na PLC Header.

2. Sieci przemysłowe

2.1. Wprowadzenie [1]

Sieci przemysłowe, tworząc podstawę zdecentralizowanej automatyki przemysłowej muszą spełniać szczególne wymagania tej właśnie branży. Oprócz warunków instalacyjnych i technicznych muszą one zapewniać odpowiednie parametry przesyłu, jak np. czasy opóźnienia raportów czy przepustowość łącza. Definicja sieci, określająca wielkości danych, implementację i zachowanie parametrów przesyłu jest poprzez „jakość usługi" (QoS – Quality of Service) spełniona. Nazwa „sieć" w przeciągu ostatnich dziesiątek lat zmieniała swoje znaczenie. I tak dziś, pod pojęciem „sieć" rozumiany jest dowolny system komunikacyjny, który umożliwia wymianę informacji pomiędzy większą ilością uczestników.

Konieczne, a zarazem charakterystyczne cechy sieci to:

- ✓ Każdy uczestnik (stacja sieciowa) posiada fizyczne połączenie z siecią – NIC (Network Interface Adapter).
- ✓ Każdy członek posiada swój własny i niepowtarzalny adres sieciowy, dzięki któremu może się bez pomyłkowo komunikować z resztą uczestników .
- ✓ Sieć zawiera tabelę adresów, dzięki której mogą być odnalezieni wszyscy jej członkowie.
- ✓ Wszystkie transfery danych w sieci odbywają się według ściśle określonych reguł, a odpowiednikiem tych zasad jest „protokół" (protokół sieciowy), do którego wszyscy uczestnicy sieci muszą się dostosować . Protokół zapewnia nie tylko standardową wymianę danych, ale również informuje o błędach i umożliwia konfigurację sieci.

Najbardziej rozpowszechnione sieci można sklasyfikować ze względu na poniższe właściwości :

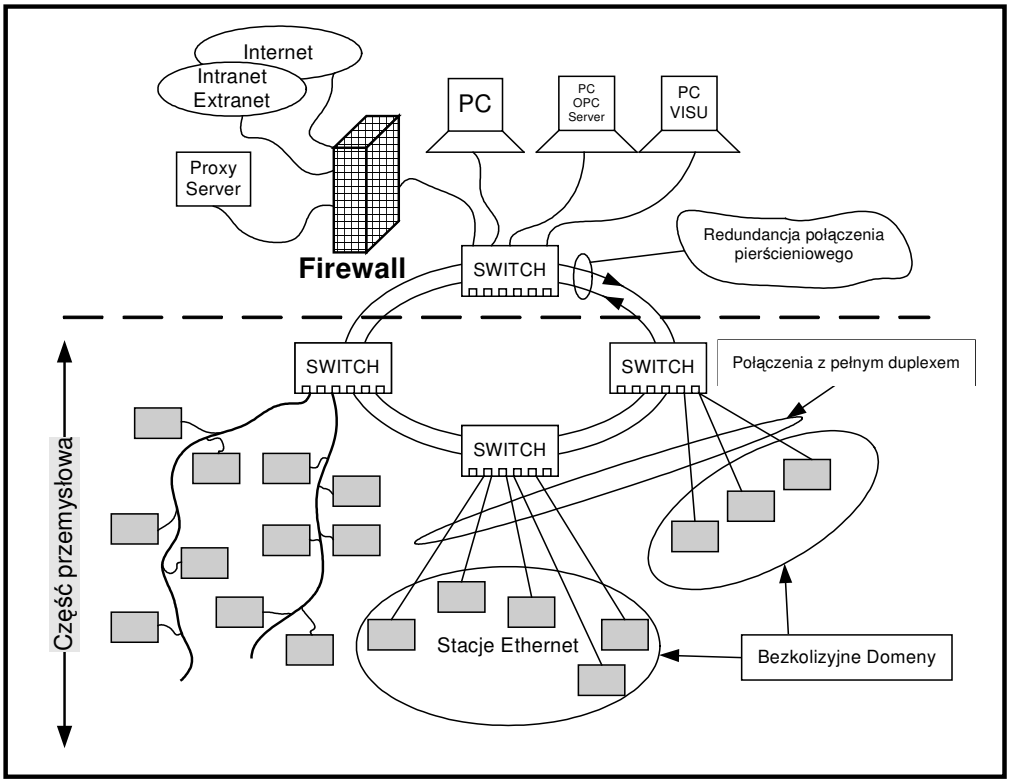
1) Rozległość
   - LAN (Local Area Network) – kilka km
   - MAN (Metropolitan Area Network) – kilkadziesiąt km
   - WAN (Wide Area Network) –kilkaset km

2) Sposób pracy
   - Single-Master/Multi-Slave
   - Multi-Master/Multi-Slave
   - Sieci symetryczne – bez centralnego „master-a”
3) Standard, na którym bazują
   - Ethernet
   - ATM (Asynchronous Transfer Mode)
   - Start-Stop
4) Protokół sieciowy
   - Ethernet- TCP/IP
   - Profibus
   - Bitbus
5) Ilość uczestników
   - 128
   - 1024
   - 2^{24}
   - 23^{32}
   - 2^{128}
6) Medium przenoszonego sygnału
   - Miedź
   - Włókna optyczne (szkło, tworzywa)
   - Radio – WLAN (Wireless Local Area Network)
   - Podczerwień
   - Mikrofale
   - GSM (Global System for Mobile Communications)
7) Rodzaj programowania API (Application Programming Interface)
   - Socket- Interface
   - Odwzorowanie procesu
   - Funkcje S7
   - MiTS
8) Topologię
   - Bus
   - Gwiazda

- Drzewo
- Pierścień

2.2. Budowa i elementy [1]

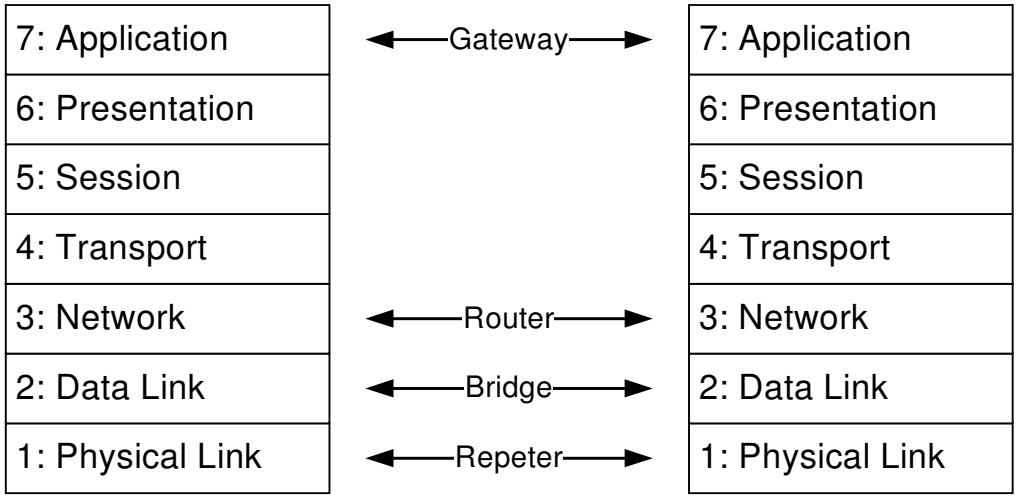
Poprzez czujniki- sensory (np.: krańcówki, obrotomierze), człony wykonawcze - aktory (np. silniki), sterowniki (PLC), hub-y i switch-e ważne dla produkcji dane trafiają do komputerów operatorskich, paneli sterowniczych, systemów wizualizacji oraz niekiedy, poprzez firewall, są dostępne w Intranecie lub Internecie. Sieci przemysłowe, w zależności od gałęzi przemysłu oraz konkretnej aplikacji, są mniej lub bardziej rozbudowane i składają się z różnych typów urządzeń pośredniczących w transferze, odbiorze i przetwarzaniu danych.



Rysunek 3. Architektura sieci na przykładzie automatyki przemysłowej.

Aby zapewnić odpowiednio szybką komunikację ze sterownikami połączonymi z siecią Ethernet projektować można wolne od kolizji domeny (Rysunek 3). Powyższy rysunek ukazuje rozgraniczenie pomiędzy częścią sieci, umieszczoną bezpośrednio w przemyśle, a stanowiskami operatorskimi, bazami danych i zdalną kontrolą. Aby zapewnić dużą pewność przy przesyle danych stosuje się redundancję, polegającą na określonym rozmieszczeniu switch-y, wyposażonych w funkcję „Spinning Tree", i ich odpowiednim okablowaniu. Redundancja ma na celu zminimalizowanie ryzyka np. poprzez stworzenie podwójnych, niezależnych dróg wymiany danych w przypadku kiedy dojdzie do nieoczekiwanej usterki. Każda redundancja posiada tzw. SPOF (Single Point of Failure - pojedynczy punkt powstania awarii), gdyż nie jest możliwe stworzenie stuprocentowej redundancji, a dążąc ku temu koszty wzrastają niewspółmiernie do zapewnionego bezpieczeństwa. Użytkownicy zewnętrzni, którzy mają ograniczony dostęp do sieci przemysłowej, sięgają po interesujące ich informacje do przygotowanego servera proxy. Bezpośredni dostęp do poszczególnych węzłów sieciowych w sieciach łączących komponenty automatyki jest umożliwiony tylko nielicznym użytkownikom (zdalne programowanie, zdalna obsługa), którzy muszą się w bezpieczny sposób logować.

Repeater, hub, bridge, router oraz gateway, są ważnymi elementami komunikacji TCP/IP zbudowanej na standardzie Ethernet. Elementy te umożliwiają większą swobodę w instalacji, możliwość połączenia różnych sieci i komunikację pomiędzy użytkownikami w sieciach o różnych protokołach. Różnica pomiędzy tymi czteroma elementami jest dobrze zobrazowana dzięki ich funkcjonalności w modelu OSI (Rysunek 4). Adekwatnie do poziomu przynależności brana jest pod uwagę mniejsza lub większa ilość danych zawartych w nagłówkach IP lub TCP potrzebna do spełnienia swoich funkcji. Szczególnie router-y i bramy (gateway-e) są skomplikowanymi urządzeniami z dużymi funkcjami obliczeniowymi .



Rysunek 4. Repeater, bridge, router i gateway w OSI modelu.

Repeater

Repeater (wzmacniacz) nie uwzględnia żadnych informacji zawartych w nagłówkach. Pełni on rolę elektrycznego lub optycznego zamiennika sygnału w systemie przesyłania danych. Umożliwia także tworzenie topologicznie skomplikowanych sieci (o strukturze drzewa), sieci rozległych (rola wzmacniacza) lub zmianę medium komunikacyjnego (np. 10base5 na 10baseT lub światłowody).

Bridge

Bridge (most) łączy dwie lub więcej (Multiport Bridge) sieci i wymienia między nimi pakiety danych. Mosty są elementami inteligentnymi, które „przyglądają się" całemu transferowi danych w sieci, do której są przyłączone. Pakiety są analizowane na podstawie ich fizycznych adresów oraz typów, a następnie przy pomocy wewnętrznych tabel decyduje się czy pakiet zostanie bez zmian przekazany dalej, czy zostaną dokonane odpowiednie zmiany a następnie przekazany dalej lub czy zostanie zignorowany ([1] - więcej informacji).

Router

Ogólna sytuacja w Internecie oraz Ethernecie obejmuje współpracę dużej liczby różnych systemów komunikacyjnych. IP – Datagram, który zostanie wysłany przez jednego użytkownika, może przechodzić przez 5, 10 lub 20 niezależnych sieci. W IP – Datagram-ie zamieszczony jest jedynie adres docelowy, brak jest jakichkolwiek informacji na temat drogi jaką ma być przesłany ani sposobu w jaki sieci są między sobą połączone. W większości przypadków nie znany jest nadawca IP – Datagram-u oraz sieć w jakiej znajduje się stacja docelowa. Dlatego w każdym punkcie wyjściowym różnych sieci komunikacyjnych musi

znajdować się komputer pośredniczący, który rozstrzyga gdzie IP – Datagram ma być przekazany. Taki komputer pośredniczący w sieciach TCP/IP nosi nazwę „router".

Routery są bardzo ważnymi elementami w sieci Internet, ponieważ tylko one umożliwiają odnajdywanie drogi w połączonych ze sobą sieciach.

Gateway

Bridges (mosty) są w stanie łączyć sieci o różnych protokołach w warstwie drugiej (np.: Ethernet z pierścieniem - Token). Router może pracować (łączyć) w sieciach, w których protokoły trzeciej warstwy są takie same. W technice są jednak przypadki, kiedy komputery o niekompatybilnych protokołach powinny wymieniać dane. Spotyka się przypadki, kiedy maszyny posiadające specyficzną dla danego producenta komunikację, mają być podłączone do sieci TCP/IP. Rozwiązanie takiego problemu następuje poprzez zastosowanie gateway-ów (bram). Bramy te transformują odpowiednie funkcje komunikacyjne (po części z dużym nakładem obliczeniowym), zapewniając jednocześnie kompatybilność sieciową.

Switch

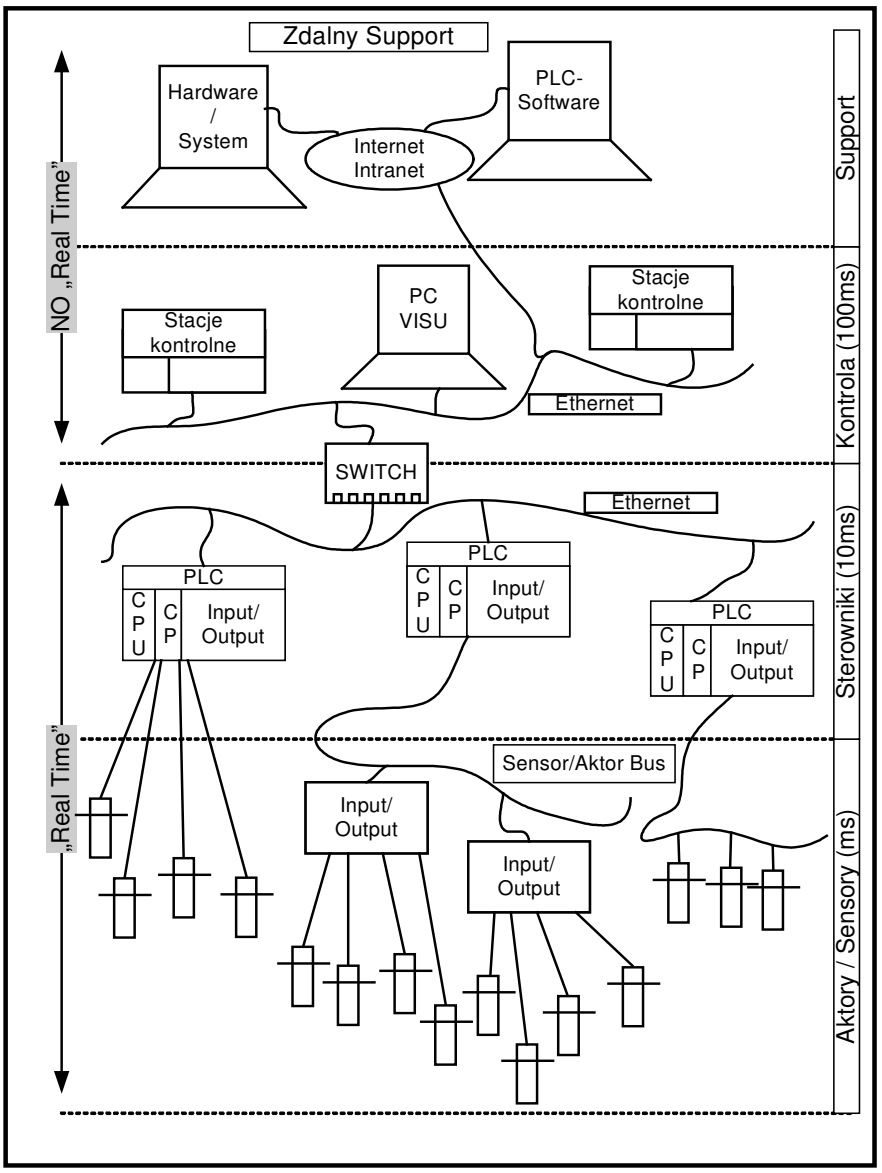
Nowoczesne elementy służące do tworzenia struktur sieciowych to switch-e. Są one węzłami pośredniczącymi miedzy wieloma portami. Telegramy otrzymane do danego portu zostają przekierowane (przełączone) wyłącznie do portu wyjściowego, który „zna" stację odbiorczą. Nowoczesne switch-e pracują obecnie z dużymi prędkościami (gigabit) i stosunkowo małymi opóźnieniami (kilka bajtów).

2.3. Ethernet [5]

Ethernet w czasie ostatnich lat stał się podstawową technologią dla sieci LAN. Specyfikacja opisująca standard Ethernet zamieszczona jest w dokumencie RFC 826. Poprzez ciągły rozwój pierwotnego standardu 10Mbit/s, Ethernet znalazł zastosowanie również w technologii Token Ring (IEEE 802.5; jednak w automatyce przemysłowej nie został szeroko rozpowszechniony), Wi-Fi (IEEE 802.11), FDDI, ATM i inne. Technologia Ethernet stała się standardem międzynarodowym i została zatwierdzona przez konsorcjum IEEE pod numerem 802.3.

Ogólnie, w myśl norm serii standardów IEEE 802 węzły w sieci lokalnej, obsługiwanej np. przez protokół normy IEEE 802.3 (Ethernet), dzielą się na dwie główne klasy:

- ✓ **DTE** – Data Terminal Equipment – urządzenia będące źródłami i odbiornikami ramek. Do tej klasy należą komputery osobiste, stacje robocze, serwery plików, itp.
- ✓ **DCE** – Date Communication Equipment – urządzenia sieciowe, które pośredniczą w przesyłaniu ramek przez sieć (tzn. odbierają i przesyłają ramkę dalej). Do tej klasy należą urządzenia typu router, switch, repeater, ale także interfejsy komunikacyjne, takie jak interfejsy kart czy modemów.



Rysunek 5. Znaczenie czasu, a architektura sieci przemysłowych.

Ramy czasowe oraz mieszczące się w nich grupy urządzeń składowych sieci, tworzą swojego rodzaju oś czasową (Rysunek 5). Owa oś zaczyna się od elementów wykonawczych, sensorów, gdzie wymaga się jak największej szybkości przesłania i zareagowania na sygnał, lub zdarzenie, poprzez poziom sterowników programowalnych i systemy nadzoru, a zakończy się na zdalnym suporcie. W zależności od wymagań czasowych na poziomie

aktorów i sensorów możemy mieć do czynienia z Buss-em systemowym Ethernet (wolniejsze czasy reakcji) lub Buss-em CAN (Controller Area Network – szybsze czasy reakcji). W warstwie sterowników, gdzie czasy przesyłanych informacji mogą kształtować się na poziomie dziesiątek milisekund, system Ethernet-owy bezproblemowo łączy sterowniki między sobą oraz przekazuje informacje do odpowiednio oddzielonej części produkcyjnej sieci, warstwy kontroli. Warstwa kontroli jest również połączona między sobą przy pomocy Ethernet-u i daje możliwość, dzięki zastosowaniu odpowiednich bram, już za pomocą innych protokołów sieciowych (np. Telnet) kontakt z zewnątrz (np. suport). Pierwsze dwie warstwy zostały przypisane do tzw. „Real Time" - czasu rzeczywistego, którego to zdefiniowanie jest bardzo trudne, gdyż dla różnych wymagań procesowych istnieją inne ramy czasowe. W jednym procesie czas krytyczny wynosi setne, dziesiętne lub nawet pełne sekundy, podczas gdy w innych o rząd lub dwa rzędy mniej. Na poziomie kontroli czas też jest ważny, ale nie jest już taki krytyczny jak w dwóch niższych warstwach, w których to zaprogramowane są bezpośrednie kroki procesów przemysłowych.

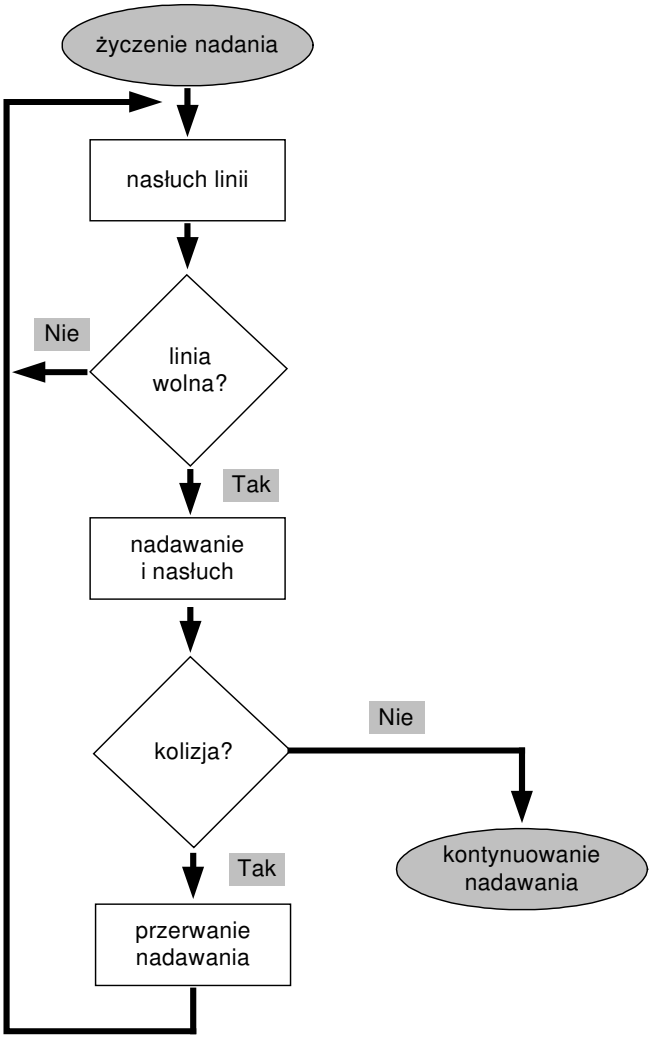
Przy licznych wariantach Ethernet-u z różnymi rodzajami medium transportowego i różnymi sposobami przesyłu dane trafiają i tak, w ten sam sposób do sieci. Standaryzacja przesyłu danych w sieci związana jest z budową telegramów Ethernet-owych (Ethernet-Frames) i protokołem wielodostępu do łącza CSMA/CD.

2.3.1 Metoda dostępu do sieci CSMA/CD

Ethernet posiada równouprawnione stacje sieciowe, tzn. każda stacja może w dowolnej chwili komunikować się z inną stacją, może wysłać informację do pewnej grupy stacji lub do wszystkich stacji w sieci. Prawo dostępu do wspólnego medium komunikacyjnego opiera się na systemie rozpoznawania kolizji w sieci – system CSMA/CD (Carrier Sense with Multiple Access and Collision Detection).

2.3.1.1 Carrier Sense (Podsłuchiwanie nośnika)

Stacja, która chce wysłać dane, sprawdza czy linia jest wolna czy też inna stacja nadaje. Jeśli linia jest wolna stacja może od razu zacząć przesyłać dane. Natomiast, gdy linia jest zajęta, dopiero po pewnym czasie następuje ponowna próba wysłania danych (Rysunek 6).



Rysunek 6. Idea kontroli dostępu do sieci CSMA/CD

2.3.1.2 Multiple Access (Wielodostęp)

Mimo powyższego zabezpieczenia może dojść do jednoczesnej próby dostępu do sieci. W przypadku, gdy dwie stacje rozpoznały wolną linię i niezależnie od siebie zaczną nadawać nastąpi nałożenie się sygnałów i dane zostaną zniszczone. Takie zdarzenie określa się jako kolizję.

2.3.1.3 Collision Detection (Wykrywanie kolizji)

Urządzenie, posiadające dane „w kolejce" do wysłania, będzie monitorowało swoją własną transmisję. Urządzenie, które wykryło kolizję, zatrzymuje wysyłanie danych i wysyła sekwencję informującą o kolizji (sygnał zagłuszania). Poziom sygnału informującego o kolizji (prądu, ponieważ zgodnie z CSMA/CD węzły są nadajnikami prądu o stabilizowanym natężeniu) jest wyższy od normalnie generowanego przez węzeł, aby mieć pewność, że

każdy węzeł odebrał sekwencję informującą o kolizji. Niepoprawny 32-bitowy sygnał MAC CRC spowoduje, że odbierający go węzeł wyśle ponownie dane w odpowiedzi na wystąpienie błędu CRC. Po przypadkowym czasie oczekiwania, każda stacja próbuje ponownie przesłać dane. Kolizje nie powodują utraty danych, gdyż stacje uczestniczące w kolizji wysyłają je ponownie, aż w końcu docierają do stacji docelowych. Nie można jednak określić, kiedy dane naprawdę dotrą do stacji docelowej, co jest dużą wadą sieci Ethernet. Metoda dostępu do sieci CSMA/CD jest aktywnie stosowana wyłącznie przez sieci komunikujące się w systemie half-duplex. W systemach full-duplex karty sieciowe nadal pracują z tą metodą, jednak tu już nie może dojść do kolizji. Możliwe jest to dzięki zastosowaniu kabli twisted-pair, gdzie jedna para żył odpowiedzialna jest za przesył, a druga za odbiór danych oraz switch-y, jako aktywnych łączników.

