

SIK = Sieci komputerowe

Tematyka całego wykładu SIK:

- Całokształt zagadnień związanych z sieciami komputerowymi, struktura intersieci, sieci fizyczne, protokoły niższych i wyższych warstw, interfejs programistyczny sieci (gniazdka), budowa i historia Internetu, konfiguracja sieci pod linux-em, narzędzia do symulowania i analizowania sieci/protokołów, ...

Plan najbliższych wykładów:

- Zarys sieci komputerowych
sieci fizyczne, intersieć, adresy węzłów, routery,
prot niższych warstw: fiz., IP, TCP, UDP,
prot wyższych warstw: FTP, HTTP, SSL, DNS, mailowe, ...

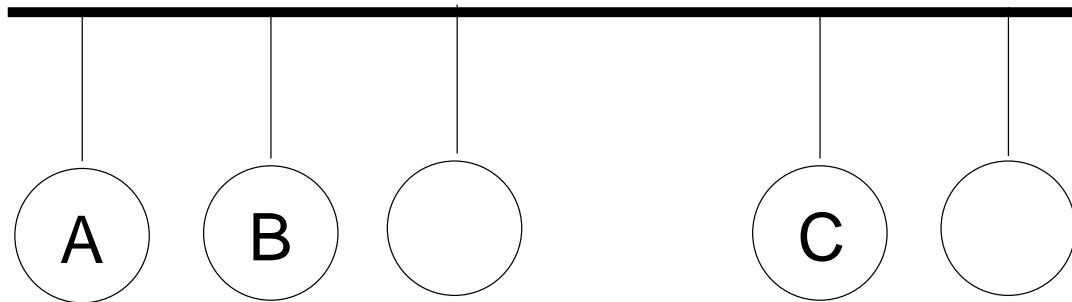
Literatura

- Comer, "Sieci komputerowe TCP/IP, tom 1", (stara książka)
- Kurose, Ross, "Sieci, od szczegółu do ogółu z internetem w tle", (nowa książka)
- Dordal, "An Introduction to Computer Networks", (pdf)
- dokumenty RFC: <https://tools.ietf.org/html/>
- materiały w wikipedii ...

Intersieć i sieci fizyczne

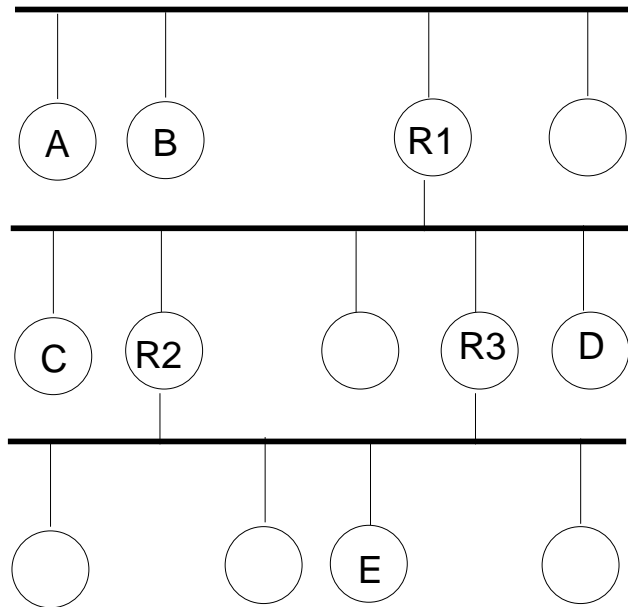
- Sieć fizyczna umożliwia wysyłanie pakietów między węzłami.
- Komputer podłączony do sieci to "węzeł".
inne nazwy węzła: host, maszyna
- Węzeł A może wysłać pakiet do węzła B (ang. unicast)
lub do wszystkich w tej samej sieci fizycznej (ang. broadcast)
- Pakiet = nagłówek + dane; inne nazwy: ramka, datagram, komunikat
nagłówek pakietu zawiera m.in. adresy sprzętowe źródłowy i docelowy...
- Interfejs sieciowy węzła (= karta sieciowa) posiada adres sprzętowy (np. eth)
patrz polecenie ifconfig ...

Sieć fizyczna:



Intersieć i sieci fizyczne

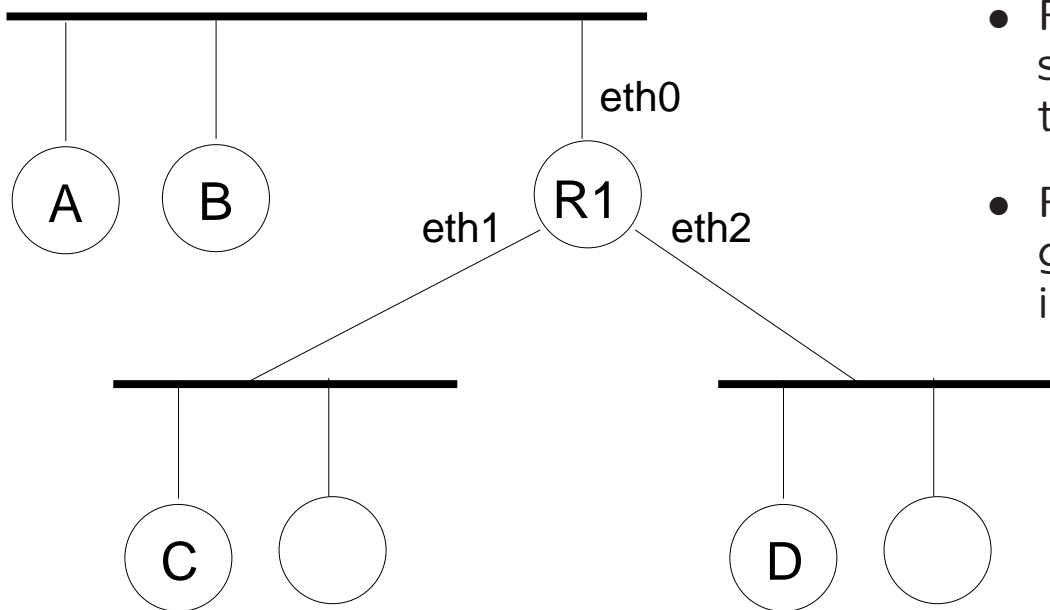
Intersieć:



- Intersieć składa się z kilku sieci fizycznych
- Przykład intersieci: Internet...
- Węzeł A może wysłać pakiet do węzła E, wtedy pakiet przeskakuje przez 2 routery, np. R1 i R2 lub R1 i R3
- Router = węzeł podłączony równocześnie do kilku sieci fizycznych, przekazuje pakiety
- Linux może działać jako router...
- "wszystkie sieci są równe"
- w Internecie: podział intersieci na AS (systemy autonomiczne) z uwagi na złożoność...

Intersieć i sieci fizyczne

Intersieć:



- Router R1 jest podłączony równocześnie do 3 sieci fizycznych
- Router decyduje przez który interfejs sieciowy wysłać pakiet to tzw. trasowanie (ang. routing).
- Router robi to na podstawie tablicy routin-gowej i adr docelowego pakietu (adr IP)

Typy sieci fizycznych

- LAN - Sieć lokalna (ang. Local Area Network)
WAN - Sieć WAN (z ang. Wide Area Network, rozległa sieć komputerowa)
WLAN - Bezprzewodowa LAN (ang. Wireless Local Area Network)
MAN - Miejska sieć komputerowa (ang. Metropolitan Area Network)
- topologia sieci: magistrala, gwiazda, ring, drzewo - dotyczy sieci fizycznej!!
- sieć lokalna typu **Ethernet**
 - standard IEEE 802.3, Ethernet II
 - skrętka (kabel) + switch (urządzenie sieciowe)
 - skrętka nieekranowana kategori e5, do 100metrów, 100Mbit/s = Fast Ethernet, są jeszcze szybsze!, wtyczka RJ-45, nie przejmować się przeplotem!
 - maszyny w sieci mają karty sieciowe Ethernet, >2 maszyny łączymy przy pomocy switch-a => topologia gwiazdy/drzewa
 - dawniej używano kabla koncentrycznego lub skrętki + hub (koncentrator/repeater)
- bezprzewodowa sieć lokalna typu **WiFi**
 - standardy IEEE 802.11, 802.11b/g/n
 - punkt dostępowy (ang. Access Point), klienci WiFi (czyli maszyny z kartami sieciowymi WiFi)

