Programowanie sieciowe

Janusz Gozdecki

B9, p.309, gozdecki@agh.edu.pl

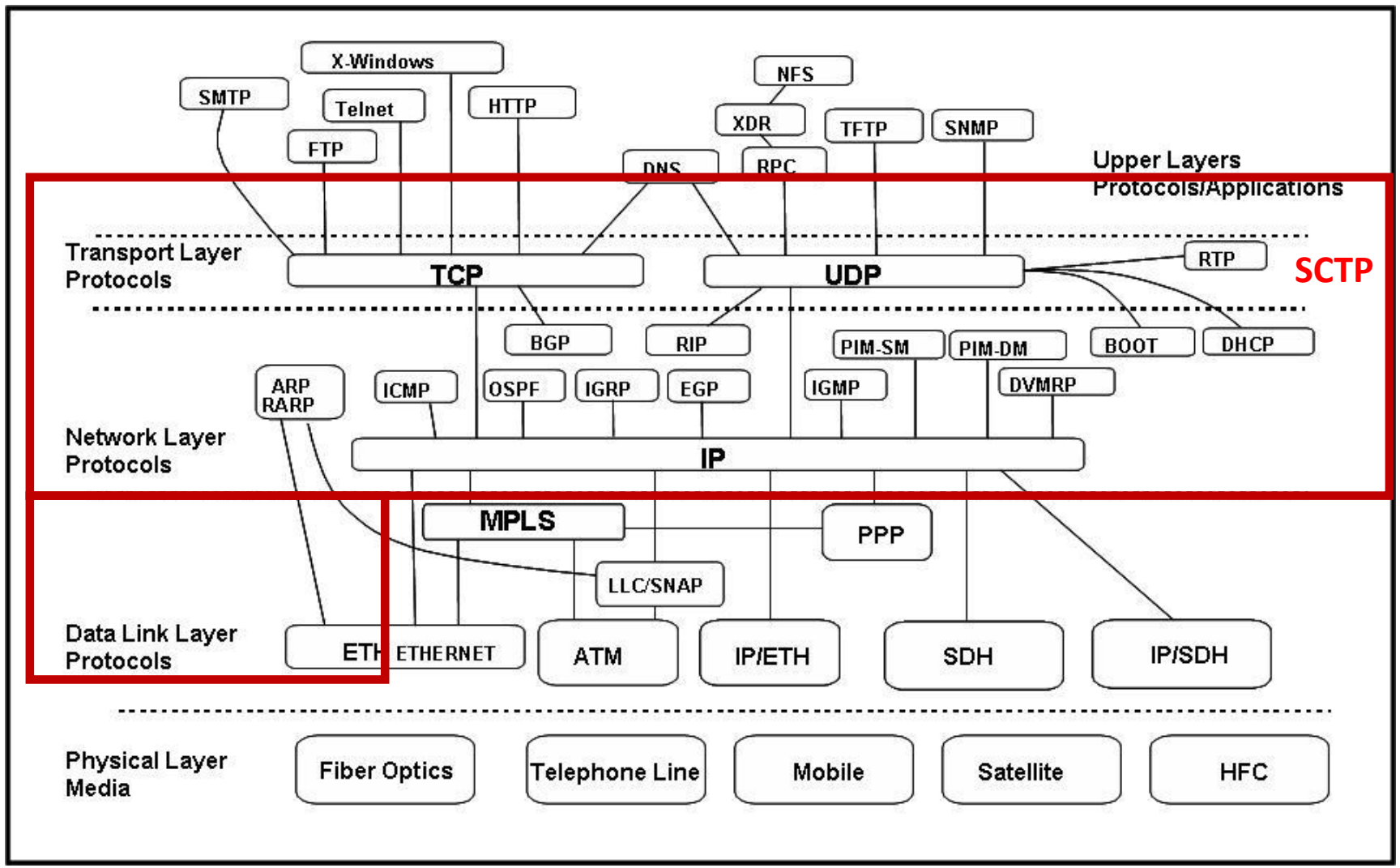
KT AGH

Wykład #1
Wstęp do programowania sieciowego

Plan wykładów.

Przedstawienie architektury i działania sieci TCP/IP. Ogólny przegląd protokołów. Adresy IP i numery portów. Standardowe usługi sieciowe.

Cel kursu – techniki programowania

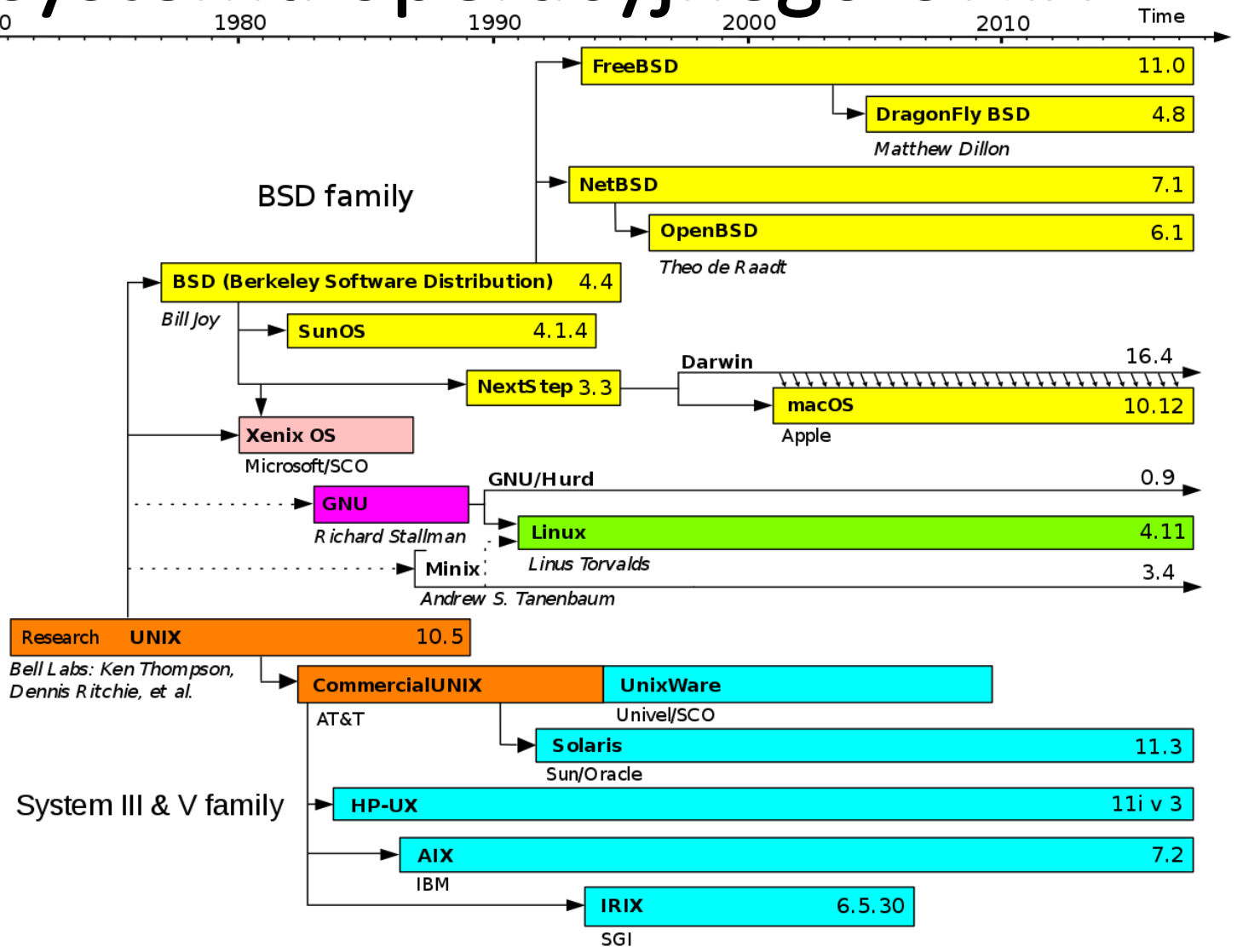


[unifiedtao-en.blogspot.com]

Historia gniazd sieciowych (*socket programming*)

- 1983 - Pierwsza wersja „gniazd” – BSD4.2
- 1986 - BSD 4.3 usprawnienie TCP
- 1988 – TCP Tahoe (powolny start, unikanie przeciążeń, szybki start) w BSD
- 1989 – Oprogramowanie sieciowe BSD v1.0
- 1990 – TCP Reno w BSD
- 1993 – Rozsyłanie grupowe w BSD
- 1994 – Oprogramowanie sieciowe BSD v3.0
- Oprogramowanie sieciowe BSD – podstawa wielu systemów (FreeBSD, NetBSD, OpenBSD)
- Linux – sieć pisana od podstaw (1990), API gniazd
- 1998 – protokół IPv6
- 2000/2013 – SCTP

Rozwój systemu operacyjnego UNIX



https://en.wikipedia.org/wiki/Darwin_(operating_system)#/media/File:Unix_timeline.en.svg

Plan wykładów (1/2)

- Wstęp do programowania sieciowego. Architektura i zasada działania sieci IP. Model klient-serwer. Adresacja w sieci IP. Podstawy programowania gniazd sieciowych. Prosty program klient-serwer.
- Sieciowe narzędzia administracyjne – wykrywanie problemów w aplikacjach sieciowych.
- Protokoły warstwy transportowej: **UDP, TCP, SCTP**. Opcje gniazd. API gniazd.
- Obsługa sygnałów. Modele obsługi wejścia-wyjścia: blokowalne, nieblokowalne, sterowane sygnałami, asynchroniczne, multipleksowane (select, poll, **epool**)

Plan wykładów (2/2)

- Implementacja protokołów rozgłoszeniowych w aplikacjach – **multicast** i broadcast.
- API DNS. Proces demona i logowanie zdarzeń w systemie.
- Gniazda surowe. Protokół ICMP. Monitorowanie sieci – biblioteka libpcap.
- Programowanie dostępu do warstwy kanałowej.
- Wstęp do programowania modułów do jądra systemu Linux.

- **Wykłady są wstępem do zajęć laboratoryjnych**

Materiały do zajęć

- Wykłady: **pluton.kt.agh.edu.pl/~gozdecki/PS_2019**
- **UNIX® Network Programming Volume 1, Third Edition: The Sockets Networking API, By W. Richard Stevens, Bill Fenner, Andrew M. Rudof**
- **IETF RFC**
- Strony ***man*** (opcja –S [numer sekcji:2,7])
- Dokumentacja bibliotek i jądra systemu Linux – głównie kody źródłowe
- Linux Device Drivers (3rd Edition), Authors: Jonathan Corbet, Allessandro Rubini & Greg Kroah-Hartman, O'Reilly

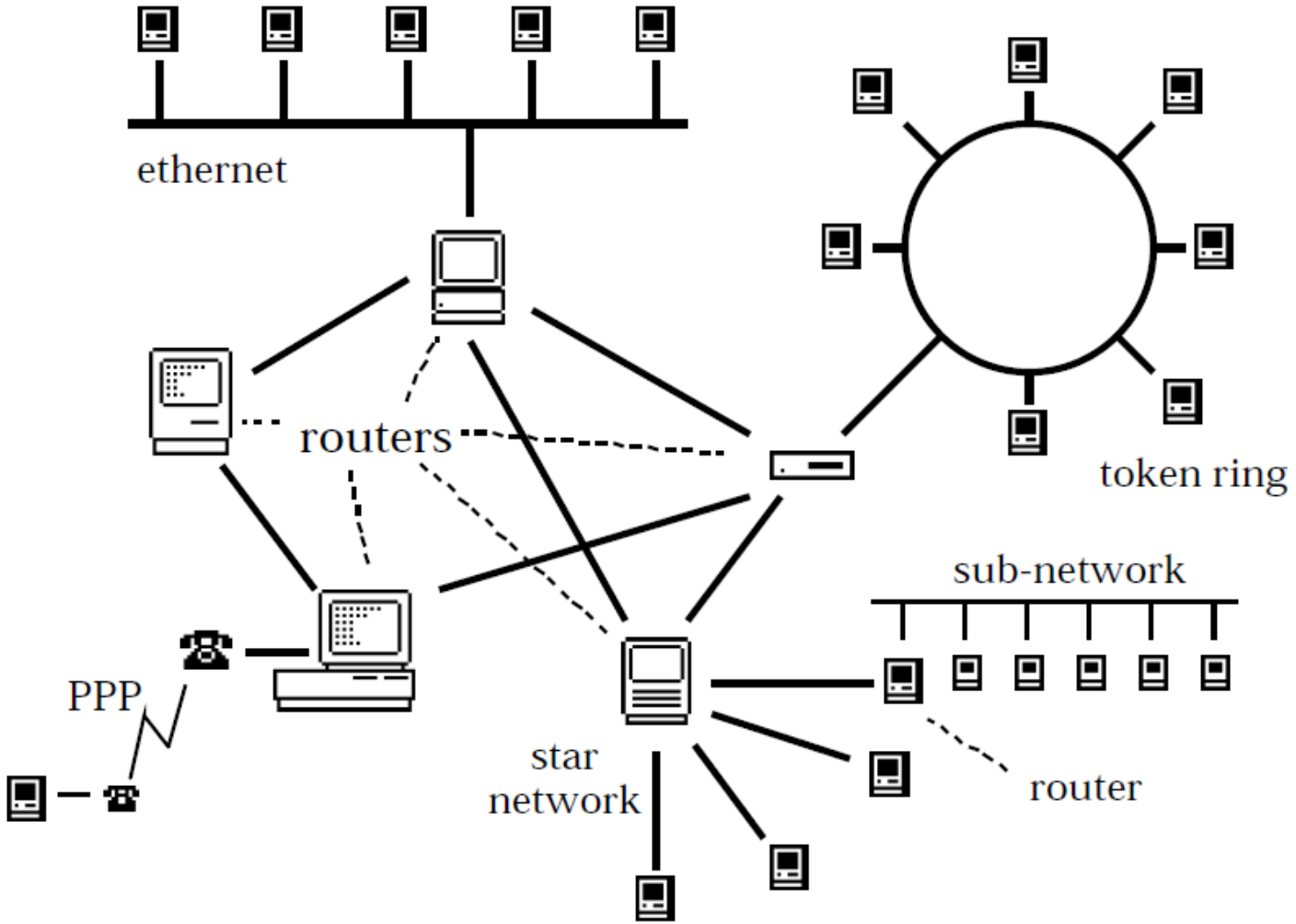
Materiały do zajęć

- **Informacje co należy przygotować do zajęć będą zamieszczane w instrukcjach do laboratorium, które wraz z przykładami do laboratorium i wykładami będą się znajdowały w katalogu:** ssh://pluton.kt.agh.edu.pl/~gozdecki/PS_2019

