Systemy wbudowane dla automatyki W11

dr inż. Krzysztof Urbański

Dodatkowe materiały do kursu
na prawach rękopisu

Źródła: materiały własne, publicznie dostępne dokumenty w dostępie otwartym oraz noty katalogowe i oficjalna dokumentacja prezentowanych rozwiązań.

Łącząc się z pi4.local / 10.0.0.198 na porcie TCP 8080 dostaniecie w odpowiedzi pewien ciąg znaków.

`SSID: SWA / Klucz: systemywbudowane`

Sieć ma warstwy

1. Fizyczna (physica): **Opisuje fizycznie istniejące medium, za pomocą którego odbywa się wymiana danych. Najpopularniejsze to:**
   a. **IEEE 802.03 – Ethernet 10/100 UTP, STP, fiber**
   b. **IEEE 802.11 a/b/g/n – WiFi**
   c. **GSM/GPRS/LTE**

2. Łącza danych (data link): **Określa strukturę i sposób wymiany danych w warstwie fizycznej. Pojedyncza porcja danych (ramka, datagram) może zawierać adres nadawcy, odbiorcy, sumę kontrolną itp.**
   a. **MTU (Maximum Transmission Unit) – maksymalna dopuszczalna dla danej konfiguracji sieci porcja danych.**
   b. **adresy MAC – Media Access Control. 48-bitowa liczbą zapisywana jako 6 oktetów heksadecymalnie (XX:XX:XX:XX:XX:XX), nazywany też adresem sprzętowym**

3. Sieci (network): **Internetwork Protocol (IP) odpowiadający IP za wyznaczanie tras pakietów. W razie konieczności fragmentacja pakietów (podział na mniejsze części) lub fragmentacja tak, aby nie została przekroczona maksymalna wielkość MTU. Tłumaczenie adresów między warstwą łącza danych a sieci odbywa się dzięki ARP (Address Resolution Protocol)**

4. Transportowa (transport): **TCP (Transmission Control Protocol) – protokół połączeniowy, kontrola kolejności i kompletności przesyłanych danych oraz UDP (User Datagram Protocol) – protokół bezpołączeniowy**

5. Sesji (session): **Warstwa sesji opisuje format wymiany danych niezaleŜnie od stosowanego protokołu (np. implementacja niektórych protokołów zarówno w postaci TCP jak i UDP). W tej warstwie są często „zaszyte" usługi typu RPC, SMB, NFS**

6. Prezentacji (presentation): **Odpowiada za przekształcenie danych do formatu obowiązującego w sieci (np. zamiana kolejności bajtów w razie potrzeby). W Internecie obowiązuje BigEndian.**

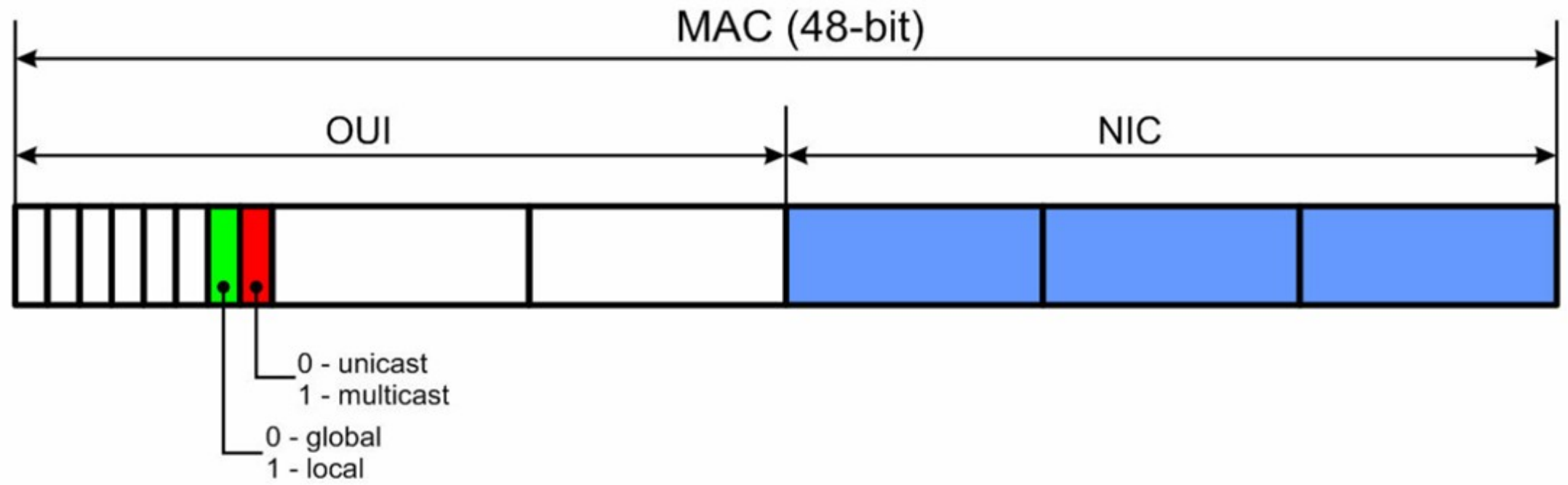
7. Aplikacji (application): **To nie tyle aplikacje, to protokoły aplikacyjne. Na tym kursie HTTP, HTTPS, DNS, NTP, DHCP**

Jak wygląda „adres fizyczny”?

Nazwa pochodzi z dawnych czasów (i czasem dzisiaj spotykanych rozwiązań), kiedy był faktycznie kodowany w specjalnych chipach ROM.

Tak naprawdę jest to adres warstwy drugiej (łącza danych).

Budowa MAC: OUI + NIC



Liczba 48-bitowa (6 oktetów) – i nie jest to dowolna liczba.
OUI – Organizationally Unique Identifier 3 oktety z pewnym ograniczeniem
NIC – Network Interface Card – 3 oktety
Pierwszy oktet OUI zawiera specjalne bity:
Bit0 I/G (Individual/Group) – gdy 1, to adres jest grupowy, w szczególności broadcast FF:FF:FF:FF:FF:FF
Bit1 U/L (Universal/Local). Gdy 0 to UAA - Universally Administered Address (producent), gdy 1 to LAA - Locally Administered Address.

Jak powiązać 2 + 3?

