Sieci Komputerowe i bazy danych

Warstwa aplikacji

dr inż. Paweł Penar

Niektóre aplikacje sieciowe

- social networking
- Web
- Wiadomości tekstowe
- e-mail
- gry
- streaming
- P2P file sharing
- Rozmowy IP (np. Skype)
- Zdalny dostęp
- …

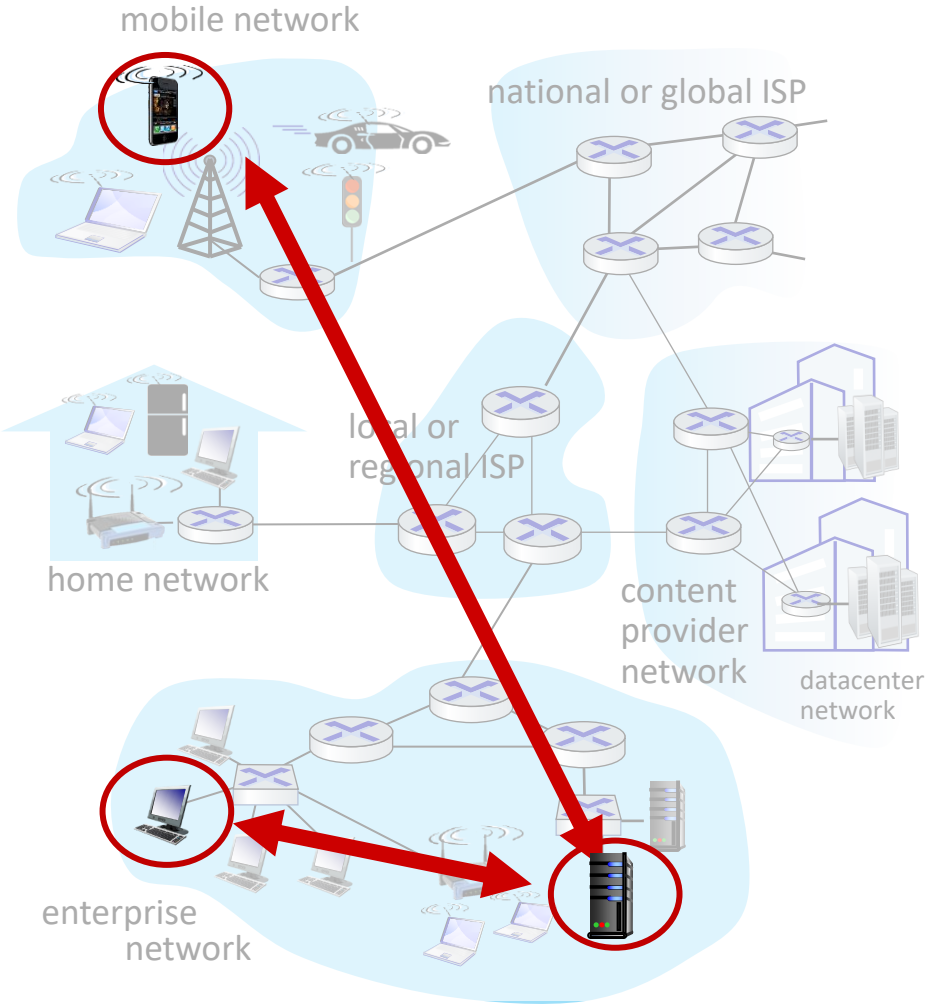
Paradygmat klient-server

server:

- zawsze dostępny host
- stały adres IP
- często centrum danych (data center)

klient:

- komunikacja z serverem
- połączony z przerwami
- może mieć dynamiczny adres
- Nie jest bezpośrednio połączony
- np. HTTP, IMAP, FTP



Skomunikowanie procesów

proces: program uruchomiony w ramach hosta

- w ramach tego samego hosta dwa procesy komunikują się używając komunikacji międzyprocesowej (zdefiniowanej przez OS)
- Procesy różnych hostach komunikują się przez wymianę wiadomości

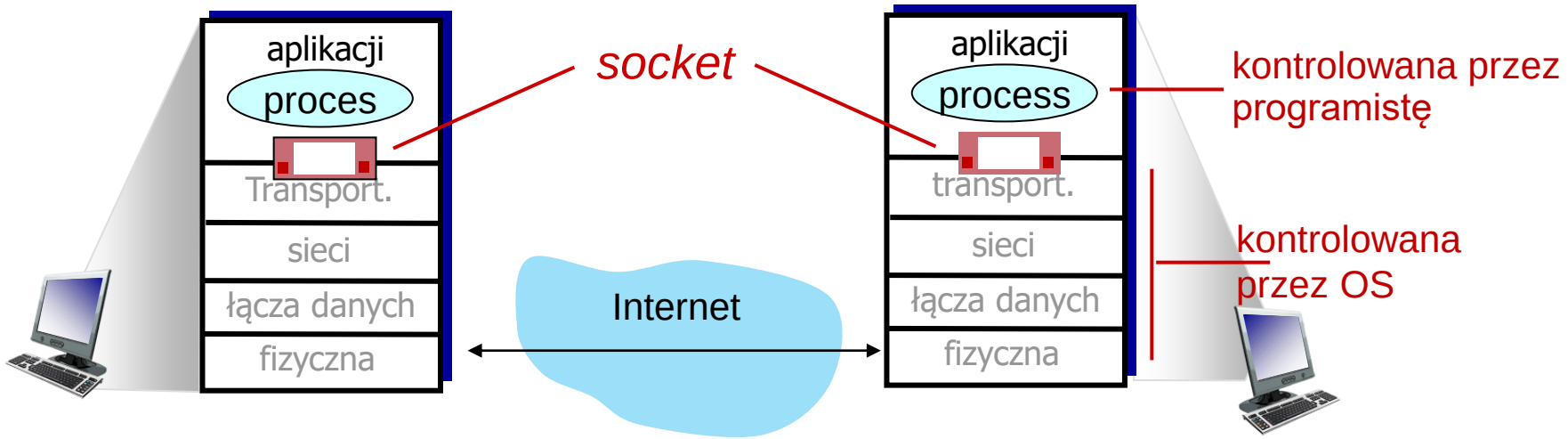
klienci, servery

proces klienta: proces który inicjuje komunikacje

proces servera: proces który oczekuje na połaczenie

Sockety

- Proces wysyła/odbiera wiadomość z/od socketu
- To punkty końcowe komunikacji pomiędzy dwoma programami pracującymi w sieci
- Zatem: adresuje proces



Adresowanie procesu

- by otrzymać wiadomość, proces musi być *identyfikowalny*
- host identyfikuje adres IP
- *Q:* czy adres IP hosta wystarczy do identyfikacji procesu?
  - *A:* nie, bo w hoście może działać wiele procesów

- *identyfikator* zawiera zarówno adres IP i numer portu powiązany z procesem w hoście.
- przykładowe numery portów:
  - HTTP server: 80
  - mail server: 25
- aby wysłać wiadomość HTTP do serwera gaia.cs.umass.edu:
  - IP address: 128.119.245.12
  - port number: 80
- krócej ...

