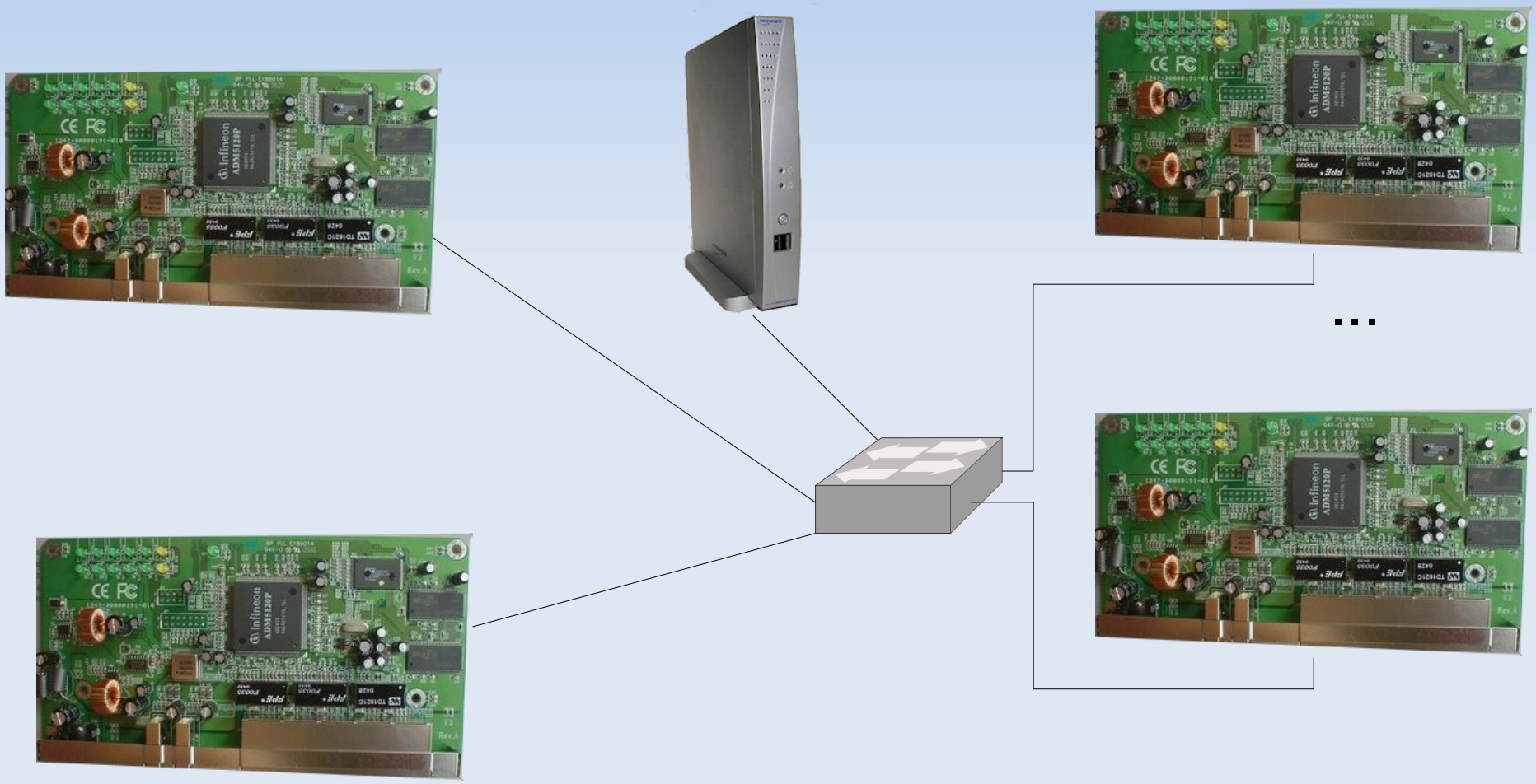
Platforma do testowania i rozwoju nowych protokołów rutingu



Agenda

- Idea projektu
- Środowisko do testowania: sprzęt czy wirtualizacja?
- Nowa koncepcja hybrydowa SDN
- Ruter programowy
- Implementacja
- Zastosowanie i możliwości
- Podsumowanie

Geneza projektu

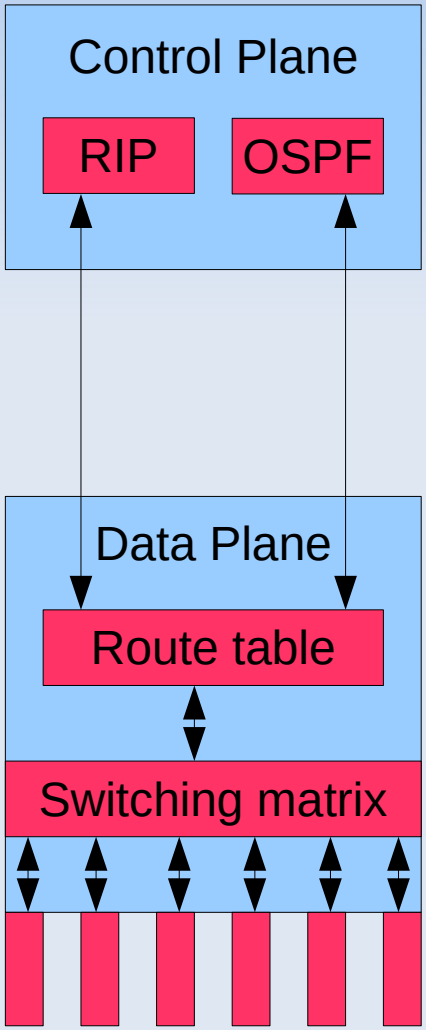
- Opracowanie platformy do testowania i implementacji nowych protokołów rutingu:
  - Niska cena
  - Energooszczędność
  - Mały rozmiar urządzeń
  - Szybka kompilacja i prosta procedura wgrywania oprogramowania
  - Wykorzystanie tylko i wyłącznie narzędzi o otwartym kodzie
  - Wykorzystanie gotowych rozwiązań
  - Interfejs programistyczny niezależny od sprzętu

Źródła inspiracji

- Udział w projektach badawczych:
  - Uruchomienie płaszczyzny sterującej na osobnej maszynie
- Sieciowe systemy wbudowane
  - Wykorzystanie taniego rutera jako płaszczyzny danych
- Ruter programowy
  - Gotowa implementacja płaszczyzny sterowania, np. projekt Quagga

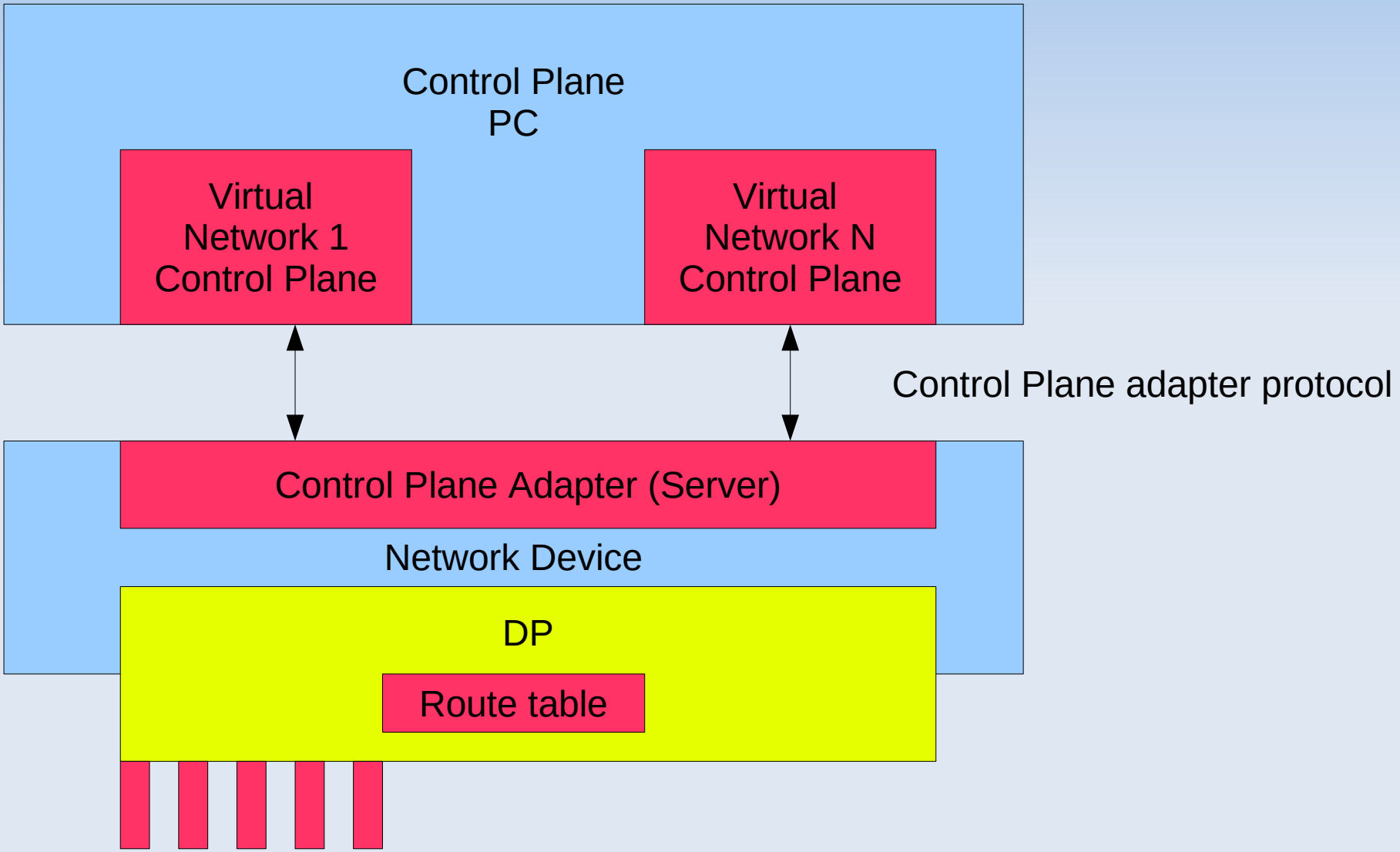
Źródła Inspiracji
Udział w projektach badawczych

- Ruter składa się z:
  - Warstwy sterującej,
  - Warstwy danych.
- Warstwa sterująca odpowiada za:
  - Odkrywanie topologii,
  - Wyznaczanie ścieżek,
  - Dodawanie tras to tablicy rutingu.
- Płaszczyzna danych:
  - Odpowada za przekazywanie pakietów zgodnie z regułami zapisanymi w tablicy rutingu.
  - Implementacja sprzętowa lub programowa.