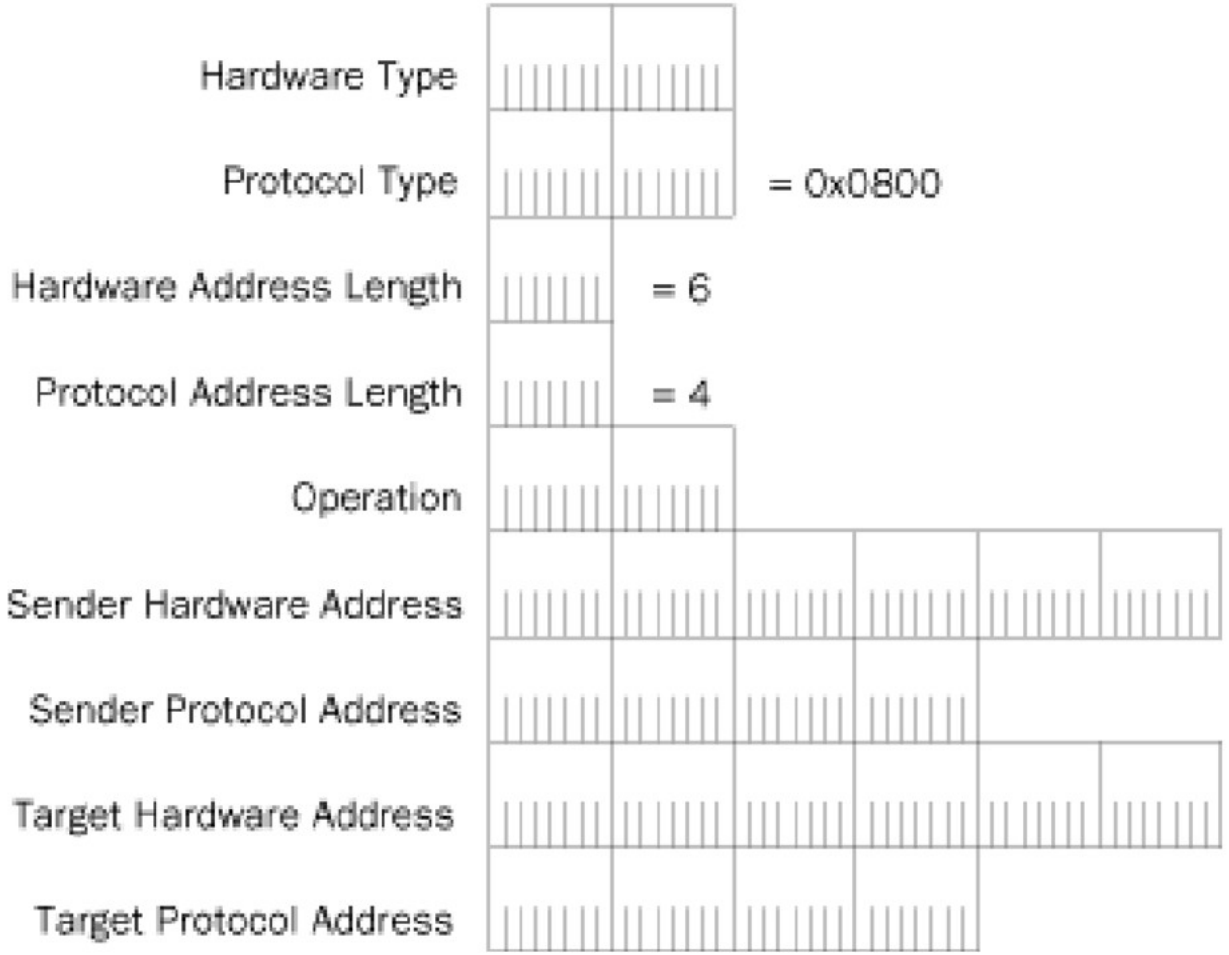
- **Adresy MAC („sprzętowe”) – niepraktyczne**
- **48 bitów w postaci HEX jest trudniejsze do zapamiętania niż 32 bity IP lub nazwa symboliczna (DNS)**
- **Zwykle nie ma stałego przyporządkowania IP-MAC**
- **rozwiązanie: mechanizm automatycznego tłumaczenia adresów warstw 2 oraz 3.**
- **adresacja w warstwie 2 działa praktycznie tylko w segmentach sieci lokalnych**

I oto wkracza cały na biało RFC826 ARP.

Budowa ramki ARP – prosto z RFC:

```
Ethernet transmission layer (not necessarily accessible to
     the user):
    48.bit: Ethernet address of destination
    48.bit: Ethernet address of sender
    16.bit: Protocol type = ether_type$ADDRESS_RESOLUTION
Ethernet packet data:
    16.bit: (ar$hrd) Hardware address space (e.g., Ethernet,
                     Packet Radio Net.)
    16.bit: (ar$pro) Protocol address space.  For Ethernet
                     hardware, this is from the set of type
                     fields ether_typ$<protocol>.
     8.bit: (ar$hln) byte length of each hardware address
     8.bit: (ar$pln) byte length of each protocol address
    16.bit: (ar$op)  opcode (ares_op$REQUEST | ares_op$REPLY)
    nbytes: (ar$sha) Hardware address of sender of this
                     packet, n from the ar$hln field.
    mbytes: (ar$spa) Protocol address of sender of this
                     packet, m from the ar$pln field.
    nbytes: (ar$tha) Hardware address of target of this
                     packet (if known).
    mbytes: (ar$tpa) Protocol address of target.
```

Budowa ramki ARP – grafika:



Budowa ramki ARP – wersja tekstowa

start [ETH_HDR | ARP] CRC stop

ETHHDR ::= HWDST(6) | HWSRC(6) | ETHtype/LEN(2)
ETHtype = 0806_{HEX} (ARP)

ARP ::= [HW_type(2) | PROT_type(2) | HW_len(1) | P_len(1) | OPER(2) | HW_snd(6) | P_snd(4) | HW_tgt(6) | P_tgt(4)]
HWtype = 0001_{HEX} (ethernet)
PROTtype = 0800_{HEX} (IP)
HWlen = 06_{HEX}
Plen = 04_{HEX}
OPER = 1_{HEX} (ARP REQEST), 2_{HEX} (ARP REPLY)

Budowa ramki ARP – wersja w języku C

```
typedef struct tEthernetHeader {
  uint8_t destination[6];
  uint8_t source[6];
  uint16_t type;
} tEthernetHeader;

typedef struct tARPHeader {
  uint16_t  hType;     //HW type
  uint16_t  pType;     //PROT type
  uint8_t   hLength;   //HW length
  uint8_t   pLength;   //PROT length
  uint16_t  operation; //REQ or REP
  uint8_t   SHA[6];
  uint8_t   SPA[4];
  uint8_t   THA[6];
  uint8_t   TPA[4];
} tARPHeader;
```

Przebieg typowej wymiany informacji w ARP

Z komputera 10.0.0.2 wykonałem komendę ping 10.0.0.1

```
# tcpdump -i eth1 –n

00:22:22:22:22:22->FF:FF:FF:FF:FF:FF arp who-has 10.0.0.1 tell 10.0.0.2

00:11:11:11:11:11->00:22:22:22:22:22 arp reply 10.0.0.1 is-at 00:11:11:11:11:11

00:22:22:22:22:22->00:11:11:11:11:11 IP 10.0.0.1 > 10.0.0.255: ICMP echo request

00:11:11:11:11:11->00:22:22:22:22:22 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply
```

Zagrożenia:

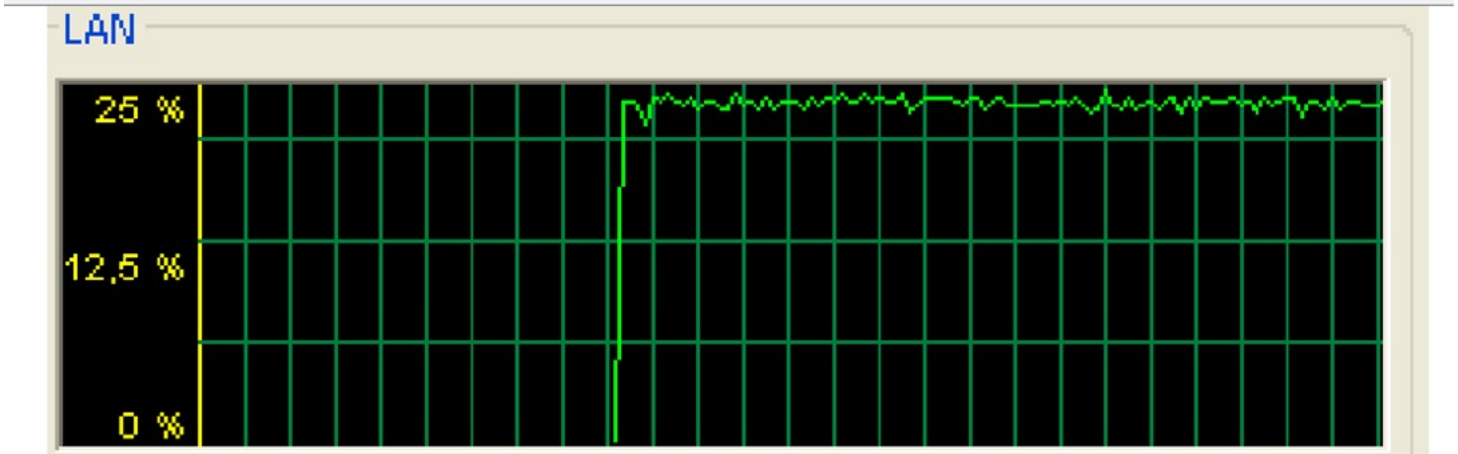
1. A co jeśli złośliwy kolega siedzący obok odpowie na broadcast?
2. A co jeśli wyślę dużo zapytań do jakiegoś urządzenia? Przez całą dobę? Z wszystkich Pico2W które znalazłem w labie?