Jakich usług potrzebują aplikacje?

Integralność danych

wydajność

- niektóre aplikacje wymagają minimalnej przepustowości
- inne aplikacje wykorzystują dowolną przepustowość

timing

- wymagania dotyczące opóźnień

bezpieczeństwo

- kodowanie, integralność danych

Usługi protokołów warstwy transportowej

Usługi TCP:

- *niezawodny transport* pomiędzy procesami
- *kontrola przepływu:* nadawca „nie przytłacza" odbiorcy
- *kontrola przeciążenia:* ogranicza nadawcę, gdy sieć jest przeciążona
- *zorientowany na połączenie:* swymaga konfiguracji połączenie klient-server
- *nie zapewnia:* synchronizacji, gwarancji przepust., bezpieczeństwa

Usługi UDP:

- *Niepewny transfer danych* pomiędzy procesem wysyłającym i odbierającym
- *nie zapewnia*: niezawodności, kontroli przepływu, kontroli przeciążenia, synchronizacji, gwarancji przepustowości, bezpieczeństwa ani konfiguracji połączenia.

  Q: po co się męczyć? *Dlaczego istnieje* UDP?

Protokoły warstwy aplikacji i protokoły warstwy transportowej

aplikacja	protokół warstwy aplikacji	protokół transpor.
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP [RFC 7230, 9110]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7230], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

Web i HTTP

Małe przypomnienie….

- strona internetowa składa się z *obiektów,* które mogą być hostowane na różnych serwerach
- obiekty to np. plik HTML, plik JPEG, aplet Java, pliki audio…
- Strona składa się z *podstawowego pliku HTML* który zawiera *referencje do obiektów* adresowalne przez *URL (Uniform Resource Locator),* np.,

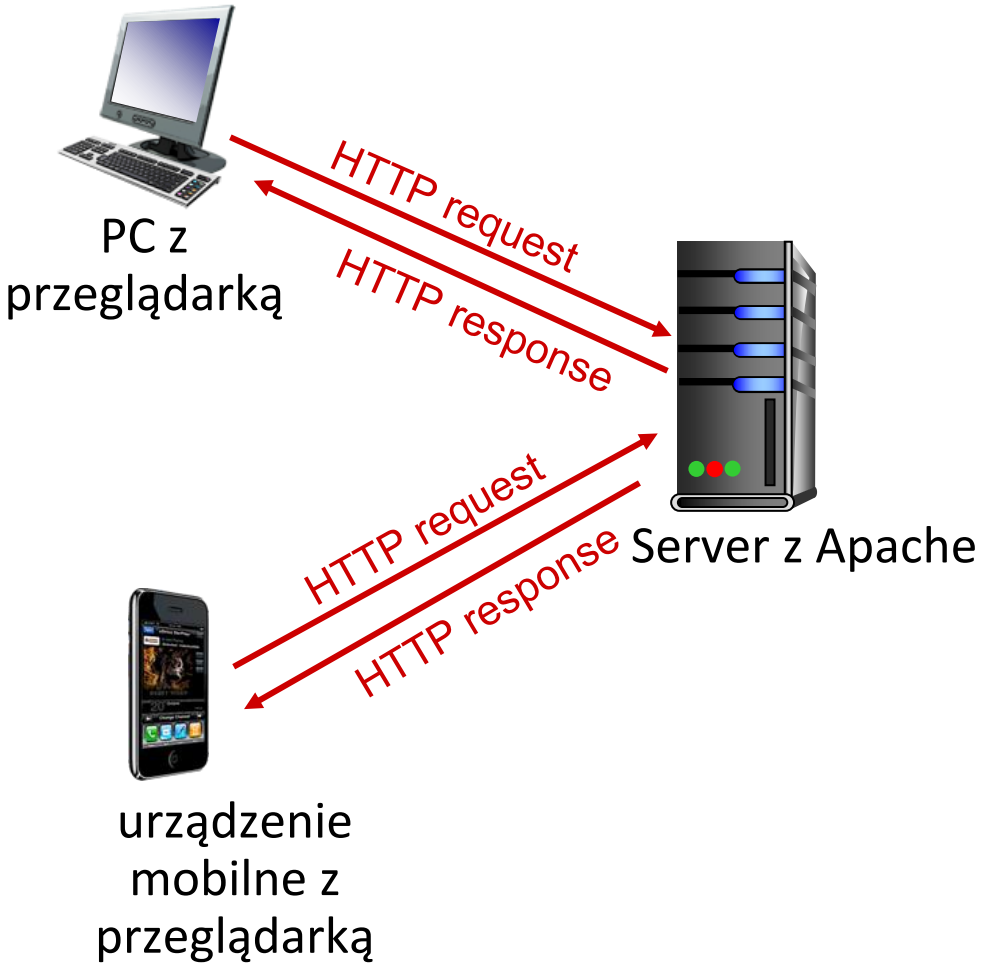
`www.someschool.edu/someDept/pic.gif`

Nazwa hosta | Nazwa ścieżki

HTTP

HTTP: hypertext transfer protocol

- protokół warstwy aplikacji
- działa w modelu klient-server:
  - *klient:* żądanie przeglądarki, odebranie, wyświetlenie obiektów strony
  - *server:* server webowy wysyła obiekty w odpowiedzi na żądanie



HTTP

HTTP używa TCP:

- Klient inicjuje połączenie TCP z serwerem (tworzy socket) na porcie 80
- server akceptuje połączenie TCP od klienta
- Komunikaty/wiadomości HTTP wymieniane pomiędzy przeglądarką klienta a serverem
- TCP zamyka połączenie closed

HTTP jest "bezstanowy"

- serwer nie przechowuje żadnych informacji o wcześniejszych żądaniach klientów

Dwa typy połączenia HTTP

Nietrwałe połaczenie HTTP

1. Otwarcie połączenia TCP
2. co najwyżej jeden obiekt przesyłany przez połączenie
3. zamknięcie połączenia TCP

Pobranie wielu obiektów wymaga wielu połaczeń

Trwałe połączenie HTTP

- Otwarcie połączenia TCP
- Wiele obiektów przesyłanych jednym połączeniem TCP
- Zamknięcie połączenia TCP

Nietrwałe połączenie HTTP: przykład

Użytkownik wybiera URL: **`www.prz.edu.pl/kmsir/home.index`**
(zawiera tekst i odniesienia do 10 obrazów jpeg)

1a. Klient HTTP inicjuje połączenie TCP z serwerem TCP www.prz.edu.pl na porcie 80

1b. Server HTTP hosta www.prz.edu.pl oczekuje na połączenie TCP na porcie 80; "akceptuje" połączenie powiadamiając klienta

2. Klient HTTP wysyła *żądanie* HTTP (zawierające URL) do gniazda połączenie TCP. Wiadomość wskazuje, że klient chce uzyskać obiekt kmsir/home.index