2.3.2 Ograniczenia długości linii w sieciach Ethernet-owych [3]

Za ograniczenie długości kabla odpowiedzialne są właściwości fizyczne otoczenia i medium nośnego. Główny wpływ na długość nośnika mają elektryczne lub optyczne właściwości tłumienia sygnału. Sygnał musi być poprawnie rozpoznany na końcu danej linii. Bezpośredni wpływ na poprawność sygnału ma jakość kabla oraz częstotliwość przesyłu. Im częstotliwość przesyłu jest większa, tym większe są wymagania jakościowe w stosunku do nośnika. Wszystkie urządzenia towarzyszące przesyłaniu sygnału w zależności od funkcji i konstrukcji wprowadzają od kilku do kilkudziesięciu bitów opóźnienia w przesyle. Ograniczenie długości domeny kolizyjnej jest związane w sieciach Ethernet-owych z metodą dostępu do sieci CSMA/CD. CSMA/CD pracuje również przy przesyle najmniejszych telegramów o rozmiarze 64 bajtów (512 bitów) sprawnie i częściowo ograniczając liczbę kolizji. Ze względu na to, że kolizji przy metodzie CSMA/CD nie da się całkowicie wyeliminować ważne jest, aby były one rozpoznawane przez stacje nadawczą. Jest to możliwe jedynie wtedy, kiedy pierwszy bit telegramu dotrze do najdalej oddalonego odbiorcy zanim ostatni bit telegramu opuści stację nadawczą. Największa odległość pomiędzy dwoma punktami końcowymi może mieć taki wymiar, który odpowiada połowie czasu sygnału przy wysłaniu 512 „bitów czasowych”. Są one uzależnione od szybkości przesyłu, czyli czy Ethernet-LAN pracuje z prędkością 10, 100, czy 1000 bit/s. W sieci o prędkości 10 Mbit/s bit czasu wynosi 100 ns, przy 100 Mbit/s – 10ns, a przy 1000 Mbit/s – 1ns. Tym wartościom odpowiadają czasy trwania: dla najmniejszego telegramu (512 bitów) odpowiednio 51,2 us (10 Mbit/s), 5,12 us (100 Mbit/s) i 512ns (1000 Mbit/s). Te założenia oraz uwzględnienie stałej prędkości rozchodzenia się sygnału w nośniku pozwalają na

oszacowanie maksymalnych odległości domeny kolizyjnej, które wynoszą: 2000 m przy 10 Mbit/s, 200m przy 100 Mbit/s oraz 20m przy 1 Gbit/s. Jako, że 20 m dla prędkości 1 Gbit/s stanowiłoby bardzo duże ograniczenie, zmieniono w tym przypadku najmniejszy dopuszczalny rozmiar telegramu i uzyskuje się też 200 m. Powyżej opisane prędkości w sieciach i odpowiadające im długości, dotyczą jedynie domen kolizyjnych, jak również nie uwzględniają innych fizycznych właściwości nośników.

W innych konfiguracjach sieciowych bazuje się na przytoczonych powyżej informacjach, jednak końcowe obliczenia długości przewodów muszą uwzględniać topologię sieci, jej rodzaj, elementy pośredniczące w przesyle informacji, itp.. Szersze informacje na temat długości sieci zawarte są w literaturze [3].

2.3.3 Model ISO/OSI [N-2]

Międzynarodowa Organizacja Standardów - ISO (International Standards Organization) wprowadziła ujednolicony model komunikacyjny łączenia systemów otwartych – OSI (Open Systems Interconnection). Model referencyjny ISO/OSI jest modelem odniesienia dla wszystkich urządzeń wymieniających dane, wprowadza standardowe, abstrakcyjne warstwy komunikacji systemów otwartych (komputery z oprogramowaniem, terminale oraz inne urządzenia końcowe), połączone różnymi (także skonkretyzowanymi) łączami komunikacyjnymi. Poniższy schemat przedstawia hierarchiczne warstwy modelu ISO/OSI (Rysunek 7).

Rysunek 7. Schemat modelu komunikacyjnego ISO/OSI.

Warstwa fizyczna (1)

Ta najniższa warstwa modelu odpowiada za fizyczną komunikację, czyli za przesyłanie przez medium komunikacyjne podstawowej informacji (strumienia bitów). Standardy w ramach tej warstwy dotyczą poziomów napięć sygnałów, czasu ich trwania, parametrów optycznych (w przypadku łączy światłowodowych), rodzajów nośnika jego fizycznych łączników, itp.

Warstwa łącza danych (2)

Warstwa ta jest odpowiedzialna za sterowanie dostępem do medium połączeniowego oraz sprawną transmisją strumieni bajtów za pomocą warstwy fizycznej. Na jej poziomie wprowadza się już pojęcie ramek (przygotowanych do transmisji pakietów danych o zmiennej długości, obłożonych polami informacji zapewniającymi skuteczność ich transmisji), które są przesyłane od nadawcy do odbiorcy. Gwarantuje ona mechanizmy kontroli i korygowania błędów, ogólnie zapewniające skuteczne przekazywanie ramek. W warstwie tej działa Ethernet.

Warstwa sieci (3)

Warstwa ta nie jest już związana z częścią sprzętową, a odpowiada za transmisje pakietów (rozumianych tu jako porcje danych na wyższym poziomie logicznym od ramek oraz zachowujących stały rozmiar) przez sieć. Jej zadanie to głownie znajdowanie odpowiednich dróg połączeń między systemami sieciowymi (routing). Pakiety przesyłane są w dwóch modelach komunikacji: połączeniowym lub bezpołączeniowym. W pierwszym przypadku najpierw ustanawiane jest połączenie sieciowe tzw. kanał wirtualny, którym przesyłane są dalej odpowiednio oznakowane i okrojone pakiety. W drugim natomiast każdy pakiet posiada niezbędną informację, potrzebną do prawidłowego dotarcia do adresata i jest przesyłany przez sieć niezależnie. Pakiety w trybie drugim mogą więc docierać do odbiorcy w innej kolejności niż zostały wysłane. Przykładem protokołu tej warstwy jest bezpołączeniowy protokół IP (Internet Protocol), na którym opiera się Internet.

Wszystkie warstwy logicznie wyższe od warstwy sieciowej pracują wyłącznie na końcowych, komunikujących się systemach. Na systemach biernie pośredniczących w komunikacji (w abstrakcyjnie logicznym sensie) są one nieaktywne.

Warstwa transportowa (4)

Ta pierwsza z wyższych warstw odpowiedzialna jest za dzielenie danych na bloki wiadomości, ich scalanie oraz transport między systemami końcowymi. Transport ten, w przeciwieństwie do zadań warstwy sieciowej, gwarantuje skuteczność poprzez kontrolę sytuacji wyjątkowych, takich jak zagubione czy powielone pakiety i zapewnienie mechanizmów nadzorujących ich kolejność i priorytety. Przykładowymi protokołami tej warstwy są bardzo rozpowszechnione w Internecie TCP (połączeniowy, Transmission Control Protocol) oraz UDP (bezpołączeniowy, User Datagram Protocol).

Warstwa sesji (5)

Warstwa ta odpowiada za sesję, czyli prowadzenie uporządkowanej wymiany danych między partnerskimi segmentami wyższej warstwy prezentacji, utrzymującymi logiczne połączenie, podczas którego wymieniają one między sobą dane. Warstwa sesji posiada mechanizmy umożliwiające sterowanie sesją miedzy aplikacjami w systemach końcowych, określa tryb sesji (przekazywanie danych: dwukierunkowe jednoczesne, naprzemienne, jednokierunkowe), realizuje funkcje synchronizujące dostęp do wspólnych zasobów oraz porządkuje proces wymiany danych. Przykładami protokołów funkcjonujących w Internecie w ramach tej warstwy są HTTP (Hyper Text Transfer Protocol) i FTP (File Transfer Protocol).

Warstwa prezentacji (6)

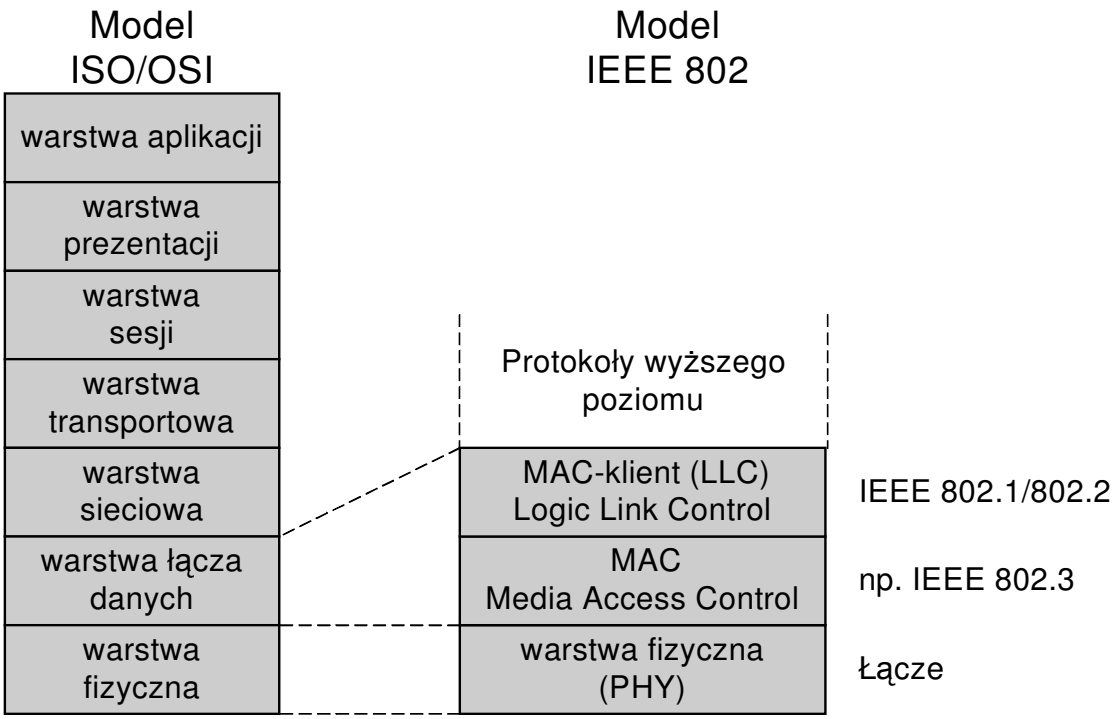
Kolejna warstwa odpowiada za konwersję typów danych przesyłanych między komunikującymi się systemami oraz dostarcza mechanizmów kompresji i szyfrowania.

Warstwa aplikacji (7)

Ostatnia warstwa jest odpowiedzialna za umożliwienie aplikacjom dostępu do komunikacji w ramach modelu ISO/OSI. Gwarantuje własność ostatecznej transparencji dla aplikacji (i użytkownika) mechanizmów komunikacji według tego modelu.

2.3.4 Ethernet a model ISO/OSI [N-2]

Podobnie jak wszystkie standardy w technice sieciowej, również Ethernet opiera się na modelu odniesienia OSI. Ethernet w modelu OSI zajmuje dwie warstwy (Rysunek 8).



Rysunek 8. Ethernet w odniesieniu do modelu referencyjnego ISO/OSI.

Podwarstwa klient-MAC

Podwarstwa MAC-klient może mieć postać jedną z poniższych:

- ✓ Warstwa LLC – Logical Link Control – jeśli węzeł należy do klasy urządzeń DTE. Podwarstwa ta odpowiada za interfejs między podwarstwą MAC (niezależnie od jej konkretnej specyfikacji zadanej przez protokół sieci lokalnej) a wyższymi warstwami modelu ISO/OSI zapewniającymi jej kompatybilność z logicznie wyższymi protokołami. Czyni to poprzez odpowiednie obudowywanie danych (pakietów tych protokołów) w kompletne ramki LLC, przesyłane dalej do niższej podwarstwy MAC (i odwrotnie). Definiuje ją standard IEEE 802.2.
- ✓ Warstwa mostowa – jeśli węzeł należy do klasy DCE. Warstwa odpowiada za interfejs między sieciami lokalnymi, używającymi tego samego lub różnego protokołu. Warstwę tą definiują podgrupy standardów IEEE 802.1.

Podwarstwa MAC

Media Access Control to z kolei podwarstwa kontroli dostępu do fizycznego łącza, dostarczająca konkretnych metod sterowania. Jest to podwarstwa specyficzna dla danego protokołu warstwy łącza danych, która konkretnie zajmuje się:

- ✓ przygotowywaniem i wysyłaniem ramek,
- ✓ wychwytywaniem ramek z ciągu bitowego w łączu,
- ✓ wykrywaniem błędów (np. poprzez sumy kontrolne),
- ✓ dołączaniem adresów MAC nadawcy i odbiorcy (48 bitowe, unikalne adresy),
- ✓ przestrzeganiem zdefiniowanych metod dostępu do łącza komunikacyjnego.

2.3.5 Ramka standardu Ethernet [N-2]

Ethernet opiera się na idei węzłów podłączonych do wspólnego medium (o kontrolowanym dostępie metodą CSMA/CD), wysyłających i odbierających za jego pomocą wyspecyfikowane poniżej ramki o zmiennej długości (Tabela 1).

<table>
<tr><td>56 bitów</td><td>8 bitów</td><td>48 bitów</td><td>48 bitów</td><td>16 bitów</td><td colspan="4">46 -1500 bajtów</td><td>32 bity</td></tr>
<tr><td rowspan="2">Preambuła</td><td rowspan="2">SFD</td><td rowspan="2">Adres odbiorcy (MAC)</td><td rowspan="2">Adres nadawcy (MAC)</td><td rowspan="2">Rozmiar pola danych LEN/TYPE</td><td colspan="4">Dane (wypełniane przez LLC)</td><td rowspan="2">`FCS</td></tr>
<tr><td>DSAP</td><td>SSAP</td><td>CTRL</td><td>NLI</td></tr>
</table>

Tabela 1. Schemat ramki standardu IEEE 802.3 (Ethernet)

Znaczenie kolejnych pól jest następujące:

1) **Preambuła** – 56 bitowe pole umożliwiające synchronizację komunikacji między odbiorcą i nadawcą.
2) **SFD** (Start Frame Delimiter) – 8 bitowe pole przyjmujące zawsze taką sama postać: 10101011 używane jako znacznik początku informacji zawartej w ramce.
3) **Adres docelowy (MAC)** – 48 bitowe pole zawierające adres MAC odbiorcy danych.
4) **Adres źródłowy (MAC)** – 48 bitowe pole zwierające adres MAC nadawcy danych.

5) **Długość pola danych** – 16 bitowe pole zawierające rozmiar (liczbę bajtów) pola danych w ramce Ethernet-u. Jego wartość służy głównie do rozpoznawania typu ramki Ethernet-owej (Ethernet I / Ethernet II) oraz protokołu wyższej warstwy logicznej modelu sieciowego (ogólnie dla wartości większych szesnastkowo od 05DC ramka jest rozpoznawana jako Ethernet II, natomiast np. dla protokołu IP wartość tego pola jest zawsze równa 0800, dla IPX 8137).
6) **Dane** – pole o zmiennej długości zawierające przesyłane dane (sekwencję bajtów dowolnej wartości). W polu danych znajdują się zwykle pakiety protokołów wyższych warstw modelu sieciowego (np.: pakiety IP, zawierające z kolei w swojej sekcji danych pakiety protokołu TCP lub Datagram-y UDP), odpowiednio wbudowane w pola specyficzne dla podwarstwy LLC (DSAP, SSAP, CTRL, NLI). Rozmiar pola danych jest ograniczony przez wartość1500 bajtów lecz nigdy nie jest mniejszy od 46 bajtów. W przypadku braku wystarczającej ilości danych do wysłania pole to jest uzupełnienie przez podwarstwę LLC do poziomu 46 bajtów.
7) **FCS** – 32 bitowe pole zawierające sekwencję sprawdzającą poprawność ramki przy użyciu sumy kontrolnej, metodą CRC (Cyclic Redundancy Check). Pole to pozwala protokołowi Ethernet-u na wykrywanie błędów transmisji ramek.

Istotną rzeczą jest to, że w Ethernecie wszystkie węzły posiadają unikalny adres MAC (48 bitowy adres, fabrycznie przypisany do węzła, np. karty sieciowej). Co ciekawe, adresy te są wspólne dla protokołów Ethernet, Wi-Fi, ATM, Token Ring, a także technologii Bluetooth oraz SCSI. Oryginalnie standard protokołu Ethernet działał w trybie half-duplex. Tylko jeden węzeł mógł przesyłać dane w określonym momencie. W nowszej rewizji standardu wprowadzono tryb full-duplex i technologię przełączania, która umożliwia transmisję i odbiór ramek w tym samym czasie w segmentowe łącze danych. Ponieważ równoczesna transmisja i odbiór spowodowałaby w trybie half-duplex kolizję, część protokołu odpowiedzialna za wykrywanie kolizji została dezaktywowana.

Transmisja ramek [N-2]

Za każdym razem, gdy węzeł w ramach podwarstwy logicznej MAC, otrzyma zlecenie wysłania ramki od podwarstwy LLC (z danymi, zawierającymi pakiety protokołów wyższych warstw modelu ISO/OSI) rozpoczyna on sekwencję transmisji, przekazując informacje z podwarstwy LLC do bufora ramki MAC. Pole preambuły oraz SFD są najpierw umieszczane w ramce. Następnie do bufora „ładowane" są adresy nadawcy oraz odbiorcy. Obliczana jest potem długość ciągu danych z podwarstwy LLC, która uzupełniona do szesnastobitowej długości pola jest w dalszej kolejności „wrzucana" do bufora ramki, zaraz przed przesłaniem tam ciągu bajtów danych z podwarstwy LLC. Na koniec obliczana jest wartość sumy kontrolnej (pole FCS) na podstawie adresów, długości danych oraz pola danych i ona kończy „ładowanie" bufora ramki MAC. Po złożeniu ramki jej transmisja będzie zależna od tego czy MAC operuje w trybie half-duplex czy full-duplex. Standard IEEE 802.3 wymaga obecnie, aby wszystkie węzły Ethernet-u miały możliwość pracy w pierwotnym trybie half-duplex (tryb full-duplex jest opcjonalny).

Odbiór ramek [N-2]

Proces odbioru ramki jest natomiast analogiczny dla dwóch trybów dostępnych w ramach MAC – half-duplex i full-duplex, z tym, że dla trybu full-duplex wymagane są oddzielne bufory ramek oraz ścieżki danych, aby umożliwić równoczesny odbiór i transmisję. Odbiór ramki jest odwrotnością transmisji. Adres nadesłanej ramki w ramach podwarstwy MAC porównywany jest z listą adresów węzła (jego adres MAC, ew. adres grupowy i adres rozgłoszeniowy) w celu weryfikacji odbiorcy ramki. Jeśli adres węzła nie zgadza się z wartością adresu MAC odbiorcy w ramce zostaje ona odrzucona. Jeśli jednak adres zostanie poprawnie zweryfikowany sprawdzana jest długość ramki oraz porównana zostaje wartość odczytanej sumy kontrolnej CRC z pola FCS odebranej ramki z wartością obliczoną na podstawie innych pól odebranej ramki. Jeśli długość ramki oraz wartość pola FCS zostały sprawdzone i się zgadzają, zostaje odczytany typ ramki z pola długości danych. Na końcu ramka zostaje rozbita na poszczególne elementy i wartość pola danych, zawierająca pola specyficzne dla LLC przesłana zostaje do tej wyższej podwarstwy.

2.4. Protokoły sieciowe [N-1]

Zestaw protokołów został opracowany w celu umożliwienia komunikacji między różnymi typami systemów komputerowych, jak również między różnymi sieciami. Agencja DARPA oraz Stanford University rozpoczęły pracę nad protokołem TCP w 1973 r. Efektem 5-letniego okresu badań było opracowanie dwóch wzajemnie uzupełniających się protokołów: protokołu połączeniowego TCP i protokołu bezpołączeniowego IP (stąd nazwa TCP/IP). W 1983 r. przyjęte zostały jako standard wojskowy. Protokoły TCP/IP są wykorzystywane w systemach UNIX-owych, sieciach lokalnych i sieciach rozległych. Protokoły służą do łączenia oddzielnych fizycznie sieci w jedną sieć logiczną.

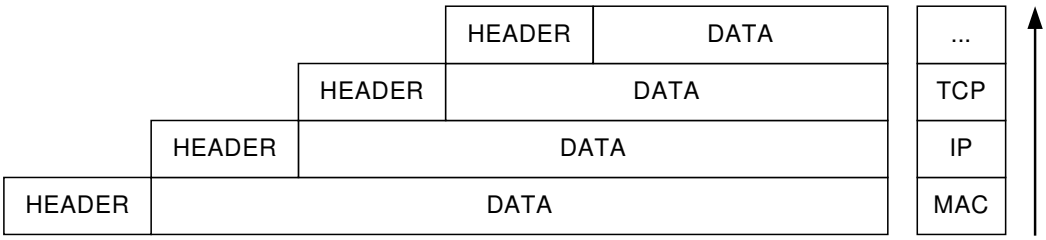
Do najistotniejszych zalet protokołów TCP/IP można zaliczyć:

- ✓ Otwartość i niezależność od specyfikacji sprzętowo-programowej systemów komputerowych.
- ✓ Możliwość integracji wielu różnych rodzajów sieci komputerowych.
- ✓ Wspólny schemat adresacji, pozwalający na jednoznaczne zaadresowanie każdego użytkownika.
- ✓ Istnienie standardowych protokołów warstw wyższych.

Protokoły TCP/IP to dzisiaj cały zestaw protokołów przeznaczonych do:

- ✓ Transferu danych: IP, TCP, UDP (User Datagram Protocol).
- ✓ Kontroli poprawności połączeń: ICMP (Internet Control Message Protocol).
- ✓ Zarządzania siecią: SNMP (Simple Network Management Protocol).
- ✓ Zdalnego włączania się do sieci: TELNET.
- ✓ Usług aplikacyjnych typu przesyłanie plików: FTP (File Transfer Protocol).

Protokół IP (Internet Protocol) przypisany jest warstwie sieciowej. Warstwa ta jest odpowiedzialna za adresowanie i znajdowanie drogi – tzw. „routing" pakietów – Datagramów (podstawowa jednostka przesyłania danych) od stacji nadawczej, poprzez wiele sieci, aż do odbiorcy. Każda stacja, która znajduje się w sieci opartej o komunikację TCP/IP musi posiadać swój własny adres IP. Dzięki temu adresowi sieć rozpoznaje miejsce docelowe i Datagram zostaje dostarczony do odpowiedniego odbiorcy, niezależnie od rodzaju środka transportu (Ethernet, ISDN i inne). IP jest protokołem bezpołączeniowym, co oznacza, że nie musi być zachowana kolejność przy przesyłaniu poszczególnych Datagram-ów do stacji odbiorczej. IP nie gwarantuje także pewnego przesłania danych, co z kolei zapewniają protokoły osadzone na wyższych warstwach (np. TCP – Rysunek 9).



Rysunek 9. Przykładowa idea budowy telegramu Ethernet-owego TCP/IP.

Każda warstwa posiada swój nagłówek (Header), z kolei warstwa nad nią wykorzystuje już tylko pole danych z poprzedniej warstwy. Telegramy warstw wyższych są jak gdyby pakowane do większego „pudełka" warstwy niższej. Dzięki takiej konstrukcji telegramów protokoły nie są związane na stałe z daną warstwą modelu ISO/OSI i w „standardowym" telegramie Ethernet-owym, oprócz protokołu TCP/IP, w wyższych warstwach mogą być zawarte inne protokoły np. UDP. Pole danych w ramce Ethernet-owej daje możliwość stosowania różnych typów telegramów przy zastosowaniu standardowych kart sieciowych.

Pod pojęciem „TCP/IP" przyjęto określać szeroką gamę protokołów. Najważniejsze z nich przedstawiają poniższe tabele.

ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
IP	**Internet Protocol** (stosowany w IPv4 i IPv6)
GGP	Gateway-to-Gateway Protocol
EGP	Exterior Gateway Protocol
RSVP	Resource Reservation Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Multicasting
OSFP	Open shortest Path First
UDP	**User Datagram Protocol**
TCP	**Transmission Control Protocol**

Tabela 2. Gama protokołów typu „TCP/IP"

Socket-Interface:	UDP i TCP
TelNet	Remote Terminal Protocol
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
TFTP	Trivial File Transfer Protocol
DNS	Domain Name System
RIP	Routing Information Protocol
RPC	Remote Procedure Calls
XDR	eXternal Data Representation
NFS	Network File System
SNMP	Simple Network Management Protocol
IPSec	Secutity Protocol

Tabela 3. Interfejsy programowe dla UDP oraz TCP

2.4.1 IP [N-1]

Internet Protocol (IP) jest protokołem bezpołączeniowym, zdefiniowanym w dokumencie RFC 791. Protokół IP jest przeznaczony do sieci z komutacją pakietów. Pakiet jest nazywany przez IP Datagram-em. Każdy Datagram jest podstawową, samodzielną jednostką przesyłaną w sieci na poziomie warstwy Internet. Datagram-y mogą być adresowane do pojedynczych węzłów lub do wielu węzłów. W przesyłaniu Datagram-ów poprzez sieci uczestniczą router-y (węzły sieci), które określają dla każdego Datagram-u trasę od węzła źródłowego do węzła docelowego.

