SIK = Sieci komputerowe

Plan tego wykładu:

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

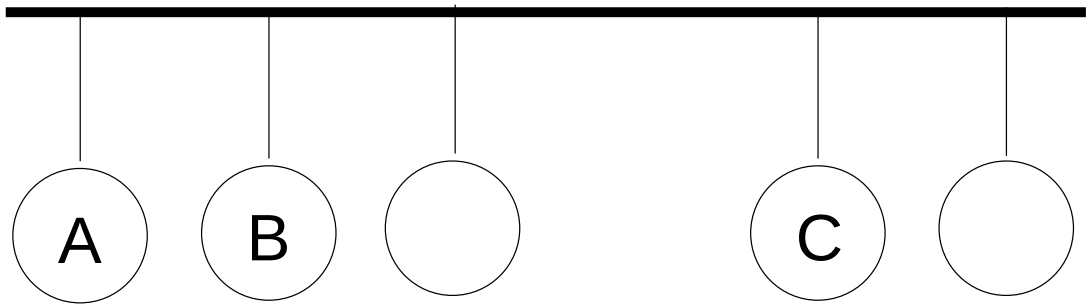
Literatura

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

Intersieć i sieci fizyczne

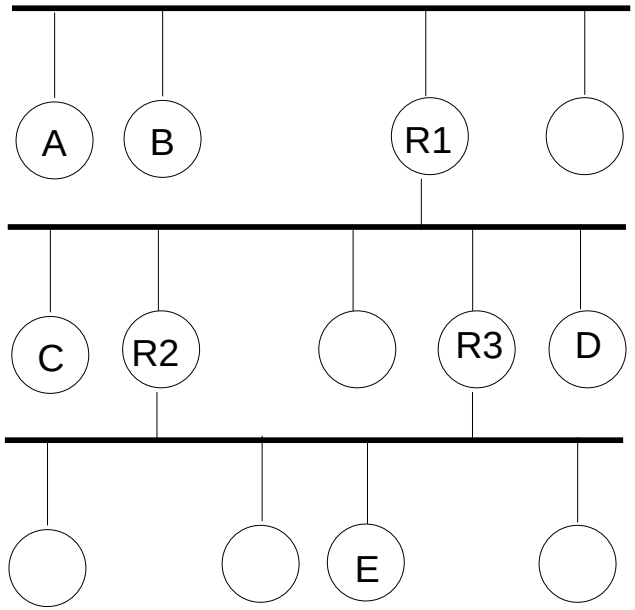
- Sieć fizyczna umożliwia wysyłanie pakietów między węzłami.
- Komputer podłączony do sieci to "węzeł".
  inne nazwy węzła: host, maszyna
- Węzeł A może wysłać pakiet do węzła B (ang. unicast)
  lub do wszystkich w tej samej sieci fizycznej (ang. broadcast)
- Pakiet = nagłówek + dane; inne nazwy: ramka, 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