Przykładowe zagrożenie: ARP flood

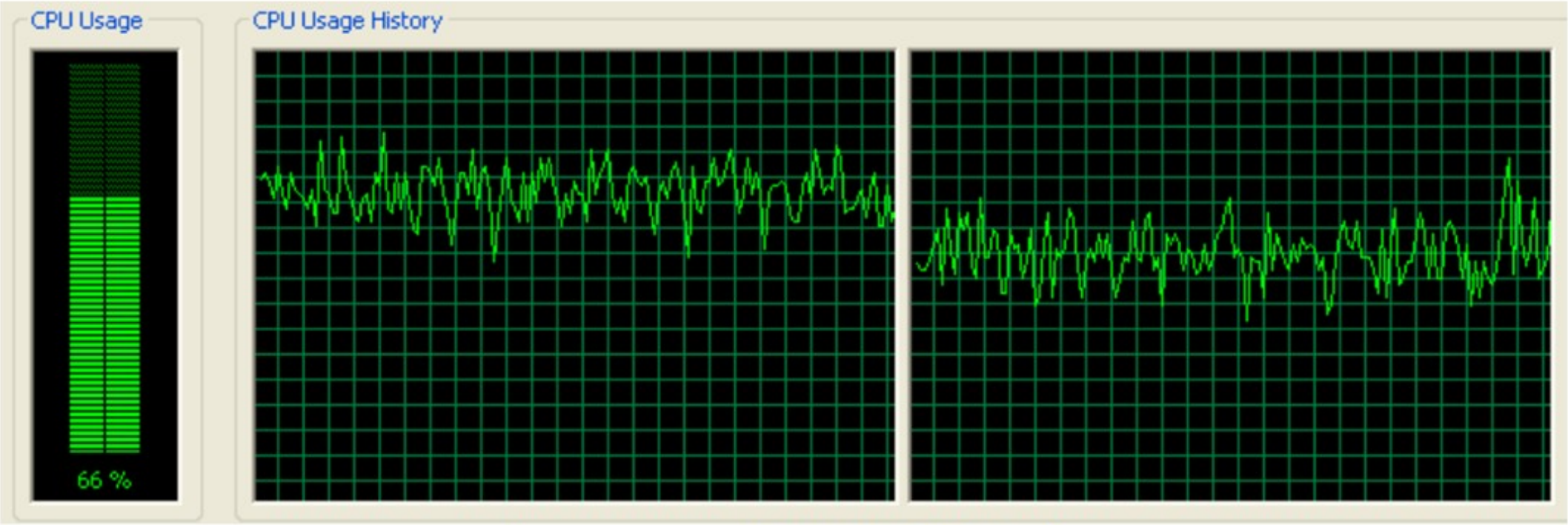
Z pozoru niewinny program:

```
void main() {
   for (;;) fake_arp_request();
}
```

...może spowodować następujące skutki:



Adapter Name	Network Utiliz...	Link Sp...	State
WAN	0 %	100 Mbps	Operational
LAN	23 %	100 Mbps	Operational
LOOPBACK	0 %	10 Mbps	Operational



(procesor P4HT, 3GHz)

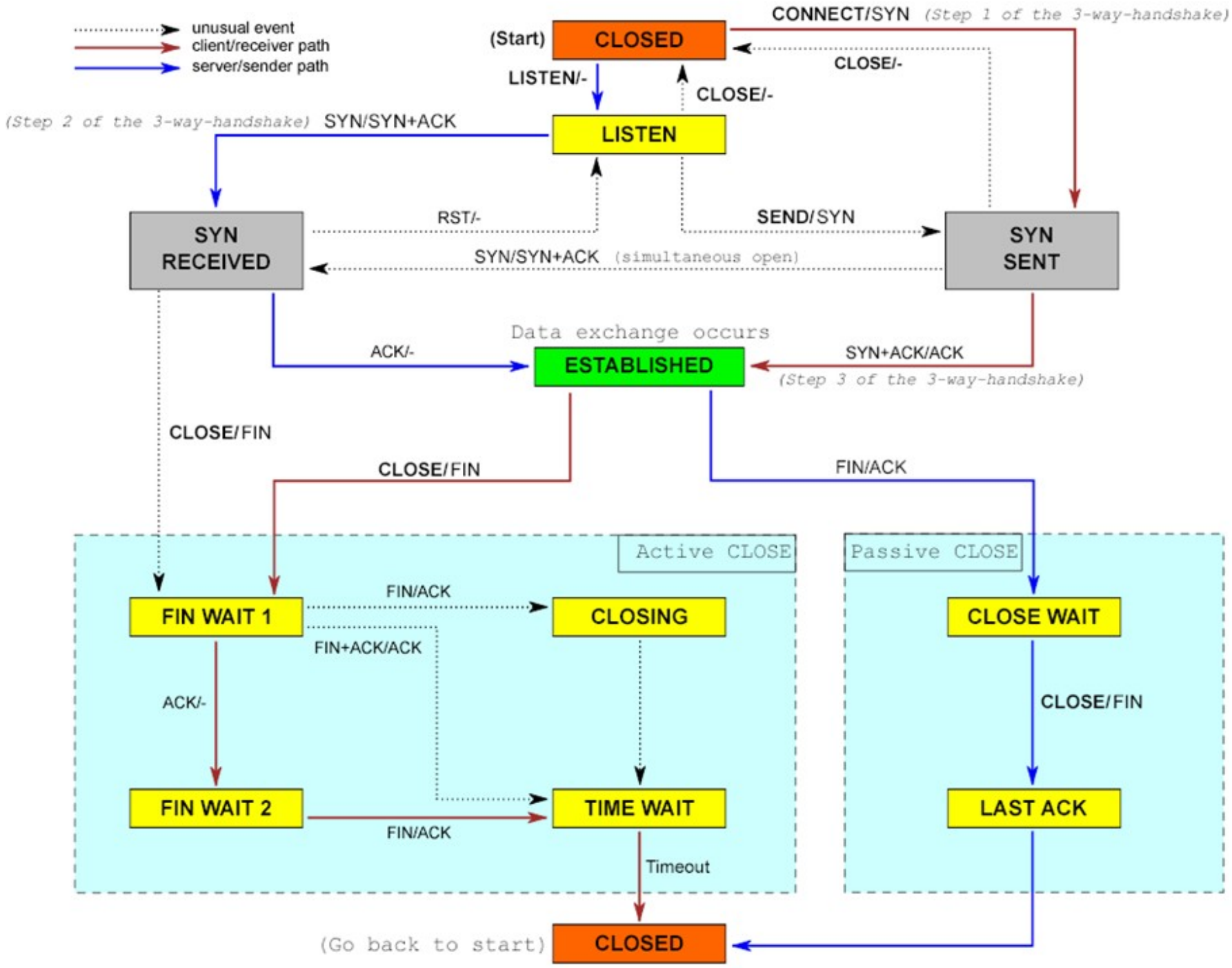
Lekka koloryzacja: wyszperałem screenshot z poprzedniego wcielenia – procesor Pentium 4 HT – efekt jest bardziej dramatycznyn niż na współczesnym i9 czy Ryzen.

