Plan całości wykładu

- Wprowadzenie (2 wykłady)
- Warstwa aplikacji (2 wykłady)
- Warstwa transportu (2-3 wykłady)
- Warstwa sieci (2-3 wykłady)
- Warstwa łącza i sieci lokalne (3 wykłady)

Plan czasowy wykładu i ćwiczeń

start

zadania programistyczne
(łącznie 16 punktów)

kolokwium 1 (koniec kwietnia)

zadania programistyczne
(łącznie 16 punktów)

kolokwium 2 (koniec maja/czerwiec)

zadania programistyczne i
zaliczenie ćwiczeń

egzamin (60 punktów)

Literatura do warstwy aplikacji

Rozdział 2, *Computer Networking: A Top-Down Approach Featuring the Internet*, wydanie 2 lub 3, J. Kurose, K. Ross, Addison-Wesley, 2004

Rozdział y 4,6, *Programowanie zastosowań sieciowych w systemie Unix*, W. R. Stevens, WNT, 1995

Rozdziały 19, 20, 22, 24, 25, *Sieci komputerowe TCP/IP*, D.E. Comer, WNT, 1997

Warstwa aplikacji

Cele:

- koncepcyjne i implementacyjne zagadnienia protokołów w. aplikacji
  - modele usług warstwy transportu
  - model klient-serwer
  - model partnerski (peer-to-peer)
- poznawanie popularnych protokołów warstwy aplikacji
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- programowanie aplikacji sieciowych
  - API gniazd

Mapa wykładu

- 2.1 Zasady budowy protokołów w. aplikacji
- 2.2 WWW i HTTP
- 2.3 DNS
- 2.4 Programowanie przy użyciu gniazd TCP
- 2.5 Programowanie przy użyciu gniazd UDP
- 2.6 Poczta elektroniczna
  - SMTP, POP3, IMAP
- 2.7 FTP
- 2.8 Dystrybucja zawartości
  - Schowki Internetowe
  - Sieci dystrybucji zawartości
- 2.9 Dzielenie plików P2P

Aplikacje sieciowe: trochę słownictwa

Proces: program działający na hoście.

- na jednym hoście, dwa procesy komunikują się przez komunikację międzyprocesową (zdefiniowaną przez System Operacyjny).
- procesy działające na różnych hostach porozumiewają się protokołem warstwy aplikacji

agent (user agent): komunikuje się z użytkownikiem i siecią.

- implementuje interfejs użytkownika i protokół warstwy aplikacji
  - WWW: przeglądarka
  - E-mail: program pocztowy
  - streaming audio/video: odtwarzacz multimediów

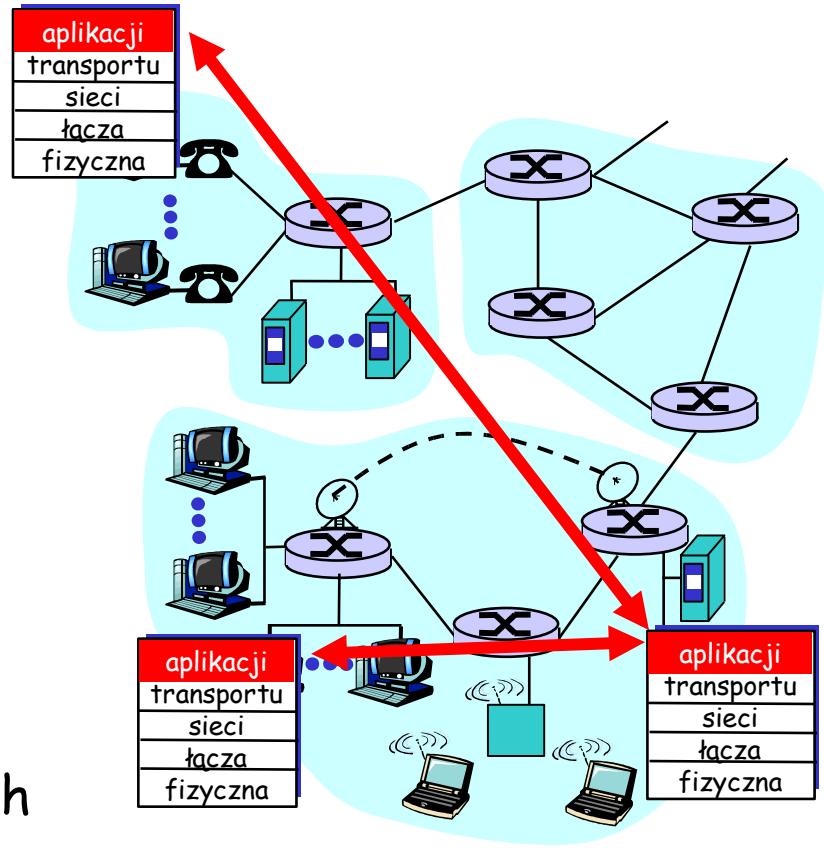
Aplikacje i protokoły warstwy aplikacji

Aplikacje: komunikujące, rozproszone procesy

- n.p., e-mail, WWW, dzielenie plików P2P, komunikatory
- działają na systemach końcowych (hostach)
- wymieniają komunikaty

Protokoły warstwy aplikacji

- "kawałek" aplikacji
- definiują komunikaty i akcje aplikacji
- używają usług komunikacyjnych niższej warstwy (TCP, UDP)



Protokół w. aplikacji określa...

- Rodzaje komunikatów, n.p., komunikaty żądania i odpowiedzi
- Składnię komunikatów: jakie pola w komunikatach i jak są oddzielane
- Znaczenie informacji w polach komunikatu
- Zasady określające kiedy i jak procesy wymieniają komunikaty

Protokoły publiczne:

- definiowane w dokumentach Request For Comments (RFC)
- pozwala na współpracę różnych systemów
- n.p., HTTP, SMTP

Protokoły prywatne:

- n.p., KaZaA

Model klient/serwer

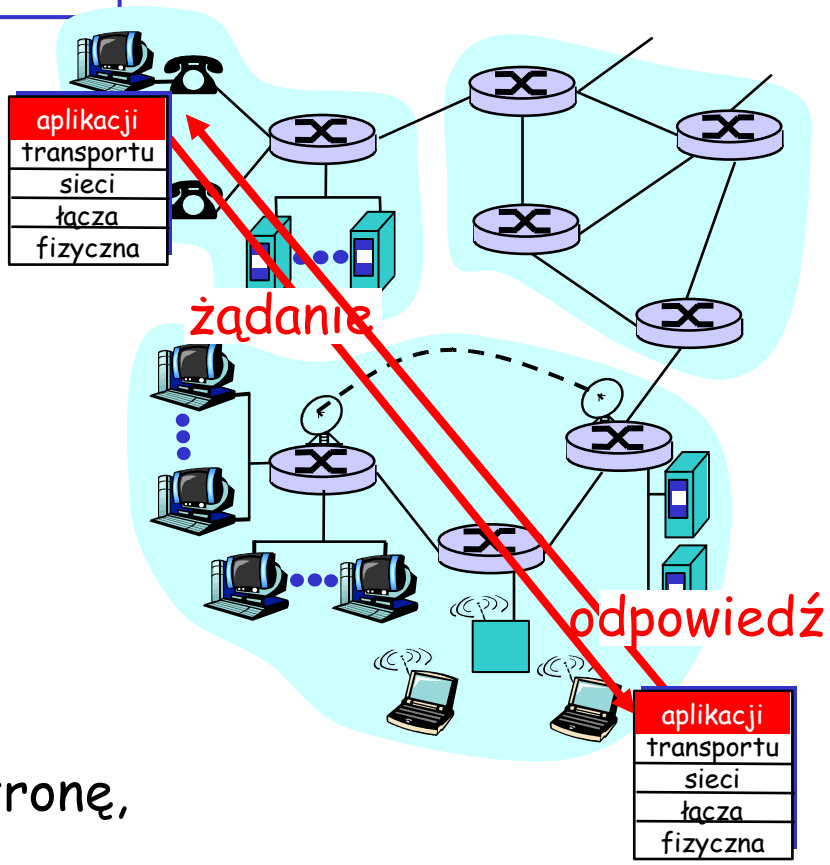
Typowa aplikacja sieciowa ma dwie części: *klienta* i *serwera*

Klient:

- rozpoczyna komunikację z serwerem ("mówi pierwszy")
- zwykle prosi o usługę serwera,
- WWW: klient implementowany przez przeglądarkę; e-mail: przez program pocztowy