Sieć Internet –
Sieć zbudowana w oparciu o protokoły z rodziny TCP/IP

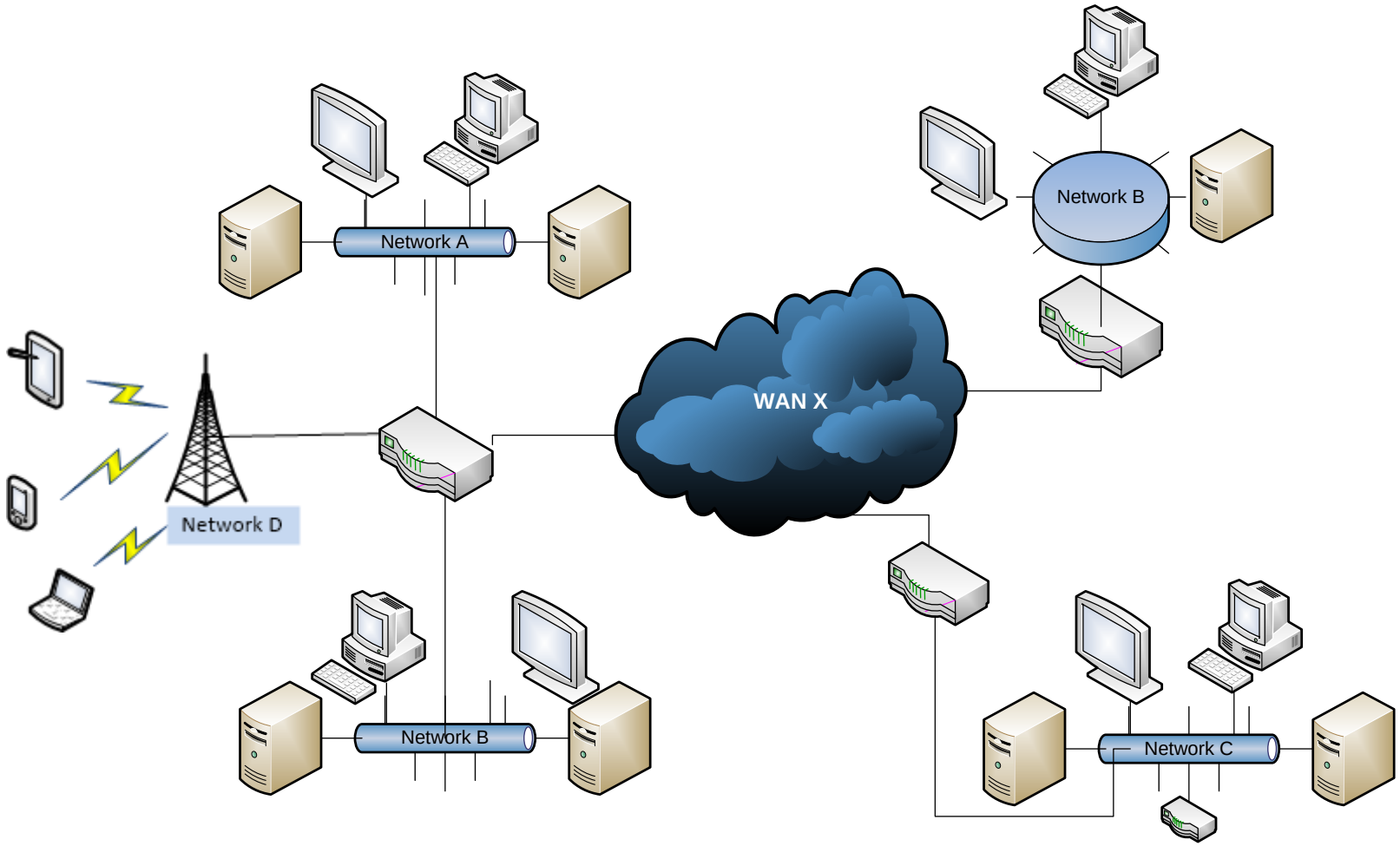
- Do komunikacji wymagane jest:
  - Połączenie fizyczne (kablowe, radiowe, itp..)
  - Wspólny język - protokoły
  - Adresacja - identyfikacja stacji w sieci

Architektura sieci TCP/IP



[Stevens_UNP]

Architektura sieci TCP/IP

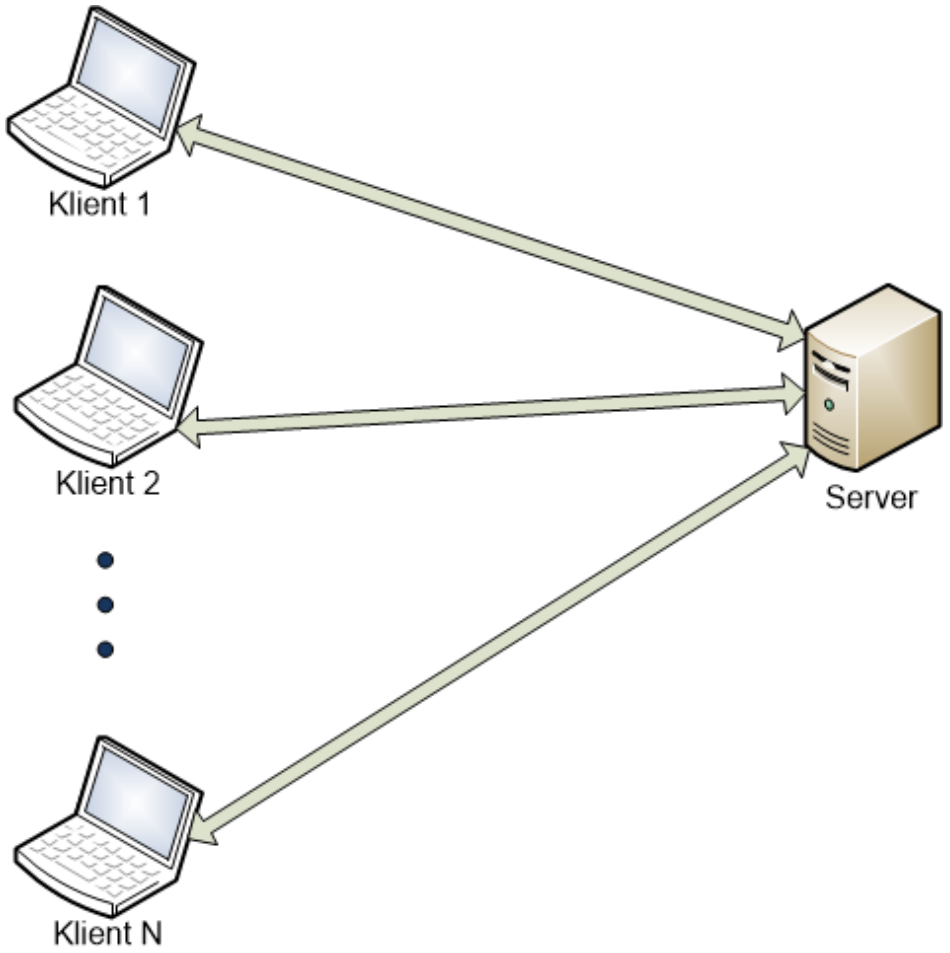


Architektura sieci TCP/IP widok programisty aplikacji użytkowych

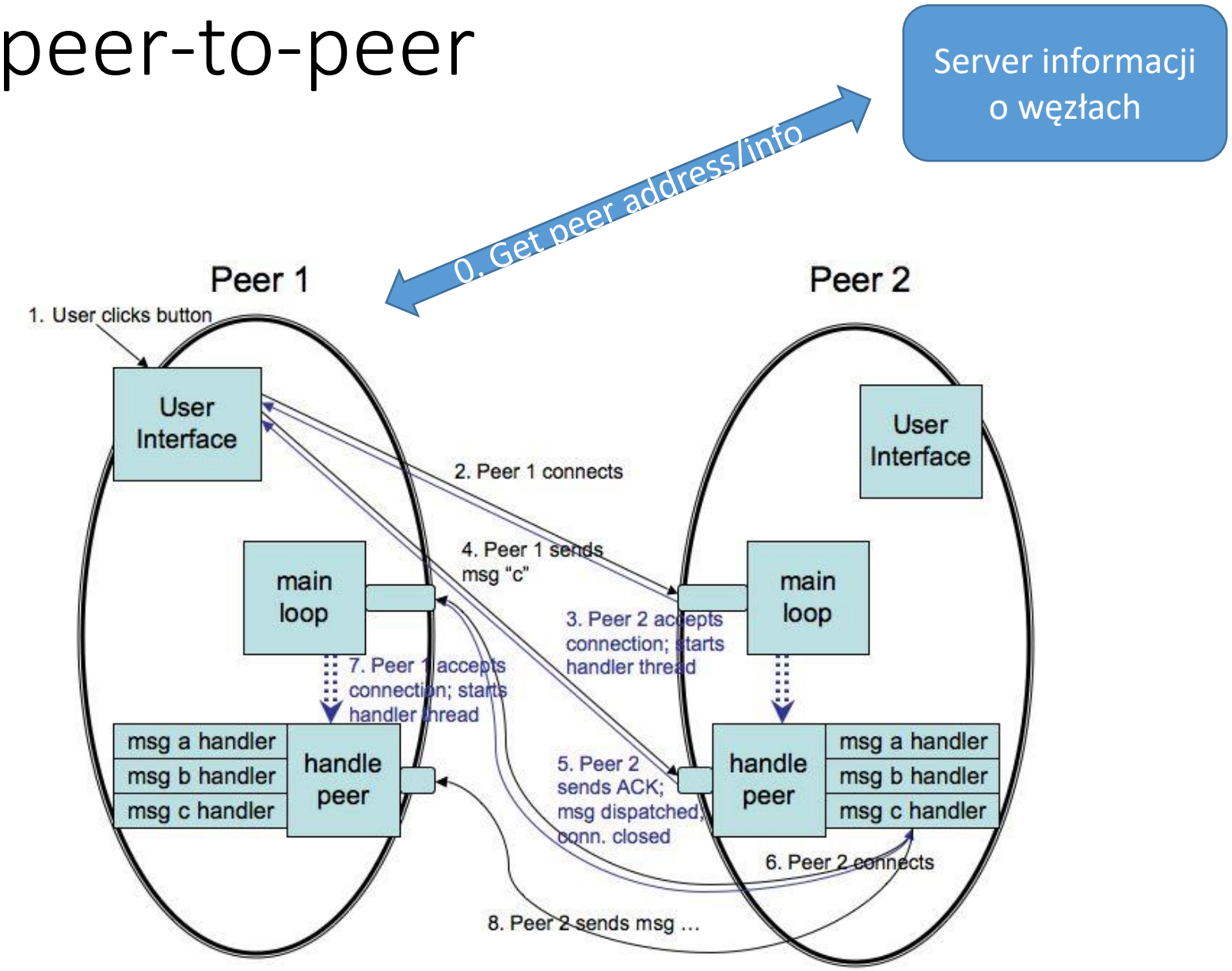
Modele obsługi (architektura oprogramowania)

