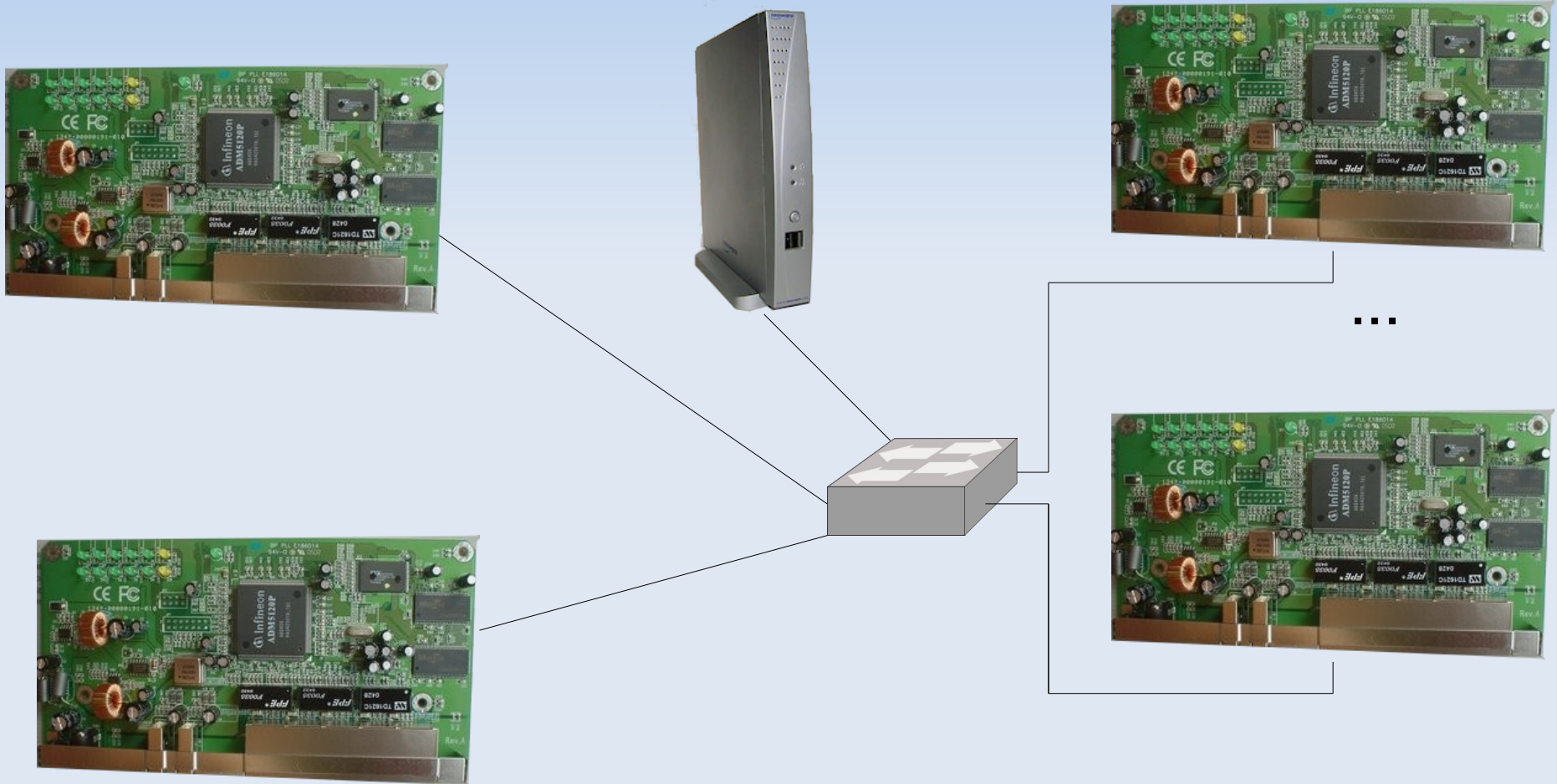


Platforma do testowania i rozwoju nowych protokołów rutingu



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

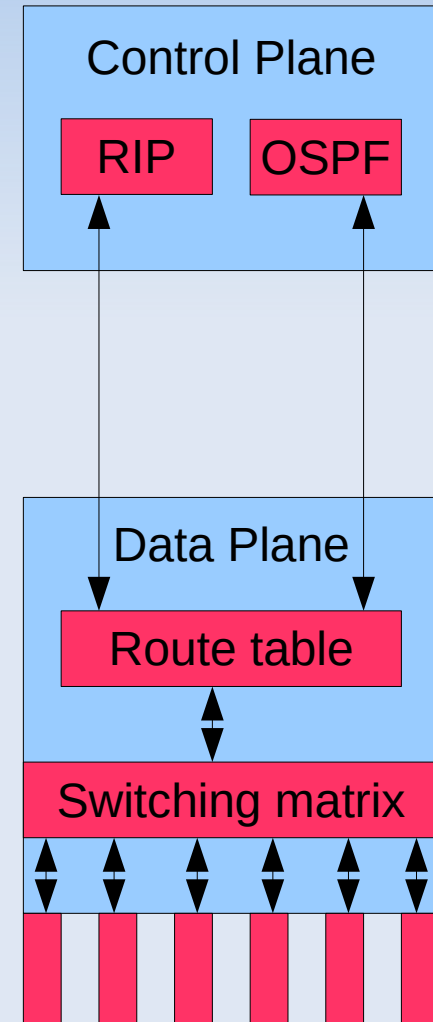
Źródła inspiracji

- 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

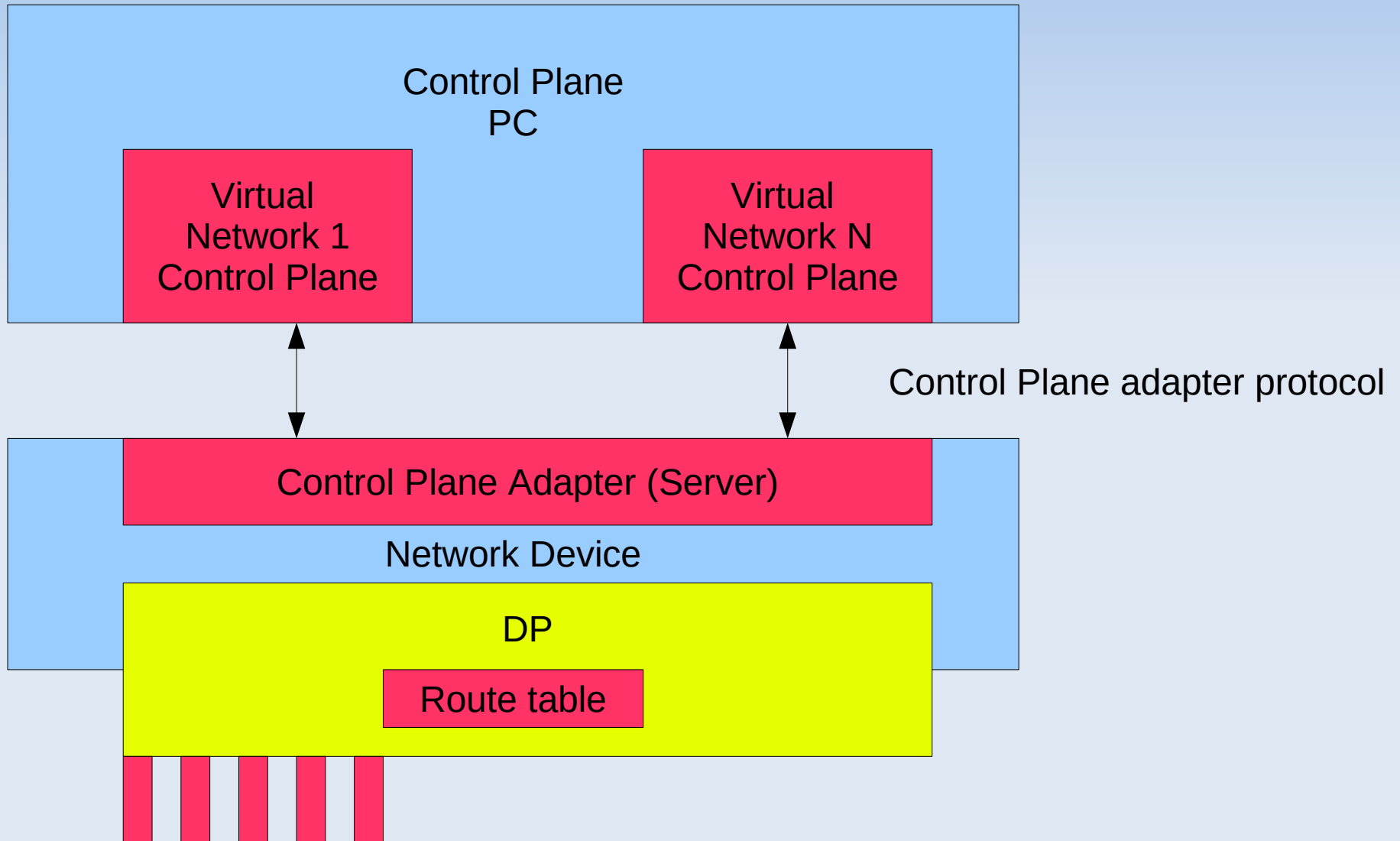
Udział w projektach badawczych