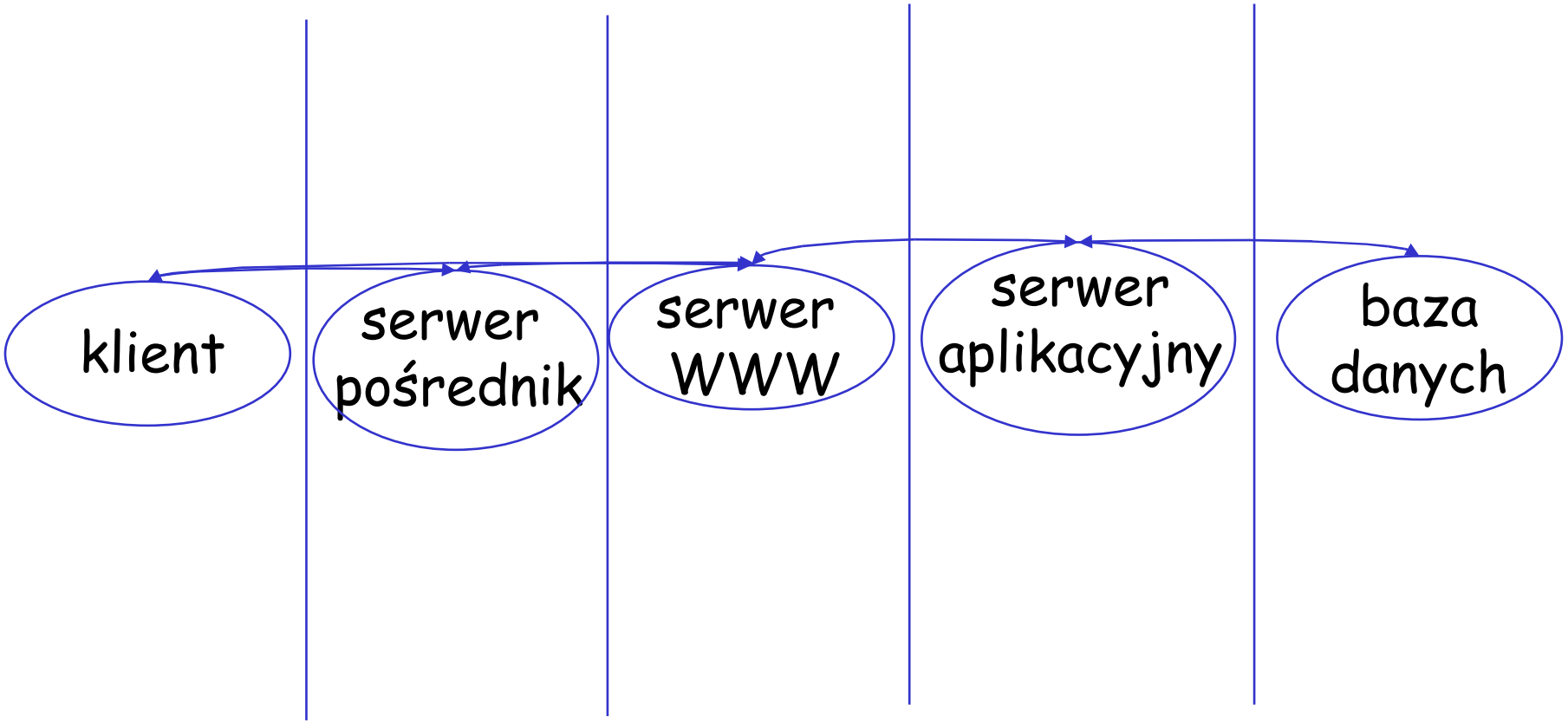
Serwer:

- udostępnia żądaną usługę klientowi
- n.p., serwer WWW wysyła żądaną stronę, serwer poczty dostarcza pocztę



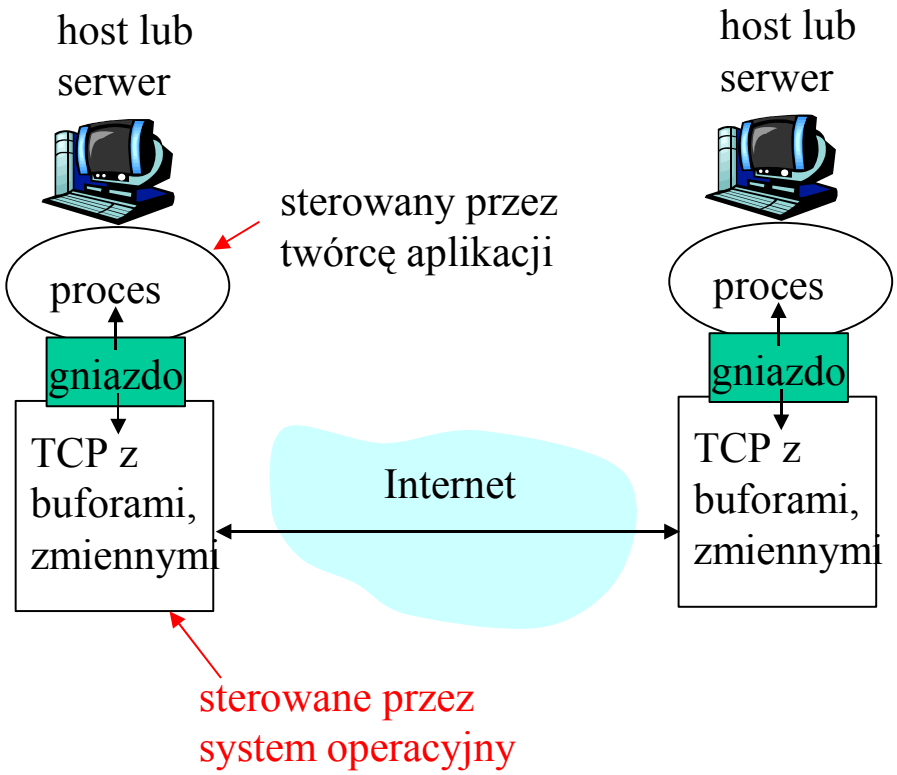
Rozwinięcia modelu klient/serwer

- Dodatkowe etapy pośrednie: płaszczyzny (ang. *multi-tier architecture*)



Procesy komunikujące przez sieć

- proces wysyła/odbiera komunikaty do/z swojego *gniazda*
- gniazdo można porównać do skrzynki pocztowej
  - proces wrzuca wiadomość do skrzynki
  - proces zakłada, że warstwa transportu dostarczy komunikat do odbiorcy



- API: (1) wybór protokołu transportowego;
  (2) możliwość zmiany niektórych parametrów

(o tym więcej później)

Adresowanie procesów:

- Żeby proces mógł odbierać komunikaty, musi mieć identyfikator
- Każdy host ma unikatowy 32-bitowy adres IP
- Pytanie: czy adres IP hosta na którym działa proces wystarczy dla identyfikacji procesu?
- Odpowiedź: Nie, wiele procesów może działać na jednym hoście
- Identyfikator zawiera adres IP oraz numer portu związany z procesem na hoście.
- Przykładowe numery portów:
  - serwer HTTP: 80
  - serwer poczty: 25
- Wrócimy do tego tematu później

Jakiej usługi transportowej potrzebuje aplikacja?

Straty

- niektóre aplikacje (n.p., audio) tolerują pewną ilość strat
- inne aplikacje (n.p., transfer plików, telnet) wymagają 100% niezawodności

Opóźnienie

- niektóre aplikacje (n.p., telefonia Internetowa, gry interaktywne) wymagają niskich opóźnień

Przepustowość

- niektóre aplikacje (n.p., audio/wideo) wymagają minimalnej przepustowości
- inne aplikacje ("elastyczne") mogą wykorzystać tyle przepustowości, ile otrzymają

Wymagania aplikacji dotyczące transportu

Aplikacja	Straty	Przepustowość	Opóźnienie
transfer plików	bez strat	elastyczna	nie
poczta elektroniczna	bez strat	elastyczna	nie
WWW	bez strat	elastyczne	nie
audio/wideo w czasie rzeczywistym	toleruje straty	audio: 5kbps-1Mbps wideo:10kbps-5Mbps	tak, setki ms
odtwarzane audio/wideo	toleruje straty	jak wyżej	tak, kilka s
gry interaktywne	toleruje straty	powyżej kilku kbps	tak, setki ms
komunikatory	bez strat	elastyczna	tak i nie

Usługi transportowe Internetu

usługa TCP:

- *połączeniowa:* wymaga inicjalizacji połączenia pomiędzy procesami
- *niezawodna komunikacja* pomiędzy procesami
- *kontrola przepływu:* nadawca nie przeciąży odbiorcy
- *kontrola przeciążenia:* nadawca zwolni, jeśli sieć zostanie przeciążona
- *nie udostępnia:* informacji o czasie, gwarancji minimalnej przepustowości

usługa UDP:

- zawodna komunikacja pomiędzy procesami
- nie udostępnia: tworzenia połączenia, niezawodności, kontroli przepływu, kontroli przeciążenia, informacji o czasie, ani gwarancji przepustowości

Pytanie: Po co? Czemu istnieje usługa UDP?

Aplikacje Internetowe: protokoły warstw aplikacji i transportu

Aplikacja	Protokół warstwy aplikacji	Wykorzystywany protokół transportowy
e-mail	SMTP [RFC 2821]	TCP
zdalny terminal	Telnet [RFC 854]	TCP
WWW	HTTP [RFC 2616]	TCP
transfer plików	FTP [RFC 959]	TCP
media strumieniowe	prywatne (n.p. RealNetworks)	TCP lub UDP
Telefonia Internetowa	prywatne (n.p., Dialpad)	zwykle UDP

Mapa wykładu

- 2.1 Zasady budowy protokołów w. aplikacji
- 2.2 WWW i HTTP
- 2.3 DNS
- 2.4 Programowanie przy użyciu gniazd TCP
- 2.5 Programowanie przy użyciu gniazd UDP
- 2.6 Poczta elektroniczna
  - SMTP, POP3, IMAP
- 2.7 FTP
- 2.8 Dystrybucja zawartości
  - Schowki Internetowe
  - Sieci dystrybucji zawartości
- 2.9 Dzielenie plików P2P

WWW i HTTP

Najpierw terminologia

- Strona WWW składa się z obiektów
- Obiekt może być plikiem HTML, obrazem JPEG, apletem Java, plikiem audio,...
- Strona WWW składa się z bazowego pliku HTML który zawiera szereg odnośników do obiektów
- Każdy obiekt posiada adres URL (Uniform Resource Locator)
- Przykładowy URL:

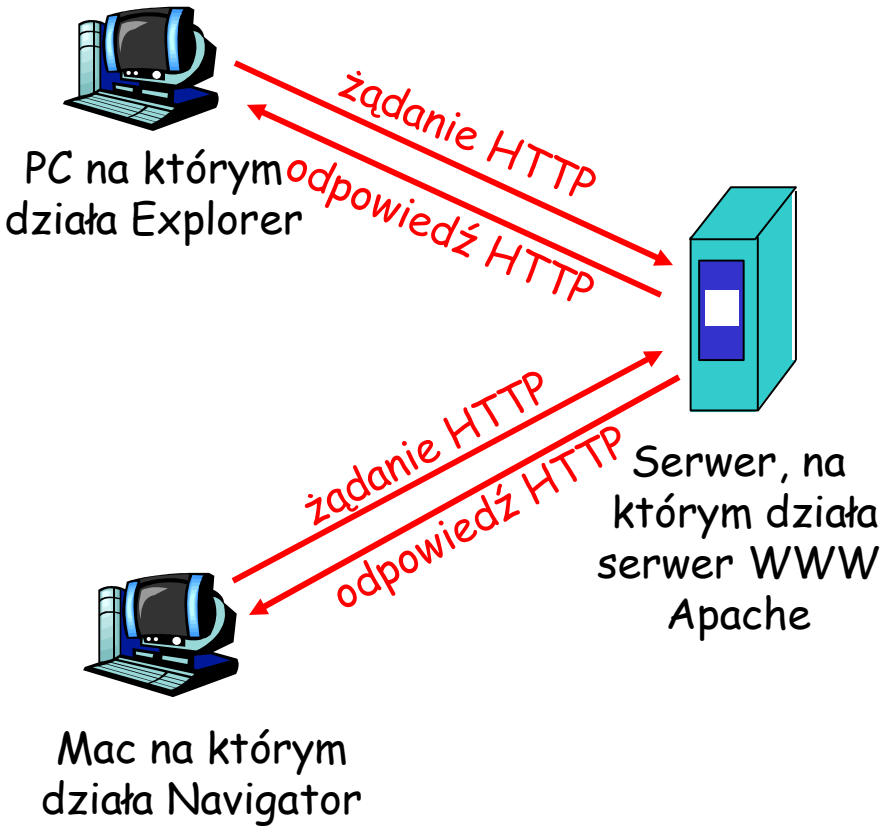
`www.szkola.edu.pl/katedra/pic.gif`

nazwa hosta ścieżka

Przegląd HTTP

HTTP: hypertext transfer protocol

- Protokół warstwy aplikacji WWW
- model klient/serwer
  - *klient:* przeglądarka żąda, otrzymuje, "wyświetla" obiekty WWW
  - *serwer:* serwer WWW wysyła obiekty w odpowiedzi na żądania
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



Przegląd HTTP (c.d.)