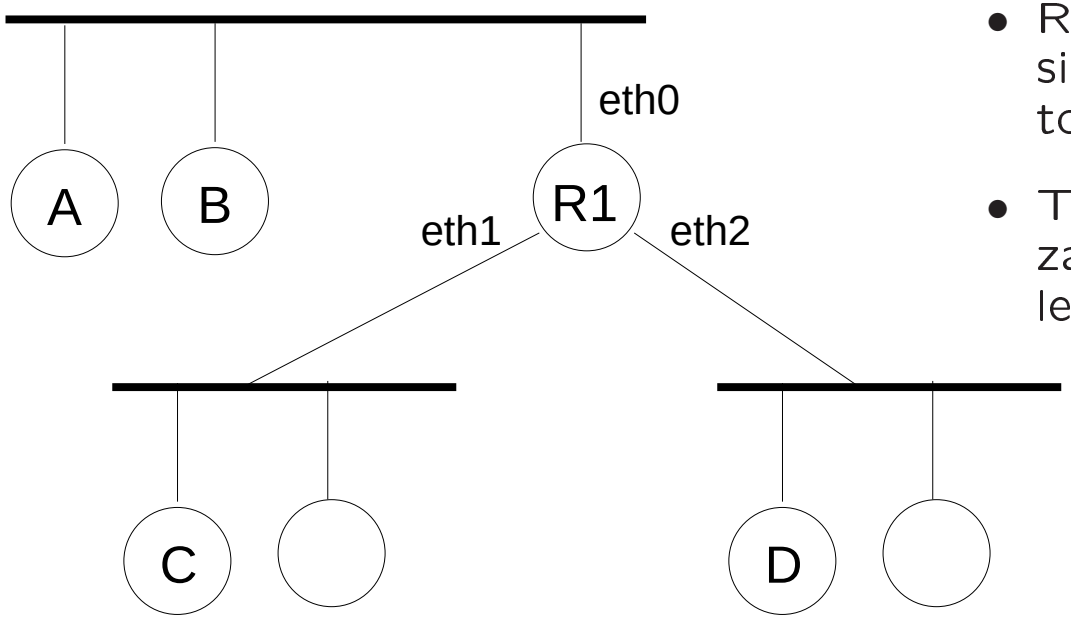
Intersieć:



- Itersieć 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

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).
- Tablica do trasowania (tablica routingowa) zawiera wpisy postaci:
  lewa sieć -> eth1, prawa sieć -> eth2

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!!

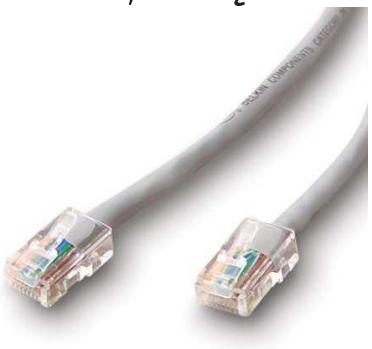
- sieć lokalna typu **Ethernet**
  - standard IEEE 802.3 (?)
  - skrętka (kabel) + switch (urządzenie sieciowe)
  - skrętka nieekranowana kategori e5, do 100metrów, 100Mbit/s = Fast Ethernet, są jeszcze szybsze!, wtyczka RJ-45, nie przejmować się przeplotem!
  - maszyny w sieci mają karty sieciowe Ethernet, >2 maszyny łączymy przy pomocy switch-a => topologia gwiazdy/drzewa
  - dawniej używano kabla koncentrycznego ...

- bezprzewodowa siec 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 telefonicze, modemy telefoniczne/akustyczne, 56Kbitów/s, bardzo długie łącze szeregowe
  - prot PPP (ang. Point to Point Protocol) przenosi pakiety IP nad łączem szeregowym (demony pppd)
  - to jest prosty przykład sieci WAN !
  - łącze szeregowe może być emulowane (bluetooth/rfcomm/IEEE 802.15, lub nad USB)

Ethernet - C.D.

RJ-45, 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 siec fizyczna ...

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, 4x8=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"
- wszystkie węzły powinny mieć inny adres IP
  (w sieci fizycznej; w intersieci - uwaga na NAT !!!)
- jeśli węzeł należy do kilku sieci to będzie miał kilka adr IP