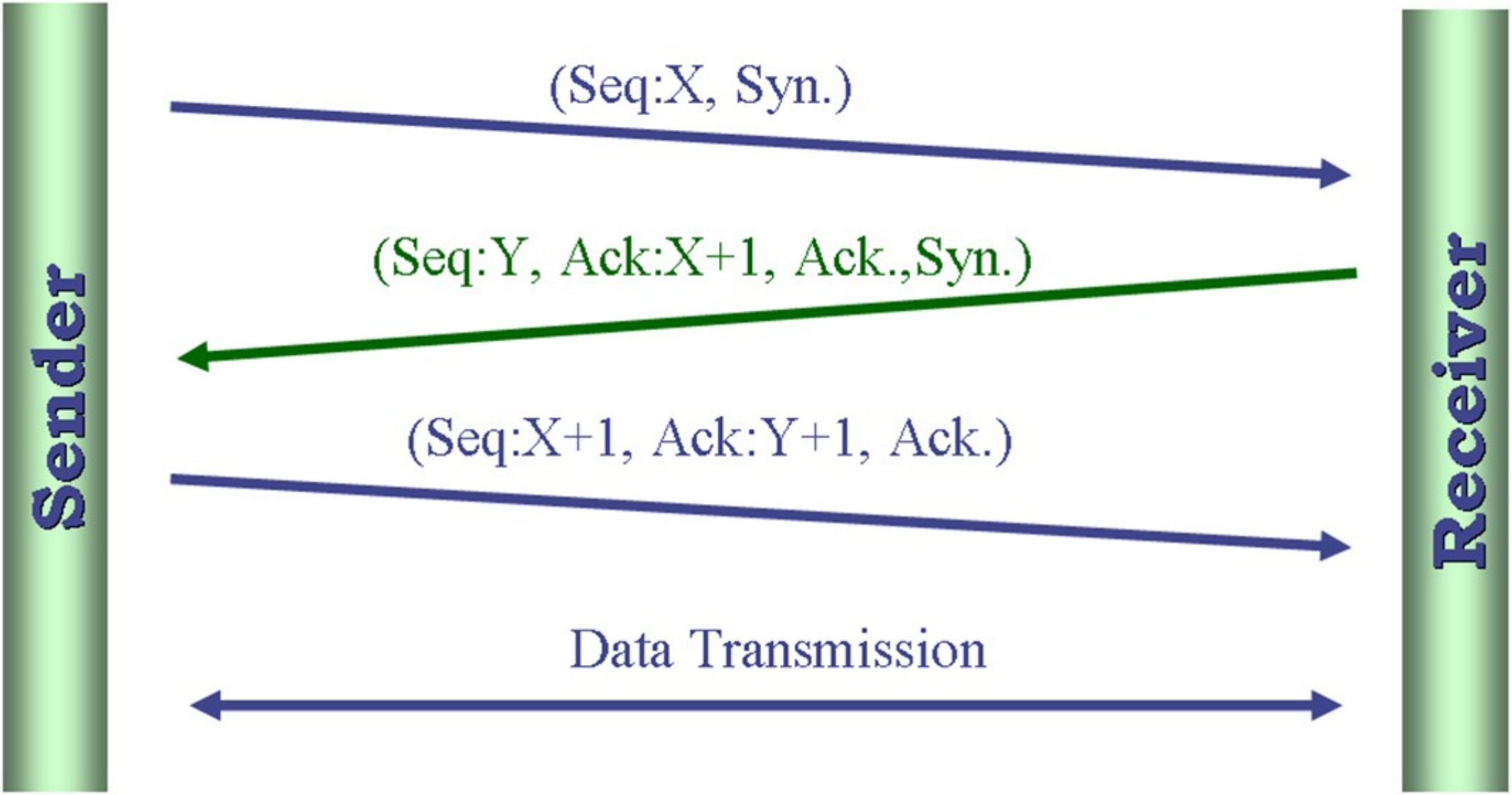
UDP — User Datagram Protocol

- **protokół bezpołączeniowy**
- gniazda SOCK_DGRAM
- zastosowania UDP
- czym jest datagram?
- **adresy broadcast**, multicast, unicast
- co się może przytrafić pakietom UDP, czyli możliwe różne scenariusze odebrania danych → patrz akcja przy barze
- socket, bind, sendto/recvfrom, closesocket

TCP — Transmission Control Protocol

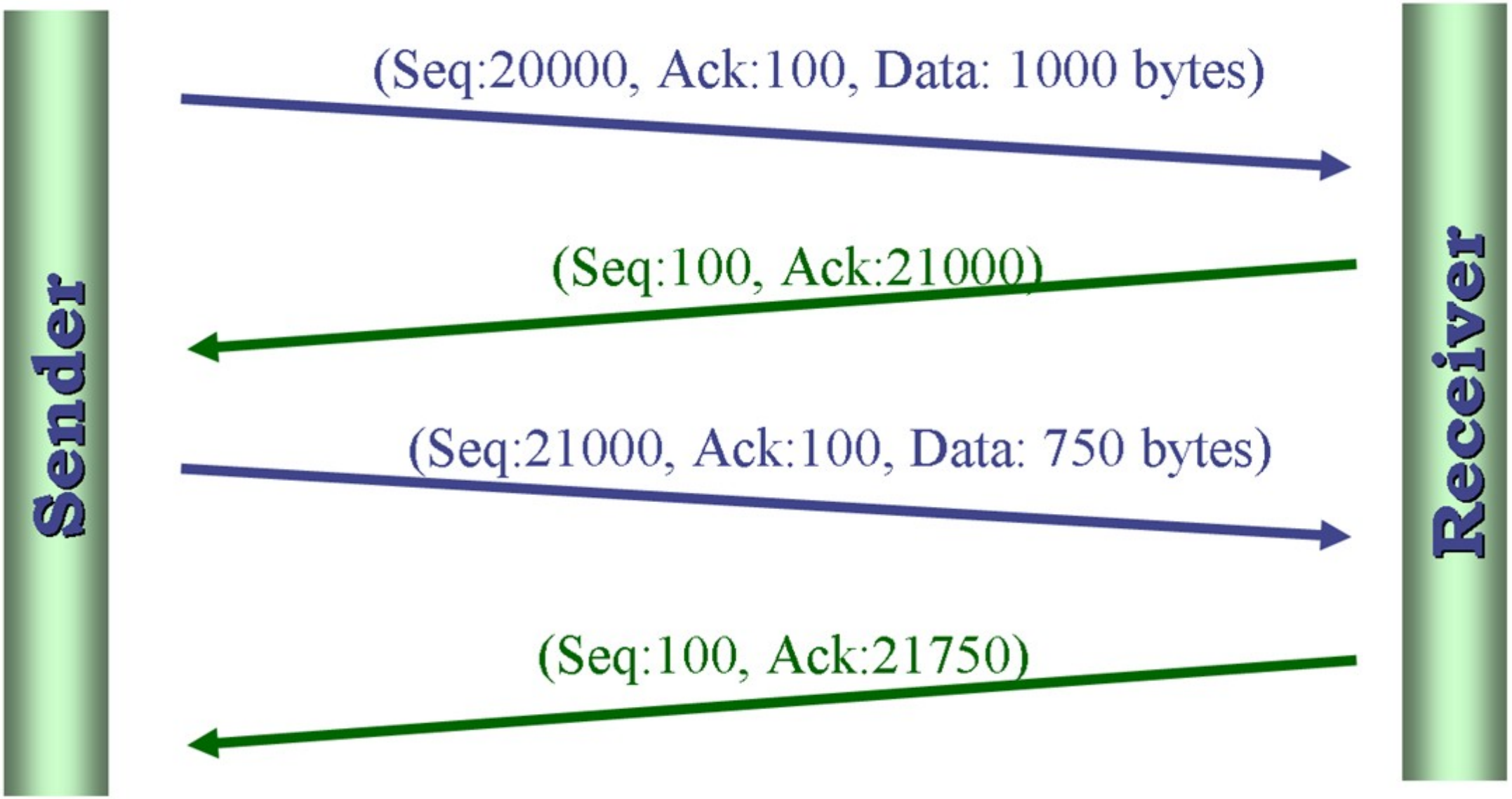
- **protokół połączeniowy**
- gniazda (pełne, połówkowe i serwerowe) SOCK_STREAM
- stos TCP
- nawiązywanie połączenia
- wymiana danych potwierdzona
- **architektura klient-serwer** (asymetria -
- socket, bind, listen+accept/connect, send+recv, closesocket


unusual event
client/receiver path
server/sender path
(Start)
CLOSED
CONNECT/SYN
(Step 1 of the 3-way-handshake)
CLOSE/-
LISTEN/-
CLOSE/-
(Step 2 of the 3-way-handshake)
SYN/SYN+ACK
LISTEN
SYN RECEIVED
RST/-
SEND/SYN
SYN SENT
SYN/SYN+ACK
(simultaneous open)
Data exchange occurs
ACK/-
ESTABLISHED
SYN+ACK/ACK
(Step 3 of the 3-way-handshake)
CLOSE/FIN
CLOSE/FIN
FIN/ACK
Active CLOSE
Passive CLOSE
FIN WAIT 1
FIN/ACK
CLOSING
CLOSE WAIT
FIN+ACK/ACK
ACK/-
CLOSE/FIN
FIN WAIT 2
TIME WAIT
LAST ACK
FIN/ACK
Timeout
(Go back to start)
CLOSED