3. Serwer HTTP odbiera wiadomość z żądaniem; tworzy wiadomość *odpowiedzi* zawierającą żądany obiekt i wysyła wiadomość przez socket.

czas

Nietrwałe połączenie HTTP: przykład

Użytkownik wybiera URL: **`www.prz.edu.pl/kmsir/home.index`**
(zawiera tekst i odniesienia do 10 obrazów jpeg)

4. Serwer HTTP zamyka połączenie TCP.

5. Klient HTTP otrzymuje wiadomość zwrotną zawierającą plik html. Analizując plik HTML, znajduje 10 obiektów jpeg, do których istnieją odnośniki

6. Kroki 1-5 powtarzają się dopóki 10 obiektów jpeg nie zostanie pobranych

czas

Żądanie HTTP

- dwa typy wiadomości HTTP: *request, response*
- HTTP request:
  - ASCII (format do odczytania przez człowieka)

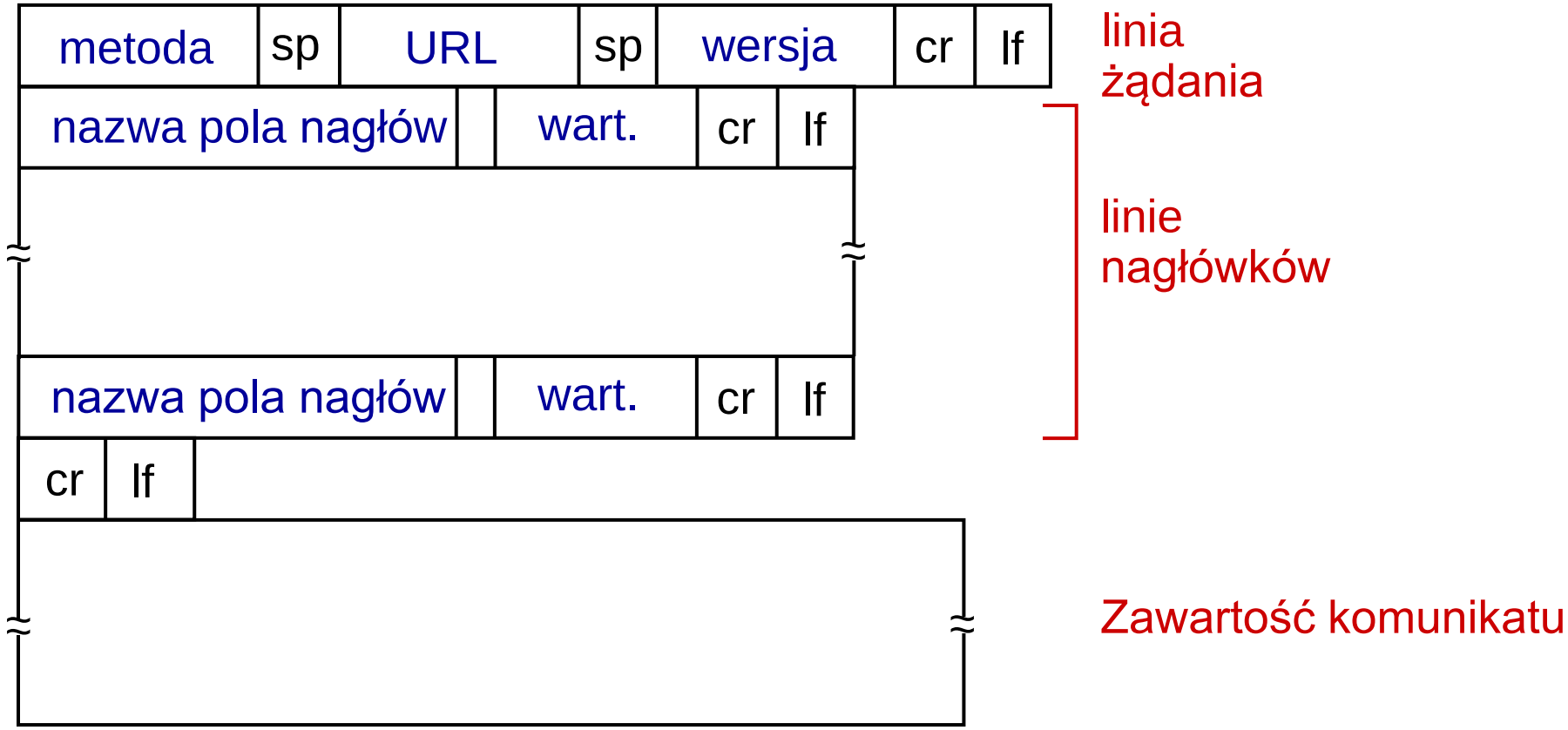
Znak powrotu karetki

Znak nowej linii

linia żądania line (metody GET, POST HEAD)

Koniec wierszy nagłówka

Żądanie HTTP: ogólny format



Metody HTTP

POST:

- związana z formularzami na stronie
- dane z formularza wysłane z klienta do serwera jako zawartość komunikatu

GET(do wysyłania danych na serwer):

- uwzględnij dane użytkownika w polu adresu URL komunikatu żądania HTTP GET (po znaku „?”):

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

HEAD :

- żąda nagłówków (**tylko**), które zostałyby zwrócone, gdyby określony adres URL został zażądany za pomocą metody HTTP GET

PUT:

- ładuje nowy plik (obiekt) na serwer
- całkowicie zastępuje plik istniejący pod określonym adresem URL treścią z pola komunikatu

Odpowiedz HTTP

linia statusu (status protokołu kod statusu komunikat)

```
HTTP/1.1 200 OK
```

Kody statusu odpowiedzi HTTP

- kod stanu pojawia się w pierwszym wierszu komunikatu odpowiedzi serwera do klienta, np..

200 OK
- żądanie powiodło się, żądany obiekt w dalszej części tej wiadomości

301 Moved Permanently
- żądany obiekt został przeniesiony, nowa lokalizacja określona w dalszej części tej wiadomości (w polu Lokalizacja)