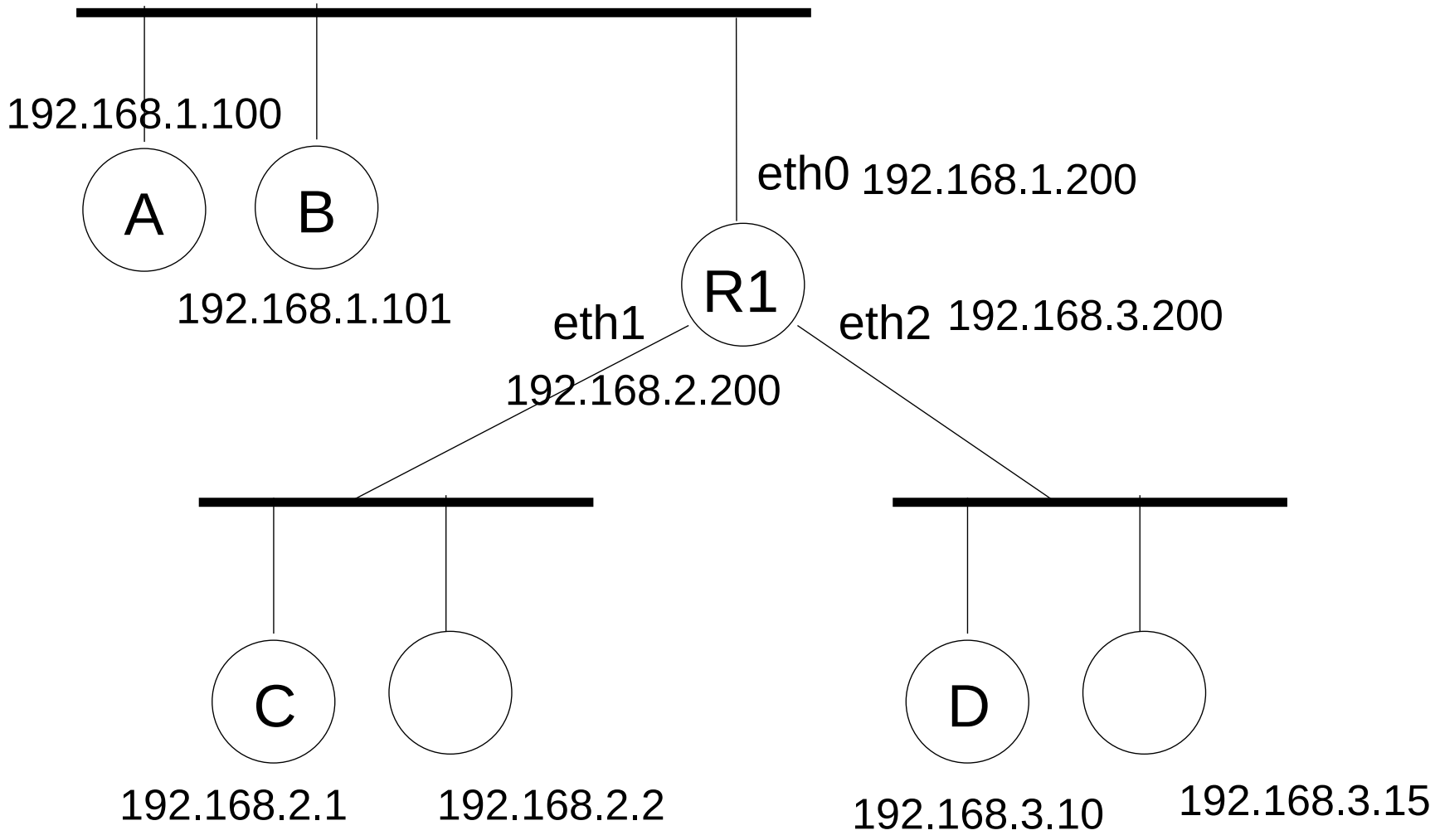
- które bity adresu IP są nr sieci, a które nr hosta?
  to zależy od "klasy adresu" i "maski podsieci" !