Używa TCP:

- klient nawiązuje połączenie TCP (tworzy gniazdo) do serwera, port 80
- serwer przyjmuje połączenie TCP od klienta
- przeglądarka (klient HTTP) i serwer WWW (serwer HTTP) wymieniają komunikaty HTTP (komunikaty protokołu warstwy aplikacji)
- połączenie TCP jest zamykane

HTTP jest "bezstanowy"

- serwer nie utrzymuje i nie wykorzystuje informacji o uprzednich połączeniach HTTP

dygresja

Protokoły, które utrzymują "stan", są złożone!

- historia (stan) musi być utrzymywana
- jeśli serwer lub klient będzie miał awarię, informacje o stanie mogą być niezgodne, muszą zostać ujednolicone
- są problemy z bezpieczeństwem

Połączenie HTTP

Nietrwałe HTTP

- Najwyżej jeden obiekt jest posyłany przez połączenie TCP, które potem zostaje zamknięte.
- HTTP/1.0 używa nietrwałego HTTP

Trwałe HTTP

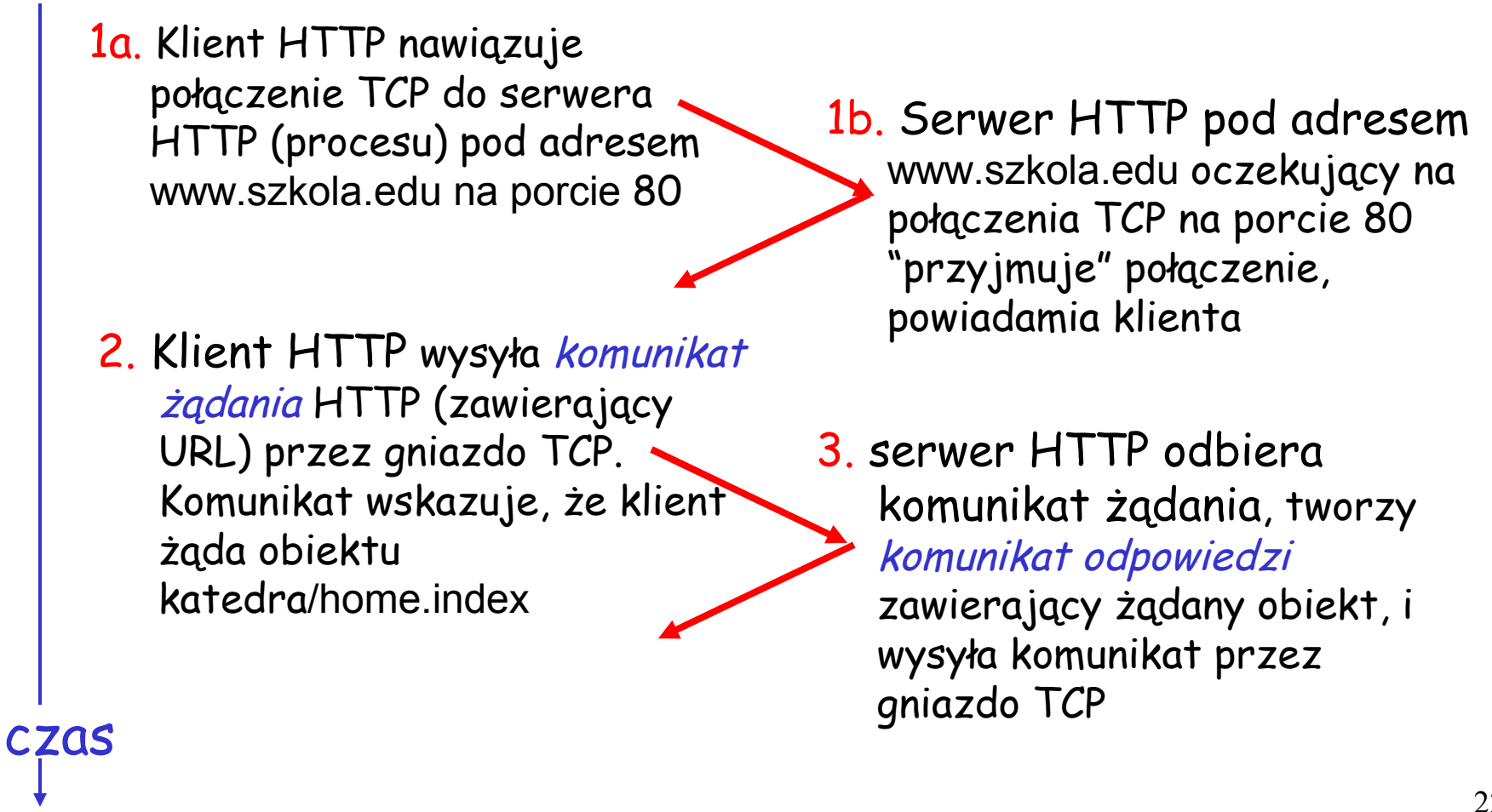
- Wiele obiektów mogą zostać posłane przez jedno połączenie TCP pomiędzy klientem i serwerem.
- HTTP/1.1 domyślnie używa trwałego HTTP

Nietrwałe HTTP

Użytkownik wprowadza adres URL

www.szkola.edu.pl/katedra/home.index

(strona zawiera tekst, wskazania do 10 obrazów jpeg)



Nietrwałe HTTP(c.d.)

4. Serwer HTTP zamyka połączenie TCP.

5. Klient HTTP odbiera komunikat odpowiedzi zawierający plik html, wyświetla html. Parsując plik html, znajduje 10 wskazywanych obiektów jpg

6. Kroki 1-5 są powtarzane dla każdego z 10 obiektów jpg

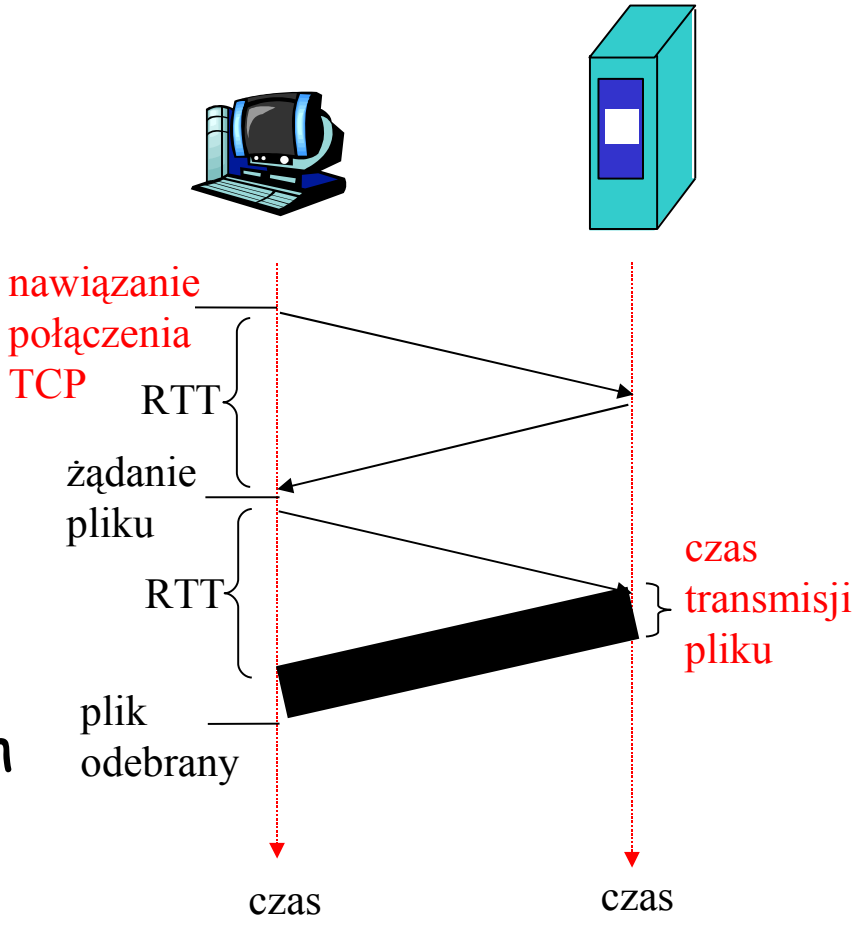
Modelowanie czasu obsługi

Definicja RTT (ang. Round Trip Time): czas potrzebny na przesłanie małego pakietu od nadawcy do odbiorcy i z powrotem.