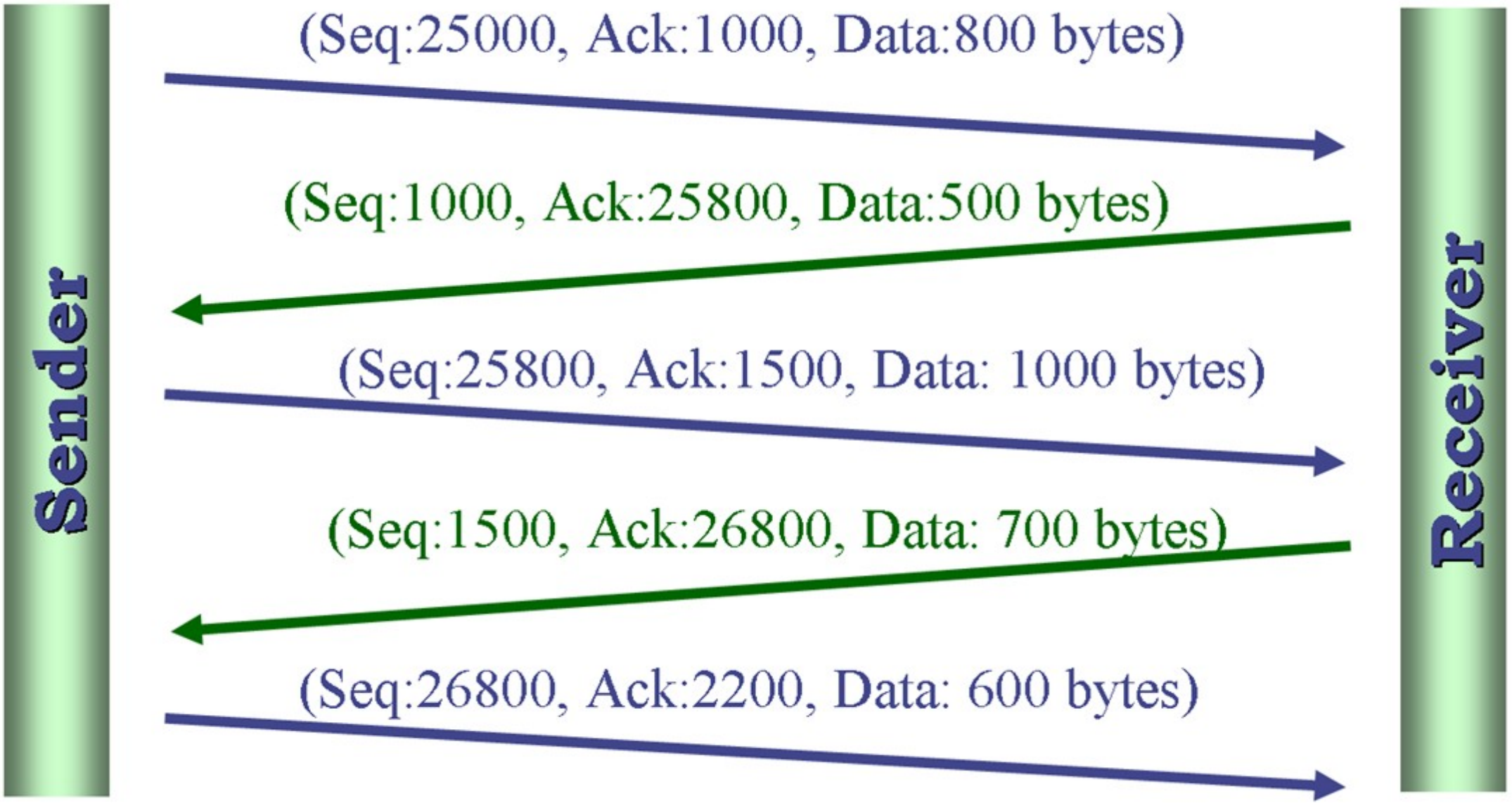


Nawiązywanie połączenia TCP

Trójstronny *handshaking*

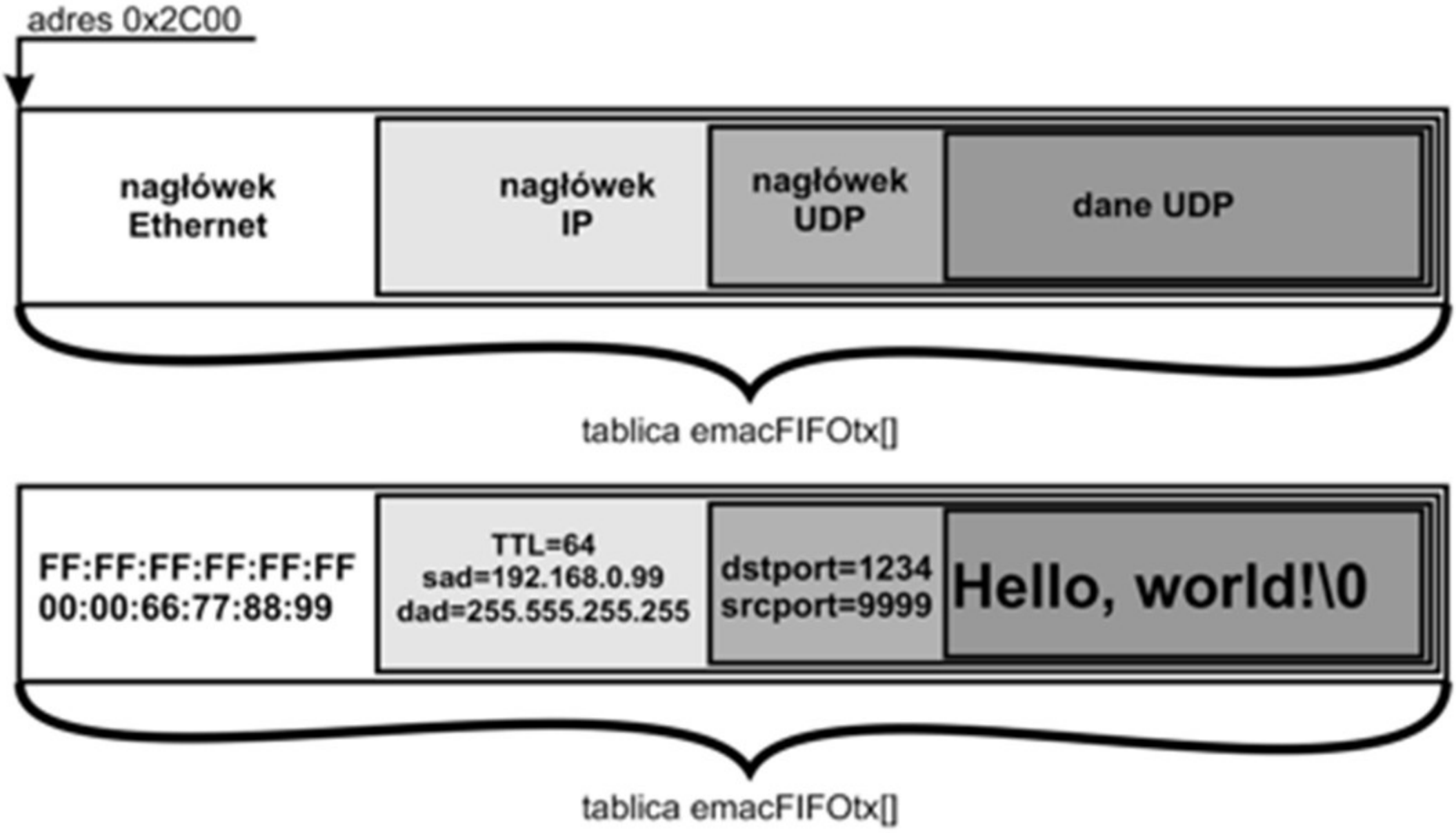


Transmisja jednokierunkowa

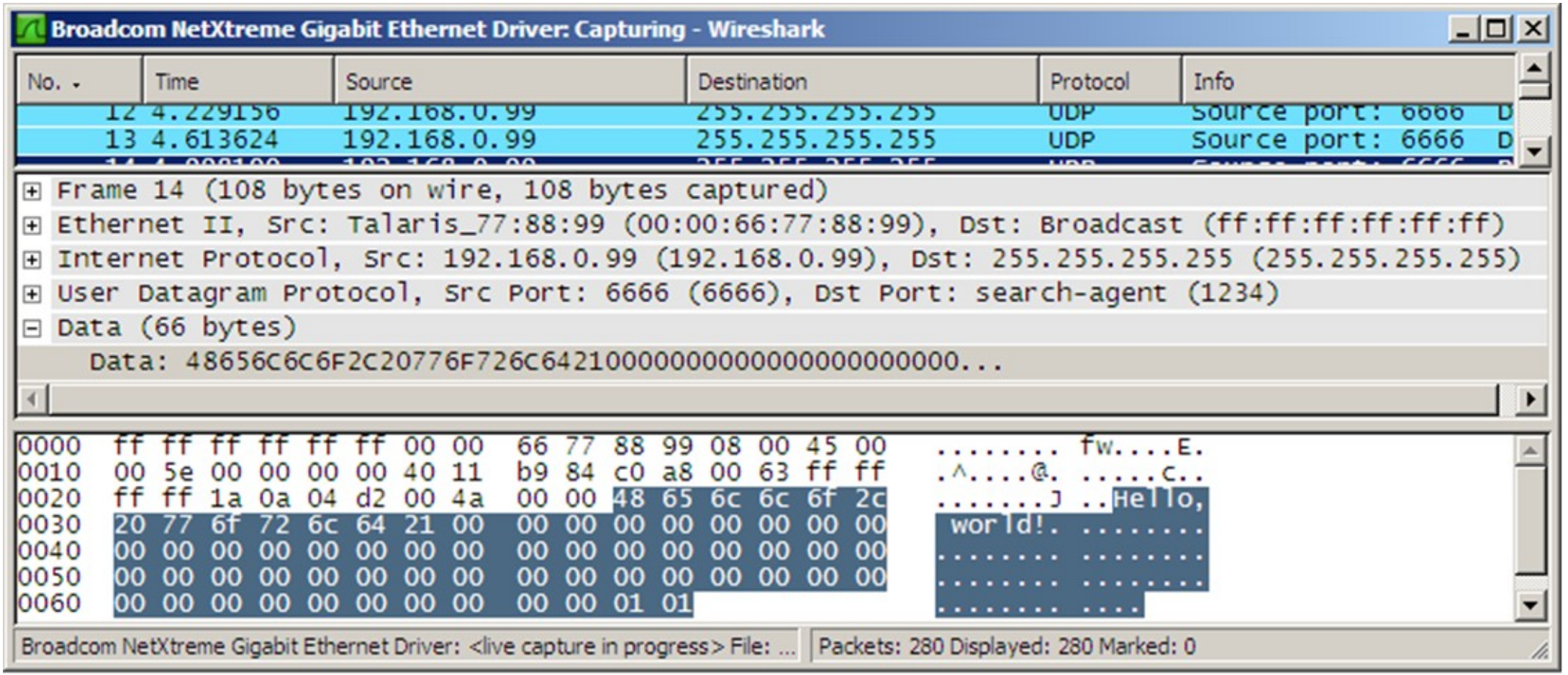


Transmisja dwukierunkowa

Budowa datagramów Ethernet, IP, UDP: enkapsulacja



Przykładowy przechwycony pakiet UDP (Wireshark)



DNS: Domain Name System

- **Narodziny: lata 80-te ubiegłego wieku**
- **Początkowo pliki /etc/hosts**
- **Następnie tzw. tablica hostów, HOSTS.TXT**
- **Obecnie zdecentralizowana usługa katalogowa DNS**
- **istnieje wspólna, nadrzędna domena (root domain, '.')**
- **root domain dzieli się na szereg TLDs – *top-level domains* oraz ccTLDs – *country code top-level domains***
- **każda domena jest zarządzana przez pewną organizację**
- **domena '.' jest zarządzana przez IANA (*Internet Assigned Numbers Authority,* http://www.iana.org).**
- **drzewiasta struktura**
- **w założeniu bardzo wolny przepływ danych (rzadko zmiany)**

FQDN – fully qualified domain name
Przykład: www.pwr.edu.pl
pl – top level domain
edu – second-level domain
…
www – host
Przypadek szczególny: *second-level domain* jest jednocześnie hostem, możemy pominąć www.

TYPE	value and meaning
A	1 a host address IPv4
NS	2 an authoritative name server
CNAME	5 the canonical name for an alias
PTR	12 a domain name pointer
MX	15 mail exchange