- Model klient-serwer
- Model peer-to-peer

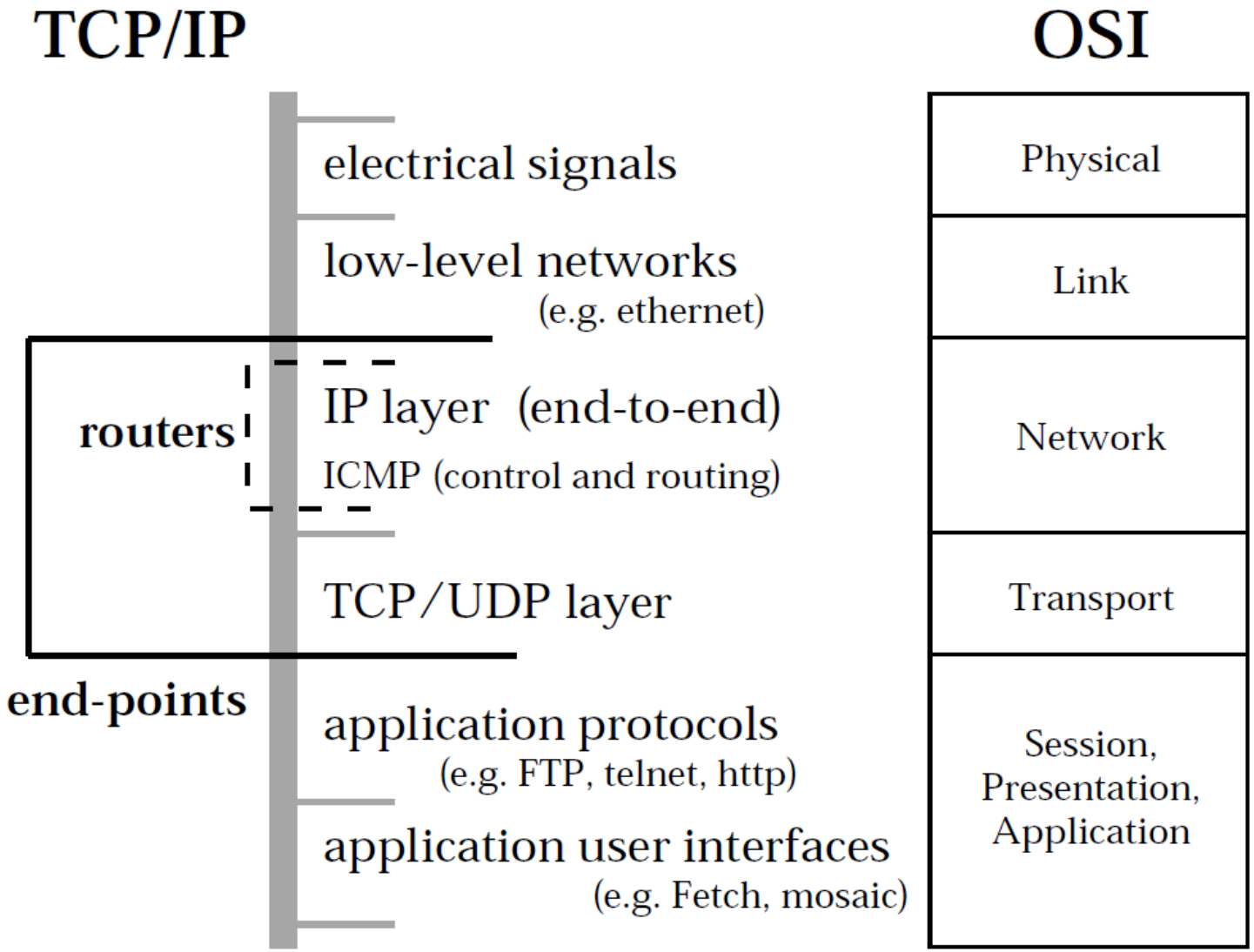
Model klient-serwer



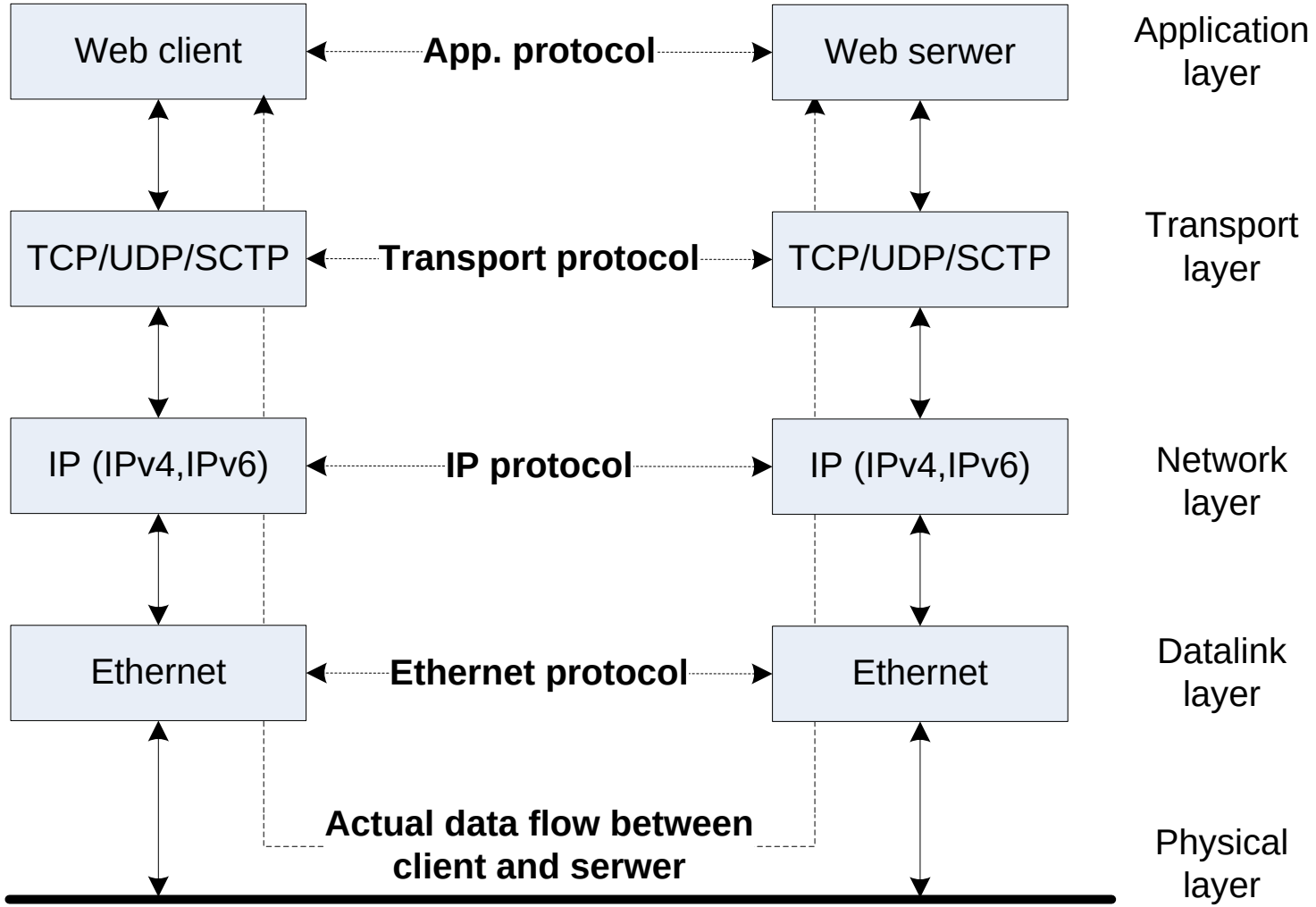
Model peer-to-peer

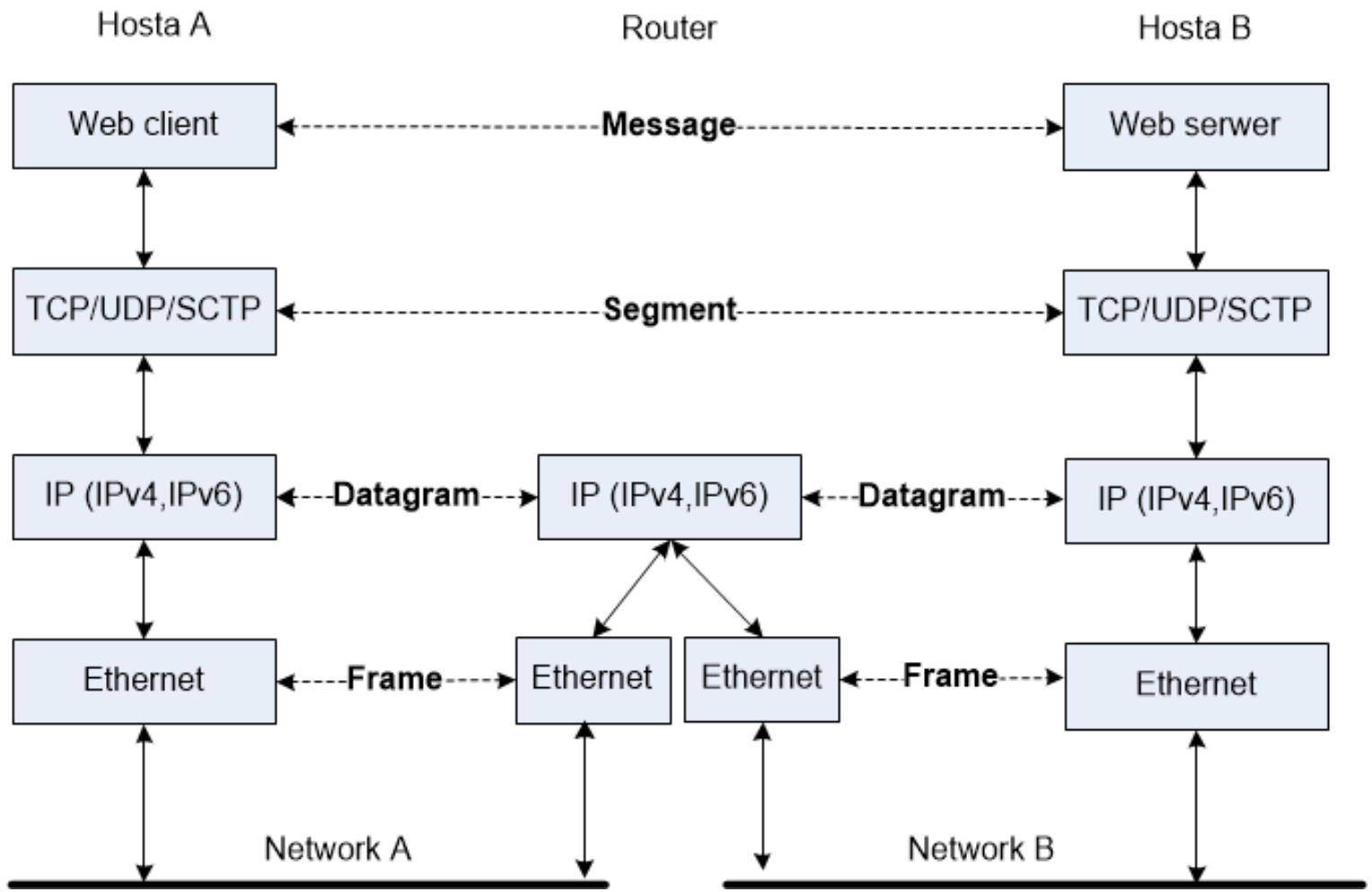


Protokoły – wspólny język

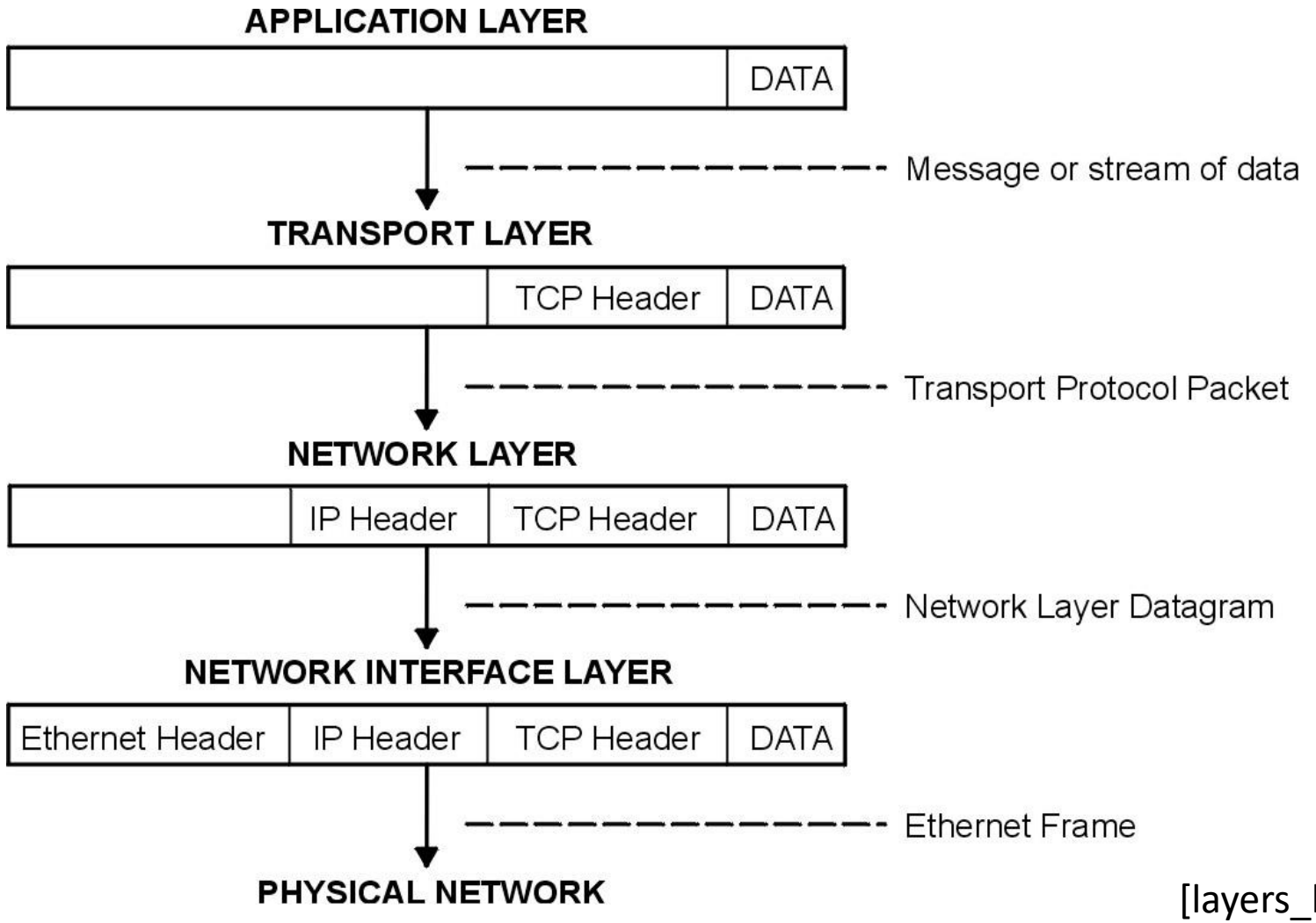


Protokoły a model warstwowy sieci TCP/IP




Hosta A
Router
Hosta B
Web client
Message
Web serwer
TCP/UDP/SCTP
Segment
TCP/UDP/SCTP
IP (IPv4,IPv6)
Datagram
IP (IPv4,IPv6)
Datagram
IP (IPv4,IPv6)
Ethernet
Frame
Ethernet
Ethernet
Frame
Ethernet
Network A
Network B