Źródła Inspiracji

Udział w projektach badawczych



Rozwiązanie sprzętowe vs symulator

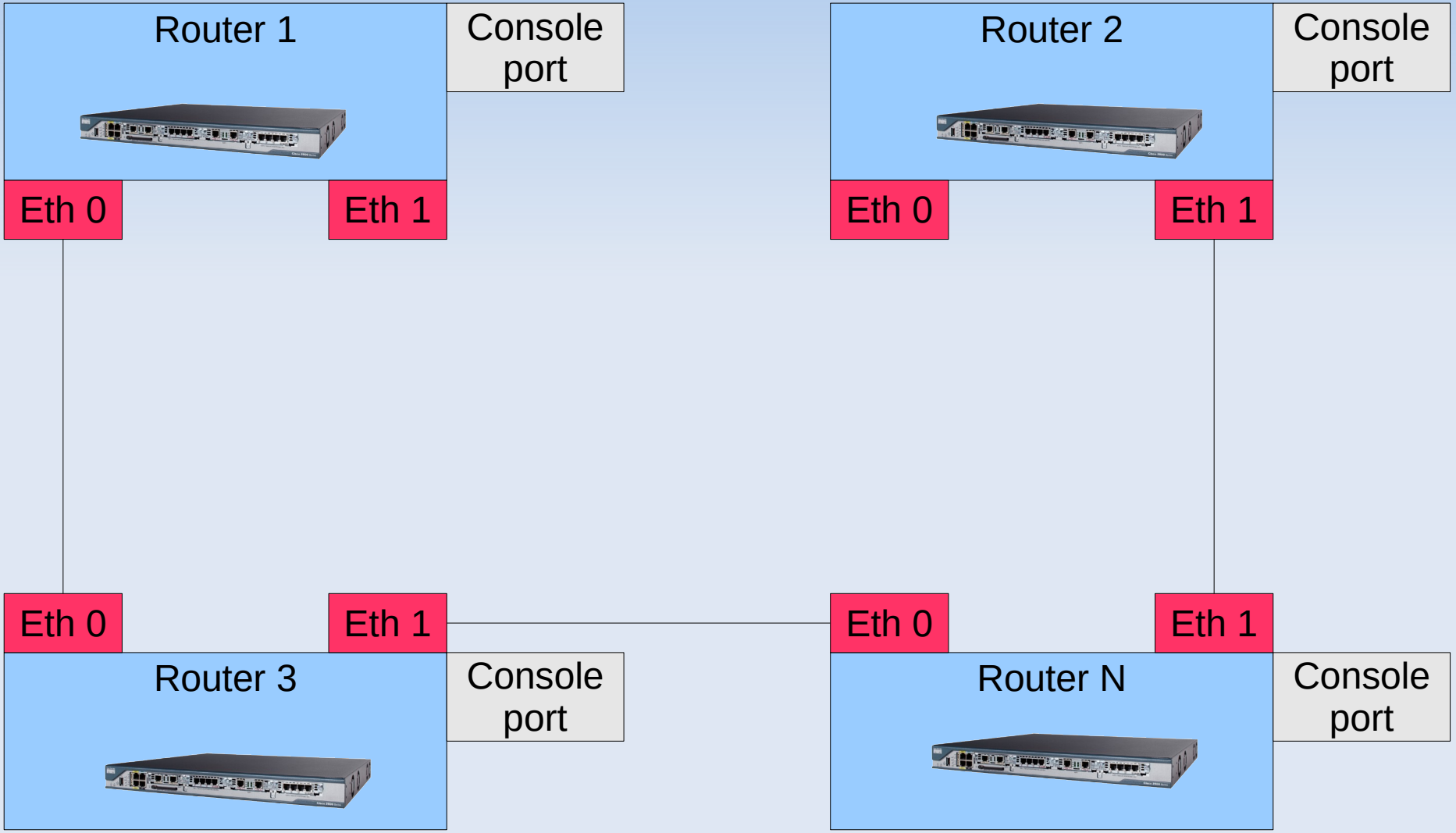
Hardware Testbed:

- Sprzęt z interfejsami, które należy połączyć
- Wymaga wielu ruterów
- Wysoka wydajność
- Wymagany jest bezpośredni dostęp do sprzętu
- Prostota modyfikacji topologii połączeń
- Wysoka cena

Virtual Testbed:

- Zastosowane są wirtualne węzły wraz z interfejsami
- Wystarczy pojedynczy serwer
- Każda maszyna może być zrealizowana jako ruter programowy
- Połączenia i topologia jest niewidoczna z zewnątrz