AAAA	28 a host address IPv6

HTTP

Obecnie najpopularniejszy jest protokół HTTP 1.1

- tekstowy format rozmowy klient-serwer (za wyjątkiem pewnych danych)
- brak szyfrowania
- korzysta z TCP
- podtrzymywanie konwersacji – możliwa wielokrotna wymiana danych w czasie trwania jednego połączenia (*keep-alive*)
- identyfikacja domeny (klient wysyła symbolicznie nazwę domeny, którą jest zainteresowany) → na jednym publicznym IP można obsłużyć wiele różnych domen symbolicznych (patrz: DNS)

Starszy (ale czasem jeszcze spotykany) HTTP 1.0

- jedno połączenie – jedna wymiana danych (domyślnie)
- brak informacji o domenie

PuTTY (inactive)

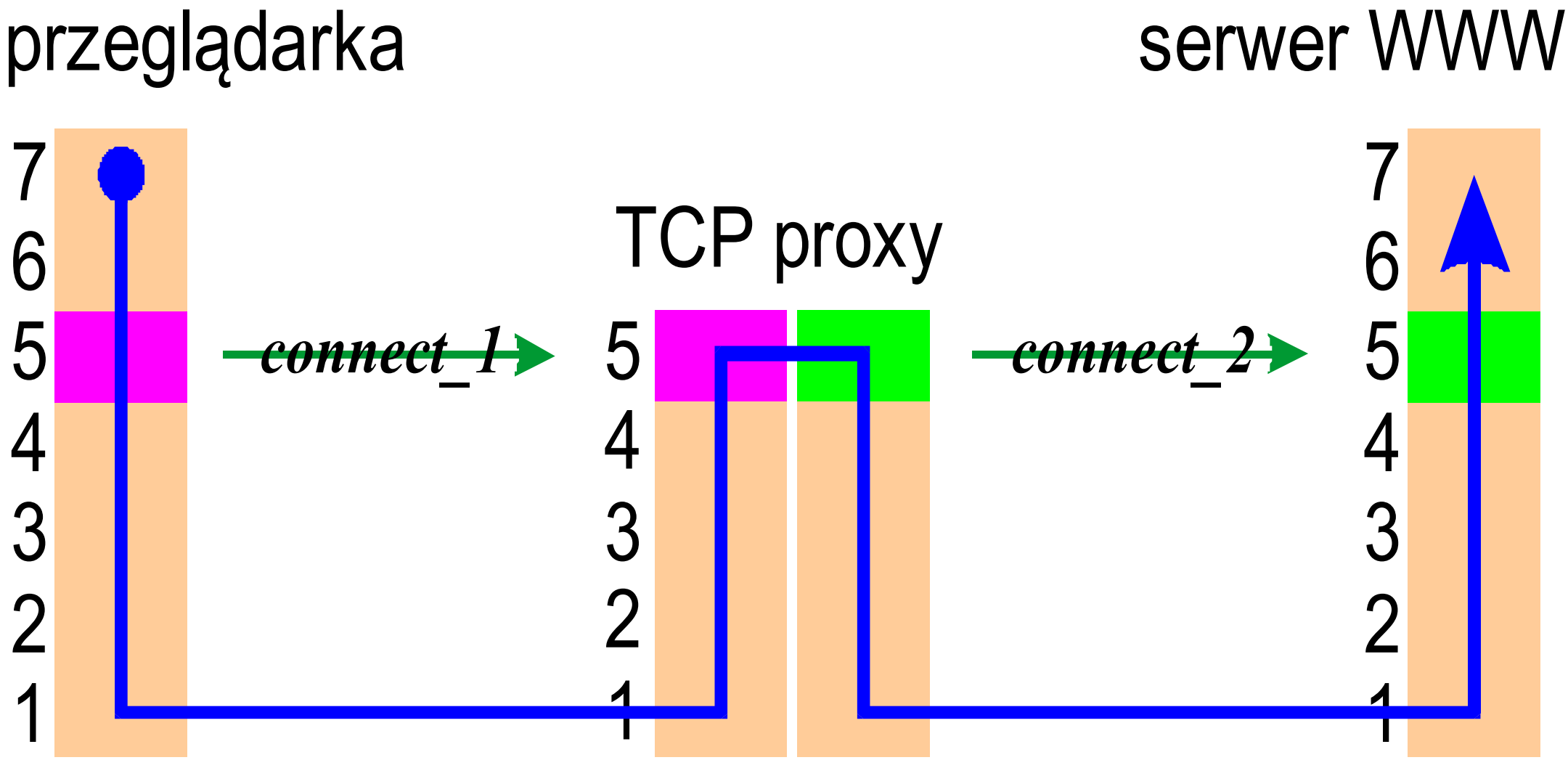
```
GET /f2.html HTTP/1.1
Host: 10.0.0.198

HTTP/1.1 200 OK
Date: Thu, 29 May 2025 20:54:58 GMT
Server: Apache/2.4.62 (Raspbian)
Last-Modified: Thu, 29 May 2025 20:48:53 GMT
ETag: "205-6364c6ced086c"
Accept-Ranges: bytes
Content-Length: 517
Vary: Accept-Encoding
Content-Type: text/html


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/D
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>SWA</title>
  </head>
  <body>
  <h1>FORM GET</h2>

<form action="/f2.html" method="POST">
  <input type="text" name="imie" value="Krzysztof Urbanski">
  <input type="email" name="email" value="krzysztof.urbanski@pwr.edu.pl">
  <input type="submit" value="Wyślij">
</form>

  </body>
</html>
```