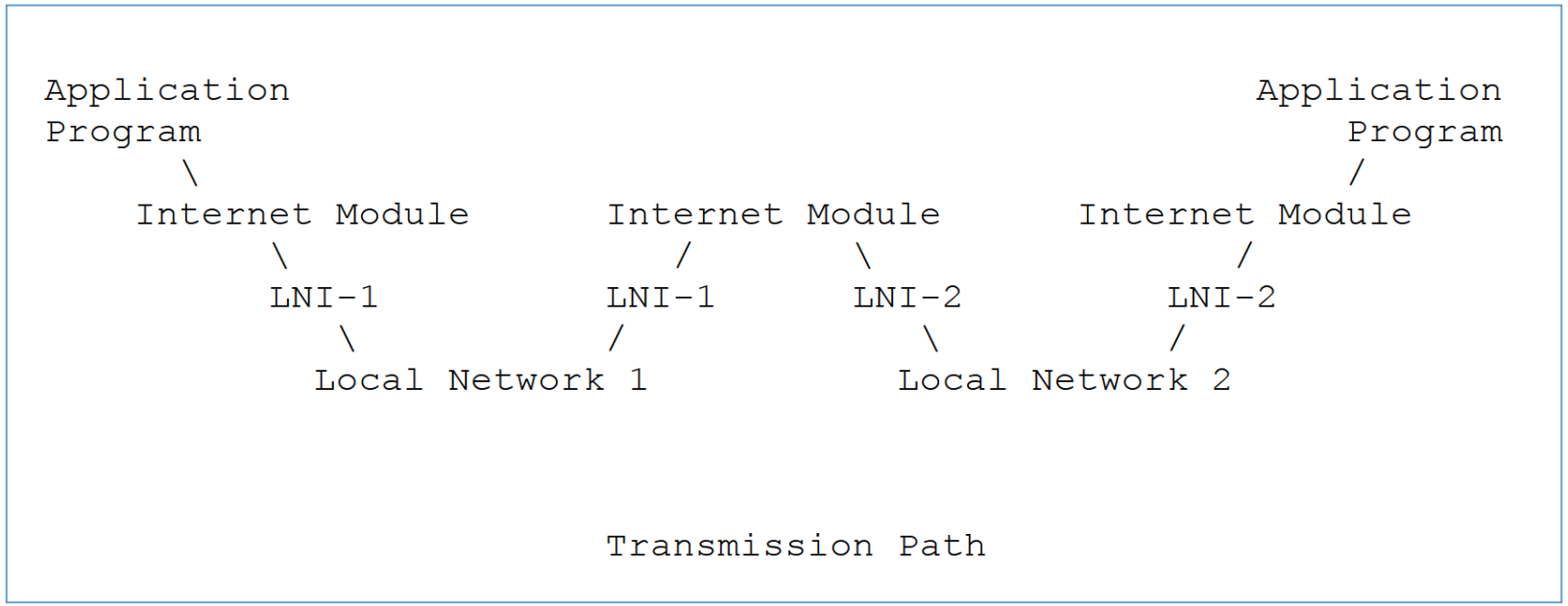
Protocols encapsulation in IP networks



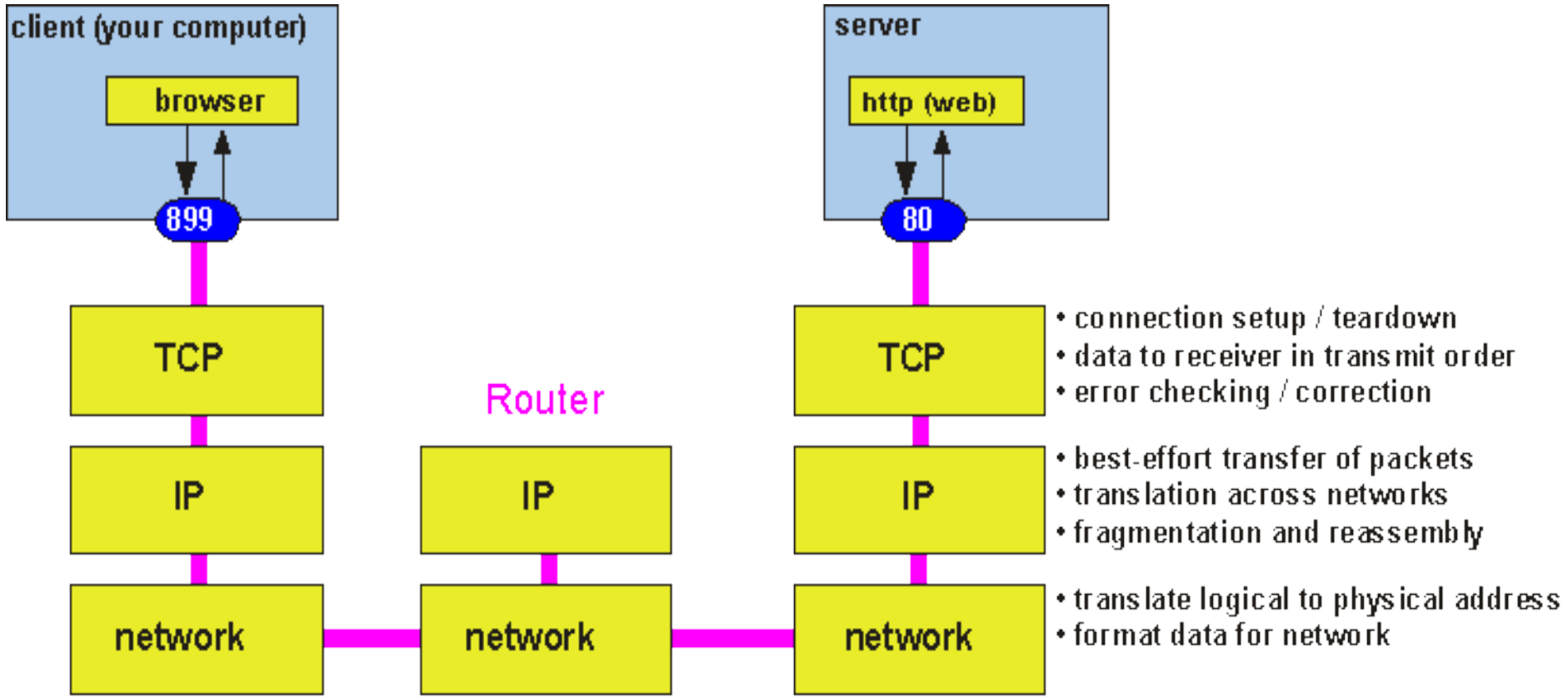
[layers_headers]

IP – IPv4(RFC791)/IPv6(2460) – zasady przekazywania datagramów

- “The internet protocol uses four key mechanisms in providing its service:
  - **Type of Service,**
  - **Time to Live,**
  - **Options,** and
  - **Header Checksum**.”

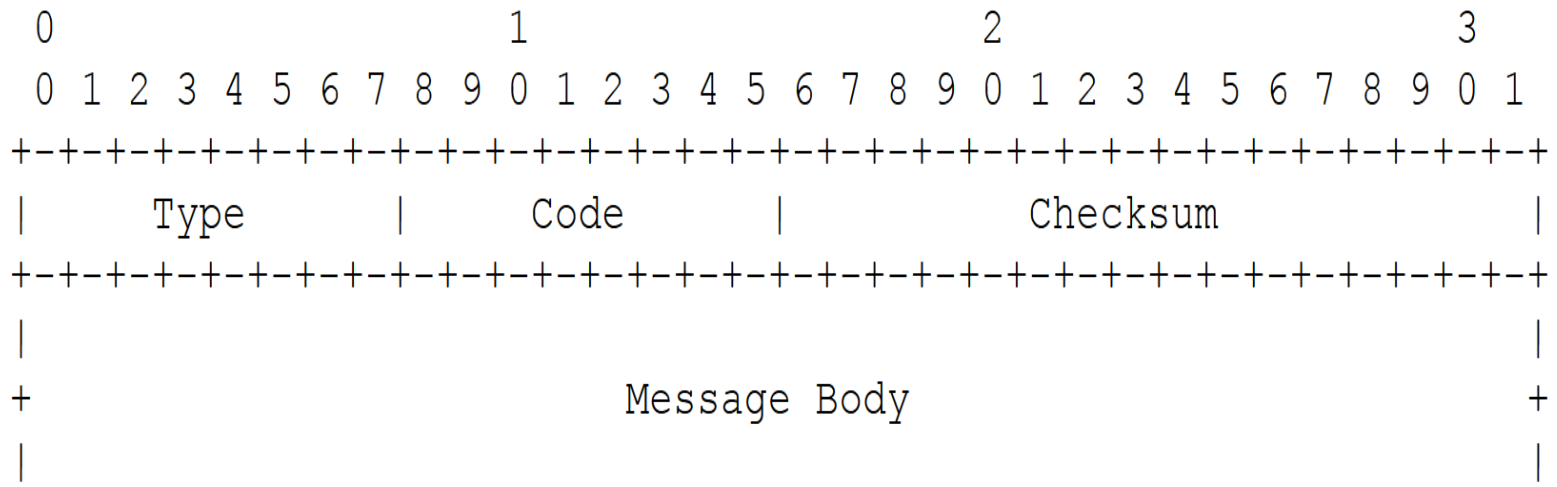


Protokoły a model warstwowy sieci TCP/IP – zasada działania protokołu IP



[enigmatic]

ICMP – międzysieciowy protokół sterowania komunikatami:
ICMPv4 (RFC792) oraz ICMPv6 (RFC4884)



- Type – typ komunikatu
- Code – kod (nie dla wszystkich komunikatów)

ICMPv4 – wybrane typy komunikatów

Type	Name	Reference
0	**Echo Reply**	[RFC792]
3	**Destination Unreachable**	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
8	**Echo Request**	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	**Time Exceeded**	[RFC792]
12	**Parameter Problem**	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]

ICMPv4 – wybrane kody komunikatu o typie 3 (Destination Unreachable)

- 0 Net Unreachable
- 1 **Host Unreachable**
- 2 **Protocol Unreachable**
- **3 Port Unreachable**
- 4 Fragmentation Needed and Don't Fragment was Set
- 5 Source Route Failed
- 6 Destination Network Unknown
- 7 Destination Host Unknown
- 8 Source Host Isolated
- 9 Communication with Destination Network is Administratively Prohibited
- 10 Communication with Destination Host is Administratively Prohibited
- 11 Destination Network Unreachable for Type of Service
- 12 Destination Host Unreachable for Type of Service
- 13 Communication Administratively Prohibited [RFC1812]
- 14 Host Precedence Violation [RFC1812] 15 Precedence cutoff in effect [RFC1812]

ICMPv6 – najważniejsze komunikaty

- ICMPv6 error messages:
  - 1 Destination Unreachable
  - 2 Packet Too Big
  - 3 Time Exceeded
  - 4 Parameter Problem
- ICMPv6 informational messages:
  - 128 Echo Request
  - 129 Echo Reply

ICMPv6 – kody dla komunikaty o typie 1 (*Destination Unreachable*)