Czas obsługi:

- 1 RTT na część inicjacji połączenia TCP
- 1 RTT na przesłanie żądania HTTP i na powrót pierwszych bajtów odpowiedzi HTTP
- czas transmisji pliku

razem = 2 RTT + czas transmisji



Trwałe HTTP

Nietrwałe HTTP:

- wymaga 2 RTT na każdy obiekt
- System operacyjny hosta musi pracować i udostępniać zasoby na każde połączenie TCP
- powolny start TCP
- jednak przeglądarki często otwierają równoległe połączenia TCP

Trwałe HTTP

- serwer zostawia otwarte połączenie po wysłaniu odpowiedzi
- następne komunikaty HTTP pomiędzy tym samym klientem i serwerem na tym samym połączeniu TCP

Trwałe HTTP bez grupowych żądań:

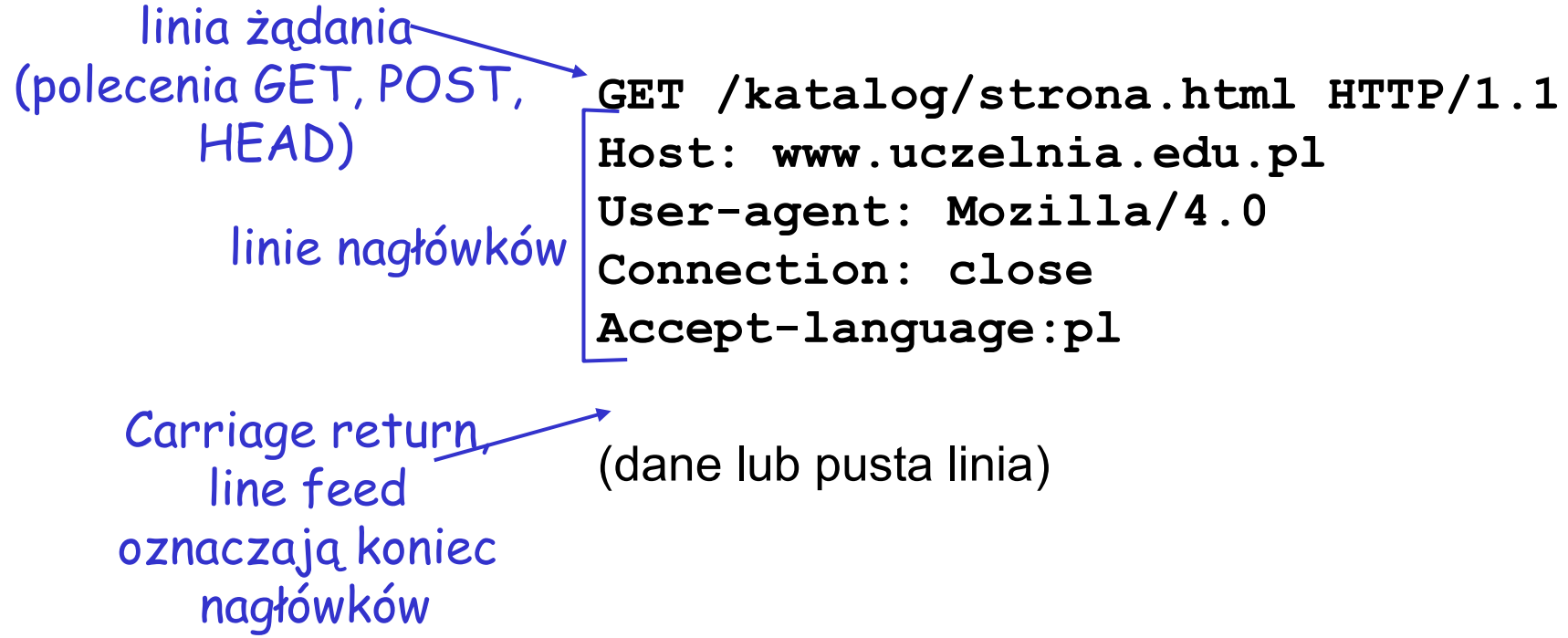
- klient wysyła nowe żądanie dopiero, gry odebrał poprzednią odpowiedź
- 1 RTT na każdy żądany obiekt

Trwałe z grupowymi żądaniami:

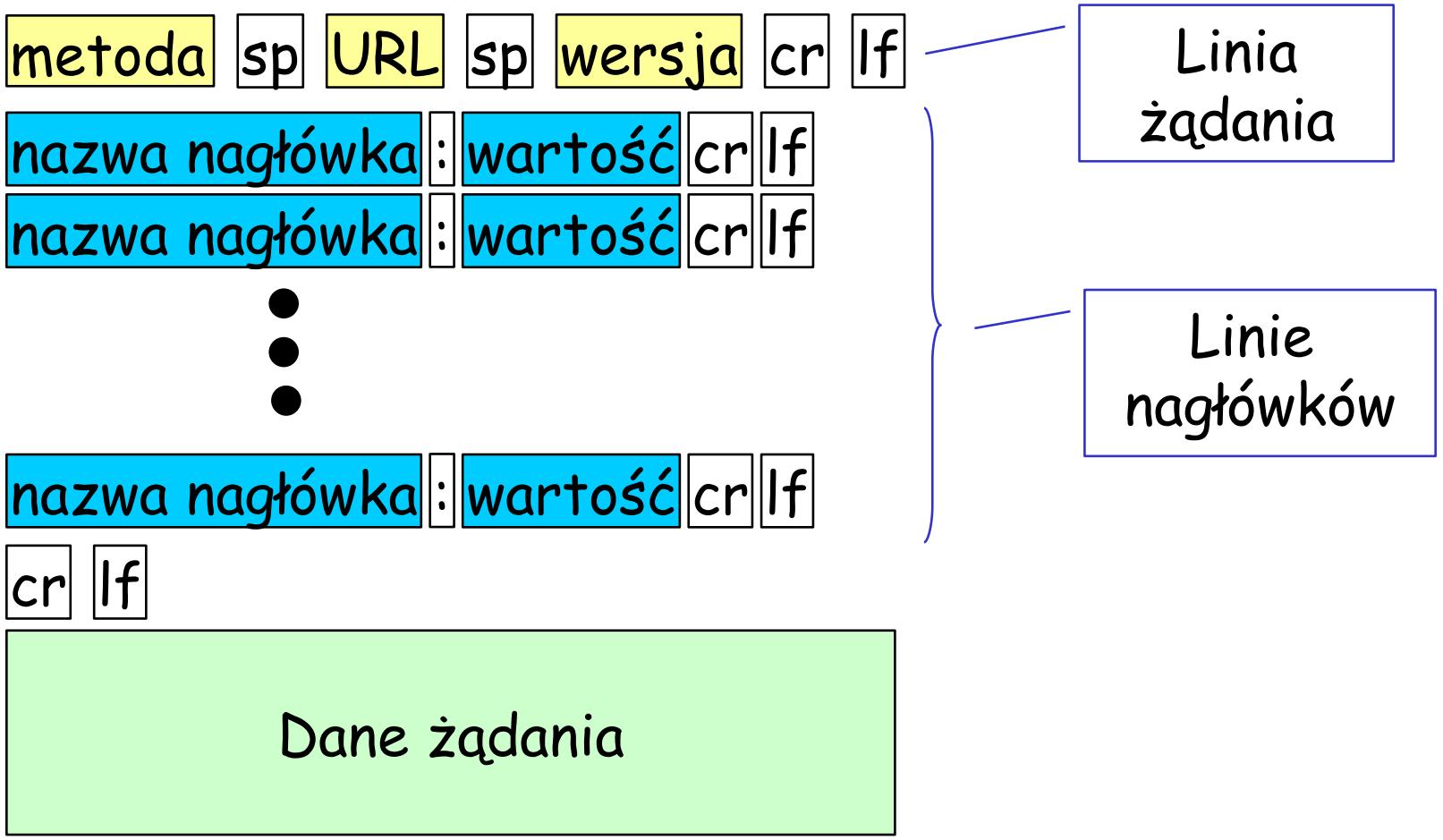
- domyślne w HTTP/1.1
- klient wysyła żądanie jak tylko znajdzie w stronie wskazany obiekt
- tylko jeden RTT dla wszystkich żądanych obiektów

Komunikat żądania HTTP

- dwa rodzaje komunikatów HTTP: *żądanie, odpowiedź*
- Komunikat żądania HTTP:
  - ASCII (format czytelny dla człowieka)



Komunikat żądania HTTP: ogólny format



Interfejs CGI

Interfejs CGI (ang. Common Gateway Interface):

- Strony WWW często zawierają formularze
- Dane z formularzy są przekazywane przez serwer WWW do skryptów

Metoda POST:

- Zawartość formularza jest posyłana do serwera w danych żądania

Kodowanie w URL:

- Używa metody GET
- Zawartość formularza jest kodowana w adresie URL żądania:

```
www.somesite.com/animalsearch?monkeys&banana
```