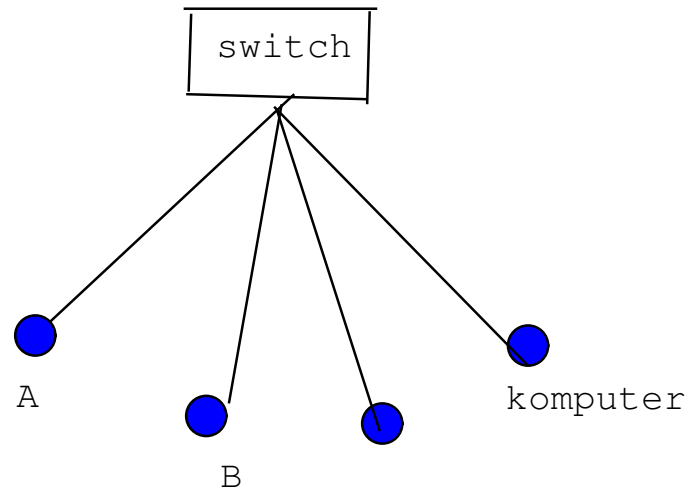
- sieci dwuwęzłowe nad łączem szeregowym
 - połączenie telefoniczne, modemy telefoniczne/akustyczne, 56Kbitów/s, bardzo długie łącze szeregowe
 - prot PPP (ang. Point to Point Protocol) przenosi pakiety IP nad łączem szeregowym (demony pppd + moduł kernelowy)
 - to jest prosty przykład sieci WAN !
 - łącze szeregowe może być emulowane (nad bluetooth/rfcomm lub nad USB)
- bezprzewodowa "personalna" sieć **Bluetooth**
 - standardy IEEE 802.15
 - prot rfcomm (łącze szeregowe nad bt), profile (usługi): DUN, PAN, OBEX, ...
- technologie sieci dostępowych (dostęp do Internetu):
 - ADSL (kabel tel, szerokopasmowy),
 - HFC ("kablówka", światł + kabel koncentryczny),
 - FTTH (GPON, światłowód)

Ethernet - C.D.

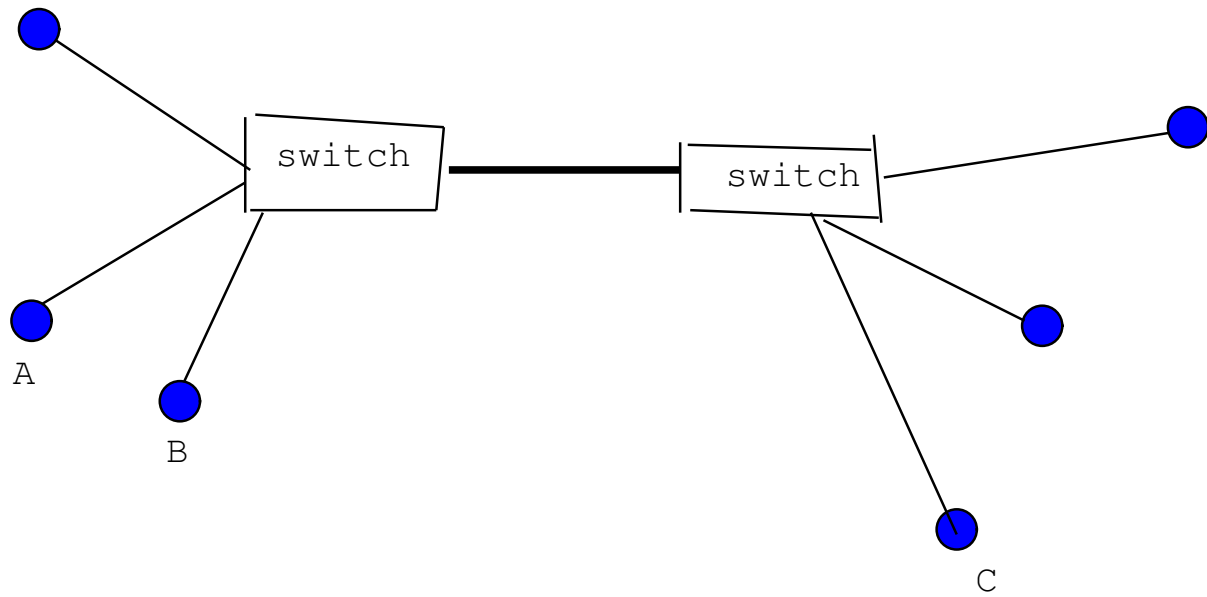
RJ-45 (8P8C),
skrętka:



- switch - pol. przełącznik, operuje na ramkach ethernetowych, dawniej bridge (pol. most), ma kilka "portów" (gniazdek RJ-45)
- switch-e można w prosty sposób łączyć kablem typu skrętka, tworząc "drzewo"
- **zasada działania switch-a:** jeśli nie wie gdzie wysłać ramkę eth, to wysyła wszędzie; poza tym dla każdego portu (gniazdka RJ-45) pamięta jakie adresy eth się za nim kryją ...
- switch vs router ?!?!?!?!?
- tzw "routery WiFi" zawierają switch + access point WiFi (połączone), pojedyncza sieć fizyczna ...



A -> B
B -> A



Adresy węzłów

właściwie nie węzłów tylko interfejsów sieciowych węzłów...

typy adresów: sprzętowe, IP, domenowe

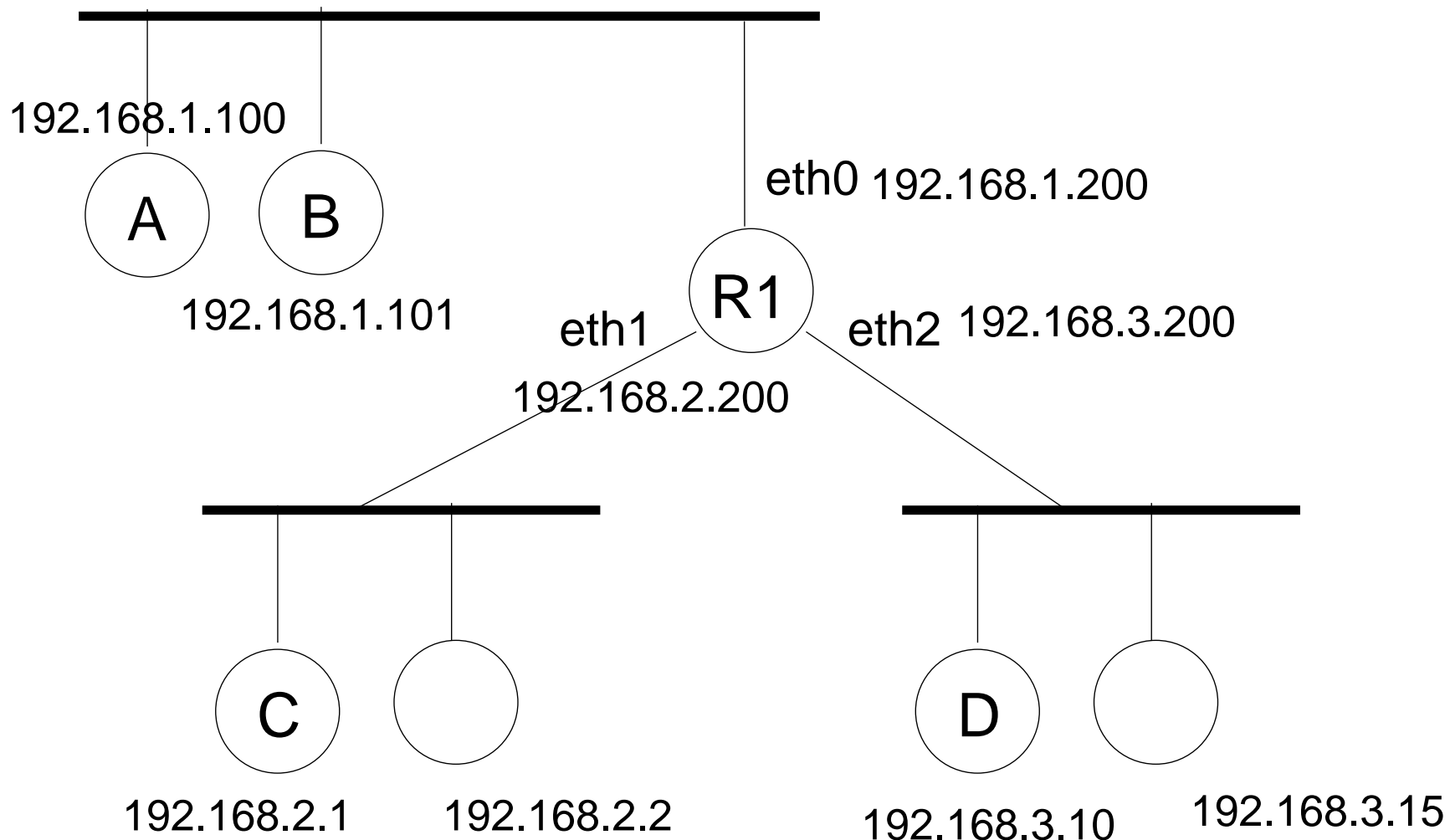
- adresy sprzętowe
np. ethernetowe, 08:9E:01:1C:9C:70, inna nazwa: adr MAC
nadawane przez producenta karty sieciowej
- adresy IP
np. 192.168.1.100, 150.254.77.44, $4 \times 8 = 32$ bity (IPv4),
przydzielanie adresów IP do interfejsu sieciowego: ręcznie, DHCP, ...
- adresy domenowe
np. wp.pl, onet.pl; serwer DNS zamienia adr domenowy na adr IP

zasady przydzielania adresów IP:

- adres IP składa się z "nr sieci" (prefiks) i z "nr hosta"
- wszystkie węzły w danej sieci fizycznej powinny mieć ten sam "nr sieci" oraz inne "nr hosta"
- wniosek: węzły intersieci mają różne adresy IP
(uwaga na NAT - to jest wyjątek !!!)
- jeśli węzeł należy do kilku sieci to będzie miał kilka adr IP

- które bity adresu IP są nr sieci, a które nr hosta?
to zależy od "klasy adresu" (dawniej) i "maski podsieci" (obecnie)
- klasa adresu X1.X2.X3.X4; decyduje prefiks bajtu X1 w zapisie binarnym!
 - klasa A: 0... , nr sieci to X1
 - klasa B: 10..., nr sieci to X1.X2
 - klasa C: 110..., nr sieci to X1.X2.X3
 - klasa D: 1110..., multicasting
- maska podsieci
 - określa jawnie które bity adresu IP są nr sieci (jedyńki w masce)
 - np. maska 255.255.255.0 dla adresu IP klasy B oznacza, że nr sieci to X1.X2.X3
 - jedynki w masce nie muszą koniecznie być spójne ani obejmować całych bajtów
 - wszystkie hosty w danej sieci fizycznej powinny mieć tą samą maskę
- adresy specjalne
 - nr hosta same jedynki: broadcast
 - nr host same zera: "ten host" (2 nr hosta są zakazane !!)
 - 127.0.0.1 = local loopback, localhost, lokalna maszyna
 - adresy prywatne (gdy nie mamy przydzielonego nr sieci w Internecie)
 - 192.168.0.0 -> 192.168.255.255
 - 10.0.0.0 -> 10.255.255.255
 - 172.16.0.0 -> 172.16.255.255
- FLSM vs VLSM (Fixed/Variable Length Subnet Mask)
jeden nr sieci nie powinien być prefiksem drugiego ?!?!
adr pryw: FLSM, adr pub VLSM dla oszczędności
(pokazać rysunek)

Intersieć z przypisanymi adresami IP (klasy C):



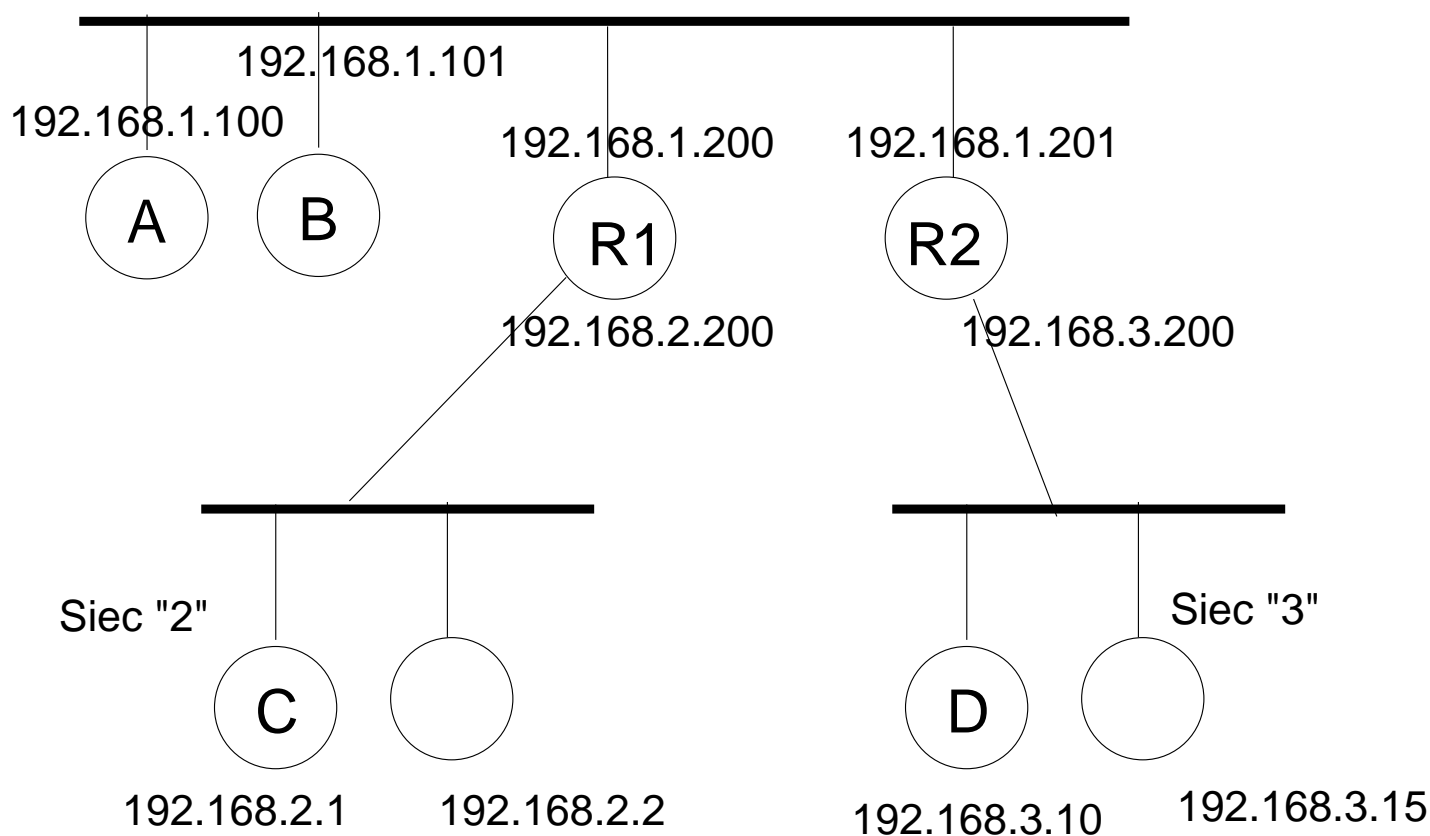
Intersieć z przypisanymi adresami IP (klasy C):

pojęcie sieci "bliskiej" i "dalekiej"...