- 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
  - kllasa D: 1110..., multicasting

- maska podsieci
  - określa jawnie które bity adresu IP są nr sieci (jedynki 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
  - jesli jako nr hosta ustawic same jedynki: broadcast
  - 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

Intersieć z przypisanym adresami IP (klasy C):



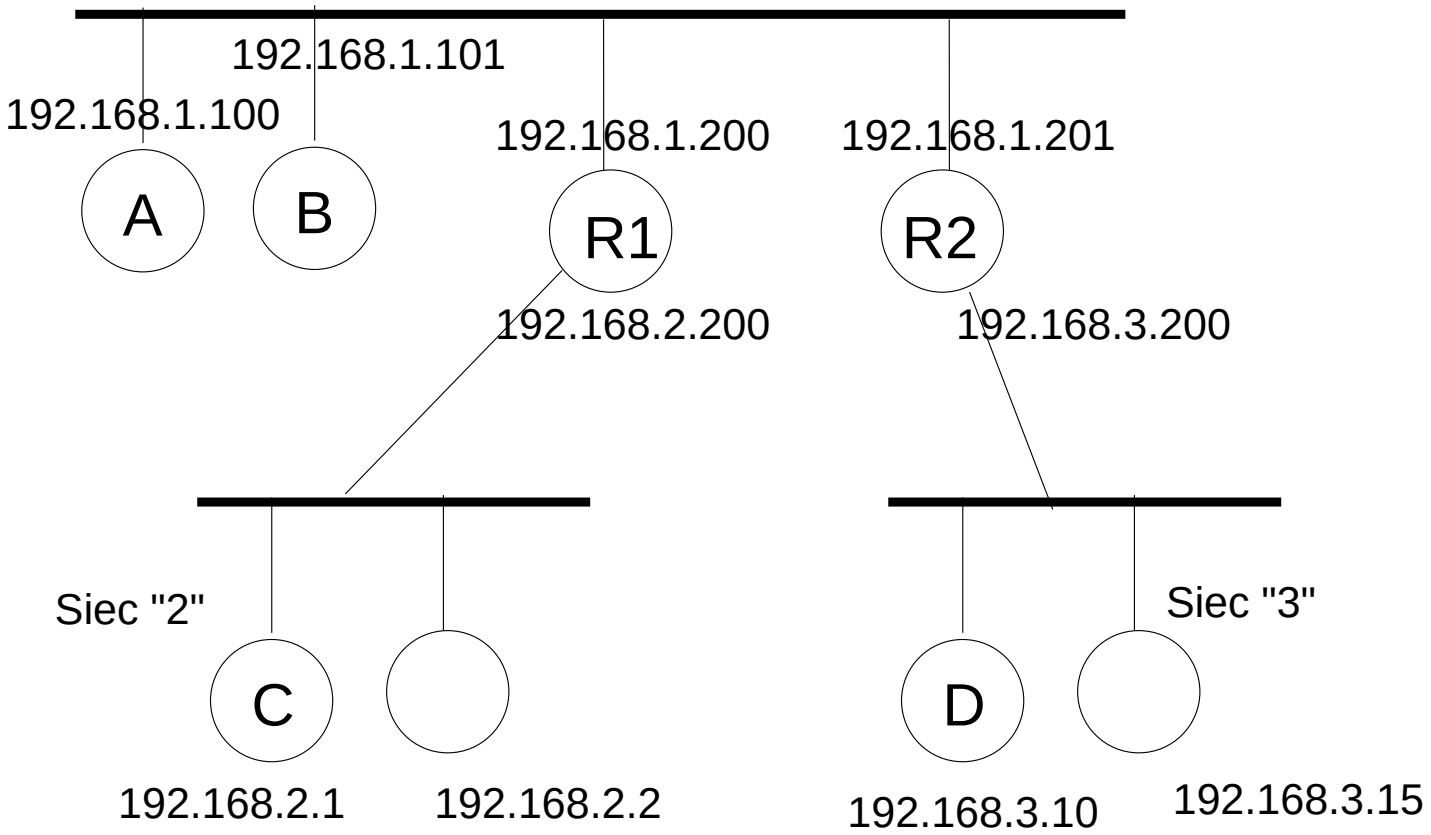
Intersieć z przypisanym adresami IP (klasy C):