400 Bad Request
- komunikat żądania niezrozumiany przez serwer

404 Not Found
- żądany dokument nie został znaleziony na tym serwerze

505 HTTP Version Not Supported

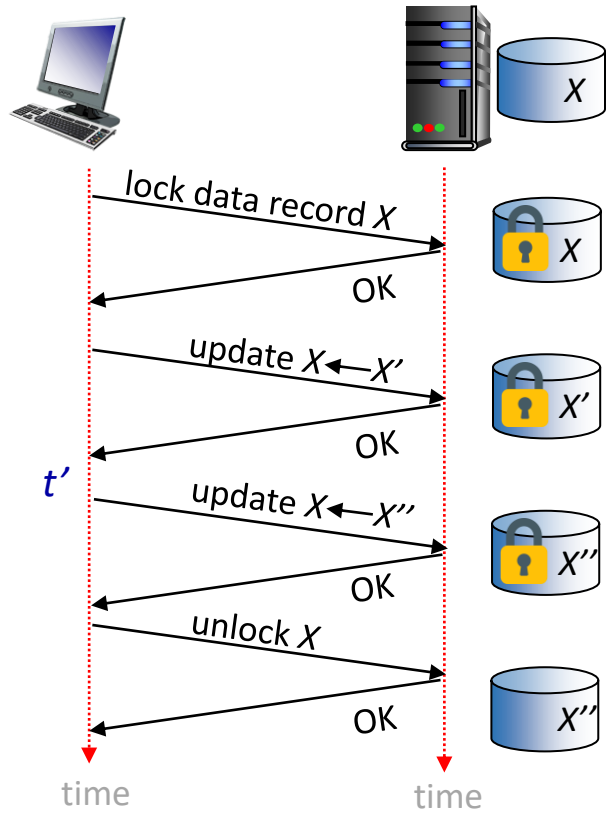
http request tool online

Cookies

Przypomnienie: HTTP jest *bezstanowy*

- brak koncepcji wieloetapowej wymiany komunikatów HTTP w celu sfinalizowania „transakcji” internetowej
  - klient/serwer nie musi śledzić „stanu” wieloetapowej wymiany
  - wszystkie żądania HTTP są od siebie niezależne
  - nie ma potrzeby „odzyskiwania” klienta/serwera z częściowo zrealizowanej, ale nigdy nie do końca zakończonej transakcji

protokół stanowy: klient dokonuje dwóch zmian w X lub żadnej



Q: co się stanie, jeśli połączenie sieciowe lub klient ulegnie awarii w t’?

Cookies

Witryny internetowe i przeglądarka używają plików *cookie*, aby zachować pewien stan między transakcjami

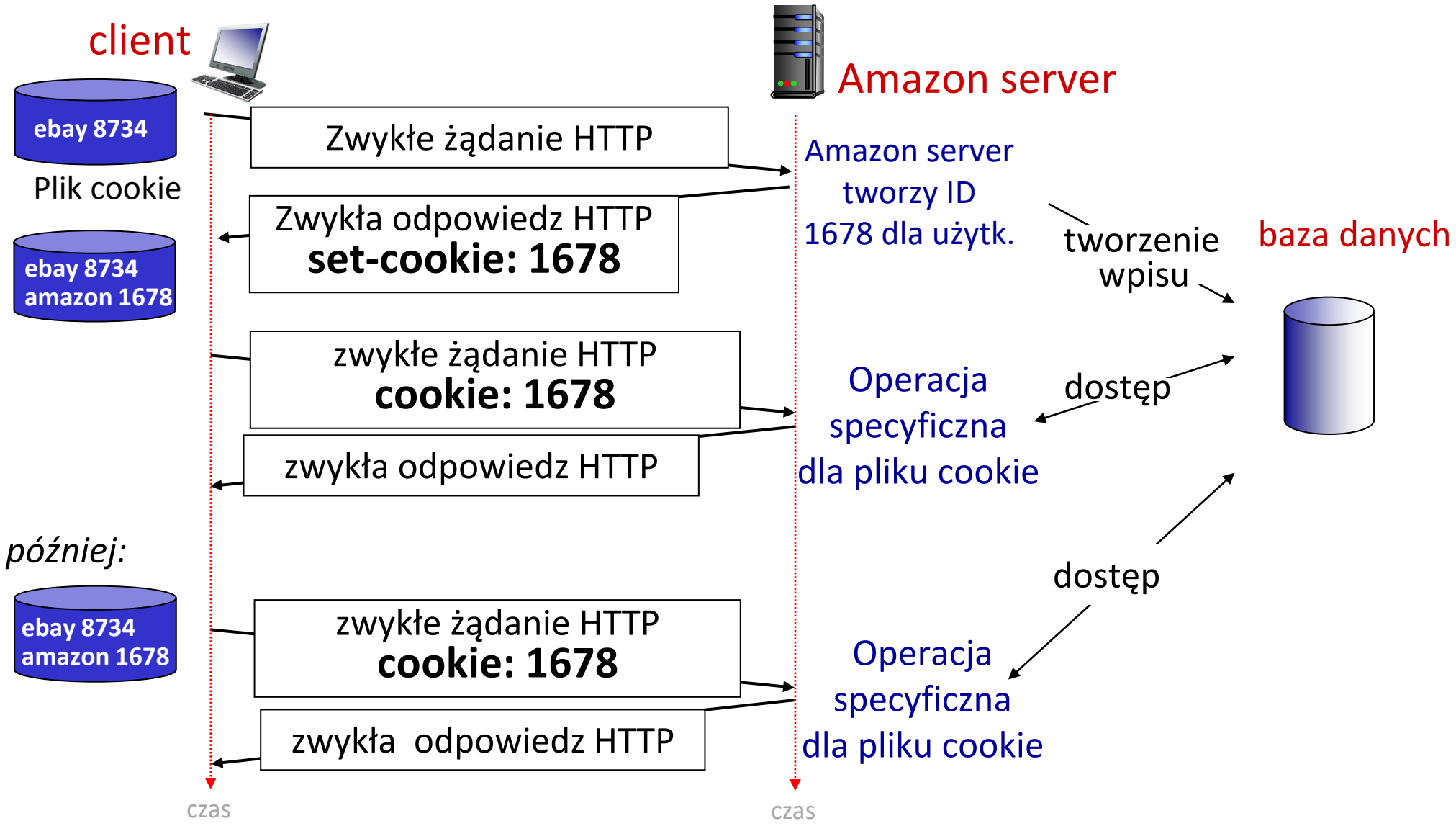
cztery komponenty:

1) nagłówek cookie w odpowiedzi HTTP
2) Nagłowek cookie w następnym żądaniu HTTP
3) plik cookie przechowywany w hoście, zarządzany przez przeglądarkę
4) baza po stronie servera

Przykład:

- Ania korzysta z przeglądarki na laptopie, po raz pierwszy odwiedza określoną witrynę e-commerce. Kiedy początkowe żądania HTTP docierają do witryny, witryna tworzy:
  - unikalny identyfikator (inaczej „plik cookie”)
  - wpis w bazie danych zaplecza dla identyfikatora
- Kolejne żądania HTTP wysyłane przez Anie do tej witryny będą zawierały wartość identyfikatora pliku cookie, umożliwiając witrynie „zidentyfikowanie” Ani.

Cookies



Cookies

Do czego mogą służyć pliki cookie?

- autoryzacja
- koszyk w e-sklepie
- rekomendacje
- Stan sesji użytkownika (np. web e-mail)

DNS: Domain Name System

ludzie: wiele identyfikatorów

- pesel, numer dowodu, nazwisko

Hosty w Internecie, routery:

- Adres IP – używany do adresacji datagramów
- "nazwa", np., prz.edu.pl – używane prze ludzi

Q: Jak mapować między adresem IP a nazwą i odwrotnie?