0 - No route to destination

1 - Communication with destination administratively prohibited

2 - **Beyond scope of source address**

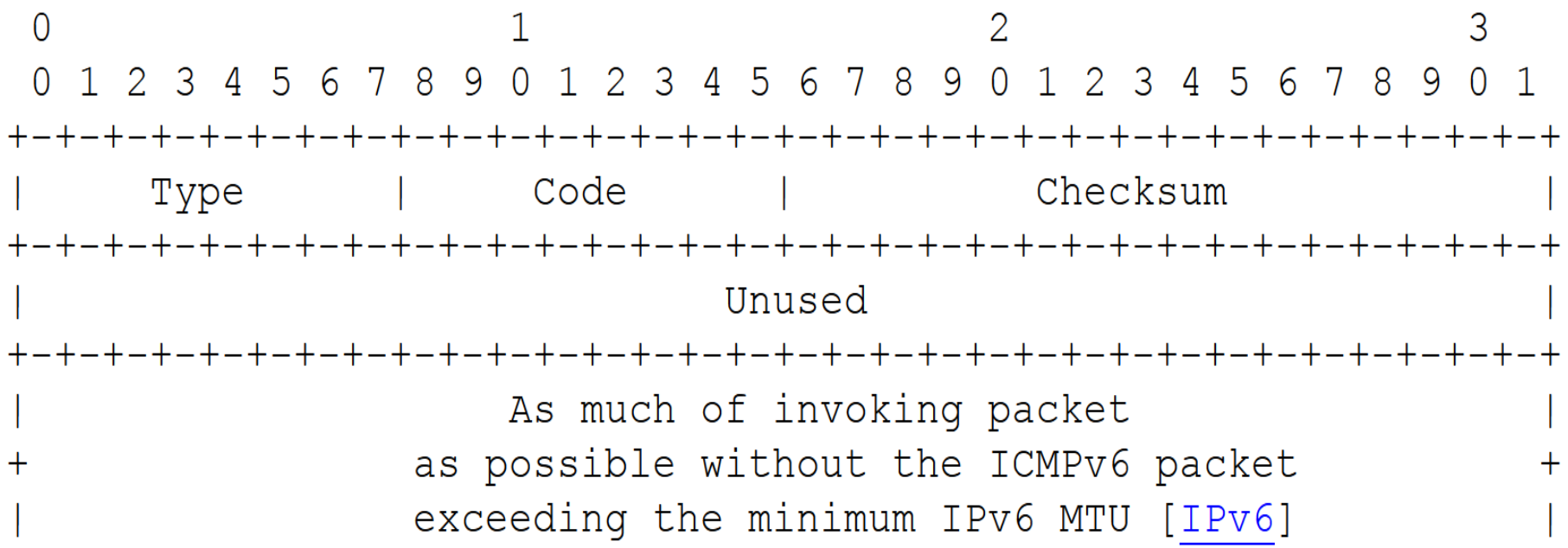
3 - **Address unreachable**

4 - **Port unreachable**

5 - Source address failed ingress/egress policy

6 - Reject route to destination

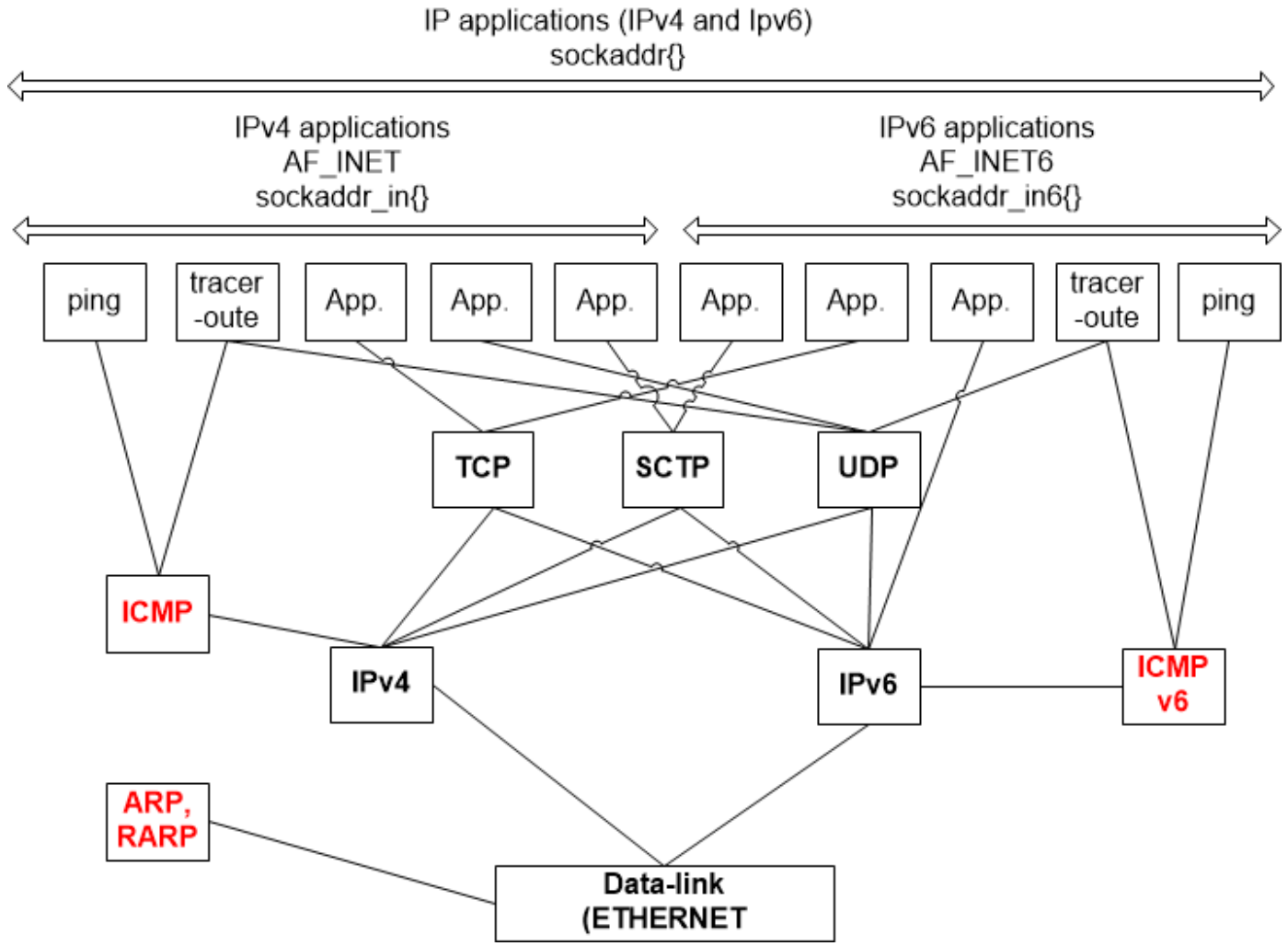
ICMPv6 - Destination Unreachable (Type 3) Format komunikatu



Adresacja

- Układ wielowarstwowy – poszczególne warstwy oprogramowania/protokoły mają własne adresy
  - Fizyczna: ul.Czarnowiejska 78, pokój 123, 73C-4
  - MAC: np. adres ETHERNET http://standards.ieee.org/develop/regauth/oui/public.html
  - Sieciowa – np. adres IPv4, lub IPv6
  - Transportowa – port (np. dla protokołu UDP lub TCP)
- Multipleksacja w warstwach protokołów sieci IP

Multipleksacja w warstwach protokołów sieci IP



Adres sieciowy

- Determinowany przez protokół sieciowy: IPv4 lub IPv6
- IPv4 – 4 oktety (149.156.114.3)
- IPv6 – 16 oktetów (2001:6d8:10:3400:215:17ff:fe42:480)

Maski adresów: określenie adresu sieci

- IPv4:
  - 10.2.10.0/24 - adres sieci
  - 10.2.10.0/255.255.255.0 - adres sieci
  - 10.2.10.1/24 - adres węzła sieci
  - 10.2.10.255 – adres rozgłoszeniowy (*broadcast*)
- IPv6:
  - fc00:1:1:1::0/64 – adres sieci
  - fc00:1:1:1::1/64 – adres węzła sieci

Przydział adresów

- IANA (Internet Assigned Numbers Authority)
- RFC 6177 - IPv6 Address Assignment to End Sites
- RFC 1918 - Address Allocation for Private Internets
  - **10.0.0.0** - 10.255.255.255 (10/8 prefix)
  - **172.16.0.0** - 172.31.255.255 (172.16/12 prefix)
  - **192.168.0.0** - 192.168.255.255 (192.168/16 prefix)
- RFC 4193 - Unique Local IPv6 Unicast Addresses
  - **FC00::/7**

Numery portów

- Adres aplikacji w systemie końcowym (**dla każdego protokołu transportowego są niezależne**) – 16 bitów
- Znane porty – RFC 1700 – nadawane przez organizację IANA (Internet Assigned Numbers Authority)
- /etc/services (nie wszystkie „znane” porty są zdefiniowane w RFC 1700)
- Znane numery portów przyporządkowane są tylko dla oprogramowania serwerów aplikacji

Przydział portów

- Internet Assigned Numbers Authority (IANA)
  - http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
- Zasady przydziału portów:
  - RFC 6335
- Stany portów:
  - Przydzielone
  - Nieprzydzielone
  - Zarezerwowane, np. porty do eksperymentów: 1023 i 1024

Podział portów:

- Porty ogólnie znane (ang. well-known ports) – 0 – 1023 – przydziela IANA (zwykle ten sam port dla tej samej usługi dla protokołu UDP, TCP i SCTP
- Porty zarejestrowane (przedział 1024-49151) – rejestrowane przez IANA, porty serwerów nieuprzywilejowanych
- Porty dynamiczne (prywatne, efemeryczne) (przedział 49152-65535)

Adres aplikacji w sieci Internet – identyfikacja aplikacji

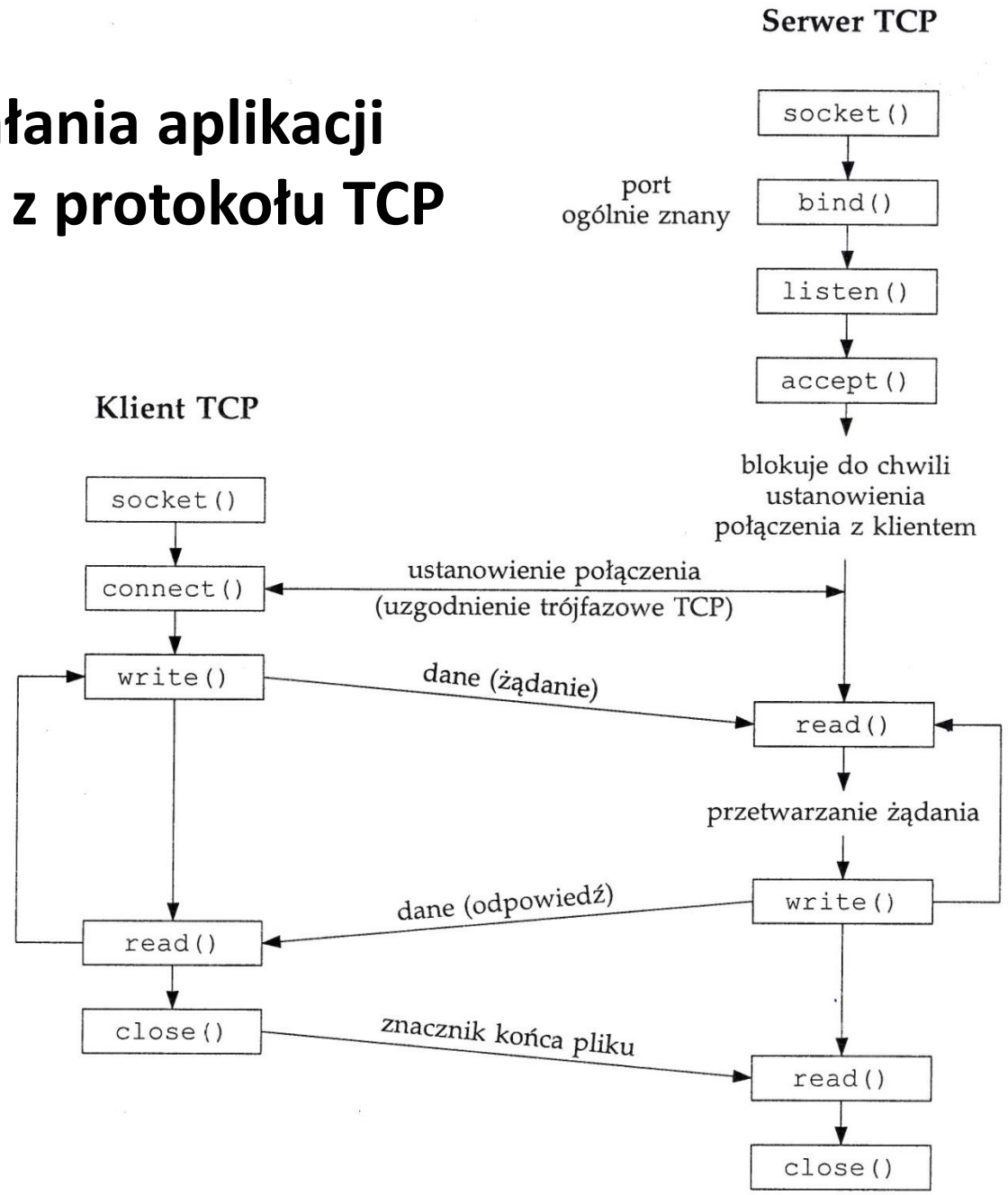
- **Adres sieciowy, protokół transportowy, port**
- **Identyfikacja połączenia**:
  - Adres sieciowy lokalny, Adres sieciowy zdalny, protokół transportowy , port lokalny, port zdalny
- **Para gniazdowa** (dla danego protokołu transportowego):
  - Adres sieciowy lokalny, port lokalny, Adres sieciowy zdalny, port zdalny