klient→serwer (żądanie 2. – zasób /images/hp1.gif)

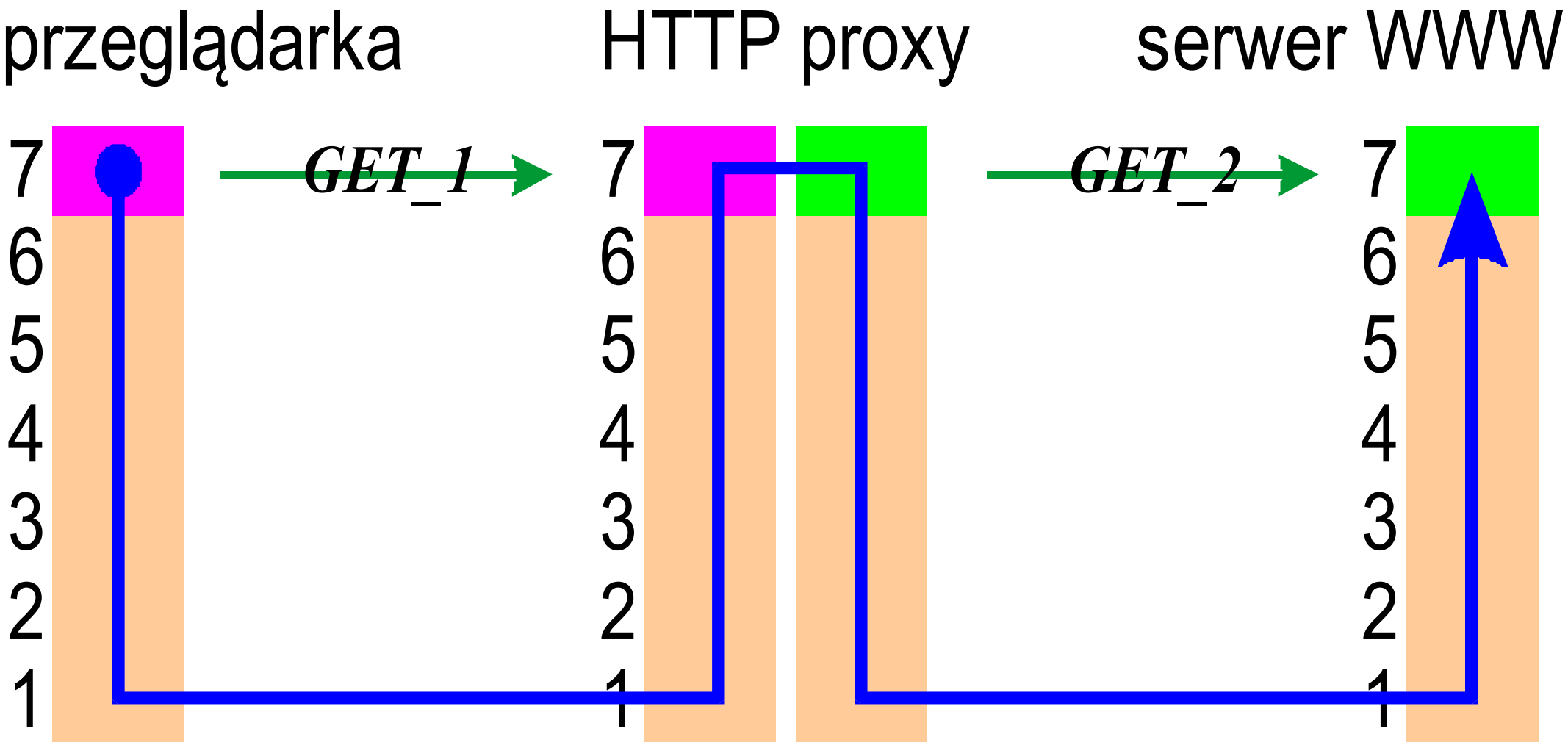
GET /images/hp1.gif HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
Accept: image/png,*/*;q=0.5
sAccept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://127.0.0.1:8080/
Pragma: no-cache
Cache-Control: no-cache
<CRLF>

Odpowiedź na żądanie 2.

HTTP/1.1 200 OK
Content-Type: image/gif
Last-Modified: Mon, 25 Apr 2005 21:06:16 GMT
Expires: Sun, 17 Jan 2038 19:14:07 GMT
Server: GWS/2.1
Content-Length: 2953
Date: Thu, 23 Mar 2006 00:44:40 GMT
<CRLF>
GIF87a2 N w , 2 N
‡¨˝ă˝ď÷ďśÓĄ9şB1¶9ŚËŚ÷÷÷÷óďďëď÷ó÷ďó÷Öă˙Ć×˙-Ç÷Ąľď„Ş÷„šÖsŽÎĄ¶Ö÷÷˙ZŠďJ}çc–
ď9qŢ1eÖ!YÖMĆIµ<Ą4„<””žµďďď˝ű˙çë˙s˘ďE-{†-çăç÷ű˙Rm-9aµcyĄĆÇÎ÷ďď
…



Schemat pośrednika działającego w warstwie gniazd BSD



Schemat pośrednika działającego w warstwie aplikacyjnej (HTTP)

Formularze HTTP: kodowanie

Sposób 1. (domyślny) application/x-www-form-urlencoded

```
<form action="/submit" method="post">
  <input type="text" name="imie" value="Krzysztof Urbanski">
  <input type="email" name="email" value="krzysztof.urbanski@pwr.edu.pl">
  <input type="submit" value="Wyślij">
</form>
```

Zakodowane dane:

`imie=Krzysztof+Urbanski&email= krzysztof.urbanski%40pwr.edu.pl`

Nazwy pól i ich wartości są kodowane:

- spacje są zastępowane znakiem plus (+).
- znaki specjalne (np. &, =, ?, %, #, +, /) są kodowane z użyciem URL encoding, jest to znak procentu (%) i dwie cyfry HEX będace ich kodem ASCII lub UTF-8. Np. znak & to %26, spacja to %20 (lub +).
- pary nazwa=wartość są oddzielane znakiem AND (&).

Całość tworzy jeden ciąg znaków, który jest wysyłany w ciele żądania HTTP (dla metody POST) lub w adresie URL po znaku zapytania (?) (dla metody GET).