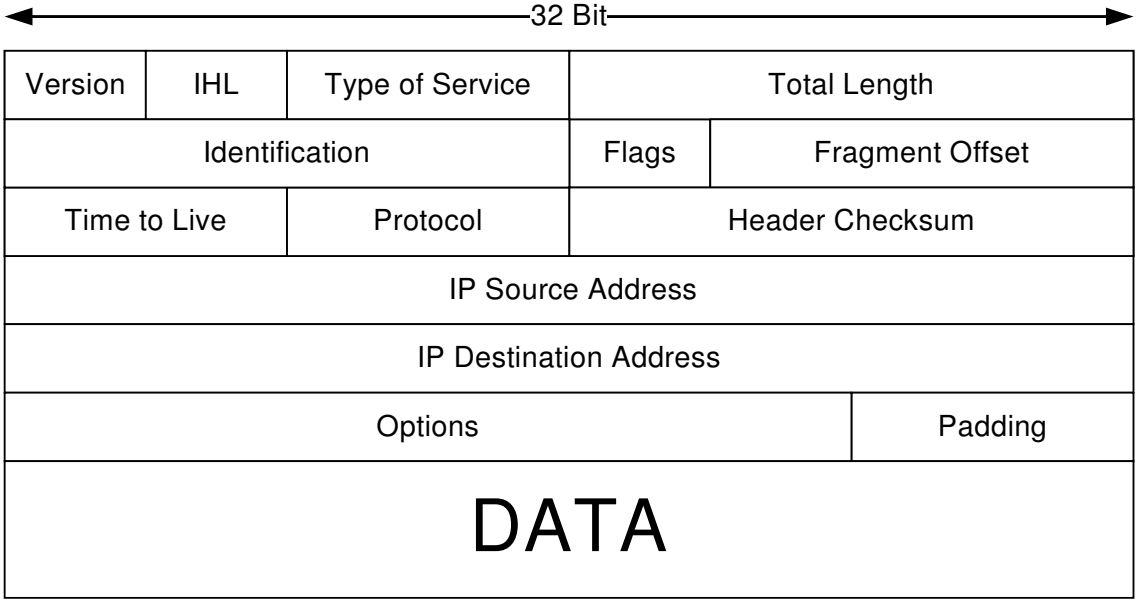
Do podstawowych funkcji protokołu IP możemy zaliczyć:

- ✓ Określenie struktury Datagram-u.
- ✓ Określenie schematu adresacji.
- ✓ Kierowanie ruchem Datagram-ów w sieci.
- ✓ Dokonywanie fragmentacji Datagram-u i odtwarzanie oryginalnego Datagram-u z jego fragmentów.

Ponieważ w różnych sieciach mogą być ustalone różne maksymalne długości Datagram-ów, w zależności od potrzeb, Datagram może być podzielony na kilka mniejszych części. Tę operacją nazywamy fragmentacją Datagram-ów. Format każdego fragmentu jest taki sam jak format każdego innego niepodzielnego Datagram-u. Konieczność fragmentacji może być również następstwem przesyłania Datagram-ów przez sieci rozległe dopuszczające inne protokoły i inne długości pakietów, np. sieci X.25 z pakietami o maksymalnej długości 128 bajtów. Kompletowanie pierwotnego Datagram-u z fragmentów dokonuje się w stacji docelowej. Z chwilą nadejścia pierwszego fragmentu ustala się czas oczekiwania na skompletowanie Datagram-u. Jeśli w tym okresie czasu nie nadejdą pozostałe fragmenty to następuje przerwanie oczekiwania i skasowanie już otrzymanych fragmentów.

Datagram przypomina ramkę sieci fizycznej. Różnica polega na tym, że nagłówek ramki zawiera adresy fizyczne, zaś nagłówek Datagram-u adresy IP (Rysunek 10). Ponieważ przetwarzaniem Datagram-ów zajmują się programy, zawartość i format Datagram-ów nie są uwarunkowane sprzętowo. Każdy IP – Datagram składa się z nagłówka i bloku danych.

2.4.1.1 Nagłówek IP - Datagramu



Rysunek 10. Format nagłówka Datagram-u IP (IPv4).

Opis poszczególnych elementów nagłówka IP:

1) **Version** - wersja (4 bity) określa numer użytej wersji protokołu IP. Jest to konieczne, ponieważ router-y lub komputery w sieci mogą używać różnych wersji IP (najbardziej rozpowszechniona - IPv4).
2) **IHL** - długość nagłówka (4 bity) określa liczbę słów 32 bitowych składających się na nagłówek Datagram-u. Typowa (minimalna) długość nagłówka wynosi 5 (5x32).
3) **Type of Service** - typ usług (8 bitów) określa jakość usług jakiej wymaga się od sieci. Znaczenie poszczególnych bitów tego pola jest następujące:
   - ✓ Pierwsze trzy pola określają tzw. pierwszeństwo, np. 000 - oznacza Datagram zwykły, 001 - priorytetowy, 010 - natychmiastowy, 011 błyskawiczny, a 100 - Datagram super błyskawiczny.
   - ✓ Bit czwarty to bit *D* określający opóźnienie w sieci (D=0 oznacza normalne, D=1 małe opóźnienie).
   - ✓ Kolejny bit to bit *T* związany z przepustowością (T=0 oznacza normalną, a T=1 dużą przepustowość).
   - ✓ Bit szósty *R* pozwala wybrać niezawodność w dostarczeniu Datagram-u (R=0 normalna, R=1 duża niezawodność).
   - ✓ Ostatnie dwa bity mają wartości równe zeru i są zarezerwowane dla przyszłych zastosowań.

4) **Total Length** – całkowita długość pakietu (16 bitów) definiuje długość Datagram-u IP w bajtach (oktetach). Maksymalna długość Datagram-u wynosi 65535 bajtów.

Kolejne trzy pola w nagłówku są wykorzystywane przez protokół IP do fragmentacji Datagram-ów i do operacji odwrotnej, tzn. do składnia z krótkich fragmentów pierwotnego Datagram-u. Te pole to identyfikacja, flaga i przesunięcie fragmentu.

5) **Identification** - identyfikacja (16 bitów) jest używana do jednoznacznego oznaczenia każdego fragmentu pierwotnego Datagram-u. Identyfikator zamieszczony w tym polu jest powtarzany we wszystkich fragmentach składających się na pierwotny Datagram.
6) **Flags** - flagi zawierają 3 bity, (pierwszy - zawsze zero, drugi określa czy można (1) czy nie można (0) fragmentować Datagram, trzeci - identyfikacja ostatniego fragmentu składającego się na pierwotny Datagram (wartość 0 określa ostatni, 1 oznacza kolejny fragment)).
7) **Fragment Offset** - przesunięcie fragmentu (13 bitów) wskazuje, którą częścią całości pierwotnego Datagram-u jest dany fragment. Poszczególne fragmenty mają pola danych o długości będącej wielkością 8 bitów. Wyjątkiem jest ostatni fragment, którego długość wynika z długości pierwotnego Datagram-u. W polu tym podaje się o ile zawartość fragmentu jest przesunięta w stosunku do początku pola danych pierwotnego Datagram-u.
8) **Time To Live** - czas życia (8 bitów) jest parametrem określającym ile czasu Datagram może przebywać w sieci. Czas życia Datagram-u ustala nadawca umieszczając w tym polu liczbę naturalną. Przy przejściu przez kolejny router liczba ta jest zmniejszana o 1. Zmniejszenie do zera powoduje odrzucenie Datagram-u.
9) **Protocol** - protokół (8 bitów) określa numer protokołu warstwy transportowej, do którego należy przesłać dane z Datagram-u; np. numer 6 oznacza protokół TCP, a numer 1 protokół ICMP.
10) **Header Checksum** - suma kontrolna nagłówka (16 bitów) służy do sprawdzania poprawności odbioru wyłącznie nagłówka Datagram-u. Jest to szesnastobitowe, jedynkowe uzupełnienie jedynkowo uzupełnionej sumy wszystkich szesnastobitowych słów nagłówka i danych pakietu. Przy obliczaniu sumy kontrolnej przyjmuje się, że pole to zawiera same zera. Suma ta podlega weryfikacji i modyfikacji, np. w trakcie zmian pola „czas życia" w każdym węźle sieci.
11) **IP Source Address** – adres stacji źródłowej (32 bity).

12) **IP Destination Address** – adres stacji docelowej (32 bity).
13) **Options** - opcje zmiennej długości będącej wielokrotnością 8 bitów, są wykorzystywane do określenia dodatkowych wymagań dotyczących sposobu przesyłania Datagram-u, np. do rejestrowania przebytej trasy lub do zapamiętania trasy zdefiniowanej w węźle źródłowym. Pole to nie musi występować w nagłówku Datagram-u.
14) **Padding** - wypełnienie jest ewentualnym dopełnieniem pola opcje do wielokrotności 32 bitów.

2.4.1.2 Adresy IP [5]

Adres IP stacji nadawczej oraz adres IP stacji docelowej jest zawarty w każdym pakiecie danych przesyłanych przez protokół IP. Adresy IP muszą być zawsze jednoznaczne.
Adres IP ma 32 bitowy rozmiar, zapisany binarnie, który jest podzielony na 4 grupy po 1 bajcie każda, oddzielone miedzy sobą kropką, np.11110001.00010101.00001111.11110000 . Taki sposób zapisu jest bardzo uciążliwy i można łatwo popełnić błąd. Z tego powodu adresy IP są przeliczane i podawane decymalnie. Adres decymalny dla powyższego przykładu ma postać: 241.21.15.240 .

2.4.1.3 Dynamiczne adresy IP - DHCP

Protokół DHCP (Dynamic Host Configuration Protocol) przypisuje dynamicznie adres IP danej stacji w sieci z przygotowanej do tego celu tablicy numerów IP na określony okres czasu. Aby takie przypisywanie było możliwe, w sieci musi znajdować się serwer DHCP, który rozporządza konfiguracją numerów IP, udostępnionych do przydzielania w sieci. Podczas boot-owania stacje meldują się w sieci i otrzymują od serwera DHCP adresy IP oraz przynależne parametry (np. maska podsieci).

2.4.1.4 Sub Network Mask

Maski podsieci służą do wyodrębniania w adresie IP części sieciowych od części użytkowych (host-ów). Zbudowana jest tak jak adres IP. Zaznacza ona część adresu IP, która odzwierciedla numer sieciowy.

Przykład maski podsieci:

- ✓ zapis decymalny – 255.255.255.0,
- ✓ zapis binarny – 11111111. 11111111. 11111111.00000000

Część numeru sieci jest wypełniona przez „1", część użytkowników wypełnia „0".
Powyższa maska podsieci jest zakwalifikowana do C-klasy ([2] - więcej informacji).

Adresy IP oraz maski podsieci są ze sobą połączone poprzez logiczną funkcję „i" (And). Wynikiem jest adres sieciowy. Podczas budowania połączenia komunikacyjnego sterownik IP porównuje własny adres IP z adresem stacji odbiorczej. Jeśli obie stacje znajdują się w tej samej podsieci to stacja docelowa może zostać połączona poprzez mechanizm ARP, bez routing-u. W przypadku, gdy stacja chce komunikować się z inna stacją spoza podsieci pakiety IP muszą być przesyłane do router-a lub bramy (gateway), aby następowało ich przekazywanie.

2.4.2 TCP [N-1]

Transmission Control Protocol (RFC 793) jest protokołem zorientowanym połączeniowo, czyli umożliwiającym zestawienie połączenia, w którym efektywnie i niezawodnie przesyłane są dane. Połączenie to charakteryzuje się możliwością sterowania przepływem, potwierdzania odbioru, zachowania kolejności danych, kontroli błędów i przeprowadzania retransmisji. Blok danych wymieniany między współpracującymi komputerami nosi nazwę segmentu - nagłówek + dane (Rysunek 11).

Ze względu na połączeniowe zorientowanie protokołu TCP w celu przesłania danych między dwoma modułami TCP, zainstalowanymi w różnych stacjach, konieczne jest ustanowienie, utrzymanie i rozłączenie połączenia wirtualnego („three-way handshake").
Ustanowienie połączenia odbywa się w następujących etapach:

- ✓ Nadawczy moduł TCP wysyła do odbiorczego modułu TCP segment z bitem SYN=1 i z proponowanym numerem INS w polu „numer sekwencyjny".
- ✓ Odbiorczy moduł TCP, jeśli zgadza się na ustanowienie połączenia, to przesyła zwrotnie segment z bitami SYN=1 i ACK=1, a w polu „numer sekwencyjny" podaje numer INS, z którym rozpocznie działanie.
- ✓ Nadawczy moduł TCP wysyła segment z potwierdzeniem otrzymania zgody (ACK=1) na ustanowienie połączenia i równocześnie zawierający dane.

W ten sposób zostaje ustanowione połączenie wirtualne między dwoma modułami TCP i mogą zostać przesyłane segmenty z danymi. Segmenty te mogą być przesyłane w tym połączeniem w obu kierunkach, ponieważ TCP umożliwia transfer danych między dwoma modułami w trybie dupleksowym.

Dla zapewnienia niezawodnej transmisji TCP wykorzystuje sekwencyjną numerację bajtów oraz mechanizm pozytywnych potwierdzeń z retransmisją. Numer sekwencyjny przypisany do każdego przesyłanego bajtu danych pozwala na jego jednoznaczną identyfikację, a także jest używany w mechanizmie przesyłania potwierdzeń. Ponieważ kolejne bajty są numerowane począwszy od INS, numer pierwszego bajtu wysłanego w połączeniu wirtualnym wynosi INS+1 (zazwyczaj INS=0).
Nadawczy moduł TCP dokonuje retransmisji danych do czasu, aż otrzyma potwierdzenie poprawnego ich przyjęcia przez odbiorczy moduł TCP. Rozpoczęcie retransmisji uwarunkowane jest przekroczeniem wcześniej ustalonego czasu oczekiwania na nadejście potwierdzenia.
Poprawność odbioru danych sprawdzana jest przy użyciu pola „suma kontrolna", znajdującego się w nagłówku segmentu. Jeżeli dane są akceptowane to moduł TCP wysyła zwrotnie pozytywne potwierdzenie. Jest ono zawarte w polu „numer potwierdzenia". Wszystkie bajty danych, o numerach sekwencyjnych mniejszych od wartości zawartej w tym polu, zostały odebrane poprawnie.
W sytuacji, gdy dane zostały odebrane poprawnie, a nadawczy moduł TCP retransmitował je np. z powodu zaginięcia segmentu z pozytywnym potwierdzeniem, odbiorczy moduł TCP ma możliwość odrzucenia nadmiarowych danych (duplikatów).

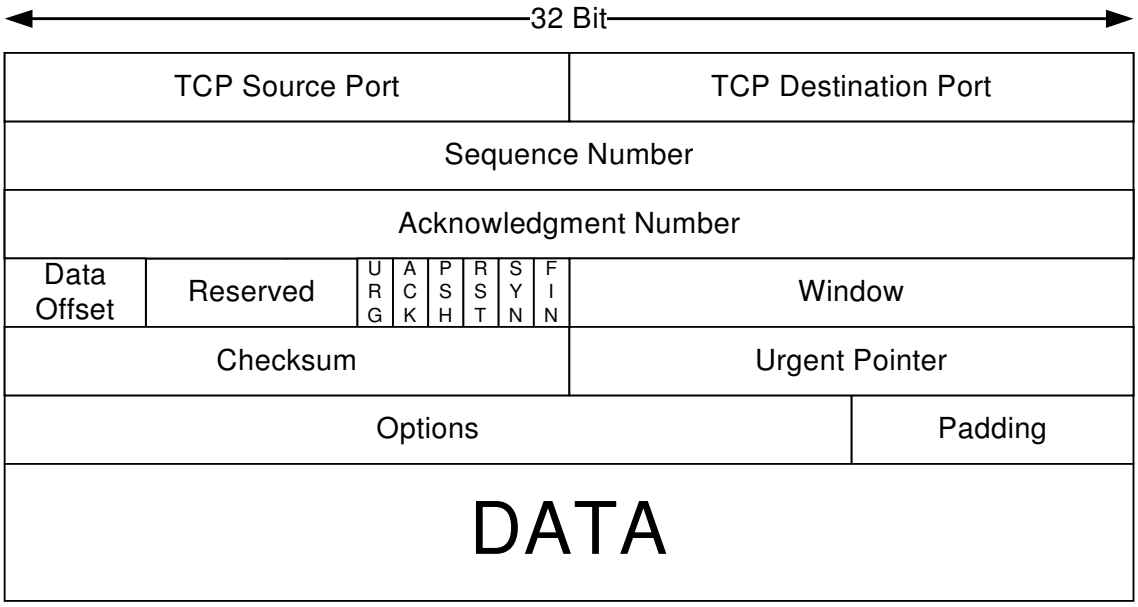
Protokół TCP umożliwia również zarządzanie buforami. Odbywa się to przez wskazanie ile bajtów odbiorczy moduł TCP jest w stanie zaakceptować. Liczba akceptowanych bajtów określona jest w polu „okno" w nagłówku segmentu przesyłanego do nadawczego modułu TCP. Liczba ta może być zmieniana w trakcie trwania połączenia wirtualnego.

TCP realizuje również koncepcję funkcji wymuszającej. Operacja ta jest realizowana wtedy, gdy aplikacja „chce mieć pewność", że wszystkie dane przekazane przez nią do modułu TCP zostały wysłane. W odpowiedzi na żądanie aplikacji, moduł TCP wysyła wszystkie dane znajdujące się w buforach w postaci jednego lub kilku segmentów do odbiorczego modułu TCP. Działanie funkcji wymuszającej sygnalizowane jest bitem PSH równym 1.

Po przesłaniu danych następuje rozłączenie połączenia wirtualnego. Jest ono realizowane trójetapowo z użyciem bitu FIN ustawionego na 1.

Należy tu przypomnieć, że moduł TCP w celu przesyłania segmentu przez sieć przekazuje go do warstwy Internet. Tam jest on umieszczany wewnątrz Datagram-u, czyli inaczej segment jest uzupełniany o nagłówek Datagram-u IP. Z kolei protokół IP przekazuje ten Datagram do warstwy dostępu do sieci, gdzie po obudowaniu o kolejny nagłówek tworzona jest ramka przesyłana przez sieć.

2.4.2.1 Nagłówek protokołu TCP



Rysunek 11. Budowa nagłówka protokołu TCP.

Znaczenie poszczególnych pól segmentu:

1) **TCP Source Port** – port źródłowy (16 bitów).
2) **TCP Destination Port** –port docelowy (16 bitów).

Oba te pola zawierają numery portów procesów aplikacyjnych, korzystających z usług TCP. Kombinacja tych numerów z adresami sieciowymi określa parę gniazd tworzących połączenie protokołu TCP.

3) **Sequence Number (Seq) –** numer sekwencyjny (32 bity). Pole to zawiera numer sekwencyjny pierwszego bajtu danych w segmencie. Ta wartość określa pozycję segmentu w strumieniu bajtów. Podczas ustanawiania połączenia(i jeśli bit „syn" w polu „znaczniki" jest ustawiony na 1) to w tym polu zawarty jest inicjujący numer sekwencyjny „INS", od którego rozpoczyna się numerację bajtów w połączeniu. Zatem pierwszy wysłany bajt ma numer „INS + 1".
4) **Acknowledgment Number (Ack)** – numer potwierdzenia (32 bity). Pole to zawiera numer sekwencyjny następnego oczekiwanego bajtu po stronie odbiorczej. Jednocześnie jest to potwierdzenie poprawnego odbioru bajtów o numerach sekwencyjnych mniejszych od zawartego w tym polu. Potwierdzenia mówią nadawcy ile bajtów danych zostało już poprawnie odebranych.
5) **Data Offset** - Pole „długość nagłówka" (4 bity) określa liczbę 32 - bitowych słów w nagłówku segmentu TCP. Tym samym określone zostaje miejsce, w którym rozpoczynają się dane. Pole to ma tak określone znaczenie tylko wtedy, gdy bit „ACK" równy jest 1.
6) **Reserved** -Pole „rezerwa" (6 bitów) jest przeznaczone dla przyszłych zastosowań. Zawiera same zera.
7) **URG, ACK, PSH, RST, SYN, FIN** – znaczniki - sześć bitów sterujących, które ustawione na 2 mają następujące znaczenie:
   - ✓ UGR - wskazuje na ważność pola „wskaźnik pilności" (0 byte!).
   - ✓ ACK - wskazuje na ważność pola „numer potwierdzania" (0 byte!).
   - ✓ PSH - wskazuje funkcję wymuszającą wysyłanie segmentu (0 byte!).
   - ✓ RST - wyzerowanie połączenia (0 byte!).
   - ✓ SYN - wskazuje, że w polu „numer sekwencyjny" umieszczony jest inicjujący numer sekwencyjny INS. Jest on przeznaczony do synchronizacji numerów sekwencyjnych w fazie ustanowienia połączenia (1 Byte!).
   - ✓ FIN - wskazuje, że nadawca nie ma nic więcej do nadania - sygnał końca danych (1 Byte!).
8) **Window** – Pole „okno" (16 bitów) określa liczbę bajtów, jaką może jeszcze zaakceptować odbiorczy moduł TCP (jaka przestrzeń pamięci jest wolna).

9) **Checksum** – Pole „suma kontrolna" jest 16 - bitowym jedynkowym uzupełnieniem jedynkowo uzupełnionej sumy wszystkich 16 - bitowych słów w segmencie. Ta suma obejmuje zarówno nagłówek jak i dane segmentu.
10) **Urgens Pointer** – Pole „wskaźnik pilności" (16 bitów) jest interpretowany tylko wtedy, gdy bit URG jest równy 1. Pole to zawiera numer sekwencyjny bajtu następującego po pilnych danych.
11) **Options** – opcje. Pole to ma długość zmienną będącą wielokrotnością 8 bitów. Zawiera ono numery opcji - każdy numer zapisany w jednym bajcie. Dla protokołu TCP zdefiniowano trzy opcje:
    - ✓ 0 - koniec listy opcji,
    - ✓ 1 - brak działania,
    - ✓ 2 - maksymalna długość segmentu.
12) **Padding** - wypełnienie. Pole to uzupełnia nagłówek do wielokrotności 32 bitów.

2.4.2.2 Numery portów [5]

W warstwie transportowej modelu ISO/OSI dla adresowania aplikacji stosuje się numery portów, które są definiowane podczas połączenia. W zależności jaki port zostanie podany w protokole TCP, następuje wysłanie pakietu do przyporządkowanego temu portowi protokołowi lub aplikacji.
Adresowanie do stacji docelowej dokonywane jest poprzez kombinację adresu IP i numeru portu. Kombinacja ta jest określana jako „Socket" (Rysunek 12).

Adres IP		Numer portu		SOCKET
197.12.18.13	+	80	=	197.12.18.13,80

Rysunek 12. Socket – kombinacja numeru IP oraz portu TCP.

Pole portu ma rozmiar 16 bitów, co oznacza, że każdy uczestnik ma do dyspozycji 65 535 portów. Określony szereg portów jest jednak zarezerwowany, z których część jest na stałe przypisana pewnym aplikacjom (HTTP, FTP i inne) i jest określana jako „well-known-ports"(Tabela 4).

Numer portu	Aplikacja
0	----------
1-255	„Well known ports”
20/21	FTP
23	Telnet
25	SMTP
80	HTTP
256-1023	Porty zarezerwowane
1024-65535	Porty użytkowników

Tabela 4. Numery portów protokołu TCP.

Obecnie trudno jest rozgraniczyć jakie porty należy zaliczyć do ogólnie znanych, gdyż część firm używa portów spoza przedziału portów standardowych i zarezerwowanych, a jednak tworzą one jakoby nowe standardy, które są wykorzystywane przez wielu użytkowników, a nigdzie nie są określone.

2.4.3 ICMP [N-1]

Protokół ten jest ściśle związany z protokołem IP i jego częścią warstwy Internet. Protokół IP, jako protokół bezpołączeniowy nie posiada mechanizmów informowania o błędach. Do tego celu przeznaczony jest protokół ICMP. Umożliwia on przesyłanie między komputerami lub router-ami informacji o błędach występujących w funkcjonowaniu sieci IP, takich jak np.:

- ✓ Brak możliwości dostarczenia Datagram-u do miejsca przeznaczenia.
- ✓ Zmiana wcześniej wyznaczonej trasy przez jeden z pośredniczących router-ów.
- ✓ Brak wolnej pamięci buforowej dla zapamiętania Datagram-u.

Informacje o tych zaburzeniach w działaniu sieci noszą nazwę komunikatów. Komunikaty protokołu ICMP są przesyłane wewnątrz Datagram-ów IP. Każdy komunikat ma własny format. Jednak wszystkie rozpoczynają się takimi samymi polami: „typ", „kod" oraz „suma kontrolna". Dalsze pola zależą od typu komunikatu ICMP. Specyfikacja tego protokołu zapisana jest w dokumencie RFC 792.