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



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

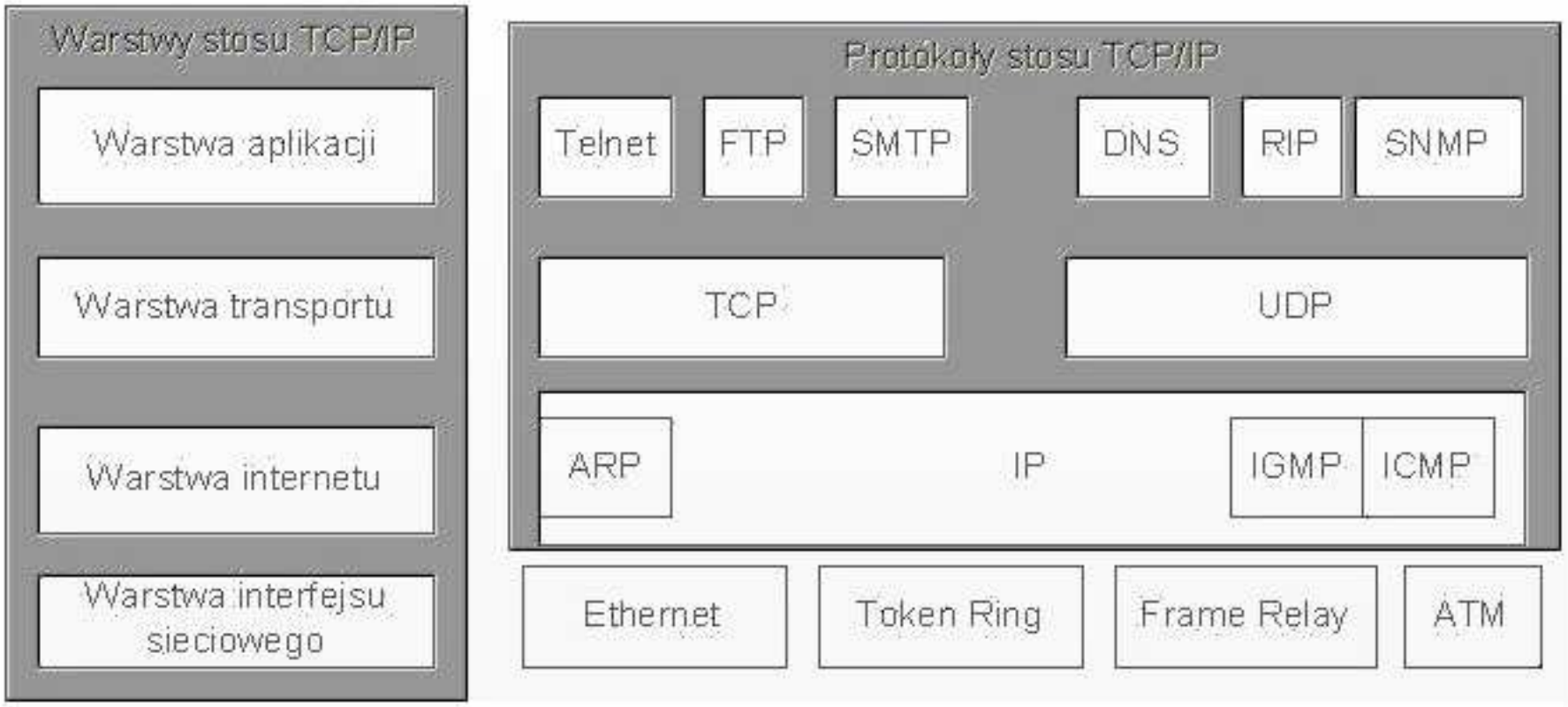
- prot TCP - warstwa transportowa
  - (wirtualne) połączenie TCP
  - można przesyłać strumien danych/bajtów
  - jest pewne

- aplikacje używają prot warstwy transportowej
  za pomocą gniazdek BSD (API, fun. systemowe)