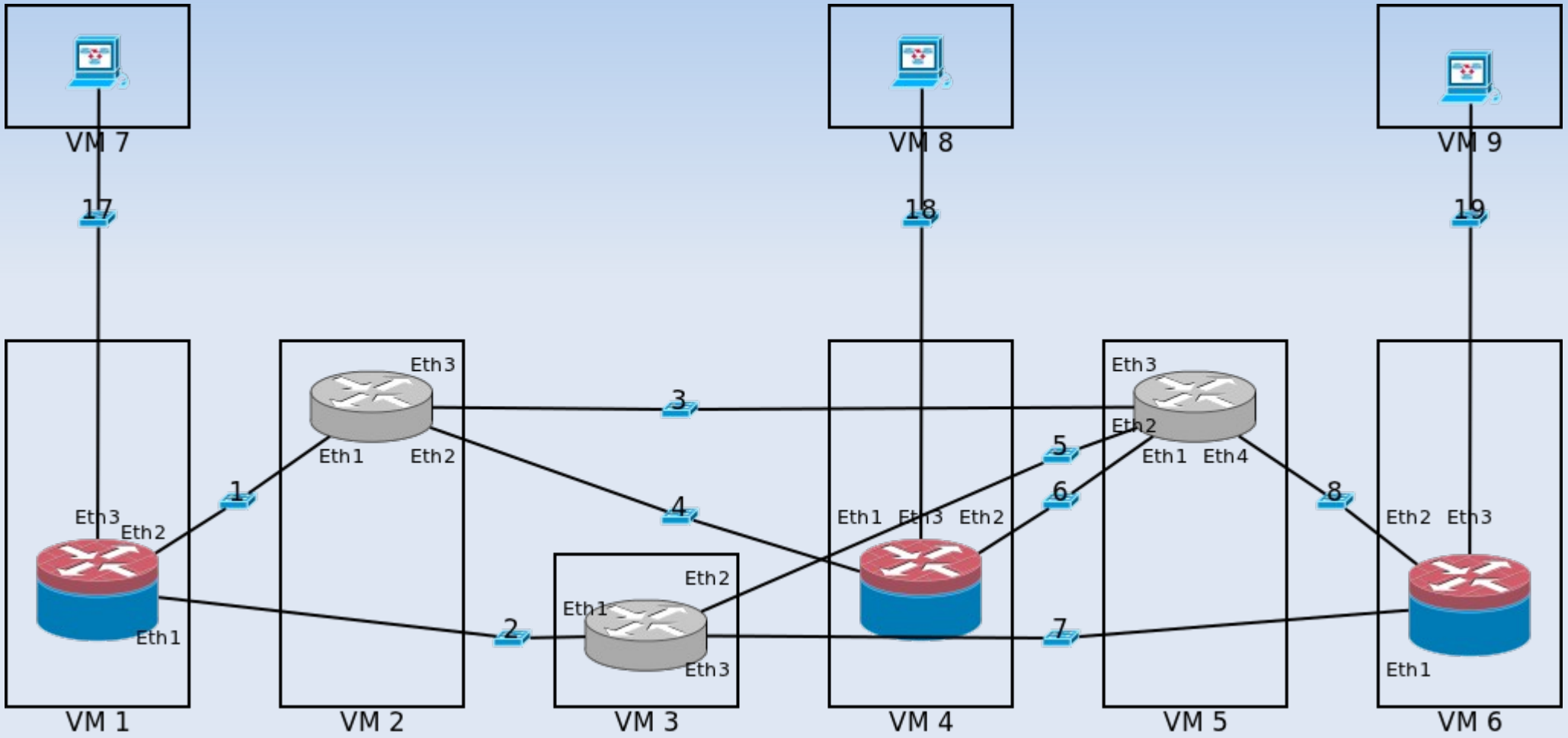
Rozwiązanie sprzętowe



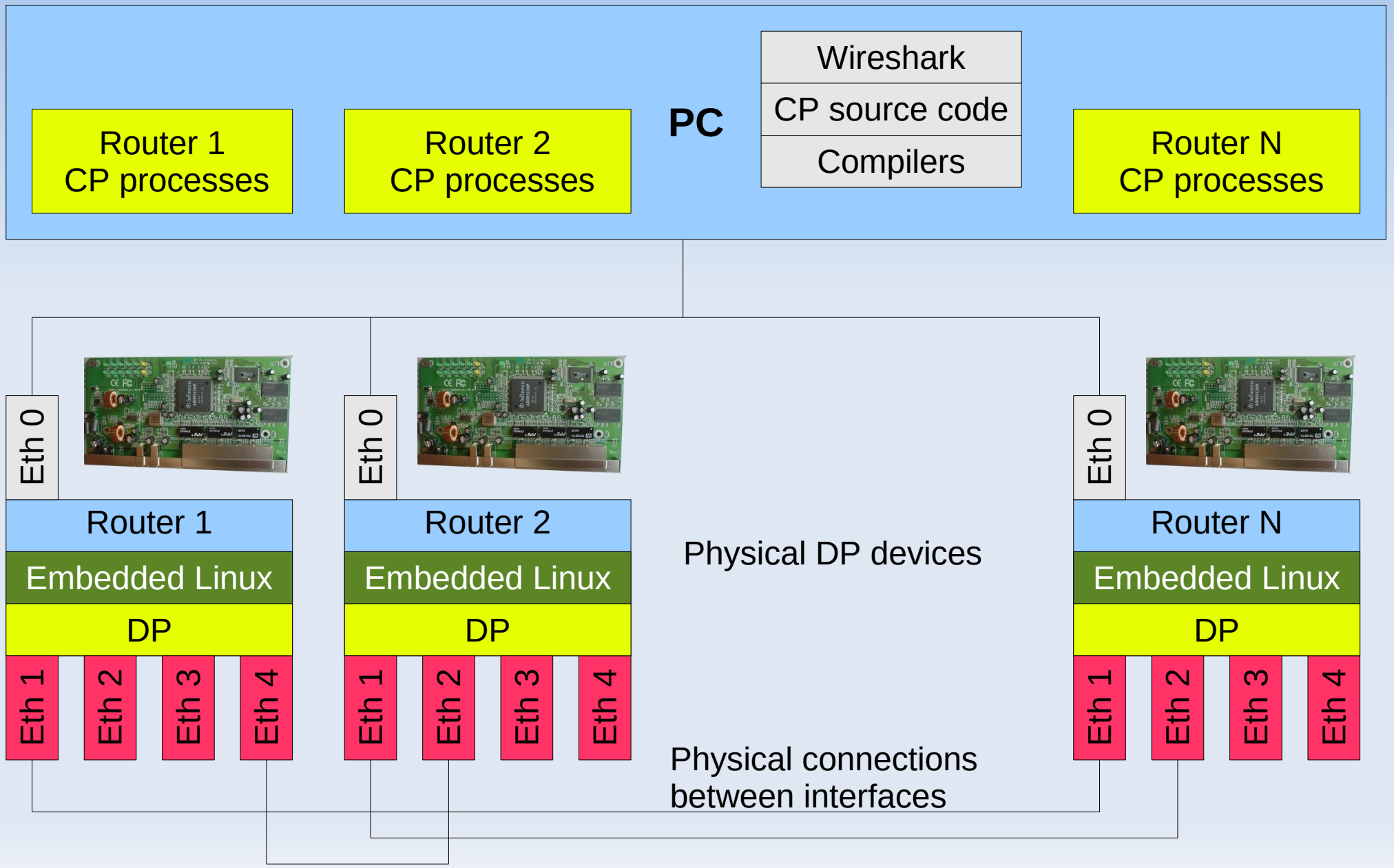
Sieć wirtualna

VM 1
Software Router
CP
DP
Eth 0
Eth 1
VM 2
Software Router
CP
DP
Eth 0
Eth 1
Virt. Bridge 1
Virt. Bridge 2
Virt. Bridge 3
Eth 0
Eth 1
VM 3
Software Router
CP
DP
Eth 0
Eth 1
VM N
Software Router
CP
DP