Najczęściej używane metody

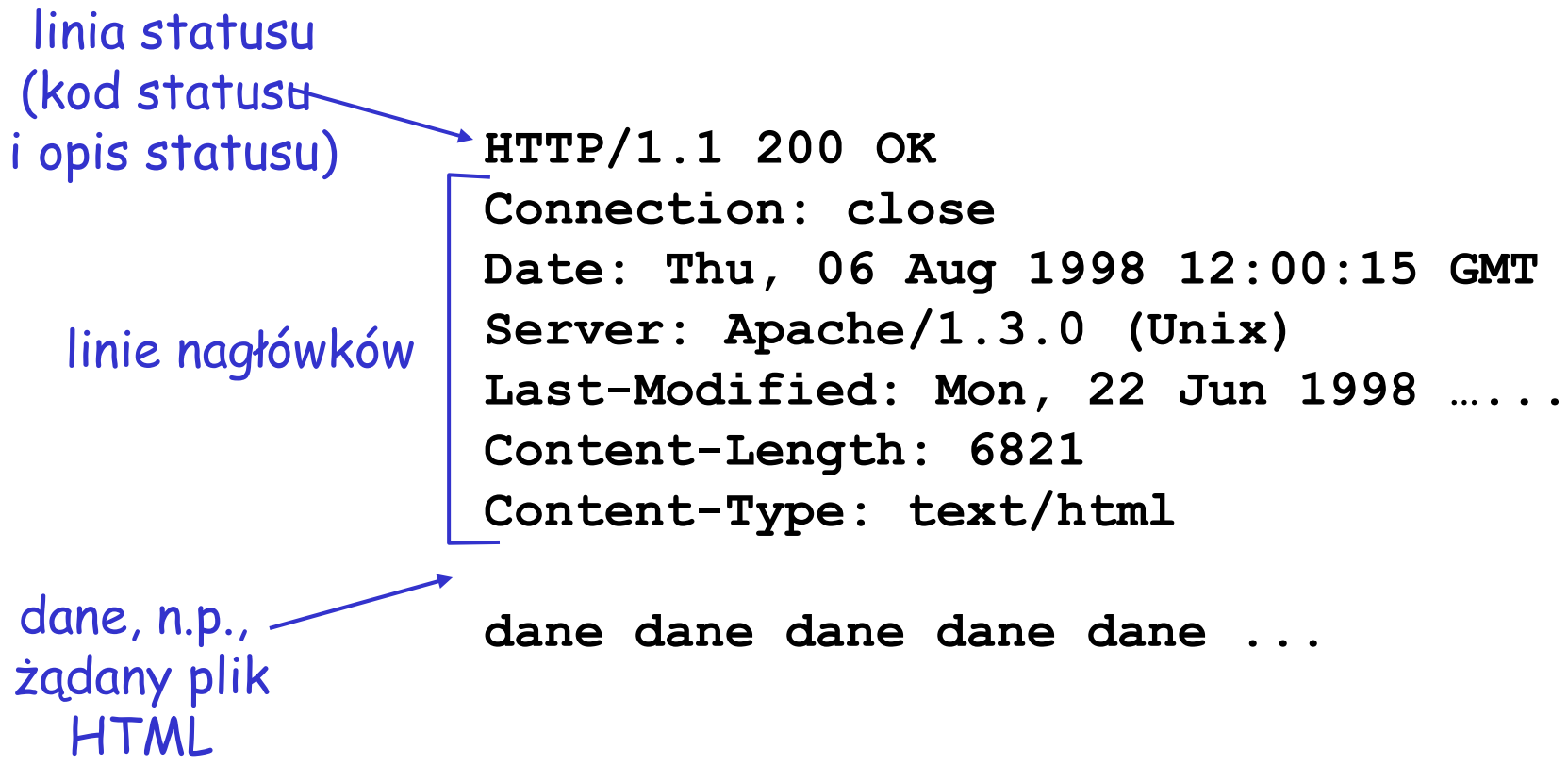
HTTP/1.0

- GET
- POST
- HEAD
  - prosi serwer o posłanie odpowiedzi bez żądanego obiektu (danych)

HTTP/1.1

- GET, POST, HEAD
- PUT
  - wysyła plik w danych żądania, który zostanie umieszczony pod adresem URL (ścieżką)
- DELETE
  - usuwa obiekt o adresie URL

Komunikat odpowiedzi HTTP



Kody statusu odpowiedzi HTTP

Zawarte w pierwszej linii odpowiedzi HTTP.

Parę przykładów:

200 OK

- żądanie pomyślnie obsłużone, żądany obiekt będzie w danych odpowiedzi

301 Moved Permanently

- żądany obiekt przeniesiony, nowy adres dalej w nagłówku (Location:)

400 Bad Request

- komunikat żądania nie został zrozumiany przez serwer (błąd składni)

404 Not Found

- żądanego obiektu nie ma na serwerze

505 HTTP Version Not Supported

Spróbujcie sami... być klientem HTTP

1. Wykonać telnet na wybrany serwer WWW:

```
telnet www.pjwstk.edu.pl 80
```

Otwiera połączenie TCP na port 80 (domyślny port serwera HTTP) pod adresem www.eurecom.fr. Wszystkie wpisane znaki zostaną przesłane na port 80 pod adresem www.eurecom.fr

2. Wpisać żądanie HTTP GET:

```
GET /index.html HTTP/1.0
```

Wpisując tę linię (nacisnąć dwa razy enter), wysyła się minimalne (ale pełne) żądanie HTTP GET do serwera HTTP

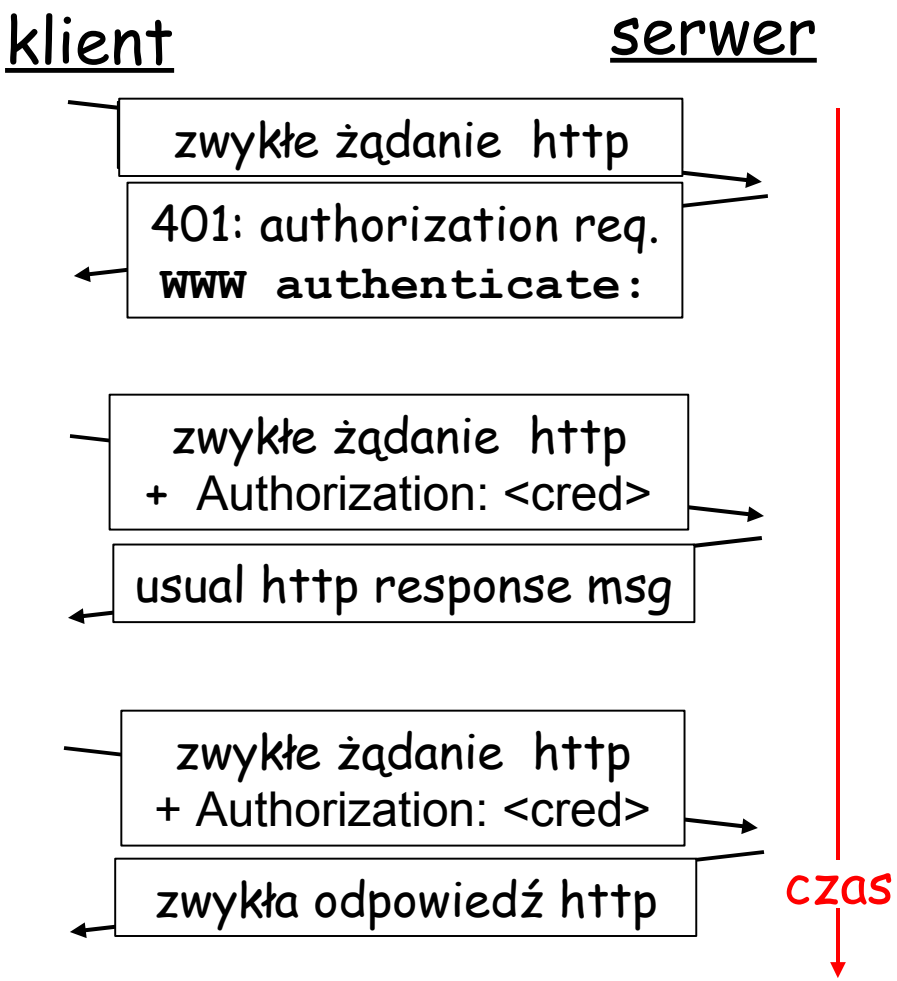
3. Obejrzeć odpowiedź przysłaną przez serwer HTTP

Interakcja użytkownika z serwerem: uwierzytelnienie

Uwierzytelnienie : kontroluje dostęp do treści serwera

- informacja uwierzytelniająca: typowo logi, hasło
- bezstanowe: klient musi przedstawić informację w *każdym* żądaniu
  - authorization: linia nagłówka w każdym żądaniu
  - jeśli nie ma nagłówka authorization, serwer odmawia dostępu, wysyła w odpowiedzi nagłówek

    `WWW authenticate`



Ciasteczka: utrzymywanie "stanu"

Wiele znanych portali WWW używa ciasteczek

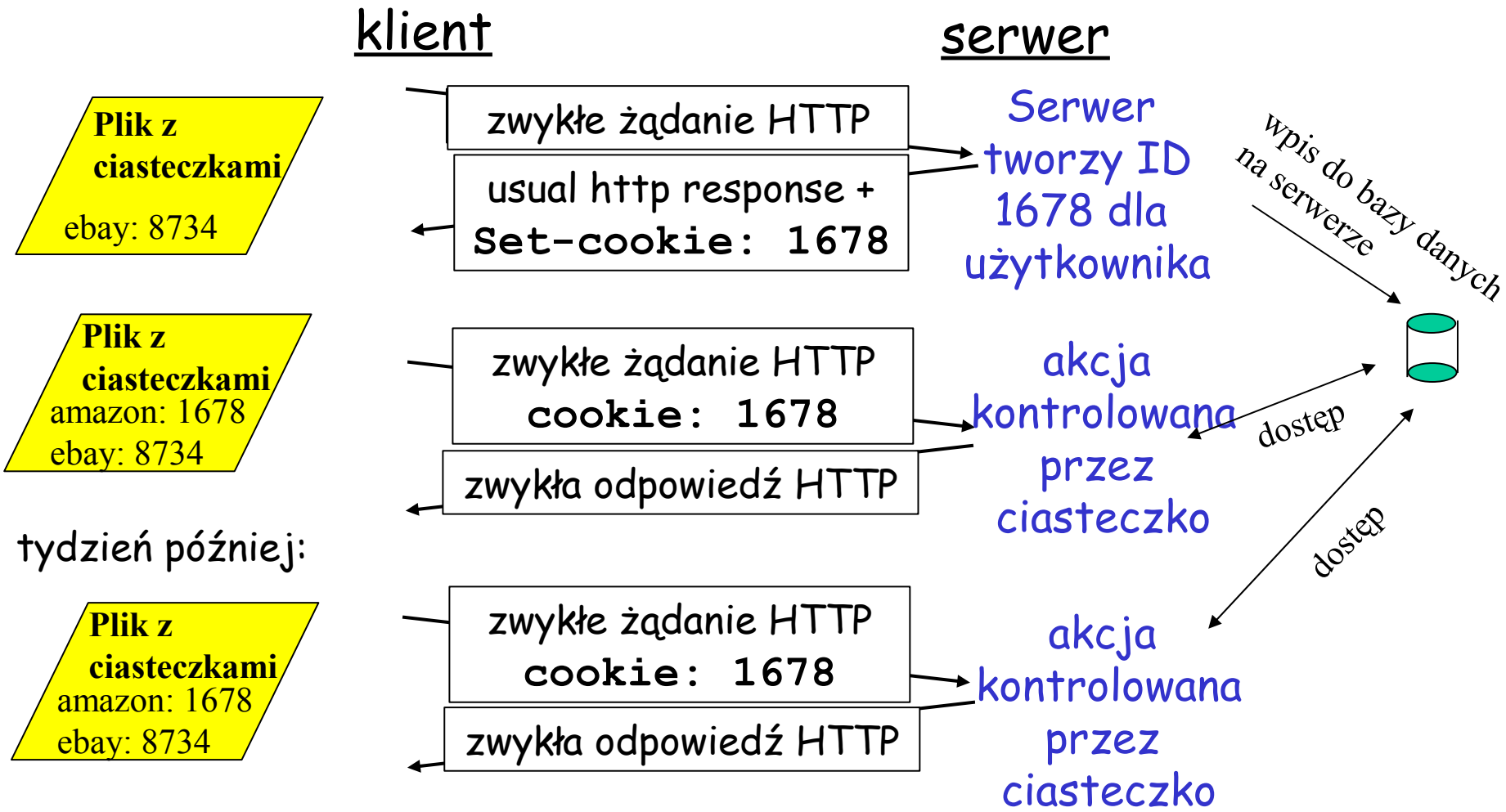
Cztery składniki:

1) nagłówek cookie w odpowiedzi HTTP
2) nagłówek cookie w żądaniu HTTP
3) plik z ciasteczkami na hoście klienta, zarządzany przez przeglądarkę klienta
4) baza danych na serwerze WWW

Przykład:

- Ania ma dostęp do Internetu z zawsze tego samego komputera
- Odwiedza pewien portal e-commerce po raz pierwszy
- Gdy pierwsze żądanie HTTP przybywa do portalu, portal tworzy unikalny identyfikator i wpis w bazie danych na serwerze

Ciasteczka: utrzymywanie "stanu" (c.d.)



Ciasteczka (c.d.)

Co można dzięki ciasteczkom:

- uwierzytelnienie
- wózki z zakupami
- rekomendacje
- stan sesji użytkownika (np. w banku elektronicznym)

dygresja

Ciasteczka a prywatność:

- ciasteczka pozwalają portalom dowiedzieć się wiele o użytkownikach
- na niektórych portalach, podaje się nazwisko i adres poczty elektronicznej
- poprzez ciasteczka, wyszukiwarki mogą poznać więcej informacji
- firmy marketingowe uzyskują informacje z wielu portali i łączą je

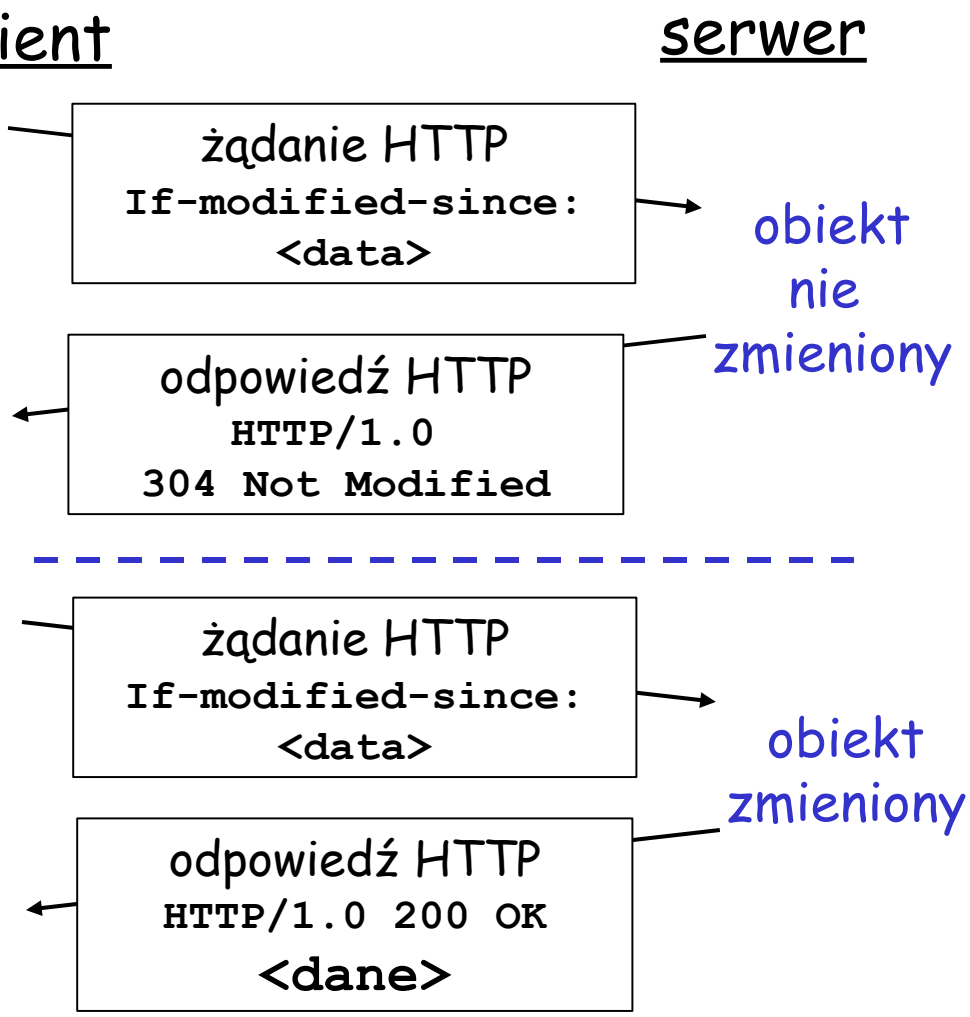
Warunkowy GET: schowki u klienta

- Cel: nie wysyłać obiektów, jeśli klient ma aktualną kopię w schowku
- klient: podaje datę kopii w schowku w żądaniu HTTP

```
If-modified-since:
  <data>
```

- serwer: odpowiedź nie zawiera obiektu, jeśli kopia jest aktualna:

```
HTTP/1.0 304 Not
  Modified
```



Mapa wykładu

- 2.1 Zasady budowy protokołów w. aplikacji
- 2.2 WWW i HTTP
- 2.3 DNS
- 2.4 Programowanie przy użyciu gniazd TCP
- 2.5 Programowanie przy użyciu gniazd UDP
- 2.6 Poczta elektroniczna
  - SMTP, POP3, IMAP
- 2.7 FTP
- 2.8 Dystrybucja zawartości
  - Schowki Internetowe
  - Sieci dystrybucji zawartości
- 2.9 Dzielenie plików P2P

DNS: Domain Name System

Ludzie: wiele identyfikatorów:

- PESEL, nazwisko, numer paszportu

Hosty, rutery Internetu:

- adres IP (32 bity) – używany do adresowania pakietów

Czy to wystarczy?

Co zrobić, jeśli adres IP musi ulec zmianie?

Jak określać usługi, które są realizowane przez wiele serwerów?

Jak odróżniać różne usługi, które są realizowane przez jeden

Domain Name System:

- *rozproszona baza danych* implementowana przez hierarchię wielu *serwerów nazw*
- *protokół warstwy aplikacji* hosty, rutery, serwery nazw komunikują się, żeby *tłumaczyć* nazwy
  - uwaga: jedna z głównych funkcji Internetu, implementowana jako protokół w warstwie aplikacji
  - złożoność na "brzegu" sieci

Rozwiązanie:
"nazwa", n.p., www.pjwstk.edu.pl – używana przez ludzi

Pytanie: jak tłumaczyć pomiędzy adresami IP i nazwami?

Serwery nazw DNS

Czemu nie scentralizować DNS?

- zagrożenie pojedynczą awarią
- ilość ruchu
- odległość od scentralizowanej bazy
- aktualizacje

taki projekt nie jest *skalowalny!*

Zasada delegacji

- organizacja zarządza strefą nazw
- w obrębie strefy, może wydzielać mniejsze strefy
- organizacja przekazuje zarządzanie za strefę innym organizacjom

- żaden serwer nie zna wszystkich odwzorowań adresów IP i nazw DNS

lokalne serwery nazw:

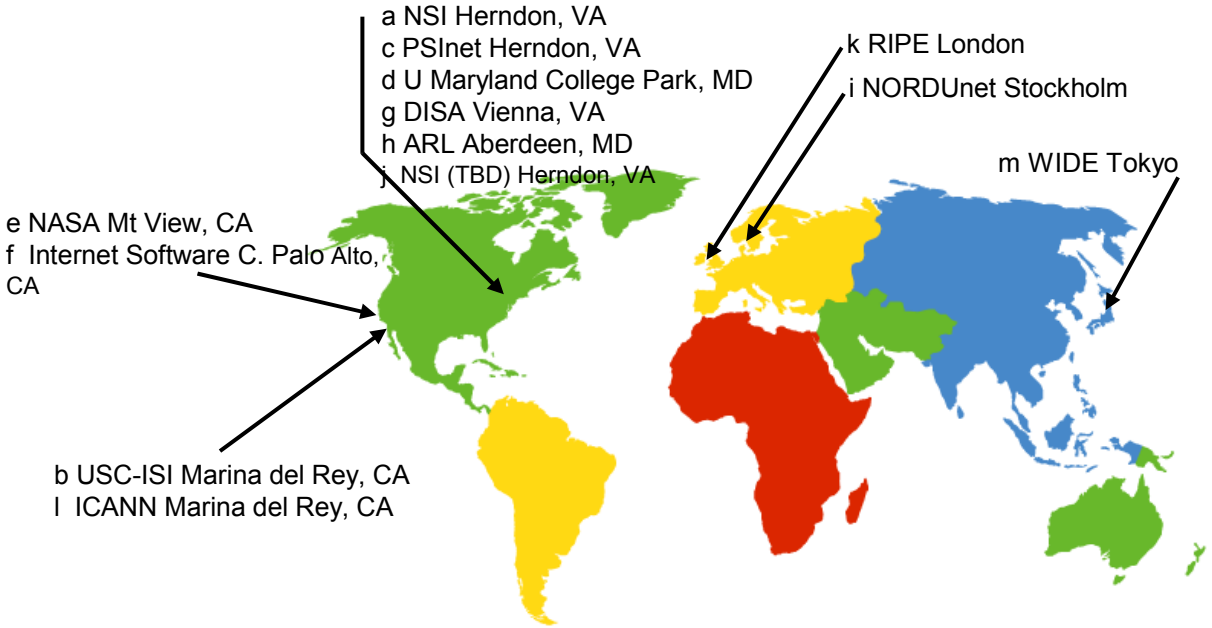
- każdy DI, organizacja ma *lokalny (domyślny) serwer nazw*
- pytanie DNS z hosta jest kierowane najpierw do lokalnego serwera nazw

autorytatywny serwer nazw:

- dla hosta: przechowuje adres IP i nazwę DNS hosta
- może dokonać odwzorowania pomiędzy nazwą i adresem dla tego hosta

DNS: serwery u korzenia

- lokalny serwer nazw pyta serwer u korzenia, gdy nie może przetłumaczyć nazwy
- serwer u korzenia:
  - kontaktuje się z serwerem autorytatywnym, jeśli nie zna odwzorowania nazwy
  - otrzymuje odwzorowanie
  - przekazuje odwzorowanie do lokalnego serwera nazw

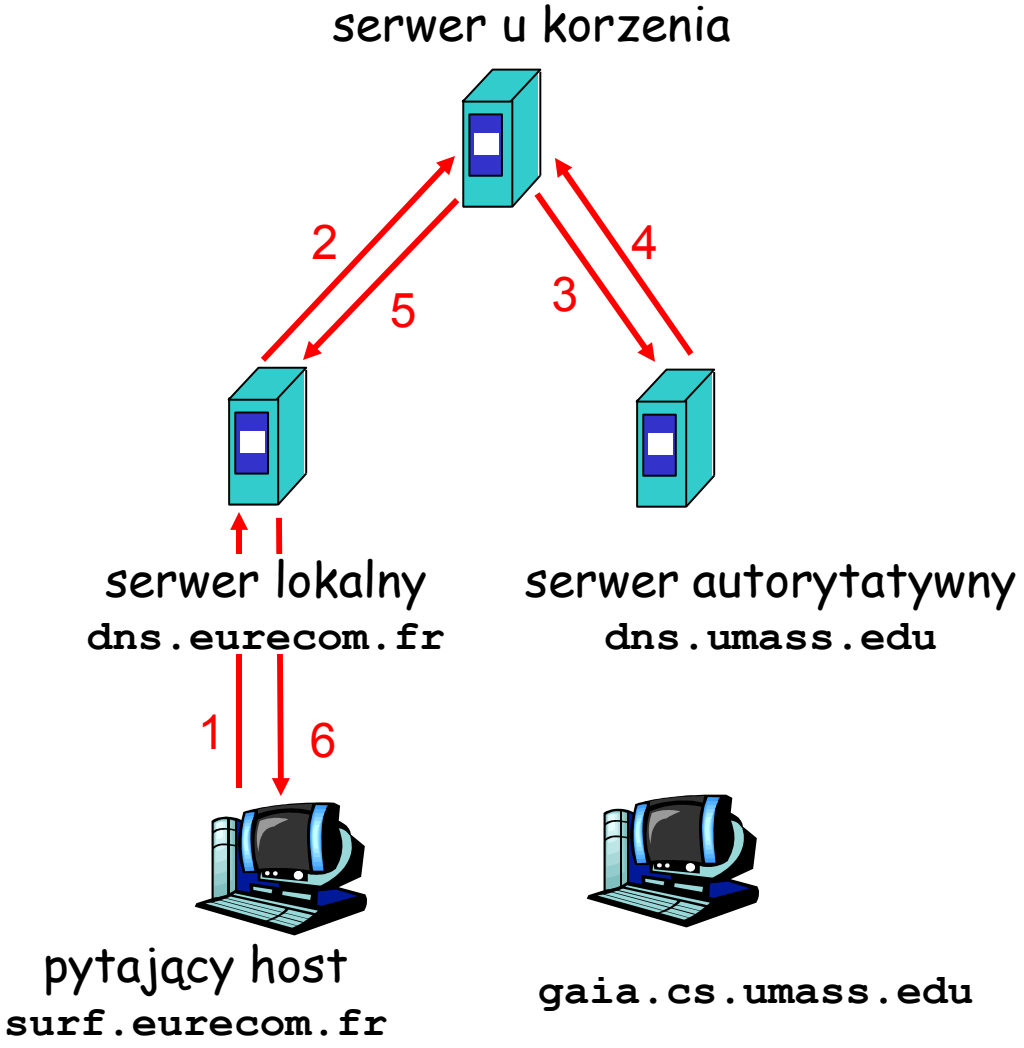


13 serwerów u korzenia na całym świecie

Prosty przykład działania DNS

host **surf.eurecom.fr** potrzebuje adresu IP **gaia.cs.umass.edu**

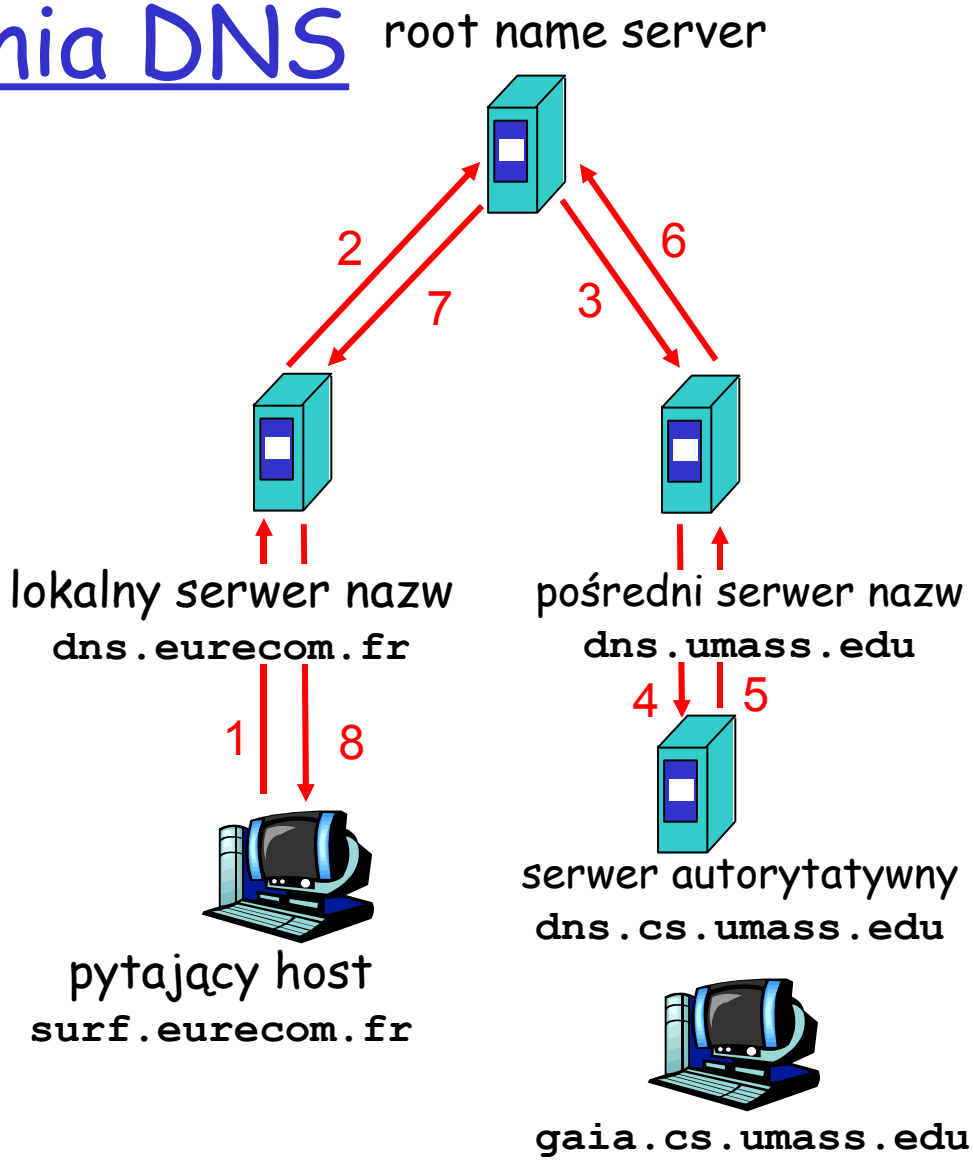
1. pyta swój lokalny serwer DNS, **dns.eurecom.fr**
2. **dns.eurecom.fr** pyta serwer u korzenia, jeśli to konieczne
3. serwer u korzenia pyta serwer autorytatywny, **dns.umass.edu**, jeśli to konieczne



Przykład działania DNS

Serwer u korzenia:

- może nie znać serwera autorytatywnego
- może znać *pośredni serwer nazw:* kogo spytać o autorytatywny serwer