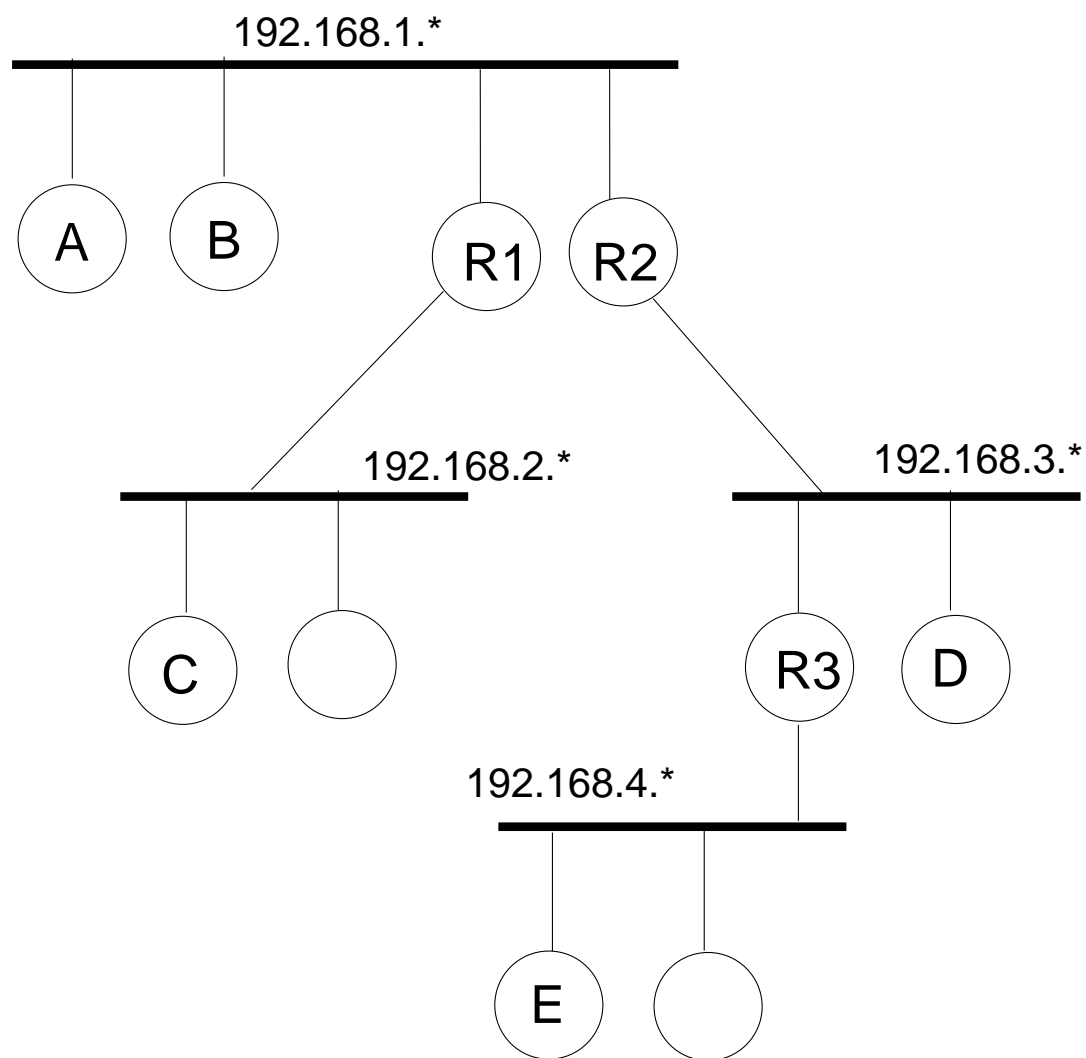
sieć bliska to ta, do której jesteśmy bezpośrednio podłączeni,

węzeł A musi wiedzieć, przez który ruter (gateway) dostać się do sieci "2" i "3" !!

w tabl. routingowej są: reg. dla sieci bliskich, dalekich, default gw



jakie wpisy muszą być w tablicach routinguowych węzłów i routerów ?



"Net-tools" = polecenia liniuxa do konfigurowania sieci

ifconfig - konfigurowanie interfejsu sieciowego

route - odczyt/modyfikowanie tablicy routingowej

iptables - filtrowanie pakietów, zaporą + NAT

ping, traceroute - testowanie sieci

dhcpcd - automat konfig interfejsu sieciowego

(patrz [http://150.254.78.111:20002/zajecia/_xowiki2/SIK_cw Tematy A i F](http://150.254.78.111:20002/zajecia/_xowiki2/SIK_cw_Tematy_A_i_F))

Protokoły IP, TCP, UDP

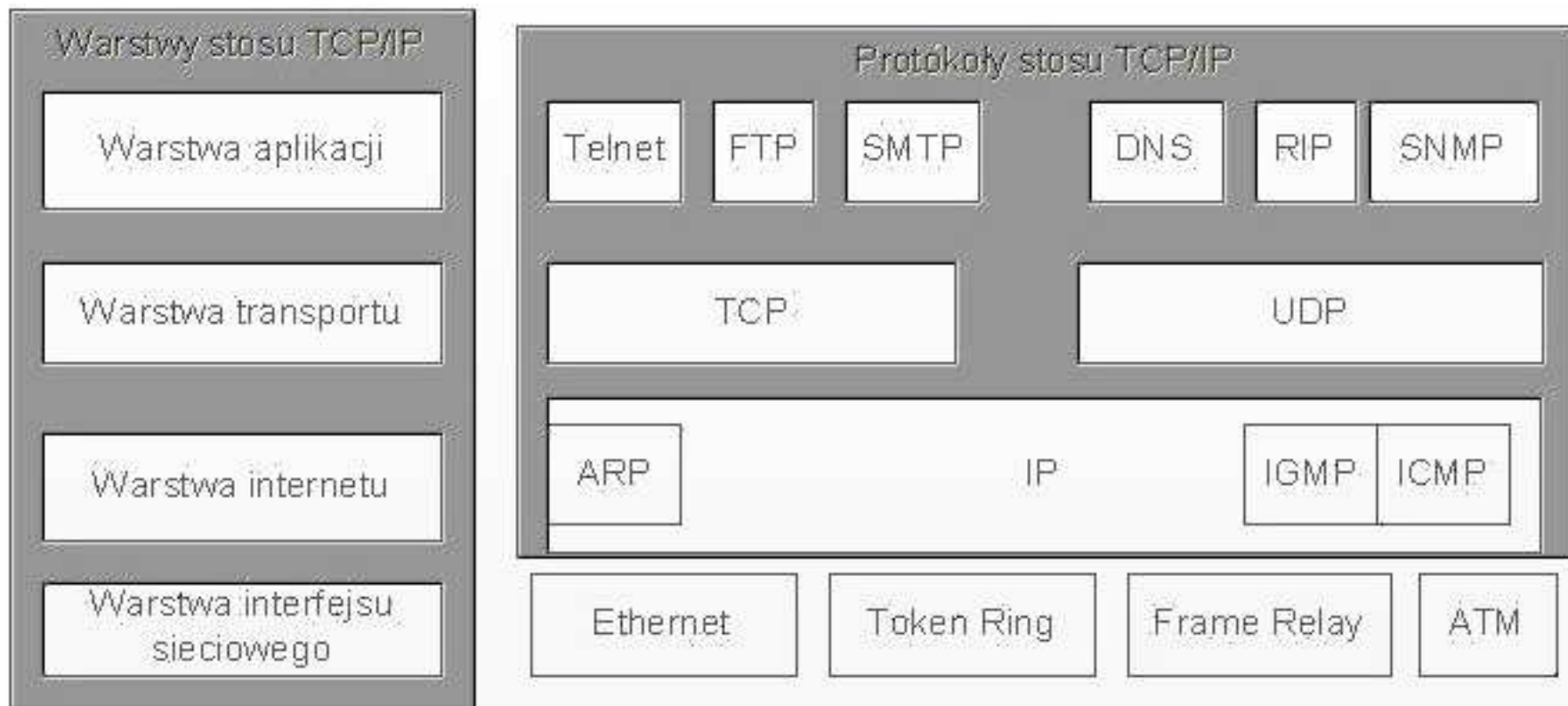
- Co to jest "protokół" ?
 - sposób w jaki hosty rozmawiają przez jakiś kanał komunikacyjny
 - m.in. definiuje format komunikatów
- prot IP - warstwa internetowa
 - przenoszenie pakietów IP przez intersiec
- prot UDP - warstwa transportowa
 - przenoszenie datagramów UDP (są nr portów)
 - niepewne
- prot TCP - warstwa transportowa
 - (wirtualne) połączenie TCP
 - można przesyłać strumień danych/bajtów
 - jest pewne
- aplikacje używają prot warstwy transportowej za pomocą gniazdek BSD (API, fun. systemowe)

Warstwy protokołów

Architektura warstwowa:

wyższa warstwa używa niższej warstwy (patrz enkapsulacja)

Podział protokołów Tcp/Ip na warstwy:



Warstwy ISO vs warstwy Tcp/Ip:



nieoczywiste warstwy ISO:

fizyczna (kabel)