2.4.4 UDP [N-1]

Protokół UDP (User Datagram Protocol) przedstawiono w dokumencie RFC 760. Jest to protokół bezpołączeniowy, nie posiadający mechanizmów sprawdzających poprawność dostarczenia danych. Protokół UDP został opracowany w celu stworzenia aplikacjom możliwości bezpośredniego korzystania z usług IP. Pozwala on na dołączanie do Datagram-ów IP adresów portów komunikujących się aplikacji.
Protokół UDP jest wykorzystywany w sytuacjach, gdy przesyłamy niewielką liczbę danych. Również protokół ten mogą używać aplikacje działające według modelu 'zapytanie-odpowiedź". Ogólnie możemy powiedzieć, że UDP może być z powodzeniem używany tam gdzie nie są wymagane usługi protokołu TCP.

2.4.5 SINEC H1 (ISO) [5]

Protokół SINEC H1 został wprowadzony przez firmę Siemens w 1985 roku. Dostępne standardy (IEEE 802.3) zostały wykorzystane i uzupełnione w zastosowaniach komunikacji przemysłowej o ten protokół . Nazwa SINEC H1 została w międzyczasie zastąpiona określeniem „SIMATIC NET".

H1 jest protokołem określonym przez normę ISO 8073 klasy 4, który jest przyporządkowany 4-tej warstwie modelu referencyjnego ISO/OSI. Norma ISO 8073 opisuje protokoły transportowe zorientowane połączeniowo. Rozróżnionych jest w tej normie 5 klas protokołów, przy czym każda klasa posiada swoje specyficzne właściwości. Protokoły klasy czwartej, spełniają poniższe kryteria:

- ✓ Likwidowanie błędów wykrytych w warstwie trzeciej.
- ✓ Rozpoznanie i likwidowanie błędów, które nie zostały wykryte w warstwie trzeciej.
- ✓ Multipleksowanie.

- ✓ Kontrola przepływu.
- ✓ Segmentacja.
- ✓ Rozpoznanie danych priorytetowych (szybkie przesłanie).

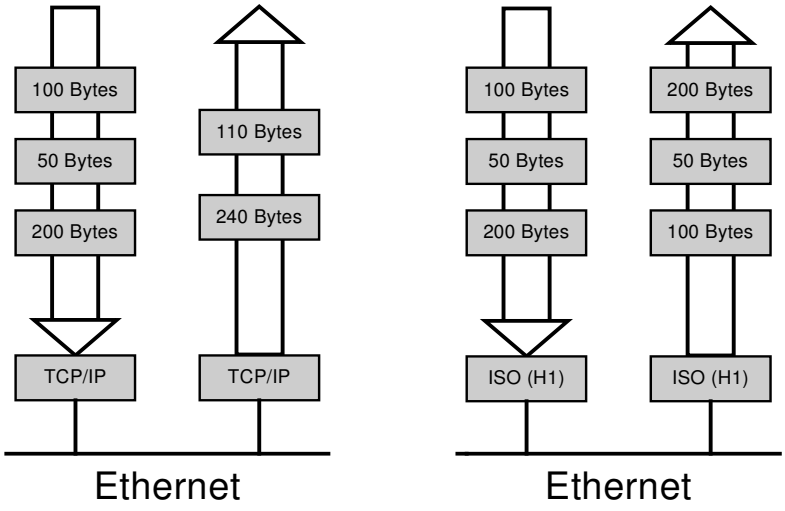
H1 jest, podobnie jak protokół TCP, protokołem zorientowanym połączeniowo, co oznacza, że przed wymianą danych musi zostać zbudowane połączenie do stacji docelowej. Dane wysłane zostają przez stację odbiorczą pokwitowane, a w przypadku błędu następuje ponowne przesłanie danych przez stację nadawczą. Taka wymiana informacji pozwala na określenie: „gwarantowana wymiana danych".
Podobnie jak w komunikacji opartej o protokół TCP, H1 umożliwia wymianę danych pomiędzy dwiema aplikacjami. Aplikacje wyższych warstw są adresowane poprzez TSAPs (analogicznie do numerów portów w połączeniu TCP). Komunikacja z wykorzystaniem protokołu H1, w odróżnieniu do protokołu TCP, nie jest zbudowana na Protokole IP. Dla tego przy komunikacji H1 brak jest adresów IP. Adresowanie przebiega w oparciu o adresy MAC i TSAP.

H1 (ISO), aby móc zaadresować aplikacje, pracuje na warstwie transportowej z wykorzystaniem TSAP (Transport Service Access Points). Dla każdego połączenia pomiędzy komunikującymi się partnerami wykorzystane są różne pary TSAP. Dzięki tym parametrom połączenie pomiędzy dwiema stacjami jest jednoznacznie określone. TSAP maja długość od 2 do 16 bajtów.

2.4.6 RFC 1006 i SPS Header

TCP zapewnia pewne oraz zgodne z kolejnością przesyłanie danych. Protokół ten pracuje jako zorientowany połączeniowo, co w tym przypadku oznacza, że przygotowane do transportu dane nie muszą pozostać przy ich dostarczeniu w takiej samej postaci. Dwie wysłane osobno grupy informacji mogą zostać dostarczone do odbiorcy jako jedna grupa. Wysłane i odebrane bloki danych nie musza być sobie równe.

H1 (ISO) w odróżnieniu od TCP funkcjonuje jako zorientowany „pakietowo". Protokół ten posiada znacznik EOM (End Of Message), co umożliwia transportowanie danych jako wiadomości. Pakiety danych dochodzą do odbiorcy w takiej samej postaci, w jakiej zostały wysłane od nadawcy. Bloki wejściowe i wyjściowe są identyczne (Rysunek 13).



Rysunek 13. Przesył danych zorientowany połączeniowo (TCP) i pakietowo (H1) .

Aby nadać TCP strukturę bloków (zorientowanie pakietowe) istnieją 2 możliwości:

- ✓ INAT SPS-Header
- ✓ ISO on TCP (RFC 1006)

INAT SPS Header

Aby pakiety danych protokołu TCP podzielić na rozpoznawane przez sterownik programowalny (PLC) bloki danych, konieczne jest rozpoznanie danych powyżej protokołu TCP. Dla transportu, poprawnego rozpoznawania, a przede wszystkim dla określania poprawnej wielkości przesyłanych danych został stworzony INAT SPS-Header, będący otwartym protokołem. Protokół ten jest rozpowszechniony na całym świecie pod nazwami SPS Header i PLC Header.

RFC 1006

Aby stworzyć komunikację globalną z dostępnymi funkcjami routing-u, powstały standardy programowe, umożliwiające osadzenie aplikacji OSI na warstwach transportowych TCP/IP. Dzięki takiej opcji powstało tunelowanie OSI-Telegramów poprzez połączenia TCP/IP. Najbardziej rozpowszechniona jest aplikacja oparta na specyfikacji RFC 1006.

Pełna nazwa RFC 1006 brzmi „ISO Transport Service on the top of the TCP", co oznacza, że usługi protokołu H1 są osadzone na komunikacji TCP/IP. Poprzez RFC 1006 komunikują się programy, które oparte są na strukturze blokowej H1, osadzonej na połączeniowo zorientowanym protokole TCP/IP. RFC 1006 korzysta zawsze z portu 102. RFC 1006 w automatyce wykorzystywany jest tylko przez firmę Siemens, której ułatwiło to przeniesienie już uprzednio wypracowanych protokołów, przy niewielkich nakładach pracy.
RFC 1006 w przeciwieństwie do INAT SPS-Header musi być zaimplementowany programowo lub sprzętowo jako specyficzne „rozszerzenie" protokołu TCP/IP we wszystkich stacjach danej sieci.

2.4.7 Protokoły reguł doboru tras i transmisji szeregowej [N-1]

W sieciach TCP/IP router-y (gateway-e) spełniają ważną rolę w zakresie kierowania ruchem Datagram-ów. Ruch ten może odbywać się zarówno wewnątrz sieci jak i dotyczyć wymiany informacji między różnymi sieciami. Ponieważ protokół IP nie określa sposobu kierowania ruchem wewnątrz sieci i między sieciami opracowano dla tych celów różne protokoły reguł doboru tras. Protokoły te mają za zadanie przede wszystkim przygotować informacje niezbędne do budowy tablic kierunków w router-ach (gateway-ach).
Przykładami protokołów wewnętrznych reguł doboru tras są protokoły RIP i OSPF.

2.4.7.1 RIP (Routing Information Protocol) [N-1]

Protokół ten zaliczamy do kategorii protokołów dystansowo-wektorowych. Protokół ten zwykle wybiera trasy o najmniejszej liczbie "przeskoków", czyli najmniejszej liczbie router-ów (węzłów), przez które muszą przejść Datagram-y na trasie od router-a źródłowego do docelowego. Najdłuższa trasa może składać się z co najwyżej piętnastu przeskoków. Jeżeli wyznaczona trasa posiada więcej niż piętnaście przeskoków to protokół RIP przyjmuje, że router docelowy jest nieosiągalny. Z tego powodu protokół ten nie może być stosowany w systemach autonomicznych składających się z dużej liczby router-ów.

2.4.7.2 OSPF (Open Shortest-Path-First) [N-1]

Protokół OSPF zaliczymy do protokołów stanu połączenia. W porównaniu z protokołami dystansowo-wektorowymi protokoły stanu połączenia wymagają większej mocy obliczeniowej, zapewniają większy stopień kontroli nad procesem kierowania ruchem Datagram-ów w sieci i szybciej dostosowują się do zmian struktury sieci. Protokół OSPF jest przystosowany do pracy w dużych systemach autonomicznych.

2.4.7.3 EGP (Exterior Gateway Protocol) [N-1]

Protokół EGP umożliwia wymianę komunikatów między parą sąsiednich router-ów zewnętrznych. Router zewnętrzny to taki sam router, który z jednej strony ma możliwość komunikowania się z innymi router-ami wewnątrz systemu autonomicznego, a z drugiej z router-ami zewnętrznymi innych systemów autonomicznych. System autonomiczny może posiadać jeden lub wiele router-ów zewnętrznych.

Każdy router zewnętrzny wymienia informacje związane z wewnętrzną regułą doboru tras z router-ami wewnętrznymi systemu autonomicznego, korzystającymi z protokołu wewnętrznej reguły doboru tras. Pozwala to router-owi zewnętrznemu na uzyskanie informacji o adresach komputerów (użytkowników końcowych), znajdujących się w systemie autonomicznym.

2.4.7.4 BGP (Border Gateway Protocol) [N-1]

Protokół BGP zdefiniowany w dokumencie RFC 1163 zaczyna zastępować protokół EGP. Router-y zewnętrzne pracujące z protokołem BGP, podobnie jak router-y z protokołem EGP, wymieniają informację o dostępności systemów autonomicznych. Ponadto przesyłane są atrybuty trasy takie jak koszty czy też zabezpieczenia przed niepowołanym dostępem. Atrybuty te również mogą zawierać informacje służące do wyboru tras na podstawie wymagań administracyjnych (nietechnicznych), np. związanych z bezpieczeństwem Datagram-ów. Na podstawie otrzymanych informacji protokół BGP wybiera najkrótszą trasę. Informacje wymieniane są jedynie przyrostowo, a nie przez przesyłanie całej bazy danych dotyczącej zewnętrznej reguły doboru tras; zatem protokół ten nie powoduje dużego przyrostu ruchu w sieci.

Protokoły TCP/IP mogą działać korzystając z wielu różnych mediów transmisyjnych. Jednym z istotniejszych nośników są łącza szeregowe z uwagi na to, że wielu zdalnych użytkowników łączy się z sieciami TCP/IP poprzez np. łącza telefoniczne, a także z uwagi na rozwój sieci rozległych pracujących z protokołami TCP/IP. Te dwa powody wymusiły standaryzację komunikacji TCP/IP poprzez łącza szeregowe, co doprowadziło do powstania dwóch protokołów dla łączy szeregowych SLIP i PPP.

2.4.7.5 SLIP (Serial Line IP) [N-1]

Protokół ten został opisany w dokumencie RFC 1055. Umożliwia on asynchroniczny lub synchroniczny transfer danych przez łącza dzierżawione lub komutowane z szybkością transmisji do 19.2 Kb/s. Pozwala łączyć ze sobą komputery, router-y i stacje robocze. Protokół SLIP w prosty sposób obudowuje Datagram-y IP podczas ich przesyłania przez łącza szeregowe. SLIP traktuje dane jako ciąg bajtów i używa następujących dwóch znaków specjalnych do oznaczania końca Datagram-u:

- ✓ Znak SLIP END (kod 192) oznacza koniec Datagram-u.
- ✓ Znak SLIP ESC (kod 219) wskazujący, że następny znak nie jest znakiem specjalnym protokołu SLIP.

W trakcie transmisji może zdarzyć się, że w nadawanym ciągu danych wystąpią sekwencje odpowiadające znakom specjalnym, co oznacza błędną ich interpretację przez odbiornik. Aby się przed tym zabezpieczyć, nadajnik bada wysyłany ciąg bajtów i wstawia dodatkowy znak ESC bezpośrednio przed bajtem odpowiadającym znakowi specjalnemu. Pozwala to zapobiec interpretowaniu przez protokół SLIP bajtu stanowiącego dane jako końca Datagram-u.

Protokół SLIP może przekazywać Datagram-y o długości do 1006 bajtów. Nie zawiera on w sobie mechanizmów detekcji i korekcji błędów, a także nie posiada mechanizmów adresowania. Oba komunikujące się systemy muszą znać wzajemnie swoje adresy i mogą przesyłać z użyciem protokołu SLIP wyłącznie Datagram-y IP. Protokół ten może być wykorzystywany jedynie w transmisji ‘punkt-punkt”. Ponadto komunikujące się systemy muszą mieć zainstalowane te same wersje protokołu SLIP.

Wymienione wyżej braki protokołu SLIP nie są istotne dla części zastosowań, a w innych zastosowaniach stanowią wielką przeszkodę. SLIP zaleca się stosować do odległych systemów, komputerów lub stacji roboczych przesyłających wyłącznie Datagram-y IP, nie zaleca się go stosować w środowisku sieci rozległych do łączenia router-ów.

2.4.7.6 PPP (Point to Point Protocol) [N-1]

Protokół PPP opracowano jako standard przeznaczony do użycia w sieci Internet. Można go również stosować w innych sieciach rozległych (RFC 1171 i 1172). Jest to protokół do transmisji synchronicznej i asynchronicznej po łączach dzierżawionych i komutowanych. Protokół PPP może przenosić pakiety pochodzące od różnych protokołów warstwy sieciowej(IP, IPX, AppleTalk, DECnet, CLNP oraz MAC). Inną właściwością PPP jest posiadanie mechanizmu adresacji IP, co pozwala zdalnym użytkownikom łączyć się w dowolnym miejscu sieci, a także ograniczeń co do szybkości transmisji. Datagram-y IP lub pakiety innych protokołów są przesyłane wewnątrz ramek protokołu PPP. Struktura tej ramki jest podobna do struktury ramki HDLC z tym, że ramka PPP ma dodatkowe pole określające od jakiego protokołu pochodzą dane zawarte w polu informacyjnym ramki. PPP obejmuje swym działaniem trzy najniższe warstwy modelu ISO/OSI.

3. Analiza sieci

Sieci przemysłowe oparte o standard Ethernet są bardzo zbliżone do zwykłych sieci lokalnych lecz szczególnymi ich cechami są:

- ✓ Duża ilość małych pakietów danych (pojedyncze bity z czujników i maszyn).
- ✓ Stosunkowo stałe obciążenie sieci.

3.1. Fizyczne granice przy analizie sieci

Podczas przeprowadzania analiz sieciowych, podłączając urządzenia monitorujące sieć Ethernet trzeba być świadomym tego, jak zbudowana jest dana sieć oraz jak ogólnie funkcjonuje w niej wymiana informacji. Można bowiem popełnić wiele błędów, nie znając podstaw funkcjonowania urządzeń sieciowych lub umieszczając analizatory w nieodpowiednich punktach sieci.

Switch-e cechują się dużą szybkością przełączania oraz ograniczają domenę kolizyjną do pojedynczego portu, dzięki czemu są w stanie zapewnić każdemu host-owi, podłączonemu do portu, osobny kanał transmisyjno-nadawczy, a nie współdzielony, tak jak koncentratory [N4]. Ostatnia funkcja jest jednocześnie wadą i zaletą switch-y, gdyż następuje odseparowanie kanałów i zwiększenie prędkości, ale chcąc analizować komunikację pomiędzy stacjami podpiętymi do takiego „przełącznika" jesteśmy zmuszeni do zakupu drogiego switch-a wyposażonego w port monitorujący, który pozwala na podsłuchiwanie wybranego portu.

Hub-y to przełączniki, które kopiują każdorazowo otrzymane informacje z portu wejściowego na pozostałe porty. Takie rozwiązanie doskonale nadaje się do monitorowania sieci, gdyż „podpinając się" z analizatorem sieci do dowolnego portu możemy obserwować całą wymianę informacji bez konieczności kupowania drogiego sprzętu.

Innymi ważnymi aspektami przy analizie sieci jest ilość danych oraz szybkości odczytu, buforowania, przesyłu i zapisu zgromadzonych danych. W komunikacji bardzo często dochodzi się do krytycznej granicy, jaką jest prędkość sygnału (prędkość światła) i droga, jaką może dany sygnał pokonać. Należy wiedzieć, iż nie jest możliwe jednoczesne monitorowanie kilkunastu portów komunikujących się z prędkością 100Mb/s, gdyż szybko

przekracza się prędkość 1Gb/s, a takiej ilości danych nie jest się w stanie na przeciętnych komputerach przetworzyć.

3.2. Przygotowanie, przesył i odbiór telegramów [N4]

Każda informacja, którą chcemy przesłać od stacji „A" do stacji „B", korzystając z sieci musi pokonać pewne etapy na swojej drodze.
Począwszy od aplikacji poprzez kartę sieciową, medium transportowe (np.: kabel, światłowód) i elementy pośredniczące (np.: switch-e, bramy) dane trafiają do karty sieciowej stacji docelowej, gdzie kierowane są do wyższych aplikacji.

Kolejność w jakiej informacje są przesyłane od aplikacji pracującej w CPU (Central Procesor Unit) – w komputerze, poprzez CP (Communication Processor) – kartę sieciową stacji „A" do stacji „B" opisują poniższe podpunkty:

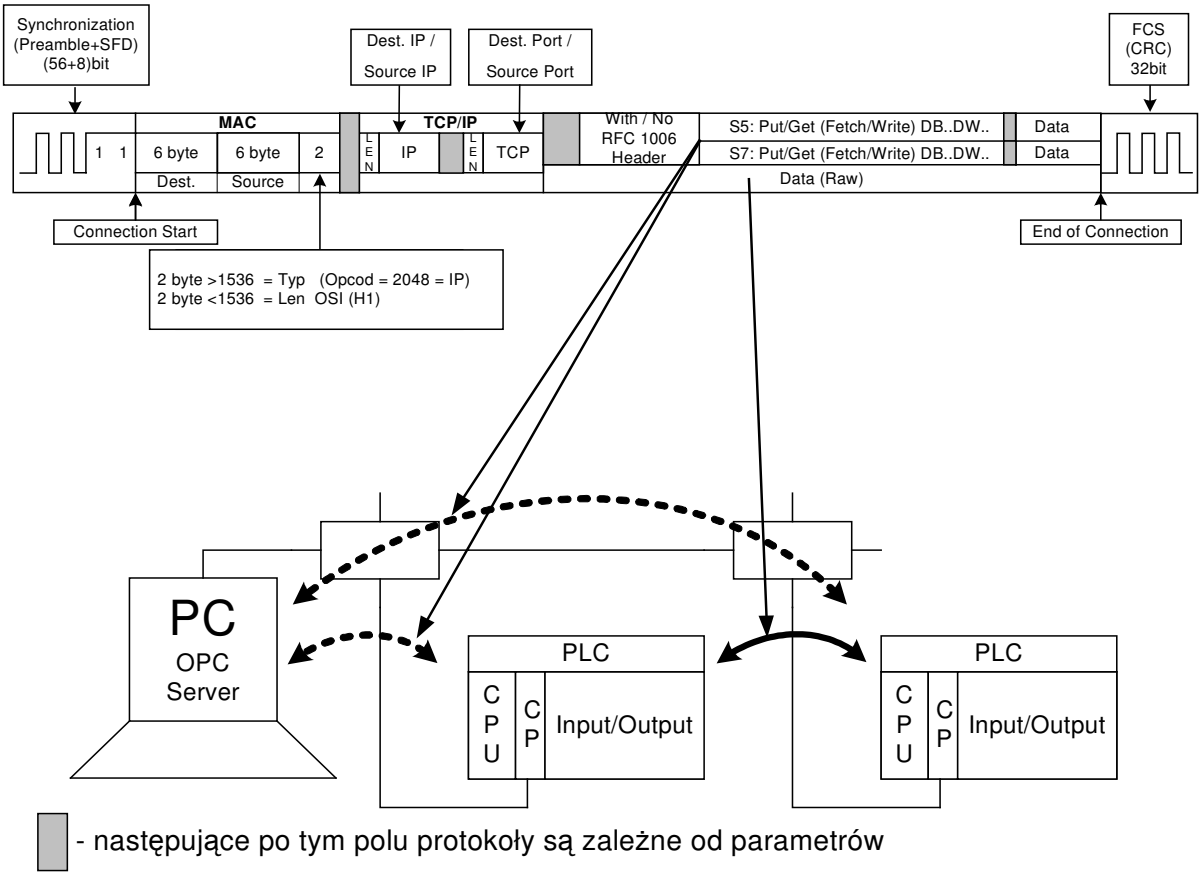
- ✓ Określenie protokołu, za pomocą którego dane zostaną wysłane (PLC Protocol, TCP/IP).
- ✓ Przekazanie danych z pamięci komputera (bieżącej aplikacji) do karty sieciowej lub z karty sieciowej do pamięci komputera, z wykorzystaniem kanału DMA, pamięci dzielonej lub programowalnych układów (we/wy). Powyższy krok ma charakter „pakowania" danych, począwszy od warstwy 7 modelu ISO/OSI (więcej w punkcie 2.3.2), aż do warstwy pierwszej dane są umieszczane w protokołach, a te otrzymują kolejno nagłówki.
- ✓ Przetwarzanie danych przez kartę sieciową, podczas którego następuje ich buforowanie (bufor pozwala na przekazywanie całych pakietów danych i dostosowanie prędkości transmisji danych w sieci do prędkości ich przetwarzania przez komputer).
- ✓ Formowanie pakietów danych, polegające na dzieleniu strumienia danych przeznaczonych do wysłania na pakiety odpowiedniej wielkości (karta na początku każdego pakietu dodaje nagłówek, a na jego końcu stopkę - CRC, co stanowi swoistego rodzaju „kopertę" dla przesyłanego pakietu danych), taki pakiet gotowy jest do wysłania.
- ✓ Zbudowanie połączenia (Three Hand Shake w TCP/IP).
- ✓ Przesłanie telegramu.
- ✓ Odebranie telegramu przez docelową kartę sieciową (w procesie odbioru nagłówek i stopka są usuwane, a otrzymane pakiety łączone są w jedną całość).

- ✓ Następnie sprawdzenie, czy wszystkie pakiety, dane dotarły (Powtórne telegramy zostają odrzucone, części telegramów są łączone razem.).
- ✓ Pokwitowanie danych przesłanych poprzez TCP/IP.
- ✓ Dekodowanie telegramów (PLC Protocol).
- ✓ Przekazanie danych do aplikacji.

Gdy ilość danych jest duża to również w warstwie aplikacji są one przesyłane we fragmentach, aby zaoszczędzić na czasie przesyłu i pamięci. W automatyce duże fragmenty danych występują bardzo rzadko, zazwyczaj aplikacje przetwarzają dane na małe fragmenty.