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



Źródła Inspiracji

Udział w projektach badawczych



Rozwiązanie sprzętowe vs symulator

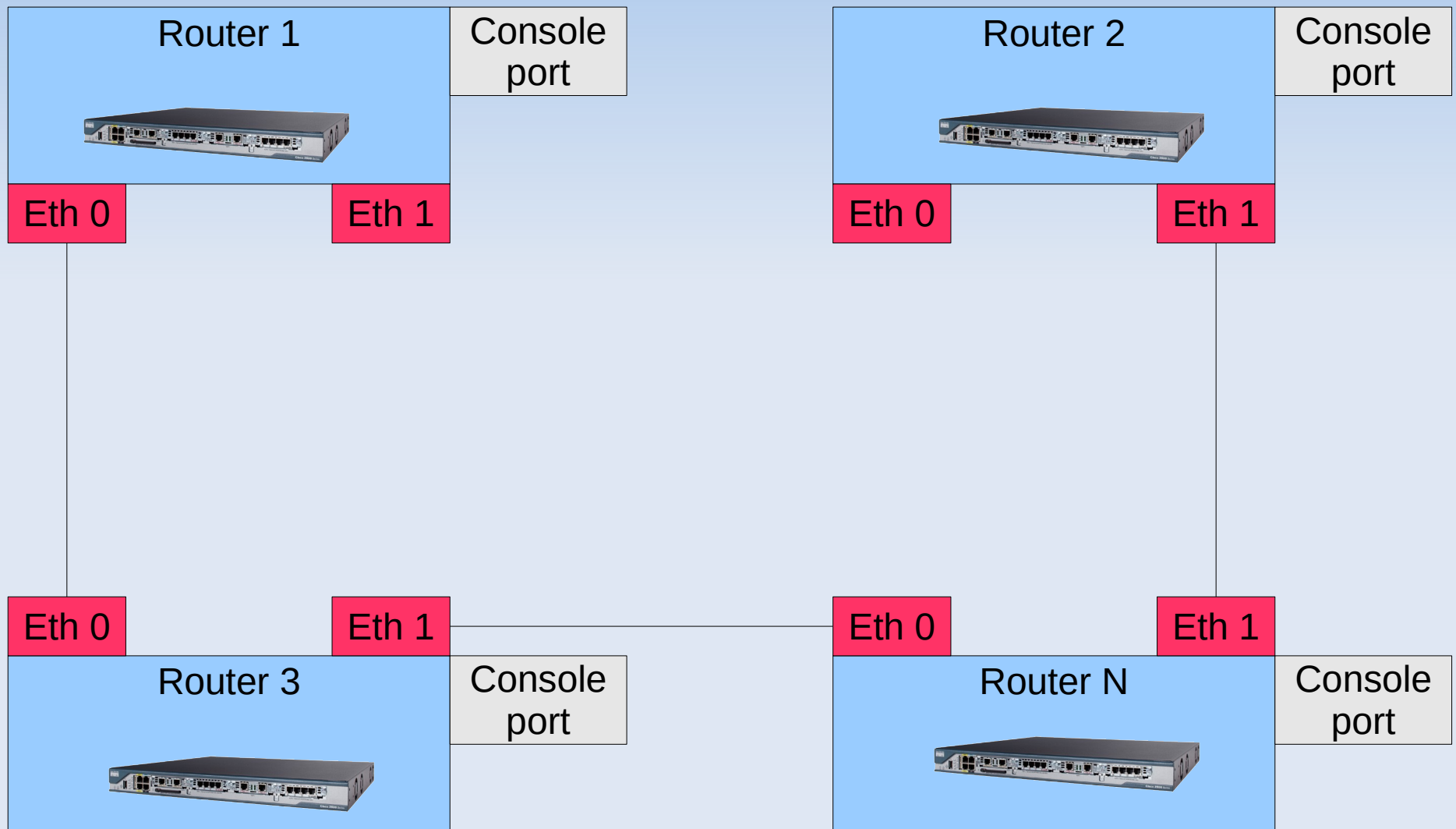
Hardware Testbed:

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

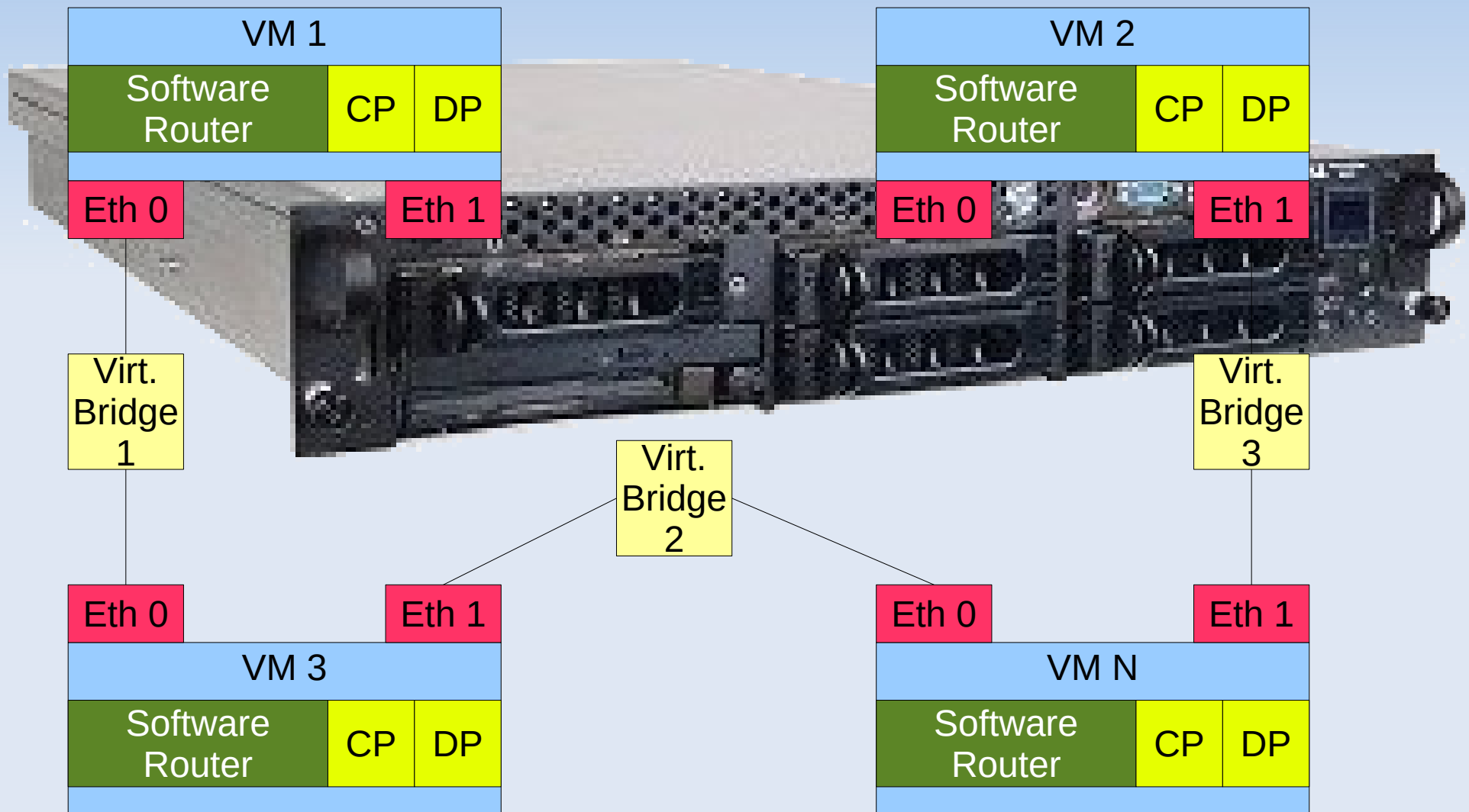
Virtual Testbed:

- 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

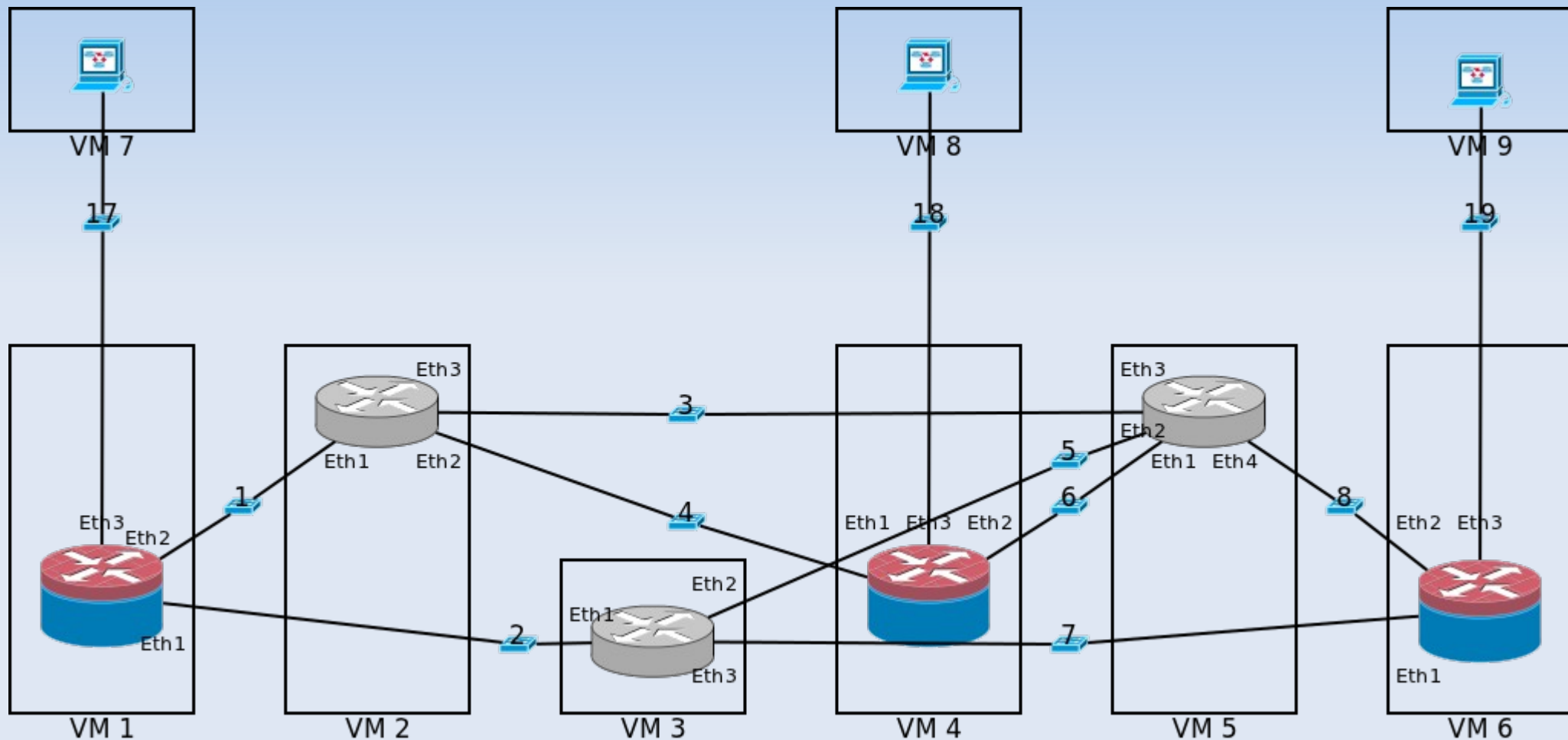
Rozwiązanie sprzętowe



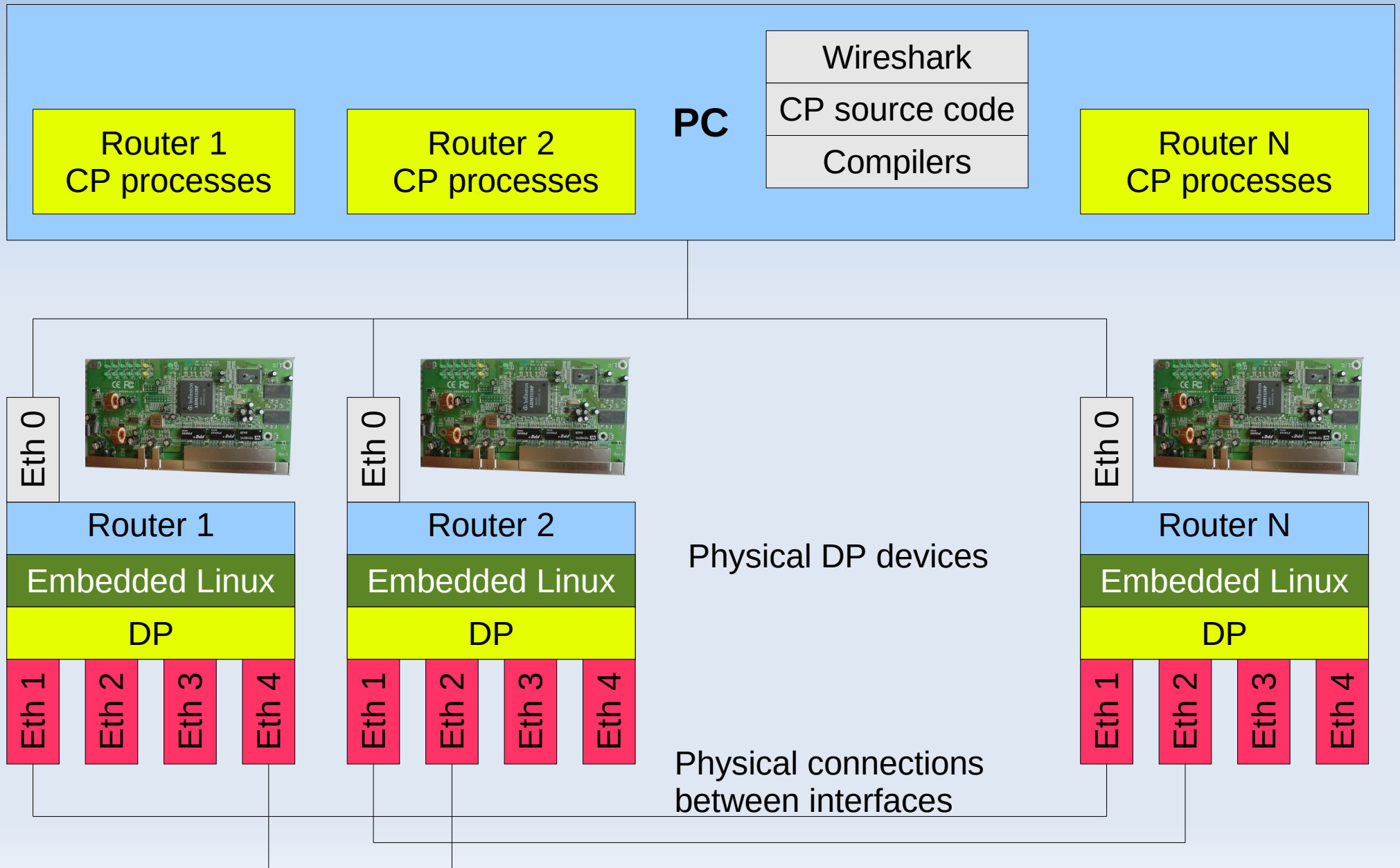
Sieć wirtualna



Przykładowy virtualny testbed



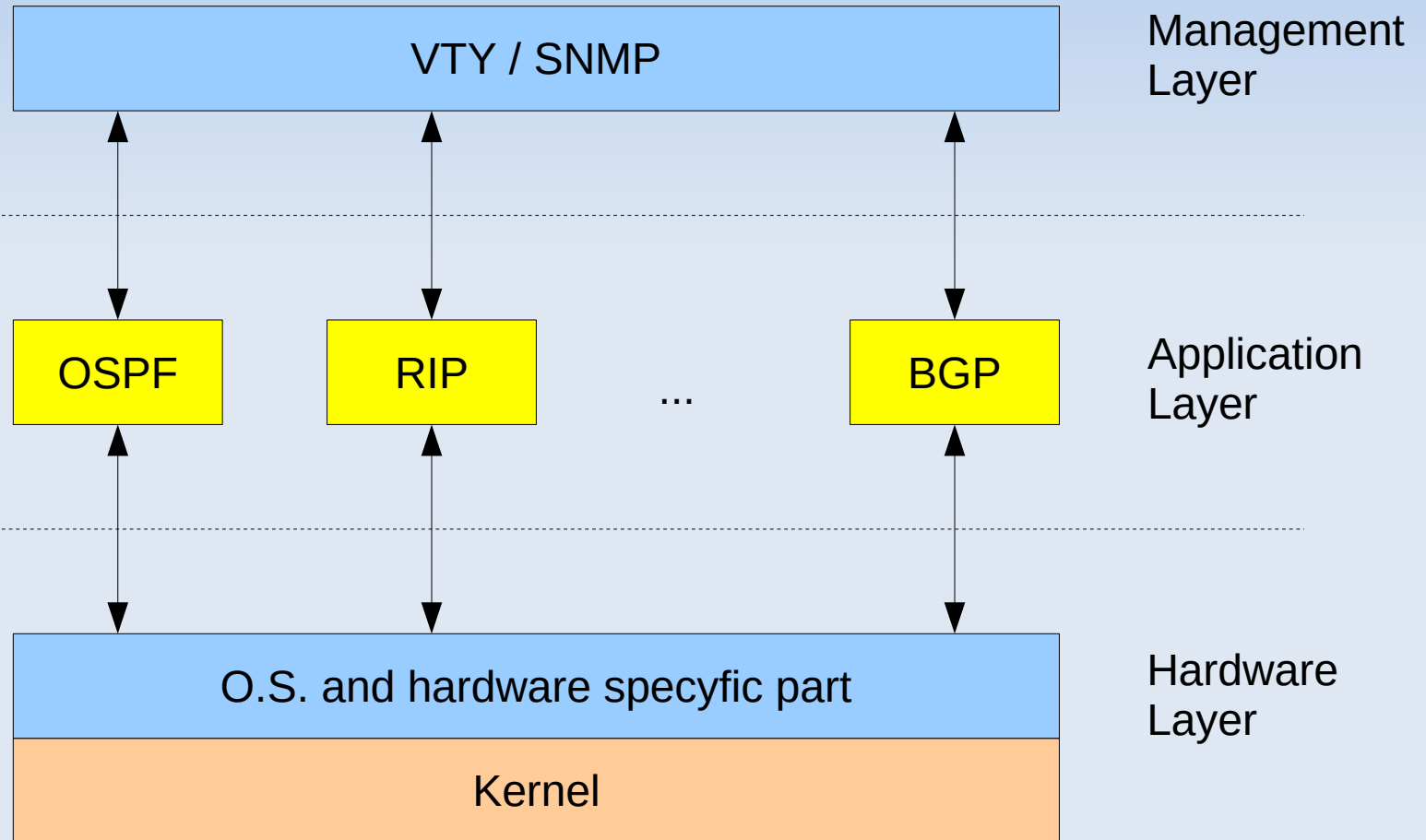
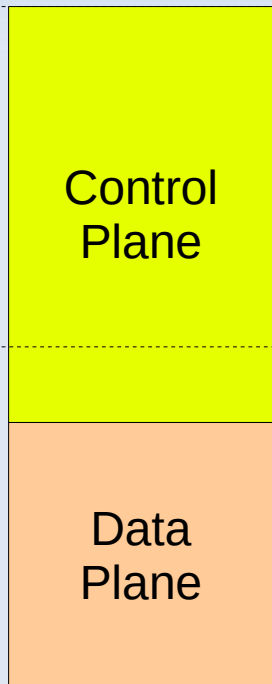
Rozwiązanie hybrydowe