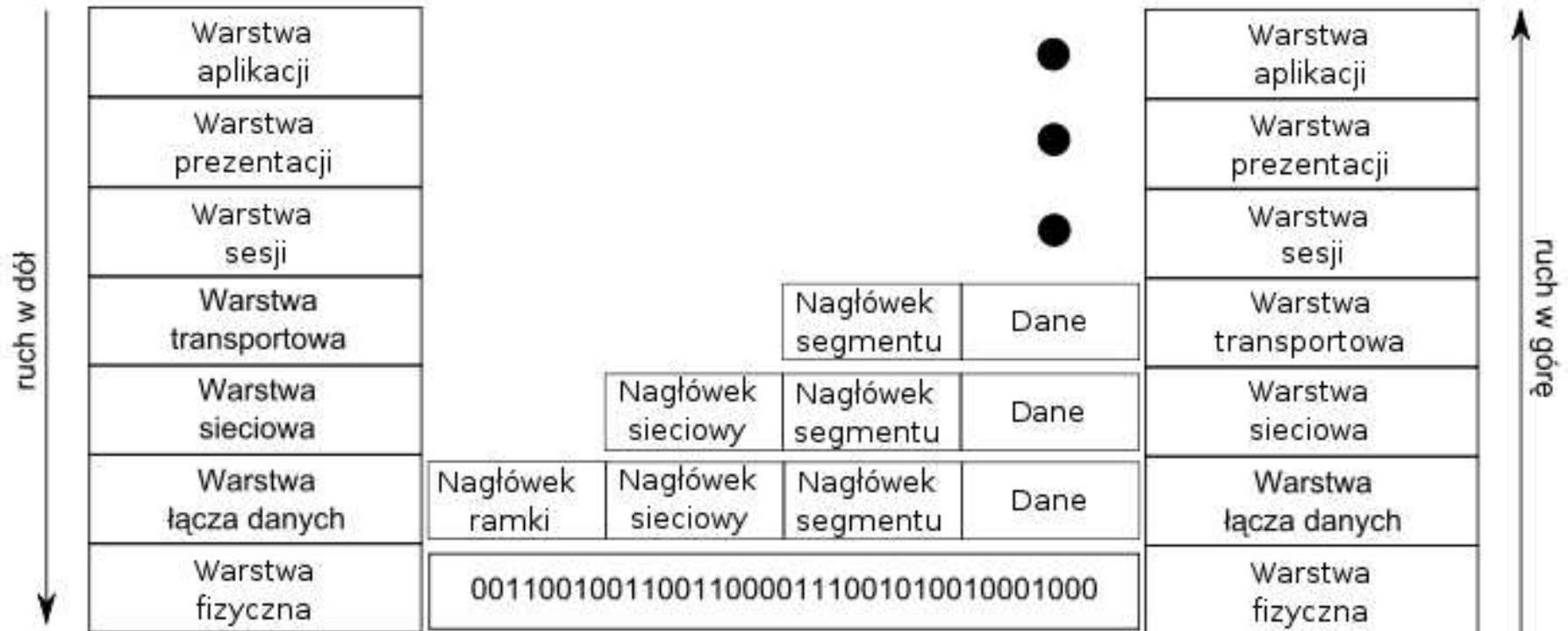
łącza danych (ramka, suma kontrolna, ack(?))

sesji (terminal, identyfikacja użytkownika)

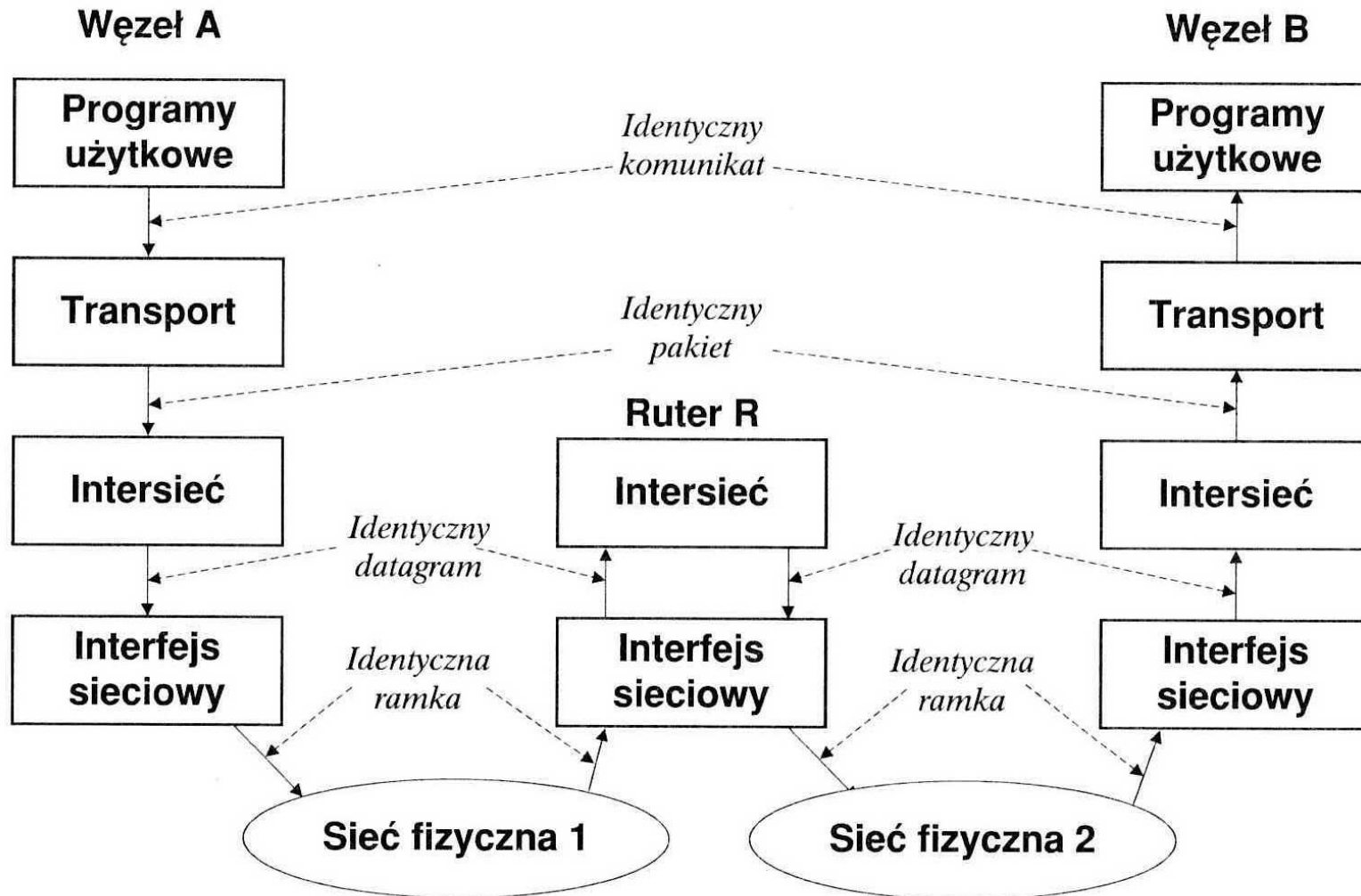
prezentacji (format danych, np. xdr, json, ASN.1)

aplikacji (mail, ftp, ...)

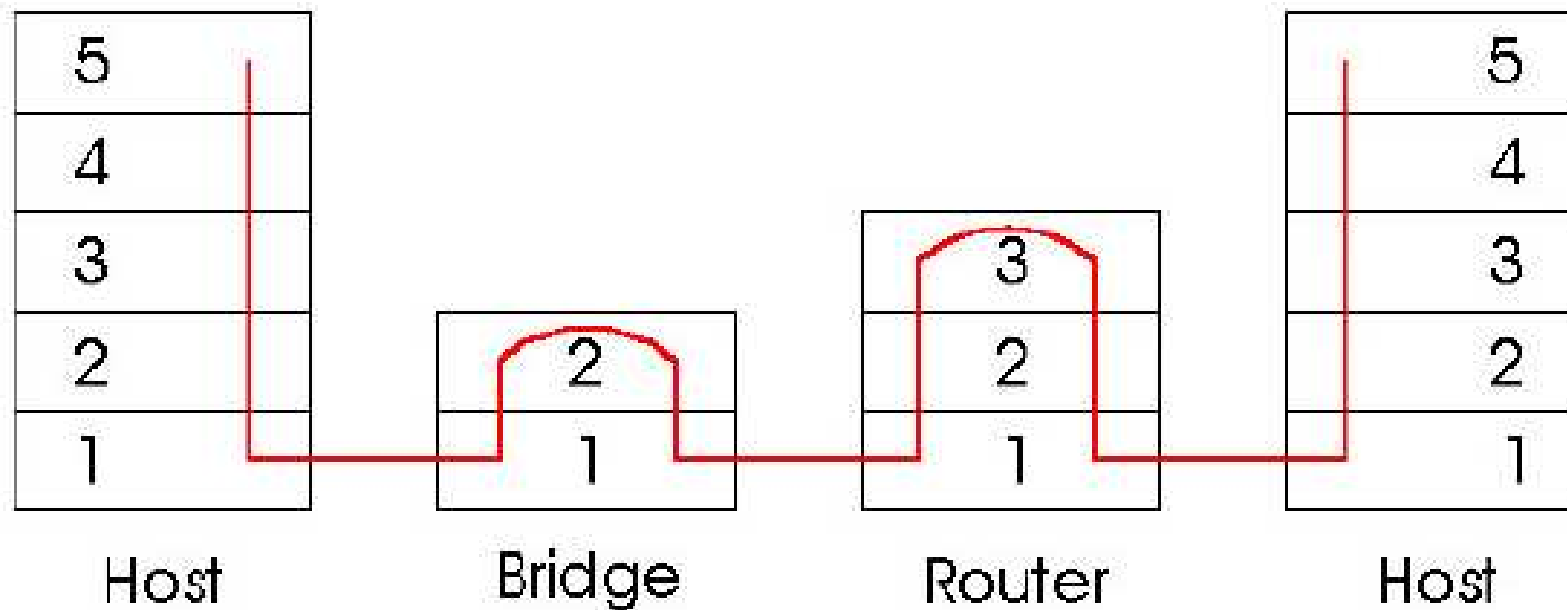
Warstwy ISO/ enkapsulacja:



Jak pakiet przechodzi przez warstwy wędrując przez intersieć...



ile sieci fizycznych jest na tym rysunku?



Protokoły niskopoziomowe

- ARP (ang. Address Resolution Protocol)
 - zamiana adresu IP na sprzętowy
 - tablica/ cache ARP, zawiera pary (adres IP, adres sprzętowy)
 - zasada działania: broadcast sprzętowy z poszukiwanym adr IP
 - przenoszony w ramce eth
- DHCP (ang. Dynamic Host Configuration Protocol)
 - przydzielanie adresu IP dla interfejsu sieciowego hosta, oraz inne sprawy: maska, default router, serwery DNS
 - w datagramie UDP
- ICMP (ang. Internet Control Message Protocol)
 - zastosowania: jest ich wiele; powszechnie znane: echo/ polecenie ping
 - komunikaty icmp posiadają pole TYP, KOD i inne w zależności od TYP:
 - TYP=8 TYP=0 echo pytanie/odp,
 - TYP=11 ttl spadł do 0 (ma zastosowanie w traceroute !)
 - TYP=4 tłumienie nadawcy, wysyłany przez przeciążony router
 - TYP=13 TYP=14 pytanie o czas i odp
 - TYP=17 prośba o maskę
 - w pakiecie IP

Protokoły warstwy aplikacji

- FTP, TELNET, DNS, HTTP, SMTP, POP3/IMAP, ...
- model klient/serwer; usługa, klient, serwer (świadczy usługę), klient rozmawia z serwerem przy pomocy powyższych prot
- "nr portu", wprowadzony w TCP i UDP, serwer oczekuje na klientów na danym nr portu, np. FTP - 21, patrz /etc/services, wiele serwerów na jednej maszynie
- FTP - przesyłanie plików
TELNET, SSH - terminal do zdalnej maszyny
DNS - zamiana adresów domenowych na IP i odwrotnie
HTTP - strony www, rozmowa między przeglądarką a serwerem www
SMTP, POP3/IMAP - prot mailowe, wysyłanie/odbieranie maili ze skrzynki
...

Protokół IP

Nagłówek pakietu IP:

0 - 3	4 - 7	8 - 15	16 - 18	19 - 23	24 - 31
Wersja	IHL	Typ usługi	Długość całkowita		
Identyfikator			Flagi	Przesunięcie fragmentu	
Czas życia	Protokół		Suma kontrolna nagłówka		
Adres źródłowy					
Adres docelowy					
Opcje			Dopełnienie		
Dane					

- pakiet IP zawiera adresy IP węzłów: źródłowego (src) i docelowego (dst)
- "czas życia", TTL, Time To Live, ile raz może przeskoczyć przez router
- IHL - długość nagłówka pakietu IP (w słowach 32bit)
- fragmentacja, gdy długość pakietu $>$ MTU sieci fizycznej (max długość ramki)

- opcje IP - rozszerzenie IP ? różne ciekawe zastosowania...
 - opcje wydłużają nagłówek pakietu ip
 - opcje mają "klasę" i "nr"
 - opcje zajmują ≥ 1 bajt (w zależności od nr opcji)
 - klasa=0 - kontrola pakietów i sieci
 - klasa=2 - pomiary
 - nr opcji=3 i 9 - swobodne/rygorystyczne trasowanie wg nadawcy (nadawca podaje listę adr ip)
 - nr opcji=7 - zapisuj trasę (w pkg jest miejsce na pewną liczbę adr ip)
 - nr opcji=4 - zapisywanie czasów wzdłuż ścieżki

Protokół UDP

- nagłówek datagramu UDP zawiera nr portu źródłowy i docelowy
- datagram UDP jest transportowany w pakiecie IP
- broadcasting , "jeden do wielu", jedyński jako nr hosta
- multicasting, "jeden do wielu", przeskakiwanie przez routery (TTL), adres docelowy ip klasy D, grupy multicastowe
- do czego służą nr portów ? (na przykładzie udp)
2 procesy na 1 maszynie oczekują na datagram udp, jak je odróżnić??
odp: przy pomocy (różnych) nr portów...
- (prot, adr IP src, port src, adr IP dst, port dst)
jeśli prot=UDP to to jest "powiązanie"
jeśli prot=TCP to to jest "połączenie"
te liczby są zawarte w nagł pakietów i pozwalają odróżnić połączenia/powiązania!

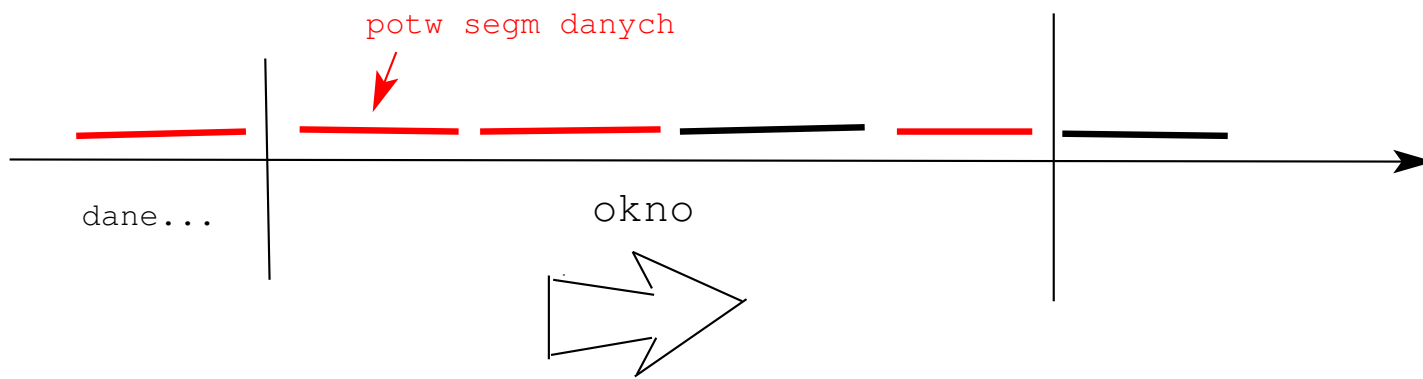
Protokół TCP

Dygresja na temat łączy (nie)nazwanych unixa:

- łączy służy do komunikacji między dwoma procesami na jednej maszynie
- łączy to rozwiązanie "problemu producenta i konsumenta" (kontrola przepływu)
- prawa rządzące łączy ...
patrz <http://mhanckow.students.wmi.amu.edu.pl/sop322B.htm>
- połączenie TCP zachowuje się dokładnie tak jak łączy !!!

Cechy połączenia TCP:

- połączenia TCP są pewne (dane się nie gubią - w przeciwieństwie do UDP ...)
- podobnie jak w UDP, używa się nr portów;
serwer oczekuje na klientów na danym nr portu
- połączenia TCP są dwukierunkowe
- kończenie / zrywanie połączenia (fun. sys. close(desk) vs problemy sieciowe)
- implementacja połączenia TCP:
segmenty TCP, wysyłanie z potwierdzaniem,
przesuwające się okno z segmentami (ang. sliding window),
kontrola przepływu za pomocą zmiany rozmiaru tego okna



.