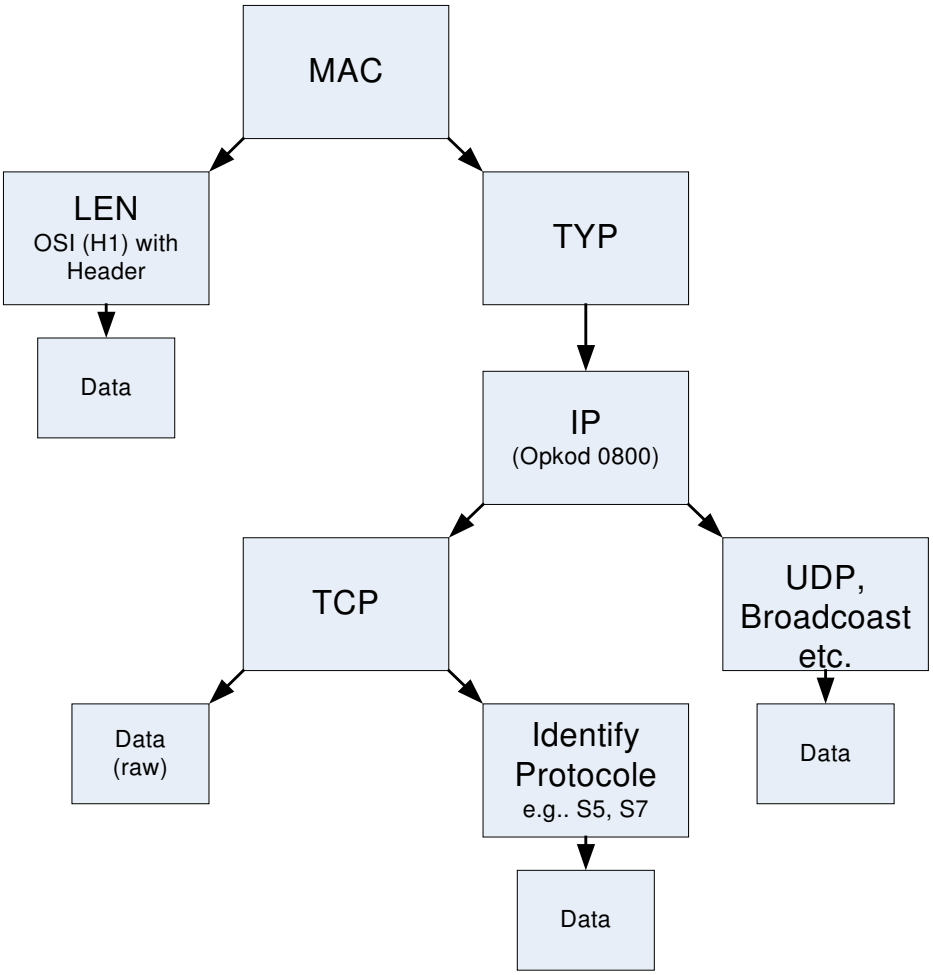
3.3. Dekodowanie telegramu

Rozpoznawanie dużej ilości telegramów odbywa się bardzo szybko, jednak w zależności od jakości karty sieciowej, proces ten zajmuje od kilku do kilkudziesięciu procent zasobów procesora. Szczegółowa budowa telegramu Ethernet-owego wraz z elementami pozwalającymi kartom sieciowym na poprawne jego wysłanie, rozpoznanie i przekazanie (synchronizacja, FCS) do wyższych warstw modelu ISO/OSI przedstawiona została na poniższym schemacie (Rysunek 14).



Rysunek 14. Telegram Ethernet-owy a różne formy przesyłanych danych.

Telegramy przesyłane w sieciach muszą być odpowiednio zsynchronizowane i wysyłane przez stację nadawczą w odpowiednim czasie, aby nie występowały kolizje (nakładanie się sygnałów, CSMA/CD – szerzej w punkcie 2.3.1). Telegram „wędrujący" w sieci w kierunku stacji docelowej jest przejmowany po synchronizacji przez podłączoną do docelowej stacji kartę sieciową. Tam, na poziomie sprzętowym, następuje sprawdzenie sumy kontrolnej CRC- gdy suma się zgadza (często ustawiana jest na „0"), to telegram zostaje przekazany do wyższej warstwy, w przeciwnym razie wysyłany jest rozkaz o ponowne jego przesłanie. Kolejnym etapem w dekodowaniu telegramu jest rozpoznanie w nagłówku Ethernet-owym typu protokołu (2 bajty). Zgodnie z przedstawionym poniżej schematem (Rysunek 15), może to być albo „LEN" (długość), albo „Type" (typ – opcode (opkod)). Jeśli wartość zapisana w postaci 2 bajtów jest większa niż 1536, to mamy do czynienia z Typem (opkod, np. opkod = 0x0800 Hex – 2048 Dec oznacza IP), jeżeli jednak wartość jest mniejsza od 1536, to mamy do czynienia z długością (OSI-H1). Te dwubajtowe pole należy rozumieć tak, że do 1536 (maksymalna długość telegramu Ethernet-owego), przedstawia wartość, a numery powyżej 1536 stanowią symbol.



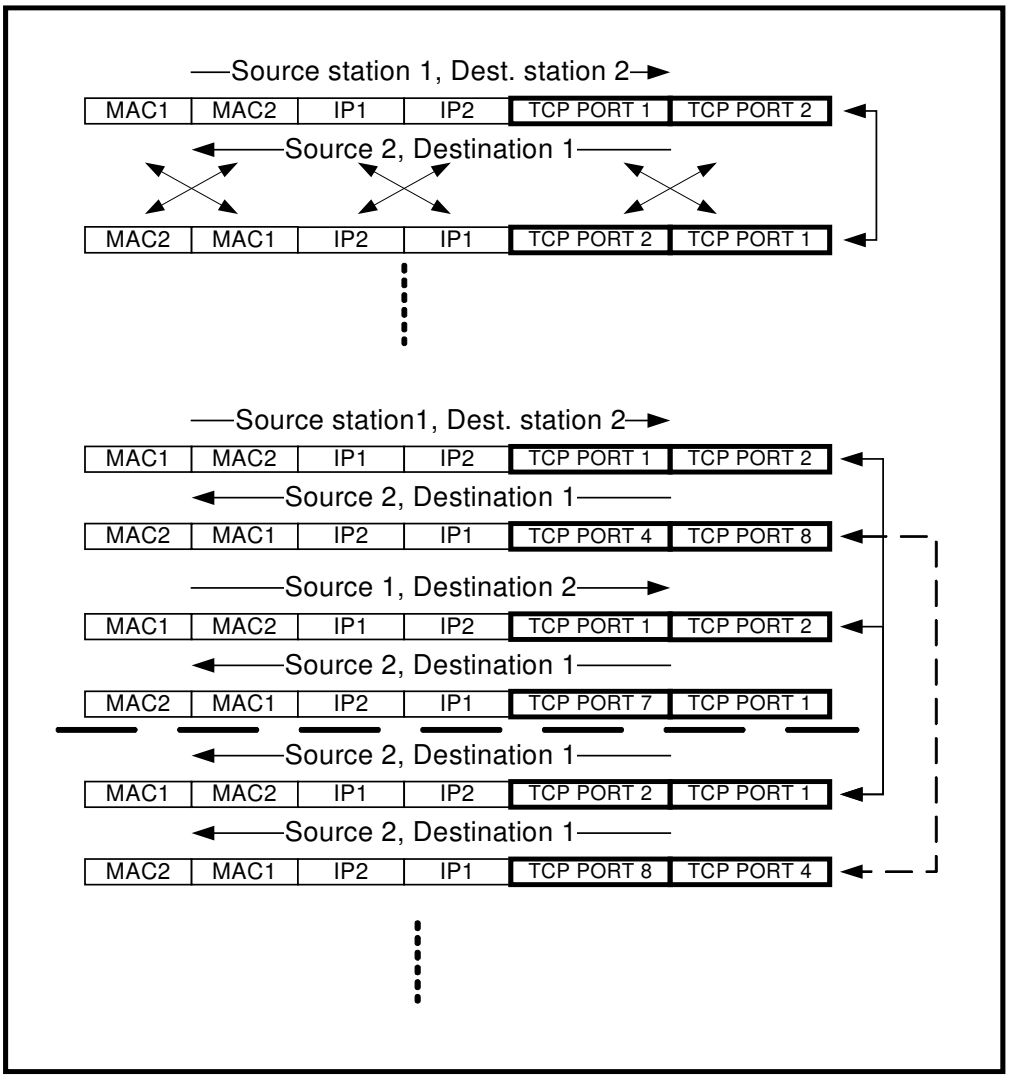
Rysunek 15. Kolejność kroków podczas dekodowania telegramu Ethernet-owego.

W przypadku protokołu OSI – H1 mamy do czynienia z danymi „raw" („surowymi"), wykorzystywanymi np. w protokołach PLC firmy Siemens. Mając do czynienia z opkodem, odpowiadającym protokołowi IP, dekodowanie przechodzi do nagłówka protokołu IP, gdzie pod pozycją „Protocol" jest zdefiniowany zagnieżdżony w IP kolejny protokół. I tak np. zapisana w postaci 8 bitów wartość 6, zapisana heksadecymalnie (0006H) odpowiada protokołowi TCP. Na protokole IP bazuje więcej protokołów, takich jak: UDP, Broadcast i inne. W zależności od portów, jakie są zapisane w chwilowym nagłówku TCP telegramu TCP/IP, jesteśmy (lub nie) w stanie określić jaki nastąpi po nagłówku TCP protokół, np. dla portu 102 mamy w branży automatyki przemysłowej do czynienia z protokołami firmy Siemens (port 80 – HTTP, 20/21 - FTP, 23 – Telnet itd.). Po rozpoznaniu portu, wiedząc jednocześnie jakiego protokołu spodziewamy się po TCP, można rozpocząć odpowiednie dekodowanie kolejnego protokołu, jakim może być np. S5 lub S7, które to dopiero będą

zawierały istotne dane dla sterownika lub innej stacji w sieci. Może być jednak tak, że TCP będzie ostatnim protokołem dla danego połączenia, a dalej umieszczone będą dane „raw".

3.4. Rozpoznanie połączenia

Aby można było przystąpić do analizy sieci Ethernet należy uprzednio „przyjrzeć się" transferowi sieciowemu, co pomoże określić połączenia pomiędzy poszczególnymi stacjami. Karta sieciowa po otrzymaniu telegramu, informuje procesor, aby ten przystąpił do pobierania informacji z bufora karty. Następnie telegramy są analizowane w aplikacji, następuje odczytywanie adresów fizycznych MAC stacji źródłowych i docelowych, kolejno sprawdzany jest protokół IP (adresy IP) i TCP (porty). Mając do dyspozycji adresy MAC, adresy IP i znając numery portów stacji odbiorczej i stacji źródłowej telegramu otrzymanego i telegramu wysyłanego jesteśmy w stanie określić czy mamy do czynienia z jednym, czy też z wieloma połączeniami (Rysunek 16).



Rysunek 16. Idea rozpoznawania telegramów i przyporządkowywania połączeń

W górnej części powyższego rysunku zauważamy, że dane telegramu wysłanego ze stacji (1) do stacji (2) z portu (1) trafiają do portu (2) stacji docelowej. Mówimy, że doszło do połączenia, jeżeli stacja (2) wyśle odpowiedź z danymi do portu (1) stacji (1) pochodzącymi od portu, do którego uprzednio trafiły dane (TCP port 2). Następuje wtedy „krzyżowanie się" portów. W drugiej, częściej spotykanej części (Rysunek 16), komunikacja przebiega pomiędzy większą ilością stacji (trzy stacje) oraz zostało zbudowanych więcej połączeń. Mając te same adresy MAC i IP można stworzyć wiele połączeń, w których to odpowiedzi na przesłane dane niekoniecznie są potwierdzane kolejnym telegramem od stacji docelowej.

Potwierdzenie otrzymania danych, może nastąpić w jednym z kolejnych telegramów i kwitować jednocześnie dane otrzymane w kilku poprzednich telegramach.

3.5. Dekodowanie wzajemnie powiązanych telegramów (stream)

Dane przesłane przez sieć ze sterownikami programowalnymi (PLC) zazwyczaj mieszczą się w jednym telegramie i jeśli sterowniki te znajdują się w jednej podsieci, to nie następuje dzielenie telegramów. W przypadku, gdy wymiana danych, pochodzących od sterowników jednej sieci do stacji znajdującej się poza tą siecią przebiega poprzez router, to może dochodzić do dzielenia telegramów. Podobna sytuacja dzielenia telegramów ma miejsce, gdy dany sterownik chce przesłać większą ilość danych, a ta przewyższa wielkość telegramu dla danego protokołu (np. RFC 1006, AS 511 itp.). Komunikacja TCP/IP zapewnia to, że wszystkie telegramy dojdą do celu oraz będą miały zachowaną kolejność.

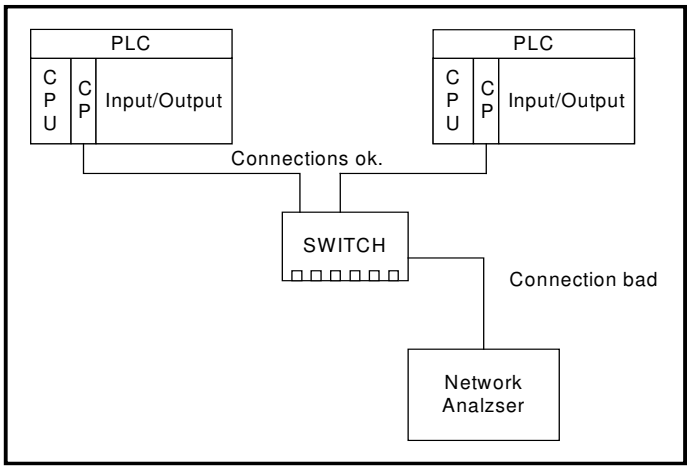
Trudność pojawia się wtedy, gdy zamiast jednego telegramu otrzymujemy kilka, a dane w nich zawarte trzeba połączyć w jedność, aby móc je rozpoznać. Aby operacja rozpoznania i przyporządkowania danych była możliwa, aplikacje wysyłające i odbierające muszą posługiwać się odpowiednim protokołem, będącym otwartym standardem (np. Modbus) lub protokołem firmowym (np. RFC 1006). Ten specjalny protokół osadzony w polu danych protokołu TCP, informuje aplikację pobierającą i sortującą dane, czy dane „wypakowane" z danego telegramu są „pełne", czy też w kolejnym protokole nastąpi dalsza część powiązanych ze sobą danych.

3.6. Granice w rozpoznawaniu danych

Granicami w rozpoznawaniu danych jest znajomość budowy i funkcjonowania specyficznych protokołów umieszczonych ponad protokołem TCP oraz natężenie wymiany informacji pomiędzy stacjami. Aby można było przeprowadzić sensowną analizę sieci w urządzeniu obserwującym sieć przez kilka tygodni, a nawet miesięcy konieczne jest wprowadzenie pewnych założeń i uproszczeń, które jednak zapewniają maksymalnie możliwą pewność analizy. Przykładowym ograniczeniem jest rozpoznanie i nie rejestrowanie ewentualnej komunikacji pomiędzy komputerami PC, gdy nie ma ona negatywnego wpływu na całą sieć. Innym ważnym założeniem, które trzeba zrealizować jest rejestracja tylko najistotniejszych informacji, gdyż ilość danych jaka przesyłana jest przez sieć w przeciągu dłuższego okresu jest zazwyczaj ogromna i niemożliwe byłoby przeanalizowanie każdego telegramu z osobna, aby dojść do przyczyny zakłócenia, a i największe dyski twarde byłyby zbyt małe.

Kolejnym uproszczeniem jest filtracja odpowiednich portów. Należy wziąć pod uwagę jeszcze inne aspekty aby możliwie maksymalnie zoptymalizować analizę sieci. Innym problemem w analizie sieci jest nie udostępnianie protokołów komunikacyjnych przez niektórych producentów sterowników programowalnych, co uniemożliwia wielu firmom rozpoznanie danych i wykorzystanie ich np. do łączenia komunikacji ze sterownikami o innych protokołach. W automatyce przemysłowej komunikacja sieciowa jest oparta o małe pakiety danych, przesyłane szybko przez sieć, gdzie każdy „bit” jest bardzo ważny, toteż stacje muszą być sprzętowo (procesor, bufor) odpowiednio przystosowane. Niedopuszczalne jest przesyłanie zbyt szybko dużych telegramów, gdyż zarówno procesory, jak i bufory poszczególnych stacji mogą powodować utratę telegramów.

Kolejnym ważnym zagadnieniem jest ingerencja urządzenia nasłuchującego w istniejącą sieć. Występuje tu szereg problemów związanych z szybkościami całej sieci lub poszczególnych gałęzi sieci. Wprowadzając do nowoczesnej sieci urządzenia sprzed kilku lat, które teoretycznie odpowiadają żądanej prędkości, często mamy do czynienia z konfliktami i błędami, które występują bardzo przypadkowo i w nieokreślonych odstępach czasowych. Bardzo ważnym zagadnieniem jest, tak jak zresztą, we wszystkich dziedzinach techniki, poprawność samego pomiaru i umiejętne oraz racjonalne „potraktowanie” zarejestrowanego błędu.



Rysunek 17. Uwzględnienie możliwości wystąpienia błędu na drodze pomiaru.

Luzy na stykach w kablach sieciowych, wpływ innych zdarzeń lub sama jakość i parametry kabla podpiętego do analizatora mogą być powodami występowania (rejestrowania) błędów. Błędy takie należy w odpowiedni sposób umieć programowo zinterpretować i nie zarejestrować jako błąd. Aby móc odpowiednio radzić sobie z nimi konieczne jest zastosowanie analiz statystycznych. Implementacja takiego algorytmu nie wchodzi w zakres realizowanego w ramach mojej pracy dyplomowej programu.

4. Rozpoznawanie telegramów - dekodowanie

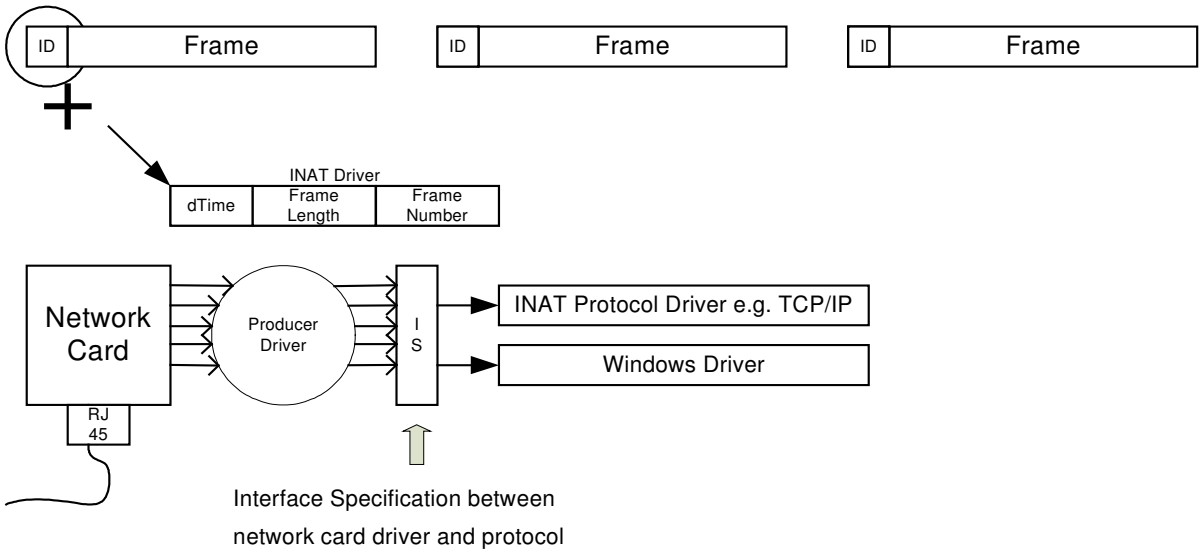
Do rozpoznawania telegramów przesyłanych przez sieć Ethernet-ową potrzebna jest karta sieciowa. Każda karta sieciowa musi być dopasowana do danego systemu operacyjnego (programu), który pobiera, przetwarza i przesyła dalej dane. Przygotowanie danych ma miejsce w sterowniku karty dostarczonym przez producenta, zawierającym informacje sprzętowe i zarządzającym częścią sprzętową („chipem") oraz potrafiącym współpracować z aplikacją, która sięga po te dane. Ze względu na konkurencję rynkową i ciągły, dynamiczny rozwój branży IT (Information Technology) konieczna jest nieustanna praca inżynierów opracowujących rozwiązania komunikacyjne kart sieciowych i dopasowujących je do rozwiązań sprzętowych i programowych.

W projekcie „Echocheck" konieczne jest przekazywanie wraz z danymi w postaci telegramów również innych parametrów, które pozwolą na sensowną analizę i zapis danych. Karta sieciowa może funkcjonować jako wewnętrzny lub zewnętrzny osprzęt, jak również może być zintegrowana na płycie głównej komputera (np. PC104). Po dostarczeniu telegramu do aplikacji (programu analizującego sieć) dokonywane jest rozpoznanie typu telegramu, analizowane są nagłówki poszczególnych protokołów, następuje przyporządkowanie połączenia oraz podejmowana jest próba analizy danych, która w zależności od ich charakteru, ilości i parametrów czasowych, kończy się zapisaniem ważnych informacji dla zarządzającego siecią lub ich odrzuceniem.

4.1. Sterownik – Driver

Sterownik karty sieciowej ma za zadanie połączyć część sprzętową sieci z interfejsem (IS – Interface Specification), będącym sprzęgiem pomiędzy sterownikiem karty sieciowej a warstwą aplikacji (Rysunek 18). Sprzęg ten w firmie Microsoft nosi nazwę NDIS (Network Driver Interface Specification) i tworzy standard API (Application Programming Interface) dla NIC (Network Interface Card), czyli sprzęg ten jest standardem programowym dla kart sieciowych. NDIS dostarcza biblioteki funkcji, używanych przez sterowniki zarówno dla warstwy MAC (Media Akcess Control), jak również dla wyższych warstw protokołów np. TCP/IP. Firma Microsoft nie jest jedynym producentem oprogramowania na świecie, toteż używanie skrótu NDIS dotyczy wyłącznie aplikacji tegoż producenta. Jak wiadomo producenci kart sieciowych i komponentów sieciowych nie są zależni od wyżej wspomnianej firmy, co sprawia, że interfejs karty sieciowej (IS) na bardzo podobnych zasadach jest wykorzystywany przez inne systemy operacyjne, jak np. systemy „UNIX-owe".

Z tego względu stosowanie skrótu NDIS w ogólnym przypadku jest niepoprawne.



Rysunek 18. Przesyłanie do aplikacji telegramów oraz parametrów sieciowych.

Aby możliwe było pobieranie telegramów oraz potrzebnych parametrów z interfejsu każdej karty sieciowej niezależnie od producenta ani systemu operacyjnego i wykorzystywanie ich dla potrzeb własnych aplikacji, firma INAT stworzyła własny sterownik. Protocol Driver firmy INAT (ID) składa się z programu obsługującego kartę sieciową oraz szeregu bibliotek, pozwalających zachować niezależność systemową. Wykorzystany w projekcie „EchoCheck" sterownik protokołów dostarcza do aplikacji m.in. takie parametry jak: czas, rozmiar telegramu (DLC) oraz numer telegramu. Ze względu na oszczędność zasobów obliczeniowych, do aplikacji często przekazywany jest pełny format czasu (data + czas) jedynie przy rozpoczęciu komunikacji, a czasy kolejnych telegramów obliczane są już tylko na podstawie przyrostów czasowych (dTime). Te charakterystyczne dane przypisywane są każdorazowo wysyłanemu telegramowi (Rysunek 18 - ID). Telegramy są numerowane od „0" (lub od przypisanej wartości) w górę, aż do numeru przekraczającego liczbę 32 bitów (unsigned long) lub do ponownego wystartowania aplikacji. Po przekroczeniu maksymalnej wartości telegramy są numerowane od 0.

4.2. Zestawienie danych dostarczonych przez sterownik do warstwy aplikacji

Aplikacja (program) sterownika firmy INAT przygotowuje dane każdego telegramu każdorazowo z nadejściem nowego pakietu telegramów, umieszczając je w pamięci komputera (np. PC 104). Lista danych, które zawiera każdy telegram TCP/IP jest przedstawiona w poniższej tabeli (Tabela 5).

Ethernet telegram (IP v4)			
Nr.	**Bit**	**Parameter**	**Protocol**
	16	DLC (Frame Size)	
1	48	MAC (ZIEL)	**Ethernet**
2	48	MAC (QUELL)	
3	16	KEN.(LEN/TYP)	
	112	14 Byte	
4	4	IP VERSION	**IP**
5	4	IHL (Header Length)	
6	8	Time of service	
7	16	TOTAL LENGTH	
8	16	Identification	
9	4	Flags	
10	12	Fragment Offset	
11	8	Time To Live	
12	8	PROTOCOL	
13	16	Header Checksum	
14	32	IP SOURCE ADDRESS	
15	32	IP DESTINATION ADDRESS	
16	0	Options (0…32) Bit	
17	0	Padding (0…32)Bit	
	160	20 Byte	
18	16	TCP SOURCE PORT	**TCP**
19	16	TCP DESTINATION PORT	
20	32	SEQUENCE NUMBER	
21	32	ACKNOWLEDGMENT NUMBER	
22	4	DATA OFFSET	
23	12	Reserved	
24	16	Window	
25	16	Checksum	
26	16	Urgent Pointer	
27	0	Options (0...24)	
28	0	Padding (0...8)	
	160	20 Byte	
29	xxx	Data (need)	**Data**
30	xxx	Data (refilling) if Data (need)<64 Byte	
	Max	1500Byte (1534)	

Tabela 5. Zestawienie parametrów telegramu przesyłanych do aplikacji.