Prosty program typu klient-serwer: usługa czasu dobowego

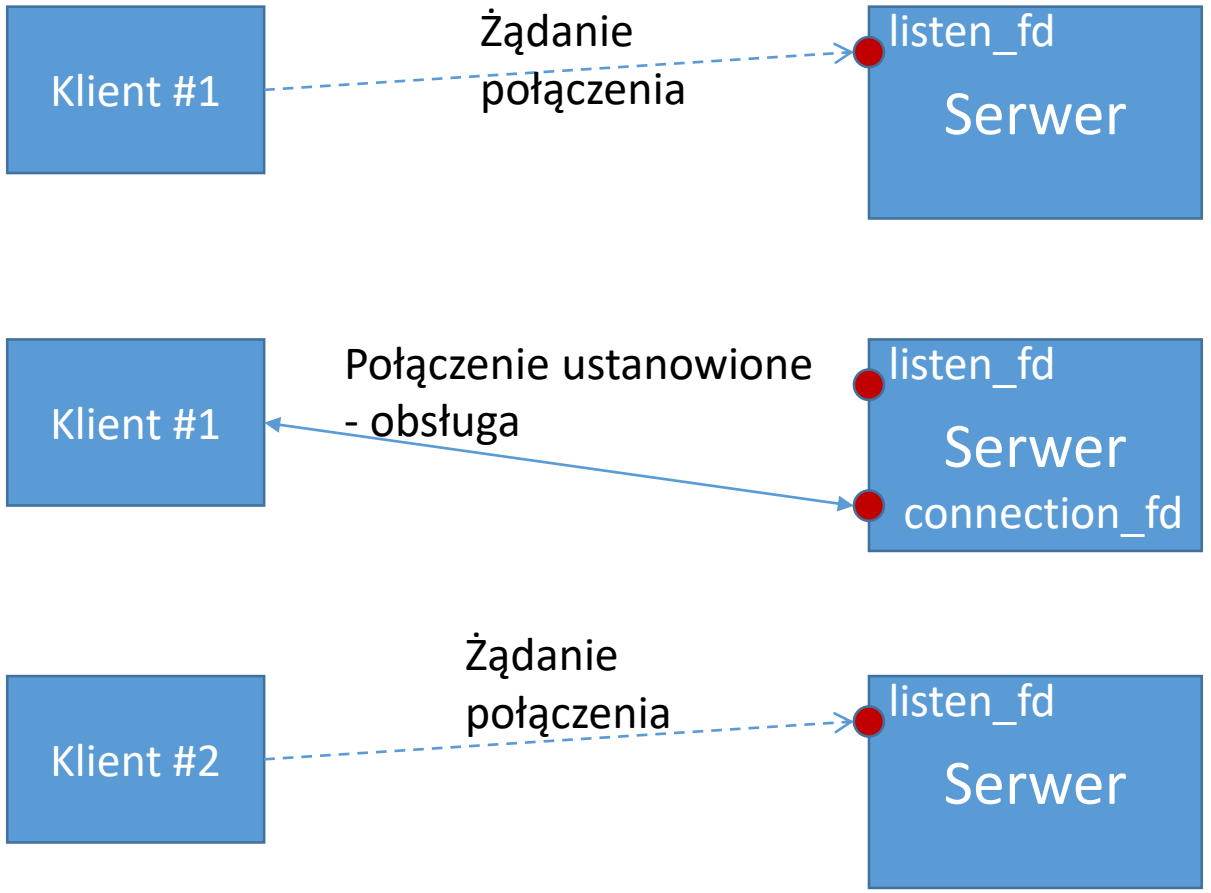
- Klient
  - Przygotowuje struktury adresowe i inicjuje gniazdo
  - Łączy się z serwerem
  - Odbiera informację
  - Wyświetla informację na ekranie
  - Koniec

- Serwer
  - Przygotowuje struktury adresowe i inicjuje gniazdo
  - Nasłuchuje
  - Akceptuje połączenie od klienta
  - Wysyła informacje
  - Kończy połączenie
  - Nasłuchuje ….

Schemat działania aplikacji korzystającej z protokołu TCP



Działanie serwera



Klient


```
1.  int
2.  main(int argc, char **argv)
3.  {
4.      int                 sockfd, n;
5.      struct sockaddr_in6 servaddr;
6.      char                recvline[MAXLINE + 1];

7.      if (argc != 2){
8.          fprintf(stderr, "usage: a.out <IPaddress> : %s\n", strerror(errno));
9.          return 1;
10.     }
11.     if ( (sockfd = socket(AF_INET6, SOCK_STREAM, 0)) < 0){
12.         fprintf(stderr,"socket error : %s\n", strerror(errno));
13.         return 1;
14.     }

15.     bzero(&servaddr, sizeof(servaddr));
16.     servaddr.sin6_family = AF_INET6;
17.     servaddr.sin6_port   = htons(13);   /* daytime server */
18.     if (inet_pton(AF_INET6, argv[1], &servaddr.sin6_addr) <= 0){
19.         fprintf(stderr,"inet_pton error for %s : %s \n", argv[1], strerror(errno));
20.         return 1;
21.     }
22.     if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0){
23.         fprintf(stderr,"connect error : %s \n", strerror(errno));
24.         return 1;
25.     }

26.     while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
27.         recvline[n] = 0;    /* null terminate */
28.         if (fputs(recvline, stdout) == EOF){
29.             fprintf(stderr,"fputs error : %s\n", strerror(errno));
30.             return 1;
31.         }
32.     }
33.     if (n < 0)
34.         fprintf(stderr,"read error : %s\n", strerror(errno));

35.     fprintf(stderr,"OK\n");
36. //   flush(stderr);

37.     exit(0);
38. }
```

Serwer


```
1.  int
2.  main(int argc, char **argv)
3.  {
4.      int                 listenfd, connfd;
5.      socklen_t               len;
6.      char                buff[MAXLINE], str[INET6_ADDRSTRLEN+1];
7.      time_t                  ticks;
8.      struct sockaddr_in6     servaddr, cliaddr;

9.      if ( (listenfd = socket(AF_INET6, SOCK_STREAM, 0)) < 0){
10.         fprintf(stderr,"socket error : %s\n", strerror(errno));
11.         return 1;
12.     }

13.     bzero(&servaddr, sizeof(servaddr));
14.     servaddr.sin6_family = AF_INET6;
15.     servaddr.sin6_addr   = in6addr_any;
16.     servaddr.sin6_port   = htons(13);      /* daytime server */

17.     if ( bind( listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0){
18.         fprintf(stderr,"bind error : %s\n", strerror(errno));
19.         return 1;
20.     }
21.     if ( listen(listenfd, LISTENQ) < 0){
22.         fprintf(stderr,"listen error : %s\n", strerror(errno));
23.         return 1;
24.     }
25.
26.     for ( ; ; ) {
27.         len = sizeof(cliaddr);
28.         if ( (connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &len)) < 0){
29.             fprintf(stderr,"accept error : %s\n", strerror(errno));
30.             continue;
31.         }

32.         bzero(str, sizeof(str));
33.         inet_ntop(AF_INET6, (struct sockaddr  *) &cliaddr.sin6_addr,  str, sizeof(str));
34.         printf("Connection from %s\n", str);

35.         ticks = time(NULL);
36.         snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
37.         if( write(connfd, buff, strlen(buff))< 0 )
38.             fprintf(stderr,"write error : %s\n", strerror(errno));
39.         close(connfd);
40.     }
41. }
```

Wybrane narzędzia sieciowe w systemie UNIX/LINUX

- ifconfig
- route
- netstat
- **ip**
- ss
- ping/ping6
- traceroute/traceroute6 (tracepath/tracepath6)
- tcpdump/wireshark
- lsof

ifconfig

```
1.  saturn:/home/gozdecki->/sbin/ifconfig eth0
2.  eth0      Link encap:Ethernet  HWaddr 00:15:17:42:04:80
3.            inet addr:149.156.114.3  Bcast:149.156.114.255  Mask:255.255.255.0
4.            inet6 addr: 2001:6d8:10:3400:215:17ff:fe42:480/64 Scope:Global
5.            inet6 addr: fe80::215:17ff:fe42:480/64 Scope:Link
6.            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
7.            RX packets:619836574 errors:0 dropped:0 overruns:0 frame:0
8.            TX packets:1103048555 errors:0 dropped:0 overruns:0 carrier:0
9.            collisions:0 txqueuelen:1000
10.           RX bytes:78641395355 (73.2 GiB)  TX bytes:1604865535150 (1.4 TiB)
11.           Interrupt:185 Memory:b8820000-b8840000
```

ifconfig

```
1.  saturn:/home/gozdecki->/sbin/ifconfig  lo


2.  lo        Link encap:Local Loopback
3.            inet addr:127.0.0.1  Mask:255.0.0.0
4.            inet6 addr: ::1/128 Scope:Host
5.            UP LOOPBACK RUNNING  MTU:16436  Metric:1
6.            RX packets:473856 errors:0 dropped:0 overruns:0 frame:0
7.            TX packets:473856 errors:0 dropped:0 overruns:0 carrier:0
8.            collisions:0 txqueuelen:0
9.            RX bytes:439405954 (419.0 MiB)  TX bytes:439405954 (419.0
    MiB)
```

ifconfig - przydział nowego adresu

- ifconfig eth0 10.2.0.2 netmask 255.255.255.0 broadcast 10.2.0.255 (ifconfig eth0 10.2.0.2/24 broadcast 10.2.0.255)
- ifconfig eth0:1 10.2.0.2 netmask 255.255.255.0 broadcast 10.2.0.255
- ifconfig eth0 down/up
- ifconfig eth0 promisc
- …

route – informacje i zmiana zawartości tablicy rutingu

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 wlan0
192.168.0.0     0.0.0.0         255.255.255.0   U     9      0        0 wlan0
```

IPROUTE2

Configuration utilities replaced by iproute2		
Purpose	Legacy utility	iproute2 equivalent
Address and link configuration	ifconfig	ip addr, ip link
Routing tables	route	ip route
Neighbors	arp	ip neigh
VLAN	vconfig	ip link
Tunnels	iptunnel	ip tunnel
Bridges	brctl	ip link, bridge
Multicast	ipmaddr	ip maddr
Statistics	netstat	ip -s, ss

ip

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
ip [ -force ] -batch filename
where  OBJECT := { link | addr | addrlabel | route |
    rule | neigh | ntable | tunnel | tuntap | maddr |
    mroute | mrule | monitor | xfrm | netns | l2tp |
    tcp_metrics | token }
OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -
    r[esolve] | -f[amily] { inet | inet6 | ipx | dnet |
    bridge | link } | -4 | -6 | -0 | -l[oops] { maximum-
    addr-flush-attempts } | -o[neline] | -t[imestamp] | -
    b[atch] [filename] | -rc[vbuf] [size]}
```

man ip

ip

```
> ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
   UNKNOWN mode DEFAULT group default    link/loopback
   00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
   pfifo_fast state DOWN mode DEFAULT group default qlen 1000
   link/ether 24:b6:fd:14:02:e0 brd ff:ff:ff:ff:ff:ff
4: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
   state UP mode DORMANT group default qlen 1000
   link/ether 80:86:f2:09:1f:dc brd ff:ff:ff:ff:ff:ff
12: wwan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state
   DOWN mode DEFAULT group default qlen 1000     link/ether
   02:80:37:ec:02:00 brd ff:ff:ff:ff:ff:ff
```

ip

```
> ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
     group default qlen 1000
    link/ether 24:b6:fd:14:02:e0 brd ff:ff:ff:ff:ff:ff
4: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
     default qlen 1000
    link/ether 80:86:f2:09:1f:dc brd ff:ff:ff:ff:ff:ff
    inet 192.168.46.137/21 brd 192.168.47.255 scope global wlan0
       valid_lft forever preferred_lft forever
    inet6 fe80::8286:f2ff:fe09:1fdc/64 scope link
       valid_lft forever preferred_lft forever
12: wwan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
     qlen 1000
    link/ether 02:80:37:ec:02:00 brd ff:ff:ff:ff:ff:ff
```

ip – dodawanie i usuwanie adresu

- ip addr add 192.168.200.201/24 brd 192.168.200.255 dev eth0
- ip -6 addr add fc00:1:1::12:12/64 dev eth0
- ip -6 addr del fc00:1:1::12:12/64 dev eth0

ping/ping6
ICMP Echo Request

- ping [opcje] destination_address
- ping -p pattern
- ping -I interface
- ping -i interval
- ping -A - ping adaptacyjny
- ping –f - flood
- ping -Q TOS
- ping -s packetsize
- ping -c count
- ping6 –F flowlabel
- ping –t TTL