Sposób 2. **text/plain**
Podobnie jak 1., ale bez kodowania znaków

Sposób 3. **multipart/form-data**
→gdy wystąpi <input type="file">

```
<form action="/upload" method="post" enctype="multipart/form-data">
  <input type="text" name="opis" value="opis">
  <input type="file" name="plik">
  <input type="submit" value="Wyślij">
</form>

-----------------------------123456789012345678901234567890
Content-Disposition: form-data; name="opis"

opis
-----------------------------123456789012345678901234567890
Content-Disposition: form-data; name="plik"; filename="nazwa_pliku.jpg"
Content-Type: image/jpeg

[...surowe dane binarne pliku...]
-----------------------------123456789012345678901234567890--
```

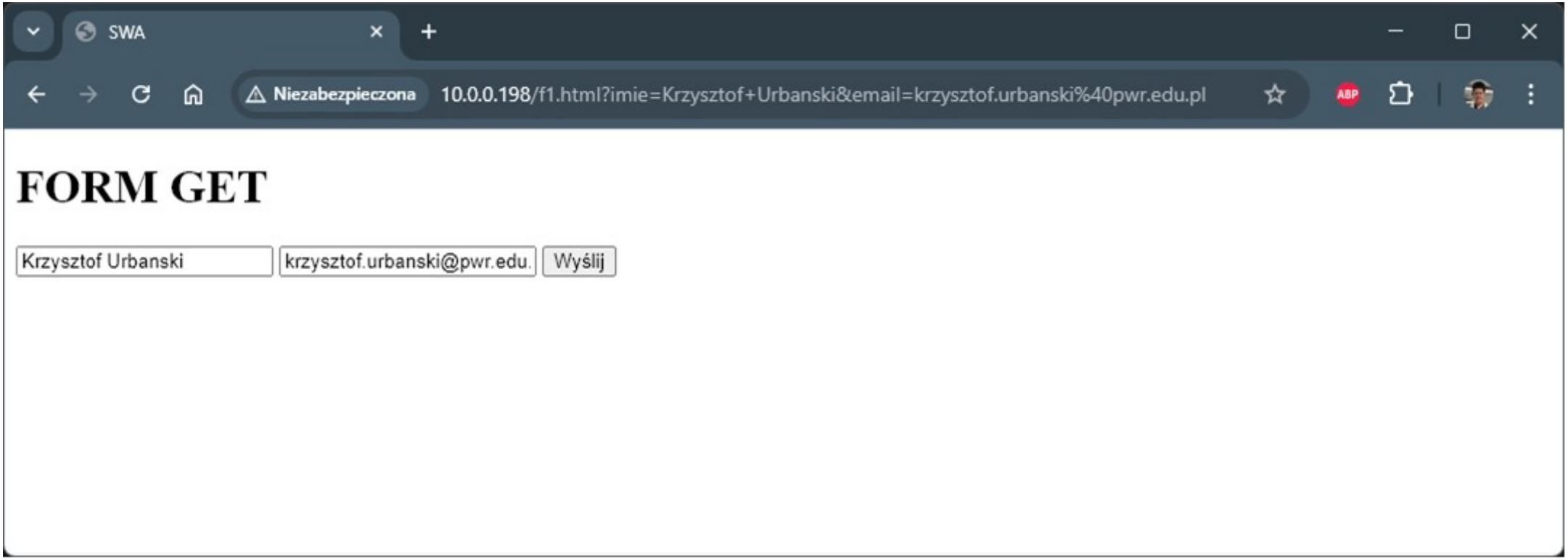
HTTP i jego metody

```
<form action="/f1.html" method="GET">
  <input type="text" name="imie" value="Krzysztof Urbanski">
  <input type="email" name="email" value="krzysztof.urbanski@pwr.edu.pl">
  <input type="submit" value="Wyślij">
</form>
```

```
<form action="/f2.html" method="POST">
  <input type="text" name="imie" value="Krzysztof Urbanski">
  <input type="email" name="email" value="krzysztof.urbanski@pwr.edu.pl">
  <input type="submit" value="Wyślij">
</form>
```

Ale też: PUT, DELETE, HEAD, OPTIONS, CONNECT, TRACE


SWA
Niezabezpieczona
10.0.0.198/f2.html
FORM POST
Krzysztof Urbanski
krzysztof.urbanski@pwr.edu.
Wyślij


SWA
Niezabezpieczona
10.0.0.198/f1.html?imie=Krzysztof+Urbanski&email=krzysztof.urbanski%40pwr.edu.pl
FORM GET
Krzysztof Urbanski
krzysztof.urbanski@pwr.edu.
Wyślij

Chcę oglądać twoje logi!

A konkretnie plik /var/log/apache2/access.log

156.17.46.4 - - [23/Mar/2006:02:57:06 +0100] "**GET /urbanski HTTP/1.1**" **301** 319 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322)"
156.17.46.4 - - [23/Mar/2006:02:57:06 +0100] "**GET /urbanski/ HTTP/1.1**" **200** 4561 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322
)"

156.17.46.4 - - [23/Mar/2006:03:03:57 +0100] "**GET /sieci HTTP/1.1**" **404** 288 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322)"

```
[root@localhost root]# telnet wemif.net 80

Connected to www.wemif.pwr.wroc.pl (156.17.46.1).

GET /

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>403 Forbidden</TITLE>
</HEAD><BODY>
<H1>Forbidden</H1>
You don't have permission to access /index.html
on this server.<P>
</BODY></HTML>
Connection closed by foreign host.
```

```
[root@localhost]# telnet wemif.net 80
Connected to www.wemif.pwr.wroc.pl (156.17.46.1).

GET /sk/ HTTP/1.1
Host: www.wemif.net

HTTP/1.1 200 OK
Date: Wed, 27 Apr 2005 23:34:38 GMT
Server: Apache-AdvancedExtranetServer...
Last-Modified: Wed, 27 Apr 2005 20:29:30 GMT
Accept-Ranges: bytes
Content-Length: 6130
Content-Type: text/html
<CRLF>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<title>WEMiF - Sieci komputerowe</title>
...
```

GET /sk HTTP/1.1
Host: www.wemif.net

HTTP/1.1 301 Moved Permanently
Date: Wed, 27 Apr 2005 23:37:15 GMT
Server: Apache-AdvancedExtranetServer...
Location: http://www.wemif.net/sk/
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

e0
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A
HREF="http://www.wemif.net/sk/">here</A>.<P>
</BODY></HTML>

GET /sk HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Wed, 27 Apr 2005 23:38:52 GMT
Server: Apache-AdvancedExtranetServer...
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

129
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>400 Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
client sent HTTP/1.1 request without hostname (see RFC2616 section 14.23): /sk<P>
</BODY></HTML>

0

A jeśli użyjemy HTTP 1.0?

Connected to www.wemif.net (156.17.46.1).

GET /sk

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /sk was not found on this server.<P>
</BODY></HTML>
Connection closed by foreign host.
```

Wybrane fragmenty access_log po tych operacjach

74.125.43.121 - - [23/Apr/2005:10:19:33 +0200] "HTTP/1.1 301 Moved Permanently" 400 299 "-" "-"
74.125.43.121 - - [23/Apr/2005:10:20:02 +0200] "GET /sk HTTP/1.1" 301 236 "-" "-"
74.125.43.121 - - [23/Apr/2005:10:20:06 +0200] "GET /sk/" 404 - "-" "-"

...oraz po innej operacji:
156.17.*.* - - [28/Apr/2005:01:44:59 +0200] "GET /sk/w8.pdf HTTP/1.1" 200 345378 "http://www.wemif.net/sk/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; pl-PL; rv:1.7.7) Gecko/20050414 Firefox/1.0.3"

GET /sk/ HTTP/1.1
Host: wemif.net
User-Agent: Moja Wlasna Przegladarka
Referer: http://dowolna.domena.com/a_tutaj/cokolwiek.html
Accept: text/html;q=0.9,
text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,
text/css,*/*;q=0.1
Accept-Language: pl, en
Accept-Encoding: gzip;q=0.9
Accept-Charset: ISO-8859-2, windows-1250;q=0.66, *;q=0.66
Range: 0-100
Keep-Alive: 30
Connection: keep-alive

74.125.43.121 - - [23/Apr/2005:11:58:24 +0200] "GET /sk/ HTTP/1.1" 200 6130 "http://dowolna.domena.com/a_tutaj/cokolwiek.html" "Moja Wlasna Przegladarka"

To trzeba wiedzieć

1. Różnice między HTTP w wersji 1.0 i 1.1. Wpływ na przebieg i szybkość transmisji zasobów WWW w warunkach znacznego obciążenia łącza, na różne odległości.
2. Klient i serwer HTTP w systemach wbudowanych – co trzeba zrobić, aby zadziałało.
3. Jak są kodowane formularze HTTP?
4. Kodowanie danych: XML a JSON
5. Porównanie metod GET oraz POST.
6. Kodowanie znaków specjalnych w formularzach.