Pamięć, gdzie umieszczona jest większość powyższych danych, jest odwzorowana strukturą „REFERENCE LIST", umieszczoną w bibliotece „displaytcp.h". Jest to biblioteka uniwersalna (*.h), dostarczająca informacji o nagłówku telegramu, jak również innych parametrach (Tabela 6).

```
typedef struct _REFERENCE_LIST {
        unsigned long Number;               // Position
        unsigned char QuellStation[6];      // Sender (MAC)
        unsigned long OwnIpAdr;             // Eigene IP Adresse
        unsigned short OwnPort;             // OwnRef und DestRef = 0 -> Eintrag leer
        unsigned char ZielStation[6];       // Ziel
        unsigned long DestIpAdr;            // Fremde IP Adresse
        unsigned short DestPort;
        unsigned short Marked;              // Markiert für Filter
        unsigned short MayOld;              // Beim Optimieren
        unsigned long SendBytes;            // Bearbeitete Daten
        unsigned long RecBytes;             // Bearbeitete Daten
        unsigned long Seq;                  // Ip Sequence Number
        unsigned long Ack;                  // Ip Ack Number NEW
        unsigned char TempSendData[IP_TEMP_DATALEN];
        unsigned char TempRecData[IP_TEMP_DATALEN];
        unsigned short SendDataLen;         // Belegter Inhalt in TempSendData
        unsigned short RecDataLen;          // Belegter Inhalt in TempRecData
        unsigned char LenOwnTSAP;           // if RFC 1006
        unsigned char OwnTSAP[64];          // if RFC 1006
        unsigned char LenDestTSAP;          // if RFC 1006
        unsigned char DestTSAP[64];         // if RFC 1006
        unsigned short IsNext;              // 0 if first
}REFERENCE_LIST;
```

Tabela 6. Jedna ze struktur biblioteki „displaytcp.h" – REFERENCE LIST

Powyższa struktura jest argumentem w dalszych podprogramach, gdzie wykorzystywane są jej pola składowe.

4.3. Rozpoznanie typu telegramu

Odczytanie obu adresów Ethernet-owych (MAC) oraz rozpoznanie typu telegramu ma miejsce w pliku źródłowym „DriverRecord.c" (pozycja 1,2 oraz 3 w tabeli 5). Wczytanie długości telegramu, który rozstrzyga o tym, czy mamy do czynienia z telegramem typu IP zrealizowane jest poprzez odczytanie z biblioteki „Layer2.h" zmiennej „DataLen" i porównanie jej z wartością 2047 (Tabela 7) zapisanej w postaci heksadecymalnej (0x7FF).

```
l2 = (DATEN_LAYER2 *)fe->Frame;
                if (MotorolaToIntel(l2->DataLen) >= 0x7ff)
```

Tabela 7. Sprawdzenie typu telegramu.

Projekt „EchoCheck" przewiduje również analizę telegramów nie należących do rodziny protokołów TCP/IP, które nie obejmują problematyki mojej pracy dyplomowej.

4.4. Rozpoznanie IP i określenie dołączonych danych

Aplikacja, w dalszej części rozkodowywania telegramu, korzysta z biblioteki „displayip.h" i zawartej w niej struktury „IP HEADER", która opisuje zmienne nagłówka IP. W zależności ile dana zmienna zajmuje miejsca w pamięci, odpowiednio musi być zdeklarowana np. „unsigned long AddressSource;"-zmiennej „AddressSource" przypisane są 32 bity w pamięci.

Struktura „IP HEADER" opisuje m.in. zmienną „Protocol", która informuje aplikację jaki protokół jest osadzony na protokole IP (Tabela 5, numer 12) . Dla TCP wartość ta wynosi 6.

Ważne są również inne parametry, takie jak wersja protokołu IP (Tabela 5, numer 4), która w sieciach przemysłowych do przesyłania stosunkowo małych telegramów jest wersją IP v4. Innymi ważnymi parametrami są: IHL, TOTAL LENGTH, IP SOURCE i DESTINATION ADRESS (kolejno numery 5, 7 14 i 15 w Tabeli 5).

4.5. Analiza danych zamieszczonych w protokole TCP

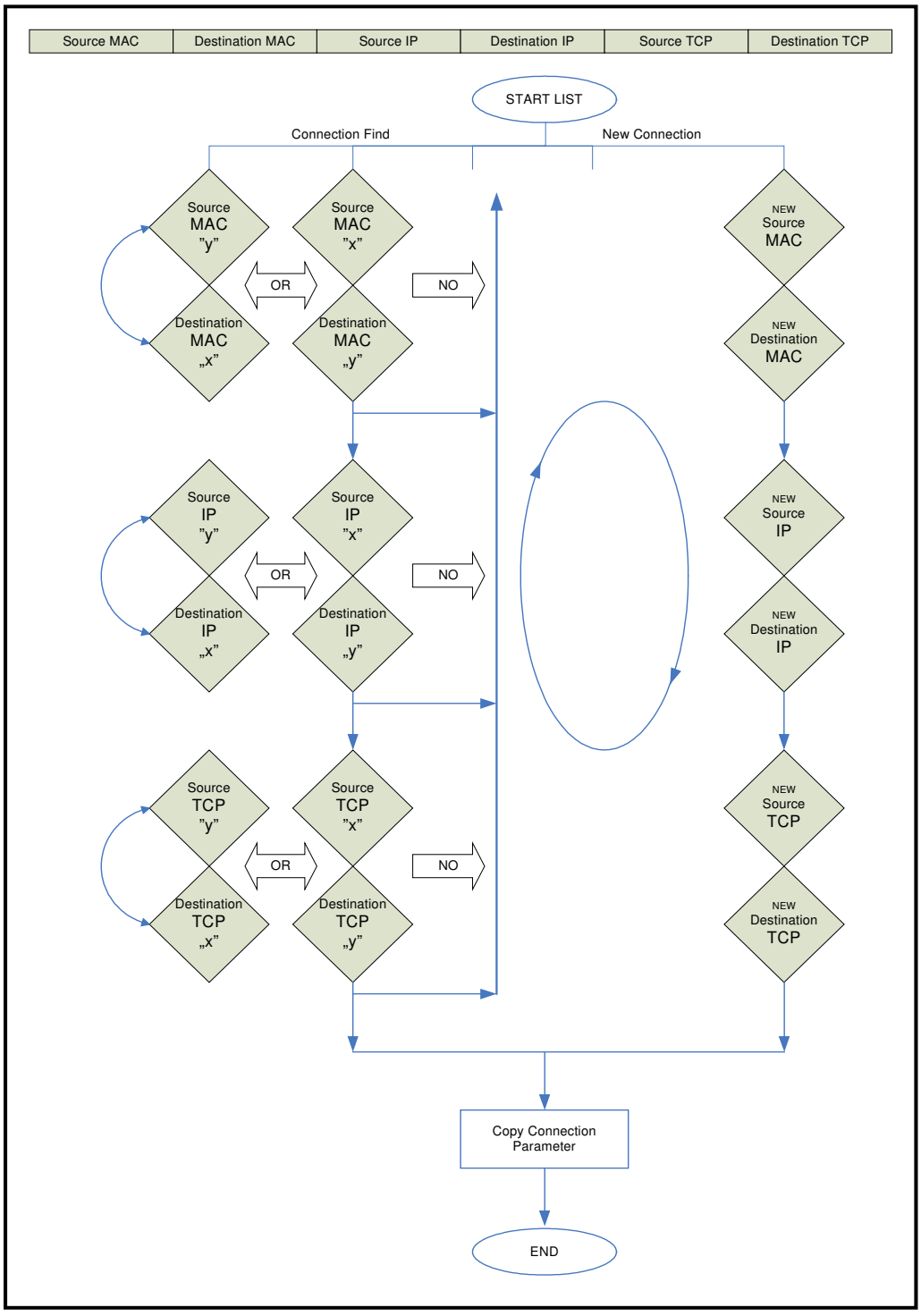
Parametry protokołu TCP opisane zostały w bibliotece „displaytcp.h" jako zmienne w strukturze „TCP HEADER". Struktura ta przekazywana jest do plików źródłowych „CalcIp.c" oraz pośrednio do pliku „DriverRecord.c", gdzie dokonywane są porównania, obliczenia i zapisy do pamięci. Ważnymi zmiennymi, z punktu widzenia projektu „EchoCheck", są: TCP SOURCE i DESTINATION PORT – (numer 18 i 19 w Tabeli 5), SEQUENCE i ACKNOWLADGE NUMBER – (numer 20 i 21 w Tabeli 5) oraz DATA OFFSET (numer 22 w Tabeli5). Inne parametry protokołu, choć też są ważne, nie będą brały bezpośredniego udziału w analizie i zapisie danych. Informacje pochodzące z pola DATA OFFSET pozwalają na określenie miejsca, od którego należy się spodziewać danych lub rozpocząć rozkodowywanie innego protokołu osadzonego na protokole TCP.

4.6. Filtrowanie połączeń

Ze względu na duży ruch w komunikacji sieciowej, niezbędne jest stosowanie różnego rodzaju filtrów na różnych poziomach aplikacji. Najważniejszym, a zarazem podstawowym filtrowaniem jest filtrowanie według adresów IP. Kolejno można filtrować ze względu na numery poszczególnych portów, którymi odbywa się komunikacja. Dodatkowym filtrem, który jednak jest nieco bardziej skomplikowany i wymaga większej mocy obliczeniowej, jest filtrowanie podług adresu MAC (należy uważać na sieci zawierające router-y). Możliwe jest bowiem, iż w sieci istnieje więcej stacji o takim samym numerze IP, wówczas konieczne jest filtrowanie na poziomie Ethernet-u (adresów MAC). Jeśli mamy do czynienia z małą podsiecią lub urządzenie analizujące znajduje się w pobliżu obserwowanej stacji i jest odseparowane od reszty sieci np. poprzez switch, to nie występuje konieczność filtrowania.

4.7. Przyporządkowanie połączeń

Mówimy o jednym połączeniu wówczas, gdy mamy do czynienia z krzyżowaniem się adresów MAC, adresów IP oraz numerów portów TCP. Przystępując do rozkodowywania telegramów trzeba koniecznie stworzyć listę, w której widoczne będą poszczególne połączenia (wprowadzenie w punkcie 3.4). Aby można było zarejestrować listę połączeń, które w zależności od charakteru danych zostałyby zapisywane do pamięci, trzeba zbudować stosowny algorytm (Rysunek 19).

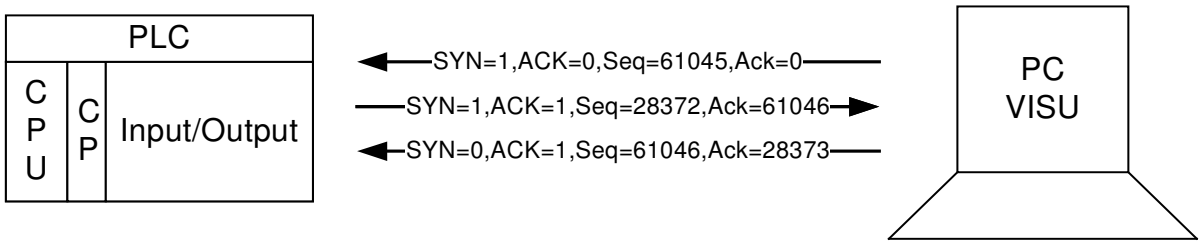


Rysunek 19. Algorytm tworzenia listy połączeń.

Logika algorytmu rozpoznawania połączeń musi być uniwersalna, a zatem dająca możliwość dekodowania telegramów przesyłanych w obu kierunkach. W projekcie „EchoCheck" odwzorowanie tej logiki ma miejsce w programie analizującym telegramy (plik źródłowy „CalcIp.c", funkcja „RefInIpListe").

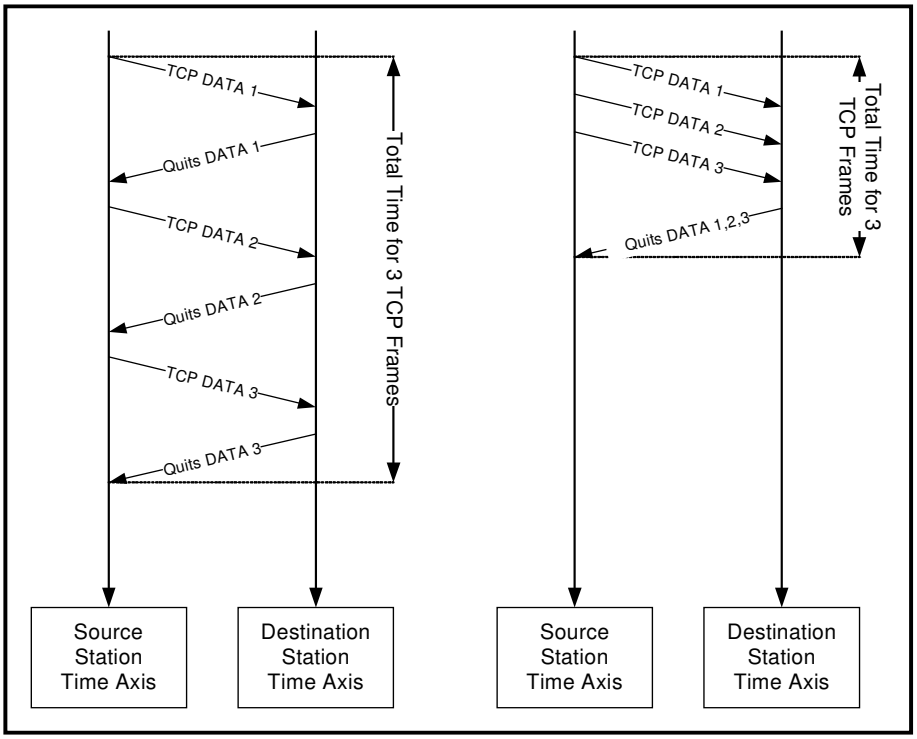
4.8. Poprawność przesyłanych telegramów

Funkcja „RefInIpListe" sprawdza również poprawność przesyłanych danych, analizując czy wszystkie dane wysłane przez jedną stację zostały poprawnie pokwitowane przez stację odbiorczą. Analiza ta polega na sprawdzaniu numerów w polu SEQUENCE NUMBER (Seq) i ACKNOWLEDGMENT NUMBER (Ack). Przy budowaniu połączenia (Rysunek 20) następuje tzw. Three-Way-Handshake, czyli stacja źródłowa wysyła do stacji docelowej bit flagowy oznaczający synchronizację (<SYN=1>, <ACK>=0), następnie stacja docelowa potwierdza chęć rozpoczęcia połączenia (<ACK =1><SYN=1>), na co stacja źródłowa odpowiada kolejnym potwierdzeniem (<ACK=1><SYN=0>) i zaczyna w kolejnym telegramie przesyłać dane (więcej w punkcie 2.4.2). Gdy jedna ze stacji chce zakończyć połączenie, to inicjuje ona zakończenie połączenia wystawiając flagę <FIN=1>. Proces zakończenia połączenia przebiega również w trzech krokach.



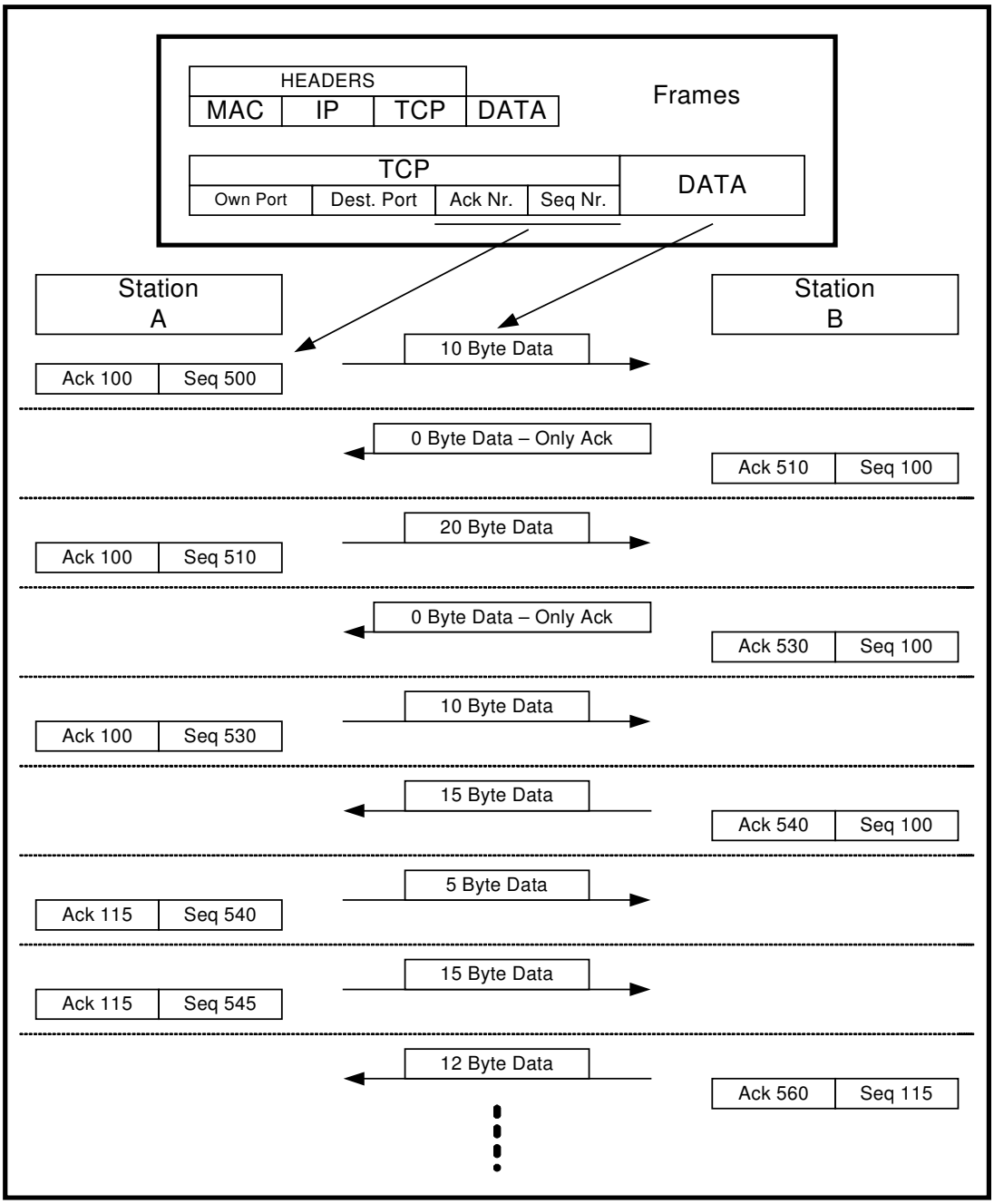
Rysunek 20. Synchronizacja połączenia na poziomie protokołu TCP.

W automatyce przemysłowej raz zbudowane połączenia pomiędzy stacjami trwają często bardzo długo (np. wizualizacje). Podczas połączenia przesłane dane mogą być kwitowane po każdym dostarczonym telegramie lub kwitowane może być jednorazowo kilka telegramów (Rysunek 21)- zależy to od „Timingu" (ustawień, odległości, łączy itd.).



Rysunek 21. Sposoby kwitowania telegramów (TCP).

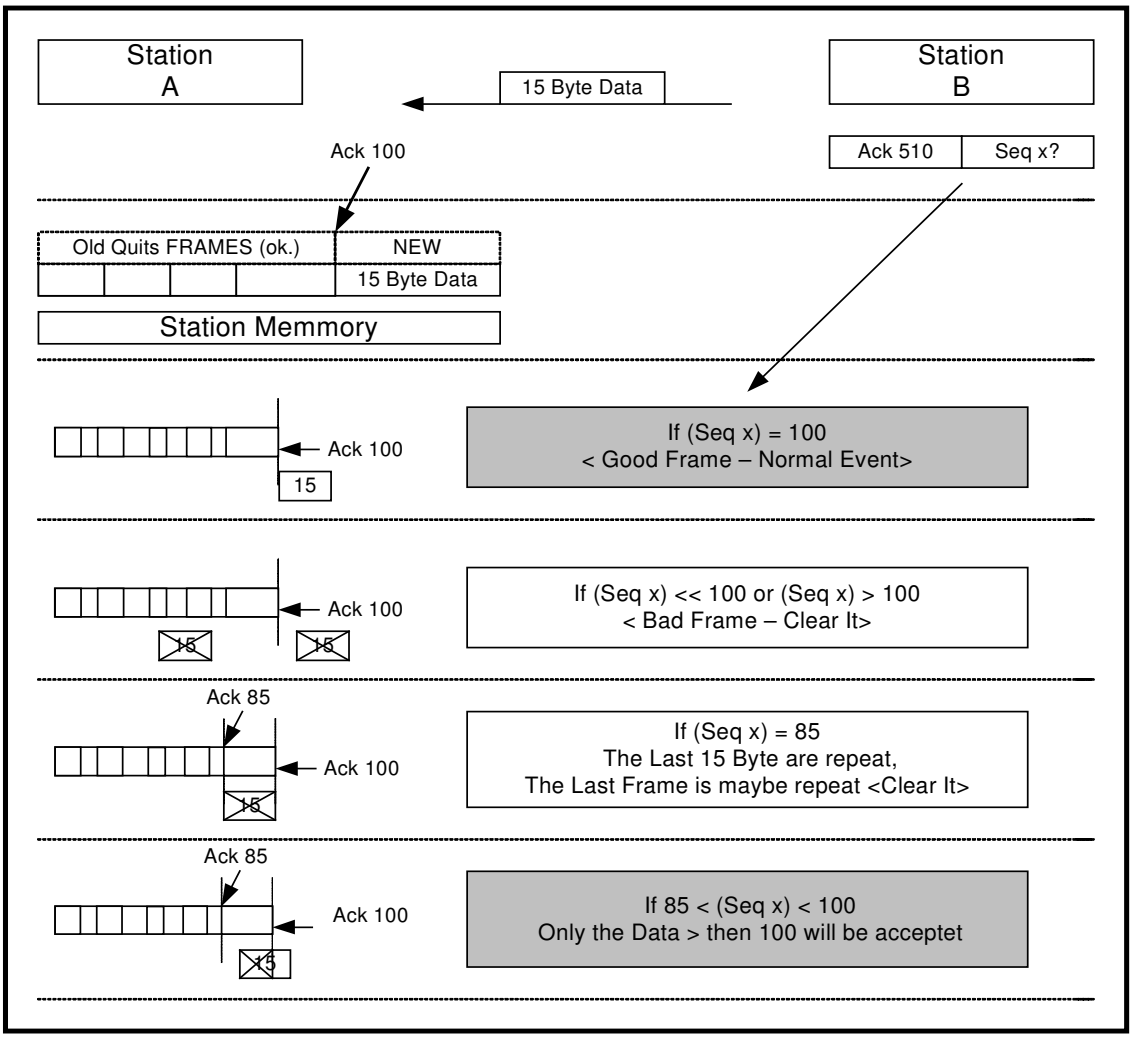
Takie potwierdzenie dostarczenia danych realizowane jest poprzez zliczanie w górę od przypadkowych numerów (Seq, Ack), przypisanych przy rozpoczęciu połączenia o wielkość przesłanych danych, z zachowaniem kolejności (Rysunek 22). Liczby (Ack) i (Seq) po osiągnięciu maksymalnej wartości 32 bitów zaczynają się od 0.



Rysunek 22. Przykłady kwitowania otrzymanych danych.

Gdy w dostarczonym telegramie numer sekwencyjny Seq x, określający odkąd nowo dostarczone dane mają się zaczynać (mają zostać umieszczone w pamięci) nie będzie się pokrywał z numerem kwitującym już uprzednio otrzymane dane (Ack 100 <> Seq x) lub jeśli numer Seq x będzie na tyle mniejszy od numeru kwitującego Ack 100, że obszar danych przesyłanych w tym telegramie nie będzie większy od obszaru już pokwitowanego, to taki telegram zostanie odrzucony. Każdy telegram posiadający numer sekwencyjny równy numerowi ostatnio pokwitowanych danych zostaje przyjęty, a dane wpisane do pamięci. W przypadku, kiedy telegram w części zawiera już pokwitowane dane a w części nowe, to dane

nie pokwitowane zostają wpisane do pamięci, a reszta danych jest odrzucona. Przypadki te zostały zobrazowane na poniższym rysunku (Rysunek 23).



Rysunek 23. Typowe i nietypowe rodzaje otrzymywanych telegramów.

Opisane i zobrazowane powyżej sytuacje zostały przeze mnie odpowiednio zaimplementowane w pliku źródłowym („CalcIp.c"). Stworzenie uniwersalnego algorytmu w kodzie programowym, uwzględniającego te przypadki, oraz skrajne warunki (osiągniecie maksymalnej liczby 32 Bit.), wymagało dużego nakładu pracy, oraz długich rozważań.

5. Dekodowanie danych przesłanych ponad protokołami TCP/IP

Dane osadzone na protokołach TCP/IP mają ,w zależności od producenta stacji, różne formaty. Urządzenia firmy Siemens w polu danych zawierają zazwyczaj kilka nagłówków, np. protokół RFC 1006, po którym następuje nagłówek protokołu S5 lub S7. Inni producenci, korzystający np. z protokołu Modbus, osadzonego bezpośrednio na protokołach TCP/IP, wysyłają dane nie posiadające nagłówka.