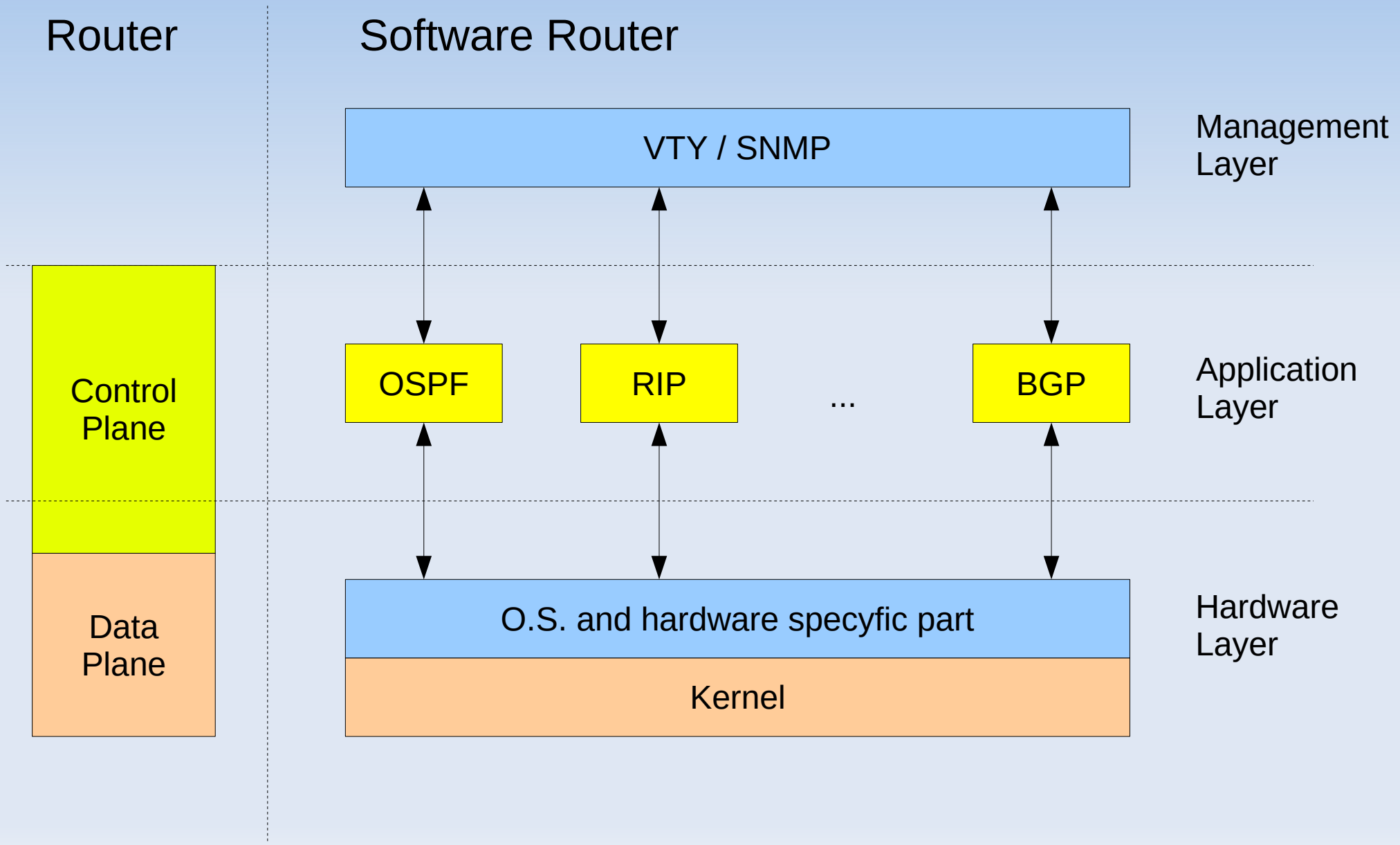
Przykładowy virtualny testbed



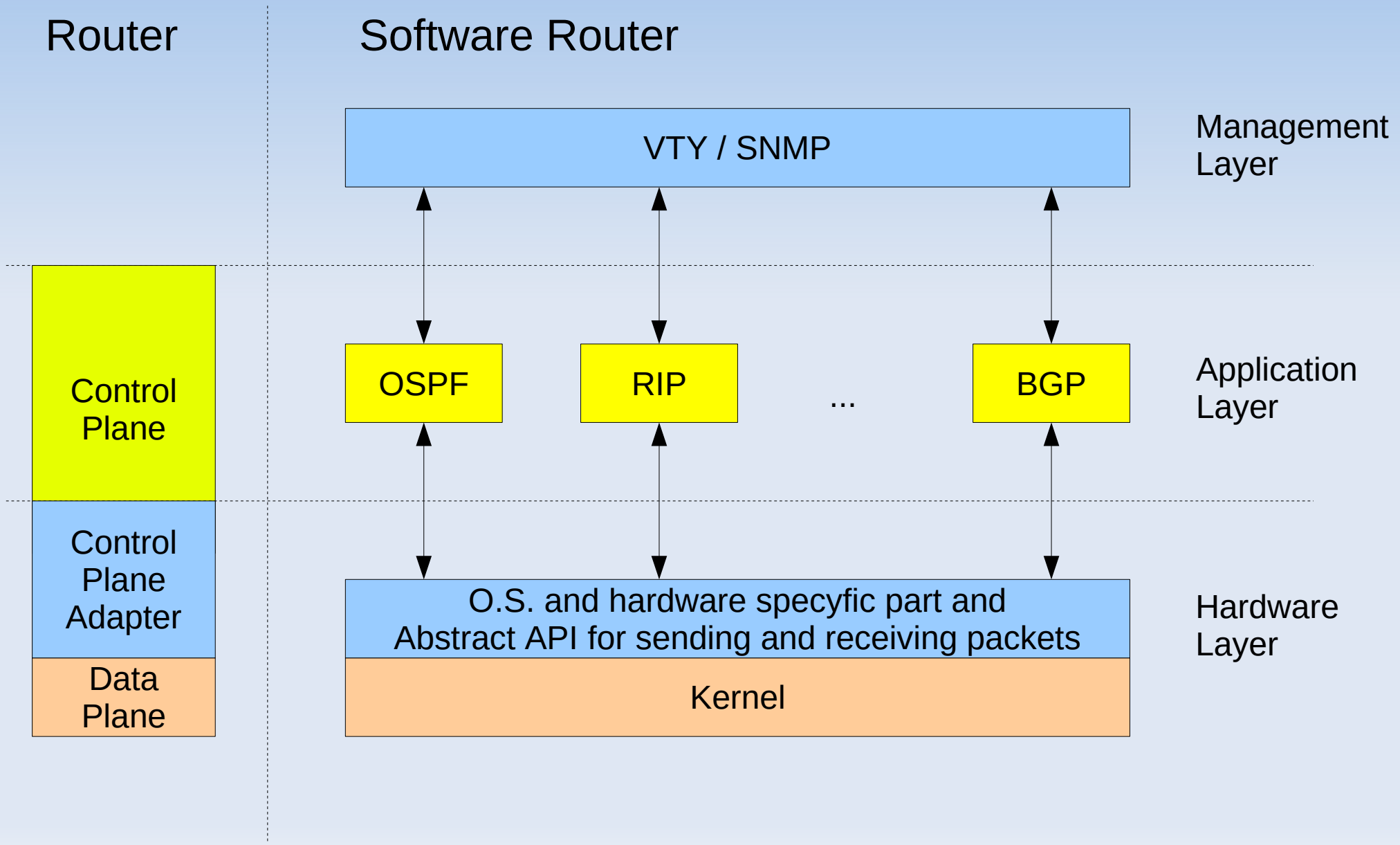
Rozwiązanie hybrydowe



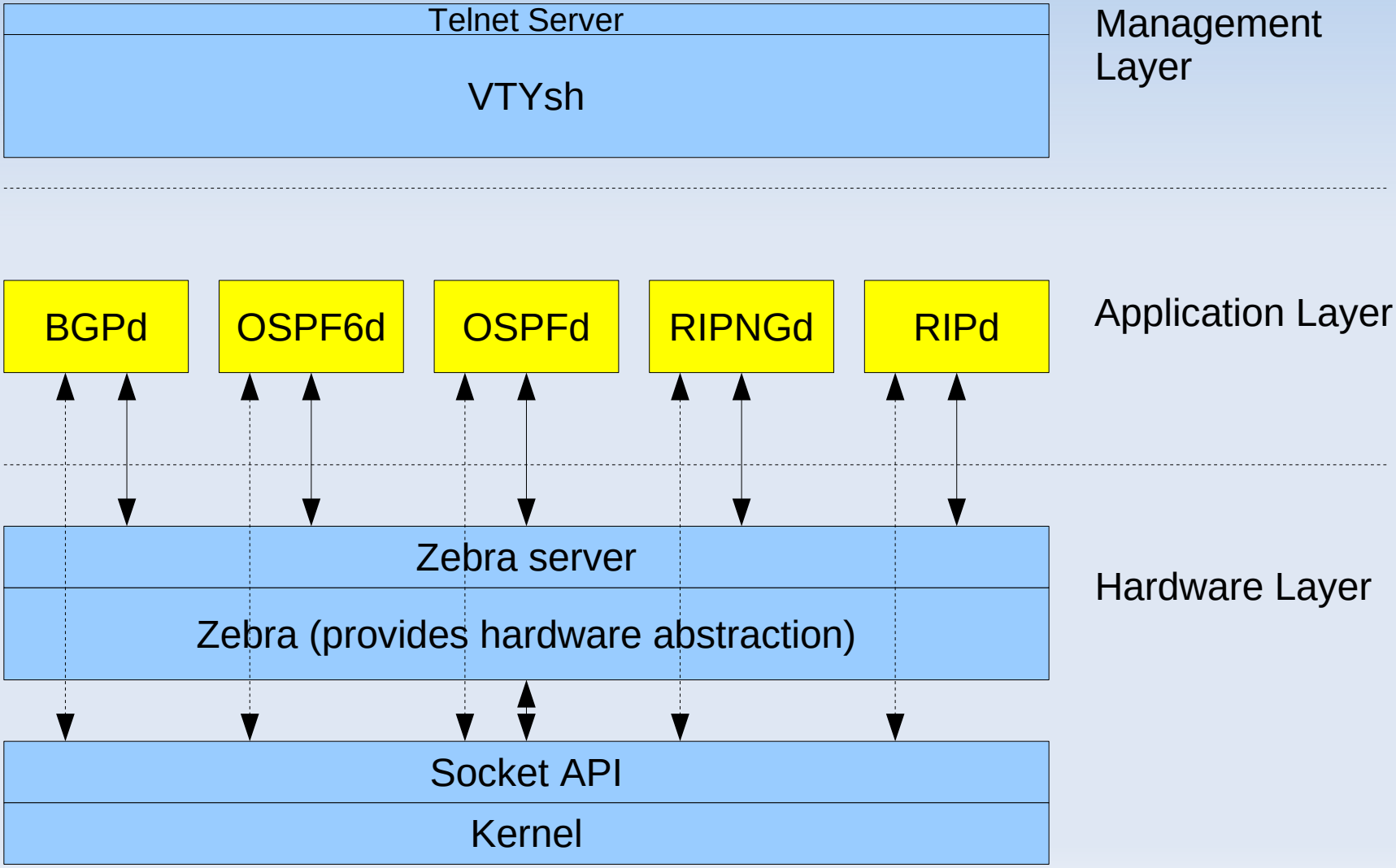
Rowziązanie sprzętowe vs programowe



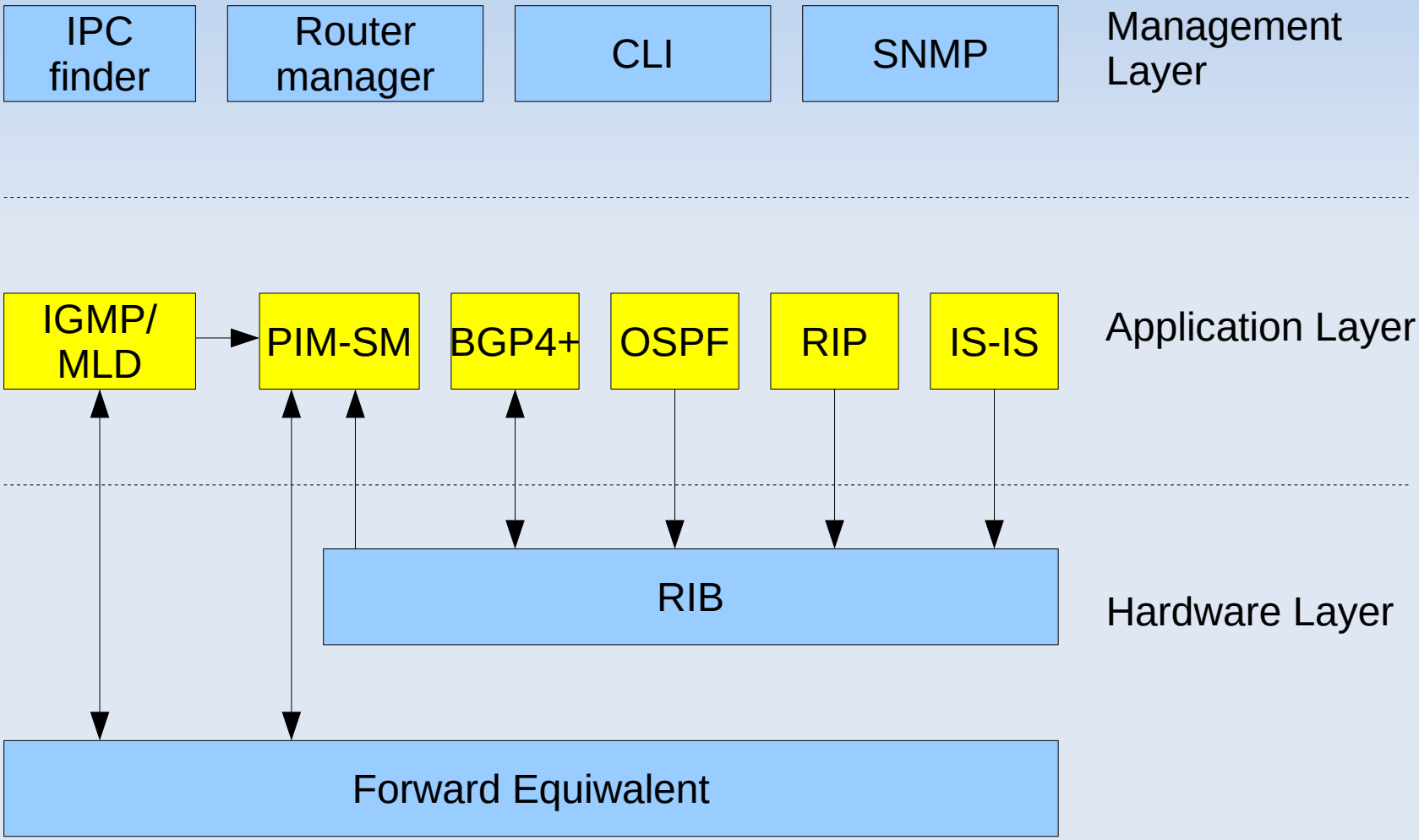
Router vs Software router architecture



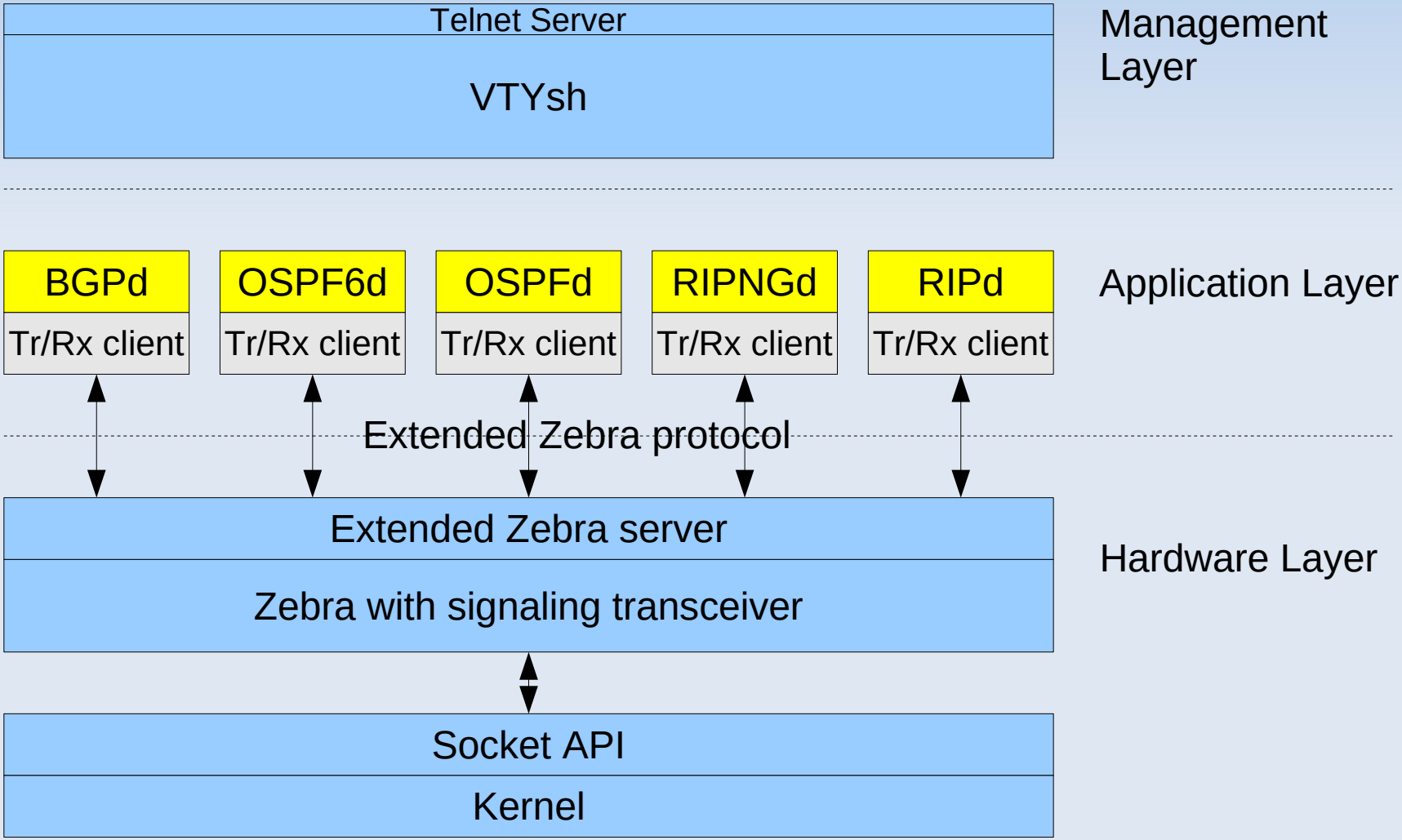
Architektura rutera programowego Quagga



Architektura rutera programowego XORP

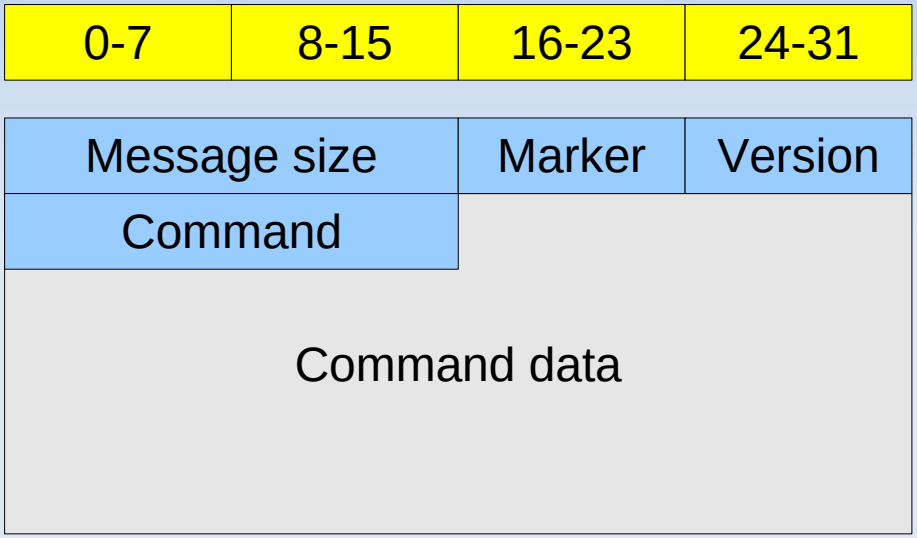


Modyfikacje rutera programowego Quagga



Protokół Zebra

Zebra protocol:



- INTERFACE_ADD
- INTERFACE_DELETE
- INTERFACE_ADDRESS_ADD
- INTERFACE_ADDRESS_DELETE
- INTERFACE_UP
- INTERFACE_DOWN

IPV4_ROUTE_ADD
IPV4_ROUTE_DELETE
IPV6_ROUTE_ADD
IPV6_ROUTE_DELETE
REDISTRIBUTE_ADD
REDISTRIBUTE_DELETE
REDISTRIBUTE_DEFAULT_ADD
REDISTRIBUTE_DEFAULT_DELETE
IPV4_NEXTHOP_LOOKUP
IPV6_NEXTHOP_LOOKUP
IPV4_IMPORT_LOOKUP
IPV6_IMPORT_LOOKUP
INTERFACE_RENAME
ROUTER_ID_ADD
ROUTER_ID_DELETE
ROUTER_ID_UPDATE

Sprzęt do budowy płaszczyzny danych

ADM5120

- MIPS CPU (up to 225 MHz)
- 16 MB RAM
- 2 MB ROM
- 5 portóœ ethenretowych
- 1 port konsolowy
- Niska cena

Implemetacja

- Open WRT
  - Ipv4 i Ipv6
  - Wersja Kamikadze
- Zmodyfikowany moduł zebra
  - Obsługa 2 dodatkowych wiadomości
  - Pakiety protokołów rutingu przesyłane za pośrednictwem modułu zebra
- Firmware
  - Wgrywany tylko raz.
  - Niezależny od protokołów rutingu.
  - Mały rozmiar najlepiej (1¾ MB)

Implementacja
Ruter programowy

- Quaga Framework
  - Napisana w C przez Yasuhiro Ohara,
  - Wysoka wydajność,
  - Kiepska dokumentacja,
  - Budowa modułowa, każdy protokół obsługiwany przez osobny moduł.
- Własna implementacja
  - Dowolny język programowania np RUST lub C#.
  - Wymagana implementacja protokołu ZEBRA.