tracepath – wyznaczanie ścieżki pomiędzy dwoma węzłami – używa UDP

```
1.      ~$ tracepath ktrtr.agh.edu.pl
2.       1?: [LOCALHOST]                                        pmtu 1500
3.       1:  192.168.40.1                                        15.526ms
4.       1:  192.168.40.1                                        18.386ms
5.       2:  net77-43-88-1.mclink.it                              5.261ms
6.       3:  net77-43-88-1.mclink.it                              4.989ms pmtu 1492
7.       3:  213.21.129.59                                       33.122ms
8.       4:  213.21.130.49                                       35.124ms
9.       5:  10-1-10.bear1.Italy2.Level3.net                     36.668ms
10.      6:  ae-21-3204.car1.Berlin1.Level3.net                  64.297ms asymm 12
11.      7:  ae-21-3204.car1.Berlin1.Level3.net                  66.568ms asymm 12
12.      8:  212.162.10.82                                       73.682ms asymm 11
13.      9:  z-poznan-gw3.krakow.10Gb.rtr.pionier.gov.pl         84.750ms asymm 12
14.     10:  149.156.0.18                                        85.393ms asymm 14
15.     11:  149.156.6.226                                       93.743ms asymm 14
16.     12:  b6rtr.agh.edu.pl                                    84.145ms asymm 14
17.     13:  ktrtr.agh.edu.pl                                    82.122ms reached
18.          Resume: pmtu 1492 hops 13 back 15
```

tracepath(6)

```
1.    >   tracepath 149.156.114.3
2.     1?: [LOCALHOST]                                      pmtu 1500
3.     1:  192.168.40.1                                     131.881ms
4.     1:  192.168.40.1                                      13.185ms
5.     2:  net77-43-88-1.mclink.it                            3.916ms
6.     3:  net77-43-88-1.mclink.it                            5.278ms pmtu 1492
7.     3:  213.21.129.59                                     31.630ms
8.     4:  213.21.130.49                                     33.844ms
9.     5:  10-1-10.bear1.Italy2.Level3.net                  291.485ms
10.    6:  ae-21-3204.car1.Berlin1.Level3.net               232.753ms asymm 12
11.    7:  ae-21-3204.car1.Berlin1.Level3.net               316.205ms asymm 12
12.    8:  212.162.10.82                                     90.735ms asymm 11
13.    9:  z-poznan-gw3.krakow.10Gb.rtr.pionier.gov.pl      143.394ms asymm 12
14.   10:  149.156.0.18                                     117.155ms asymm 14
15.   11:  149.156.6.226                                    166.161ms asymm 14
16.   12:  b6rtr.agh.edu.pl                                 222.376ms asymm 14
17.   13:  ktrtr.agh.edu.pl                                 203.678ms asymm 15
18.   14:  no reply
19.   15:  no reply
```

traceroute – wyznaczanie ścieżki pomiędzy dwoma węzłami – używa UDP/ICMP/(TCP)

```
$ traceroute  ae3.mx1.fra.de.geant.net
traceroute to ae3.mx1.fra.de.geant.net
  (62.40.98.130), 64 hops max
  1   149.156.203.249  20,917ms  18,342ms  17,696ms
  2   149.156.119.17  22,778ms  18,290ms  17,580ms
  3   149.156.6.222  20,288ms  16,723ms  18,879ms
  4   149.156.0.217  18,716ms  21,058ms  20,324ms
  5   212.191.224.69  25,737ms  28,832ms  26,459ms
  6   62.40.125.245  28,737ms  27,354ms  25,818ms
  7   62.40.98.130  46,280ms  65,524ms  47,892ms
```

traceroute(6)

```
1.    $ traceroute 149.156.114.3
2.    traceroute to 149.156.114.3 (149.156.114.3), 64 hops max
3.      1   192.168.40.1  1,226ms  0,762ms  0,898ms
4.      2   77.43.88.1  2,512ms  1,646ms  1,767ms
5.      3   213.21.129.59  79,950ms  86,828ms  90,987ms
6.      4   213.21.130.49  95,175ms  167,961ms  204,358ms
7.      5   212.133.7.33  174,308ms  234,784ms  204,622ms
8.      6   4.69.161.6  204,624ms  306,857ms  78,756ms
9.      7   4.69.161.6  73,652ms  83,253ms  64,886ms
10.     8   212.162.10.82  68,697ms  70,036ms  88,477ms
11.     9   212.191.224.70  92,222ms  430,445ms  571,238ms
12.    10   149.156.0.18  76,199ms  389,681ms  96,585ms
13.    11   149.156.6.226  373,451ms  433,176ms  451,580ms
14.    12   149.156.6.220  441,147ms  413,517ms  409,976ms
15.    13   149.156.119.18  79,829ms  369,985ms  265,796ms
16.    14   *  *  *
17.    15   *  *  *
18.    16   *  *  *
19.    17   *  *  *
```

netstat

netstat {--interfaces|-i} [--all|-a] [--extend|-e[--extend|-e]] [--verbose|-v] [--program|-p] [--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]

```
1.  janusz@janusz-V131:~$ netstat -6 -aunpve
2.  (Not all processes could be identified, non-owned process info
3.   will not be shown, you would have to be root to see it all.)
4.  Active Internet connections (servers and established)
5.  Proto Recv-Q Send-Q Local Address           Foreign Address         State
    User       Inode       PID/Program name
6.  udp6       0      0 :::40983                :::*
    107        11605       -
7.  udp6       0      0 :::20980                :::*
    0          1271269     -
8.  udp6       0      0 :::13408                :::*                        0
    1273875     -
9.  udp6       0      0 :::5353                 :::*
    107        11603       -
```

netstat

```
1.  $ netstat -r
2.  Kernel IP routing table
3.  Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
4.  default         local0          0.0.0.0         UG        0 0          0 wlan0
5.  snullnet0       *               255.255.255.0   U         0 0          0 wlan0

6.  $ netstat -r -6
7.  Kernel IPv6 routing table
8.  Destination                    Next Hop                   Flag Met Ref Use If
9.  fe80::/64                      ::                         U    256 0     0 wlan0
10. ::/0                           ::                         !n   -1  1 30724 lo
11. ::1/128                        ::                         Un   0   3 38709 lo
12. fe80::8286:f2ff:fe09:1fdc/128  ::                         Un   0   1     0 lo
13. ff00::/8                       ::                         U    256 0     0 wlan0
14. ::/0                           ::                         !n   -1  1 30724 lo
```

netstat

```
1.      V131:~$ netstat -6 -atnpve

2.    (Not all processes could be identified, non-owned process info
3.     will not be shown, you would have to be root to see it all.)
4.    Active Internet connections (servers and established)
5.    Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
      Inode       PID/Program name
6.    tcp6       0      0 :::139                  :::*                    LISTEN      0
      1922        -
7.    tcp6       0      0 :::80                   :::*                    LISTEN      0
      12500       -
8.    tcp6       0      0 :::22                   :::*                    LISTEN      0
      11766       -
9.    tcp6       0      0 ::1:631                 :::*                    LISTEN      0
      1276001     -
10.   tcp6       0      0 ::1:6010                :::*                    LISTEN      1000
      1175369     -
11.   tcp6       0      0 ::1:6011                :::*                    LISTEN      1000
      1230862     -
12.   tcp6       0      0 ::1:6012                :::*                    LISTEN      1000
      1267488     -
13.   tcp6       0      0 :::445                  :::*                    LISTEN      0
      1921        -
14.   tcp6       0      0 ::1:2947                :::*                    LISTEN      0
      1954        -
15.   tcp6       0      0 :::7                    :::*                    LISTEN      0
      1268291     -
16.   tcp6       0      0 2001:6d8:10:3400::50888 2001:6d8:10:1060::6:993 ESTABLISHED 1000
      1274099     5193/thunderbird
17.   tcp6       0      0 2001:6d8:10:3400::50477 2a00:1450:4013:c01::443 ESTABLISHED 1000
```

netstat

```
V131:~$ netstat -ie
eth0      Link encap:Ethernet  HWaddr 00:15:17:41:D2:D0
          inet addr:149.156.203.6  Bcast:149.156.203.127  Mask:255.255.255.128
          inet6 addr: fe80::215:17ff:fe41:d2d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6696660426 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12232952947 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:525660317237 (489.5 GiB)  TX bytes:17293189021738 (15.7 TiB)
          Interrupt:185 Memory:b8820000-b8840000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6285240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6285240 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1676225230 (1.5 GiB)  TX bytes:1676225230 (1.5 GiB)
```

SS

- ss – podgląd stanów gniazd w systemie, więcej informacji o stanach TCP niż inne narzędzia
- Przykłady:
  - ss -t -a display all TCP sockets (-a – all sockets states).
  - ss -u -a display all UDP sockets.
  - ss -o state established '(dport = :ssh or sport = :ssh)‘
    display all established ssh connections.
  - ss -x src /tmp/.X11-unix/*
    find all local processes connected to X server.
  - ss sport = :ssh - list connections to ssh deamon
  - ss -o state fin-wait-1 '(sport = :http or sport = :https)' dst 193.233.7/24
    List all the tcp sockets in state FIN-WAIT-1 for the www server to network 193.233.7/24 and look at their timers.

How Do I Filter Sockets Using TCP States?

- The syntax is:
  - *## tcp ipv4 ##* ss -4 state FILTER-NAME-HERE
  - *## tcp ipv6 ##* ss -6 state FILTER-NAME-HERE
- Where FILTER-NAME-HERE can be any one of the following,
  - **established**
  - **syn-sent**
  - **syn-recv**
  - **fin-wait-1**
  - **fin-wait-2**
  - **time-wait**
  - **closed**
  - **close-wait**
  - **last-ack**
  - **listening**
  - **closing**
  - **all** : All of the above states
  - **connected** : All the states except for listen and closed
  - **synchronized** : All the connected states except for syn-sent
  - **bucket** : Show states, which are maintained as minisockets, i.e. time-wait and syn-recv.
  - **big** : Opposite to bucket state.

ss - przykład

```
root@mesh51:~# ss -aute
Netid          State          Recv-Q  Send-Q     Local Address:Port     Peer Address:Port
udp            UNCONN     0       0              10.2.70.51:698                 *:*      ino=5729 sk=dcfe5480
udp    UNCONN     0      0                          *:698                          *:*      ino=5728 sk=dcfe5260
tcp    LISTEN     0      128                       :::http                        :::*      ino:5627 sk:dcdb15c0
tcp    LISTEN     0      128                       ::: ssh                        :::*      ino:7087 sk:dcdb1060
tcp    LISTEN     0      128                        *: ssh                         *:*      ino:7085 sk:dd103040
tcp    LISTEN     0      1                         *:2006                         *:*      ino:5500 sk:dd103a00
tcp    LISTEN     0      128                  127.0.0.1:6010                       *:*      ino:217470
     sk:ddd450c0
tcp    LISTEN     0      128                       ::1:6010                       :::*      ino:217469
     sk:dcdb0040
tcp    ESTAB      0      0                    10.2.0.51:ssh                 10.2.0.37:51255
     timer:(keepalive,103min,0) ino:217431 sk:ddd45a80
```

tcpdump

- Podgląd ramek w warstwie kanałowej (łącza danych)
- Ważniejsze opcje: -i interface, -e [link header], -A [ASCI], -w file, -r file, -n[no names resolver], -nn[no port resolver] -v, -vv [verbose], -I [monitor mode], -x [print in hex]

tcpdump

1. 17:46:27.959359 STP 802.1d, Config, Flags [none], bridge-id 4028.00:22:0c:65:57:00.8007, length 42
2. 17:46:31.031086 IP 192.168.46.238.5353 > 224.0.0.251.5353: 0 [2q] PTR (QU)? _airplay._tcp.local. PTR (QU)? _raop._tcp.local. (49)
3. 17:46:31.031498 IP6 fe80::c11:36b3:d012:b3bd.5353 > ff02::fb.5353: 0 [2q] PTR (QU)? _airplay._tcp.local. PTR (QU)? _raop._tcp.local. (49)
4. 17:46:31.031782 STP 802.1d, Config, Flags [none], bridge-id 4028.00:22:0c:65:57:00.8007, length 42
5. 17:46:31.032000 IP 192.168.46.238.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _airplay._tcp.local. PTR (QM)? _raop._tcp.local. (49)
6. 17:46:31.032381 IP6 fe80::c11:36b3:d012:b3bd.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _airplay._tcp.local. PTR (QM)? _raop._tcp.local. (49)
7. 17:46:34.103051 STP 802.1d, Config, Flags [none], bridge-id 4028.00:22:0c:65:57:00.8007, length 42
8. 17:46:34.103354 IP 192.168.46.238.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _airplay._tcp.local. PTR (QM)? _raop._tcp.local. (49)
9. 17:46:34.103673 IP6 fe80::c11:36b3:d012:b3bd.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _airplay._tcp.local. PTR (QM)? _raop._tcp.local. (49)
10. 17:46:34.103936 STP 802.1d, Config, Flags [none], bridge-id 4028.00:22:0c:65:57:00.8007, length 42
11. 17:46:36.493578 IP 192.168.46.137.60998 > 173.194.40.7.443: Flags [.], ack 1621158276, win 46, options [nop,nop,TS val 18664673 ecr 856607654], length 0
12. 17:46:36.529535 IP 173.194.40.7.443 > 192.168.46.137.60998: Flags [.], ack 1, win 394, options [nop,nop,TS val 856652692 ecr 18642141], length 0
13. 17:46:37.175178 STP 802.1d, Config, Flags [none], bridge-id 4028.00:22:0c:65:57:00.8007, length 42