Gniazda BSD

- patrz <http://mhanckow.students.wmi.amu.edu.pl/sop322D.htm>
pokazać "dziedzine internetową/ gniazdka strumieniow"
pokazac dziedzine internetową/ gniazdka datagramowe
- gniazda BSD w językach skryptowych/ dynamicznych (język Tcl)
pokazać zachowanie połączenia TCP ...
- rola nr portu w połączeniach TCP
(zwł. po stronie serwera, gniazdko passywne i gniazdka aktywne)
rola nr portu w datagramach UDP

```
## serwer (w j. Tcl)
#
socket -server obsluga 10000

proc obsluga {s args} {
    puts "obsluga: podlaczyl sie $s"
    fileevent $s readable "obslugaKli $s"
}
proc obslugaKli s {
    if {[eof $s]} { puts "obslugaKli: close $s"; close $s; return }
    set linia [gets $s]
    puts "obslugaKli: linia od $s, $linia"
    puts $s "odp na $linia"; flush $s
}

## klient
#
set s [socket localhost 10000]

puts $s "A ku ku !!!"; flush $s
gets $s
    #% odp na A ku ku !!!

close $s
```

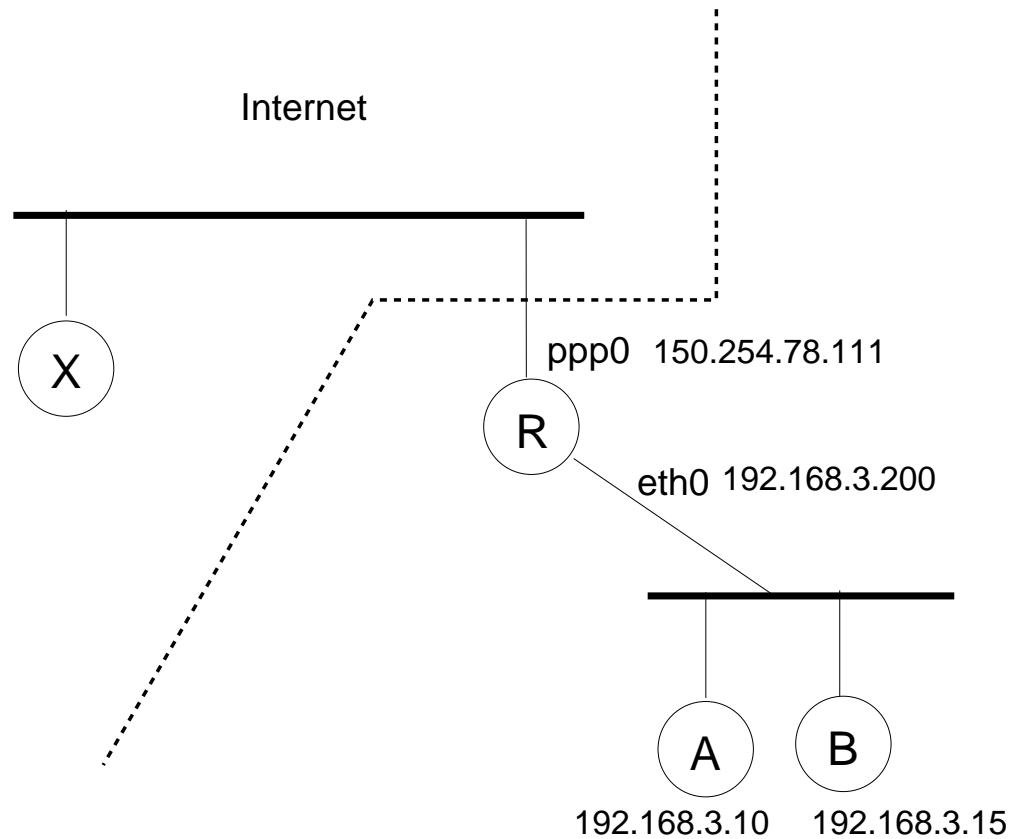

Więcej o routerach ...

- NAT (ang. Network Address Translation)
zamiana adresów IP i/lub nr portów pakietów przechodzących przez router
gdy wraca "odpowieź" wykonuje się na pakiecie operacje odwrotną !
- SNAT, MASQ, modyfikowanie adresów IP i nr portów **źródłowych**
umożliwia dostęp do internetu z sieci lokalnej, z adresami prywatnymi!
MASQ jak SNAT, ale gdy router ma zmienny adres IP
- DNAT, modyfikowanie adresów IP i nr portów **docelowych**
umożliwia udostępnianie w internecie serwerów, pracujących na maszynach w sieci lokalnej z adresami prywatnymi
(o ile router ma publiczny adres IP ...)
- zapora sieciowa, czyli odrzucanie niektórych pakietów IP
- linux: wszystko (NAT i zapory) robimy poleceniem *iptables* !

```
iptables -A INPUT -p tcp --dport 8080 -j ACCEPT; # wpuszczamy tcp, dport=8080
iptables -A INPUT -j DROP; # odrzucamy wszystkie pakiety
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o ppp0 -j SNAT --to 150.254.78.111
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 -j DNAT \
  --to 192.168.3.10:8015
```

Jak działa SNAT/ DNAT ??

```
iptables -t nat -A POSTROUTING -o ppp0 -j SNAT --to 150.254.78.111
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 -j DNAT \
  --to 192.168.3.10:8015
```



Prot. nad warstwą transportową: FTP

- służy do kopiowania plików File Transfer Protocol
- model klient/serwer
- zasada działania: używa 2 połączeń TCP;
 1. połączenie dla komend,
przez to połączenie klient wysyła rozkazy do serwera i otrzymuje odpowiedzi
(patrz wydruk z sockspy, ftp01.txt)
 2. połączenie dla danych
służy do kopiowania plików; tworzone gdy to jest potrzebne wiele razy...
- dwa tryby tworzenia połączenia dla danych:
 - aktywne FTP:**
klient tworzy połączenie dla komend do serwera czekającego na porcie 21,
port klienta w to N; klient wysyła komendę "PORT N+1" i czeka na połączenie od
serwera na porcie N+1,
serwer tworzy połączenie dla danych do klienta czekającego na porcie N+1
 - pasywne FTP:**
klient tworzy połączenie dla komend do serwera czekającego na porcie 21,
klient wysyła komendę "PASV" a serwer odpowiada z nr portu M, na którym będzie
oczekiwał na połączenie dla danych (i robi to),
klient tworzy połączenie dla danych do serwera czekającego na porcie M

wydruk z sockspy - prot FTP

14:40:45

220 FTP server ready.

14:40:45

USER anonymous

14:40:45

230 Anonymous user logged in

14:40:45

PWD

14:40:45

257 "/" is your current location

14:40:45

PASV

14:40:45

227 Entering Passive Mode (127,0,0,1,226,167)

14:40:45

TYPE A

14:40:45

200 TYPE is now ASCII

14:40:45

LIST -la /.

14:40:45

150 Accepted data connection

...

Prot. nad warstwą transportową: HTTP

- przeglądarka ściąga strony z serwera WWW za pomocą prot. HTTP
- ma także inne zastosowania, np. tzw REST API...
- model klient/serwer, połączenie TCP (wielokrotnie tworzone)
- opisane w dokumencie RFC 2616 (HTTP/1.1), <http://tools.ietf.org/html/rfc2616>

- żądanie HTTP:

```
GET /index.html HTTP/1.0 [CRLF]
Accept: image/gif, image/jpeg [CRLF]
User-Agent: Mozilla/4.0 [CRLF]
Host: www.cs.huji.ac.il:80 [CRLF]
Connection: Keep-Alive [CRLF]
[CRLF]
```

.....

pierwsza linia zawiera: metode, url, wersje protokołu
następne linie to tzw "nagłówki"
potem "pusta linia" i ew. dane (dane metody POST)

- metody w żądaniu HTTP:

GET - pobieranie zasobu na który wskazuje URL w żądaniu HTTP
(nie powinno niczego modyfikować na serwerze!!)