Rowziązanie sprzętowe vs programowe

Router

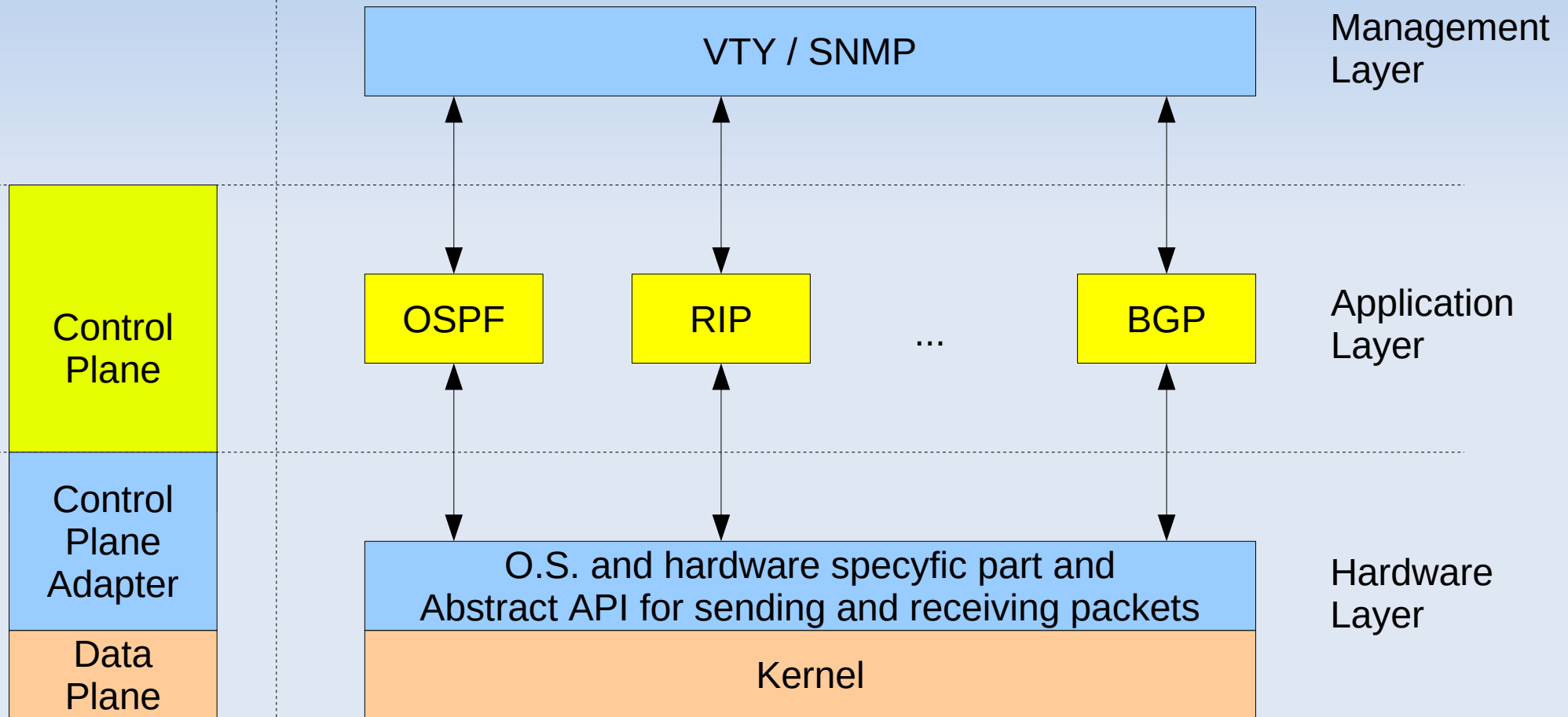
Software Router



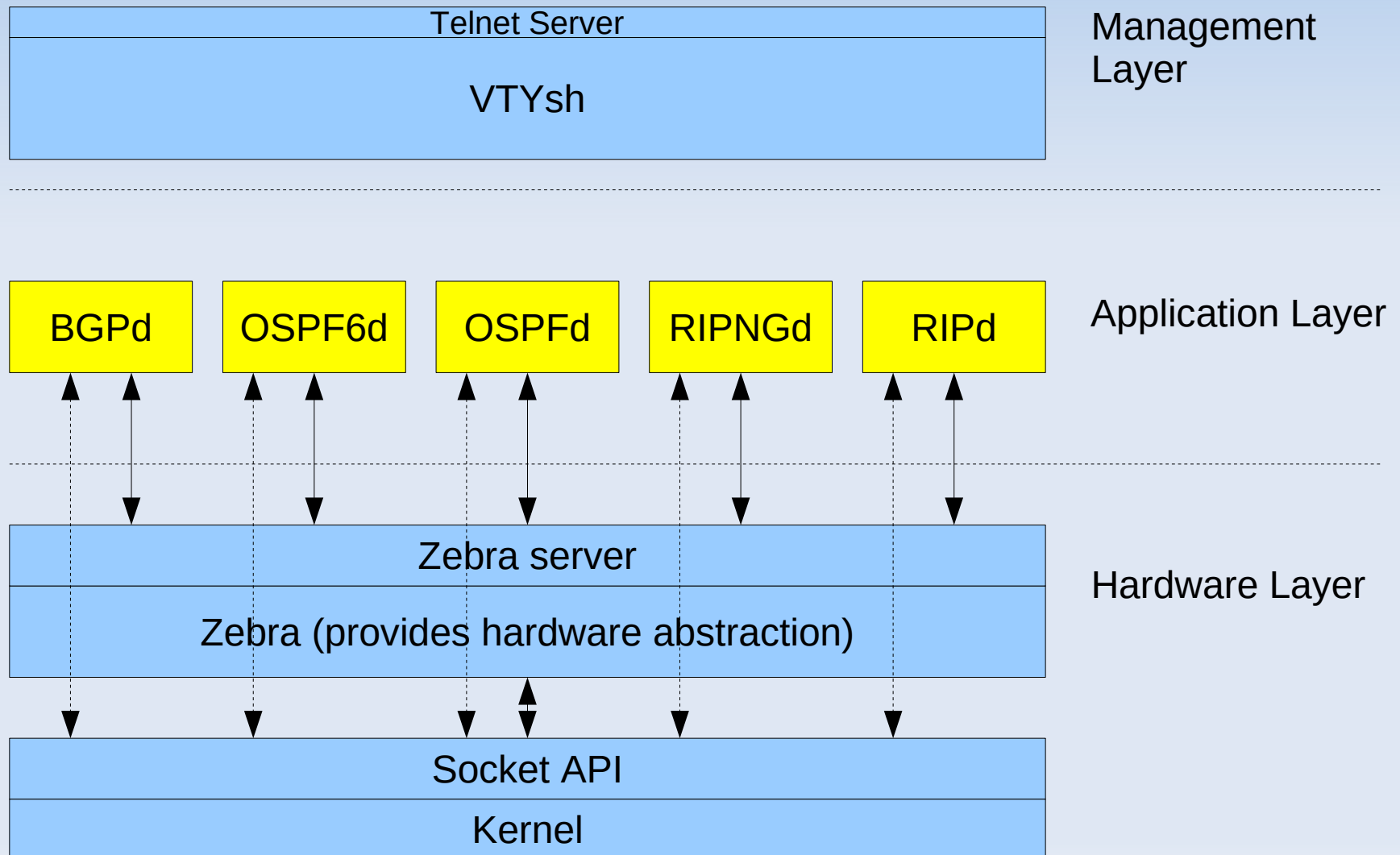
Router vs Software router architecture

Router

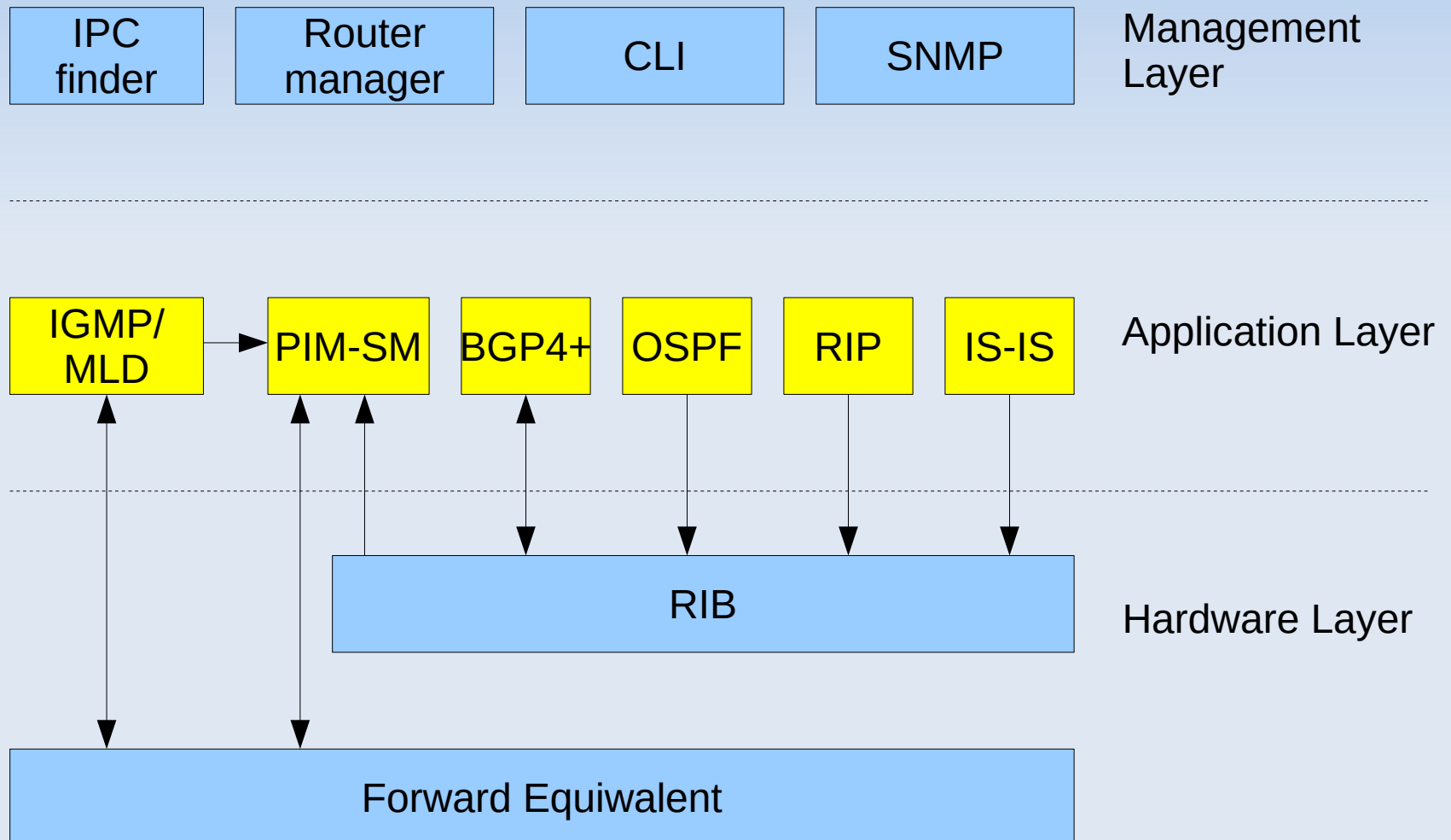
Software Router



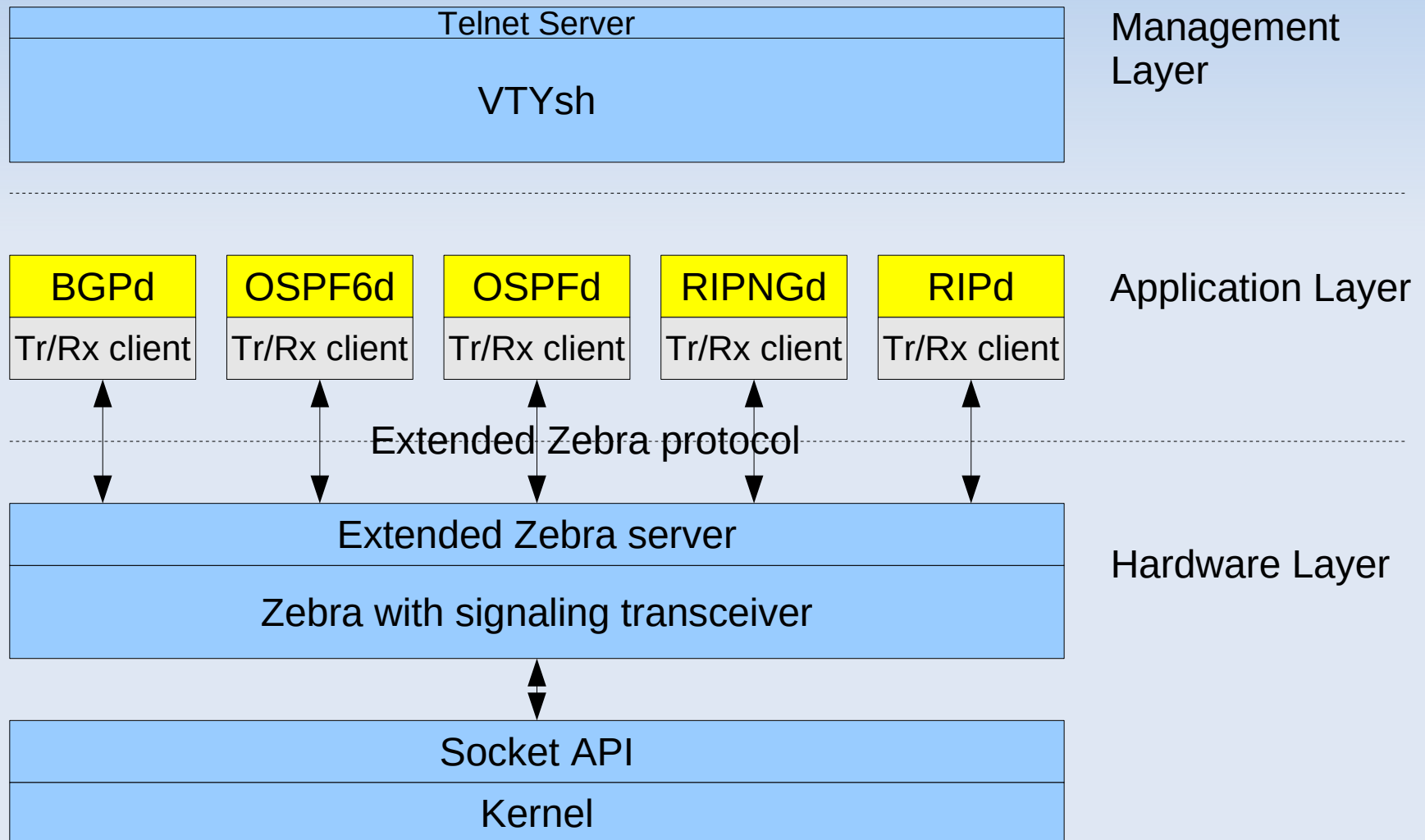
Architektura rutera programowego Quagga



Architektura routera programowego XORP



Modyfikacje rutera programowego Quagga



Protokół Zebra

Zebra protocol:

0-7	8-15	16-23	24-31
Message size	Marker	Version	
Command			
Command data			

- 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 portów ethernetowych
- 1 port konsolowy
- Niska cena



Implementacja

- 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 (1³/₄ 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