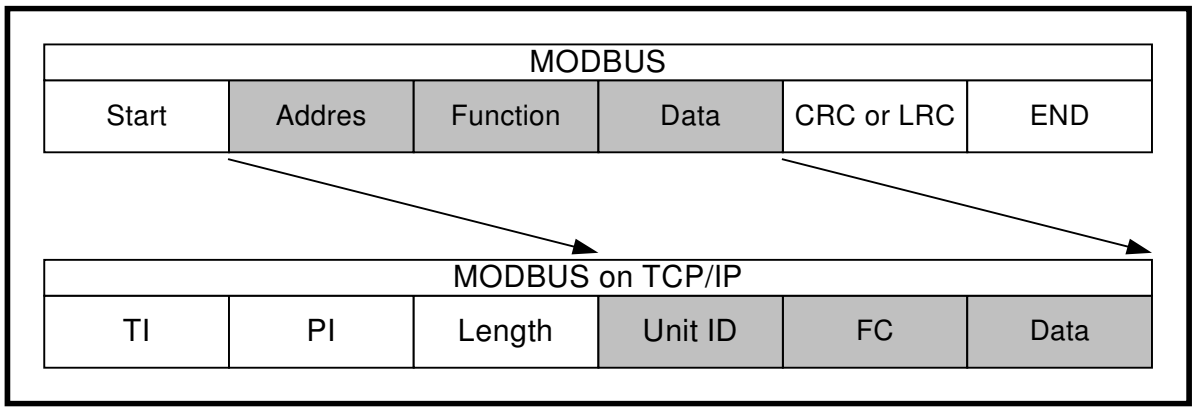
5.1. Protokół Modbus (Modbus on TCP) [N-6]

Jest to protokół bardzo prosty i tani, co ma swoje wady i zalety. Każda wysłana informacja do stacji odbiorczej musi zostać odpowiednio pokwitowana przez tę stację, aby możliwe było wysłanie kolejnych danych. Analizując komunikację sieciową opartą o standard Ethernet-owy oraz osadzoną na protokołach IP i TCP trudno jest rozpoznać kiedy ma się do czynienia z protokołem Modbus. Owa uciążliwość wynika z faktu, że protokół ten nie posiada specyficznego nagłówka, a dane są przesyłane w formie „strumienia binarnego". W praktyce jednak przyjęte jest, że stacje komunikujące się przy pomocy protokołu Modbus on TCP w protokole TCP używają portu 502. Wiedząc, że ma się już do czynienia z tym protokołem, można w odpowiedni sposób rozpoznać w binarnym ciągu znaków tryb zapisu bądź odczytu danych, gdyż jest to istotne w projekcie „EchoCheck". Ramka tego protokołu składa się minimalnie z 8 bajtów (sam nagłówek – „read multiple") a maksymalnie z 261 bajtów.

Modbus Header	
16 Bit	TID
16 Bit	PID
8 Bit	Length (0)
8 Bit	Length
8 Bit	UID
8 Bit	FC
Data(0-253 byte)	

Tabela 8. Ramka protokołu Modbus on TCP

Specyfikacja protokołu Modbus jest nieco inna dla różnych form przesyłu danych (Modbus RTU, ASCII i TCP). Wymienione pozycje nagłówka w ramce (Tabela 8) i opisane poniżej, dotyczą protokołu Modbus TCP. Pierwsze 16 bitów przypisane jest do TID (Transaction Identifier) i zazwyczaj ma wartość 0. Kolejne pole w nagłówku to PID (Protocol Identifier), któremu również jest przypisana wartość 0. Pole długości składa się z dwóch podpól, pola „Length (0)" oraz „Length". Pierwszemu z tych pól (pole o wyższym bajcie) jest przypisana również wartość 0, choć teoretycznie może ona przyjmować wartości od 0 do 256. Drugiemu polu długości (o niższym bajcie) przypisana jest liczba bajtów towarzyszących. Pole to określa ile bajtów nastąpi po bajcie przypisanym do pola Length włącznie. Pole UID (Unit Identifier, Modbus Slave Address) określa adres stacji „slave" (podporządkowanej), podpiętej do sieci komunikującej się protokołem Modbus. Pole to nie ma jednak znaczenia w protokole Modbus osadzonym na TCP/IP. Ostatnie pole nagłówka – FC (Function Code) określa charakter całego telegramu. W zależności od kodu zapisanego w tym polu może to być żądanie (Request), odpowiedź (Confirm) lub bardzo efektywna metoda jednoczesnego przesyłania żądania i odpowiedzi w jednym telegramie.



Rysunek 24. Ramka protokołu Modbus TCP/IP w odniesieniu do Modbus RTU.

Jeśli opkod świadczy o tym, że mamy do czynienia z telegramem żądającym, to mogą nastąpić dwie opcje: można żądać zapisu lub odczytu. Żądanie odczytu zawiera informację o miejscu odkąd ma się zacząć czytanie i jaki obszar ma objąć. Żądanie zapisu zawiera takie same dane jak żądanie odpowiedzi, ale po danych określających długość znajdują się dane „użyteczne". Jako odpowiedź (Confirm) możemy otrzymać błąd ,gdy wartość bitu wynosi 7 – najwyższy bit lub, gdy uprzednio miało miejsce żądanie odczytu danych przesłane zostaną żądane dane. Nieco inaczej wygląda to w telegramie kombinowanym. Protokół ten jest szerzej opisany w jednym z materiałów źródłowych firmy INAT, dostępnych na firmowej stronie internetowej [9]. Wykrycie tego protokołu oraz rozpoznanie jego charakteru

zostało również zaimplementowane w programie, wchodzącym w skład mojej pracy dyplomowej.

5.2. Protokół RFC 1006

Protokół ten jest wykorzystywany w automatyce przemysłowej wyłącznie przez firmę Siemens. Aby nie opracowywać nowego protokołu, tworzącego połączenia (TSAP) oraz określającego wielkość danych, przesyłanych ponad nagłówkami TCP/IP do sterownika (TPDU), firma Siemens przeniosła i zaimplementowała swój protokół ponad protokołem TCP, wykorzystując do tego celu protokół RFC 1006. Wielkość TPDU jest jak gdyby odpowiednikiem Window (okna) w protokole TCP lecz nie jest ona dynamiczna a stała, przyjmując wartość odpowiadającą modelowi sterownika (PLC). Sterownik w odpowiedzi na zapytanie powiadamia stację nadawczą, jakiej wielkości dane może przyjmować. Po zbudowaniu połączenia na poziomie protokołu TCP (Rysunek 20), kiedy to następuje synchronizacja portu 102 (po stronie stacji docelowej) z dowolnym portem (przypadkowym, jako własnym) stacji nadawczej i na odwrót, następuje zbudowanie połączenia na poziomie protokołu RFC 1006.

Budowa połączenia w protokole RFC 1006 przebiega podobnie jak w protokole TCP, jednak zamiast numerów portów obie stacje musza mieć uprzednio sparametryzowane numery TSAP - „skrzyżowane", czyli numer własny TSAP stacji nadawczej musi być równy numerowi odbiorczemu stacji docelowej i odwrotnie. Przydzielając stacji odbiorczej i nadawczej dla obu pól TSAP identyczne numery również bezproblemowo dochodzi do połączenia. Budowaniu tego połączenia towarzyszy parametr TPDU, który został powyżej opisany. Raz tak zbudowane połączenie pomiędzy stacjami w automatyce przemysłowej, może trwać od kilku sekund do kilku miesięcy (np. OPC serwery). Zakończenie telegramu ma miejsce, gdy aktywne jest pole nagłówka RFC 1006 określone skrótem EOT (End Of Text). Każda stacja może posiadać więcej połączeń RFC 1006, co wykorzystuje się do odpowiedniego zarządzania poszczególnymi połączeniami, w celu zoptymalizowania komunikacji sterownika z innymi stacjami i różnymi zadaniami. Szczegółowe informacje na temat protokołu RFC 1006 można znaleźć w ogólnodostępnych specyfikacjach RFC. Rozkodowanie tego protokołu oraz dalszych danych do niego dołączonych stanowi również tematykę mojej pracy dyplomowej i zostało zrealizowane w plikach źródłowych (PlcHeaderDecode.c, ProtocolDecode.c, S5.c).

5.3. INAT PLC Header [8]

Zorientowana połączeniowo kombinacja protokołu TCP/IP jest w stanie zbierać małe jednostki danych i łączyć je w większe telegramy. Ta właściwość zwiększa sprawność sieci. Wymagane jest jednak w protokołach powyżej protokołu TCP rozpoznawanie danych (nagłówek), takie jakie ma miejsce w innych protokołach (FTP, HTTP). Dla transportowania danych do CPU (Central Procesor Unit) sterownika Simatic S5 lub S7 wykorzystywany jest opisany w poniższej tabeli INAT PLC Header (Tabela 9). Nagłówek ten ma rozmiar ośmiu bajtów, które zawierają następujące informacje:

Byte Nr.	Meaning
Byte 0	0x4d ´M´
Byte 1	0x4b ´K´
Byte 2[3]	Datalen LSB[1] Length of the Data after INAT Header
Byte 3[3]	Datalen MSB[2] Header Length
Byte 4[3]	Bit 0 = 1, if follow next Frame
Byte 5[3]	0
Byte 6[3]	SeqNo. LSB[1]
Byte 7[3]	SeqNo. MSB[2]
Datalen Bytes	Use Data

[1]LSB: Least (Lower) Significant Byte (Najniższy bajt)

[2]MSB: Most Significant Byte (Najwyższy bajt)

[3] Byte 2 / 3 Byte 4 / 5 and Byte 6 / 7 create together Data Type – Short, INTEL – Type

Tabela 9. INAT PLC – Header

Acknowladges (potwierdzenia)

Jeśli „Datalen" (Byte 2) równy jest zeru oznacza to, że nie mamy do czynienia z danymi „użytecznymi" (Use Data), a jedynie z telegramem " życiowym" (Life Data Ack). Potwierdzenia danych umożliwiają śledzenie połączenia (jego podtrzymywanie), które nie jest przewidziane w protokołach TCP/IP jako telegram "życiowy". Czasy śledzenia połączenia odpowiadają czasom protokołu H1. Dzięki temu system taki staje się z punktu widzenia PLC lub PC kompatybilny do H1 (odwzorowuje H1).

Sequence Number

Bajt 6 i 7 oznacza numer sekwencyjny, który podczas budowania połączenia**,** jak również w trakcie wysyłania telegramów „życiowych", posiada wartość zerową, natomiast w czasie przesyłania danych (Use Data) jest o jeden zliczany w górę. Ten licznik telegramów służy dodatkowemu zabezpieczeniu transportu danych.

Połączenia Fetach i Write

Podczas poleceń Fetach (odczyt) i Write (zapis) 16 pierwszych bajtów danych odpowiada podczas rozpoczęcia zadania nagłówkowi SINEC AP (Siemens). Nagłówek SINEC AP jest również zastosowany w komunikacji poprzez H1.

Wysyłanie / odbieranie danych

Podczas wysyłania danych poprzez S5 TCP/IP w jednym telegramie zostaje wysłanych maksymalnie 512 bajtów danych „użytkowych" (Use Data). Ta maksymalna wielkość jest zdefiniowana poprzez „Kachelblockgröße". Podczas odbioru pakiet danych może osiągnąć rozmiar 1460 bajtów. Te graniczne wartości są automatycznie śledzone przez protokół TCP/IP, co oznacza, że śledzenie po stronie użytkownika nie jest już konieczne.

Transport danych bez nagłówka

Można sparametryzować protokół tak, aby na początku telegramu nie było nagłówka. W takim wypadku odpowiedzialne za przesył danych są programy aplikacyjne, pracujące w stacji źródłowej i docelowej. Powinno się jednak uwzględnić następujące fakty:

- ✓ Podczas zadań „Send Direct" (wysyłanie bezpośrednie) i „Receive Direct" (bezpośredni odbiór) nie mogą zostać przekroczone odpowiednie limity czasowe, aż do przejęcia telegramu. Przy zaniechaniu tych limitów czasowych mogłoby dojść do przepełnienia bufora (np. poprzez zapytania - Fetch), co z kolei uniemożliwiłoby ponowne zsynchronizowanie zapytań i odpowiedzi (Fetach i Write).
- ✓ Zachowanie określonego mechanizmu podczas blokowego przesyłu danych jest konieczne, aby można było rozpoznać kiedy następuje koniec przesyłania danych (Use Data).
- ✓ Po stronie odbiorczej można stwierdzić, że telegramy pochodzące z bufora odbiorczego (tej strony) są odczytywane zanim stacja przeciwna wyśle kolejny telegram.
- ✓ Niezbędne jest stworzenie kontroli połączenia na poziomie aplikacji.

5.4. Protokół S5 [8]

Kolejnym protokołem bazującym na Ethernecie i rozkodowywanym przez program DriverRecord jest protokół firmy Siemens – AP-S5. Nagłówek protokołu oraz ogólną strukturę ramki danych przedstawiają dla różnych przypadków poniższe tabele.

0	Systemkennung.....	S	H E A D E R	0	Systemkennung.....	S
1	.....	5		1	.....	5
2	..Länge.Header.....	=16d.		2	..Länge.Header.....	=16d.
3	..Kenn. OP-Code.....	=01..		3	..Kenn. OP-Code.....	=01..
4	..Länge OP-Code.....	=03..		4	..Länge OP-Code.....	=03..
5	..OP-Code.....	=05..		5	..OP-Code.....	=06..
6	..ORG-Block.....	=03..		6	..Quittungsblock.....	=0Fh
7	..Länge ORG-Block.....	=08..		7	..Länge Q-Block.....	=03..
8	ORG-Kennung.....			8	..Fehler-Nr.....	=Nr...
9	..DBNR.....			9	..Leerblock.....	=FFh
A	Anfangs-	H		A	..Länge Leerbl.....	7
B	..adresse.....	L..		B		
C	Länge.....	H		C		
D	.....	L..		D		
E	..Leerblock.....	=FFh.		E	frei	
F	Länge Leerbl.	2		F		
					Daten bis zu 64 k jedoch nur, wenn Fehler-Nr. = 0	
Read - Telegram żądający (czytaj)				**Read - Telegram kwitujący**		

Tabela 10. Nagłówek 16 bajtowego telegramu AP-S5 podczas czytania z S5 CPU.

0	Systemkennung	S	H E A D E R	0	Systemkennung.....	S
1		5		1	.....	5
2	..Länge.Header.....	=16d.		2	..Länge.Header.....	=16d.
3	..Kenn. OP-Code.....	=01..		3	..Kenn. OP-Code.....	=01..
4	..Länge OP-Code.....	=03..		4	..Länge OP-Code.....	=03..
5	..OP-Code.....	=03..		5	..OP-Code.....	=04..
6	..ORG-Block.....	=03..		6	..Quittungsblock.....	=0Fh
7	..Länge ORG-Block.....	=08..		7	..Länge Q-Block.....	=03..
8	ORG-Kennung.....			8	..Fehler-Nr.....	=Nr...
9	..DBNR.....			9	..Leerblock.....	=FFh
A	Anfangs-	H		A	..Länge Leerbl.....	7
B	..adresse.....	L..		B		
C	Länge.....	H		C		
D	.....	L..		D		
E	..Leerblock.....	=FFh.		E	frei	
F	Länge Leerbl.	2		F		
Daten bis zu 64 K						
Write - Telegram rozkazujący (pisz)				**Write - Telegram kwitujący**		

Tabela 11. Nagłówek 16 bajtowego telegramu AP-S5 podczas zapisu do S5 CPU.

W kolejnej tabeli (Tabela 12) przedstawione są parametry przekazywane przez „ORG-Kennung” – ósmy bajt w nagłówku AP-S5. Przypisując liczbę od 1 do 17, przekazywane są do (i ze) sterownika odpowiednie dane.

#define KENNUNG_BAUSTEIN	1
#define KENNUNG_MERKER	2
#define KENNUNG_EINGANG	3
#define KENNUNG_AUSGANG	4
#define KENNUNG_PERIPHERIE	5
#define KENNUNG_ZAEHLER	6
#define KENNUNG_TIMER	7
#define KENNUNG_SYSTEMDATEN	8
#define KENNUNG_ABSOLUT	9
#define KENNUNG_ERW_BAUSTEIN	10
#define KENNUNG_EXTMEM	16
#define KENNUNG_EXT_PERIPHERIE	17

Tabela 12. Oznaczenia jakie są przypisane do ORG-Kennung.

Składowa nagłówka nazwana „OP-Code", której mogą być przypisane liczby od 3 do 6, decyduje o rodzaju telegramu przesyłanego do (i ze) sterownika (Tabela 13).

#define AP_OPCODE_WRITE	3
#define AP_OPCODE_WRITE_ACK	4
#define AP_OPCODE_READ	5
#define AP_OPCODE_READ_ACK	6

Tabela 13. Zestawienie oznaczeń określających rodzaj telegramu dla AP-S5.

W przypadku, gdy nie dochodzi do odczytu (zapisu) danych ze (do) sterownika w polu „Fehler-Nr....." nagłówka protokołu pojawia się numer określający przyczynę niepowodzenia (Tabela 14).

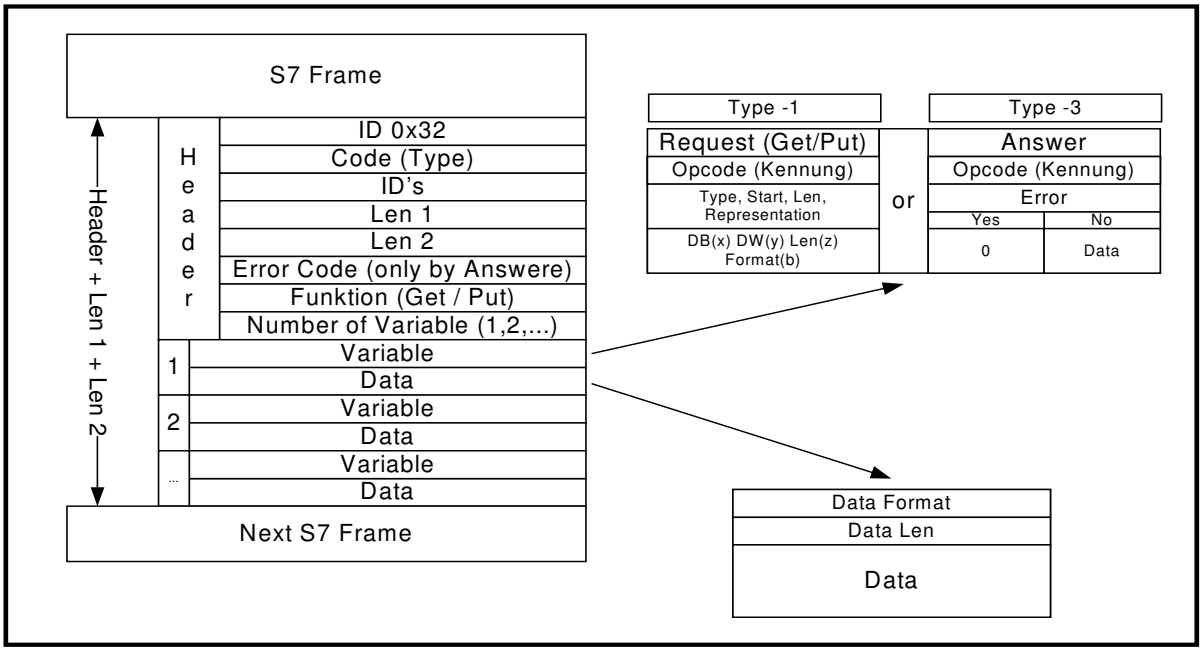
#define S5ERR_BAD_TYP	1
#define S5ERR_NO_DB	2
#define S5ERR_TOO_SHORT	3
#define S5ERR_QVZ	4

Tabela 14. Zdefiniowane błędy przy komunikacji ze sterownikiem S5.

Zacieniowane pola oznaczają „realnie możliwe błędy", co oznacza, że przypadek pierwszy i ostatni nigdy nie mają miejsca, ale oba zostały zdefiniowane przez producenta. Protokół ten (Sinec-AP) został udostępniony w dużej mierze (ale nie cały), co pozwoliło producentom oprogramowania służącego do wizualizacji jego zaimplementowanie i rozpowszechnienie. Tylko dzięki udostępnieniu bliższych informacji o tym protokole stał się on popularny w sieciach automatyki przemysłowej. Mogą się nim posługiwać tylko niektóre sterowniki z rodziny S5. Jego rozkodowanie, także ma miejsce w programie stanowiącym centralną część mojej pracy dyplomowej (plik źródłowy – „S5.c").

5.5. Protokół S7

Ostatnim protokołem, który jest rozkodowywany przez program DriverRecord, jest protokół S7 firmy Siemens. Ramka tego protokołu oraz dane mogą być osadzone na protokole RFC 1006, protokole stworzonym przez firmę INAT – PLC Header lub bezpośrednio na protokole TCP. Jest on wykorzystywany przy komunikacji ze sterownikami programowalnymi serii S7-300 i S7-400 wyżej wymienionej firmy. Sterowniki serii S7-200 posiadają własny protokół, nie będący w pełni kompatybilnym do protokołu S7. Firma Siemens nie udostępnia informacji o budowie nagłówka ani o pozostałej części tegoż protokołu. Korzystając z „Know-How" firmy INAT przedstawiam na poniższym rysunku budowę telegramu S7 (Rysunek 25).



Rysunek 25. Budowa telegramu S7 firmy Siemens.

Nagłówek tego protokołu posiada charakterystyczny parametr (ID = 0x32h), zapisany w postaci heksadecymalnej, który jednocześnie pozwala przy rozkodowywaniu strumienia danych na znalezienie początku telegramu S7. W kolejnym polu nagłówka określony jest typ telegramu. Gdy wartość umieszczona w tym polu wynosi 1 to mamy do czynienia z zapytaniem, gdy natomiast 3, to jest to odpowiedź. Kolejno mamy do czynienia z numerami ID oraz długościami. Długość pierwsza (Len 1) przeznaczona jest do określania poleceń i parametrów (Command and Parameter). Druga długość (Len 2) określa ilość danych następujących po nagłówku. W przypadku gdy mamy do czynienia z odpowiedzią to otrzymamy informację o błędzie. Jeśli w przesyłaniu telegramu lub w sterowniku wystąpił błąd to w uzyskanej odpowiedzi nie będzie przeniesionych żadnych danych. Natomiast, gdy nie wystąpi żaden błąd, to telegramowi z odpowiedzią będą towarzyszyć dane. „Numer zmiennych" (Number of Variable) to pole informujące o tym, ilokrotnie po nagłówku protokołu będą występować na przemian , lub w innej kombinacji pola zmiennych (Variable) i pola danych (Data). Niekiedy zdarza się w protokole S7 nietypowy zapis ilości danych, gdyż dane są liczone po połowie bitu (0,5 bitu). Pole zmiennych informuje, czy mamy do czynienia z żądaniem (Request) czy też z odpowiedzią (Answer). Żądać można zapisu danych lub odczytu danych. Jako odpowiedź otrzymać możemy dane i informację o braku błędu lub tylko informację o błędzie (bez danych). W opcji żądania znajdują się informacje na temat miejsca, długości i rodzaju (formatu) pobieranych (zapisywanych) danych.

Rozmiar telegramu S7 można uzyskać, sumując obie długości (Len 1, Len 2) oraz nagłówek tego protokołu. Znajdowanie protokołu S7 z uprzednio przygotowanych list połączeń ma miejsce w pliku źródłowym „ProtocolDecode.c". Wymiana danych na poziomie tego protokołu przebiega synchronicznie (po żądaniu przychodzi odpowiedź). To ile danych mamy otrzymać, wynika tylko i wyłącznie z żądania które uprzednio zostało wysłane. Pisząc programy w różnych językach programowania, należy mieć na uwadze sposoby pracy procesorów, na których dany program będzie wykonywany. W kodzie źródłowym mojej pracy często można spotkać komendę „MotorolaToIntel", która odpowiednio dla danego formatu liczb dokonuje konwersji Little-Endian <-> Big-Endian.

6. Program analizujący komunikację w Ethernet-owej sieci przemysłowej.

6.1. Wstęp

Dopełnieniem podstaw teoretycznych, a zarazem efektywną częścią mojej pracy dyplomowej jest program komputerowy napisany w języku „C" za pomocą narzędzia programistycznego jakim jest „Microsoft Visual C++ 6.0". Cały projekt oraz nazwa programu głównego to „DriverRecord". W programie tym, oprócz innych podprogramów napisanych wyłącznie pod kątem projektu „EchoCheck" – „długo-okresowa" analiza Ethernet-owej sieci przemysłowej, wykorzystywane są również biblioteki opracowane przez pracowników firmy INAT GmbH, będące wynikiem wieloletnich prac. Biblioteki te zapewniają m.in. kompatybilność systemową (różne wersje systemów operacyjnych), sprzętową (karty sieciowe różnych producentów i o różnych systemach zliczania) oraz zawierają listy rozkazów, którymi komunikują się poszczególne grupy sterowników programowalnych. Takie modułowe, a nie jednolite tworzenie programów oraz odpowiedni sposób konstrukcji podprogramów pozwala na późniejsze wykorzystanie już uprzednio stworzonej i zaimplementowanej logiki, co oszczędza czas i pieniądze.

W ramach pracy dyplomowej stworzyłem następujące podprogramy:

- ✓ „DriverRecord.c"
- ✓ „CalcIp.c"
- ✓ „Logger.c"
- ✓ „PlcHeaderDecode.c"
- ✓ „ProtocolDecode.c"
- ✓ „CpMemS5Subs.c"
- ✓ "ModbusTCP.c"
- ✓ "S5.c"
- ✓ "S7.c"

Funkcje jakie pełnia poszczególne podprogramy oraz program główny („DriverRecord.c") omówione zostaną szerzej w kolejnych podpunktach.