Warstwy protokołów

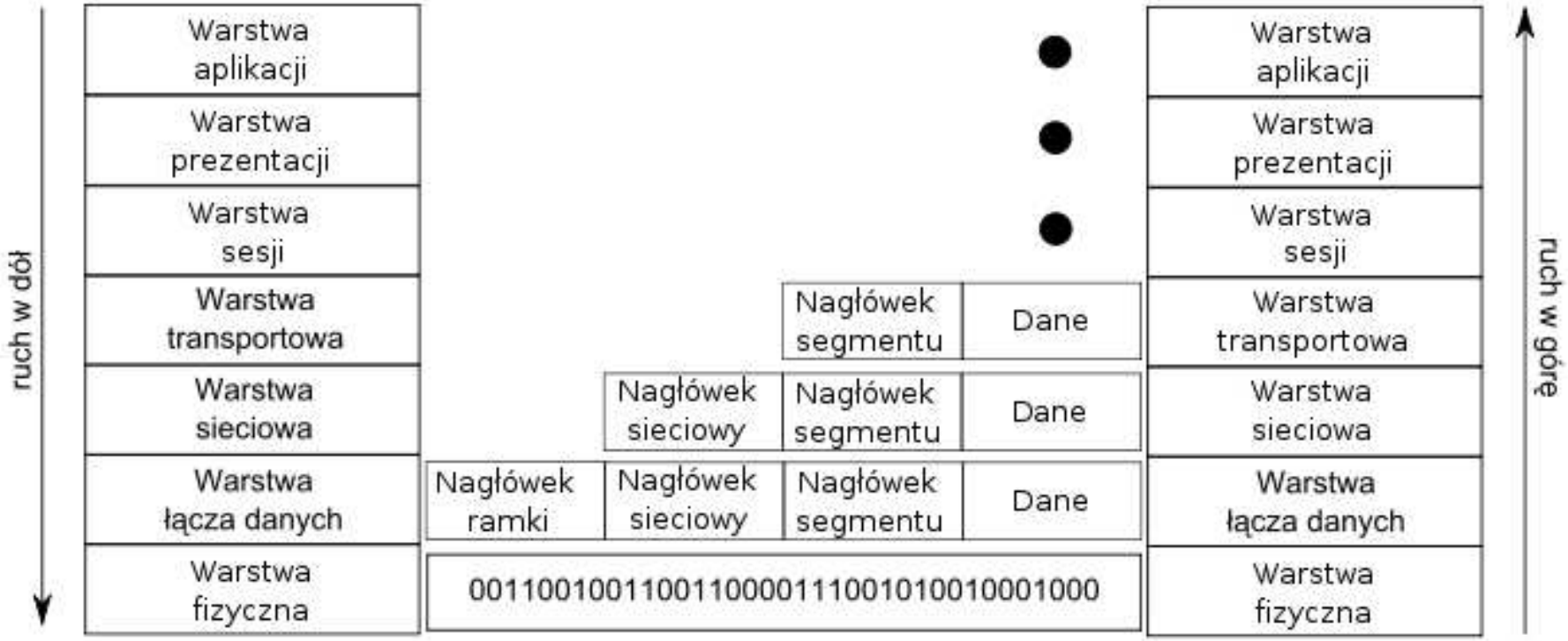
Architektura warstwowa:
wyższa warstwa używa niższej warstwy (patrz enkapsulacja)

Podział protokołów Tcp/Ip na warstwy:



Warstwy ISO vs warstwy Tcp/Ip:

Warstwy ISO/ enkapsulacja:



Protokoły niskopoziomowe

- ARP (ang. Address Resolution Protocol)
  - zamiana adresu IP na sprzętowy
  - tablica/ cache ARP, zawiera pary (adres IP, adres sprzętowy), uczy się ...
- DHCP (ang. Dynamic Host Configuration Protocol)
  - przydzielanie adresu IP dla interfejsu sieciowego hosta,
  oraz inne sprawy np. maska posieci, default router, serwery DNS
- ICMP (ang. Internet Control Message Protocol)
  - zastosowania: echo (polecenie ping), router informuje nadawce pakietu, że nie może go przekazac dalej (polecenie traceroute)

Protokoły warstwy aplikacji

- FTP, TELNET, DNS, HTTP, ...
- 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
  ...