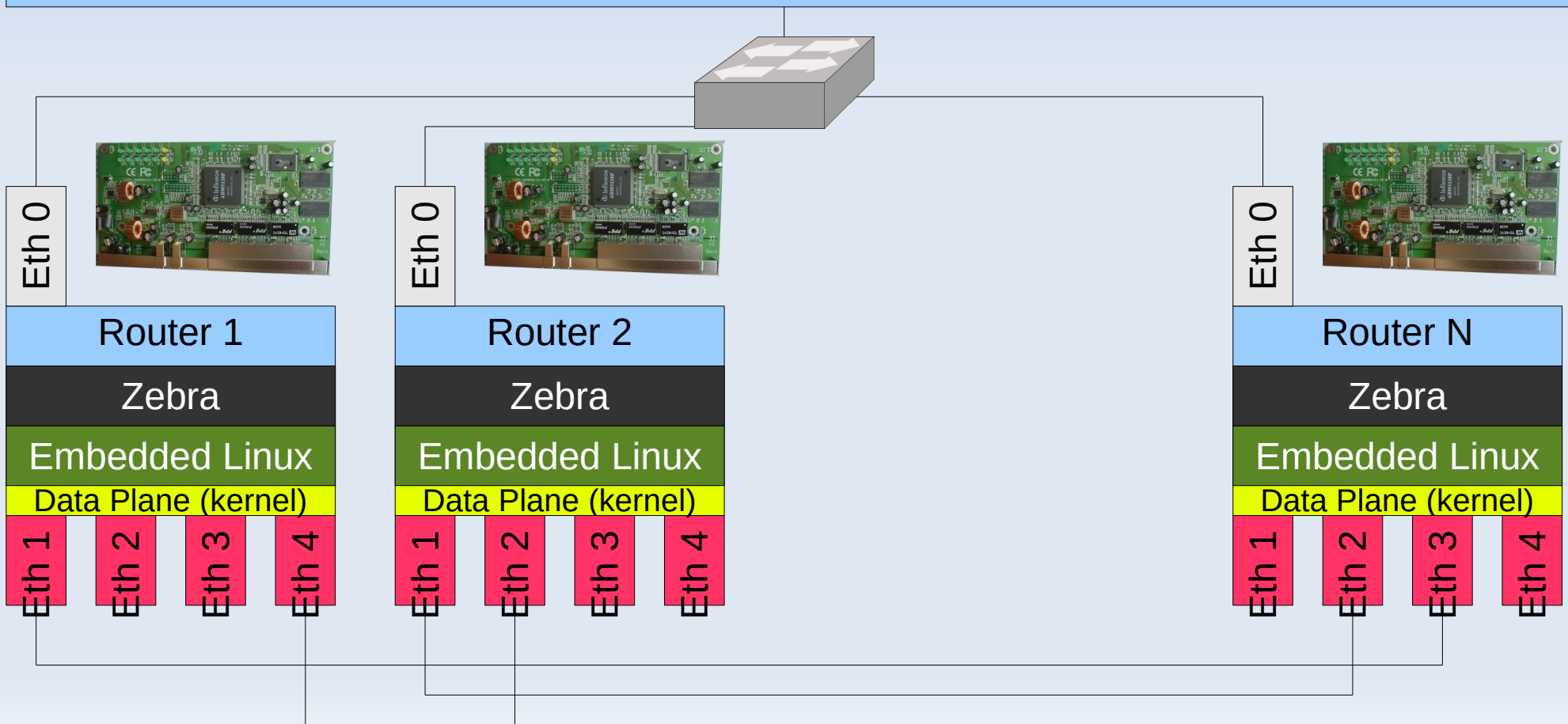
Router 2
CP processes

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

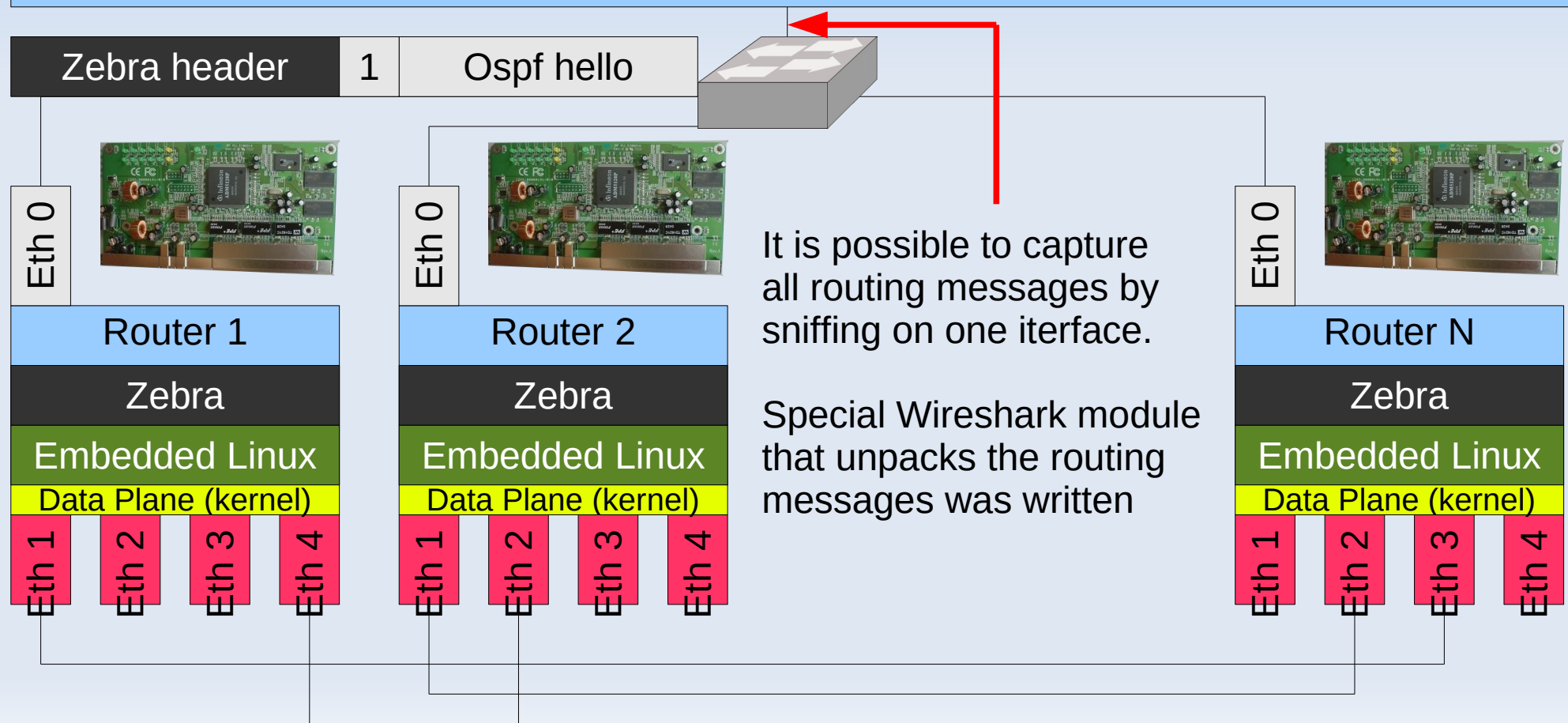
Router 2
CP processes

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