Domain Name System (DNS):

- *rozproszona baza danych* zaimplementowana w hierarchii wielu *serwerów nazw*
- *protokół warstwy aplikacji*
- serwery DNS komunikują się w celu rozpoznania nazw

DNS

Usługi DNS:

- translacja nazwa hosta - IP
- aliasing hosta
  - nazwa kanoniczna i alias nazwy
- Alias serwerów pocztowych
- rozkład obciążenia:
  - zreplikowane serwery WWW: wiele adresów IP odpowiada jednej nazwie

Q: Dlaczego nie scentralizować DNS?

- Pojedyńczy punkt awarii
- Wielkość ruchu
- odległa centralna baza danych
- Konserwacja

A: Nie jest skalowalny !

DNS

ogromna rozproszona baza danych:

- ~ biliony rekordów, każdy prosty

obsługuje wiele bilionów zapytań dziennie:

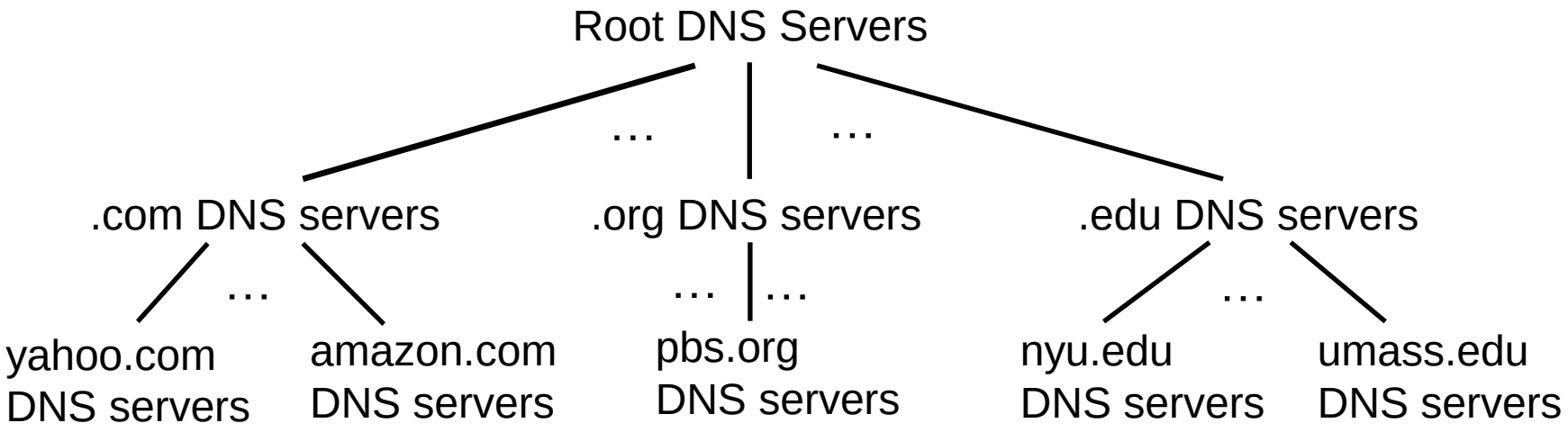
- *Znacznie więcej odczytów*
- *wydajność ma znaczenie*: prawie każda transakcja internetowa wchodzi w interakcje z DNS - liczą się msek!

organizacyjne, fizycznie zdecentralizowane:

- miliony różnych organizacji odpowiedzialnych za ich zapisy

„kuloodporne”: niezawodność, bezpieczeństwo

DNS: rozproszoną, hierarchiczna baza danych



Root

Top Level Domain

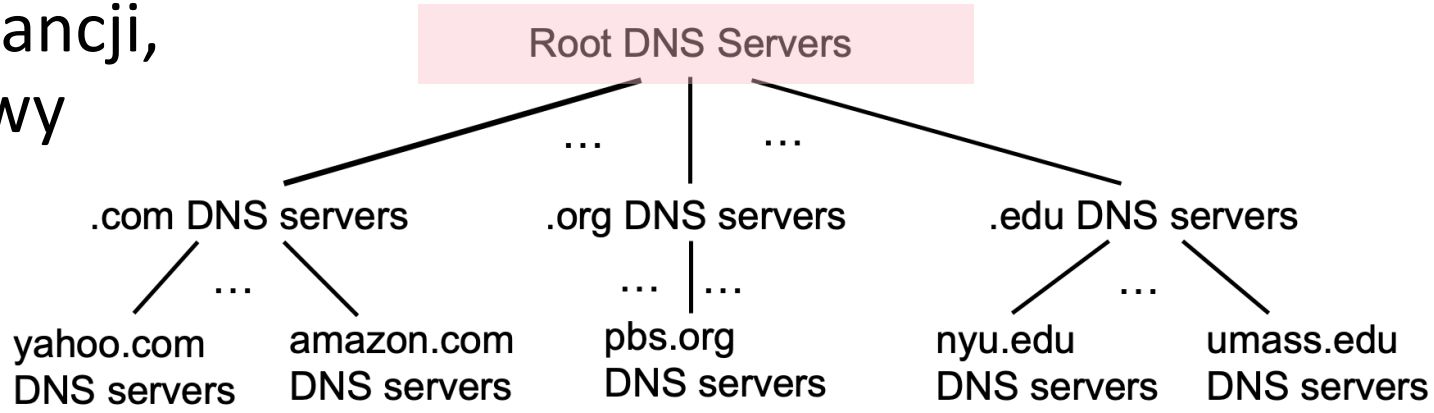
Autorytatywne

Klient chce adres IP dla www.amazon.com; pierwsze przybliżenie:

- klient wysyła zapytanie do serwera głównego, aby znaleźć serwer DNS .com
- klient wysyła zapytanie do serwera DNS .com, aby uzyskać serwer DNS Amazon.com
- klient wysyła zapytanie do serwera DNS amazon.com, aby uzyskać adres IP dla www.amazon.com