Scenariusze użycia
Przechwytywanie pakietów

Router 1
CP processes

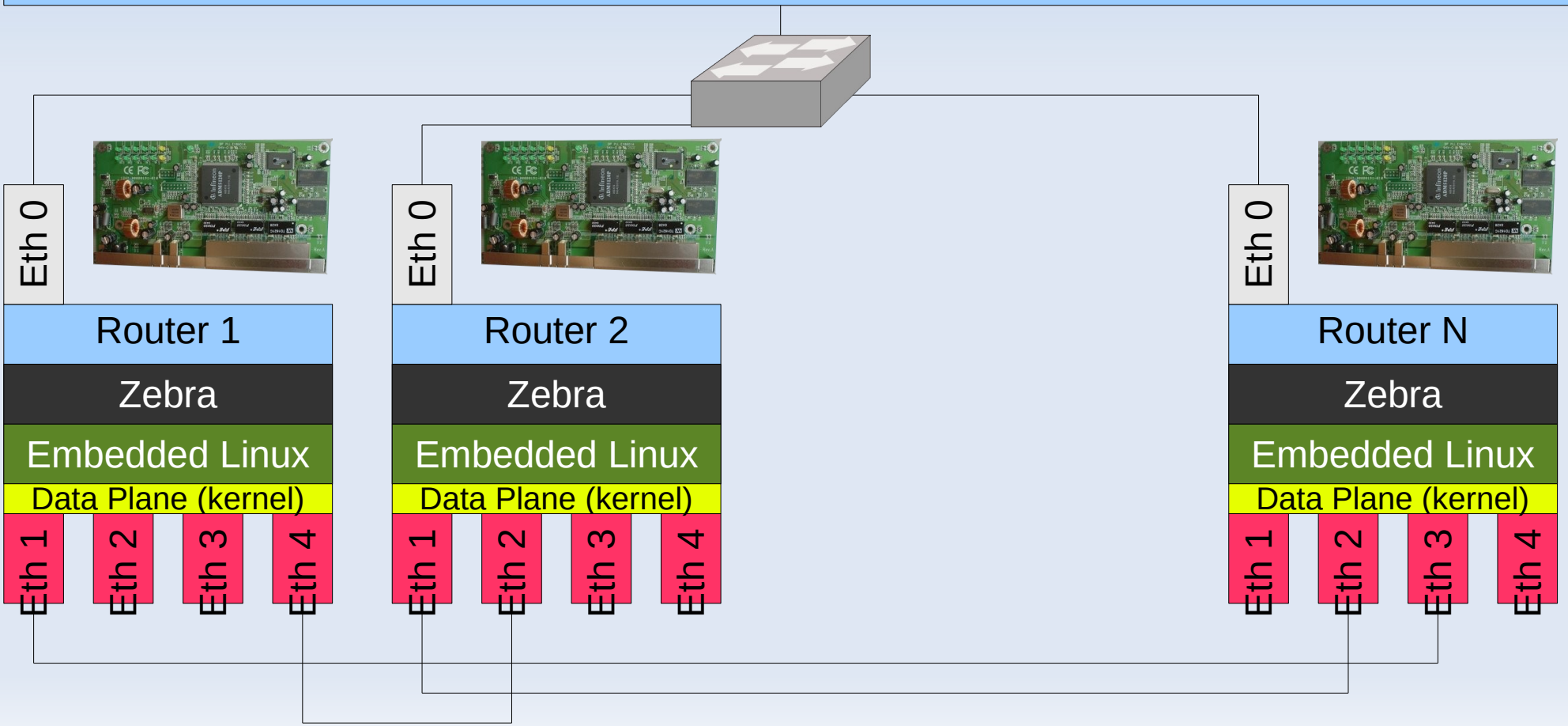
Router 2
CP processes

PC

Router N
CP processes

Ospf hello

Router 1 wants to send Ospf hello packet via interface 1



Scenariusze użycia
Przechwytywanie pakietów

Router 1
CP processes

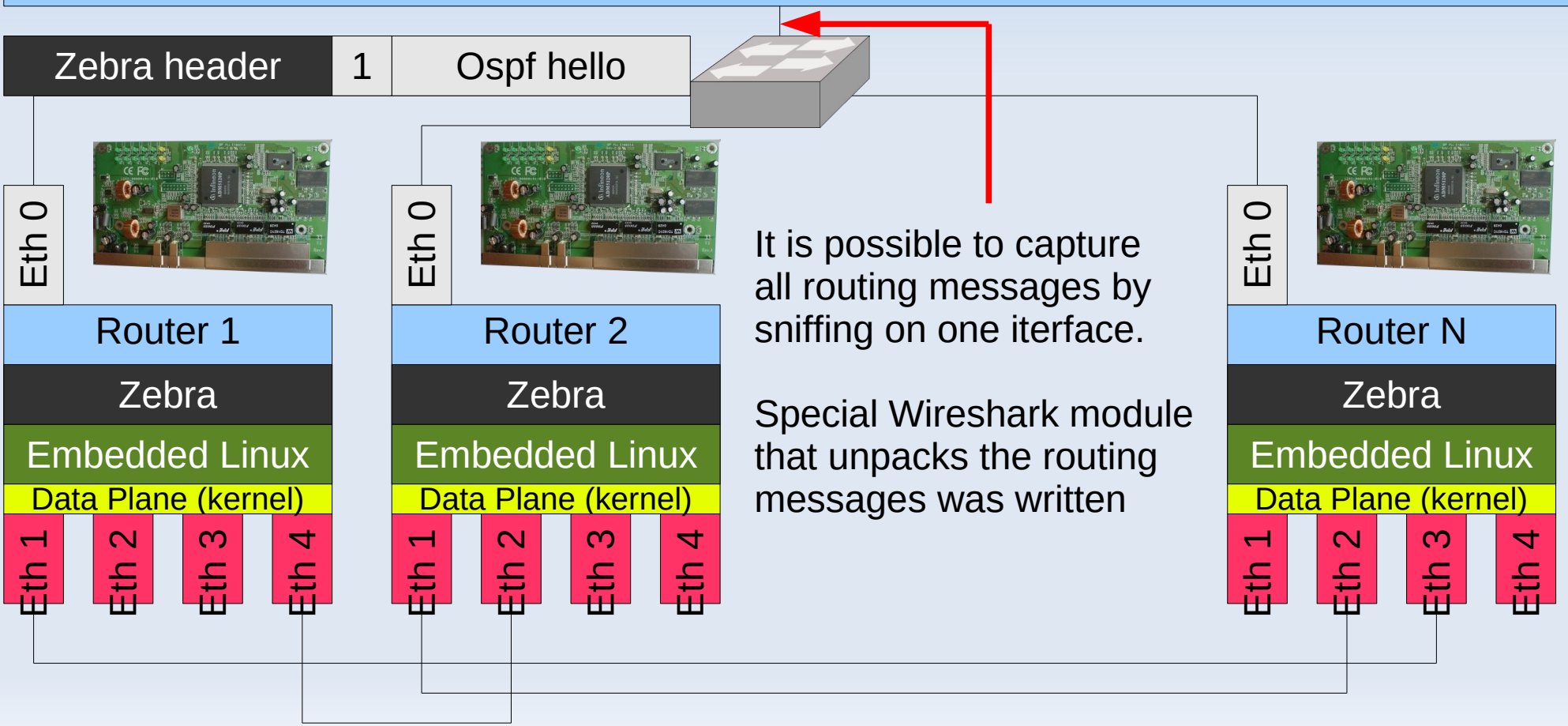
Router 2
CP processes

PC

Router N
CP processes

Router 1 wants to send Ospf hello packet via interface 1

Hello message is encapsulated into Zebra message. Additional info about interface number



It is possible to capture all routing messages by sniffing on one iterface.