Scenariusze użycia

Przechwytywanie pakietów

Router 1
CP processes

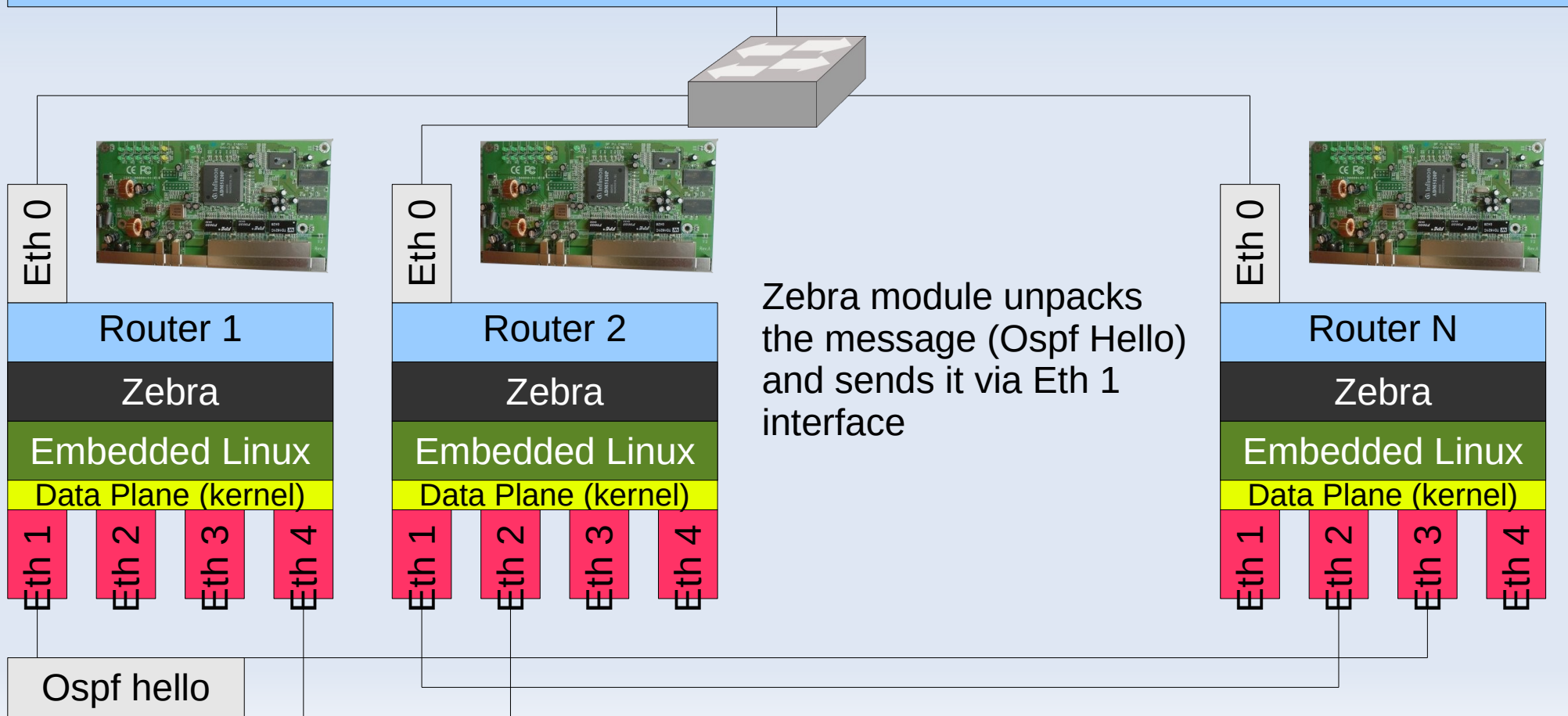
Router 2
CP processes

PC

Router N
CP processes

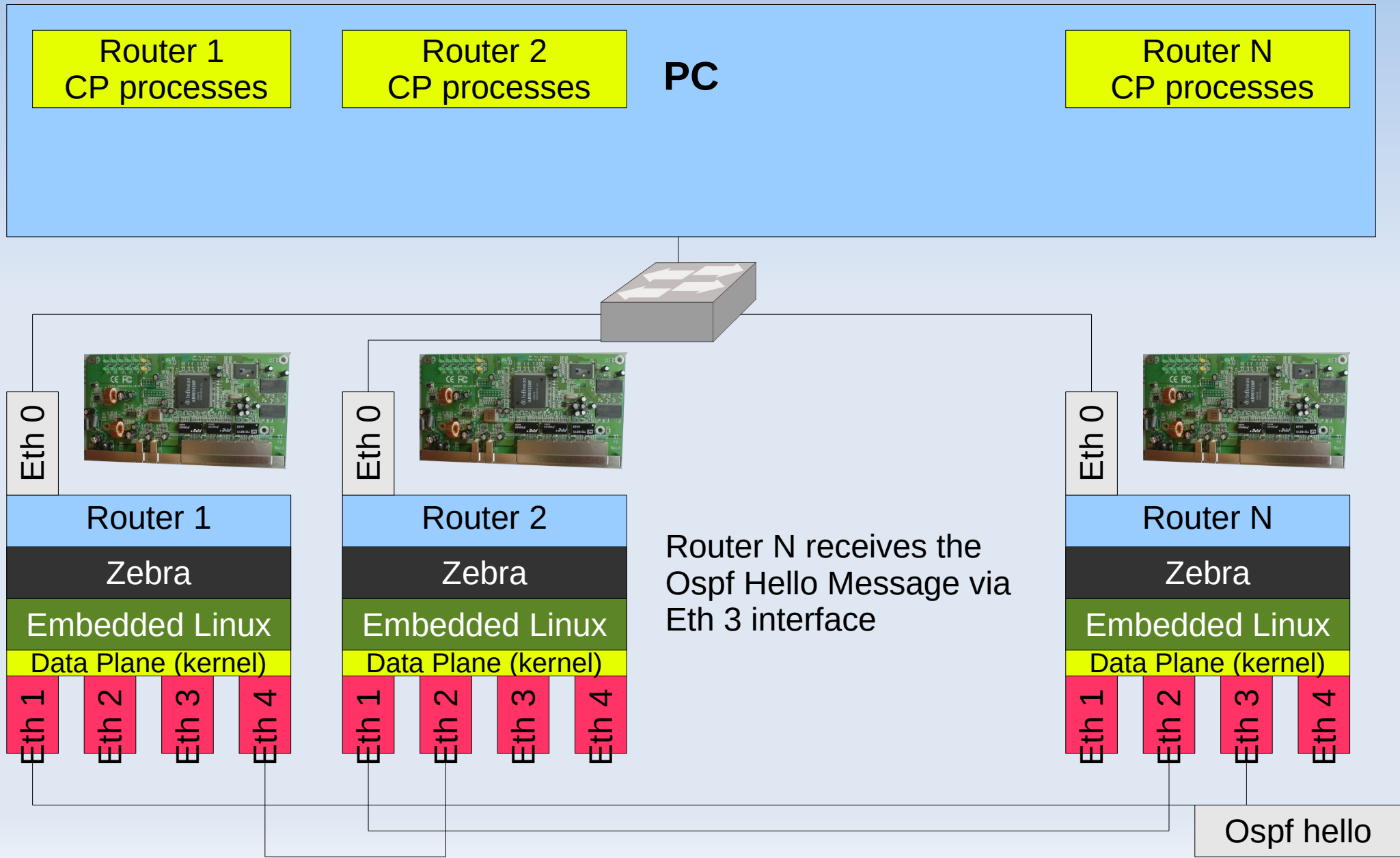
Router 1 wants to send Ospf hello packet via interface 1