DNS: serwery root

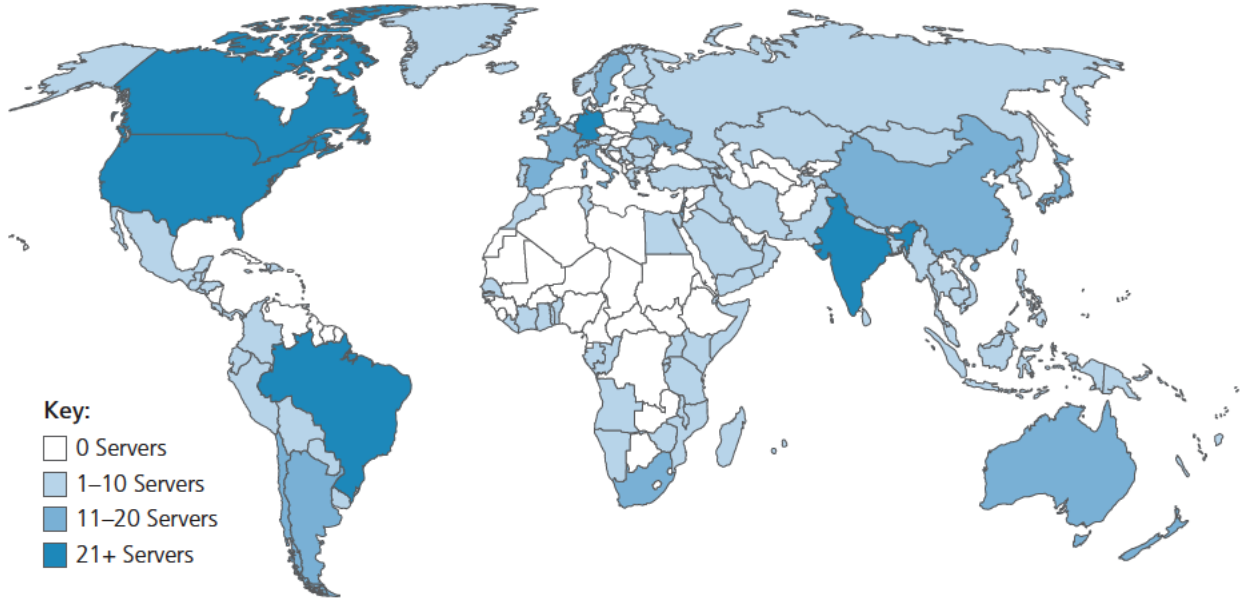
- Serwery DNS ostatniej instancji, które mogą rozwiązać nazwy
- https://root-servers.org



DNS: serwery root

- Serwery DNS ostatniej instancji, które mogą rozwiązać nazwy
- https://root-servers.org
- *niezwykle ważna* funkcja
  - Internet nie mógłby bez niego funkcjonować!
- ICANN (Internet Corporation for Assigned Names and Numbers) zarządza główną domeną DNS
- https://dns.pl

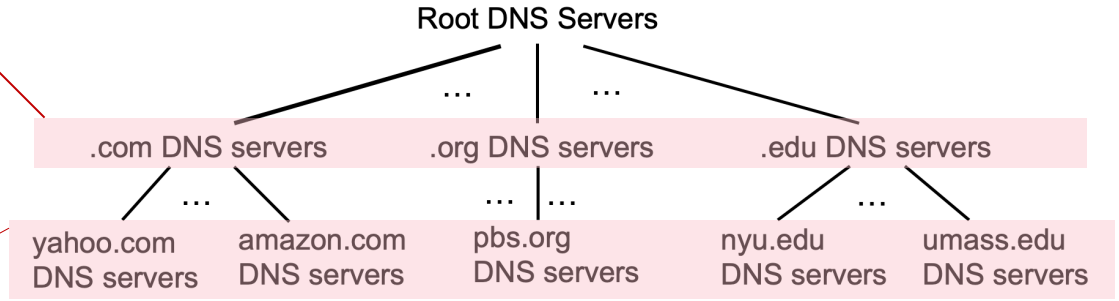
13 logicznych „serwerów” z nazwami głównymi na całym świecie, każdy „serwer” replikowany wielokrotnie



Top-Level Domain, serwery autorytatywne

Serwery Top-Level Domain (TLD):

- odpowiedzialny za domeny .com, .org, .net, .edu, .aero, .jobs, .museums oraz wszystkie domeny krajowe najwyższego poziomu np.: .cn, .uk, .fr, .ca, .jp
- Rozwiązania sieciowe: autorytatywny rejestr domeny .com, .net
- Edukacja: .edu



Autoratatywne serwery DNS

- własne serwery DNS organizacji, zapewniające wiarygodne odwzorowania nazw hostów na adresy IP dla hostów organizacji
- może być utrzymywana przez organizację lub usługodawcę

Lokalny serwer DNS

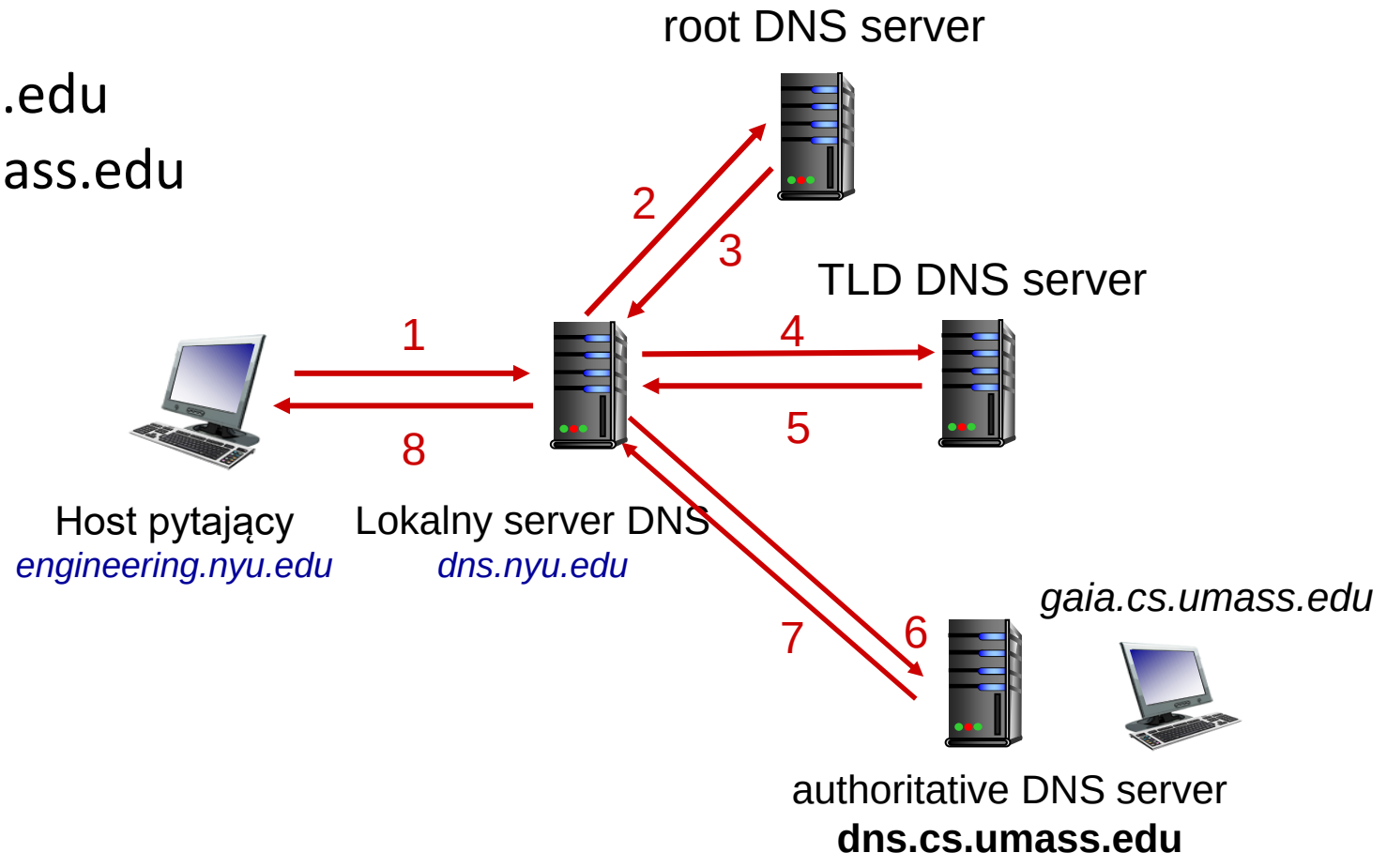
- gdy host wysyła zapytanie DNS, jest ono wysyłane do lokalnego serwera DNS. Lokalny serwer DNS zwraca odpowiedź, odpowiadając:
  - z lokalnej pamięci podręcznej ostatnich par tłumaczeń nazwa-adres (prawdopodobnie nieaktualne!)
  - przekazanie żądania do hierarchii DNS w celu rozwiązania
- każdy ISP ma lokalny serwer nazw DNS;
- lokalny serwer DNS nie należy ściśle do hierarchii