Special Wireshark module that unpacks the routing messages was written

Scenariusze użycia
Przechwytywanie pakietów

Router 1
CP processes

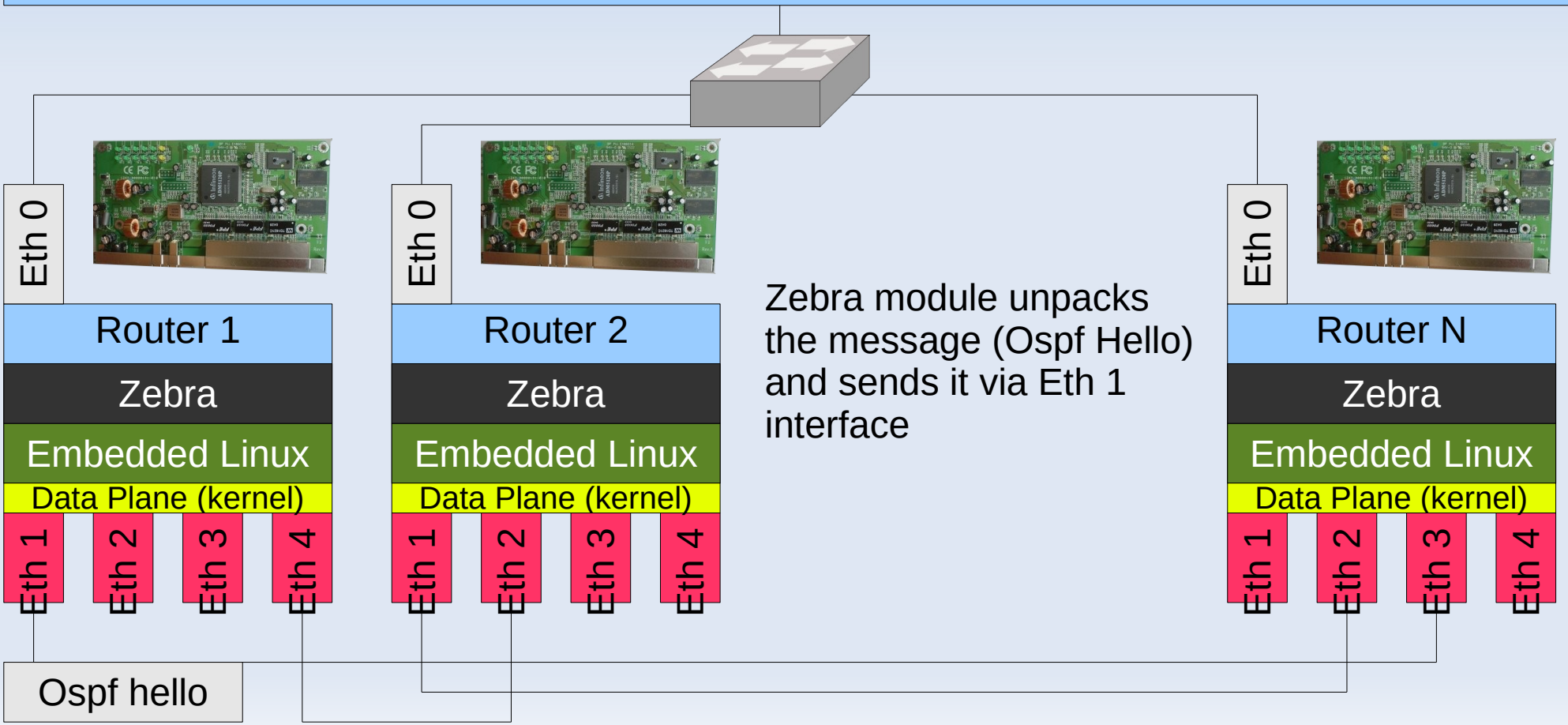
Router 2
CP processes

PC

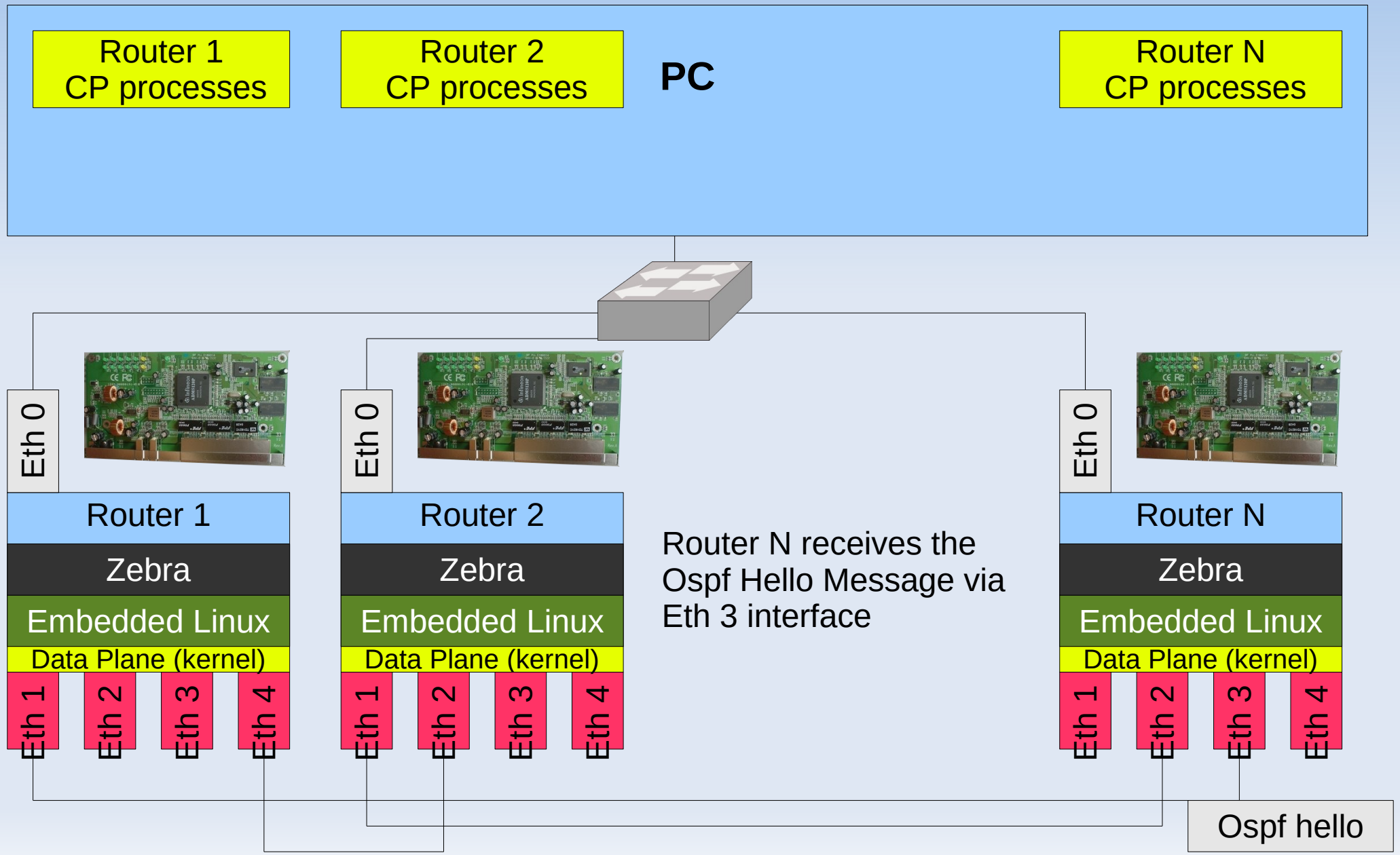
Router N
CP processes

Router 1 wants to send Ospf hello packet via interface 1

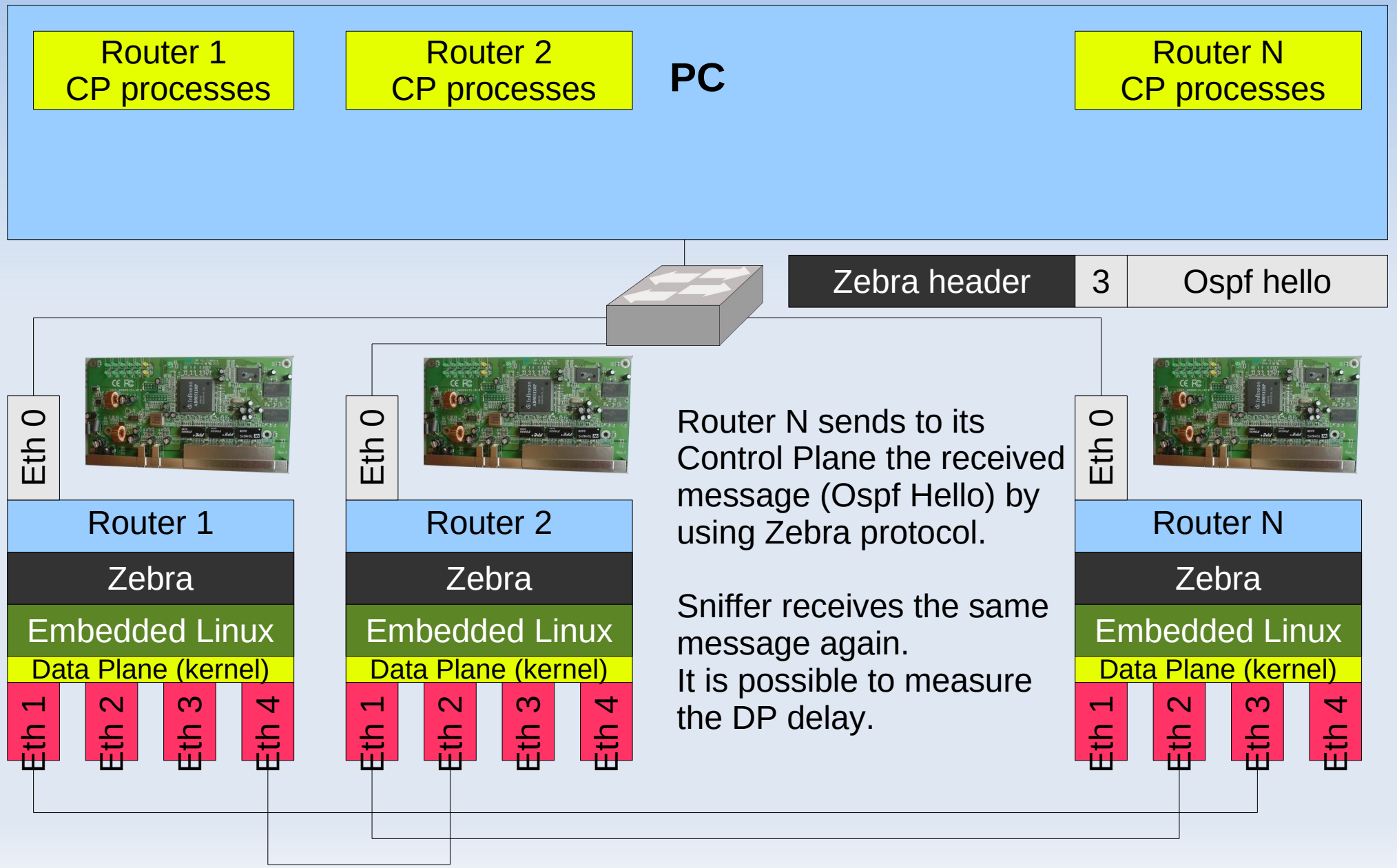
Hello message is encapsulated into Zebra message. Additional info about interface number