POST - przyjęcie danych od klienta (np. z formularza HTML)

HEAD - jak GET, ale nie pobiera danych zasobu (same nagłówki)

PUT - podobne do POST, ale URL oznacza co innego ("obiekt", a nie "metode")

DELETE - usuwanie zasobu

- ważne nagłówki w żądaniu HTTP:
 - "Host: ???" - umożliwia tworzenie "wirtualnych hostów" (1 adres IP, wiele adresów domenowych), obowiązkowy w HTTP/1.1
 - "Connection: Keep-Alive" - jedno połączenie używane do wielu zapytań HTTP
 - "Authorization: Basic cXFxOnFxcQ==" - uwierzytelnianie klienta typu "basic" (jeśli kod odp http = 401 to przeglądarka otwiera okno user/passwd)
 - "Cookie: ???" - ciasteczka wysyłane przez przeglądarkę do serwera

- odpowiedź HTTP:

```
HTTP/1.0 200 OK [CRLF]
Date: Fri, 31 Dec 1999 23:59:59 GMT [CRLF]
Content-Type: text/html [CRLF]
Content-Length: 1354 [CRLF]
[CRLF]
<html> [CRLF]
<body> [CRLF]
<h1>Hello World</h1> [CRLF]
.....
```

pierwsza linia zawiera: wersję prot, kod odpowiedzi i jej słowny opis
następne linie zawierają nagłówki odpowiedzi HTTP
potem "pusta linia" i dane odpowiedzi, np. HTML lub coś innego...

- kody w odpowiedzi HTTP:
 - 200 OK - prawidłowa odpowiedź
 - 302 Found - tzw "redirekt", przeglądarka powinna przełączyć się na inny URL
 - 401 Unauthorized - strona wymaga, aby użytkownik się uwierzytelnił
 - 404 Not Found - serwer nie znalazł zasobu
- ważne nagłówki w odpowiedzi HTTP:
 - "Content-Type: text/html" - typ odpowiedzi jako mime
 - "Content-Type: text/html; charset=utf-8"
 - "Content-Length: 1354" - długość odpowiedzi HTTP
 - "Set-Cookie" - serwer zmusza przeglądarkę żeby utworzyła ciasteczko

- przekazywanie dodatkowych parametrów do żądania HTTP:

1. zmienne w url-u, met. GET

```
http://localhost:8001/np02/plik1.tcl?x=1234&y=4321
```

```
# kodowanie znaków przy pomocy %kod, tzw "x-url-encoding"
```

2. "dane POST" za pusta linia, met. POST

żądania http typu POST (lub GET) są tworzone przez formularz w pliku HTML, w przeglądarce, guzik submit lub przez biblioteki http, pokazać przykład http02.tcl ...
kwestia kodowania znaków (utf-8 ? iso8859-2 ?)

```
POST /np02/plik1.tcl HTTP/1.0[CRLF]
```

```
Accept: /*/[CRLF]
```

```
Host: localhost:8001[CRLF]
```

```
User-Agent: Tcl http client package 2.5.2[CRLF]
```

```
Content-Type: application/x-www-form-urlencoded[CRLF]
```

```
Content-Length: 15[CRLF]
```

```
[CRLF]
```

```
x=12345&y=54321[CRLF]
```

- ciasteczka czyli Cookies, session_id ...

- w żądaniu HTTP:

Cookie: nazwa1=wartość1; nazwa2=wartość2; ...

- w odpowiedzi HTTP:

Set-Cookie: nazwa=wartość; expires=DATA; path=ŚCIEŻKA; secure
expires - czas życia ciasteczka u klienta
path - jaki url-i na hostie to ciasteczko dotyczy
secure - tylko dla HTTPS

- jak działają ciasteczka?

- + tworzone przez odpowiedź http serwera www (Set-Cookie:)

- + dopóki się nie przeterminują, wysyłane przez przeglądarkę do serwera www (Cookie:) w każdym żądaniu http

- + fizycznie ciasteczka są przechowywanymi w plikach u klienta (przez przeglądarkę)

- **zastosowanie ciasteczek:** umożliwiają przechowywanie "zmiennych sesyjnych" na serwerze www;

- + co to są "zmiennne sesyjne" ? zmienne związane z sesją użytkownika

- + co to jest "sesja użytkownika" ? ciąg kliknięć (w przeglądarce), przez danego użytkownika, które nie są zbyt oddzielone w czasie...

- + dlaczego ciasteczka są niezbędne? bo serwer http jest "bezstanowy"

- + identyfikator sesji jest przechowywany w ciasteczku u klienta ...

- obsługa sesji użytkownika na przykładzie frameworka webowego "OpenACS":
ciasteczko z identyfikatorem sesji: *ad_session_id*
parametry obsługi sesji:
SessionRenew = 5min (czas po którym "odnawia się" ciasteczko sesji)
SessionTimeout = 20min (bez odnawiania sesja znika po tym czasie)
SessionLifetime = 7dni (sesja znika)
pokazać przykład *oacs_session.tcl* ...
- *ad_session_id*/ pytanie 1: jak jest minimalny czas między kliknięciami, po którym sesja może zniknąć???
- *ad_session_id*/ pytanie 2: dlaczego *SessionRenew* >0 ? wskazówka: strona www z 1000 obrazków ...

Gniazdko "bezpieczne" - SSL/TLS

- SSL = Secure Socket Layer, TLS = Transport Layer Security, OpenSSL = implementacja SSL/TLS (biblioteka programistyczna i polecenie openssl)
- Skrócony opis pojęć kryptograficznych:
klucz symetr, asymetr (pub/pryw), fun haszująca, podpis elektr, certyfikat SSL (zawiera klucz pub, podpisany przez CA)
- Co zapewnia SSL/TLS?
szyfrowanie danych, żłośliwe zmiany niemożliwe, uwierzytelnianie serwera, uwierzytelnianie serwera i klienta
- Co jest potrzebne po stronie serwera ?
certyfikat SSL serwera (z kluczem pub serwera), klucz pryw serwera
- Co jest potrzebne po stronie klienta ?
jeśli klient chce sprawdzić certyfikat SSL serwera, to musi podać certyfikat SSL CA (który podpisał elektronicznie certyfikat serwera)
podobnie w drugą stronę...
- Zasada działania ...
 1. serwer wysyła do klienta swój cert+klucz pub
 2. klient wymyśla klucz symetryczny X do szyfrowania danych, szyfruje X kluczem pub serwera i wysyła go do serwera
 3. serwer odszyfrowuje swoim kluczem pryw X; teraz oba końce mają X służący do szyfrowania danych płynących przez połączenie...

- polecenie openssl:

```
# openssl: szyfrowanie/deszyfrowanie metoda Blowfish
# "-e" encode, "-d" decode, "-a" base64, "-bf" Blowfish
echo "tekst do zaszyfrowania" | openssl enc -e -a -bf -k haslo > qqg.txt
cat qqg.txt | openssl enc -d -a -bf -k haslo
```

- bezpieczna odmiana HTTP: HTTPS
to samo co http, ale używa połączenia TCP nad SSL/TLS ...
- pakiet "tls" j. Tcl, komenda **tls::socket** zamiast **socket**, pokazać przykład...

```
## klient, socket/tls, j. Tcl
#
package re tls

set s [tls::socket localhost 10000]
# + bez uwierzytelniania serwera

set s [tls::socket -require 1 -cafile cacert.pem localhost 10000]
# + wymagamy uwierzytelnienia serwera
# "-require 1" klient chce sprawdzenia certyfikatu serwera
# "-cafile" certyfikat CA (chodzi m.in. o klucz publiczny CA w tym pliku
# dzięki któremu klient sprawdza podpis na certyfikacie serwera)

tls::handshake $s
# ustalenie klucza symetrycznego do szyfrowania danych...

puts $s "A ku ku !!!"; flush $s

close $s
```

```
## serwer, socket/tls, j. Tcl
#
package re tls

proc haslo {} {return "qwerty"}; # haslo do klucza pryw serwera

tls::socket -server obsluga -password haslo \
  -keyfile privkey_2.pem -certfile cert_2.pem \
  10000
  # "-keyfile" klucz pryw serwera
  # "-certfile" cert serwera (zawiera klucz pub serwera)

proc obsluga {s args} {
  puts "obsługa: $s sie podlaczyl"
  tls::handshake $s
  puts "obsługa: $s sie podlaczyl 2"
  fileevent $s readable "obsługaKli $s"
}

proc obsługaKli s {
  if {[eof $s]} { puts "obsługaKli: $s close"; close $s; return }
  set linia [gets $s]
  puts "obsługaKli: od $s: $linia"
  puts $s "odp na $linia"; flush $s
}
```