DNS: zapytanie iteracyjne

Przykład: host engineering.nyu.edu chce uzyskać adres gaia.cs.umass.edu

Zapytanie iteracyjne:

- serwer, z którym się skontaktowano, odpowiada nazwą serwera, z którym należy się skontaktować
- „Nie znam tej nazwy, ale zapytaj ten serwer”

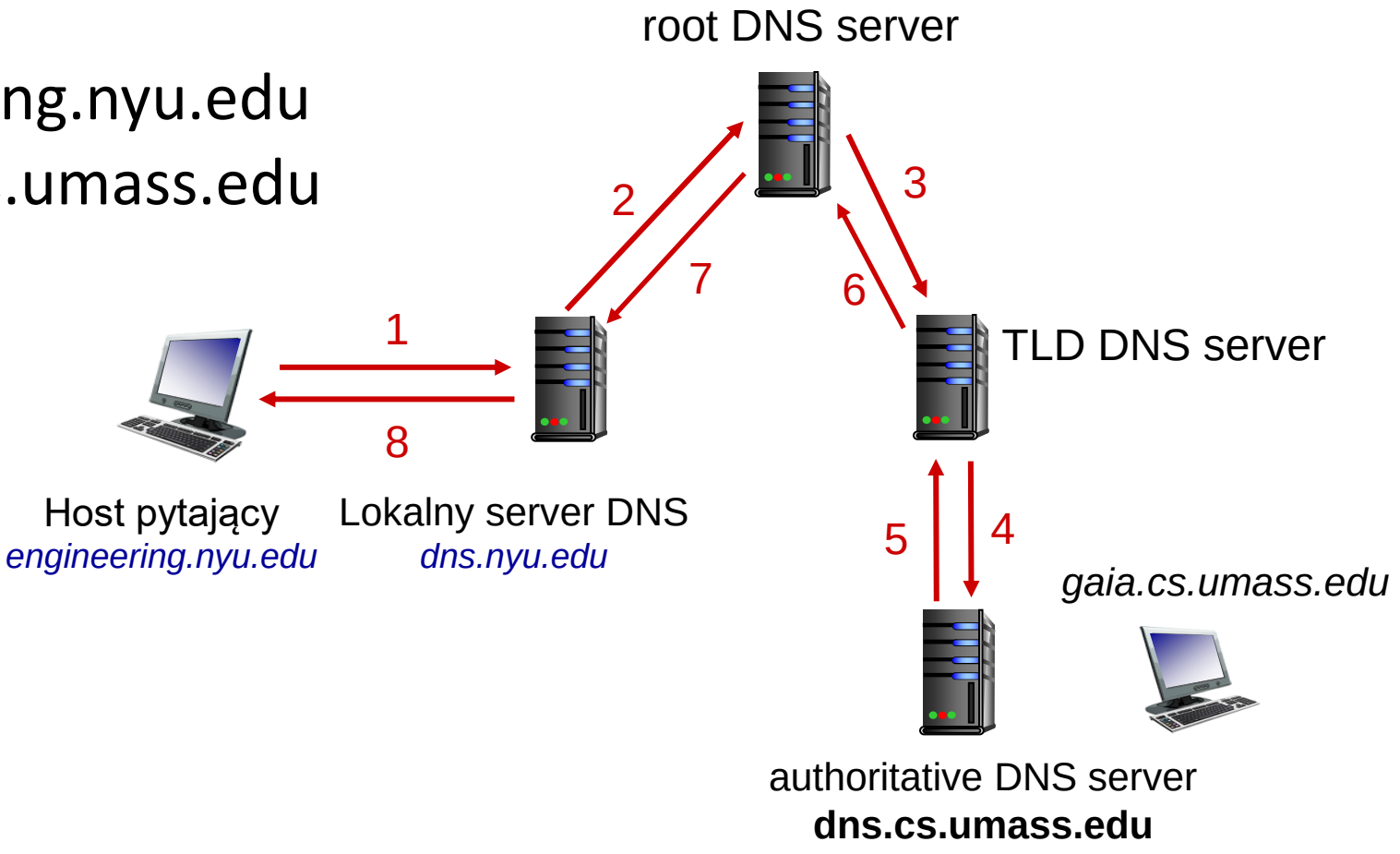


DNS: zapytanie rekurencyjne

Przykład: host engineering.nyu.edu chce uzyskać adres gaia.cs.umass.edu

Zapytanie rekurencyjne:

- nakłada ciężar rozpoznawania nazw na serwer nazw
- duże obciążenie na wyższych poziomach hierarchii?



Caching DNS

- gdy (dowolny) serwer nazw nauczy się mapowania, *buforuje* mapowanie i *natychmiast* zwraca buforowane mapowanie w odpowiedzi na zapytanie:
  - buforowanie poprawia czas odpowiedzi
  - wpisy w pamięci podręcznej po pewnym czasie (TTL) znikają
  - Serwery TLD są zwykle buforowane na lokalnych serwerach nazw
- wpisy w pamięci podręcznej mogą być *nieaktualne*
  - jeśli host zmieni adres IP, może nie być znany w całym Internecie, dopóki wszystkie TTL nie wygasną!!