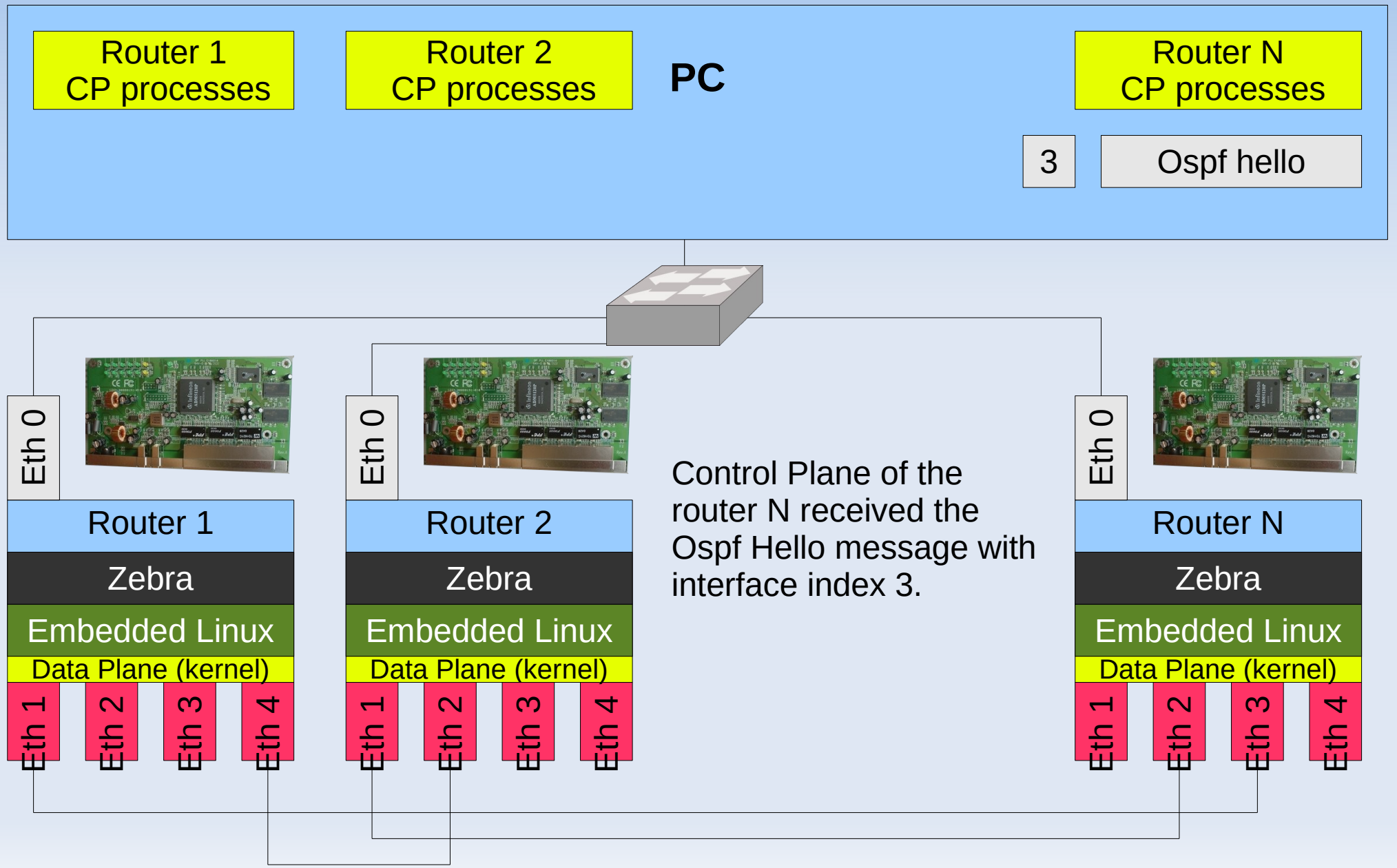
Scenariusze użycia
Przechwytywanie pakietów



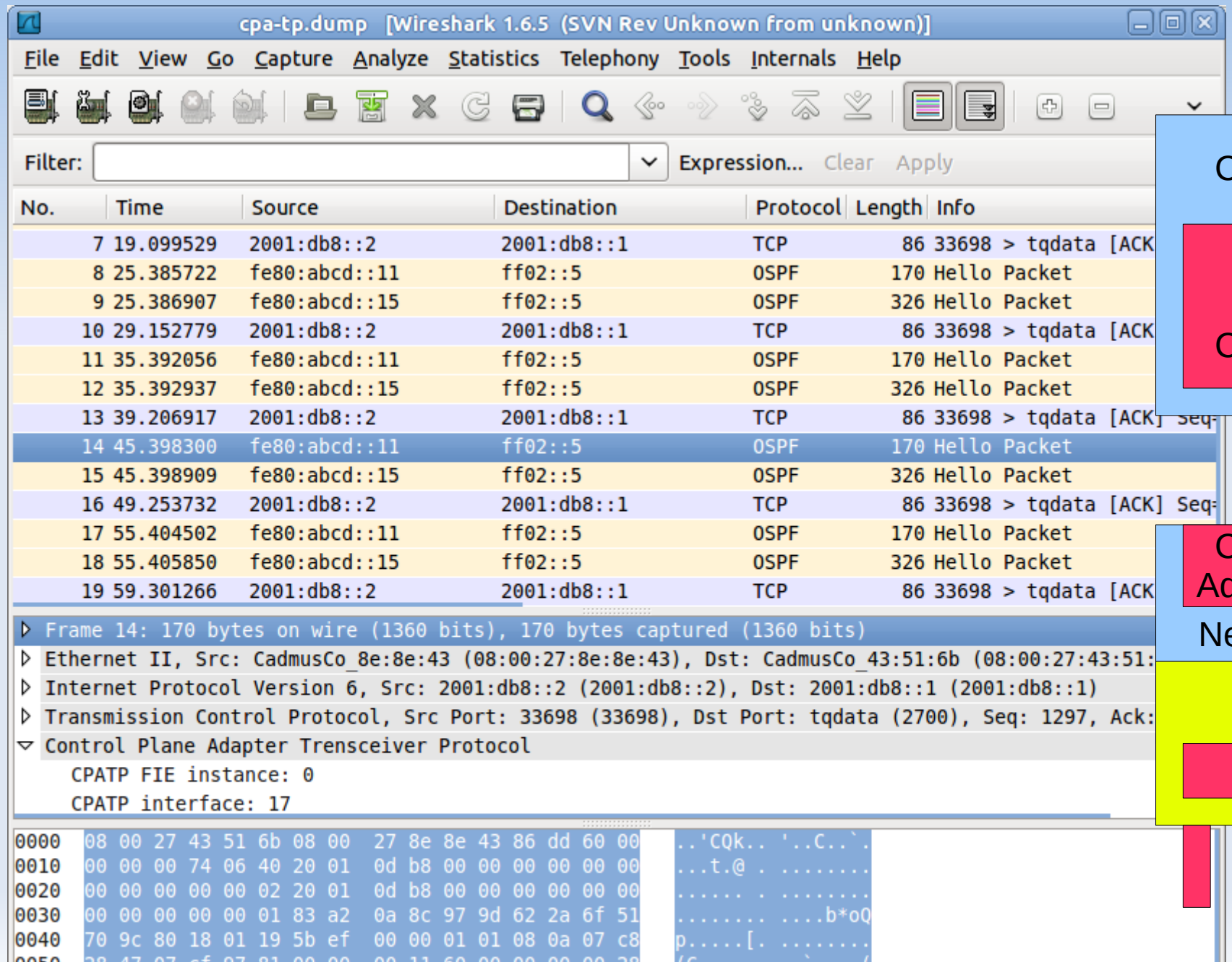
Scenariusze użycia
Przechwytywanie pakietów

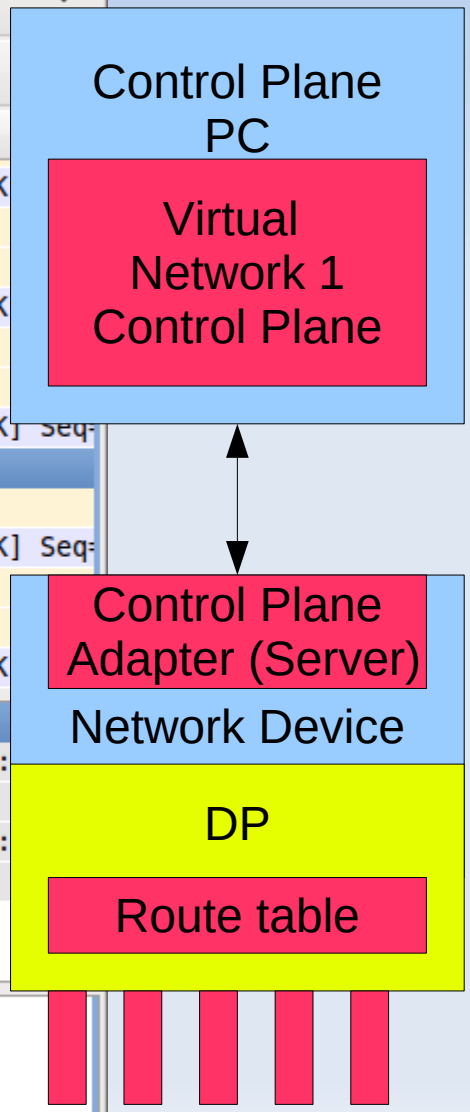


Scenariusze użycia
Przechwytywanie pakietów



Scenariusze użycia Wireshark





Podsumowanie

- Rozwiązanie przyjazne dla użytkownika
  - Brak konieczności stosowania skomplikowanych narzędzi typu devop
- By zmienić implementację protokołu, wystarczy restart oprogramowania działającego na maszynie obsługującej płaszczyznę sterującą
- Węzły są fizycznie odseparowane i mają fizyczne połączenia
- Wystarczy nasłuch w jednym punkcie by zaobserwwać wymianę wszystkich pakietów od protokołów rutingu.

Podsumowanie

- Zaimplementowano
  - Moduł Zebra
  - Ospf V2
  - Ospf V3
- Strona projektu:
  - https://github.com/adamkaliszan/SdnQuagga
- Rozwiązanie podobne do Software Defined Networks
- Możliwość zastosowania nowocześniejszego sprzętu do obsługi płaszczyzny danych
  - Banana Pi R2