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



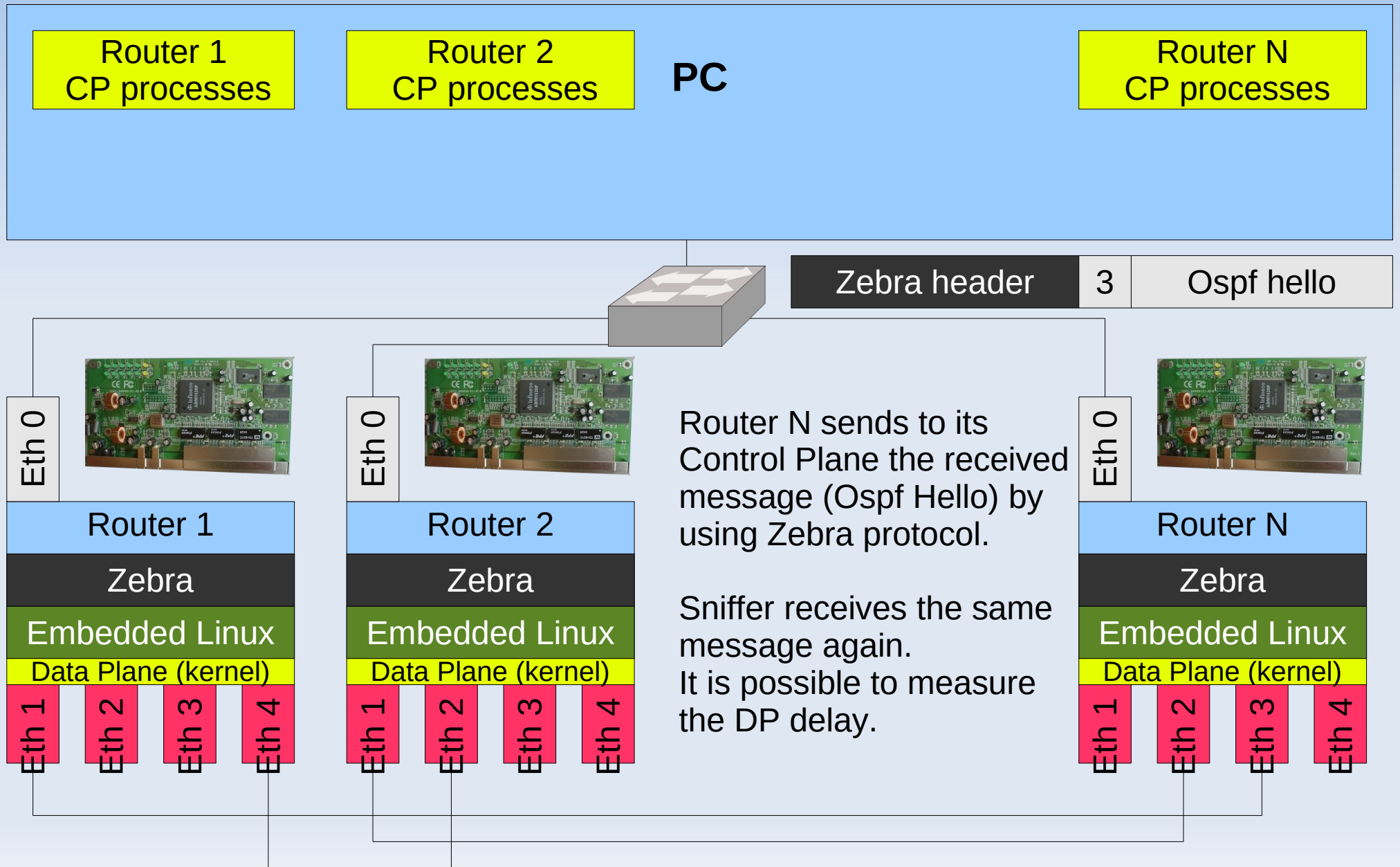
Scenariusze użycia

Przechwytywanie pakietów



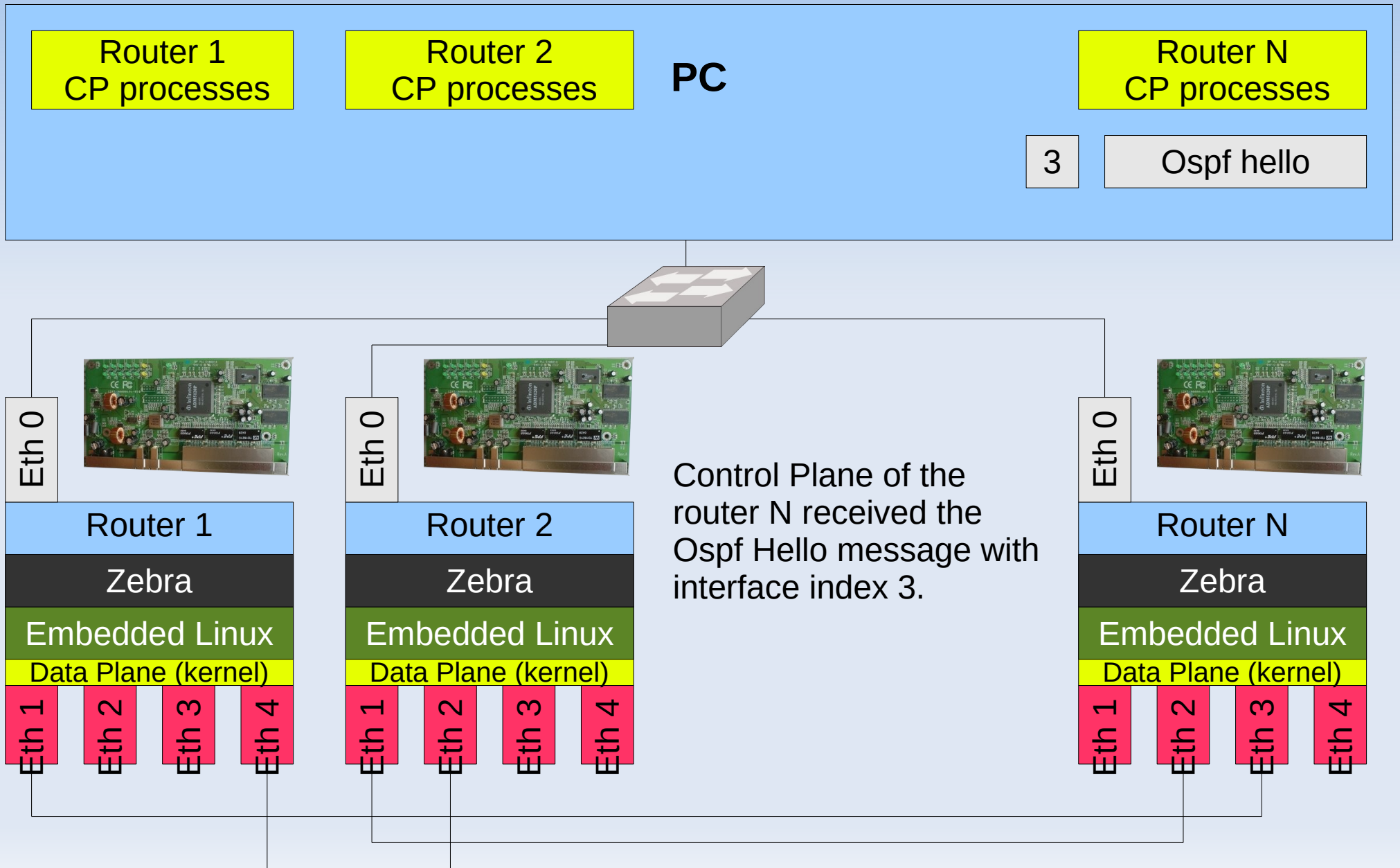
Scenariusze użycia

Przechwytywanie pakietów



Scenariusze użycia

Przechwytywanie pakietów



Scenariusze użycia Wireshark

cpa-tp.dump [Wireshark 1.6.5 (SVN Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
7	19.099529	2001:db8::2	2001:db8::1	TCP	86	33698 > tqdata [ACK]
8	25.385722	fe80:abcd::11	ff02::5	OSPF	170	Hello Packet
9	25.386907	fe80:abcd::15	ff02::5	OSPF	326	Hello Packet
10	29.152779	2001:db8::2	2001:db8::1	TCP	86	33698 > tqdata [ACK]
11	35.392056	fe80:abcd::11	ff02::5	OSPF	170	Hello Packet
12	35.392937	fe80:abcd::15	ff02::5	OSPF	326	Hello Packet
13	39.206917	2001:db8::2	2001:db8::1	TCP	86	33698 > tqdata [ACK] Seq-
14	45.398300	fe80:abcd::11	ff02::5	OSPF	170	Hello Packet
15	45.398909	fe80:abcd::15	ff02::5	OSPF	326	Hello Packet
16	49.253732	2001:db8::2	2001:db8::1	TCP	86	33698 > tqdata [ACK] Seq-
17	55.404502	fe80:abcd::11	ff02::5	OSPF	170	Hello Packet
18	55.405850	fe80:abcd::15	ff02::5	OSPF	326	Hello Packet
19	59.301266	2001:db8::2	2001:db8::1	TCP	86	33698 > tqdata [ACK]

▶ Frame 14: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)

▶ Ethernet II, Src: CadmusCo_8e:8e:43 (08:00:27:8e:8e:43), Dst: CadmusCo_43:51:6b (08:00:27:43:51:6b)

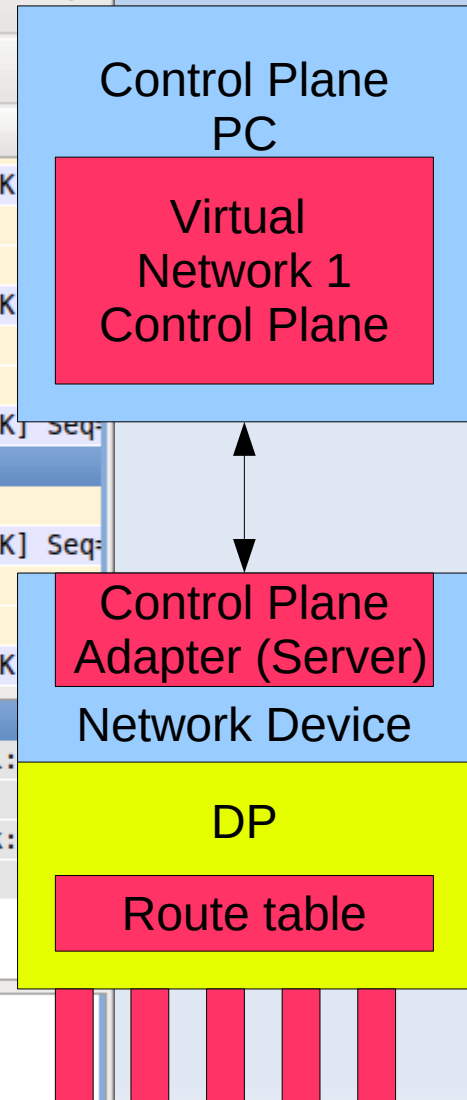
▶ Internet Protocol Version 6, Src: 2001:db8::2 (2001:db8::2), Dst: 2001:db8::1 (2001:db8::1)

▶ Transmission Control Protocol, Src Port: 33698 (33698), Dst Port: tqdata (2700), Seq: 1297, Ack: 2700

▼ Control Plane Adapter Transceiver Protocol

CPATP FIE instance: 0

CPATP interface: 17



```
0000 08 00 27 43 51 6b 08 00 27 8e 8e 43 86 dd 60 00  ..'CQk.. '...C...
0010 00 00 00 74 06 40 20 01 0d b8 00 00 00 00 00 00  ...t.@ .
0020 00 00 00 00 00 02 20 01 0d b8 00 00 00 00 00 00  .....
0030 00 00 00 00 00 01 83 a2 0a 8c 97 9d 62 2a 6f 51  .....b*oQ
0040 70 9c 80 18 01 19 5b ef 00 00 01 01 08 0a 07 c8  p.....[.
0050 28 47 07 cf 07 81 00 00 00 11 60 00 00 00 00 20  /c
```

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 zaobserwować wymianę wszystkich pakietów od protokołów routing.

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