Protokół IP

Nagłówek pakietu IP:

<table>
<tr><th>0 – 3</th><th>4 - 7</th><th>8 - 15</th><th>16 - 18</th><th>19 - 23</th><th>24 - 31</th></tr>
<tr><td>Wersja</td><td>IHL</td><td>Typ usługi</td><td colspan="3">Długość całkowita</td></tr>
<tr><td colspan="3">Identyfikator</td><td>Flagi</td><td colspan="2">Przesunięcie fragmentu</td></tr>
<tr><td>Czas życia</td><td colspan="2">Protokół</td><td colspan="3">Suma kontrolna nagłówka</td></tr>
<tr><td colspan="6">Adres źródłowy</td></tr>
<tr><td colspan="6">Adres docelowy</td></tr>
<tr><td colspan="3">Opcje</td><td colspan="3">Dopełnienie</td></tr>
<tr><td colspan="6">Dane</td></tr>
</table>

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

Protokół UDP

- nagłówek datagramu UDP zawiera nr portu źródłowy i docelowy
- datagram UDP jest transportowany w pakiecie IP
- broadcasting , "jeden do wielu", jedynki 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 serwery na 1 maszynie oczekują na datagram udp...
  2 klientów na 1 maszynie wysyła datagram udp...
  chodzi o tzw. gniazdka (ang. sockets) ...

Protokół TCP

Dygresja na temat łączy (nie)nazwanych unixa:

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

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

Więcej o routerach ...

- NAT (ang. Network Address Translation)
  zamiana adresów IP i/lub nr portów pakietów przechodzących przez router
  gdy wraca "odpowiedź" 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 1.2.3.4
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
  --to 192.168.1.100:8015
```

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 połaczenie 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 wysyla 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

Prot. nad warstwą transportową: HTTP

- przeglądarka ściąga strony z serwera WWW za pomocą prot. HTTP
- ma także inne zastosowania, może być używany przez programy (inne niż przeglądarka)
- model klient/serwer, jedno 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
nastepne 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" (jest też "digest"; pokazać przykład z sockspy...)
  "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 przeglądarke 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
     żadania 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 hoscie to ciasteczko dotyczy
      # secure - tylko dla HTTPS
    ```

  - jak działają ciasteczka?
    + tworzone przez odpowiedź http serwera www (Set-Cookie:)
    + dopóki się nie przeterminują, wysyłane przez przeglądarke 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 "zmienne sesyjne" ? zmienne związane z sesją użytkownika
    + co to jest "sesja użytkownika" ? ciąg kliknięć (w przeglądarce), przez danego użytkownika, które nie są zbytnio 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 międz kliknięciami, po którym sesja może zniknąć???

- ad_session_id/ pytanie 2: dlaczego SessionRenew >0 ? wskazówka: strona www z 1000 obrazków ...

Gniazdka "bezpieczne" - SSL/TLS

- SSL = Secure Socket Layer, TLS = Transport Layer Security,
  OpenSSL = implementacja SSL/TLS (biblioteka programistyczna i polecenie openssl)
- Skrócony opis pojęć kryptograficznych (patrz link)
- Co zapewnia SSL/TLS? (patrz link)
  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 sprawdzic certyfikat SSL serwera, to musi podac certyfikat SSL CA (który podpisal 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, i szyfruje X kluczem pub serwera i wysyła do serwera
  3. serwer odszyfrowuje swoim kluczem pryw X; teraz oba końce mają X służący do szyfrowania danych płynących przez połączenie

- przyklad: ssl_kli.tcl i ssl_ser.tcl
- 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 > qqq.txt
cat qqq.txt | openssl enc -d -a -bf -k haslo
```

- bezpieczna odmiana HTTP: HTTPS
  to samo co http, ale używa połączenia TCP nad SSL/TLS ...