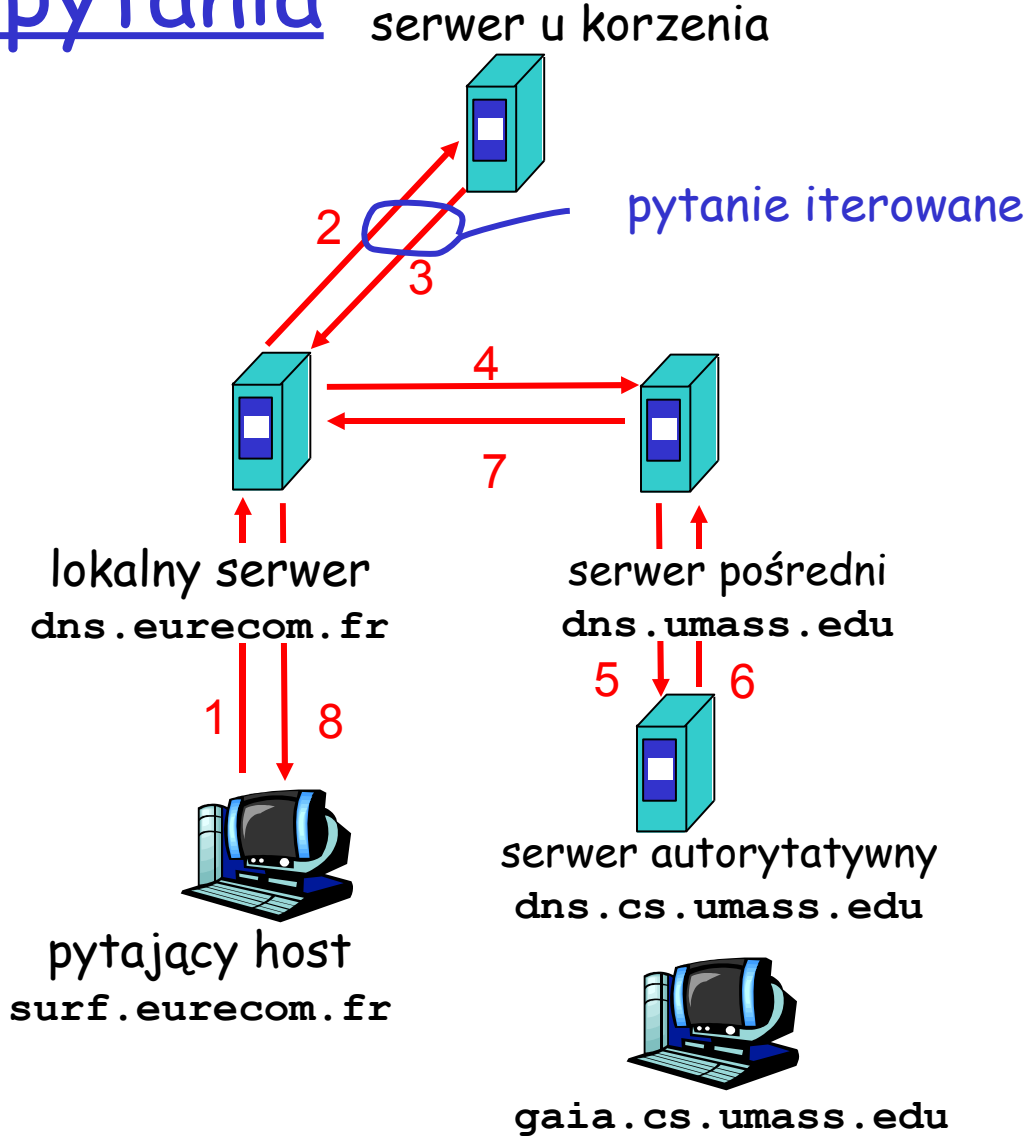
DNS: iterowane pytania

pytanie rekurencyjne:

- obciąża pytany serwer zadaniem zdobycia odpowiedzi
- duże obciążenie?

pytanie iterowane:

- pytany serwer odpowiada adresem serwera, który należy pytać dalej
- "Nie znam tej nazwy, ale spytaj ten serwer"



DNS: schowki i aktualizacja rekordów

- gdy (dowolny) serwer nazw pozna odwzorowanie, *zachowuje je w schowku*
  - pozycje w schowku ulegają dezaktualizacji (znikają) po pewnym czasie
- mechanizmy aktualizacji (powiadamiania) są projektowane prze zIETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Rekordy DNS

DNS: rozproszona baza danych przechowująca rekordy zasobów (RZ)

format RZ: **(nazwa, wartość, typ,czas życia)**

- Typ=A
  - **nazwa hosta**
  - **wartość** jest adresem IP
- Typ=NS
  - **nazwa jest domeną** (n.p. edu.pl)
  - **wartość** jest adresem IP autorytatywnego serwera nazw dla tej domeny
- Typ=CNAME
  - **nazwa** jest aliasem dla pewnej "kanonicznej" (prawdziwej) nazwy

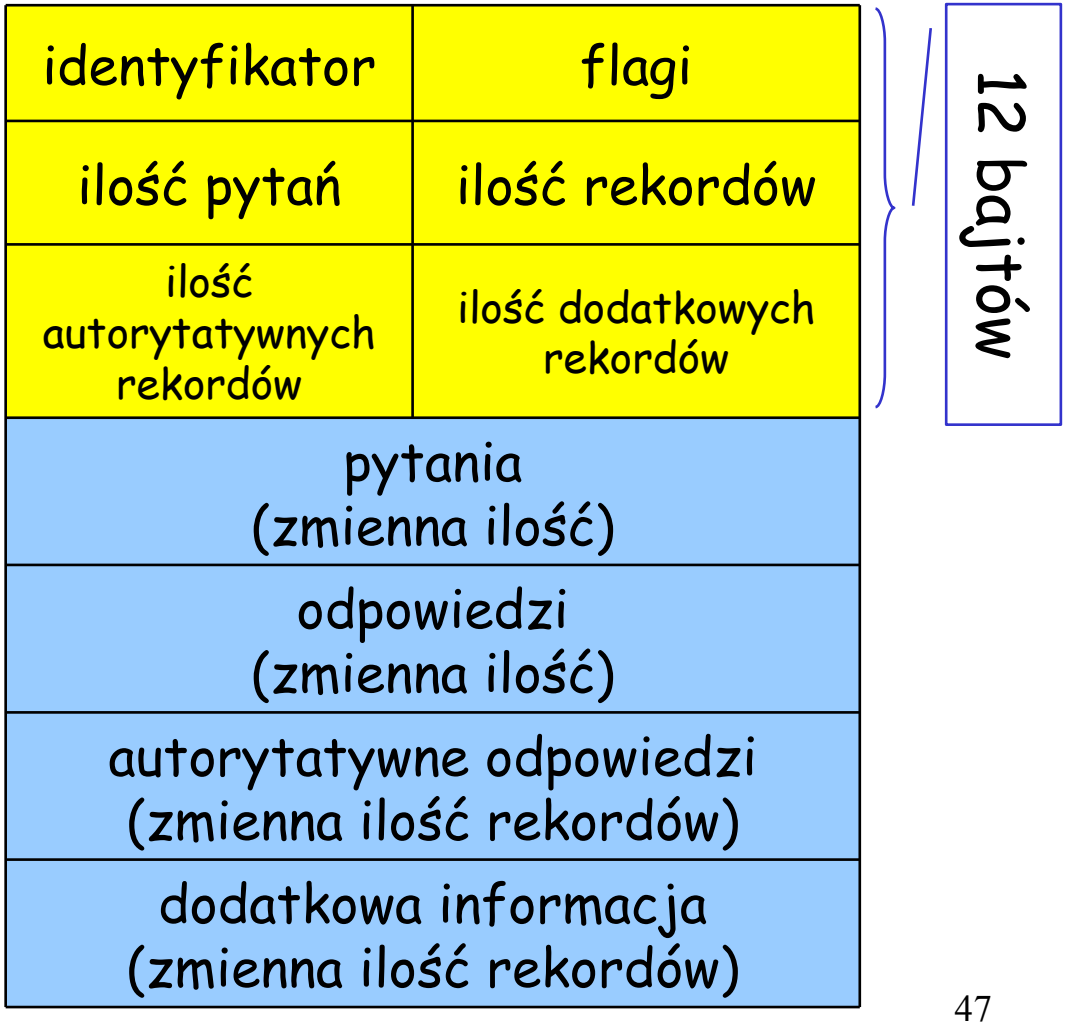
    `www.ibm.com` jest naprawdę `servereast.backup2.ibm.com`
  - **wartość** jest nazwą kanoniczną
- Typ=MX
  - **wartość** jest nazwą serwera poczty związanego z nazwą

Protokół, komunikaty DNS

Protokół DNS : komunikaty *pytania* i *odpowiedzi*, oba z tym samym *formatem komunikatu*

nagłówek komunikatu

- identyfikacja: 16 bitów na numer pytanie, odpowiedź używa tego samego numeru
- flagi:
  - pytanie lub odpowiedź
  - żądana rekurencja
  - rekurencja dostępna
  - odpowiedź jest autorytatywna



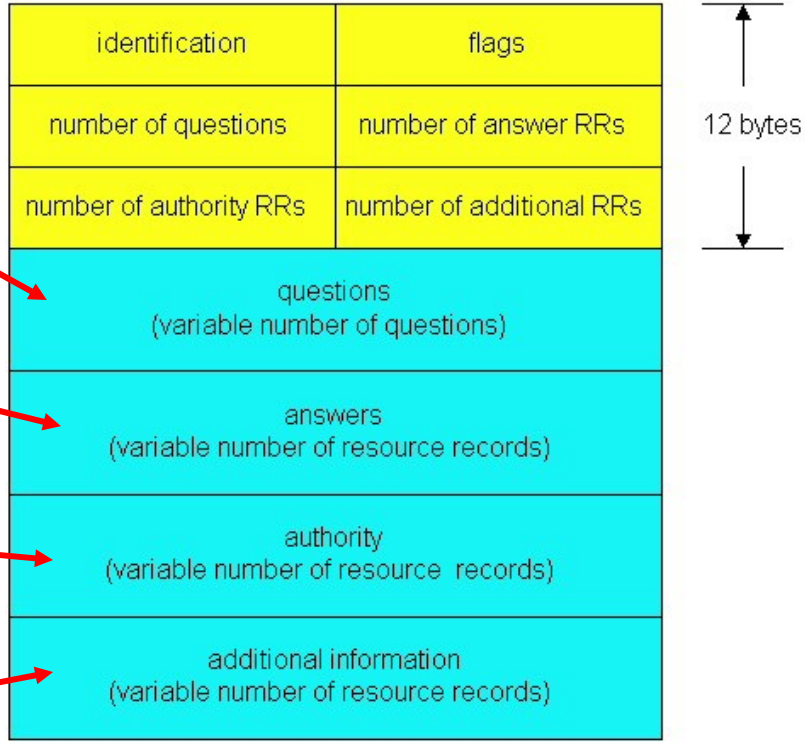
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



Mapa wykładu

- 2.1 Zasady budowy protokołów w. aplikacji
- 2.2 WWW i HTTP
- 2.3 DNS
- 2.4 Programowanie przy użyciu gniazd TCP
- 2.5 Programowanie przy użyciu gniazd UDP
- 2.6 Poczta elektroniczna
  - SMTP, POP3, IMAP
- 2.7 FTP
- 2.8 Dystrybucja zawartości
  - Schowki Internetowe
  - Sieci dystrybucji zawartości
- 2.9 Dzielenie plików P2P