6.2. Ramowy opis funkcjonowania programu „DriverRecord”

6.2.1 Program główny - "DriverRecord.c"

- ✓ Przejście do funkcji "main".
- ✓ Sprawdzenie czy istnieje INAT Driver (sterownik sięgający po telegramy z karty sieciowej – wymagane jest jego osobne zainstalowanie, gdyż jest częścią Sniffera firmy INAT GmbH - "**NetSpector**").
- ✓ Przydzielenie odpowiedniego rozmiaru pamięci dla otrzymywanych z karty sieciowej telegramów (rs.Cb = sizeof(RECBUFFERSIZE)) oraz sprawdzenie dostępności tej pamięci.
- ✓ Wyzerowanie w sterowniku INAT Driver, numeru pierwszego telegramu (fh.StartNo = 0) oraz przyrostu czasu (dTime), gdyż parametry te przesyłane będą w nagłówku przesyłanym wraz z każdym telegramem do aplikacji.
- ✓ Rozpoczęcie rejestracji komunikacji „przechodzącej” przez kartę sieciową komputera, na którym uruchomiony został ten program.
- ✓ Przypisanie wartości w pętli "while", odpowiada ilościom cykli pobrania telegramów.
- ✓ Sprawdzenie, czy mamy do czynienia z telegramem o protokole IP (if (MotorolaToIntel(l2->DataLen) >= 0x7ff)), bowiem tylko takie są w tym programie analizowane.
- ✓ Jeśli jest to telegram IP, to **dalszy tok analizy kierowany jest do pliku źródłowego "CalcIp.c".**
- ✓ Ustawienie parametru "DosSleep(100)", który decyduje o czasie rejestracji.
- ✓ Uwolnienie zarezerwowanej pamięci („free(mfe)”).
- ✓ Wywołanie podprogramu "**Logger.c**", gdzie zawarte są wyniki rejestracji.
- ✓ Zakończenie rejestracji.
- ✓ Zamknięcie "INAT Drivera".

6.2.2 Plik źródłowy „CalcIp.c”

- ✓ Sprawdzenie wersji protokołu IP (w automatyce przemysłowej jest obecna na razie jedynie wersja IP v4.).
- ✓ Sprawdzenie poprawności długości telegramu dla wersji IP v4.
- ✓ Sprawdzenie protokołu osadzonego na protokole IP (czy jest to protokół TCP).
- ✓ Wczytanie do pamięci numerów IP, numerów portów (w TCP) oraz numerów Sequence i Acknowladge (w TCP) jedynie pod warunkiem, że nie spełniają założeń funkcji "**GetPortToWork**". W funkcji tej wymienione są numery portów TCP, którymi nie odbywa się komunikacja w świecie sterowników. Są to takie porty jak: 22 (ssh), 80 (http), 137 (Windows 95-98), 138 i 139 (Windows 2000/XP/NT/2003 without active directory), 143 (IMAP mail), 443 (shttp), 445 (Windows 2000/XP/NT/2003 with active directory), 0.
- ✓ Przejście do funkcji "**REFERENCE_LIST**" - sprawdzanie wpisu połączenia w liście połączeń, gdzie następuje cała procedura zobrazowana (Rysunek 19) i opisana w punkcie 4.7, oraz do funkcji "**EintragenInRef**", jeśli takie połączenie jeszcze nie jest na liście.
- ✓ Przejście do funkcji "**RefInIpListe**"(skomplikowana logika), gdzie:
  - odnajdywane jest wpisane do tymczasowej pamięci połączenie,
  - określony jest kierunek przesyłu danego telegramu,
  - następuje podział na dwa kierunki, dla każdego istniejącego połączenia,
  - sprawdzana jest poprawność numerów "Ack. i Seq" (poprawne kwitowanie i odbiór nowego telegramu - szerzej w punkcie 4.8 (Rysunek 22),
  - następuje wywołanie funkcji "**NextLayerCheck**", a co się z tym wiąże, automatyczne przejście do pliku źródłowego "**PlcHeaderDecode.c**",
  - w przypadku pojawienia się nowego telegramu następuje wywołanie funkcji "**NewData**".

- ✓ Funkcja "**NewData**", zawiera logikę rozpatrywania poprawności przesyłanych nowych danych umieszczonych ponad protokołami TCP/IP. Cel tej logiki został opisany przeze mnie w punkcie 4.8 (Rysunek 23). Logika ta obejmuje również skrajne przypadki, kiedy przesłany może być błędny telegram (zły numer sekwencyjny) lub gdy numer sekwencyjny przypada na "krawędzi" liczby 32 Bit, a nowe pokwitowanie zaczyna się od liczby leżącej blisko "0". Przypadek taki jest "normalny", jednak w tego rodzaju analizie musi on być wzięty pod uwagę, aby nie wysuwać błędnej opinii, o zajściu błędu podczas komunikacji.

6.2.3 Plik źródłowy „PlcHeaderDecode.c”

- ✓ W funkcji "**NextLayerCheck**" następuje sprawdzenie, jaki protokół jest osadzony na protokole TCP. Jako pierwsze sprawdzane są porty źródłowe i docelowe w połączeniach zamieszczonych na liście.
- ✓ Kiedy jeden z portów jest to port np. 102 wiadomo, że następuję protokół RFC 1006 (przejście do funkcji "**Rfc1006**").
- ✓ Jeśli jeden z portów jest to port np. 502, wiadomo wtedy, że następuję protokół ModbusTCP (przejście do funkcji "**ModbusTcp**", zamieszczonej w piku źródłowym "**ModbusTcp.c**", gdzie następuje już dekodowanie tego protokołu).
- ✓ Gdy nie jest to ani protokół RFC 1006, ani ModbusTCP, sprawdzane jest istnienie protokołu PlcHeader (INAT).
- ✓ Jeśli nagłówek telegramu będzie posiadał charakterystyczne "M" i "K" dla protokołu PlcHeader, to nastąpi przejście do funkcji "**PlcHeader**"), jeśli natomiast nie zostanie w powyższych trzech krokach rozpoznany rodzaj telegramu, to program przechodzi do funkcji "**ProtocolCheck**" zamieszczonej w pliku źródłowym "**ProtocolDecode.c**".
- ✓ Funkcja "**NextLayerCheck**" rozpatruje również przypadki, kiedy dane następujące po protokołach TCP i IP, mogą zostać "poszatkowane" np. poprzez router, wówczas należy "sklejać" poszczególne bity w kolejnych telegramach danego połączenia. Proces taki musi trwać, aż do momentu kiedy można jednoznacznie rozkodować nagłówek, cały telegram lub odrzucić błędne dane. Jest to przypadek skrajny i rzadko występujący, jednak w przypadku takiej analizy musi zostać uwzględniony.

6.2.4 Plik źródłowy „ProtocolDecode.c”

W tym pliku źródłowym, poprzez funkcję "**ProtocolCheck**" inicjowane jest przejście do dekodowania protokołów S5 i S7 (pliki źródłowe "**S5.c**" i "**S7.c**"). Przejście do tej funkcji ma miejsce bezpośrednio po wykryciu portu 102 w telegramie przesyłanym w dowolnym kierunku (S7), jak również w przypadku, gdy nie można było uprzednio określić z jakim protokołem ma się do czynienia (więcej w opisie pliku źródłowego "**PlcHeaderDecode.c**").

6.2.5 Plik źródłowy „Logger.c”

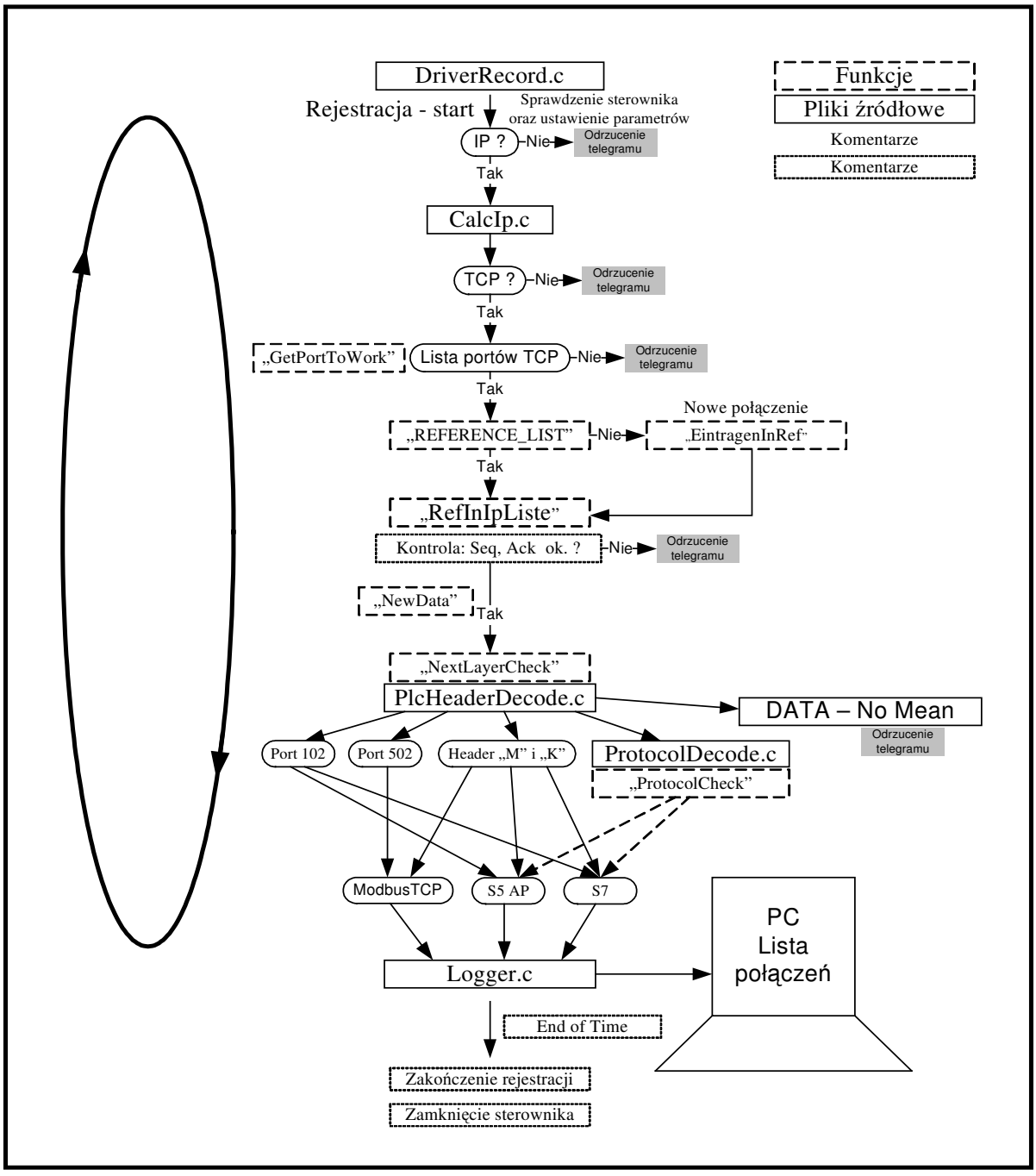
Do tego pliku kierowane są wyniki z listy zapisanej w pamięci, wszystkie połączenia ze sterownikami, które miały charakter zapisu danych zostają tu wyświetlone. Kod programowy zawarty w tym pliku ma za zadanie wyświetlić również pełną datę i godzinę ("stempel czasowy"), kiedy dane zdarzenie miało miejsce. Wszystkie pliki źródłowe, w których ma miejsce ostateczne dekodowanie poszczególnych protokołów sterowników programowalnych (PLC), dostarczają tych danych. Są to pliki: "**ModbusTcp.c**", "**S5.c**" oraz "**S7.c**".

6.2.6 Pliki źródłowe: "ModbusTcp.c", "S5.c" oraz "S7.c"

Pliki te zawierają struktury programu ściśle związane z poszczególnymi opisanymi budowy nagłówków zawartych w punkcie 5 niniejszej pracy. Toteż dalszy i wnikliwy opis samej logiki jest tu zbędny.

6.3. Ramowy schemat funkcjonowania programu

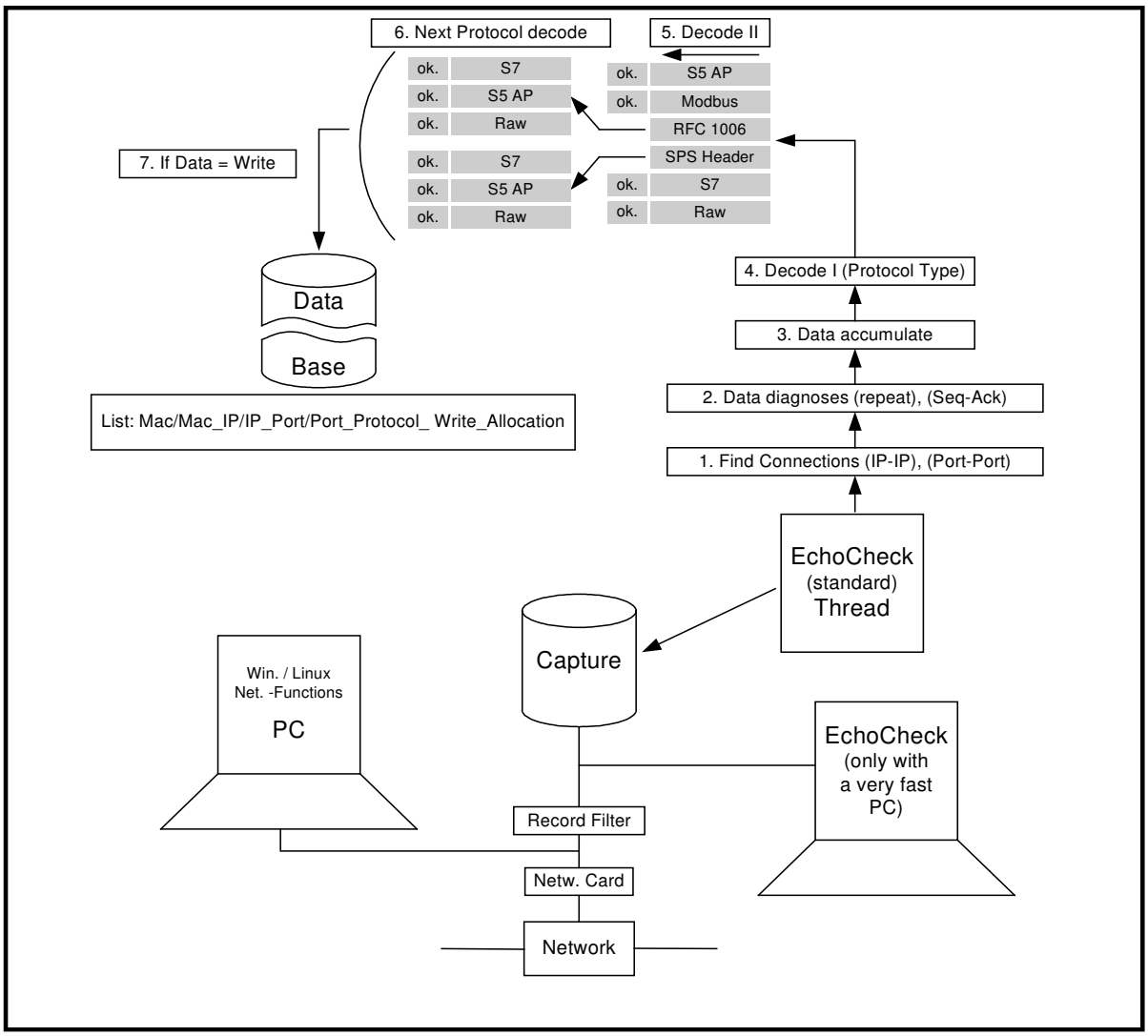
Schemat funkcjonowania programu „DriverRecord" został sporządzony w oparciu o opis plików źródłowych z punktów 6.2.1 – 6.2.6.



Rysunek 26. Rola poszczególnych funkcji i plików źródłowych w funkcjonowaniu programu „DriverRecord".

6.4. Ramowy przebieg dekodowania telegramów w programie „DriverRecord”

Program „DriverRecord” analizuje protokoły komunikacyjne sterowników programowalnych w sieciach Ethernet-owych, osadzone na protokołach TCP/IP. Dla lepszego zrozumienia i wyjaśnienia sposobu tego rodzaju analizy komunikacji w sieciach przemysłowych zobrazowany został proces pobierania, przetwarzania i zapisu danych (Rysunek 27).



Rysunek 27. Idea działania programu „DriverRecord”.

Umieszczona na poziomie warstwy fizycznej modelu ISO/OSI karta sieciowa pobiera poprawne i zaadresowane do niej telegramy Ethernet-owe. Po przygotowane przez kartę sieciową telegramy może sięgać jednocześnie więcej aplikacji, jak np. komputery PC, pracujące z różnymi systemami operacyjnymi i wykonującymi odpowiednie zadania, wymagające również korzystania z sieci. Dlatego ważnym aspektem przy wszelkiego rodzaju

analizach sieci jest umiejętność równoległego pobierania danych przez inną aplikację („DriverRecord") z interfejsu karty sieciowej, bez wprowadzania jakichkolwiek zakłóceń, mogących spowodować utratę danych przesyłanych do podstawowej aplikacji komputera lub urządzenia. Z tego właśnie powodu jakiekolwiek wstępne filtrowanie telegramów może być przeprowadzone dopiero na poziomie sterownika danej aplikacji (Rysunek 18). Możliwa jest analiza telegramów sieciowych przekazywanych bezpośrednio ze sterownika do aplikacji („EchoCheck"), umieszczonej na bardzo szybkim komputerze. Aby w czasie rzeczywistym przeanalizować i ewentualnie rozkodować wszystkie telegramy, nie tracąc przy tym rodzaju analizy żadnych informacji, konieczne jest zastosowanie kosztownego sprzętu. Celem projektu „EchoCheck" jest jednak stworzenie oprogramowania, urządzenia, które będzie tworzyło kompaktową, niedrogą aplikację, o niewielkich rozmiarach, małej mocy obliczeniowej, ale potrafiącej pewnie analizować komunikację sieciową. Aby móc spełnić te założenia konieczne jest zoptymalizowanie całego procesu analizy. Jednym ze sposobów jest wstępne filtrowanie odpowiednich portów, którymi nie są przesyłane protokoły sterowników programowalnych, a których analiza kosztowała by kolejne zasoby obliczeniowe i pamięciowe (więcej odnośnie założeń i rodzajów optymalizacji w punkcie 3.6.). Telegramy, po wstępnym filtrowaniu są wpisywane do tymczasowej pamięci, zobrazowanej (Rysunek 27) jako Capture, która umożliwia gromadzenie telegramów i ich dalszą obróbkę. Do telegramów tych dostęp umożliwiony ma osobny Thread (wątek), który pojedynczo pobiera telegramy z pamięci. Taki telegram jest przyporządkowywany do zapisanej w pamięci listy połączeń (Punkt 1 – Rysunek 27). Gdy jest to pierwszy telegram nowego połączenia, w liście połączeń inkrementowana jest nowa pozycja (Rysunek 19). Następnie na poziomie protokołu TCP, sprawdzane jest, czy jest to telegram powtórzeniowy, czy też następuje przesłanie w pewnej części już pokwitowanych danych, a w pewnej części nowych danych (Punkt 2 – Rysunek 27). W takim przypadku tylko nowe, niepokwitowane dane zostają zapisane, a pokwitowana część jest odrzucona (Rysunek 23). Tak wstępnie rozkodowany telegram - dane, już po odrzuceniu nagłówka IP i TCP, jest gromadzony dla poszczególnych połączeń i zapisywany w celu dalszego rozkodowywania, lub następuje oczekiwanie na kolejną część danych danego połączenia, niezbędną do rozpoczęcia dekodowania (szerzej w punkcie 3.5). W punkcie czwartym (Rysunek 27), miejsce ma dekodowanie danych osadzonych na protokole TCP. Dane te mogą mieć charakter danych „surowych" (raw), protokołów przeznaczonych do rozpoznawania długości danych (RFC 1006 i SPS-Header) oraz protokołów służących do komunikacji z odpowiednimi sterownikami programowalnymi – PLC (S5 AP, Modbus, S7). Na tym samym rysunku w punkcie piątym przedstawiona jest

sytuacja, kiedy następuje dalsze dekodowanie danych zawartych w protokołach RFC 1006 i SPS-Header. Mogą to być również dane „surowe", jak również protokoły sterowników programowalnych (S7 i S5 AP). Kolejnym krokiem (Punkt 6 – Rysunek 27) , który ma miejsce bezpośrednio po punkcie czwartym (gdy nie mamy do czynienia z protokołem RFC 1006 i SPS-Header) lub po punkcie piątym, jest dekodowanie danych, które decydują o reakcji sterownika na ich zawartość – są to „dane użyteczne" (protokoły sterowników PLC). W zależności od charakteru jaki mają dane przekazywane do sterownika, następuje przetworzenie tych danych i reakcja sterownika. W przypadku gdy dane te zmieniają ustawienia w pamięci sterownika (Write - wpis), następuje protokołowanie tego zajścia w pamięci.

7. Podsumowanie oraz kierunki dalszego rozwoju.

7.1. Podsumowanie

Cel, którym była rejestracja i dekodowanie komunikacji w sieci przemysłowej bazującej na Ethernecie został osiągnięty. W czasie sześciu miesięcy spędzonych w firmie INAT GmbH w Norymbergii oraz w wyniku aktywnej wymiany informacji na drodze elektronicznej z promotorem mojej pracy dyplomowej, doktorem inżynierem Zbigniewem Mantorski poznałem budowę oraz różne sposoby funkcjonowania sieci przemysłowych, łączących sterowniki programowalne (PLC), przekształtniki energoelektroniczne i inne urządzenia peryferyjne komunikujące się przy pomocy tego standardu przemysłowego. Oprócz szerokiej wiedzy teoretycznej na temat sieci Ethernet-owych jaka została mi przekazana i jaką literatura światowa udostępnia, zapoznałem się z wieloma praktycznymi zagadnieniami, które w sieciach przemysłowych funkcjonują, i są stosowane przez wielu praktyków, a niejednokrotnie brak informacji na ich temat w literaturze. Poprawne zrozumienie komunikacji, gdzie występują również różnego rodzaju zakłócenia i błędy, dało mi obraz zalet i wad takiej komunikacji oraz nauczyło diagnozowania przyczyn, takiego a nie innego zachowania sterowników na dane polecenia.

W wyniku bardzo dobrej współpracy z właścicielem firmy INAT GmbH, Dipl.- Ing. W. Krings, kierującym działem „Entwiklung", powstał program „DriverRecord". Program ten jest bazową częścią projektu „EchoCheck". Umożliwia on analizę sieci Ethernet i rejestrację istotnych, z punktu widzenia projektu EchoCheck, telegramów przesyłanych w sieciach ze sterownikami programowalnymi. Zostały wykonane liczne testy z udziałem różnych protokołów PLC. Program ten jest własnością firmy INAT GmbH.

7.2. Kierunki dalszego rozwoju

Bazowa część projektu „EchoCheck" będzie poszerzona o kolejne ważne protokoły sieciowe, takie jak np. AB (Allen Bradley) i inne. Dopracowane zostaną rozwiazania analizy i wprowadzone zostaną elementy umożliwiające parametryzację urządzenia z poziomu dozoru. Do części Ethernet-owej zostanie dołączona karta umożliwiająca pasywne uczestnictwo (nasłuch) w sieci polowej – Profibus. Całe urządzenie ma stanowić kompaktową całość, do której dostęp poprzez odpowiednie połączenia i urządzenia, będzie możliwy z każdego zakątka świata.

8. Źródła

8.1. Literatura, czasopisma, materiały firmowe

[1] Industrieautomation mit Ethernet-TCP/IP und Web-Technologie – F. Furrer

[2] Technik der IP-Netze – A. Badach, E. Hoffmann

[3] Ethernet – Technologien und Protokolle für Computervernetzung – J. Rech

[4] Moderne Datenkommunikation – Franz-Joachim Kauffels

[5] Grundlagen der Kommunikation – INAT GmbH

[6] Języki C i C ++ , Twój pierwszy pogram – Alan R. Neibauer

[7] Programmieren in C – Kernighan, Ritchie

[8] S5-TCP/IP- INAT GmbH

[9] Echolink Manual - INAT GmbH

8.2. Internet

[N-1] http://forum.msstudio.com.pl

[N-2] www.codeguru.pl

[N-3] www.inat.de

[N-4] www.wikipedia.org

[N-5] www.rfc-editor.org/

[N-6] www.m-system.co.jp/

8.3. Normy i specyfikacje

IETF (Internet Engineering Task Force) - RFC (Request For Comment)

RFC 760 – User Datagram Protocol (UDP)

RFC 791 – Internet Protocol (IP)

RFC 792 – Internet Control Message Protocol (ICMP)

RFC 793 – Transmission Control Protocol

RFC 826 – Ethernet

RFC 1006 – ISO on TCP

RFC 1055 – SLIP

RFC 1163 – BGP

RFC 1171/1172 – PPP

ANSI/IEEE Std 802.1D, Part 3: Media Access Control (MAC) Bridges, Institute of Electrical and Electronics Engineers, Inc., New York USA, 1998.

IEEE Std 802.3, Part 3: Carrier Sense with Multiple Access and Collision Detection (CSMA/CD) access method and physical layer specifications, Institute of Electrical and Electronics Engineers, Inc., New York USA, 2000.

IEEE Std 488.1-1987, Standard Digital Interface for Programmable Instrumentation, 1992