Rekordy DNS

DNS: rozproszona baza danych przechowująca rekordy zasobów

Format rekordu: (`name, value, type, ttl`)

type=A

- `name` – nazwa hosta
- `value` – adres IP

type=NS

- `name` - domena (e.g., foo.com)
- `value` – nazwa hosta serwera autorytawnego dla domeny

type=CNAME

- `name` to alias do nazwy kanonicznej (prawdziwej)
- www.ibm.com to naprawę servereast.backup2.ibm.com
- `value` to nazwa kanoniczna

type=MX

- `value` is name of SMTP mail server associated with `name`

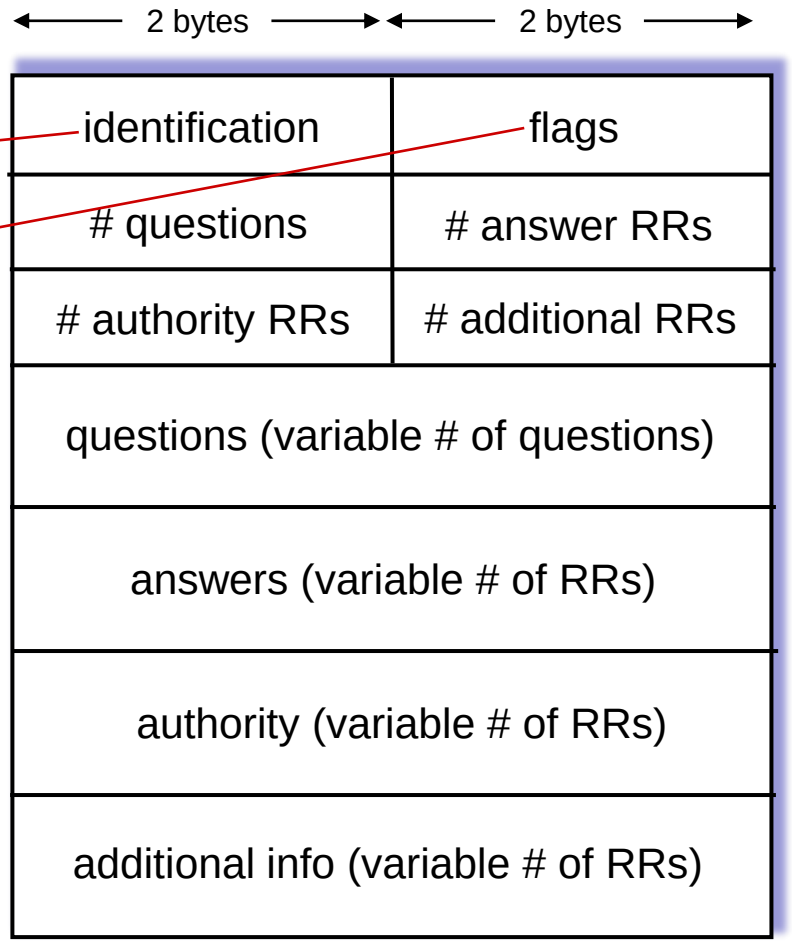
Wireshark

Protokół DNS

DNS *Zapytanie* i *odpowiedz* DNS mają taki sam *format:*

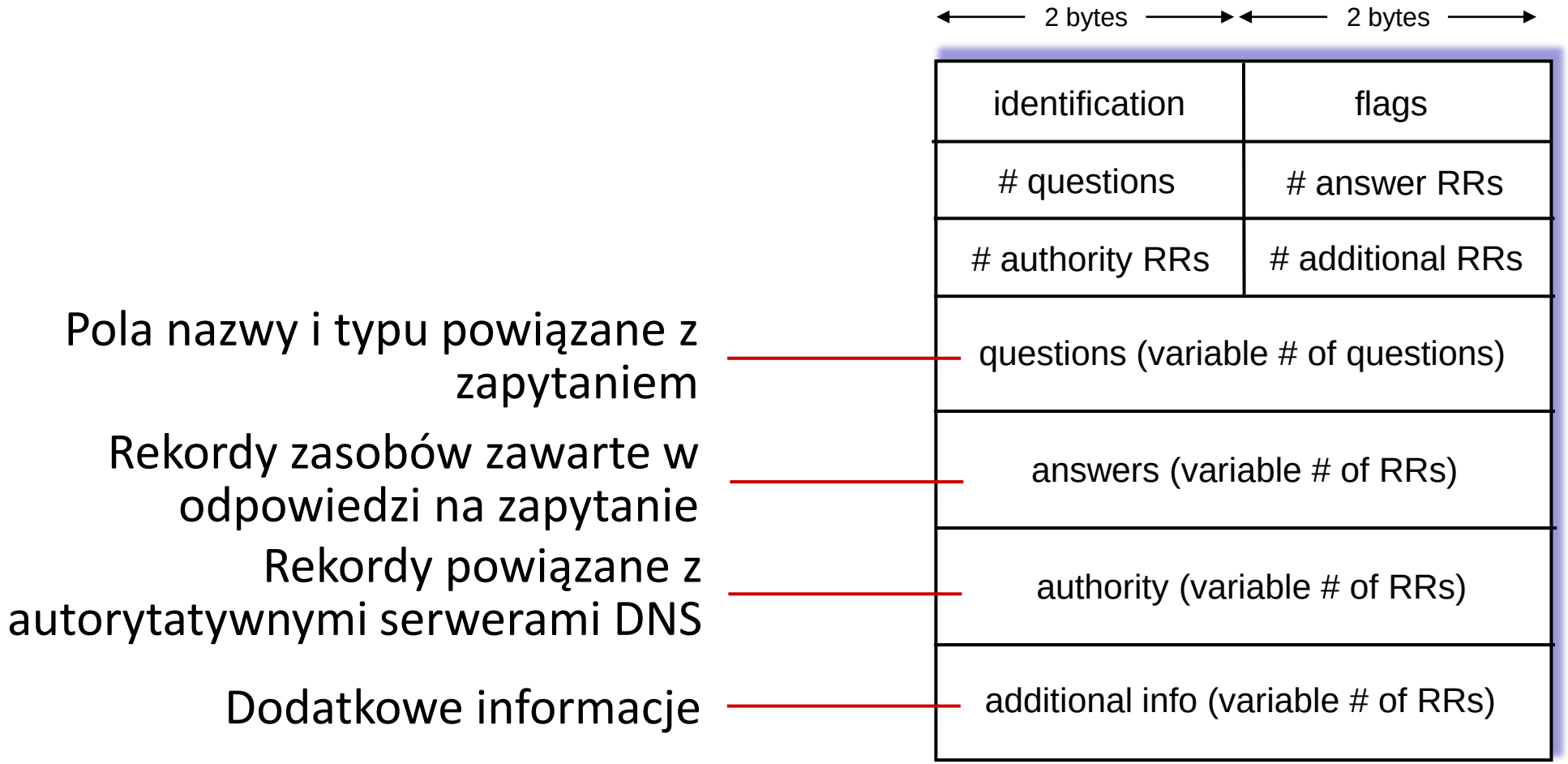
nagłówek:

- identification: 16 bitowy numer zapytania; odpowiedz ma taki sam numer jak zapytanie
- flagi:
  - zapytanie lub odpowiedz
  - rekurencja pożądana
  - rekurencja dostępna
  - odpowiedz jest autorytatywna



DNS protocol messages

DNS *Zapytanie* i *odpowiedz* DNS mają taki sam *format:*



DNS: Przykład

przykład: nowy startup “Network Utopia”

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

    `(networkutopia.com, dns1.networkutopia.com, NS)`

    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com