tcpdump

1. 17:49:17.319478 IP 74.125.71.95.443 > 192.168.46.137.49543: Flags [.], seq 1:1419, ack 211, win 341, options [nop,nop,TS val 975749 ecr 18704847], length 1418
2. 17:49:17.319500 IP 192.168.46.137.49543 > 74.125.71.95.443: Flags [.], ack 1419, win 32, options [nop,nop,TS val 18704879 ecr 975749], length 0
3. 17:49:17.331339 IP 173.194.40.26.443 > 192.168.46.137.43929: Flags [P.], seq 16788:18188, ack 1665, win 356, options [nop,nop,TS val 856796450 ecr 18704705], length 1400
4. 17:49:17.331392 IP 192.168.46.137.43929 > 173.194.40.26.443: Flags [.], ack 18188, win 67, options [nop,nop,TS val 18704882 ecr 856796450], length 0
5. 17:49:17.342364 IP 192.168.40.1.53 > 192.168.46.137.26118: 55535 4/4/4 A 199.16.156.6, A 199.16.156.38, A 199.16.156.70, A 199.16.156.230 (243)
6. 17:49:17.349597 IP 136.243.25.61.80 > 192.168.46.137.38434: Flags [.], seq 18197:19637, ack 2456, win 37453, length 1440
7. 17:49:17.349621 IP 192.168.46.137.38434 > 136.243.25.61.80: Flags [.], ack 19637, win 64800, length 0
8. 17:49:17.358085 IP 199.96.57.6.80 > 192.168.46.137.41535: Flags [P.], seq 14281:15709, ack 561, win 31, options [nop,nop,TS val 495707339 ecr 18704786], length 1428
9. 17:49:17.358112 IP 192.168.46.137.41535 > 199.96.57.6.80: Flags [.], ack 15709, win 60, options [nop,nop,TS val 18704889 ecr 495707339], length 0
10. 17:49:17.368492 IP 74.125.71.95.443 > 192.168.46.137.49543: Flags [.], seq 1419:2837, ack 211, win 341, options [nop,nop,TS val 975749 ecr 18704847], length 1418
11. 17:49:17.368518 IP 192.168.46.137.49543 > 74.125.71.95.443: Flags [.], ack 2837,

tcpdump – filtrowanie pakietów

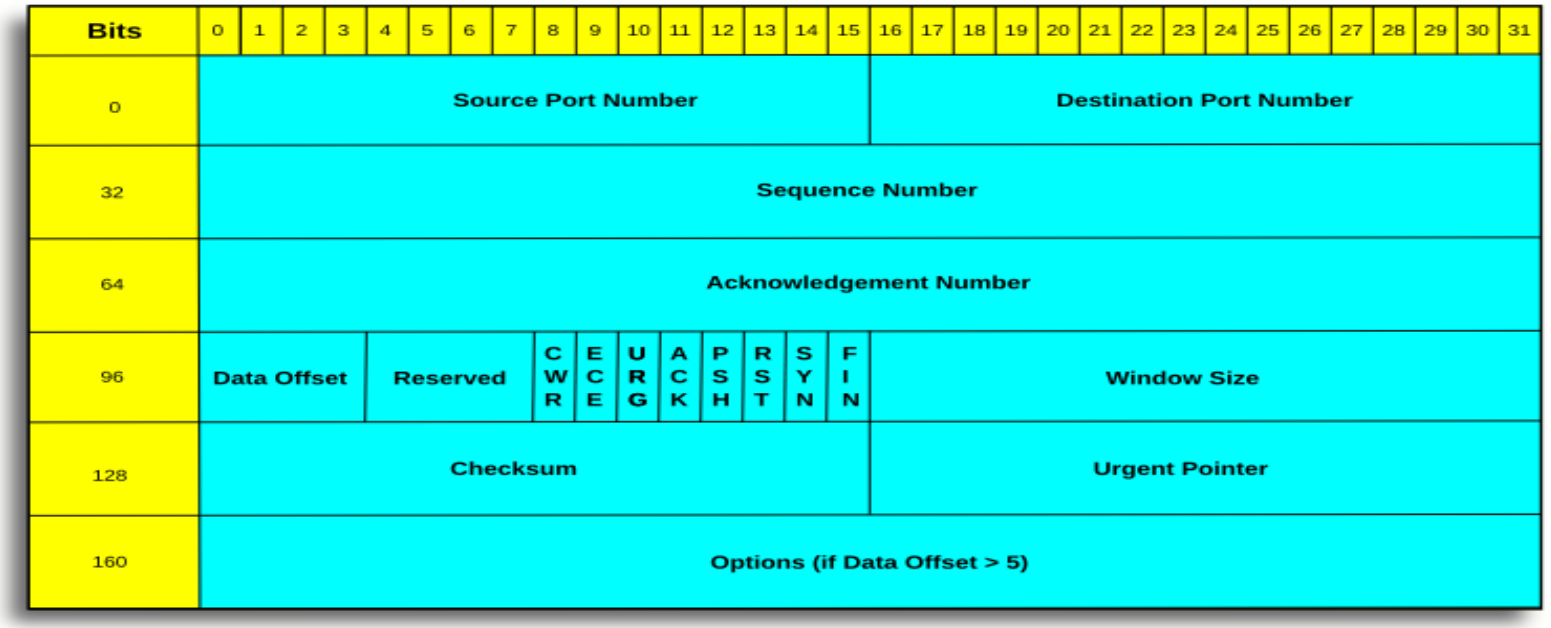
- Filtr składa się z jednego lub kilku kluczy podstawowych. Klucz składa się zwykle z kwalifikatora i id, który może być napisem lub liczbą
- Kwalifikatory typu: host, net, port, portrange (`host saturn', `net 128.3', `port 20', `portrange 6000-6008'.
- Kwalifikatory kierunku: src, dst, src or dst, src and dst, ra, ta, addr1, addr2, addr3, and addr4. (`src foo', `dst net 128.3', `src or dst port ftp-data'.
- Kwalifikator protokołu: ether, fddi, tr, wlan, ip, ip6, arp, rarp, decnet, tcp i udp. (Np. `ether src foo', `arp net 128.3', `tcp port 21', `udp portrange 7000-7009', `wlan addr2 0:2:3:4:5:6'.
- Inne słowa kluczowe: broadcast, multicast, gateway, greater, less

Ustawianie filtru na interfejsie - format filtru - przykłady

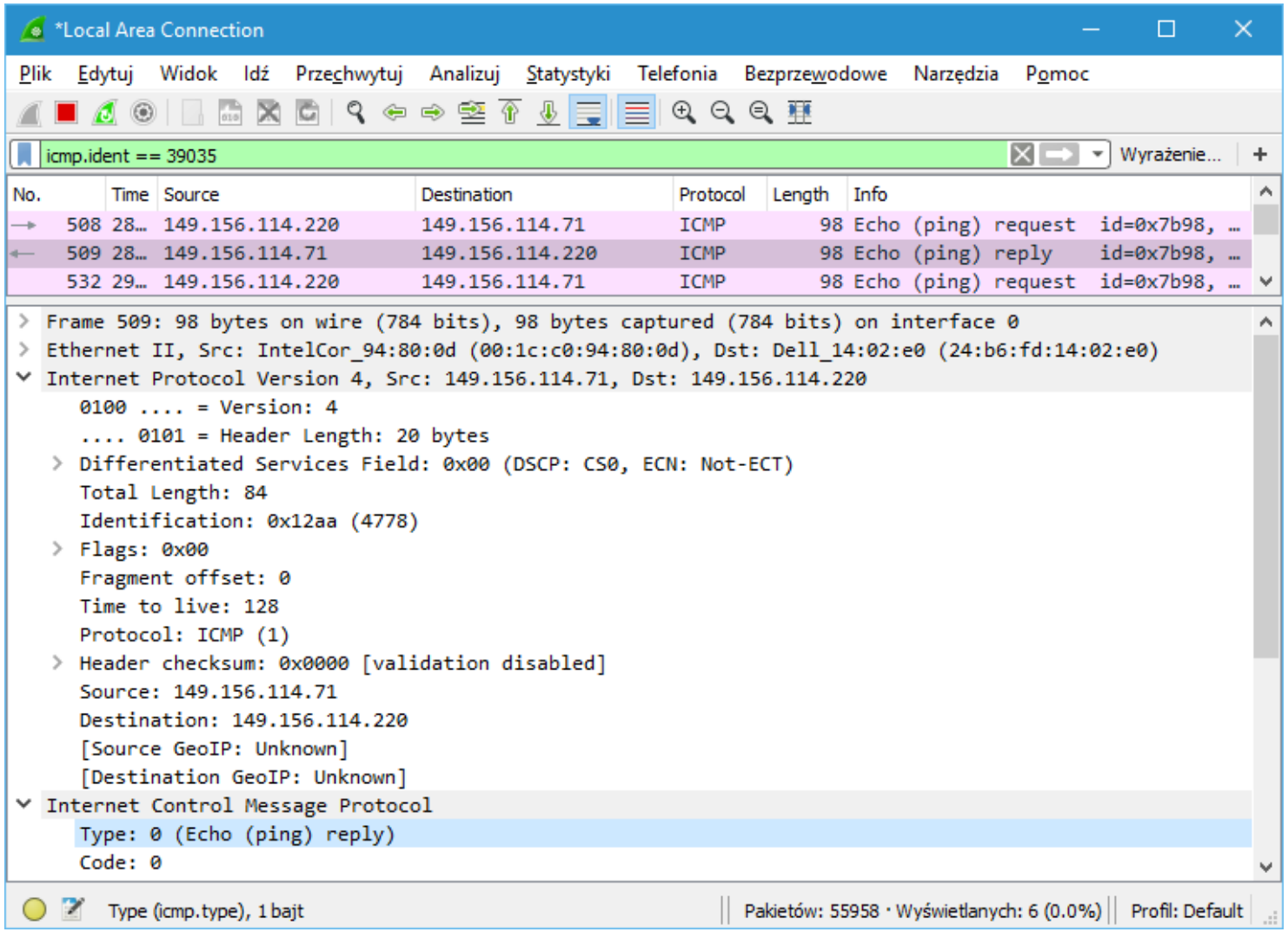
- ether broadcast
- ip broadcast
- ether multicast
- ip6 multicast
- host 149.156.114.3 and not port ftp and not port ftp-data
- tcp and port 80 and tcp[13:1] & 0x7 != 0
- (tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550)

tcp and port 80 and tcp[13:1] & 0x7 != 0

(tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550)



wireshark


*Local Area Connection
Plik Edytuj Widok Idź Przechwytuj Analizuj Statystyki Telefonia Bezprzewodowe Narzędzia Pomoc
icmp.ident == 39035
Wyrażenie...
No. Time Source Destination Protocol Length Info
508 28… 149.156.114.220 149.156.114.71 ICMP 98 Echo (ping) request id=0x7b98, …
509 28… 149.156.114.71 149.156.114.220 ICMP 98 Echo (ping) reply id=0x7b98, …
532 29… 149.156.114.220 149.156.114.71 ICMP 98 Echo (ping) request id=0x7b98, …
Frame 509: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: IntelCor_94:80:0d (00:1c:c0:94:80:0d), Dst: Dell_14:02:e0 (24:b6:fd:14:02:e0)
Internet Protocol Version 4, Src: 149.156.114.71, Dst: 149.156.114.220
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x12aa (4778)
Flags: 0x00
Fragment offset: 0
Time to live: 128
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
Source: 149.156.114.71
Destination: 149.156.114.220
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Type (icmp.type), 1 bajt
Pakietów: 55958 · Wyświetlanych: 6 (0.0%)
Profil: Default

LSOF

- lsof – informacje o otwartych plikach
- lsof . – procesy, które korzystają z bieżącego katalogu
- lsof -i – informacje o gniazdach sieciowych
- lsof -i 6 – informacje o gniazdach IPv6 (-i4 – IPv4)
- lsof –iTCP – informacje o gniazdach TCP
- lsof -i :22 – filtr na porcie 22
- lsof -i@172.16.12.5 – filtr na adres
- lsof -i@172.16.12.5:22 - filtr na adres i port
- lsof -c ssh – informacje o plikach otworzonych przez proces którego nazwa zaczyna się od ‘ssh’ (lsof -i -c ssh -a – tylko gniazda)
- lsof –p 12345 - informacje o gniazdach otwartych przez proces o PID 12345

LSOF

- lsof -i -sTCP:LISTEN – gniazda TCP w stanie LISTEN
- lsof -i | grep -i LISTEN – wersja z ‘grep’
- lsof -i -sTCP:ESTABLISHED
- lsof -i @fw.google.com:2150=2180 – zakres portów
- -a - operator AND
- lsof -u daniel -a -i @1.1.1.1 - użytkownik i sieć
- kill -9 `lsof -t -u daniel` - usuwanie procesów użytkownika
- kill -HUP `lsof -t -c sshd` - restart demona SSHD